MASTER EN SCIENCES INFORMATIQUES

Analyse de contexte et évolution d'un outil expérimental de support aux évaluations de la maturité des processus de la maintenance S3mAssess®

Di Tomaso, Cédric

Award date:
2008

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'informatique.
Ecole de Technologie Supérieure (ETS), Montréal, Canada.
Année académique 2007-2008

Analyse de contexte et évolution d'un outil expérimental de support aux évaluations de la maturité des processus de la maintenance : $S^{3mAssess®}$

Auteur : Cédric DI TOMASO

<cedric.ditomaso@gmail.com>

Mémoire présenté en vue de l'obtention du grade de maître en informatique

---

Faculté d'Informatique - rue Grandgagnage, 21, B-5000 Namur (Belgium)

Résumé

La maintenance d'un logiciel est la phase du cycle de vie représentant bien souvent le coût le plus important d'un logiciel. C'est donc tout naturel que les entreprises veulent optimiser leur processus de maintenance et minimiser les risques liés à l'utilisation de mauvaises pratiques de travail. Toutefois, avant d'y arriver efficacement, il convient d'évaluer le niveau de maturité de toutes leurs activités de maintenance déjà en place et ce, de la manière la plus complète et fiable possible. A cette fin, il existe déjà le modèle CMMi qui permet l'évaluation de la maturité des processus du développement de logiciel. Cependant, ce modèle est très contraignant en termes de ressources nécessaires et s'adapte peu au contexte de la petite maintenance en entreprises. En effet, ce modèle est difficilement applicable à ces dernières car il s'avère trop conséquent, trop couteux et trop long à mettre en oeuvre. C'est pourquoi un modèle spécifique d'amélioration du processus dit "de petite maintenance du logiciel" a été mis au point. Ce modèle se nomme $S^{3m®}$.

C'est dans ce contexte d'adaptation à des ressources limitées qu'est née l'idée de concevoir un logiciel permettant de supporter des évaluations rapides mais néanmoins pertinentes. Ce logiciel se nomme $S^{3mAssess®}$ et s'appuie sur le modèle $S^{3m®}$. Une première version de cet outil expérimental de support aux évaluations de maturités fut déjà développée mais il s'est avéré au final qu'elle était non-fonctionnelle et non-aboutie.

Ce mémoire a pour but d'analyser le contexte de la maintenance du logiciel et plus particulièrement, comment supporter l'évaluation de la maturité de ses processus. Pour cela, il s'agira de procéder à la réingénierie de l'outil existant afin qu'il rencontre finalement ses exigences initiales. Une fois cet objectif atteint, il s'agira de rechercher comment améliorer le processus d'évaluation et surtout, comment fournir une analyse finale aussi complète que pertinente et qui permettra aux managers d'une organisation de prendre les meilleures décisions d'amélioration possibles. Le logiciel devra suivre les recommandations de la norme ISO/IEC 15504 et pourra être utilisé à des fins de détermination de la capacité des processus de maintenance et d'amélioration de ceux-ci.

Au final, le but sera de supporter l'évaluateur tout au long de son processus d'évaluation en automatisant sa tâche, en concentrant toutes les informations en un seul et même endroit et en lui fournissant une analyse aussi complète que pertinente sur base de ses données d'évaluation.

Mots clés :
Maintenance, logiciel, processus, amélioration, modèle, ressources, entreprise, outil, support, évaluation, réingénierie, capacité, analyse.

Abstract

The software maintenance stage is often the most costly in the life cycle of a software. Companies should have a growing interest in the improvement of their maintenance processes and in reducing the risks related to the use of inneficient practices in this area. However, before attaining that goal, it is necessary to assess the capability of all current maintenance activities. That assessment has to be as comprehensive and reliable as possible. To achieve that, companies use the CMMi model which allows a capability assessment of the software development processes. Using the CMMi is costly in terms of resources and not really suitable to the particular context of small maintenance. Indeed, the use of the CMMi is not adequate for small maintenance activities. That is why a specific improvement model aimed at "small software maintenance" activities and suitable to small groups has been developed. It is named the software maintenance maturity model ($S^{3m®}$).

The idea of designing a support tool allowing to carry out quick but relevant assessments comes from that context of adjusting to limited resources of small maintenance organisation. That tool is called $S^{3mAssess®}$ and integrates the concepts of the $S^{3m®}$ model. A first version of the experimental support tool was already developed but it was finally non-functional and incomplete.

The goal of this thesis is to analyse the context of software maintenance and more specifically, how to support the maturity assessment of its process. To achieve that, the existing assessment support tool will be reengineered and used in real situations in order to improve it. Then, the thesis will seek how to improve the assessment process and above all, how to supply results that are as comprehensive as relevant and that will allow managers of organisation to take the best improvement decisions. The assessment support tool will have to respect the recommendations of the ISO/IEC 15504 international standard and will be used for the determination of the software maintenance process capability and finally, for improvement.

As a result of this research, one of the goals will be to support the assessor during the assessment process in capturing the assessment results and in providing a comprehensive analysis based on the assessment data's.

Keywords :
Maintenance, software, process, improvement, model, resources, enterprise, tool, support, assessment, reengineering, capability, analysis.

Avant-propos

Je remercie mon maître de stage, Dr Alain April, qui a rendu possible l'accomplissement de ce stage de part ses travaux et sa disponibilité.

Je remercie également mon promoteur, Dr Naji Habra, pour son aide et sans qui la réalisation de ce mémoire eut été impossible.

Enfin, je tiens aussi à remercier tous les intervenants, professeurs, membres d'entreprises, etc. ayant participé à un moment ou à un autre dans l'élaboration de cette recherche.

Sincères remerciements à vous et à tous ceux m'ayant soutenu durant ce travail.

Table des matières

Liste des abréviations et sigles

API	Application Programming Interface
BD	Base de Données
CASE	Computer-Aided Software Engineering
CMMi	Capability Maturity Model Integration
COTS	Commercial-Off-The-Shelf
DDC	Demandes De Changement
DDL	Data Definition Language
ETS	Ecole de Technologie Supérieure située à Montréal, Canada
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
ISO/IEC	the International Organization for Standardization / the International Electrotechnical Commission
JDBC	Java DataBase Connectivity
KPA	Key Process Area ou *itinéraire* en français
OLA	Operation Level Agreement, un accord interne traitant de la livraison des services que doit offrir une organisation IT
RM	Requête de modification
ROI	Return On Investment
RP	Requête de problème
SD	Structure de Données
SEI	Software Engineering Institute
SGBD	Système de Gestion des Bases de Données
SI	Système d'Information
SLA	Service Level Agreement, un accord écrit entre un Fournisseur de Services et le(s) Client(s) décrivant les niveaux de service conclus pour un service
SPICE	Software Process Improvement Capability and dEtermination [ISO/IEC, 1998]
SQL	Structured Query Language
SVN	Subversion, un logiciel de gestion des versions
SWEBOK	SoftWare Engineering Body Of Knowledge
URL	Uniform Resource Locator
VPN	Virtual Private Network
WEB	World Wide Web, littéralement la "toile (d'araignée) mondiale"

Glossaire

- **Action d'amélioration de processus** : une action planifiée et exécutée pour améliorer tout ou partie d'un processus logiciel. Une action d'amélioration de processus peut contribuer à l'atteinte de plus d'un but de processus

- **Amélioration de processus** : action prise pour changer les processus d'une organisation de sorte qu'ils rencontrent les besoins business d'une organisation et qu'ils atteignent leurs buts business plus efficacement

- **Attribut de processus** : une caractéristique mesurable de capacité d'un processus, applicable à tout processus

- **Capacité améliorée** : une meilleure capacité que celle actuellement évaluée, justifiée par un programme d'amélioration de processus crédible

- **Capacité d'un processus** : l'aptitude d'un processus à atteindre un but requis [ISO/IEC, 1998]. Aussi défini comme étant "la gamme des résultats attendus qui peuvent être atteints en utilisant un processus spécifique." [Paulk et Curtis, 1993]

- **Catégorie de processus** : un ensemble de processus ayant le même espace général d'activité

- **Cohésion** : unité logique d'un objet, toutes ses parties contribuent à un objectif commun. Une forte cohésion est souhaitable pour l'adaptation au changement

- **Contexte d'un processus** : l'ensemble des facteurs, documentés dans les entrées d'une évaluation, qui influence le jugement, la compréhension et la comparabilité des cotations des attributs de processus

- **Cotation d'un attribut de processus** : un jugement du niveau d'accomplissement de la capacité définie d'un attribut de processus, pour le processus évalué

- **Cotation d'un niveau de capacité d'un processus** : une représentation du niveau de capacité atteint par un processus, dérivée à partir des cotations des attributs du processus évalué

- **Couplage** : interdépendances entre objets : relations définies sur plusieurs objets et caractérisées par les services offerts. Un faible couplage est souhaitable pour l'adaptation au changement

- **Détermination de la capacité d'un processus** : une évaluation et analyse systématique, par rapport à une capacité cible, des processus logiciels sélectionnés au sein d'une organisation, effectuées avec le but d'identifier les forces, faiblesses et risques associés au déploiement des processus en vue de rencontrer une exigence particulière

- **Dimension des processus** : l'ensemble des processus intégrant les aspects fonctionnels d'un modèle de référence des processus et des capacités de processus. Les processus sont groupés en catégories d'activités liées

- **Enregistrement d'évaluation** : une collection ordonnée et documentée d'informations pertinentes à une évaluation et qui ajoute à la compréhension et à la vérification des *profiles de processus* générés par l'évaluation

- **Evaluateur compétent** : une personne ayant démontré les aptitudes, les compétences et l'expérience nécessaires pour réaliser des évaluations de processus

- **Evaluateur provisoire** : une personne ayant les aptitudes et compétences pour réaliser des évaluations sous les conseils et la supervision d'un évaluateur compétent

- **Evaluation de processus** : une évaluation disciplinée des processus logiciels d'une organisation par rapport à un modèle de référence

- **Indicateur d'évaluation** : un attribut objectif ou une caractéristique d'une pratique ou un produit de travail permettant le jugement de la performance, ou capacité, d'un processus implémenté

- **Input, ou *entrée*, d'évaluation** : la collection d'informations requises avant qu'un processus d'évaluation puisse commencer

- **Instrument d'évaluation** : un outil ou un ensemble d'outils utilisés durant une évaluation afin d'assister l'évaluateur lorsqu'il évalue la performance ou la capacité des processus et lorsqu'il manipule des données d'évaluation et enregistre les résultats d'évaluation

- **Maturité d'un processus** : L'étendue à laquelle un processus spécifique est défini explicitement, géré, mesuré, contrôlé et efficace. La *maturité d'un processus* est évaluée afin d'en établir sa *capacité* (voir *Capacité d'un processus*)

- **Requête de Modification (RM)** : un terme générique utilisé pour identifier les changements proposés à un produit logiciel en maintenance. La RM peut être classée plus tard comme une correction ou une amélioration et catégorisée comme maintenance corrective, adaptative ou perfective. Les RMs sont aussi appelées *requêtes de changement*

- **Niveau de capacité d'un processus** : un point sur une échelle ordinale (de capacité d'un processus) qui représente la capacité en amélioration du processus en cours d'exécution ; chaque niveau étant construit à partir de la capacité du niveau inférieur

- **Output, ou *sortie*, d'évaluation** : tous les résultats tangibles venant d'une évaluation (voir *Enregistrement d'évaluation*

- **Pratique** : une activité d'ingénierie du logiciel ou de management qui contribue à la création de résultats (produits de travail, livrables) d'un processus ou qui améliore la capacité d'un processus

- **Preuve objective** : une information qualitative ou quantitative, des enregistrements ou des exposés de faits pertinents aux caractéristiques d'un élément ou d'un service ou de l'existence et implémentation d'un élément de processus et qui sont basés sur une observation, une mesure ou un test pouvant être vérifié

- **Processus** : un ensemble d'activités liées transformant des entrées en sorties. Le terme "activités" couvre l'utilisation de ressources

- **Processus défini** : la définition opérationnelle d'un ensemble d'activités en vue d'atteindre un but spécifique. Un processus défini peut-être caractérisé par des standards, procedures, formations, outils et méthodes

- **Processus logiciel** : le processus ou l'ensemble des processus utilisés par une organisation ou par un projet pour planifier, gérer, exécuter, monitorer, controler et améliorer ses activités logicielles

- **Produit de travail (*work product*)** : un objet, ou *livrable*, associé à l'exécution d'un processus. Un produit de travail peut-être utilisé, produit ou changé par un processus

- **Profile de processus** : l'ensemble des cotations des attributs de processus pour un processus évalué

- **Programme d'amélioration de processus** : toutes les stratégies, politiques, buts, responsabilités et activités concernées par l'atteinte des buts d'amélioration. Un programme d'amélioration de processus peut être plus long qu'un cycle complet d'amélioration de processus

- **Rapport de Problème (RP)** : un terme utilisé pour identifier et décrire les défaillances détectées dans un produit logiciel en production. Le personnel de maintenance arrêtera son travail actuel pour s'occuper de ces problèmes en première priorité

- **Scope, ou *étendue*, de l'évaluation** : une définition des limites de l'évaluation, fournie en tant que partie des inputs de l'évaluation, incluant les limites organisationnelles de l'évaluation, les processus à inclure et le contexte au sein duquel les processus opèrent (voir *Contexte du processus*

- **Unité organisationnelle** : la partie d'une organisation qui est sujette à une évaluation.
  Une unité organisationnelle peut dépoyer un ou plusieurs processus ayant un contexte de processus cohérent et opérant au sein d'un ensemble cohérent de buts de business.
  Une unité organisationnelle est typiquement une partie d'une organisation plus large. Si l'organisation est petite, l'unité organisationnelle peut-être l'organisation toute entière. Cela peut-être par exemple :
  - un projet spécifique ou un ensemble de projets liés ;
  - une unité d'une organisation concentrée sur une/des phase(s) spécifique(s) du cycle de vie d'un logiciel comme l'acquisition, le développement, la maintenance ou le support ;
  - une partie d'organisation responsable de tous les aspects d'un produit particulier ou d'un ensemble de produits.

Liste des tableaux

Table des figures

Introduction

Méthodologie de recherche définie via le Cadre de Basili

Cette partie introduit le mémoire en présentant l'objet du travail, les motifs de l'étude, la méthode de travail, les sources utilisées et les points importants de l'étude contextualisés dans le cheminement de travail. Pour cela, une méthode de recherche ayant fait ses preuves sera suivie, c'est à dire celle définie par le cadre de Basili (v. [Basili *et al.*, 1986] et une application dans [Bourque et Côté, 1991]). Cette méthode structure le travail de recherche et propose un schéma présentant l'avancement des travaux et des résultats.

Le projet de réaliser un logiciel d'évaluation de la maturité du processus de maintenance sera donc organisé selon trois parties. Une première partie visera à brosser un état des lieux sur la littérature traitant le sujet de ce projet. Une deuxième partie traitera l'élaboration de la construction du projet. Enfin, la troisième partie expérimentera le projet avec des entreprises réelles, ce qui permettra au final de recenser les faiblesses, d'analyser comment elles ont été gérées et d'en tirer les possibilités de travaux futurs.

Définition

Il s'agit ici de définir le projet, ou plus précisément, de définir sur quelles bases théoriques il repose. La première étape dans la démarche de recherche vise ainsi à analyser l'existant et à obtenir une base théorique solide avant de commencer tout nouveau développement. De même, les motivations du projet, ses sujets de recherche, les parties prenantes et l'analyse de son domaine d'application sont définis dès cette première étape.

Définition				
Motivation	**Sujet**	**But**	**Utilisateurs de la recherche**	**Domaine**
Analyse de contexte et maintenance d'un outil expérimental de support aux évaluations de la maturité du processus de la maintenance ($S^{3mAssess®}$)	– Analyser le contexte de la maintenance du logiciel – Comprendre la norme ISO 15504 – Comprendre le modèle $S^{3m®}$ – Définir la taxonomie de la réingénierie – Procéder à la réingénierie de l'outil d'évaluation $S^{3mAssess®}$ – Intégrer les Demandes De Changement (DDC) venant de divers intervenants clés (voir Annexes) – Intégrer les DDC nécessaires : améliorations personnelles du logiciel $S^{3mAssess®}$ – Intégrer les DDC provenant de la méthode d'évaluation $S^{3m}Assessment$ – Recherche sur l'analyse d'une évaluation de maturité $S^{3m®}$, intégration des résultats de recherche dans $S^{3mAssess®}$ et expérimentation avec des entreprises	– Valider le logiciel de support $S^{3mAssess®}$ – Proposer des résultats d'évaluation pertinents pour une entreprise : gestion d'historiques, analyse de deltas, sortie des résultats sur feuilles de calculs, calculs de moyennes, etc. – Modifier l'outil afin de permettre une évaluation de maturité $S^{3m®}$ selon deux méthodes : selon une méthode "standard" d'application du modèle $S^{3m®}$ et selon la méthode $S^{3m}Assessment$	– Les experts du domaine de la maintenance du logiciel – Les évaluateurs – Les managers d'une unité évaluée – Les consultants – Les managers d'entreprise désirant obtenir le niveau de maturité de divers partenaires potentiels – Les chercheurs en maintenance	– Assistance aux entreprises dans la compréhension et dans l'application du modèle $S^{3m®}$ via le logiciel de support – Développement de l'outil $S^{3mAssess®}$ pour améliorer le processus de maintenance des équipes de maintenance

Planification

La deuxième étape de la méthodologie de recherche consiste à planifier les éléments théoriques nécessaires pour ce mémoire et à en dresser une revue bibliographique. Cette revue matérialisera la première partie du mémoire.

Planification		
Etapes du projet	**Entrants du projet**	**Livrables de l'étape**

– Supporter une entreprise dans sa démarche d'évaluation :
  – Comprendre le modèle $S^{3m®}$ et aider les entreprises à l'appliquer via le logiciel $S^{3mAssess®}$
  – Comprendre la norme ISO/IEC 15504 et l'appliquer dans le cas de $S^{3mAssess®}$
  – Analyser les documents et interviews réalisés par les entreprises
  – Procéder à la réingénierie de $S^{3mAssess®}$ en le confrontant directement aux besoins des entreprises
	– A. April and A. Abran. *Améliorer la maintenance du logiciel*. Loze-Dion, 2006.
	– La norme ISO/IEC 15504, first edition 1998
	– La documentation du logiciel $S^{3mAssess®}$
	– D.-A. Paquette, A. April, and A. Abran. Assessment results using the software maintenance maturity model ($S^{3m®}$). Ecole de Technologie Supérieure.
	– T. M. Pigoski. *Practical software maintenance : best practices for managing your software investment*. Wiley computer publishing, 1996.
	– *... voir la bibliographie de ce mémoire*
		– Partie I du mémoire :
		  – Introduction à la maintenance du logiciel
		  – Synthèse des parties une à cinq de [ISO/IEC, 1998] nécessaires pour l'évaluation de processus
		  – Synthèse du modèle $S^{3m®}$
		  – Ingénierie et Réingénierie : notions de base

Opérations

La troisième étape de la démarche vise en premier lieu à analyser le contexte de $S^{3mAssess®}$, à identifier les parties prenantes du projet et à définir comment celles-ci vont contribuer à l'évolution de l'outil. Il s'agit ensuite de donner accès au matériel adéquat aux parties prenantes afin qu'un échange puisse avoir lieu. De plus, l'établissement d'un processus de maintenance devra être réalisé dès le début pour mener à bien le projet de maintenance de l'outil. Enfin, la réingénierie de l'application constituera une phase importante de cette troisième étape.
Finalement, cette étape s'achèvera par la création de la deuxième partie de ce mémoire qui présentera tout d'abord l'outil, son contexte et son domaine, et qui décrira ensuite toute sa phase de maintenance.

Opérations		
Préparation	**Exécution**	**Analyse**
– Identifier les participants qui vont contribuer à la recherche – Analyser comment appliquer efficacement le modèle $S^{3m®}$ à travers le logiciel de support $S^{3mAssess®}$ – Préparation des conférences téléphoniques : planification, démonstrations, réponses aux questions des clients – Mise à disposition aux entreprises de l'outil d'évaluation, support et analyse des besoins – Echanges d'informations avec les entreprises => *spécification d'ordre un* – Analyse de la méthode d'évaluation S^{3m}*Assessment* (Scampi allégé) => *spécification d'ordre deux* – Recherches sur l'amélioration de l'analyse finale rendue par le logiciel $S^{3mAssess®}$ => *spécification d'ordre trois* – Requêtes de maintenance reçues de la part du maître de stage et des clients	– Donner accès aux clients aux documents du modèle $S^{3m®}$ ainsi qu'à l'outil de support accompagné de son guide d'utilisation – Etablir un processus de maintenance pour mener à bien ce projet – Réingénierie de $S^{3mAssess®}$ dans un but de pertinence des résultats et d'applicabilité au contexte de l'entreprise – Maintenance de $S^{3mAssess®}$ selon le processus de maintenance préalablement établi	– Partie II du mémoire : – Présentation de $S^{3mAssess®}$, de son domaine d'application et de son contexte technique – Maintenance de $S^{3mAssess®}$: description, critique des choix de conception, justification des solutions apportées suite aux résultats de recherche et gestion des problèmes – Evolutions apportées à $S^{3mAssess®}$ suite aux résultats de recherche, justifications et suivi des recommandations issues de normes internationales

Interprétation

Cette dernière étape de la démarche méthodologique de recherche a pour but d'expérimenter ce qui a été développé dans l'étape précédente. Cette expérimentation se fera auprès des intervenants présentés en annexe à ce mémoire.
Il en ressortira des faiblesses et des améliorations qui seront soit déjà prises en compte dans le processus de maintenance ou qui seront adressées à des travaux futurs selon leur moment d'occurrence.

Interprétation		
Contexte d'interprétation	**Extrapolation**	**Travaux futurs**
– Présentation des révisions de $S^{3mAssess®}$ aux diverses parties prenantes pour validations – Analyse des faiblesses et des possibilités d'amélioration (aussi bien de l'outil que du modèle) suite aux confrontations au contexte réel des entreprises	– Validation des documents explicatifs et de support fournis aux clients – Démonstration des évolutions de $S^{3mAssess®}$ – Validation des évolutions de l'outil et plus particulièrement de la pertinence des résultats d'analyse qui en sont issus – Présentation de l'outil et de son utilité à de nouvelles parties prenantes – Etude de cas : réalisation d'une évaluation complète dans un cas réel – Auto-évaluation selon le modèle $S^{3m®}$ du processus de maintenance définis pour ce projet – Evaluation d'applicabilité du modèle $S^{3m®}$ à travers l'outil – Approche "multi-feedback" suite aux révisions de l'outil	– Résultats de l'étude de cas – Conclusions sur le processus de maintenance appliqué à l'outil suite à l'auto-évaluation – Améliorations possibles de $S^{3mAssess®}$ et du modèle $S^{3m®}$

Première partie

Revue de la littérature

Cette partie du mémoire commence par introduire les concepts de la maintenance du logiciel, puis synthétise la norme ISO/IEC 15504 qui permet d'évaluer la maturité des processus du logiciel. Ensuite, le modèle $S^{3m®}$ (pour *Software Maintenance Maturity Model*), spécifiquement conçu pour évaluer la maturité des processus de la maintenance du logiciel, sera décrit. Finalement, quelques notions de base de réingénierie, nécessaires dans le contexte de ce mémoire, seront définies.
Les aspects d'évaluation seront le centre d'intérêt de cette première partie du mémoire, ce qui permettra par la suite de s'appuyer sur des concepts qui permettront aux entreprises d'évaluer la maturité de leurs processus de maintenance grâce au modèle $S^{3m®}$ et grâce à son outil de support : $S^{3mAssess®}$.

Chapitre 1

Qu'est-ce que la maintenance du logiciel ?

Ce chapitre a pour but d'exposer au lecteur le contexte de la maintenance du logiciel. A cette fin, sa nécessité, ses différentes catégories et ses objectifs seront tout d'abord explicités. Cela permettra de fixer une bonne idée générale de ce qu'elle représente et des problématiques qui peuvent en surgir. Ensuite, une section donnera la définition de la maintenance du logiciel telle qu'elle est reprise actuellement selon un consensus international. En effet, il faut savoir que sa définition à évoluer d'année en année selon les expériences acquises dans ce domaine. Ainsi, le critère du/des moment(s) où elle intervient prendra une part très importante dans cette définition consensuelle.
Finalement, une dernière section présentera la notion de maintenabilité d'un logiciel qui permet principalement d'évaluer et d'améliorer la maintenance d'un logiciel.

Chronologiquement, la maintenance débute quand quelqu'un délivre un *Rapport de Problème* ou une *Requête de modification* pour un "système en opération" (voir la section 3.3). Toutefois, selon [Pigoski, 1996] ce n'est pas la meilleure approche. Il propose plutôt que son démarrage coïncide avec la prise de décision de développer un nouveau système. Ainsi, les organisations de maintenance auront plus de chances de réussir à obtenir un système maintenable si elles sont impliquées tout au long de son développement. Selon lui, la participation de la maintenance et la budgétisation devraient démarrer dès l'accord pris qu'un nouveau système sera développé. Cette participation devrait commencer bien avant la signature d'un quelconque contrat de développement et avant le moindre effort de développement.

1.1 Pourquoi la maintenance est-elle nécessaire ?

La question de savoir pourquoi la maintenance est nécessaire peut être posée. Si les développeurs logiciels ont bien réalisé leur travail, est-elle encore nécessaire ? Si les systèmes sont construits en utilisant des outils *computer-aided software engineering* (CASE) ou s'ils sont développés uniquement en utilisant des produits *commercial-off-the-shelf* (COTS), pourquoi la maintenance du logiciel est-elle nécessaire ? Toutes ces questions sont importantes pour expliquer la nécessité de la maintenance. La suite présentera également en quoi la maintenance du logiciel peut avoir un lien avec des besoins d'évolution, avec notamment l'existence d'un type de maintenance bien particulier : la maintenance perfective ou "évolutive". En effet, l'une des raisons les plus évoquées de nos jours de maintenir un logiciel est le fait qu'un logiciel est appelé à évoluer et n'est plus que rarement construit une fois pour toutes.

Pendant la phase d'*Opération et Maintenance* de tout cycle de vie du logiciel (voir la description des "modèles" du Cycle de vie dans le cours [Habra, 2006]) choisis par une organisation, les utilisateurs commencent par utiliser le logiciel (voir figure 1.1). Ceux-ci trouvent alors des mauvais aspects du système et d'autres qu'ils voudraient voir ajoutés au nouveau système car l'environnement d'affaire évolue constamment. Ils fournissent un feedback aux managers opérationnels qui a leur tour transmettent les rapports de problèmes et requêtes de modifications au mainteneur. Ce dernier réalise alors les corrections et les améliorations approuvées et installe les changements. Dès l'utilisation de la nouvelle version, les utilisa-

teurs commencent encore une fois à accumuler des demandes. De cette manière, ils perpétuent le cycle de maintenance et ils prolongent la vie du produit. La phase de maintenance s'avère souvent être la plus longue du cycle de vie d'un logiciel. A titre d'exemple, en utilisant le cycle de vie du modèle en cascades, cette phase peut durer de trois à cinq ans. Pour la plupart des systèmes opérationnels d'entreprise, la phase de maintenance dépasse de loin la phase de développement en termes de temps et de coût.

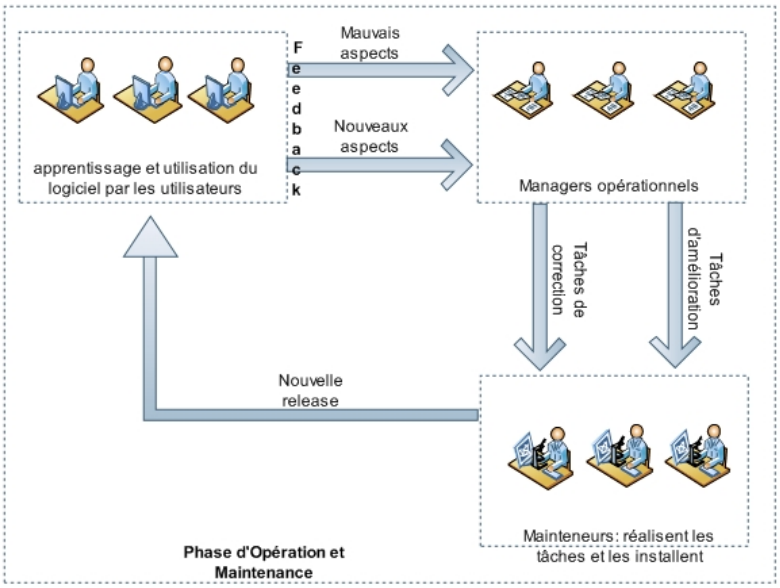


FIG. 1.1 – Le cycle général de la maintenance

La longue durée du processus d'Opérations et de Maintenance peut-être expliqué par les deux lois de Lehman [Lehman, 1980]. Sa première loi, la *Law of Continuing Change* qui déclare qu'un système a besoin de changer dans le but d'être utile. La seconde loi, la *Law of Increasing Complexity* exprime le fait que la structure d'un système se détériore au fur et à mesure qu'il évolue. A terme, il devient moins coûteux de réécrire tout le système car il est devenu très difficile de le faire évoluer.
Ces lois permettent de comprendre ce qui se passe pour le logiciel quand il est utilisé mais quels sont les problèmes que la maintenance tente de résoudre ? Selon Martin et McClure, la maintenance du logiciel doit être réalisée dans le but de :

- Corriger des erreurs ;
- Corriger des défauts de conception ;
- Interagir avec d'autres systèmes ;
- Réaliser des améliorations ;
- Réaliser les changements nécessaires pour le système ;
- Réaliser les changements dans les fichiers ou bases de données ;
- Améliorer la conception ;
- Adapter les programmes de sorte que les différents hardwares, software, caractéristiques de système et les services de télécommunication peuvent être utilisés.

D'autres raisons peuvent être de prévenir les problèmes ou d'améliorer la performance. Les logiciels évoluent et c'est pourquoi la maintenance est nécessaire.

1.2 Les catégories de la maintenance

E.B. Swanson of UCLA (voir citeSwa76) a créé trois catégories différentes de maintenance :

- **Corrective :** les changements nécessités suite à des défaillances dans un logiciel. Les changements correctifs réparent les "défaillances", c'est à dire quand le système ne s'exécute pas comme il a été conçu ou annoncé.
- **Adaptative :** ce sont les efforts nécessités suite à des changements dans l'environnement du système logiciel.
- **Perfective :** tous les changements, insertions, retraits, modifications, extensions et améliorations faites à un système pour rencontrer les besoins d'évolution et d'extension de l'utilisateur.

En addition à ces trois catégories, l'IEEE en définit une quatrième appelée **maintenance préventive** dont le but est de prévenir les problèmes avant qu'ils n'arrivent. Pour les systèmes critiques en fiabilité tels que les engins spatiaux ou les systèmes aéronautiques, la prévention des problèmes est très importante.

Faire la distinction entre la correction et l'amélioration des logiciels est nécessaire. La correction est nommée maintenance corrective tandis que l'amélioration se nomme maintenance adaptative et perfective (voir figure 1.2).
Les catégories de la maintenance du logiciel officielles et reconnues par l'industrie ont été normalisées en

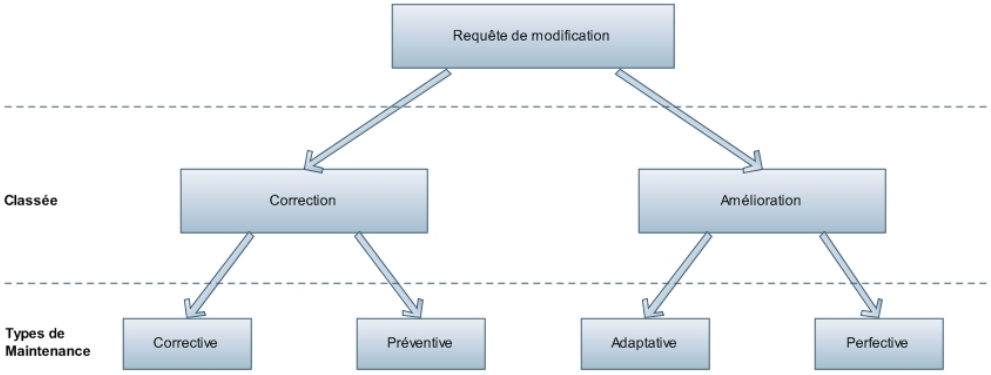


FIG. 1.2 – Les catégories de la maintenance du logiciel (*adapté de [ISO/IEC, 2005]*)

1991 dans la norme ISO/IEC 14764. Les catégories indiquent clairement que la maintenance ne se limite pas à de la correction de défaillances.
L'amélioration de systèmes existants peut s'avérer dans certains cas encore plus coûteuse que le développement d'un nouveau système, si toutefois sa base n'est pas bien structurée. En pratique, une entreprise peut être forcée de changer un système quand il n'est plus possible de le maintenir, c'est à dire quand les changements sont trop lents pour la vitesse à laquelle elle les désire ou quand la technologie a tellement changé qu'il est maintenant techniquement désuet. Par exemple, dans le cas de Cobol, il est maintenant devenu très difficile de trouver des ressources.
C'est pourquoi, confondre l'amélioration et la correction des défaillances peut mener à des erreurs dans l'estimation du temps et dans le budget pour implémenter les changements. Cela peut aussi ternir la réputation des développeurs en faisant croire aux gens qu'ils ont effectués un mauvais travail. Les mainteneurs de logiciels doivent avoir de bonnes définitions pour les corrections et les améliorations et un processus forçant une classification propre. Enfin, il peut être précisé qu'en pratique les clients préfèreront les améliorations aux corrections. Ainsi, il est préférable que les développeurs créent un logiciel cohérent et fiable et qu'ensuite les mainteneurs exercent des tâches d'améliorations adaptées aux nouvelles demandes des clients.

1.3 Les objectifs de la maintenance

ISO/IEC 14764 décrit l'objectif du processus de la maintenance du logiciel comme l'action de gérer la modification, la migration et la retraite de composants comme un logiciel, des parties d'application, des manuels d'opération, etc. Il ne s'agit pas ici d'une définition mais d'une idée de ce que poursuit un processus de maintenance. Une définition de la maintenance du logiciel sera donnée dans la section suivante.
L'origine des requêtes peut être causée par un problème découvert ou par la nécessité d'amélioration ou d'adaptation du système. Il s'agit ensuite de réaliser cet objectif tout en préservant l'intégrité des opérations organisationnelles. Si ce processus est bien implémenté, les résultats suivants sont obtenus [ISO/IEC, 1998] :

- une stratégie de maintenance sera développée pour gérer la modification, la migration et la retraite de composants système selon la stratégie de sortie des produits ;
- l'impact de l'organisation, des opérations et des interfaces sur le système existant en opération sera défini ;
- les spécifications, documents de conception et les stratégies de test seront mis à jour ;
- les composants systèmes en modification seront développés avec des tests associés démontrant que les exigences système ne seront pas compromises ;
- les mises à niveau de système et de logiciel seront migrées dans l'environnement du client ;
- suite à une requête, les logiciels et systèmes seront retirés de l'utilisation d'une manière contrôlée minimisant les perturbations des clients.

1.4 Les activités de la maintenance s'étendent sur plusieurs phases

En ne commençant le processus de maintenance qu'après avoir reçu le produit à maintenir, l'organisation de maintenance est mal préparée pour la maintenance du produit et l'ensemble des mainteneurs, clients et utilisateurs en pâtissent grandement. Suite à ce genre de problèmes, [Pigoski, 1996] propose cette définition de la maintenance :

> *La maintenance est la totalité des activités qui sont requises afin de procurer un support, au meilleur coût possible, à un système logiciel. Certaines activités débutent avant la livraison du logiciel et d'autres activités ont lieu après sa livraison finale. Les activités avant la livraison finale incluent la planification pour les opérations après livraison, la supportabilité et la détermination logistique. Les activités après livraison incluent la modification du logiciel, l'entrainement et l'exécution d'un help desk.*

Cette définition veut corriger l'ancienne vue de la maintenance qui la considérait comme une activité d'après livraison. Comme résultat, les actions nécessaires d'avant livraison, telles que la planification de la maintenance, étaient souvent négligées voir complètement évitées. D'ailleurs, encore aujourd'hui, beaucoup de méthodologies de développement ne représentent même pas l'étape de la maintenance du logiciel. Il a donc fallu apporter des modifications aux définitions traditionnelles étant donné que la maintenance du logiciel est maintenant une activité majeure consommant des ressources considérables.
Cette nouvelle définition reconnait que l'implication avant livraison fait partie du processus de la maintenance. Elle souligne aussi certains points, comme la formation et le help desk, étant des parties vitales après livraison. Une implication tôt dans le développement permet d'éviter des situations de surcharge de travail dans un temps réduit pour les équipes de maintenance. Ils peuvent ainsi commencer leurs tâches au plus tôt, comme par exemple la prévention des problèmes, et planifier leur processus à l'avance. Un autre avantage est la réduction des coûts.

En 2005, la norme internationale ISO/IEC 14764 ([ISO/IEC, 2005]) définissait la maintenance comme ceci :

> *La totalité des activités requises pour fournir un support, au meilleur coût possible, à un système logiciel. Les activités sont exécutées lors de l'étape d'avant livraison ainsi que lors de l'étape d'après livraison.*
>
> *NOTE : Les activités avant la livraison finale incluent la planification pour les opérations d'après livraison, la supportabilité et la détermination logistique. Les activités après livraison incluent la modification du logiciel, l'entrainement et l'exécution d'un help desk.*

La définition donnée par Pigoski en 1996 est donc restée à jour puisqu'elle était encore reprise en 2005 dans la norme ISO [ISO/IEC, 2005]. Il est à noter que M. Pigoski a piloté le développement de cette norme ISO. Ajoutons enfin que plusieurs des concepts utilisés dans le modèle de la maturité de la maintenance du logiciel, S^{3m}® (*Software Maintenance Maturity Model*), qui sera présenté au chapitre 3, ont été incorporés à la version 2005 du guide *SWEBOK* qui est un corpus de connaissances en génie logiciel (*Software Engineering Body of Knowledge*).

1.5 Notion de maintenabilité

Lorsque les activités de maintenance sont effectuées, une caractéristique essentielle devant être considérée pour tout logiciel est la *maintenabilité*. La maintenabilité est un but clé guidant dans les diverses étapes d'une méthode de maintenance du logiciel ainsi que dans celles d'une méthode d'ingénierie du logiciel. Elle est définie de la manière suivante dans la norme [ISO/IEC, 1999] :

> *La capacité du produit logiciel à être modifié. Les modifications peuvent inclure des corrections, des améliorations ou des adaptations du logiciel face à des changements dans l'environnement, dans les exigences et dans les spécifications fonctionnelles.*

La norme définit ensuite cinq caractéristiques de maintenabilité :
- **L'analysabilité** : La capacité du produit logiciel à être diagnostiqué pour des défaillances ou pour des causes d'échec dans le logiciel, ou pour identifier les parties devant être modifiées.
- **La changeabilité** : La capacité du produit logiciel à permettre qu'une modification spécifiée soit implémentée. Il est à noter que l'"implémentation" inclut le codage, la conception et les changements dans la documentation.
- **La stabilité** : La capacité du produit logiciel à éviter des effets non-attendus des modifications du logiciel.
- **La testabilité** : La capacité du produit logiciel à permettre qu'un logiciel modifié soit validé.
- **La conformité de maintenabilité** : La capacité du produit logiciel à adhérer aux standards ou aux conventions traitant de la maintenabilité.

Maintenant que ces caractéristiques sont définies, il est intéressant de préciser que ce sont les développeurs qui sont essentiellement responsables de la maintenabilité d'un logiciel étant donné que ce sont eux qui l'ont créé. Or, en pratique, le succès d'un projet de développement n'est jamais mesuré sur ce critère. Le succès d'un projet est toujours uniquement mesuré par son coût, sa durée et le fait d'avoir livré ce qu'on avait promis du côté des fonctionnalités. Il serait dès lors intéressant d'inclure cet aspect de maintenabilité au centre des préoccupations de développement de manière à diminuer les futurs coûts de maintenance du logiciel. De même, le processus de maintenance en serait amélioré de part une efficacité accrue et une durée de modification plus courte augmentant au final la satisfaction du client.

Il existe plusieurs façons de quantifier la maintenabilité d'un logiciel. Un certains nombre de facteurs liés à l'environnement de développement sont définis : la disponibilité d'une équipe du logiciel qualifiée, une structure de système compréhensible, la facilité de manipuler le système, l'utilisation de langages de

programmation standardisés, l'utilisation de systèmes d'exploitation standardisés, une structure de documentation standardisée, la disponibilité de cas de tests, des outils de débuggage intégrés et la disponibilité d'un ordinateur convenable pour conduire la maintenance.

Ensuite, un certains nombre de mesures de maintenabilité peuvent également être utilisées dans le but de mesurer l'effort nécessaire lors de la maintenance. La norme ISO 9126 (v. [ISO/IEC, 2001]) définit ainsi deux types de métriques pour les caractéristiques propres à la maintenabilité. La partie deux de la norme définit les métriques externes tandis que la partie trois définit les métriques internes. Ces métriques ne seront pas détaillés ici mais pour illustrer la différence entre les deux types, voici les définitions des *métriques de la maintenabilité* dans chacun des deux cas :

- **En tant que métrique externe** : une métrique de maintenabilité externe devrait être capable de mesurer des attributs tels que le comportement du mainteneur, de l'utilisateur ou du système incluant le logiciel lorsque le logiciel est maintenu ou modifié durant les phases de test ou de maintenance.
- **En tant que métrique interne** : les métriques de maintenabilité interne sont utilisées pour prédire le niveau d'effort requis pour modifier le produit logiciel.

Ensuite, la norme ISO définit les métriques des cinq caractéristiques de la maintenabilité selon les deux points de vue.
Enfin, il est possible d'utiliser des mesures de temps : de compréhension du problème, d'administration, de collection des outils de maintenance, d'analyse du problème, de la spécification du changement, de correction, de tests locaux, de tests globaux, de rapport de maintenance et de récupération totale.

Chapitre 2

Synthèse de la norme ISO/IEC TR 15504 (SPICE) : Technologie de l'information - Evaluation des processus du logiciel

Ce chapitre présente une norme internationale permettant d'évaluer à quel niveau d'élaboration se situent toutes les activités entourant la vie d'un logiciel. Plus précisément, cette norme décrit comment évaluer efficacement et pertinemment le niveau de maturité atteint par les processus du logiciel. Cependant, la norme ne se limite pas aux seuls processus présents dans la maintenance du logiciel. C'est la raison pour laquelle ne seront présentées ici que les parties une à cinq de la norme restreints aux concepts nécessaires dans le cadre de ce mémoire, c'est à dire ceux utiles pour la maintenance du logiciel et utilisés par le modèle S^{3m}®. Quelques concepts additionnels ont toutefois été abordés en annexe à ce mémoire (voir *Sélection et utilisation d'un modèle compatible* et *Utilisation des indicateurs*).

La norme ISO/IEC 15504 (aussi appelée « *SPICE* », pour *Software Process Improvement and Capability dEtermination*) a été élaborée par l'organisation internationale de standardisation (ISO) et par la commission électrotechnique internationale (IEC) dans le but d'établir des normes internationales afin de pouvoir réaliser des évaluations de processus pouvant ensuite mener à des *améliorations de processus* et/ou à la *détermination de leur capacité*. L'amélioration du processus vise à déterminer quelles sont les faiblesses du processus et quelles en sont les bonnes pratiques manquantes pour ensuite améliorer ces points faibles et introduire les bonnes pratiques manquantes. Tandis que la détermination de la capacité d'un processus vise à évaluer le niveau de ses différentes pratiques en les mesurant grâce à une échelle de cotations (voir la section "Une architecture pour les processus logiciels").
De cette manière, l'ISO/IEC 15504 fournit un cadre pour l'évaluation des processus logiciels. Ce cadre va intéresser les organisations impliquées dans les activités de planification, de management, de monitoring, de contrôle en améliorant l'acquisition, la fourniture, le développement et l'opération. La norme fournit une approche structurée pour l'évaluation des processus logiciel et aide une organisation à déterminer si ses processus sont adaptés aux exigences issues des bonnes pratiques et des clients et si les processus d'une autre organisation sont adaptés à un type de contrat particulier signé avec un client. Le cadre pour l'évaluation de processus encourage une organisation à s'"auto-évaluer", aborde l'adéquation du management des processus évalués, prend en compte le contexte dans lequel les processus évalués opèrent, produit un ensemble de cotations de processus (un profil de processus) plutôt qu'un résultat "réussite ou échec" et est approprié pour tous les domaines d'application et pour toutes les tailles d'organisation.
La norme permet donc d'améliorer les processus et de déterminer leur capacité. Pour qu'une organisation puisse améliorer la qualité de ses produits, elle doit posséder une méthode d'évaluation de processus prouvée, pertinente et fiable et doit avoir un moyen d'utiliser les résultats dans un programme cohérent d'amélioration continue. L'utilisation de l'évaluation de processus au sein d'une entreprise devrait encourager la culture de l'amélioration continue et l'établissement de mécanismes pour supporter et maintenir

cette culture et l'ingénierie de processus pour rencontrer les exigences d'affaire et l'optimisation des ressources. Ceci permettra alors aux entreprises de maximiser leur réceptivité face aux exigences des clients et du marché, de minimiser les coûts générés lors du cycle de vie de leurs produits et aura finalement pour résultat de maximiser la satisfaction de l'utilisateur.
Quant à l'acheteur, il pourra bénéficier de la détermination des processus afin de réduire ses incertitudes dans la sélection des fournisseurs de systèmes logiciel et de contrôler les risques qu'il peut avoir avec tel ou tel fournisseur.
Selon le modèle de référence défini à la partie deux de [ISO/IEC, 1998], l'objectif du processus d'évaluation des processus est de déterminer la mesure dans laquelle les processus logiciels standards de l'organisation contribuent à l'atteinte des buts d'affaire et aide l'organisation à se concentrer sur la nécessité de l'amélioration continue des processus.
Au final, le processus d'évaluation permettra :

- de déterminer la capacité actuelle de l'organisation et de ses processus pour produire les divers produits et services de manière cohérente avec les objectifs d'affaire ;
- les forces et faiblesses relatives des processus logiciel standards de l'organisation seront comprises ;
- des comptes rendus d'évaluation précis et accessibles seront réalisés et maintenus ;
- des revues des processus standards de l'organisation seront exécutées à des intervalles appropriés pour s'assurer de leur capacité continue et de leur efficacité grâce aux résultats d'évaluation.

Précisons enfin que le contexte du processus est important à prendre en compte car il influence le jugement d'une pratique et le degré de comparabilité entre les profiles de processus.

2.1 Les composants de l'ISO 15504

La norme est composée de neuf parties comme l'illustre la figure ci-dessous :

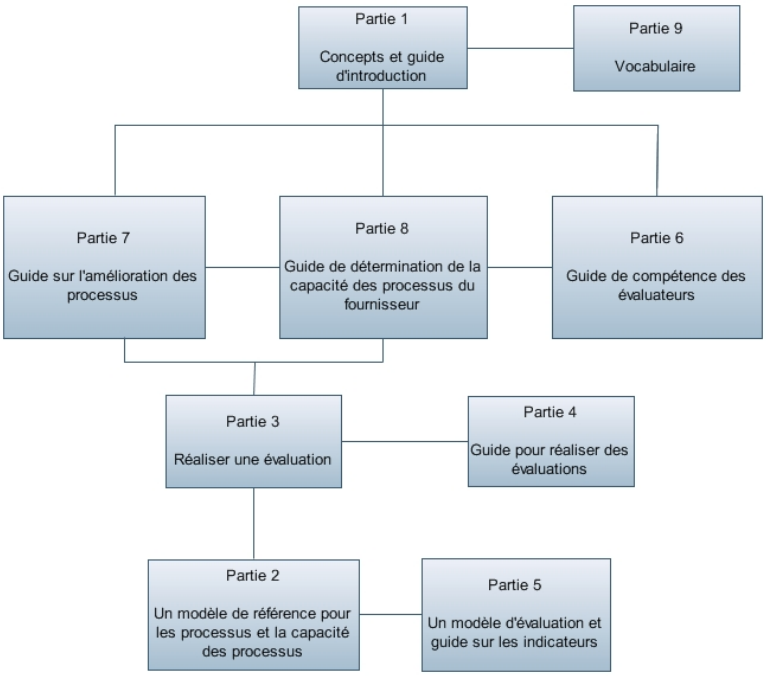


FIG. 2.1 – Les composants de l'ISO 15504 (*adapté de [ISO/IEC, 1998]*)

La partie 1 est le point d'entrée dans l'ISO 15504, elle décrit comment les parties de la suite tiennent ensemble. La partie 2 définit un modèle de référence, à deux dimensions, pour décrire les processus et la capacité des processus utilisés dans l'évaluation d'un processus. La partie 3 définit les exigences pour réaliser une évaluation d'une façon telle que les résultats seront répétables, fiables et cohérents. La partie 4 fournit un guide de réalisation d'évaluations de processus logiciels en interprétant les exigences de la partie 2 et 3 pour différents contextes d'évaluation. La partie 5 fournit un modèle exemplaire directement compatible avec le modèle de référence de la partie 2 pour réaliser des évaluations de processus. La partie 6 décrit la compétence, l'éducation, l'entrainement et l'expérience des assesseurs qui sont pertinents pour conduire des évaluations de processus. La partie 7 décrit comment définir les inputs et comment utiliser les résultats d'une évaluation pour l'amélioration de processus. La partie 8 décrit comment définir les inputs et comment utiliser les résultats d'une évaluation pour la détermination de la capacité des processus. Enfin, la partie 9 reprend le vocabulaire de tous les termes spécifiquement définis pour l'ISO 15504.

Afin d'y voir plus clair, voici une illustration des relations entre l'évaluation de processus, l'amélioration de processus et la détermination de la capacité de processus via une vue de haut niveau ;

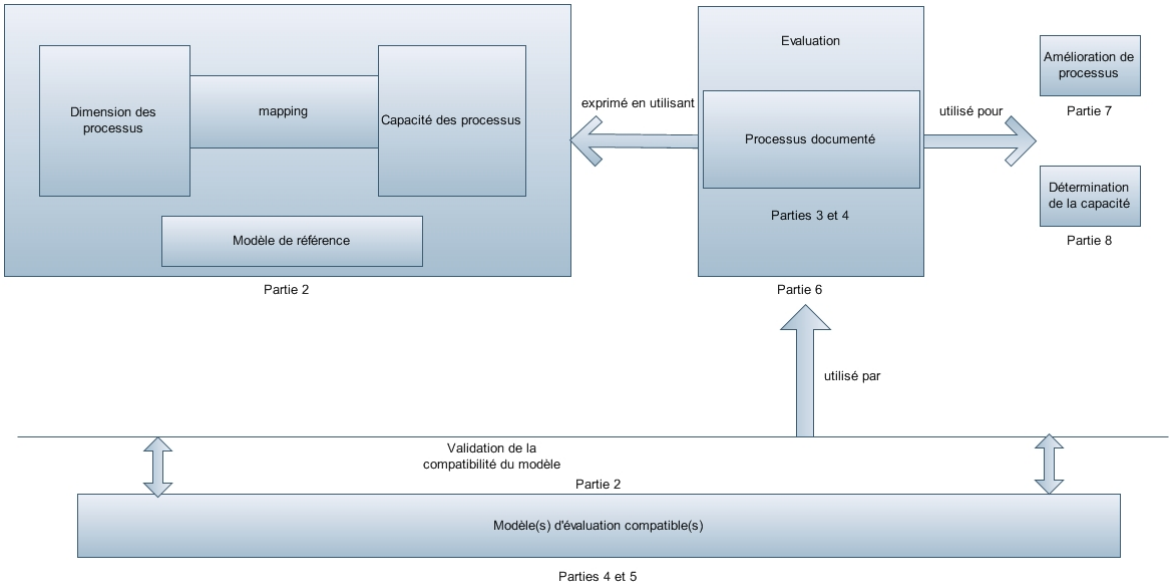


FIG. 2.2 – Vue générale des relations entre les éléments de l'ISO 15504 (*adapté de [ISO/IEC, 1998]*)

2.2 Le contexte de l'évaluation de processus

Le processus d'évaluation est initié par la demande du sponsor de l'évaluation. Les inputs de l'évaluation définissent l'identité du sponsor de l'évaluation, son objectif, son étendue, les responsabilités, les critères de compétence, l'identité du/des modèles utilisé(s) pour l'évaluation, l'identité des évaluateurs, les éventuelles contraintes et les informations additionnelles obtenues durant l'évaluation. Les inputs devraient être définis avant la phase de collection des données d'une évaluation et être approuvés par le sponsor de l'évaluation.

Une évaluation est exécutée en évaluant des processus sélectionnés par rapport à un modèle d'évaluation. Ce modèle doit être compatible avec le modèle de référence défini dans la partie deux de l'ISO 15504. Ce modèle à deux dimensions comporte un ensemble de processus et un ensemble d'attributs de processus. Les attributs de processus s'appliquent à tous les processus. Ils sont groupés dans des niveaux de capacité qui peuvent être utilisés pour déterminer la capacité du processus. Le résultat de l'évaluation inclut quant à lui un ensemble de profiles de processus et éventuellement un niveau de capacité pour chaque processus évalué.

Le processus d'évaluation contient au moins cinq activités spécifiques *(v. les parties trois et quatre de [ISO/IEC, 1998] pour le détail de ces activités)* :

1. **la planification** : Un plan pour l'évaluation devrait être développé et documenté ;
2. **l'obtention des données** : Les données nécessaires pour évaluer les processus selon l'étendue de l'évaluation. Elles devraient être obtenues d'une manière systématique et ordonnée ;
3. **la validation des données** : Des actions devraient être entreprises pour s'assurer que les données validées couvrent suffisamment l'étendue de l'évaluation ;
4. **la cotation de processus** : Une cotation devrait être assignée et basée sur des données validées pour chaque attribut de processus ;
5. **la création de rapports** : Les résultats d'évaluation devraient être documentés et rapportés au sponsor de l'évaluation.

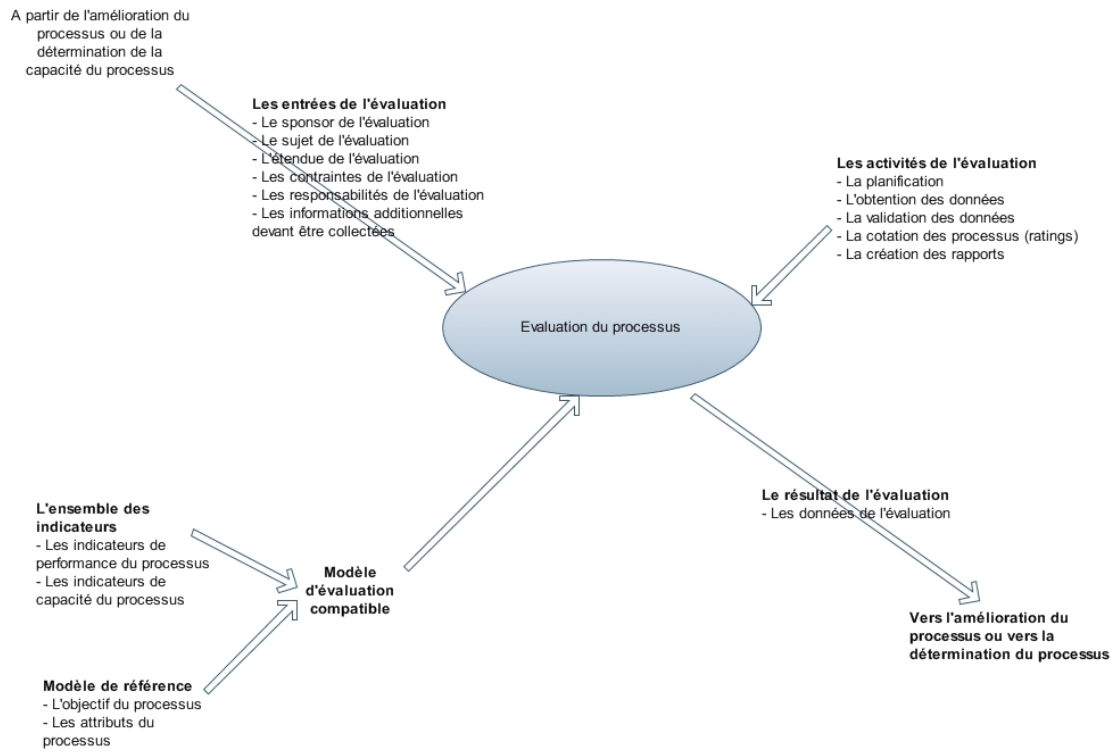


FIG. 2.3 – Le contexte de l'évaluation de processus (*adapté de [ISO/IEC, 1998]*)

Le processus d'évaluation doit être documenté et les évaluateurs doivent utiliser des indicateurs objectifs de performance et de capacité pour donner une cotation. Ajoutons enfin que le processus d'évaluation est effectué soit par une équipe ayant au moins un évaluateur compétent qui a les compétences décrites dans la partie six de l'ISO 15504 ou soit, sur une base continue en utilisant des outils adaptés pour l'obtention des données et validé par un évaluateur compétent. Le prototype expérimental $S^{3mAssess®}$, qui sera analysé dans la partie deux de ce mémoire, est un exemple d'outil permettant le support à l'évaluation.

2.3 Une architecture pour les processus logiciels

Le modèle de référence de la partie deux comprend donc une approche à deux dimensions pour l'évaluation de la capacité des processus : une dimension définissant les processus devant être évalués, l'autre décrivant l'échelle de mesure de la capacité. Tout modèle compatible avec le modèle de référence peut-être utilisé pour l'évaluation et les résultats des évaluations conformes pourront être traduits sur une base commune.
Les processus du modèle de référence sont groupés dans cinq catégories selon le type de leur activité : Client-Fournisseur, Ingénierie, Support, Management et Organisation. Ces cinq catégories sont ensuite regroupées dans trois groupes de processus du cycle de vie comme ceci :

- Les **processus principaux du cycle de vie** : comprend les catégories de processus d'**Ingénierie** et **Client-Fournisseur** ;
- Les **processus de support du cycle de vie** : comprend la catégorie de processus de **Support** ;
- Les **processus d'organisation du cycle de vie** : comprend les catégories de processus de **Management** et d'**Organisation** ;

Répondre à l'objectif d'un processus représente la première étape dans la construction de la capacité de processus. Les buts de processus seront atteints dans une organisation via diverses activités détaillées,

tâches et pratiques exécutées pour produire les divers produits. Ce sont ces tâches, activités, pratiques et caractéristiques de produits qui sont les indicateurs démontrant si l'objectif du processus a été atteint. L'évolution de la capacité du processus est exprimée en termes d'attributs de processus, lesquels sont ensuite groupés dans des niveaux de capacité. Chaque niveau de capacité représente une évolution incrémentale dans le management et le contrôle des processus de sorte que les modèles d'évaluation fournissent un itinéraire pour améliorer la capacité. Un attribut de processus décrit une facette de la capacité générale du management et de l'amélioration de l'efficacité d'un processus dans l'atteinte de son objectif et dans sa contribution aux buts organisationnels de l'entreprise. Une échelle de six niveaux de capacité est définie et ces niveaux sont caractérisés par un ensemble de neuf attributs de processus. Les attributs de processus sont utilisés pour déterminer si un processus a atteint une capacité donnée. Chaque attribut mesure un aspect particulier de la capacité processus. Ils sont eux-mêmes mesurés sur une échelle de pourcentages et par conséquent fournissent un aperçu plus détaillé des aspects spécifiques de la capacité requise pour supporter l'amélioration de processus et la détermination de la capacité. La figure ci-dessous illustre la structure à deux dimensions du modèle de référence.

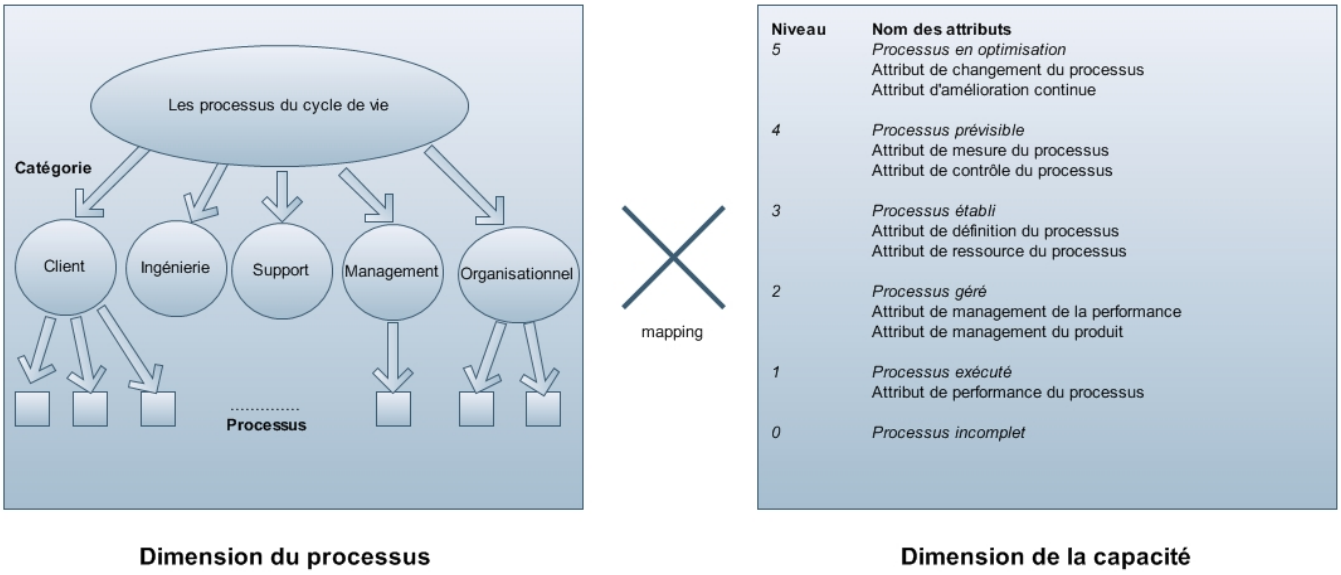


FIG. 2.4 – Les dimensions du modèle de référence (*adapté de [ISO/IEC, 1998]*)

Chaque niveau de capacité fournit une amélioration majeure de capacité de performance d'un processus. Les niveaux constituent un chemin rationnel de progression pour l'amélioration de la capacité de chaque processus. Il y a six niveaux de capacité dans le modèle de référence :

Niveau 0 : *Incomplet*
Il y a un échec général dans l'atteinte de l'objectif du processus. Il n'y a pas ou peu de produits ou résultats de processus identifiables

Niveau 1 : *Exécuté*
L'objectif du processus est généralement atteint. La réussite peut ne pas être rigoureusement planifiée et suivie. Le personnel de l'entreprise reconnait qu'une action devrait être exécutée et il y a un accord général sur le fait que cette action est réalisée comme et quand elle est requise. Il y a des produits identifiables pour les processus et ils témoignent de la réussite du processus.

Niveau 2 : *Géré*
Le processus délivre des produits selon les procédures spécifiées et il est planifié et suivi. Les produits sont

conformes aux standards spécifiés et aux exigences. La principale distinction avec le niveau *Exécuté* est que la performance du processus délivre maintenant des produits qui satisfont complètement aux exigences de qualité dans un temps défini et avec les ressources disponibles.

Niveau 3 : *Etabli*
Le processus est réalisé et géré en utilisant un processus défini et basé sur de bons principes d'ingénierie du logiciel. Les implémentations individuelles du processus utilisent des versions de standards approuvées et ajustées et des processus documentés pour l'atteinte des résultats du processus. Les ressources nécessaires pour établir la définition du processus sont aussi allouées. La distinction principale avec le niveau *Géré* est qu'ici, le processus utilise un processus défini qui est capable d'atteindre ses objectifs.

Niveau 4 : *Prévisible*
Le processus défini est réalisé systématiquement en pratique dans des limites de contrôle définies pour atteindre ses objectifs définis. Des mesures détaillées de performance sont collectées et analysées. Cela mène à une compréhension quantitative de la capacité du processus et à une aptitude améliorée pour prédire et gérer la performance. La performance est gérée quantitativement. La qualité des produits est quantitativement connue. La distinction principale avec le niveau *Etabli* est que le processus défini est maintenant réalisé systématiquement dans des limites définies pour atteindre ses objectifs.

Niveau 5 : *En optimisation*
La performance du processus est optimisée pour rencontrer les besoins d'affaire actuels et futurs et le processus peut être répété dans la rencontre de ses buts d'affaire définis. L'efficacité quantitative du processus et le rendement des cibles pour la performance sont établis et basés sur les buts d'affaire de l'organisation. Le contrôle continu du processus par rapport à ses buts est obtenu en ayant un feedback quantitatif et l'amélioration est obtenue grâce à l'analyse des résultats. La distinction principale avec le niveau *Prévisible* est que les processus définis et standards changent dynamiquement et s'adaptent pour rencontrer les buts d'affaire actuels et futurs.

La figure 2.4 illustre que le modèle de référence définit les neuf attributs de processus suivants : l'attribut de performance du processus, l'attribut de management du produit, l'attribut de management de la performance, l'attribut de ressource du processus, l'attribut de définition du processus, l'attribut de contrôle du processus, l'attribut de mesure du processus, l'attribut d'amélioration continue et l'attribut de changement du processus. Un attribut de processus représente une caractéristique mesurable pour le processus. L'échelle de cotations est une échelle de pourcentages allant de zéro à cent pourcent et représentant la réussite de l'attribut.
Une cotation est attribuée à chaque attribut de processus selon l'échelle *NPLF* suivante :

- **N :** Non atteint : de 0 à 15% : Il y a peu ou pas de preuve de la réalisation de l'attribut défini dans le processus évalué.
- **P :** Partiellement atteint : de 16 à 50% : Il y a la preuve d'une approche systématique solide et de la réalisation de l'attribut défini dans le processus évalué. Cependant, certains aspects de la réalisation peuvent être imprévisibles.
- **L :** Principalement atteint (Largely achieved) : de 51 à 85% : Il y a la preuve d'une approche systématique solide et de la réalisation significative de l'attribut défini dans le processus évalué. La performance du processus peut varier à certains endroits ou dans certaines unités de travail.
- **F :** Entièrement atteint (Fully achieved) : de 86 à 100% : Il y a la preuve d'une approche complète et systématique et de la réalisation complète de l'attribut défini dans le processus évalué. Il n'existe pas de faiblesse significative dans les unités organisationnelles définies.

L'ensemble des cotations d'attribut d'un processus forme son profil. Le résultat d'une évaluation inclut l'ensemble des profiles pour tous les processus évalués. Ajoutons que le niveau de capacité atteint par un processus est dérivé des cotations de ses attributs selon le modèle de niveaux de capacité défini ci-dessous :

Echelle	Attributs du processus	Rating
Niveau 1	Performance du processus	Largement ou complètement
Niveau 2	Performance du processus	Complètement
	Management de la performance	Largement ou complètement
	Management du produit	Largement ou complètement
Niveau 3	Performance du processus	Complètement
	Management de la performance	Complètement
	Management du produit	Complètement
	Définition du processus et ajustement	Largement ou complètement
	Ressource du processus	Largement ou complètement
Niveau 4	Performance du processus	Complètement
	Management de la performance	Complètement
	Management du produit	Complètement
	Définition du processus et ajustement	Complètement
	Ressource du processus	Complètement
	Mesure du processus	Largement ou complètement
	Contrôle du processus	Largement ou complètement
Niveau 5	Performance du processus	Complètement
	Management de la performance	Complètement
	Management du produit	Complètement
	Définition du processus et ajustement	Complètement
	Ressource du processus	Complètement
	Mesure du processus	Complètement
	Contrôle du processus	Complètement
	Changement du processus	Largement ou complètement
	Amélioration continue	Largement ou complètement

TAB. 2.1 – Cotations des niveaux de capacité (*adapté de [ISO/IEC, 1998]*)

L'objectif de cette exigence est d'assurer l'uniformité de la signification des niveaux de capacité d'un processus. Le tableau 2.1 exprime les évaluations nécessaires des différents attributs du processus pour qu'il atteigne un niveau déterminé. Les résultats de l'évaluation peuvent prendre plusieurs formes : sous forme de tableau ou sous forme graphique. Les entreprises préfèreront la forme graphique qui permet d'avoir un aperçu rapide des faiblesses à améliorer et qui permet de se comparer rapidement à d'autres entreprises. Ci-dessous, un exemple de résultat graphique d'une évaluation du niveau de maturité des processus ;

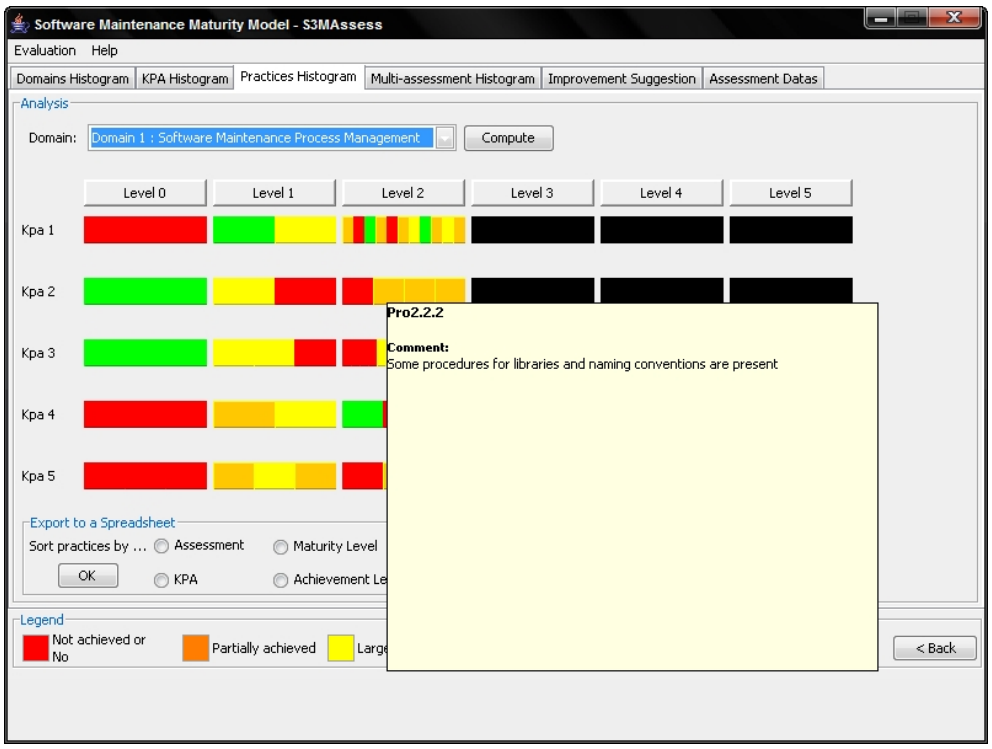


FIG. 2.5 – Exemple graphique d'un résultat d'une évaluation issu de $S^{3mAssess®}$
Légende : Rouge = N, Orange = P, Jaune = L et Vert = F, Gris = pratique non applicable

Ce graphe représente les résultats d'une cotation NPLF pour les itinéraires d'un domaine en fonction des pratiques issues du modèle $S^{3m®}$ jusqu'au troisième niveau de maturité.

2.4 Réaliser des évaluations de processus

Pour une organisation voulant comparer ses résultats d'évaluation avec une organisation similaire, il est important que ceux-ci soient fiables, cohérents et répétables. De ce fait, une évaluation rencontrant les exigences de l'ISO 15504 en est une qui :

- utilise un processus d'évaluation qui rencontre au minimum les exigences spécifiées dans la partie trois de l'ISO 15504 ;
- est basée sur un modèle d'évaluation qui est compatible avec le modèle de référence défini dans la partie deux de l'ISO 15504 ;
- utilise un ensemble compréhensible d'indicateurs de performance et de capacité de processus ;
- produit des profiles de processus utilisant le schéma de cotation d'attributs de processus défini dans ISO 15504-2 (partie deux de la norme) et ;
- a la preuve objective démontrant que les conditions ci-dessus ont été rencontrées.

2.4.1 Les approches d'évaluation

Plusieurs approches d'évaluations sont possibles. Les plus populaires sont : l'auto-évaluation et l'évaluation indépendante ;

1. **L'auto-évaluation :** Elle est exécutée par une organisation pour évaluer la capacité de ses propres processus logiciel. Le sponsor de ce type d'évaluation est normalement interne à l'organisation.
2. **L'évaluation indépendante :** Elle est conduite par des évaluateurs indépendants de l'unité organisationnelle évaluée. Par exemple, une organisation peut vouloir vérifier de manière indépendante

que ses processus d'évaluation fonctionnent correctement ; dans ce cas, le sponsor appartient à la même organisation mais pas à la même unité organisationnelle que celle évaluée.

2.4.2 Les facteurs de succès pour l'évaluation de processus

Certains facteurs sont essentiels à une évaluation de processus efficace comme l'*engagement* du sponsor lui-même envers les objectifs établis, la *motivation* des participants, la *confidentialité*, la *pertinence* et la *crédibilité*. Le fait que les évaluations de processus se concentrent sur le processus et non sur la performance des membres de l'unité organisationnelle implémentant le processus permet d'aider à augmenter la *motivation*. Le but est ainsi de rendre les processus plus efficaces pour supporter les objectifs d'affaires définis et non de blâmer les individus. La *confidentialité* pour les sources d'informations et pour la documentation obtenue lors de l'évaluation est aussi importante pour s'assurer que les participants ne se sentent pas menacés ou aient des inquiétudes. En effet, certaines informations fournies pourraient appartenir à l'organisation.
Enfin, précisons tout d'abord que les membres de l'unité organisationnelle doivent être conscient que l'évaluation peut apporter des bénéfices directs et indirects, ceci afin d'assurer la *pertinence* de l'évaluation. Ensuite, pour la *crédibilité*, il est nécessaire de présenter des résultats objectifs et représentatifs de la réalité. Il est important que toutes les parties puissent avoir confiance en l'évaluateur, c'est à dire dans son expérience adéquate, dans son impartialité et dans sa compréhension adéquate de l'unité organisationnelle et de ses affaires, sinon l'évaluation ne sera pas crédible.

2.4.3 Le cycle de la multi-évaluation

La section "*Le contexte de l'évaluation de processus*" présentait les cinq activités spécifiques minimum d'un processus d'évaluation : la planification, l'obtention des données, la validation des données, la cotation du processus et la création des rapports. Ensuite, à la fin d'une évaluation, certains résultats et exigences sont attendus. Le cycle de la multi-évaluation incluant des exigences et des résultats à l'entrée et à la sortie d'évaluations successives est représenté à la figure 2.6.

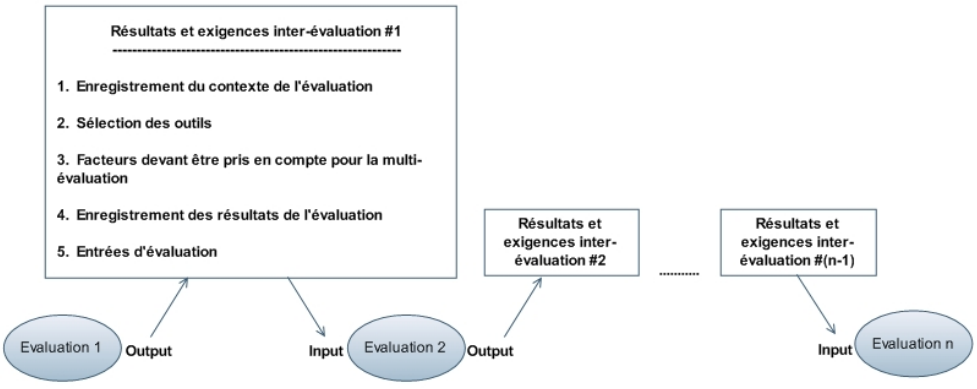


FIG. 2.6 – Le cycle de la multi-évaluation

1. Enregistrement du contexte de l'évaluation.
   L'enregistrement du contexte d'évaluation est important car le raffinement et la complexité du processus implémenté en seront dépendants au sein de l'unité organisationnelle. Le contexte du processus influencera un évaluateur dans sa façon de juger et de coter les attributs du processus. Son contexte influencera également le degré de comparabilité entre les attributs de processus et/ou les cotations de leur niveau de capacité. Par exemple, évaluer des processus d'une organisation bancaire sera différent d'évaluer une organisation aéronautique. Dans le premier cas, l'accent sera mis uniquement

sur la précision des chiffres tandis que dans le second, l'attention sera portée à la fois sur la précision, la fiabilité et la répétatibilité des processus. L'organisation aéronautique utilisera des systèmes critiques où des vies sont en jeux et ce contexte apporte bien entendu des exigences particulières pour les processus.
Dans le cas d'évaluation de départements différents au sein d'une même entreprise, les exigences requises seront différentes selon qu'il s'agisse du département Conception ou du département Opération par exemple. Il est donc nécessaire de situer le contexte d'un processus pour pouvoir évaluer sa maturité sous l'angle adéquat.
Ajoutons que les évaluateurs devront rester compétents entre les évaluations afin qu'ils puissent utiliser correctement le processus d'évaluation documenté. A cette fin, ils pourront par exemple recevoir des cours. Ces exigences ne seront pas détaillées davantage ici. Pour plus de détails, voir [ISO/IEC, 1998] à la partie trois.

2. Sélection des outils.
Dans toutes les évaluations, les informations ont besoin d'être obtenues, enregistrées, sauvegardées, comparées, traitées, analysées, récupérées et présentées. En général, un processus d'évaluation documenté est supporté par divers outils pour l'obtention d'informations, le traitement et la présentation. Pour certaines évaluations, les outils de support peuvent être manuels et basés sur papier (formulaires, questionnaires, check-lists, etc.). Dans certains cas, le volume et la complexité des informations d'évaluation sont tels que l'utilisation d'outils informatiques de support est nécessaire.
Leurs objectifs sont d'aider un évaluateur à réaliser une évaluation d'une manière cohérente et fiable, de réduire la subjectivité de l'évaluateur et d'aider à s'assurer de la validité, de l'utilisabilité et de la comparabilité des résultats d'évaluation. Dans le but d'atteindre ces objectifs, les instruments et outils ont besoin de rendre accessible aux évaluateurs le modèle d'évaluation et ses indicateurs.
Dans l'objectif de traiter les informations d'évaluation, les outils peuvent fournir un support non négligeable. La convenance d'un outil dépend du mode d'utilisation et de la méthodologie d'évaluation. Pour assurer une efficacité optimale, les outils devraient être choisis ou conçus en fonction du processus d'évaluation.
Les outils de support peuvent être utilisés de plusieurs manières :
- par les évaluateurs pour capturer les informations ;
- par les représentants de l'unité organisationnelle évaluée avant une évaluation pour un traitement futur ;
- par les représentants de l'unité organisationnelle de manière continue pendant le cycle de vie du développement du logiciel et à diverses étapes pour mesurer l'adéquation au processus, la progression de l'amélioration du processus ou pour recueillir des informations en vue de faciliter une évaluation future ;
- après l'évaluation pour récupérer et organiser les informations d'évaluation afin de faciliter la planification du processus d'amélioration ou l'analyse de la détermination de la capacité ;
- d'une manière distribuée pour l'auto-évaluation au sein d'une organisation ;
- pour assister l'évaluateur dans le traitement des informations collectées ;
- pour enregistrer et récupérer les résultats d'évaluation, les rendre plus utilisables pour la planification du processus d'amélioration ou pour l'analyse de la détermination de la capacité ;
- pour assister l'évaluateur dans son analyse des résultats post-évaluation comme l'analyse des résultats d'amélioration de processus par rapport à l'historique de performance ou comme l'analyse du profil du fournisseur par rapport à un profil cible ;
- pour collecter les informations de manière incrémentale et d'une manière distribuée, pour collecter les informations lors d'étapes "check-point" dans la performance d'un processus ou quand un nombre d'unités organisationnelles vont être évaluées de manière incrémentale ;
- pour générer des profiles de résultat ou pour aider dans l'analyse des lacunes de performance.

La compétence pour utiliser les outils sélectionnés est un facteur clé dans l'assurance que les informations seront collectées, enregistrées, traitées et analysées d'une manière fiable, répétable et appropriée. En addition à la compétence dans l'exécution des outils, l'entrainement et/ou l'expérience devraient fournir une bonne compréhension théorique des principes sous-jacents au modèle d'évaluation, aux indicateurs et à la cotation.

Des outils particuliers peuvent être spécifiés comme faisant partie intégrante d'un processus d'évaluation documenté. Ainsi, quelques points sont à prendre en considération afin de pouvoir sélectionner les outils les plus adéquats qui pourront être utilisés tout le long d'une évaluation. Toutefois, des outils complémentaires comme des traitements de texte, des outils de présentation, feuilles de calculs, etc. ne seront pas pris en considération ici bien qu'ils puissent s'avérer très utiles dans la préparation des rapports et dans la présentation des résultats.
La sélection des outils peut être influencée par :
- l'étendue et l'objectif de l'évaluation ;
- son assistance dans la collecte et dans le stockage des données ;
- la disponibilité d'un modèle d'évaluation compatible à travers l'ensemble des indicateurs définis ;
- la capacité de capturer et de maintenir des informations de support comme défini dans les entrées d'évaluation ;
- le support d'un processus de cotation et d'agrégation des résultats selon le schéma de cotation défini dans la partie deux de l'ISO 15504 ;
- le support d'une représentation des résultats dans des formes qui permettent une interprétation simple et directe de leur signification et de leur valeur ;
- la capacité de stocker et de récupérer les résultats d'évaluation pour une utilisation ultérieure dans un processus d'amélioration ou de détermination de capacité ;
- la mise à disposition d'une ségrégation appropriée des différentes classes d'informations et de données permettant de les utiliser ou de les distribuer de différentes manières ;
- la capacité de garder de manière sécurisée les informations obtenues afin de rencontrer des contraintes de confidentialité ;
- la capacité d'exécuter dynamiquement des étendues et des réductions afin de supporter des besoins culturels, organisationnels, de sponsor ou d'évaluation spécifiques ;
- la capacité de proposer un contrôle de configuration adéquat de l'outil et de ses résultats ;
- la capacité à être divisé par processus et par fonction ;
- des considérations de portabilité ;
- la capacité de manipuler de multiples entrées d'évaluateurs différents ;
- la capacité d'interagir avec d'autres outils (métriques, outils CASE, etc.) ;
- les performances en temps réel : rapidité d'insertion et de récupération des informations.

3. Facteurs à prendre en compte.
Dans le cas de la multi-évaluation, un certain nombre de facteurs doivent être pris en compte pour pouvoir comparer les résultats. Parmi ceux-ci :
- la taille de l'échantillon utilisé pour générer les cotations influence la précision avec laquelle les résultats peuvent être comparés ;
- les buts des évaluations qui ont généré les résultats de l'évaluation. Par exemple, il n'y aurait pas de sens à comparer une évaluation visant à identifier les meilleures (ou pires) pratiques avec une autre visant à identifier des pratiques représentatives ;
- le processus d'évaluation documenté ou le(s) modèle(s) utilisé(s) ;
- la compétence des évaluateurs ;
- la franchise des participants ;
- la durée de l'évaluation ;
- la motivation de l'évaluateur ;

– la volonté des participants à être francs et disponibles.

4. Enregistrement des résultats de l'évaluation.
   Entre chaque évaluation, il est nécessaire de recueillir des informations pertinentes qui permettront d'améliorer les processus et/ou de déterminer leur capacité. Les résultats serviront aussi d'entrées à l'évaluation suivante et permettront successivement d'avoir une base afin de ne pas redémarrer à zéro. En enregistrant les résultats, non seulement une comparaison sera possible avec les mêmes processus évalués dans une organisation différente mais cela permettra aussi de tracer l'évolution de la maturité au sein de l'organisation évaluée. Ainsi, un historique et des analyses de deltas seront possibles et aideront les entreprises dans leurs prises de décision suite à leurs évaluations successives.
   La figure 2.6 illustre le cycle d'une multi-évaluation et plus précisément les résultats qui en découlent. Ces résultats sont issus de la cinquième activité d'une évaluation : la création de rapports. Cette dernière activité veut en effet que les résultats d'évaluation soient documentés et rapportés au sponsor de l'évaluation. Précisons que le résultat principal d'une évaluation se présente comme un ensemble de profiles de processus. Un profil de processus est un ensemble de cotations. Ce sont ces profiles qui permettront ensuite de comparer les processus entre diverses évaluations ou par rapport à ceux d'une autre organisation.
   Les exigences sur les résultats d'une évaluation se présentent donc comme ceci :
   (a) Rédaction des informations pertinentes de l'évaluation et permettant de comprendre son résultat ;
   (b) L'enregistrement de l'évaluation devra comprendre au minimum :
      i. la date de l'évaluation ;
      ii. les entrées de l'évaluation ;
      iii. l'identification des preuves objectives obtenues ;
      iv. l'approche d'évaluation utilisée ;
      v. l'ensemble des profiles des processus résultant de l'évaluation ;
      vi. l'identification de toute information additionnelle ayant été identifiée, lors de l'évaluation, comme supportant le processus d'amélioration ou le processus de détermination de la capacité.

5. Les entrées d'une évaluation.
   Dans le but d'augmenter la probabilité d'obtenir des cotations répétables pour les processus évalués, il faut s'assurer, via des exigences, que les résultats des évaluations seront cohérents entre eux et qu'ils permettront de justifier les cotations. Ces exigences sont définies en détails dans la partie trois, clause quatre et dans la partie quatre, clause 6.2 de [ISO/IEC, 1998].
   Les entrées d'évaluation devront notamment :
   – avoir un objectif d'évaluation aligné sur les objectifs d'affaire ;
   – définir l'étendue de l'évaluation. Limiter le nombre de processus et d'attributs de processus utilisés dans l'évaluation aura pour effet de concentrer l'enquête. Des facteurs doivent également être inclus comme la relation entre l'étendue de l'évaluation et la capacité de fournir des cotations, le niveau de capacité actuel des processus et les contraintes sur la durée d'évaluation. Enfin, la sélection de l'unité organisationnelle doit refléter l'utilisation des résultats d'évaluation attendue par le sponsor ;
   – définir les contraintes d'évaluation : la disponibilité des ressources clés, le temps disponible, les parties de l'unité organisationnelle à exclure, etc. ;
   – définir l'identité des modèles utilisés pour cette évaluation. Ils devront être compatibles avec les bonnes pratiques de l'ingénierie du logiciel rencontrant les exigences définie à la partie deux de [ISO/IEC, 1998] ;

- définir l'identité des évaluateurs ainsi que leurs responsabilités. En effet, le nombre d'évaluateurs peut varier entre les différentes évaluations mais leurs connaissances et expériences doit entretenir la confiance dans les résultats de l'évaluation ;
- inclure les critères de compétence de l'évaluateur responsable de l'évaluation. La partie six de [ISO/IEC, 1998] fournit un guide concernant les compétences requises pour l'évaluateur responsable ;
- définir l'identité des évalués et de l'équipe de support avec les responsabilités spécifiques pour l'évaluation. La sélection des évalués doit être représentative de l'unité organisationnelle évaluée afin de fournir une vue précise de la capacité du processus ;
- inclure toutes informations additionnelles pouvant être collectées pendant l'évaluation afin de pouvoir supporter le processus d'amélioration ou le processus de détermination de la capacité. Ce sont par exemple les données spécifiques (ou métriques) nécessaires pour quantifier la capacité de l'organisation à rencontrer un objectif d'affaire particulier.

Une façon de rencontrer ces exigences est de les inclure dans un outil approprié qui permet l'enregistrement des observations et des preuves tout au long du processus d'évaluation. Il sera étudié, dans les parties II et III de ce mémoire, comment $S^{3mAssess®}$ intègre ces exigences.

Chapitre 3

Synthèse du modèle $S^{3m_{®}}$

Le chapitre précédent a introduit les concepts associés à un modèle d'évaluation de processus. Ce chapitre décrit maintenant un modèle d'évaluation de la maturité du processus de la maintenance du logiciel. Il est appelé « $S^{3m®}$ »pour *Modèle de la Maturité à Maintenir le logiciel* et est implémenté dans un outil de support : $S^{3mAssess®}$. Ce modèle est conforme au modèle de référence présenté dans le chapitre précédent (voir la partie 2 de la norme [ISO/IEC, 1998]). Toutefois, d'autres modèles comme le CMMi et Camélia, et normes comme l'ISO 9001 ont également influencés le développement du modèle (voir la section 3.1.1).

3.1 Introduction au modèle $S^{3m®}$

Cette section présente en quoi consiste le modèle d'évaluation de la capacité à maintenir le logiciel. Il spécifie plusieurs niveaux de maturité pour la mise en œuvre et la promotion de bonnes pratiques en gestion et en techniques de la maintenance du logiciel. Il est un moyen permettant de lancer et de guider un programme d'amélioration continue et spécialisé dans la maintenance du logiciel.
Par ailleurs, il est utile de préciser que ce modèle s'applique plus particulièrement à la maintenance du logiciel *applicatif*.
Le chapitre 1 présentait de manière générale la maintenance du logiciel. Cette dernière se subdivise en deux : la maintenance du logiciel système et la maintenance du logiciel applicatif :

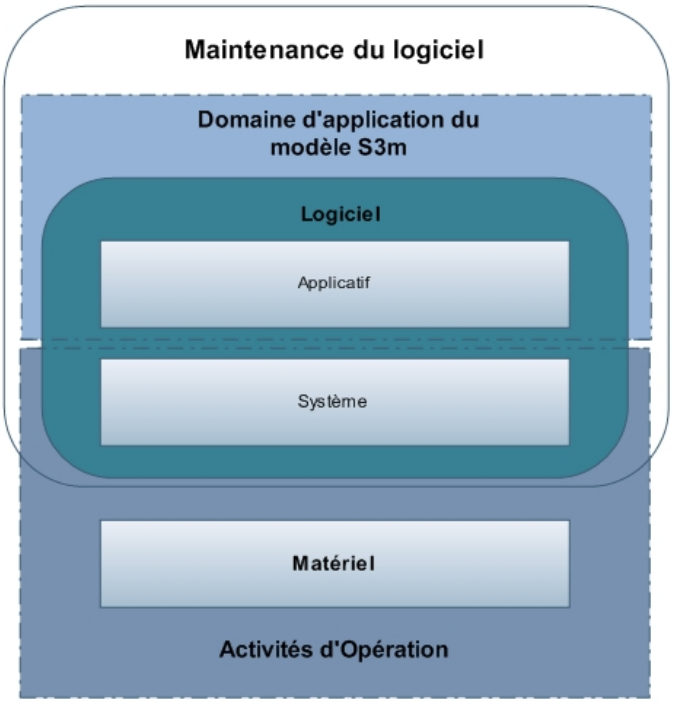


FIG. 3.1 – Domaine d'application du modèle $S^{3m®}$

Il est nécessaire d'établir une distinction claire entre l'opération d'un logiciel et sa maintenance. La norme ISO/IEC 14764 précise que les activités d'opération (copies de sécurité, recouvrement et administration des ordinateurs, etc.) sont effectuées par le personnel d'opération des systèmes informatiques et sont exclues de la portée de la norme ISO sur la maintenance des logiciels. Cette distinction de concepts est importante à mettre en avant car il est également peu commun qu'un gestionnaire confonde les opérations informatiques avec la maintenance des logiciels. Il existe une interface importante entre la maintenance et les opérations qui vise surtout à s'assurer que les infrastructures supportant les logiciels applicatifs soient opérationnelles et efficaces (par exemple : gestion des changements, appels concernant une défaillance en production, recouvrement du logiciel et des données, reprise des travaux suite à une panne ou à un désastre, etc.).

3.1.1 Les fondations du modèle

L'architecture du modèle $S^{3m®}$ est fondée sur le modèle développé par le SEI (CMMi pour le logiciel - version continue) et sur le modèle exploratoire Abran-Zitouni qui, lui-même, avait été fondé sur l'approche du modèle Camélia. Cette référence au modèle CMMi permet d'utiliser certains de ses documents lorsqu'ils sont pertinents pour mieux comprendre certaines activités de la maintenance du logiciel. En optant pour l'approche continue du CMMi, le modèle peut plus facilement :

1. se conformer à la norme ISO/IEC 15504 ;
2. obtenir une approche et une cote plus détaillée pour chaque domaine, itinéraire et facette ;
3. situer une pratique à travers les niveaux de maturité pour en démontrer la progression du niveau 0 (incomplet) aux niveaux supérieurs.

A la mise en oeuvre du modèle, il faut établir des bases qui permettent d'identifier, dans une organisation, les points forts et les faiblesses des pratiques actuelles de la maintenance du logiciel, ceci afin que les initiatives d'amélioration soient mieux orientées et dirigées. Grâce à une bonne compréhension des lacunes, les organisations seront plus en mesure d'établir des liens entre les résultats escomptés et les

bonnes pratiques, et ainsi à améliorer leur capacité à gérer et à mener à bien la maintenance des logiciels. Par ailleurs, la méconnaissance des meilleures pratiques et des bénéfices de celles-ci sur les résultats des activités de la maintenance du logiciel mène à des investissements inappropriés et mène par conséquent à :

1. créer de l'insatisfaction de la part de la clientèle ;
2. empêcher l'optimisation des délais et des coûts des services ;
3. accepter de maintenir des logiciels sans connaître leur niveau de qualité ;
4. retarder le déploiement d'améliorations, d'innovations et d'investissements en Technologie de l'information.

Ce qui mène à la définition de la maintenance de Pigoski et d'April utilisée dans ce modèle de maturité :

> *La maintenance est la totalité des activités qui sont requises afin de procurer un support, au meilleur coût possible, d'un logiciel. Certaines activités débutent avant la livraison du logiciel, donc pendant sa conception initiale, mais la majorité des activités ont lieu après sa livraison finale (l'équipe de développement ayant maintenant terminé son travail et étant affectée à d'autres travaux)* [Pigoski et April, 2004].

Le modèle $S^{3m®}$ se restreint aux aspects des processus de la maintenance du logiciel qui sont directement sous le contrôle de l'organisation. Il suit une approche pratique qui vise à appliquer les connaissances éprouvées du génie logiciel pour offrir un outil approprié visant à l'amélioration de la maintenance des logiciels dans tous les types d'industries et de toutes tailles. Ensuite, il est de la responsabilité de l'organisation de l'appliquer en respectant ses objectifs d'affaires et en tenant compte des contraintes de coûts et de délais qui lui sont propres.

Le modèle $S^{3m®}$ propose une approche structurée pour répondre à un certain nombre de problématiques présentes dans le contexte de la maintenance du logiciel comme le fait qu'elle soit une discipline opérée principalement dans le monde industriel, que le décalage est assez grand entre la littérature universitaire et les bonnes pratiques industrielles, que les propositions d'améliorations des consultants sont souvent non éprouvées et issues d'activités de marketing et que le vocabulaire est souvent hétérogène et mal défini. Ajoutons qu'il est difficile d'acquérir et d'adapter une méthodologie spécifique de la maintenance du logiciel étant donné que :

1. il n'existe pas de panacée universelle (pas de recette à appliquer telle quelle) ;
2. la littérature est souvent trop optimiste (proposant des "potions magiques") ;
3. des aspects prennent tout leur sens à partir d'une certaine taille des logiciels et des organisations.

Pour conclure cette introduction au modèle, mentionnons que la maintenance du logiciel, dans ce modèle, comprend nécessairement les activités de support opérationnel, de correction et d'évolution du logiciel. Voici quelques caractéristiques propres au domaine de la maintenance [Abran et Nguyenkim, 1993] :

1. Les requêtes de modifications (RM) parviennent d'une manière plus ou moins aléatoire et ne peuvent pas être planifiées individuellement dans un processus annuel de budgétisation ;
2. Les RM sont évaluées et placées par ordre de priorité, par le programmeur ou par son patron, et ne font pas l'objet d'autorisation individuelle par un gestionnaire senior ;
3. La charge de travail de la maintenance n'est pas gérée au moyen de techniques de gestion de projet mais plutôt par l'utilisation de files d'attentes qui sont parfois supportées par un logiciel du bureau d'aide *help desk* ;
4. La taille et la complexité des requêtes de la maintenance font en sorte que le travail est généralement effectué par seulement un ou deux programmeurs ;

5. Les travaux sont ordonnés de manière à satisfaire directement l'utilisateur et à s'assurer du bon fonctionnement quotidien des logiciels en marche ;
6. Les priorités changent rapidement et les rapports de problèmes (RP) nécessitant des corrections immédiates du logiciel en production prennent priorité sur n'importe quel autre travail en cours ;
7. Dans beaucoup d'organisations, la maintenance est souvent effectuée par des groupes organisationnels différents des groupes de développement de logiciels.

C'est pour l'ensemble de ces raisons qu'il a fallu développer un modèle de maturité propre à la maintenance du logiciel. Précisons par ailleurs que la majorité de ces caractéristiques se sont belles et bien retrouvées lors de la maintenance de $S^{3mAssess}$® effectuée dans le cadre de ce mémoire comme l'illustrera plus en détails la partie II.
Les caractéristiques du contexte du modèle sont les suivantes :

1. il porte sur des activités de maintenance de petite à taille moyenne et non pas sur des activités de grande envergure, qui devraient être gérées comme de grands projets. Toutefois, il utilise les techniques éprouvées de gestion de projet. Dans les cas spécifiques de grands projets de maintenance, c'est plutôt le modèle du CMMi qui devrait être utilisé ;
2. il est axé sur une perspective client ;
3. il est pertinent pour la maintenance :
   (a) des logiciels applicatifs développés et maintenus à l'interne ;
   (b) des progiciels configurés et maintenus à l'interne ou avec l'aide d'un sous-traitant ;
   (c) pour évaluer les pratiques utilisées en impartition avec un fournisseur externe.
4. il intègre des références à des pratiques additionnelles pertinentes de la maintenance du logiciel et propres à d'autres modèles d'amélioration du logiciel et de la qualité en général :
   – ITIL Best Practice for Service Support [Agency, 2001a],
   – ITIL Best Practice for Service Delivery [Agency, 2001b],
   – Cm^3 - Corrective Maintenance Maturity Model [Kajko-Mattsson, 2001],
   – Cm^3 - Maintainer's Education and Training Maturity Model [Kajko-Mattsson *et al.*, 2001],
   – Malcolm-Baldrige, [malcolm baldrige national quality program, 2005],
   – IT Service CMM [Niessink *et al.*, 2002].

Les pratiques du modèle S^{3m}® sont donc issues d'autres normes et documents de références. Toutefois, il est utile de préciser que la conformité aux exigences d'une pratique exemplaire S^{3m}® ne signifie pas nécessairement qu'il y a conformité avec toutes les exigences des normes ou des documents de référence correspondants.

3.2 Aperçu du modèle S^{3m}®

Cette section donne un aperçu de la portée du modèle et de ses objectifs. Ainsi, son objectif premier est d'offrir aux organisations de maintenance de logiciels un moyen permettant de démarrer et de guider un programme d'amélioration continue spécialisé pour la maintenance du logiciel.
Le modèle ne nécessite pas de questionnaire pour la cueillette des informations. Il propose des pratiques à évaluer selon un ordre croissant dans les niveaux, c'est à dire que l'évaluateur commence par les pratiques de niveau 0 jusqu'au niveau 5. Ce modèle est aussi utilisé pour évaluer la capacité d'un fournisseur d'impartition ou d'un sous-traitant de la maintenance du logiciel et il peut également être utilisé pour faire une auto-évaluation de sa propre organisation de maintenance du logiciel. Ensuite, il s'agit de présenter les résultats sous forme d'histogramme selon les pratiques, itinéraires et domaines de capacité du modèle.

Utilisation Le modèle $S^{3m®}$ et sa méthode d'évaluation sont utilisés pour toutes sortes d'objectifs comme :

1. Guide de bonnes pratiques de la maintenance ;
2. Source de gabarits pour des processus à définir ou à améliorer ;
3. Référentiel pour l'étalonnage des performances du processus de maintenance des logiciels d'une organisation par rapport aux pratiques exemplaires en industries ;
4. Aide dans le choix d'un fournisseur dans le cadre des négociations précontractuelles pour de la sous-traitance ou de l'impartition ;
5. Aide dans la détermination des occasions d'amélioration au sein d'une organisation de maintenance du logiciel selon un modèle d'autoévaluation.

Attirons l'attention du lecteur sur le fait que ce modèle, ainsi que ses outils, ne constituent ni un processus ni un modèle de cycle de vie de la maintenance du logiciel. Il se réfère plutôt aux bonnes pratiques en maintenance du logiciel. Celles-ci peuvent être utilisées pour améliorer le processus ou le cycle de vie ainsi que les services de la maintenance de l'organisation. Une organisation utilisant ce modèle se doit de rester responsable en utilisant son jugement professionnel concernant les interprétations reliées aux circonstances spécifiques de son environnement. Elle doit garder à l'esprit que les pratiques peuvent être adaptées tout en conservant leur intention et leur objectif initial.

3.2.1 Portée organisationnelle

Le modèle $S^{3m®}$ est basé sur l'hypothèse que toute organisation vise avant tout à satisfaire ses clients et à atteindre ses objectifs stratégiques. Les clients perçoivent souvent un fournisseur comme une boîte noire et c'est la raison pour laquelle le modèle considère l'ensemble des services de la maintenance du logiciel comme une unité autonome agissant comme un centre de profits, ceci afin de démontrer la valeur stratégique et commerciale de l'ensemble. Ainsi, il faut faire abstraction des frontières organisationnelles internes dans le cadre d'une évaluation $S^{3m®}$.

3.2.2 Perspective et objectifs du modèle

Le modèle a été développé selon le point de vue du client tel qu'il est perçu dans un environnement commercial concurrentiel. Dans ce contexte, la *capacité* est donc définie comme étant :

> *« L'aptitude de l'organisation à fournir de façon constante un service de la maintenance du logiciel qui répond aux critères suivants :*
>
> 1. *Le respect des attentes et des priorités du client ;*
> 2. *L'utilisation des meilleures pratiques de l'industrie de la maintenance du logiciel ;*
> 3. *Des coûts de services transparents et compétitifs ;*
> 4. *Des délais de services les plus courts possible. »*

L'évaluation selon le modèle permet d'établir un programme d'amélioration dont l'objectif ultime est la satisfaction du client et l'atteinte des objectifs stratégiques plutôt qu'une conformité rigide aux critères du modèle.

3.2.3 Avantages

Un niveau de capacité plus élevé aura une signification bien particulière selon le point de vue du client, voulant que l'organisation offrant les services de maintenance du logiciel réponde mieux à ses exigences et à celles du marché, ou selon celui de l'organisation de maintenance :

Avantages du point de vue du client	**Avantages du point de vue de l'organisation de maintenance**
a) Les services et prestations sont clairs et font l'objet d'une entente de services	a) Des coûts de maintenance expliqués à la clientèle et alloués clairement à chaque service de la maintenance selon les termes de l'entente de services
b) Les coûts et les temps d'attente sont rapportés et minimisés	b) Des requêtes et des activités placées en ordre de priorité, suivies et supervisées, ce qui permet de bien répondre aux niveaux de services convenus par l'entente de services
c) L'organisation procure un taux maximal de satisfaction	c) Les meilleures pratiques de l'industrie sont utilisées, vérifiables et promues résultant en une approche systématique de travail dont le personnel et la société sont fiers de démontrer les avantages à leur clientèle
	d) Une capacité croissante d'atteindre un bon niveau de qualité dans les objectifs de toutes les étapes du processus de maintenance du logiciel

TAB. 3.2 – Avantages d'un haut niveau de capacité dans le modèle $S^{3m®}$ selon le point de vue

Afin d'atteindre le plus haut niveau de capacité possible, l'organisation de la maintenance devrait chercher à :

1. s'assurer que le principe d'amélioration continue fasse partie de sa culture et que les mécanismes et les processus associés à cette culture soient mis en place ;
2. concevoir et à maximiser le processus de la maintenance du logiciel pour que ce dernier permette de respecter les exigences et objectifs définis pour le produit ou le service de maintenance visé ;
3. maximiser de façon continue l'utilisation des ressources engagées dans l'exécution des services de la maintenance du logiciel.

3.2.4 Mise en œuvre du modèle

Une entreprise devrait intégrer le modèle $S^{3m®}$ dans son programme interne d'amélioration continue du processus, lequel devrait normalement comprendre des évaluations périodiques (tous les 12 à 24 mois) menées à l'échelle de l'organisation. Les résultats des évaluations périodiques servent alors à alimenter un programme d'amélioration continue de la qualité, ainsi que des autres activités permanentes ou stratégiques de l'organisation.
Un autre aspect important dans le modèle est que les pratiques sont génériques et ne se limitent pas à des technologies, à des structures organisationnelles ou à des cultures d'entreprise spécifiques. Il importe donc que chaque organisation décide comment elle doit réaliser les buts visés par les objectifs génériques et spécifiques du domaine des processus de cette pratique.

3.3 Description du modèle $S^{3m®}$

Ce modèle d'évaluation de la capacité de la maintenance du logiciel inclut les éléments essentiels de plusieurs références du domaine du génie logiciel spécialisé en maintenance tel qu'énuméré à la section 3.1. Ainsi, il offre des bonnes pratiques et des itinéraires pour l'amélioration des processus de la maintenance du logiciel applicatif. Les décisions de sélection et d'implantation technologique sont spécifiques de chaque organisation. Dans le but d'utiliser le modèle, il convient d'implanter le modèle générique d'évaluation dans le contexte propre à l'organisation. Pour ce faire, il faut utiliser son jugement professionnel pour évaluer comment le contexte spécifique se compare au modèle générique d'évaluation, c'est-à-dire que les domaines, les itinéraires, les facettes et les bonnes pratiques du modèles devraient être couvertes par les processus de maintenance du logiciel de l'entreprise. Il s'agit là de la première étape pour évaluer selon $S^{3m®}$. Il est important que le diagramme de contexte de l'organisation soit suffisamment semblable au diagramme générique du modèle car sinon il ne serait pas applicable, les modes d'interaction avec le groupe de la maintenance étant trop différents. En effet, dans ce cas le modèle ne serait pas adapté pour ce type d'organisation car il n'a pas été conçu pour améliorer la capacité dans ce type d'organisation. Les pratiques et l'architecture de $S^{3m®}$ ont pris leurs bases à partir de l'expérience acquise dans des organisations fonctionnant de manière semblable au diagramme de contexte générique du modèle, illustré à la figure 3.2.

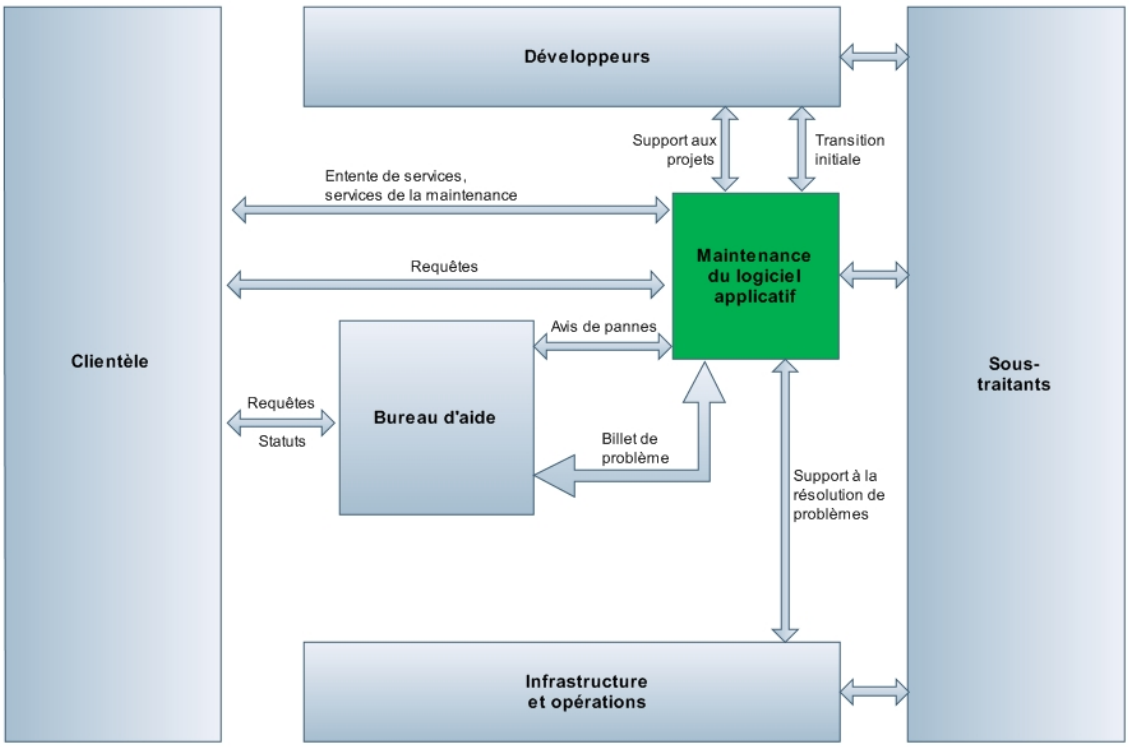


FIG. 3.2 – Diagramme de contexte du $S^{3m®}$ (*adapté de [April et Abran, 2006]*)

La figure 3.2 décrit le diagramme de contexte d'une unité organisationnelle typique de la maintenance du logiciel. Cette unité organisationnelle doit interagir avec plusieurs intervenants.
« Le gestionnaire de la maintenance doit s'assurer que les applications opérationnelles opèrent en douceur. Il doit réagir rapidement aux pannes et s'assurer que le service est restauré le plus rapidement possible. Il doit aussi observer un niveau de service établi et conserver la confiance de la clientèle quant à la compétence de son équipe à fournir un service de qualité dans les limites du budget attribué [Abran et Nguyenkim, 1993]. »

Dans la figure 3.2, la première et la plus importante interface est celle avec la clientèle. Cette interface se fait bien souvent au quotidien par plusieurs communications et services rapides concernant la résolution de problèmes opérationnels (RP), le support opérationnel et la gestion de requêtes de modifications (RM) aux logiciels applicatifs en production. Ces requêtes transitent soit directement vers le groupe de maintenance ou soit via le bureau d'aide, auquel cas elles sont appelées "billets". Une requête de modification (RM) correspond à une demande de changement, ce qui correspondra à réaliser une maintenance perfective, ou adaptative en considérant que la requête fait suite à un changement dans l'environnement. Tandis qu'un rapport de problème (RP) est issu d'un incident et la requête donnera donc lieu à une maintenance corrective. Il est à noter que les rapports de problèmes nécessitant des corrections de l'application de production auront la priorité sur n'importe quel autre travail en cours. Les RM seront quant à elles insérées dans une liste de requêtes par ordre de priorité par le programmeur ou son patron, ou parfois par un comité de priorités. La liste des requêtes est souvent supportée par un logiciel de bureau d'aide *help desk* utilisant des techniques de gestion des files d'attente.
Le processus d'acceptation ou de refus du travail de la maintenance peut-être schématisé comme ceci :

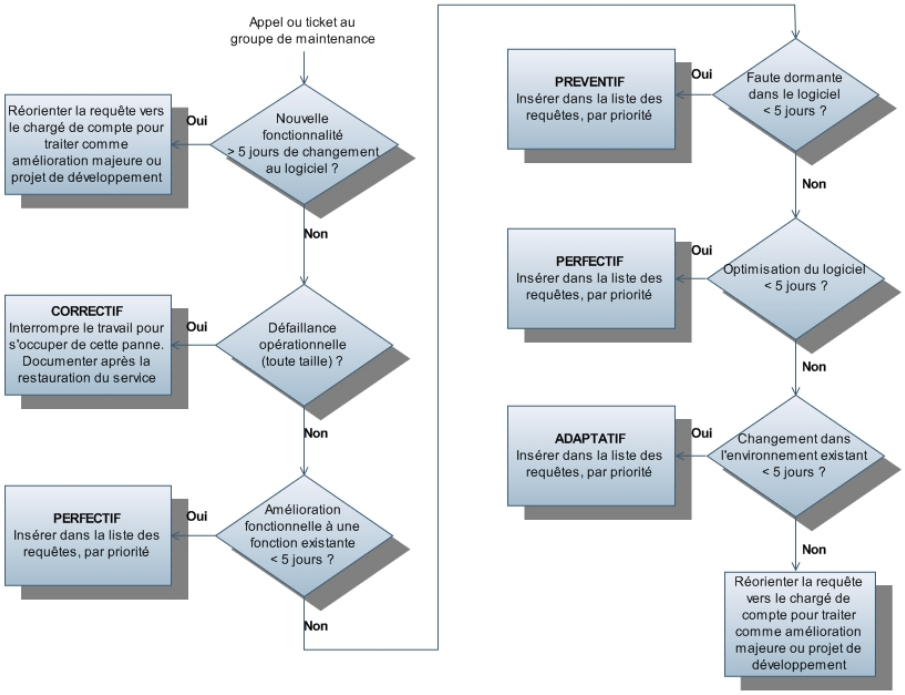


FIG. 3.3 – Processus d'acceptation ou de refus du travail de la maintenance (*adapté de [April et Abran, 2006]*)

Une autre interface quotidienne de la maintenance est celle avec le groupe des infrastructures et des opérations. Là, le processus plus global de la résolution de problèmes des logiciels y est traité. C'est encore une fois le bureau d'aide qui redirigera le billet issu du client vers l'unité organisationnelle qui peut solutionner le problème. De manière moins fréquente, une autre des activités de l'interface est de coordonner les reprises de services à la suite d'un désastre. Dans ce cas, un plan de relève devra avoir été établi au préalable et être suivi.
L'interface avec les développeurs devient plus opérationnelle et elle permet de satisfaire les besoins quotidiens de la clientèle. Il est important ici de bien contrôler la transition des logiciels afin de diminuer le nombre de problèmes. Entre ces deux intervenants, il existe aussi l'interface de support aux développeurs fournis par des mainteneurs expérimentés. Ces activités nécessitent l'expertise acquise sur les logiciels existants.
Finalement, l'utilisation de sous-traitants est une autre pratique de l'industrie et se manifeste selon toutes

sortes d'interfaces. Certains types de sous-traitants peuvent être rencontrés comme ceux qui : développent ou configurent des logiciels, faisant partie de l'équipe de maintenance, qui effectuent des activités de maintenance selon un contrat spécifique et qui remplacent complètement ou partiellement les unités organisationnelles de la maintenance. Dans ce cas, le groupe de maintenance devra négocier et bien comprendre les ententes contractuelles et gérer les contrats, les relations et les services rendus.

3.3.1 Les classes de processus de la maintenance

La figure 3.4 regroupe les processus de la maintenance en trois classes :

1. les processus opérationnels (primaires selon ISO/IEC 12207) de la maintenance ;
2. les processus de support aux processus opérationnels ;
3. les processus organisationnels offerts à toute la société par d'autres unités organisationnelles.

Le modèle générique de processus représenté à la figure 3.4 a pour objectif de donner un point de référence pour interpréter et mieux comprendre le point de vue des différentes bonnes pratiques du modèle. Il établit un cadre conceptuel et une terminologie qui servira, lors de la mise en situation, d'explications dans les exemples de pratiques spécifiques.

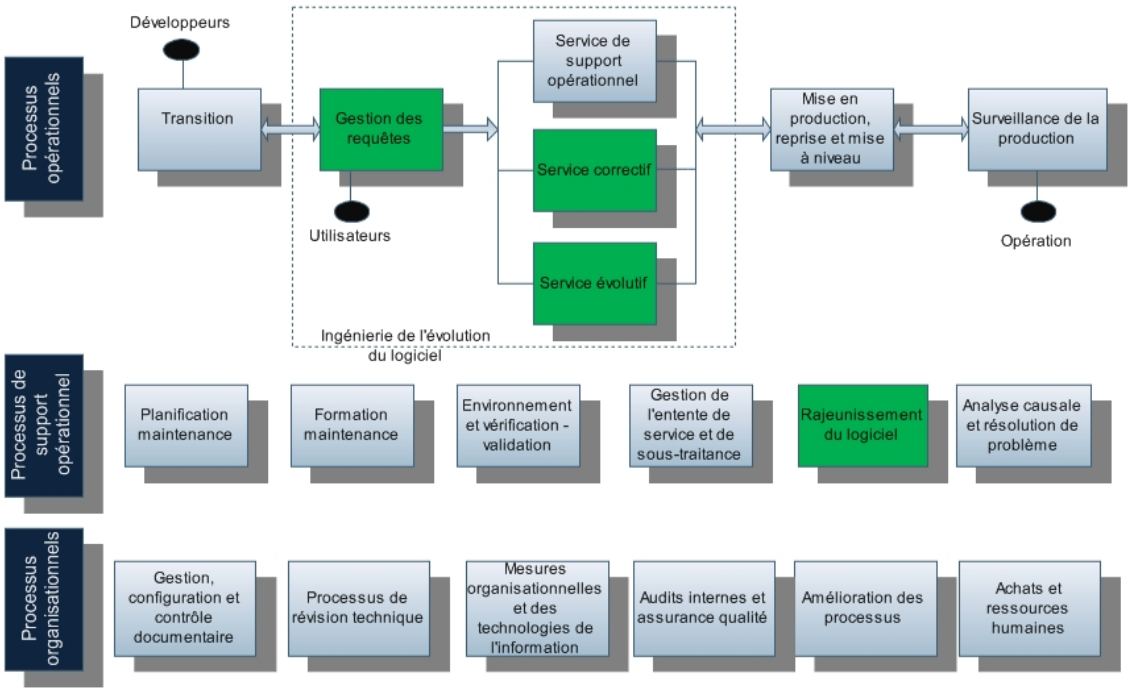


FIG. 3.4 – Classification des processus de la maintenance du logiciel (*adapté de [April et Abran, 2006]*)

Tous les processus de cette classification ne seront pas détaillés ici hormis la Gestion des requêtes, le Service correctif, le Service évolutif et le Rajeunissement du logiciel car ces quatre processus ont été particulièrement utilisés lors de la maintenance de $S^{3mAssess}$® décrite dans la partie suivante de ce mémoire. Pour plus de détails, voir [April et Abran, 2006], page 140.

3.3.2 Les processus opérationnels

Une unité organisationnelle générique de maintenance utilise les *processus opérationnels* au moment où elle reçoit un logiciel à maintenir. Le logiciel débute alors son cycle d'acquisition ou d'analyse.
Une fois le logiciel sous la responsabilité de la maintenance et encadré par une entente de services, le

processus de « Gestion des requêtes »gère les requêtes et incidents provenant de toutes sources. Ce sont ces requêtes de services quotidiennes qui doivent être gérées efficacement. La première étape est de décider s'il faut les traiter ou s'il faut les acheminer vers d'autres unités organisationnelles en fonction de leur nature et de leur taille. Les requêtes acceptées sont identifiées, placées en ordre de priorité, assignées et traitées selon les catégories de services opérationnels officielles de l'ISO :

1. Support opérationnel ne nécessitant pas de modification du logiciel ;
2. Correction du logiciel ;
3. Evolution du logiciel.

Certaines requêtes ne nécessitent pas de modification du logiciel. Dans le modèle $S^{3m®}$, elles sont classées comme *« support opérationnel »* car elles consistent à donner des informations, des conseils, documenter, etc.
Par contre, les requêtes nécessitant des modifications seront analysées, autorisées, effectuées, vérifiées et "mises en service" (mise en production). Finalement, elles seront surveillées pour s'assurer que l'environnement opérationnel ne se dégrade pas.
Le Service correctif vise à la correction du logiciel. Il s'agit plus exactement des changements rendus nécessaires par les *défaillances* ou les *défauts*. Une *défaillance* se produit lorsque le logiciel ne fonctionne plus à un moment donné alors qu'il fonctionnait correctement auparavant. Un *défaut* est une erreur dans la conception du logiciel. Il peut par exemple être induit par une corruption des données ou par des changements erronés effectués par des informaticiens. A termes, les défauts peuvent entraîner une défaillance car ils n'apparaissent pas tout de suite. La maintenance corrective est généralement mal vue car les problèmes ne viennent pas du client. Enfin, les équipes réalisant ce type de maintenance n'ont pas toujours accès aux spécifications et aux documents de conception, ce qui complique bien entendu leur tâche. Ce problème lié au manque de documentation a justement été rencontré lors de la maintenance de $S^{3mAssess®}$. La partie II présentera comment ce problème a été géré.
Le Service évolutif vise quant à lui à faire évoluer le logiciel selon de nouvelles exigences du client. Ainsi, ce service peut modifier le comportement du logiciel, proposer de nouvelles fonctions, améliorer (voir redévelopper) des fonctions existantes d'une application, développer de nouvelles fonctionnalités afin de faire face à de nouvelles exigences.

3.3.3 Les processus de support opérationnels et les processus organisationnels

Un processus requis par un processus opérationnel de la maintenance du logiciel est classé *Processus de support opérationnel*. Cette classe de processus comporte des processus qui appuient les processus opérationnels. Citons particulièrement le processus de Rajeunissement du logiciel qui vise à redocumenter, faire de la réingénierie et de la rétro-ingénierie. Ce processus s'intéresse à la résolution de problèmes reliés au déclin d'un logiciel ou à un changement majeur dans son environnement, c'est-à-dire dans les règles d'affaires, l'architecture, le langage de programmation, etc. Il y aura donc ici une tentative d'augmenter soit sa qualité, soit son efficacité ou soit sa disponibilité générale. Dans certains cas, il y aura tentative de réduction des coûts et du temps d'attente de sa maintenance. Il faut explorer les composants du logiciel pour essayer de le documenter, d'en dériver de l'information supplémentaire ou d'en changer la structure pour le rendre plus compréhensible et maintenable. L'objectif principal de ce processus est d'utiliser des techniques de rajeunissement du logiciel quand le besoin a été confirmé et approuvé lors des planifications de la maintenance :

- redocumenter le logiciel ;
- restructurer le logiciel ;
- effectuer une rétro-ingénierie du logiciel.

A la suite du succès de ce processus, les résultats suivants peuvent être attendus :

1. la baisse des coûts de la maintenance du logiciel ;
2. l'augmentation du moral des employés de la maintenance ;
3. la réduction des temps de service pour une requête de modifications provenant de la clientèle ;
4. la réduction des temps de formation d'une ressource de la maintenance ;
5. la réduction de la complexité du logiciel. La partie II du mémoire illustrera un exemple concret de réduction de la complexité, notamment au niveau de l'architecture du logiciel ;
6. l'augmentation de la qualité du logiciel (par exemple : qualité de sa documentation, de sa structure)

Ce processus est particulièrement repris dans le modèle $S^{3m®}$, au domaine "Support à l'ingénierie d'évolution", itinéraire numéro cinq "Rajeunissement du logiciel", identifié par "Sup5" (voir la section ci-dessous 3.3.4).

Les *processus organisationnels* sont quant à eux offerts par la division des Technologies de l'information et par d'autres divisions de l'organisation. La maintenance doit aussi s'intégrer dans les activités d'amélioration de l'organisation et des Technologies de l'information. Pour ce faire, elle doit suivre également les processus de la classe "Processus organisationnels".

Ce modèle générique sert à distinguer les trois classes de processus impliqués dans la maintenance du logiciel. Au début d'un processus d'amélioration, il est important d'identifier des processus plus opérationnels et ceux qui vont supporter l'unité organisationnelle. Le modèle présenté à la figure 3.4 n'est qu'un modèle de référence théorique, il est nécessaire de créer son propre diagramme en élaborant une classification qui utilise la terminologie et les processus de son organisation. Ce diagramme de haut niveau est important pour présenter un sommaire des grands processus de la maintenance du logiciel de l'unité organisationnelle.

Le $S^{3m®}$ a donc été conçu pour inclure :

1. les processus de la figure 3.4 ainsi que ses interfaces (figure 3.2) ;
2. des logiciels développés et maintenus à l'interne de l'organisation et dont le code source est disponible ;
3. des logiciels maintenus à l'interne et qui ont été acquis d'un fournisseur : le code source n'est pas disponible et il faut travailler quotidiennement avec le fournisseur ;
4. l'évaluation des processus utilisés par le fournisseur d'impartition de la maintenance.

3.3.4 Les bases du modèle

Le modèle $S^{3m®}$ regroupe des pratiques selon quatre domaines de la maintenance du logiciel :

1. Gestion du processus ;
2. Gestion des requêtes ;
3. Ingénierie d'évolution ;
4. Support à l'ingénierie d'évolution.

Chacun de ces quatre domaines regroupe ensuite de quatre à cinq itinéraires (pour un total de 18 pour tout le modèle) qui rassemblent logiquement les bonnes pratiques du modèle tel qu'illustré à la figure 3.5 :

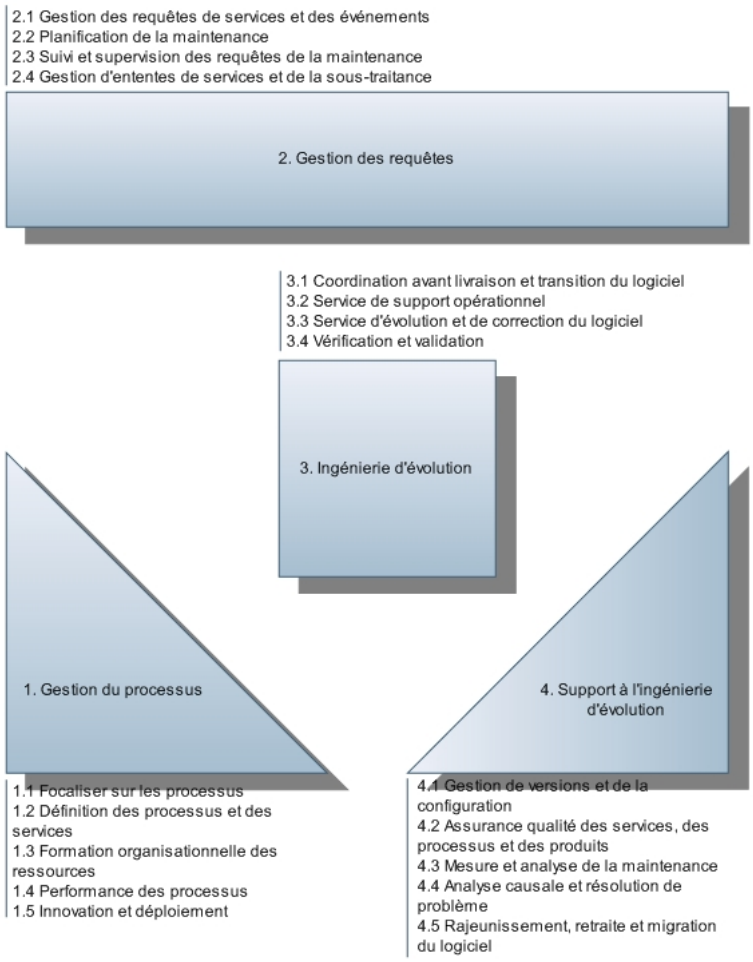


FIG. 3.5 – Domaines et itinéraires des processus du $S^{3m®}$ (*adapté de [April et Abran, 2006]*)

3.3.5 Echelle de maturité du $S^{3m®}$

Les pratiques des quatre domaines du modèle sont évaluées selon une échelle de six niveaux de maturité. Chacun des six niveaux est caractérisé comme illustré au tableau 3.3.

Niveau	**Nom du niveau**	**Risque**
0	Incomplet	le plus élevé
1	Exécuté	très haut
2	Discipliné et orienté requête	haut
3	Personnalisé et orienté processus	moyen
4	Géré quantitativement	bas
5	En optimisation	très bas

TAB. 3.3 – Niveaux de maturité (et niveau de risque associé) du $S^{3m®}$ (*adapté de [April et Abran, 2006]*)

La sémantique des niveaux est semblable à celle de l'échelle des niveaux de maturité de la norme Spice décrite au chapitre précédent. Ajoutons que la conformité au niveau 3 du modèle $S^{3m®}$ ne signifie pas pour autant une conformité automatique aux documents qui ont servi de référence au modèle.

3.3.6 Concept d'itinéraire et de facette

Le modèle $S^{3m®}$ est fondé sur le concept d'un itinéraire à suivre étape par étape. Un itinéraire est un ensemble de pratiques liées qui couvre plusieurs niveaux sur l'échelle $S^{3m®}$.
De plus, il existe aussi le concept de *facette* qui crée des regroupements au sein d'un itinéraire. Ainsi, un itinéraire peut prendre en compte un certain nombre de facettes.
Au sein d'un itinéraire ou d'une facette donnée, l'ordre de priorité des pratiques est fondé sur le degré de priorité dans la maîtrise des processus. Les pratiques les plus prioritaires se retrouvent au niveau le plus bas (niveau 0), tandis que les pratiques les plus sophistiquées sont situées au niveau le plus élevé (niveau 5).
L'architecture du modèle $S^{3m®}$ peut être résumée comme ceci : chaque domaine de capacité comprend plusieurs itinéraires qui comprennent à leur tour plusieurs pratiques couvrant plusieurs niveaux du modèle (voir [April et Abran, 2006], tableau 5.2 pour obtenir la liste des domaines, itinéraires et facettes).
Le modèle $S^{3m®}$ est conforme à la norme ISO 15504. Chaque domaine de processus possède, avec sa liste de bonnes pratiques détaillées, un texte décrivant sa conformité à la norme ISO 15504. De plus, la nomenclature du modèle $S^{3m®}$ est conforme à la norme internationale. Il suit ses recommandations pour l'identification des six composantes descriptives des processus : un **Identifiant** ayant une structure pouvant créer une architecture du général au spécifique, un **Nom** étant une courte phrase encapsulant l'objectif du processus, un **Type** étant un chiffre compris entre 1 et 5, l'**Objectif** étant un texte explicatif traitant des objectifs de haut niveau, les **Résultats attendus** décrivant les résultats observables et une **Note** d'informations additionnelles concernant le processus.

Enfin, le niveau de capacité est mesuré en évaluant un ensemble d'attributs. Chaque attribut est utilisé pour mesurer un aspect particulier de la capacité d'un processus. Chaque attribut est mesuré à l'aide d'une échelle de pourcentage calibrée de la manière suivante :

Niveau de capacité	**Nom du niveau de capacité**	**Pourcentage de calibration**
N	Pas atteint	de 0 à 15%
P	Partiellement atteint	de 16 à 50%
L	Principalement atteint	de 51 à 85%
F	Entièrement atteint	de 86 à 100%

TAB. 3.4 – Niveaux de capacité des attributs de processus du $S^{3m®}$

3.3.7 Profil de secteur $S^{3m®}$

Le modèle $S^{3m®}$ est subdivisé en quatre domaines de processus comme le présentait la section 3.3.5. Pour atteindre un niveau du modèle $S^{3m®}$, une organisation doit atteindre ce niveau **dans chacun** des quatre domaines de capacité. Les résultats d'évaluation sont présentés sommairement sous forme d'histogramme appelé « *Profil de secteur* $S^{3m®}$ ». L'établissement d'un profil de secteur est une caractérisation importante d'une organisation de maintenance de logiciels puisqu'il illustre les points forts et les points faibles de ses processus. La figure 3.6 illustre un exemple de profil pour le domaine 1, où "KPA" est l'acronyme pour "Key Process Area" ou "itinéraire" en français. Comme l'illustre la figure, il est possible d'atteindre certaines pratiques à un niveau donné sans avoir entièrement atteint toutes les pratiques du niveau précédent (par exemple, le premier itinéraire (Focaliser sur les processus) a des pratiques de niveau 2 non atteintes et des pratiques de niveau 3 entièrement atteintes).

FIG. 3.6 – Exemple de profil de secteur pour les six niveaux de maturité $S^{3m®}$
Légende : Rouge = Pas atteint, Orange = Partiellement atteint, Jaune = Principalement atteint et Vert = Entièrement atteint

3.3.8 Conventions

Chaque pratique exemplaire est numérotée de façon unique (identifiant) conformément à la convention suivante :

- Identifiant du domaine de capacité :
  - Pro - Gestion du processus,
  - Req - Gestion des requêtes,
  - Evo - Ingénierie d'évolution,
  - Sup - Support à l'ingénierie d'évolution.
- Type de processus : Pro(1) - (1 à 5) ;
- Itinéraire, ex. : Pro.1 - Focalisation sur les processus de la maintenance du logiciel ;
- Niveau de maturité, ex. : Pro.1.2 - les pratiques de niveau de maturité 2 de pro.1 ;
- Pratique exemplaire, ex. : Pro.1.2.1 - la première pratique du niveau 2 de Pro.1 :

3.3.9 Les niveaux de maturité du $S^{3m®}$

Ce paragraphe va présenter la sémantique des six niveaux de maturité du modèle. Ce modèle de maturité est basé sur une approche pratique et reconnaissable facilement par tous les intervenants dans le domaine de la maintenance du logiciel. Pour plus de détails, voir [April et Abran, 2006] pages 153-157.

- **Niveau 0 - Inexistant** : le processus n'est pas effectué par l'organisation, l'organisation n'est pas consciente de l'existence de ce processus ou il n'en existe pas d'évidence.
- **Niveau 1 - Initial, improvisé** : reconnaissance de la pratique mais elle est faite de façon informelle.
- **Niveau 2 - Discipliné - Répétable mais intuitif** : conscience de la bonne pratique, mise en place de cette pratique ou exécution d'une pratique similaire.
- **Niveau 3 - Processus personnalisé** : la pratique exemplaire est comprise et effectuée conformément à une procédure organisationnelle documentée.
- **Niveau 4 - Géré quantitativement et mesurable** : la pratique exemplaire est effectuée formellement, est gérée quantitativement, conformément à un objectif et avec des limites établies.
- **Niveau 5 - En optimisation** : la pratique exemplaire est sous contrôle statistique, conformément à un objectif et à l'intérieur des limites établies.

Au final, le modèle $S^{3m®}$ compte quatre domaines de capacité, 18 itinéraires, 75 facettes et environ 500 pratiques.

Chapitre 4

Ingénierie et Réingénierie : notions de base

Ce chapitre présente quelques notions de base d'ingénierie et de réingénierie nécessaires dans un processus de maintenance du logiciel et en particulier, nécessaires au processus de maintenance réalisé dans la partie II de ce mémoire.
La rétro-ingénierie évolue comme un lien important dans le cycle de vie d'un logiciel mais il existe beaucoup de confusion à propos de la terminologie employée, surtout par rapport à un autre concept qui est la réingénierie. Cette confusion vient de la rencontre entre les communautés de la maintenance du logiciel et du développement du logiciel. C'est pourquoi, sept concepts clés seront définis dans ce chapitre afin de fixer le vocabulaire.

Tout d'abord, les principaux objectifs poursuivis dans la réingénierie seront donnés. Suivra ensuite la distinction entre deux approches dont les définitions sont souvent mélangées : la Réingénierie et la Rétro-ingénierie. Une fois ces deux approches bien distinguées, la définition de sept concepts clés sera donnée. La synthèse des connaissances et une conclusion finale clôtureront ce chapitre. Il est à préciser que la réalisation de ce chapitre s'est principalement basée sur l'article [Chikofsky et II, 1990].

4.1 Les objectifs

L'objectif premier de la rétro-ingénierie du logiciel est d'augmenter la compréhension qu'on peut en avoir afin de le maintenir et de le faire évoluer plus facilement. Il existe cinq objectifs clés en rétro-ingénierie :

- Supporter la complexité. Il s'agit de développer des méthodes pour mieux gérer le volume et la complexité des systèmes. Ceci peut être fait grâce à des outils automatisés combinés à des environnements CASE.
- Générer des vues alternatives. Les représentations graphiques ont longtemps été considérées comme des aides à la compréhension mais les créer et les maintenir constitue un ralentissement dans le processus. C'est pourquoi il existe des outils de rétro-ingénierie facilitant la génération ou la regénération de représentations graphique. De même, il est possible de créer des formes alternatives de représentations non graphiques avec ces outils afin de former une part importante de la documentation du système.
- Récupérer des informations perdues. L'évolution continue de grands systèmes durables mène à perdre des informations sur sa conception. En effet, les modifications sont rarement inscrites dans la documentation et ceci est d'autant plus vrai lorsqu'elles s'opèrent à un niveau plus haut que le code lui-même. Ainsi, même si ce n'est pas son but premier, la rétro-ingénierie – et particulièrement la récupération de la conception – est une façon de sauvegarder les systèmes existants. Elle permet de garder une certaine maîtrise sur des systèmes dont le fonctionnement est incompréhensible ou lorsqu'il est difficile de comprendre comment les programmes individuels interagissent pour former un système.

- Détecter les effets de bord. Une conception initiale hasardeuse et des modifications successives peuvent mener à des ramifications non voulues et à des effets de bord entravant les performances du système. La rétro-ingénierie peut fournir des observations au-delà de celles que l'ingénierie en avant peut fournir et elle peut aider à détecter des anomalies et des problèmes avant que les utilisateurs ne les rapportent comme défaillances.
- Faciliter la réutilisation. La rétro-ingénierie peut aider à détecter les composants d'un système candidats à la réutilisation.

4.2 Réingénierie et rétro-ingénierie

Après avoir donné les objectifs poursuivit par la *Rétro-ingénierie*, il convient de la distinguer de la *Réingénierie* du logiciel.
La réingénierie est l'examen et la modification du logiciel pour le reconstruire dans une forme nouvelle. Elle est la forme de modification d'un logiciel la plus radicale et la plus coûteuse. La réingénierie est principalement un projet d'envergure et n'est que rarement utilisée pour améliorer la maintenabilité d'un logiciel existant. Elle est surtout utilisée pour modifier un logiciel vieillissant en se basant sur l'existant et en le réutilisant au mieux.

La rétro-ingénierie du logiciel consiste quant à elle à analyser un logiciel et ses artefacts pour en extraire une connaissance et pour décrire les interrelations de ses composants. Il s'agit en fait d'une ingénierie inverse permettant de générer une représentation du logiciel dans une forme ou à des niveaux d'abstraction plus élevés que la documentation existante. A la différence de la réingénierie, la rétro-ingénierie est passive, c'est-à-dire qu'elle **ne modifie pas** le logiciel existant et **ne crée pas** de nouveau logiciel. C'est un processus d'examen et pas un processus de changement ou de réplication. Elle consiste par exemple à produire des graphes d'appels et des graphes de contrôles de flot à partir du code source existant. Le but est d'acquérir une compréhension suffisante de la conception pour aider la maintenance, renforcer la phase d'amélioration ou supporter le remplacement.

L'article [Chikofsky et II, 1990] apprend que souvent, les mainteneurs d'un système n'en sont pas les concepteurs et qu'ils doivent par conséquent dépenser beaucoup de ressources pour examiner et comprendre ce système. Les outils de rétro-ingénierie peuvent toutefois aider dans cette tâche. Dans ce contexte, la rétro-ingénierie est une partie du processus de maintenance aidant à comprendre le système de telle façon qu'il est possible d'y apporter des changements appropriés. Enfin, il est à préciser qu'en rétro-ingénierie, le système à examiner est généralement le point de départ de l'exercice.

4.3 Définition de sept concepts

Afin de bien cadrer les quelques concepts de réingénierie et de rétro-ingénierie utilisés lors de la maintenance de $S^{3mAssess®}$, il est utile de préciser les définitions suivantes [Chikofsky et II, 1990] :

1. **L'ingénierie *en avant*** : il s'agit du processus traditionnel de passer des hauts niveaux d'abstraction et logique vers l'implémentation physique du système. Le terme "en avant" n'est pas nécessaire car implicite lorsqu'il est fait mention d'ingénierie mais il est écrit ici explicitement pour distinguer ce processus de la rétro-ingénierie.
2. **Rétro-ingénierie** : ce processus a déjà été décrit ci-dessus mais il est possible d'y ajouter qu'il implique généralement d'extraire des morceaux de conception et de construction ou de synthétiser des abstractions moins dépendantes de l'implémentation. Ainsi, la rétro-ingénierie peut débuter à partir de n'importe quel niveau d'abstraction ou d'étape du cycle de vie du logiciel. Donc, même si elle débute souvent à partir d'un système fonctionnel, ce n'est pas une exigence.

Enfin, il existe beaucoup de sous-tâches à la rétro-ingénierie. Deux principales d'entre elles sont la *redocumentation* et la *récupération de la conception*.

3. **Rétro-ingénierie des bases de données** : tentative de reconstitution des schémas logique et conceptuels de la base de données ou des fichiers d'une application à partir du code de définition des structures de données, du code source des programmes qui utilisent les données et de toute source d'information pertinente. Ceci, dans le but de comprendre leur structure, leurs contraintes d'intégrité, ainsi que leur sémantique la plus probable.
4. **Redocumentation** : c'est la création ou la révision d'une représentation équivalente au sein d'un même niveau d'abstraction. Les formes résultantes sont généralement considérées comme vues alternatives. La redocumentation est la plus simple et la plus ancienne forme de rétro-ingénierie. Le préfixe "re-" souligne l'intention de retrouver la documentation sur le système, laquelle existait ou aurait dû exister.
   Enfin, certains outils sont utilisés pour réaliser la redocumentation d'une manière élégante. Un but clé de ceux-ci est de fournir une façon plus simple de visualiser les relations parmi les composants du programme de sorte qu'il soit possible de reconnaître et de suivre les chemins clairement.
5. **Récupération de la conception** : la récupération de conception est un sous-ensemble de la rétro-ingénierie dans lequel la connaissance du domaine, les informations externes et la déduction ou des raisonnements flous sont ajoutés aux observations du système pour identifier des abstractions à plus hauts niveaux que celles obtenues en examinant le système directement. Selon Ted Biggerstaff : "la récupération de conception recrée des abstractions de conception à partir d'une combinaison de code, de documentation de conception existante (si disponible), de l'expérience personnelle et de la connaissance générale du problème et des domaines d'application...". La récupération de conception doit donc reproduire toutes les informations requises pour comprendre complètement ce que fait un programme, comment il le fait, pourquoi il le fait, etc. Il s'agit par conséquent de traiter beaucoup plus d'informations que les conventionnelles représentations de l'ingénierie du logiciel ou du code.
6. **La restructuration** : c'est la transformation d'une forme de représentation à une autre au même niveau d'abstraction tout en préservant le comportement externe du système (fonctionnalités et sémantiques). Un exemple de transformation de restructuration est de modifier le code pour améliorer sa structure et ainsi passer d'une forme non structurée à une forme structurée. Toutefois, le sens peut être plus large avec par exemple la normalisation de données qui est une transformation de restructuration de données servant à améliorer un modèle logique de données dans le processus de conception d'une base de données.
   La restructuration permet de réaliser des changements en connaissant la structure d'un programme mais il n'est pas nécessaire d'en connaître le sens. De plus, elle n'implique normalement pas de modification car pour cela, une analyse des nouvelles exigences est nécessaire. Cependant, cela pourrait mener à de meilleures observations du système suggérant des changements qui pourraient améliorer des aspects du système.
   Enfin, la restructuration est souvent utilisée comme une forme de maintenance préventive servant à améliorer l'état physique du système.
7. **Réingénierie** : aussi connue comme rénovation et récupération, elle est l'examen et la modification d'un système pour le reconstituer dans une nouvelle forme. La réingénierie inclut généralement certaines formes de rétro-ingénierie (pour obtenir une description plus abstraite) suivies de certaines formes d'ingénierie en avant ou de restructuration. Ceci peut inclure des modifications dans le respect de nouvelles exigences non rencontrées par le système originel.

4.4 Synthèse des connaissances

Afin d'avoir une idée plus claire et plus synthétique de tous ces concepts, il est intéressant de les lier entre-eux et de les recenser chacun à leur place dans un schéma global.

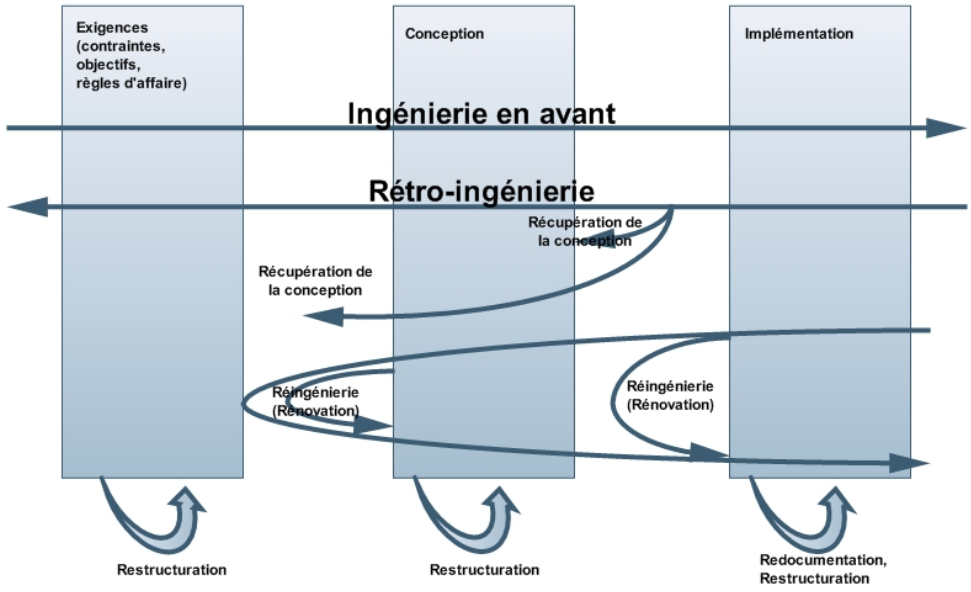


FIG. 4.1 – Relation entre les termes au sein des phases du cycle de vie d'un logiciel

La rétro-ingénierie et les processus qui y sont liés sont des transformations entre ou au sein des niveaux d'abstraction représentés ici en termes de phases du cycle de vie. La figure ci-dessus illustre notamment que les étapes de restructuration peuvent se retrouver dans les trois étapes du cycle de vie tandis que la redocumentation est propre à l'implémentation. Ceci c'explique par le fait que des représentations d'un logiciel se font typiquement sur base de son code. Ces représentations peuvent par exemple être des listings du code dans une forme plus agréable, des diagrammes reflétant le flux de contrôle ou la structure du code, etc. Par contre, la restructuration peut se faire à tous les niveaux d'abstraction. Ainsi, des représentations des exigences, contraintes, de conception ou des structures de données peuvent être transformées tout en préservant le comportement externe du système (ses fonctionnalités et sa sémantique).
La récupération de la conception s'opère lors de la rétro-ingénierie du logiciel en obtenant, selon chaque niveau d'abstraction, une compréhension supplémentaire au seul examen du système lui-même.
Enfin, la réingénierie visant à la rénovation du logiciel peut s'effectuer uniquement sur le code ou remonter jusqu'à la conception.

Ces sept concepts sont tous particulièrement importants pour ce mémoire étant donné qu'ils ont tous fait l'objet d'une activité lors de la maintenance de $S^{3mAssess®}$. Ainsi, il a été effectué :

- de l'ingénierie en avant lors de la conception et implémentation des nouvelles fonctionnalités,
- de la rétro-ingénierie principalement lors des premières phases de compréhension de l'outil,
- de la rétro-ingénierie et ensuite de la réingénierie de la base de données principalement lors des deux premières révisions de l'application,
- de la redocumentation afin de recréer les documents manquants ou incomplets obtenus lors de l'acquisition de l'outil,
- de la récupération de conception lors de la création des schémas de conception allié aux rencontres avec le maître de stage, Dr Alain April, et suite aux diverses lectures permettant de comprendre le domaine d'application de l'outil (principalement [April et Abran, 2006] et [April, 2005]),
- de la restructuration avec une implémentation différente tout en préservant le comportement de l'outil et menant par la même occasion à de la maintenance préventive (ex. la génération d'exceptions suite à des requêtes erronées sur la base de données),
- une phase de réingénierie incluant des phases de rétro-ingénierie, de redocumentation et d'ingénierie en avant et ayant permis de modifier l'outil expérimental initial et d'implémenter de nouvelles exigences.

La réingénierie de la base de données a comporté des phases de rétro-ingénierie pour la compréhension et des phases d'ingénierie en avant avec l'ajout de contraintes d'intégrité. La partie II du mémoire présentera concrètement comment tout ceci a été réalisé.

4.5 Conclusion

Les objectifs et définitions ont permis d'identifier ce qui est nécessaire pour procéder à la réingénierie d'un logiciel. Suite à cela, il découle plusieurs méthodes, techniques et outils en fonction des besoins et objectifs spécifiques à chaque cas particulier. Ainsi, il existe de nombreuses méthodes, techniques et outils décrits dans divers articles produits par divers auteurs. Cela vient donc du fait que beaucoup d'activités différentes, mais pouvant être complémentaires, sont possibles et qu'elles s'appliquent à divers systèmes. Une importante différence de systèmes est par exemple la révision d'une base de données et la révision d'un programme. Les méthodes utilisées dans ces deux cas peuvent s'avérer complètement différentes.

Conclusion et résumé des connaissances

La première section de cette partie introduisait le concept de la maintenance du logiciel : pourquoi est-elle nécessaire, quelles en sont les catégories, quels sont les objectifs, quand devrait-elle commencer pour être efficace et enfin la notion de la maintenabilité du logiciel était abordée, incluant ses caractéristiques et les manières de les mesurer. Cette section avait pour but de poser le contexte de la maintenance et de sensibiliser le lecteur sur les buts poursuivis.

Il vint ensuite la synthèse de la norme ISO/IEC 15504 : Technologie de l'information - Evaluation de processus logiciel. Cette norme est particulièrement importante car le modèle $S^{3m®}$ s'en est largement inspiré pour sa manière d'évaluer le processus de la maintenance. Ainsi, le contexte d'évaluation de processus était posé avec notamment les cinq activités spécifiques de l'évaluation : la planification, l'obtention des données, la validation des données, la cotation de processus et la création de rapports. Ensuite, l'architecture pour les processus du logiciel était présentée avec son approche à deux dimensions pour l'évaluation de la capacité : la définition des processus devant être évalués et la définition de l'échelle de mesure de la capacité. Les niveaux de capacité sont repris quasiment tel quel dans le modèle $S^{3m®}$. En effet, ils permettent de constituer un chemin rationnel de progression pour l'amélioration de la capacité de chaque processus. A cela, une échelle de cotation *NPLF* était définie pour représenter la réussite de chaque processus dans chaque niveau.
Enfin, des recommandations sur la façon d'évaluer des processus de manière fiable, cohérente et répétable étaient présentées. Plus particulièrement, un cycle de multi-évaluation spécifiant cinq exigences nécessaires entre chaque évaluation. Cette section est intéressante dans le cadre de l'outil $S^{3mAssess®}$ car elle lui fournit des exigences à respecter afin d'obtenir des évaluations pertinentes. De plus, l'outil permet justement l'enregistrement des observations et des preuves tout au long d'un processus d'évaluation.
Au final, la norme ISO 15504 occupe une place de premier plan dans le cadre de ce mémoire car elle fournit toutes les recommandations sur le contexte de l'évaluation de processus logiciel, elle a inspiré grandement la partie évaluation du processus de maintenance du modèle $S^{3m®}$ et l'outil développé dans ce mémoire s'appuie directement sur ces concepts.

Une fois le contexte général de la maintenance du logiciel posé et les recommandations de la norme ISO 15504 établies, le troisième chapitre suivit logiquement avec la description du modèle $S^{3m®}$. Ce modèle d'évaluation de la maturité du processus de la maintenance est conforme au modèle de référence présenté dans le chapitre deux. Il spécifie plusieurs niveaux de maturité pour la mise en oeuvre et la promotion de bonnes pratiques en gestion et en techniques de la maintenance du logiciel. Il est un moyen permettant de lancer et de guider un programme d'amélioration continue spécialisée pour la maintenance du logiciel. L'introduction précisait le domaine d'application du modèle centré sur le logiciel applicatif et soulignait la distinction entre l'opération d'un logiciel et sa maintenance. Ensuite, les fondations du modèle firent l'objet d'une sous-section car elles permettent de comprendre sur quoi a été bâti le modèle et dans quel but. Ainsi, le modèle du SEI (CMMi pour le logiciel – version continue) constitue une source qui a permis de créer un modèle d'évaluation spécialisé sur la maintenance beaucoup plus simple à mettre en oeuvre. En effet, à l'instar du CMMi, une évaluation de maturité du processus de maintenance basée sur $S^{3m®}$ peut être effectuée assez rapidement et avec des ressources limitée tout en respectant néanmoins les recom-

mandations de diverses normes, dont notamment l'ISO 15504. Ces exigences de limitation des ressources nécessaires, de relative rapidité d'évaluation et de spécialisation sur la maintenance font partie des principales caractéristiques et buts du modèle.
Un aperçu global du modèle donne que son objectif premier est d'offrir aux organisations de maintenance de logiciels un moyen permettant de démarrer et de guider un programme d'amélioration continue spécialisé pour la maintenance du logiciel. Le modèle propose des pratiques à évaluer selon un ordre croissant dans les niveaux, c'est à dire que l'évaluateur commence par les pratiques de niveau 0 jusqu'au niveau 5. Ce modèle est aussi utilisé pour évaluer la capacité d'un sous-traitant de la maintenance du logiciel ou pour faire une autoévaluation de sa propre organisation de maintenance du logiciel. Ensuite, il s'agit de présenter les résultats sous forme d'histogramme selon les pratiques, itinéraires et domaines de capacité du modèle. Il important de rappeler que le modèle propose un guide de pratiques exemplaires et une classification de processus de maintenance dont la responsabilité revient à l'organisation de les adapter en fonction de ses buts stratégiques.

Finalement, des éléments de réingénierie ont été passés en revue pour clore cette première partie du mémoire. Il était nécessaire de définir une taxonomie de tous les termes utilisés en rétro-ingénierie et en réingenierie afin de mieux comprendre quels en sont les activités et quels sont les buts poursuivis dans chaque cas. Ces éléments prennent également une part relativement importante dans ce mémoire étant donné qu'il est très rare de s'adonner à des activités de maintenance sans passer par ces phases. D'ailleurs, la deuxième partie de ce mémoire illustrera un cas concret de maintenance effectuée sur l'outil $S^{3mAssess®}$, laquelle comprenait également une part importante de rétro-ingénierie et de réingénierie.

L'un des problèmes approchés dans ce mémoire est de proposer des résultats d'évaluation pertinents afin d'aider les décisionnaires dans leur prise de décision sur les pratiques à améliorer selon les objectifs d'affaire, le temps, le coût, les ressources allouées, etc. La partie II présentera une manière plus complète, par rapport aux résultats actuels du modèle, pour analyser les résultats, notamment en les chiffrant, en calculant des moyennes par niveau, en offrant un historique d'évaluations, en permettant de faire des analyses comparatives entre différentes évaluations, etc. Une analyse des deltas entre plusieurs évaluations sera aussi proposée afin que l'organisation puisse déterminer les effets des améliorations de certaines pratiques sur d'autres (l'amélioration d'une pratique peut se faire au détriment d'une autre, un employé clé a pu s'en aller de l'entreprise entre deux évaluations, etc.). De plus, cela permettra d'analyser l'évolution de la maturité au fil des évaluations périodiques.

Deuxième partie

Evolution du prototype expérimental de support aux évaluations de la maturité des processus de la maintenance ($S^{3m}Assess_{®}$)

Après avoir introduit les concepts nécessaires à l'évaluation de la maturité des processus, en particulier dans le contexte de la maintenance d'un logiciel, cette deuxième partie du mémoire va décrire la maintenance et l'évolution d'un outil de support aux évaluations de la maturité des processus de la maintenance : $S^{3mAssess®}$. Des problématiques liées à la maintenance seront analysées sous divers points de vue. La toile de fond à cet effet sera donc un projet concret de maintenance. Le développement de cette deuxième partie tirera ainsi profit d'un cas concret pour développer les problématiques et analyser comment celles-ci ont pu être abordées.
Le projet concret débuta à partir d'une version prototype de l'outil qui n'était utilisable que pour l'illustration des concepts du modèle $S^{3m®}$ et pour aider à le comprendre. Il était impossible de l'utiliser pour réaliser une évaluation et encore moins pour en analyser les résultats. Grâce à ce projet, l'outil est devenu un compagnon quasiment indispensable à tout évaluateur $S^{3m®}$ et plus généralement, à toute entreprise effectuant une évaluation de maturité et voulant en analyser les résultats.

Afin de fixer les idées sur les objectifs de recherche poursuivis dans le cadre de cette deuxième partie du mémoire, voici une énumération des questions clés de cette recherche :

Q1. Comment mener à bien un projet de maintenance incluant une phase relativement importante de réingénierie, lorsque la documentation est largement insuffisante, lorsque les développeurs ne sont plus joignables, lorsque les clients sont pressés d'obtenir un résultat et quand l'effectif de l'équipe de maintenance est très réduit ?

Q2. Comment rendre l'outil tout à fait compatible avec la norme ISO/IEC 15504 ?

Q3. Comment améliorer le processus d'évaluation utilisé dans l'outil :
1. Améliorer la méthode d'évaluation standard et développer l'analyse finale au sein de l'outil
2. Rechercher comment fournir une analyse complète et pertinente qui facilite la prise de décision quant à l'amélioration des processus de maintenance ou qui facilite la détermination de la capacité des processus de maintenance
3. Intégrer la méthode S^{3m}*Assessment* dans l'outil

Deux chapitres principaux constitueront cette partie II. Tout d'abord, une *présentation du contexte de l'outil*, tel qu'il a été durant cette recherche et à son issue, sera proposée en incluant son analyse des besoins, son analyse conceptuelle et la technologie employée. Ensuite, la *maintenance de l'outil* sera abordée, c'est-à-dire la méthodologie mise en place et la critique, du point de vue de la maintenance, d'une architecture bien particulière du logiciel. La présentation de la méthodologie de maintenance permettra de répondre principalement à la première question de recherche, Q1.
Ensuite, les *développements apportés à* $S^{3mAssess®}$ *suite aux résultats de recherche* seront analysés. Ces évolutions feront suite aux demandes spécifiques des clients, du maître de stage, à une analyse de contexte et suite à l'application de la méthode S^{3m}*Assessment*. Les nouvelles fonctionnalités seront principalement liées au traitement des résultats des évaluations et à la manière de les représenter, à des suggestions d'améliorations, à l'ajout d'une nouvelle méthode d'évaluation et à la prise en compte de la multi-évaluation permettant l'analyse des deltas obtenus entre deux évaluations successives. Une politique d'amélioration sera aussi suggérée à l'utilisateur dans sa façon d'utiliser l'outil. Ces sections répondront aux deux questions de recherche restantes, Q2 et Q3.
Des détails plus techniques seront disponibles en annexe afin de ne présenter ici au lecteur que les points les plus pertinents. Les problèmes rencontrés dans ce cas pratique de maintenance et leurs résolutions seront mis en évidence.

Chapitre 5

Présentation de l'outil : $S^{3mAssess_{®}}$

Un logiciel se voit investir beaucoup d'efforts et d'argent même après qu'il ait été mis en opération ou livré aux clients et ce, pendant de nombreuses années. Ces investissements importants sont nécessaires à sa maintenance et à son évolution. Malheureusement, les tâches de maintenance et d'évolution du logiciel sont souvent négligées par le management et le manque de planification de ces tâches mène à un management de crise. Avec des développeurs adoptant des méthodologies de développement "agile" (voir le cours [Habra, 2006]), les informations et documentations fournies aux mainteneurs sont d'ailleurs encore plus pauvres. Cela entraîne des risques pour la fonction de maintenance qui est par conséquent assujettie à des efforts encore plus importants, risquant de générer au final des défaillances lors des changements apportés à un logiciel. De plus, il s'avère que les plus importants problèmes rencontrés dans la maintenance du logiciel viennent souvent du management plutôt que de la part technique.

Pour répondre à ces problèmes, le modèle $S^{3m®}$ a été développé ainsi que son outil de support aux évaluations, $S^{3mAssess®}$. Etant basé sur ce modèle, l'outil présente (voir la sous-section 3.1.1) les caractéristiques suivantes :

- il est basé sur une perspective client ;
- il est pertinent pour la maintenance d'applications logicielles :
  1. développées et maintenues en interne de l'entreprise ;
  2. configurées et maintenues en interne ou avec l'aide d'un sous-traitant ;
  3. externalisées par un fournisseur externe.
- il offre une approche d'amélioration basée sur des catégories de maintenance ;
- il couvre des processus de cycles de vie logiciels internationaux et des standards de maintenance comme ISO12207, ISO14764, ISO9001 et ISO20000 ;
- il couvre des parties pertinentes du CMMi, un modèle de référence pour l'amélioration du logiciel.

$S^{3mAssess®}$ est également basé sur la norme ISO/IEC 15504 présentée en partie I. Il permet d'automatiser les évaluations des pratiques $S^{3m®}$ et d'obtenir des analyses des mesures recueillies. Toutefois, il est utile de mentionner dès à présent que seules les pratiques des niveaux 0 à 2 ont été implémentées. Les pratiques supérieures ne sont pas accessibles dans l'outil car elles sont payantes. Des exemples illustrés de l'utilisation de l'outil sont disponibles dans la partie III du mémoire et dans les annexes.

L'outil développé a pour but de fournir du support à un évaluateur dans sa tâche d'évaluer (selon le modèle $S^{3m®}$) la maturité d'un processus de maintenance et de l'aider à comprendre les concepts sous-jacents à ce modèle. Son intention est de faciliter les activités d'évaluation et d'amélioration des processus de maintenance en centralisant à la fois toutes les pratiques pertinentes issues de divers modèles et normes internationales, et toutes les données et résultats d'évaluation.

Afin de représenter au mieux les spécifications et concepts inhérents à l'outil, un Use Case diagram, un schéma Entité-Relation, un diagramme de classes et un diagramme d'activité vont être présentés dans les deux prochaines sections. Il est important de préciser dès à présent que ces schémas sont les résultats finaux des phases de réingénierie réalisées au cours de ce projet. Seules quelques bribes de schémas obsolètes étaient disponibles au départ du projet.
Tout d'abord, le UC diagram permettra de dégager les actions des différents types d'utilisateurs de l'application. Le schéma Entité-Relation illustrera les concepts issus du contexte de l'application. Le diagramme de classes présentera les concepts liés à l'outil et permettant son fonctionnement. Enfin, le diagramme d'activité montrera un scénario typique d'utilisation de $S^{3mAssess®}$ de manière dynamique.

5.1 Analyse des besoins de $S^{3mAssess®}$

Dans son utilisation en tant que support aux évaluations de maturité, l'outil s'adresse à un évaluateur expérimenté, ayant des aptitudes adaptées, une connaissance des processus logiciels et ayant, au préalable, reçu une formation au modèle $S^{3m®}$. En effet, celui-ci devra être capable d'évaluer une pratique issue du modèle selon l'échelle *NPLF* présentée dans la partie I de ce mémoire. Cela implique que cet utilisateur puisse se poser les bonnes questions quant aux attributs caractérisant chaque pratique. Il est vital que l'évaluateur soit suffisamment compétent pour obtenir au final un résultat pertinent. Dans le cas contraire, de mauvaises décisions d'amélioration pourraient être prises suite à une évaluation incorrecte des pratiques utilisées par l'organisme évalué (voir [ISO/IEC, 1998] - partie six : Guide ISO sur la compétence d'un évaluateur).

5.1.1 Use Case Diagram de $S^{3mAssess}$®

En vue d'illustrer les acteurs, leurs rôles et leurs actions possibles dans l'utilisation de l'outil, le langage UML est utilisé via un *Use Case diagram* (illustré à la figure 5.1).

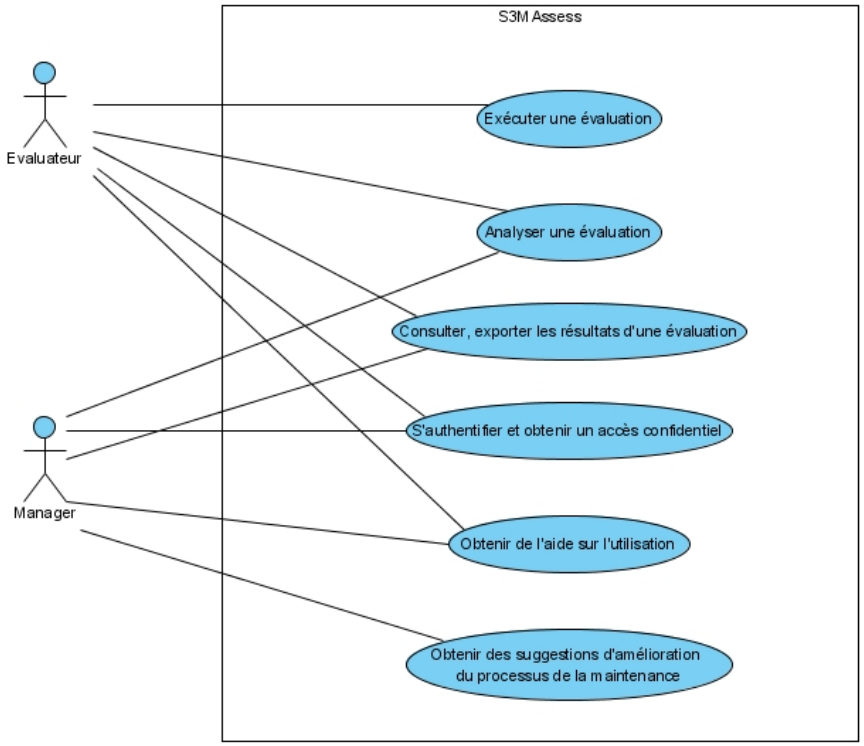


FIG. 5.1 – Use Case Diagram de $S^{3mAssess}$®

Deux types d'acteurs sont à distinguer : l'évaluateur et le manager. La différence majeure est que l'évaluateur est typiquement celui qui évaluera les processus d'une organisation, tandis que le manager sera l'acteur attaché à l'entreprise évaluée et devant prendre une décision d'amélioration, ou devant présenter les résultats d'évaluation au(x) preneur(s) de décisions, suite à l'évaluation du niveau de maturité du processus de la maintenance. A noter que l'évaluateur peut appartenir ou non à l'organisation de maintenance.
Quatre actions peuvent par contre être partagées entre ces deux acteurs :

- analyser une évaluation via les graphiques de maturité, les commentaires, moyennes, etc.
- exporter les résultats vers une feuille de calcul en vue de préparer une présentation d'entreprise, de trier les résultats selon un certain critère de pertinence, etc.
- s'authentifier à $S^{3mAssess}$® de manière à pouvoir utiliser le logiciel de manière totalement confidentielle et indépendante des autres utilisateurs. La confidentialité des données d'évaluation est très importante dans le contexte d'évaluations de sociétés concurrentes et cette exigence est d'ailleurs reprise dans la norme ISO 15504 (voir [ISO/IEC, 1998]).
- obtenir de l'aide sur l'outil. En vue d'aider l'utilisateur à comprendre l'outil sans même devoir lire un manuel d'utilisation, de la rétroaction lui est proposé via des bulles d'informations et autres affichages à l'interface.

5.1.2 Schéma Entité-Relation

Ce schéma illustre de manière logique les liens entre les concepts issus du monde réel, c'est-à-dire les concepts qu'un client manipule dans sa tâche d'évaluation de maturité selon le modèle S^{3m}®. Un

premier schéma conceptuel a été dérivé à partir d'un schéma relationnel, lui-même obtenu à partir du code implémentant la base de données dans sa version prototype. Ensuite, le schéma relationnel initial a reçu des modifications suite aux trois révisions de l'outil et de la base de données. Enfin, le schéma relationnel de la troisième révision a été conceptualisé vers ce schéma conceptuel. Le schéma conceptuel final présenté dans cette section découle donc d'une activité de réingénierie.
Il est à préciser que la documentation à ce propos était obsolète, voire complètement inexistante, et ce fût une des grandes difficultés rencontrées lors de la maintenance de l'outil dans sa phase de rétro-ingénierie. Il n'y a pas non plus eu de phase de transition entre le développement et la maintenance de $S^{3mAssess®}$ et il était impossible d'entrer en contact avec les développeurs initiaux. De plus, il n'existait aucune relation entre les tables et aucune contrainte d'intégrité (impliquant des données non cohérentes et ne facilitant donc pas la compréhension de la sémantique des colonnes). Par conséquent, il a fallu comprendre la sémantique des attributs et tables via les données qui étaient encore présentes en BD, via leur utilisation dans le code du programme, via les noms utilisés et selon la logique du modèle $S^{3m®}$.
La sous-section 6.2.1 analysera plus en détails comment s'est déroulée la conceptualisation des structures de données. La figure suivante est le résultat final de cette conceptualisation :

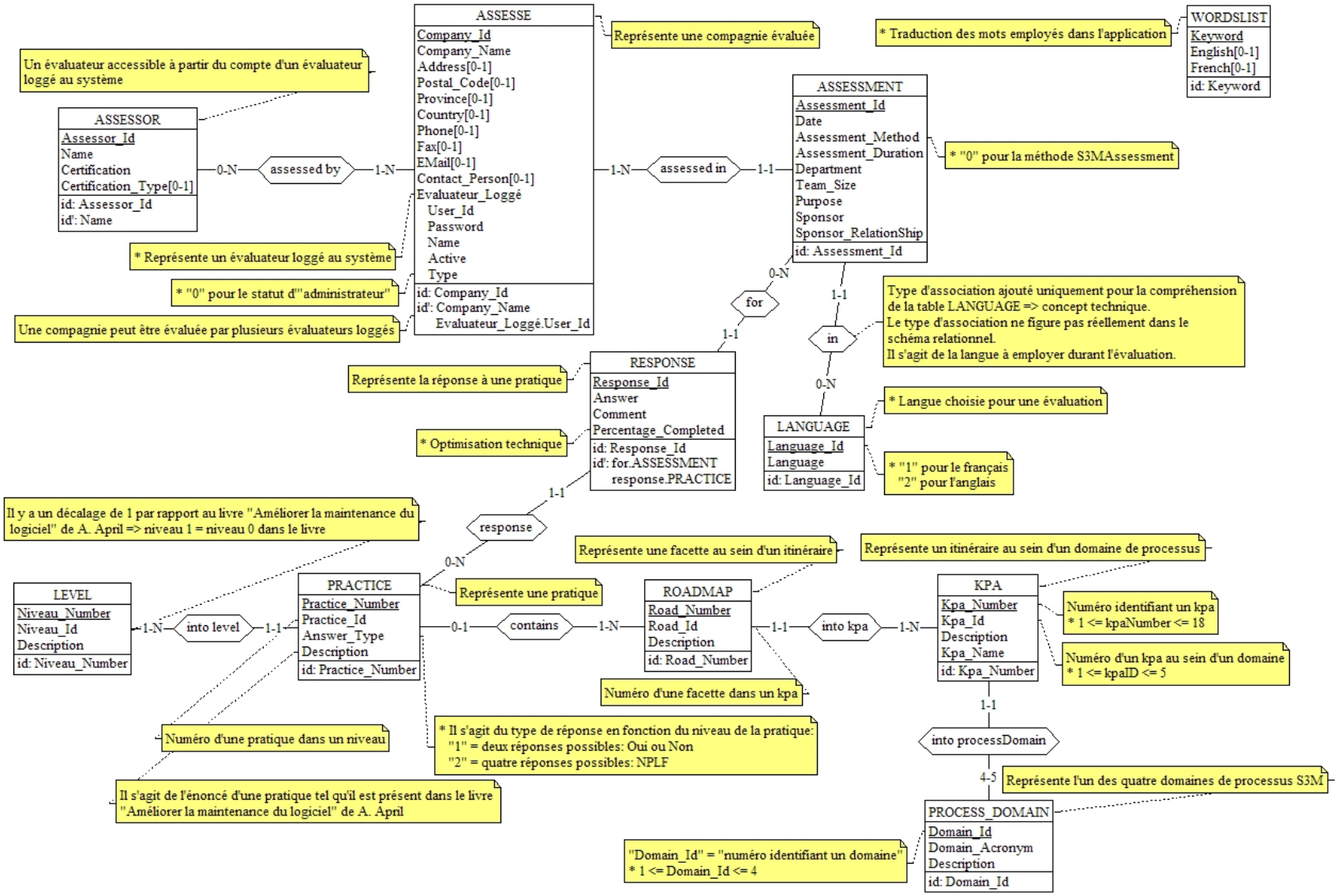


FIG. 5.2 – Schéma Entité-Relation de $S^{3mAssess®}$

Le schéma ci-dessus illustre non seulement les concepts utilisés par l'utilisateur du logiciel mais également l'architecture du modèle $S^{3m®}$. En effet, il est possible de comparer les relations entre les entités *niveau*, *question*, *roadmap*, *kpa* et *processdomain* avec la présentation faite à la section 3.3.4. Par conséquent et de part son fondement même, l'outil permet bel et bien à l'utilisateur de comprendre le fonctionnement du modèle. Cela lui permet donc de remplir un de ses objectifs principaux : aider à la compréhension du modèle sur lequel il repose.
Par ailleurs, bien qu'un schéma conceptuel ne doive représenter que des concepts du domaine d'application, quelques notes concernant des constructions techniques ont été ajoutées. Ceci vient du fait que ce schéma provient d'une phase de rétro-ingénierie et que cela permettra d'aider les futurs mainteneurs et utilisateurs de l'outil. En effet, un des buts de la rétro-ingénierie est de redocumenter. Les astérisques indiquent donc un concept ou une note technique. Etant donné l'absence officielle de schéma logique, ces concepts techniques ont été laissés volontairement dans le schéma conceptuel (voir la section 6.2.1).
Dernière précision, certains noms de tables et colonnes ont également été normalisés lors de la phase de normalisation du schéma conceptuel. En effet, certains manques au niveau de la syntaxe, sémantique et du choix des noms ont été observés. Ceux-ci n'ont pas été modifiés en BD étant donnés les problèmes que cela engendre après implémentation. Ils ont toutefois été normalisés pour ce schéma conceptuel afin de garantir une certaine rigueur et afin d'en faciliter la lecture.
Le chapitre traitant de la maintenance de $S^{3mAssess®}$ reviendra sur les contraintes d'intégrité qui ont été ajoutées au cours des trois révisions. Il retracera l'évolution du schéma relationnel initial.

5.2 Analyse conceptuelle de $S^{3mAssess®}$

5.2.1 Diagrammes de classes

Les diagrammes de classes de $S^{3mAssess®}$ présentent de manière statique les composants de l'outil ainsi que leurs relations. Une classe représente un *objet*, ou *concept*, comprenant un ensemble d'attributs le caractérisant et un ensemble de méthodes permettant de manipuler cet objet. Pour profiter du fait que le langage orienté objet Java a été utilisé pour implémenter ce logiciel, le diagramme de classes représentera non seulement les concepts liés à l'application mais aussi les objets Java constituant l'architecture même de $S^{3mAssess®}$. Cet avantage du double emploi peut être justifié ici car la découpe initiale en classes a été réalisée de manière satisfaisante.

Deux diagrammes de classes vont être présentés ici : tout d'abord un diagramme représentant la relation entre les divers paquetages de classes composant l'application et ensuite un diagramme simplifié ne reprenant que les classes, attributs et méthodes principales de certains de ces paquetages. Pour une version plus détaillée de ces diagrammes, voir en annexe.
Enfin, le diagramme de classes représentant la gestion de la persistance des données est à consulter plus loin car sa complexité a fait l'objet d'une section à part entière (v. la section 6.2.2.3).

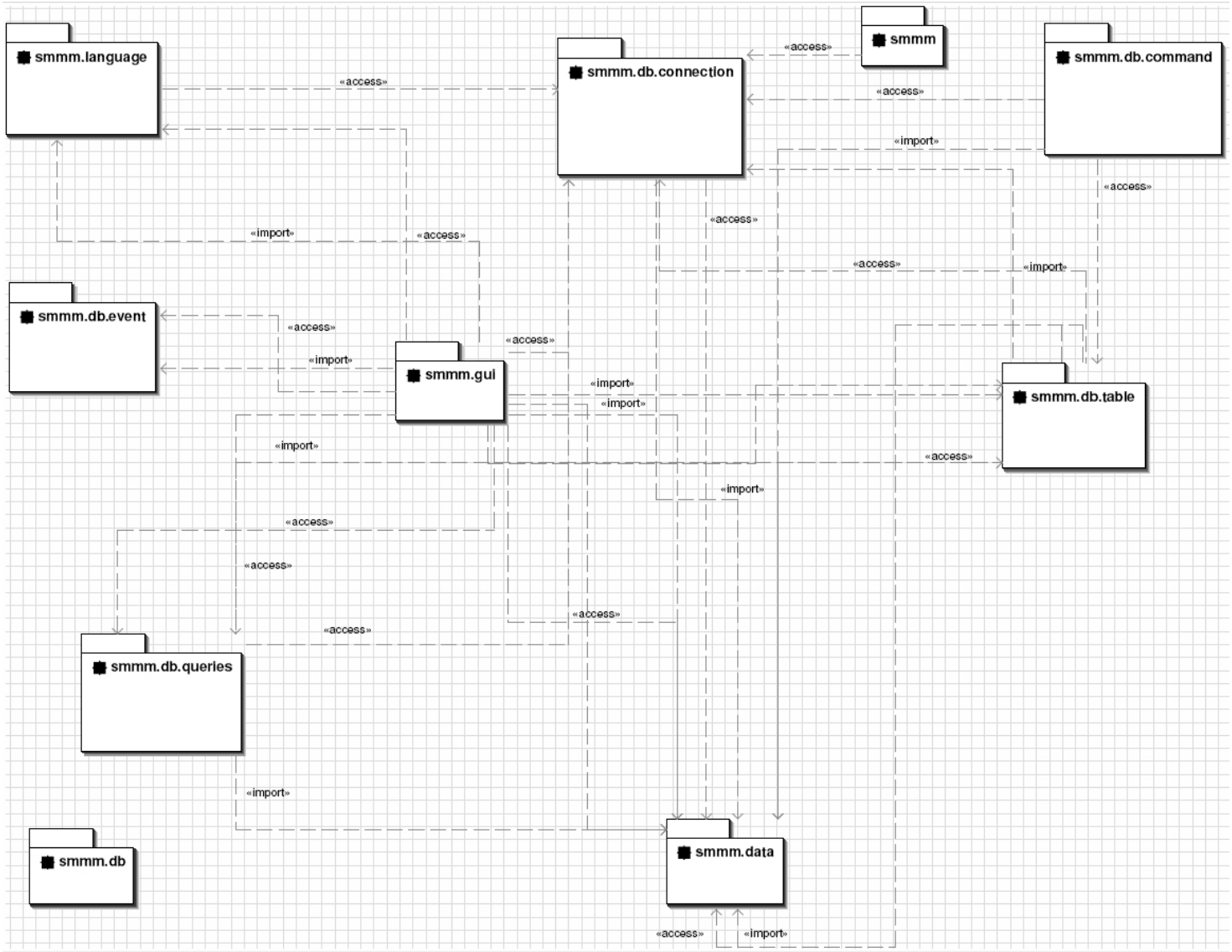


FIG. 5.3 – Diagramme de classes de $S^{3mAssess®}$ – Vue des paquetages

Dix paquetages sont présents, gérant chacun un type de tâche bien particulier : le lancement de l'application (smmm), les interfaces (smmm.gui), les langues (smmm.language) et la base de données. En ce qui concerne la gestion de cette dernière, les tâches sont réparties en sept paquetages gérant : la connexion à la BD (smmm.db.connection), les différents types de *commandes* qu'il est possible d'exécuter sur une table (smmm.db.command), les événements que ces commandes génèrent (smmm.db.event), la gestion individuelle de chaque table (smmm.db.table), les commandes spécifiques de type "requête" (ici, principalement pour recevoir les réponses d'une évaluation) (smmm.db.queries) et les constantes utilisées pour gérer la BD (smmm.db).
Deux types de relations figurent dans ce premier schéma : la relation *"import"* et *"access"*. La première signifie que des classes d'un paquetage importent d'autres classes appartenant à d'autres paquetages. Autrement dit, d'autres objets sont encapsulés dans l'objet qui importe.
La relation "access" signifie simplement qu'une classe accède à une autre classe via un appel de méthode ou via un accès à un attribut après avoir "instancié" un objet.

Le diagramme suivant présente maintenant les relations entre les classes principales de $S^{3mAssess®}$. Ici, l'abstraction portera uniquement sur les classes constituant l'interface graphique étant donné que les tâches de Business y sont également implémentées :

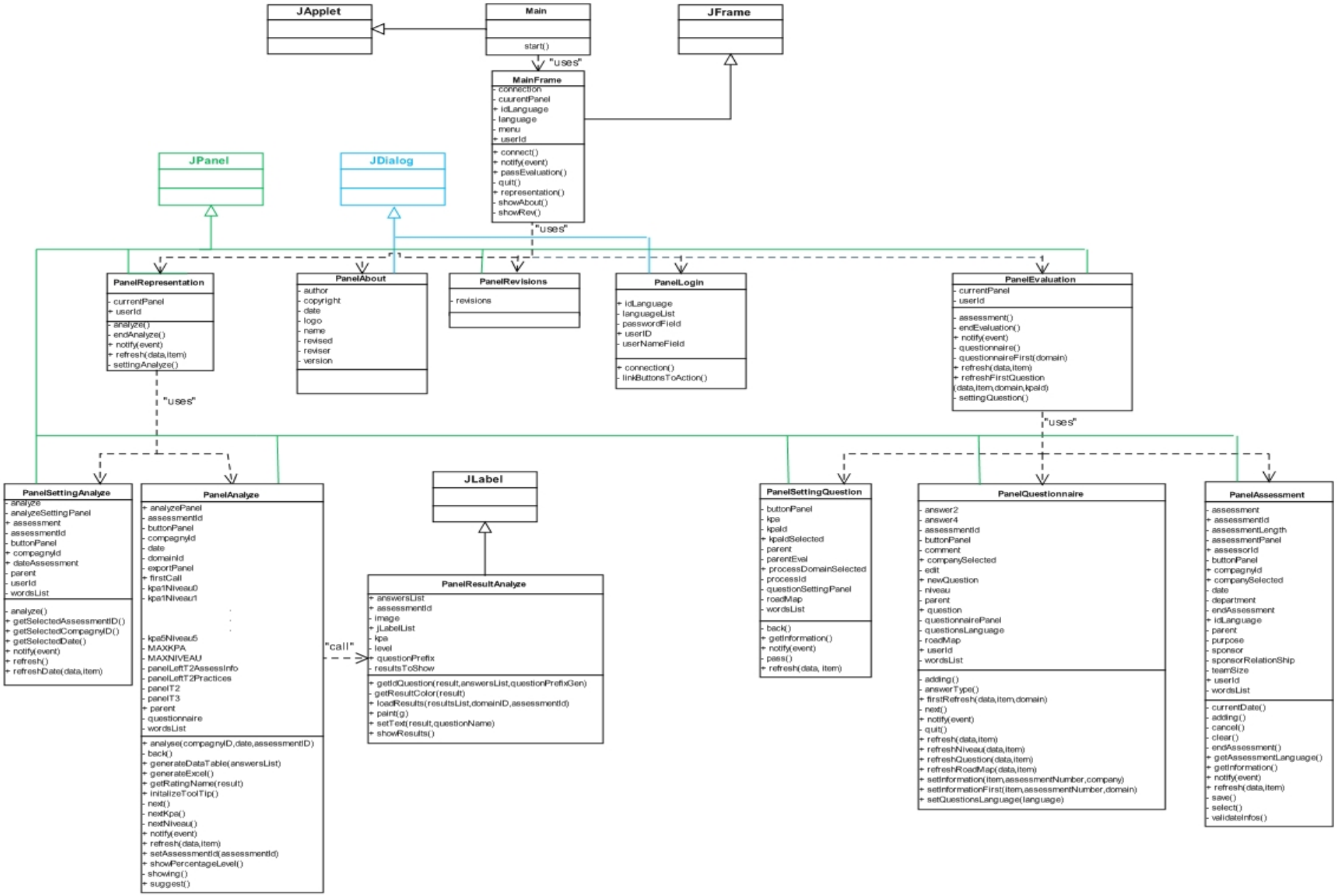


FIG. 5.4 – Diagramme de classes de $S^{3mAssess®}$ – Vue des interfaces

La lecture de ce schéma commence par la classe Main lançant la méthode start(). Un objet MainFrame est alors créé et lance une fenêtre permettant à l'utilisateur de s'identifier via un login et un mot de passe. Si les informations d'identification sont correctes, le programme $S^{3mAssess®}$ se lance et l'utilisateur a le choix de débuter une tâche en sélectionnant l'interface à afficher via un menu. A son instanciation, l'objet MainFrame crée en fait tous les panneaux, c'est-à-dire les écrans qu'il est possible d'afficher : un panneau permettant de représenter les résultats d'une évaluation (PanelRepresentation), un panneau affichant des informations générales sur l'outil (PanelAbout), un panneau affichant les modifications apportées au cours de chacune des trois révisions réalisées dans le cadre de ce projet (PanelRevisions), le panneau permettant l'identification préalable de l'utilisateur (PanelLogin) et un panneau permettant de réaliser une évaluation (PanelEvaluation).

Une fois que l'utilisateur a fait son choix à partir du menu, deux panneaux majeurs ont pu être sélectionnés : PanelEvaluation ou PanelRepresentation. Etant donné la complexité de leur tâche, ils utilisent chacun d'autres classes comme ceci :

- **PanelEvaluation** :
  - PanelAssessment : panneau permettant à l'utilisateur d'entrer les informations générales concernant l'évaluation qu'il s'apprête à opérer ;
  - PanelSettingQuestion : panneau permettant à l'utilisateur de choisir le domaine et l'itinéraire qu'il veut évaluer ;
  - PanelQuestionnaire : panneau permettant à l'utilisateur d'évaluer les pratiques du modèle $S^{3m®}$ qu'il a choisies via le panneau précédent.

- **PanelRepresentation** :
  - PanelSettingAnalyze : panneau permettant à l'utilisateur de sélectionner l'évaluation d'une compagnie réalisée à une certaine date ;
  - PanelAnalyze : panneau permettant à l'utilisateur d'analyser une évaluation. Il pourra alors notamment afficher un graphe NPLF représentant les résultats et affichant les éventuels commentaires. Il lui sera également loisible d'exporter ces résultats vers une feuille de calculs. Pour une liste exhaustive des possibilités d'analyse, voir le chapitre 6.2.3 et le guide utilisateur en annexe.
    Via des onglets, il pourra obtenir les informations générales d'évaluation et les résultats sous forme de tableau. Enfin, il pourra également choisir l'onglet permettant d'afficher des suggestions d'amélioration de pratiques par domaine et par niveau de domaine.
    Finalement, il est à préciser que la classe PanelAnalyze s'appuie sur la classe PanelResultAnalyze qui lui permet principalement d'afficher le graphe NPLF ainsi que les informations y afférant.

Enfin, dans un but de complétude de l'information donnée par ce diagramme d'abstraction des interfaces, les relations d'héritage des classes implémentées vers les interfaces Java ont été ajoutées. Ces interfaces sont : JApplet, JFrame, JPanel, JDialog et JLabel.

5.2.2 Diagramme d'activités

Le diagramme qui va suivre représente, selon une vue de haut niveau, la dynamique de l'exécution de l'activité majeure de $S^{3mAssess®}$: évaluer et ensuite analyser un processus de maintenance. Les diverses étapes et les relations de précédence impliquées dès la connexion de l'utilisateur jusqu'à l'analyse des évaluations sont ainsi illustrées.

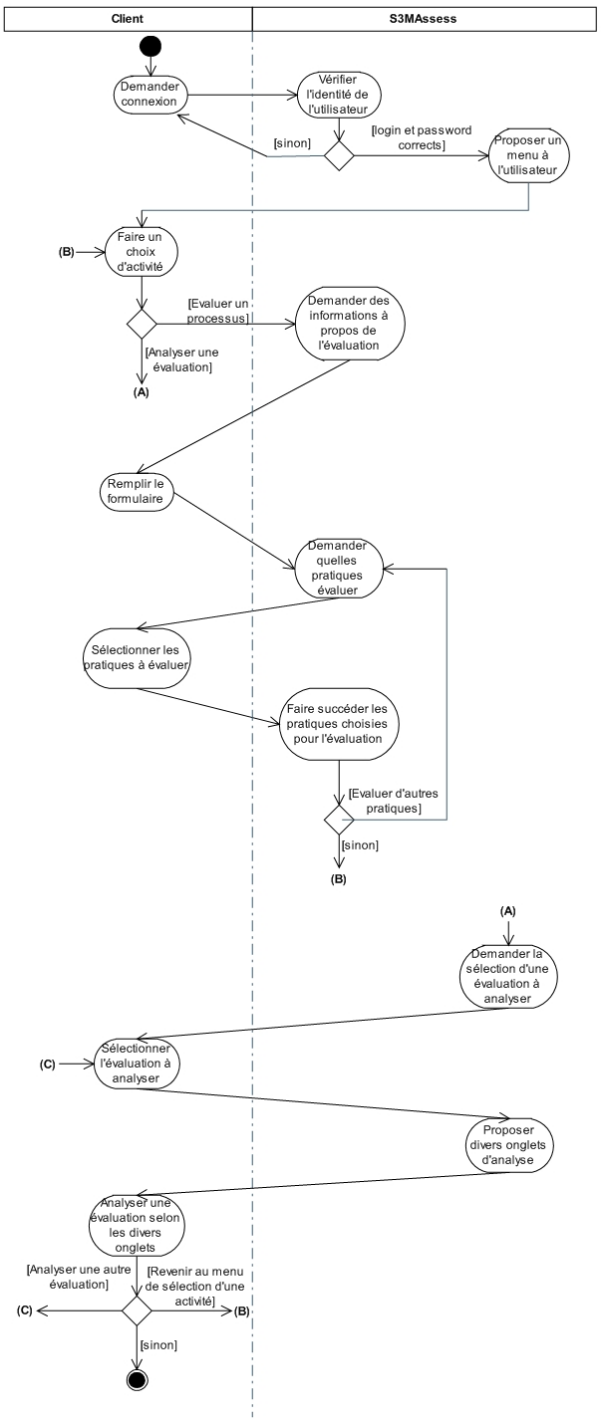


FIG. 5.5 – Diagramme d'activités – Vue de haut niveau

5.3 Technologies employées

Cette section présente les divers choix technologiques qui ont été fait pour réaliser le projet de maintenance de $S^{3mAssess®}$. Ainsi, seront présentés le langage de programmation orienté objet Java, le principe des applets Java, le type de base de données qui a été retenu pour l'outil, son Système de Gestion de Base de Données, le serveur HTTP Apache permettant à un utilisateur distant de se connecter à l'outil et les techniques d'accès distants utilisés afin de garantir un certains niveau de sécurité. Cependant, si les notions principales seront présentées dans cette section, l'utilisateur sera néanmoins invité à se rendre dans la partie 11 *Technologies employées* des annexes s'il désire davantage de précisions.

5.3.1 Le langage Java

Java est un langage de programmation orienté objet et un environnement d'exécution portable. Ce langage a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels qu'Unix, Microsoft Windows ou Mac OS avec peu ou pas de modification.

Les concepteurs de ce langage ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.). Une particularité souvent énoncée est que cette approche permet de développer des logiciels plus maintenables de part le fait que les objets permettent de scinder le programme en plusieurs sous-tâches plus simples. Chaque objet ou *classe* permet aussi de ne traiter qu'un type de problème particulier. Les notions de division en tâches simples et d'unité logique d'un objet permettent théoriquement à un mainteneur de comprendre plus rapidement et facilement et de maintenir plus efficacement un logiciel. Toutefois, le chapitre 6 illustrera, notamment via la section 6.2.2.2, que ceci n'est malheureusement pas toujours aussi simple. Ainsi, des aspects de couplage et de cohésion mal gérés (voir le cours [Habra, 2006]), l'absence de la phase de transition entre les phases de développement et de maintenance du logiciel, le manque de documentation, l'absence de contact entre développeurs et mainteneurs, etc. peuvent malgré tout mettre à mal la maintenance d'un logiciel.

Le langage Java a été un bon choix dès le départ de l'implémentation de $S^{3mAssess®}$ car il permet naturellement de modulariser. A chaque objet peut être attribuée une tâche simple et spécifique. Cela permet de simplifier la complexité de la tâche de départ en la divisant en plusieurs objets traitant des sous-tâches plus simples.
Toutefois, la maintenabilité du système reste liée au fait qu'une bonne découpe initiale est primordiale à la compréhension du système et à sa logique et au fait que plus le nombre de classes est élevé, plus la compréhension devient difficile. De plus, il existe une règle fondamentale propre à la maintenance : *la complexité augmente avec l'évolution du logiciel.* En effet, les modifications et nouveautés amènent à complexifier l'architecture initiale du logiciel. Ceci est induit par la nécessité de créer de nouvelles structures de données, d'en modifier d'autres, d'éventuellement détourner l'utilisation initiale de certains objets, méthodes ou variables et est aussi induit de manière importante par la succession des développeurs ayant chacun leurs propres affinités et idées.

Au final, l'approche orientée objet permettant de construire des programmes en tant que composants enfichables (des objets ayant chacun un but spécifique et pouvant être réutilisés), la possibilité de créer des applications client-serveur et la qualité d'indépendance de la plateforme d'exécution font du langage Java un bon choix pour l'implémentation de $S^{3mAssess®}$.

5.3.2 Applet Java

Une *Applet* est un logiciel s'exécutant dans la fenêtre d'un navigateur internet. Cette approche permet d'éviter au client de devoir installer une application et un serveur de base de données sur son ordinateur. En effet, non seulement un client ne veut pas forcément encombrer son ordinateur d'un tel système, mais il n'a pas forcément le temps ni les connaissances techniques à cette fin. L'expérience du cas concret de maintenance de $S^{3mAssess®}$ a démontré ce fait. C'est pourquoi, l'utilisation d'une applet disponible via une simple URL est pertinente.
Pour plus de détails sur l'utilisation des applets dans le cadre de $S^{3mAssess®}$, notamment à propos de la mise en place d'un certificat de sécurité, voir la partie 11 *Technologies employées* à la section "Applet Java" en annexe.

5.3.3 Une base de données relationnelle

La base de données relationnelle a été choisie par rapport à d'autres mécanismes d'enregistrement pour ses performances, sa fiabilité et sa facilité d'accès aux données. Etant donné la complexité des données à enregistrer et le nombre de requêtes nécessaires à l'exécution de $S^{3mAssess®}$, une BD relationnelle est plus adaptée que des mécanismes comme l'écriture dans un simple fichier sur le disque dur, la sérialisation Java, la création d'un fichier XML ou qu'une base de données objet. Ainsi, il n'est pas nécessaire de devoir lire séquentiellement toutes les lignes d'un fichier avant de pouvoir lire la ligne voulue. De même, l'insertion, la mise à jour et la récupération se font plus facilement via un langage structuré de requêtes : *SQL* (Structured Query Language).

Des techniques intéressantes entourent ce type de BD comme les diagrammes Entité-Relation aidant à concevoir la BD et à en communiquer les parties entre différentes personnes. Egalement, il existe la technique de la *Normalisation* qui est un processus permettant d'extraire les données redondantes pour les stocker dans des tables séparées. Finalement, un langage structuré permettant de communiquer avec la BD via des requêtes (SQL) a été spécifiquement mis au point. Une application l'incluant dans son code peut également communiquer avec la BD et ainsi enregistrer, effacer ou obtenir des informations à partir d'elle.

5.3.4 Le Système de Gestion de la Base de Données

Le SGBD ayant été retenu est *MySQL*. MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées. Il est multithread, multiutilisateurs. C'est un logiciel libre développé sous double licence en fonction de l'utilisation qui en est faite : dans un produit libre (open-source) ou dans un produit propriétaire.
MySQL fonctionne sur beaucoup de plateformes différentes. $S^{3mAssess®}$ a ainsi été développé dans un environnement Windows et est accessible aux clients dans un environnement Unix. La figure 5.6 illustre le déploiement du système sur les serveurs de l'Ecole de Technologie Supérieure de Montréal via la connexion d'un utilisateur à l'applet $S^{3mAssess®}$.

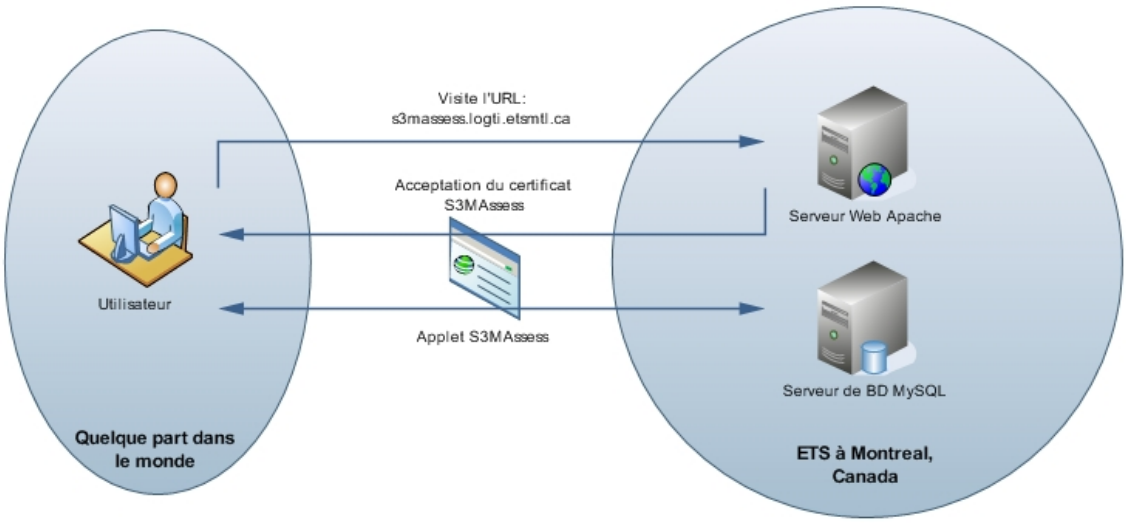


FIG. 5.6 – Déploiement de $S^{3mAssess®}$

L'utilisateur se connecte au serveur web Apache, où qu'il soit dans le monde, en entrant dans son navigateur Internet l'URL liée à $S^{3mAssess®}$. Le serveur web répond à cette requête en lui demandant d'accepter un certificat permettant à l'outil de lire et écrire sur le disque dur de l'utilisateur. S'il accepte, l'applet s'exécute dans la machine virtuelle Java de l'ordinateur de l'utilisateur. C'est ensuite via cette applet que l'utilisateur interagira avec le SGBD MySQL basé à Montréal.
Une des spécificités de MySQL est de pouvoir gérer plusieurs moteurs au sein d'une seule base. Chaque table peut utiliser un moteur différent au sein d'une base afin d'optimiser l'utilisation de chaque table. A ce propos, la base de données de $S^{3mAssess®}$ est passée du moteur MyISAM à InnoDB. InnoDB est devenu le moteur utilisé sur toutes les tables de la base de données car il gère les clés étrangères et les transactions.

5.3.5 Serveur HTTP Apache

Cette sous-section introduit la notion de serveur web, ou plus précisément, de serveur HTTP. Elle présente ensuite ce qu'est un serveur Apache.

5.3.5.1 Serveur Web et serveur HTTP

Un serveur HTTP est un logiciel servant des requêtes respectant le protocole de communication client-serveur HyperText Transfer Protocol (HTTP), qui a été développé pour le World Wide Web.
Un ordinateur sur lequel fonctionne un serveur HTTP est appelé serveur web. Le terme "serveur web" peut aussi désigner le serveur HTTP (le logiciel) lui-même. Les deux termes sont utilisés pour le logiciel car le protocole HTTP a été développé pour le web et les pages web sont en pratique toujours servies avec ce protocole. C'est le cas de $S^{3mAssess®}$ car lorsque l'utilisateur entre l'URL d'accès à l'applet, il se connecte en fait à une page web.

5.3.5.2 Apache

Apache est un logiciel de serveur HTTP. Lorsque que l'utilisateur entre l'URL d'accès à l'applet $S^{3mAssess®}$, son navigateur communique avec un serveur HTTP Apache. Cet exemple est illustré à la figure 5.6. Apache est le serveur HTTP le plus populaire du web et c'est à lui qu'appartient la responsabilité de répondre aux requêtes HTTP provenant des clients distants.

5.3.6 Accès à distance

Cette sous-section présente tout d'abord la manière qui a été utilisée pour continuer à maintenir de manière sécurisée, à partir de la Belgique, l'outil $S^{3mAssess®}$ disponible à partir d'un serveur situé au Canada. Ensuite, elle présente la manière sécurisée qui a été suivie pour permettre à des clients distants d'accéder à l'applet via son URL tout en préservant la sécurité des réseaux de l'ETS.

5.3.6.1 Accès distant VPN

A la suite du stage effectué à Montréal, un accès *VPN* (Virtual Private Network) a été utilisé afin de pouvoir continuer à maintenir l'outil à partir de l'extérieur des murs de l'ETS. Ainsi, les réseaux locaux de l'ETS, situés au Canada, sont séparés de la machine servant à maintenir $S^{3mAssess®}$ située en Belgique (voir fig.5.7). Entre les deux, la liaison se fait grâce au réseau public Internet. Dans ce cas de figure, la machine située en Belgique est appelée "client VPN" et la machine avec laquelle ce client communique est appelée "serveur d'accès distant".
Pour plus de détails concernant les accès distant VPN, voir la partie *11. Technologies employées, sous-section "Accès distant VPN"* en annexe.

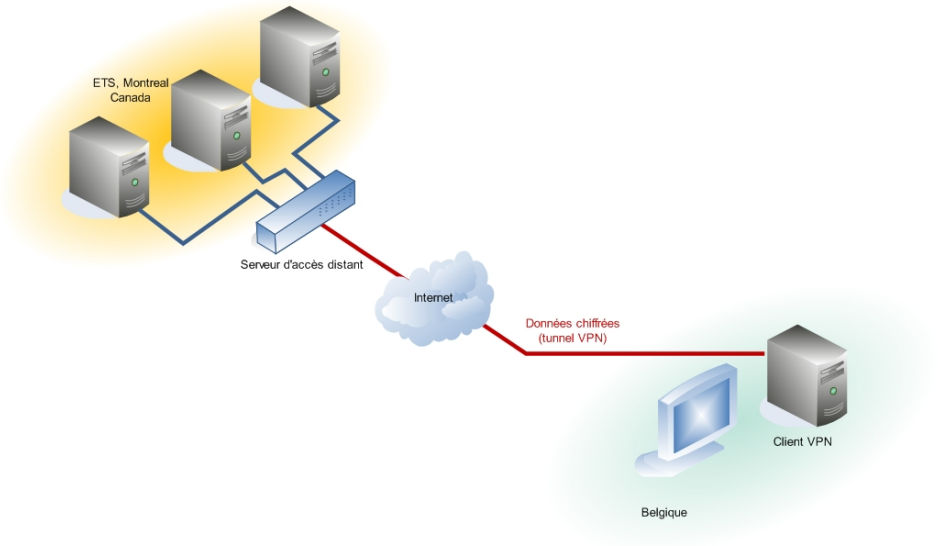


FIG. 5.7 – Réseau privé virtuel

5.3.6.2 Zone démilitarisée DMZ

Au sein de l'infrastructure de l'ETS se trouve une machine particulièrement utile pour $S^{3mAssess®}$ puisqu'elle sert à la fois de serveur HTTP (Apache) et de serveur de base de données (MySQL). En plus de la technique du VPN explicitée ci-dessus, cette machine est placée en *DMZ* afin de garantir le maximum de sécurité pour le réseau interne de l'ETS. Pour en savoir davantage, voir la partie *11. Technologies employées, sous-section "Zone démilitarisée DMZ"* en annexe.

5.4 Résumé

Ce chapitre avait pour but de présenter l'outil $S^{3mAssess®}$ sous plusieurs angles : son analyse des besoins, son analyse conceptuelle et tous les choix technologiques qui ont été fait pour l'encadrer et l'im-

plémenter. Ainsi, le Use Case diagram a permis de dégager les actions que peuvent entreprendre les divers utilisateurs de l'application et le schéma conceptuel présentait les concepts inhérents au contexte de l'outil. Ensuite deux diagrammes de classes présentaient sous différentes abstractions la structure de l'outil. Enfin, le diagramme d'activité présentait un exemple typique d'utilisation.
Pour terminer, tous les choix technologiques nécessaires à l'encadrement et à la mise en oeuvre de $S^{3mAssess®}$ ont été présentés et justifiés.
Le chapitre suivant va maintenant introduire la phase de maintenance de l'outil en présentant la démarche méthodologique établie et suivie. Ensuite, la réingénierie de la BD sera présentée en revenant sur les points d'intérêts majeurs de cette phase. Enfin, le chapitre détaillera la réingénierie de l'application avec notamment les présentations et critiques, du point de vue de la maintenance, de son architecture à haut niveau et de son architecture de persistance des données. Les évolutions apportées pour répondre aux questions de recherches de ce mémoire seront finalement détaillées.

Chapitre 6

Maintenance de $S^{3mAssess_{®}}$ par sa Réingénierie et son Evolution

$S^{3mAssess®}$ est un outil permettant d'évaluer la maturité d'un processus de maintenance du logiciel. Cependant, dans le contexte de ce projet, il fait lui-même l'objet d'une phase de maintenance. Ce chapitre présente ainsi un cas concret de maintenance et permet d'illustrer pratiquement les problèmes pouvant survenir lors de la maintenance d'un logiciel. Il présente aussi comment ces problèmes ont été traités.

Pour cela, ce chapitre se compose de deux sections bien distinctes. La première décrit la méthodologie et le cycle de vie de maintenance qui ont été mis en place afin de répondre aux problématiques et enjeux liés à ce projet. Le contexte initial de ce projet de maintenance de $S^{3mAssess®}$ sera d'abord établi. Ensuite, le cycle de vie mis en place pour répondre aux problématiques et enjeux du projet sera défini et justifié. Finalement, la méthodologie de maintenance sera présentée.
La deuxième section décrit quant à elle la réingénierie de l'application. A cette fin, cette section se subdivise en trois sous-sections. La première sous-section décrit la réingénierie de la base de données de $S^{3mAssess®}$. La seconde décrit la réingénierie de l'architecture de l'outil et la critique selon le point de vue de la maintenance. Finalement, la troisième sous-section décrit l'évolution de l'outil et la justifie en fonction des besoins réels des entreprises et du contexte du projet.

6.1 Méthodologie et cycle de vie de maintenance

6.1.1 Contexte initial

$S^{3mAssess®}$ a initialement été conçu en 2000 dans le langage C++ et était lié à une base de données constituée de fichiers textes. Il a ensuite progressivement migré vers le langage Java lié à une base de données MySQL, le but étant à terme de le porter sur le WEB. Ainsi, de la version construite en C++ en 2000 jusqu'à la version Java de 2004, plusieurs développeurs et mainteneurs, n'entrant pas réellement en contact, se sont succédés. La section 6.2 expliquera en quoi cela a t'il pu poser quelques problèmes pour la maintenance de l'outil appelée à être réalisée dans le cadre de ce projet.

Lors de l'acquisition de $S^{3mAssess®}$ au début de ce projet, celui-ci était dans un état expérimental inutilisable comme version opérationnelle. Il ne servait principalement qu'à illustrer les principes du modèle $S^{3m®}$ sur lequel il est basé. En effet, les interfaces graphiques permettaient de réaliser des captures d'écran à cette fin mais il était absolument impossible de réaliser complètement et correctement une quelconque évaluation.

Toutefois, la phase de migration vers le langage Java a été menée jusqu'à la création de l'architecture de l'outil. La découpe en classes, présentée dans la section 5.2 et détaillée en annexe, a été jugée suffisam-

ment correcte pour débuter son évolution à partir d'une base solide. En effet, la découpe en paquetages et en classes a été réalisée selon un bon niveau d'abstraction simplifiant une première approche de compréhension globale de l'architecture pour le mainteneur. C'est la raison pour laquelle la décision de conserver cette découpe a été prise. D'ailleurs, cette décision se justifie par le fait qu'un redéveloppement à partir de zéro d'un logiciel ne cadre pas réellement avec l'optique de la maintenance du logiciel présentée dans la première partie de ce mémoire (contexte d'un projet de maintenance de petite à moyenne taille).
Cependant, si la découpe initiale en classes a bien été jugée cohérente, le logiciel restait incapable d'atteindre l'objectif premier pour lequel il avait été conçu, c'est à dire *évaluer la maturité d'un processus de maintenance*. La phase de migration du langage C++ vers Java avait fait perdre l'intégrité de l'outil. Après analyse, il s'est avéré que ce problème venait du fait que :

- les classes Java ne communiquaient pas correctement entre-elles ;
- la base de données était dans un état incohérent ;
- il n'existait aucune contrainte d'intégrité sur la base de données ;
- l'application gérait mal les requêtes de base de données ;
- plusieurs erreurs étaient disséminées dans des fonctions maîtresses.

A l'acquisition de l'application pour maintenance, le but initial était de la faire évoluer en améliorant la fonctionnalité d'évaluer la maturité de la maintenance et en analysant et en concevant des exigences beaucoup plus complètes quant aux résultats d'évaluation. Si cet objectif initial a été réalisé, il a malheureusement été entravé suite à la découverte des problèmes cités précédemment. Une importante phase de réingénierie, à la fois de l'application et de la base de données, a donc dû être réalisée avant de pouvoir commencer une quelconque phase d'évolution du logiciel. Il était en effet primordial de concevoir de nouvelles exigences sur une base solide afin d'éviter le risque de voir tous les efforts d'évolution perdus.
L'aboutissement de la remise sur pied du logiciel a donné naissance à la première révision de maintenance du total des trois que compte $S^{3mAssess®}$. Ceci introduit donc la sous-section suivante quant à la méthodologie suivie pour mener à bien ce projet de maintenance.

6.1.2 Choix et définition du cycle de vie de maintenance de $S^{3mAssess®}$

Après une analyse des besoins, du contexte de maintenance et de l'état de l'outil à son acquisition, deux options de cycle de vie furent retenues pour la réalisation du projet de maintenance : par phases incréments/itérations ou en spiral à cycle incrémental et itératif. Si dans un premier temps le modèle en spiral fût choisi, c'est finalement le *cycle par phases* qui a été suivi en pratique. En effet, une des contraintes dans l'environnement empêchait l'utilisation du modèle en spiral car il est plus couteux en temps. Le problème provenait de l'irrégularité des contacts avec les clients qui apprenaient le modèle $S^{3m®}$ simultanément à la phase de maintenance de $S^{3mAssess®}$ et qui ont préféré effectuer rapidement une évaluation selon une méthode ad hoc qu'ils avaient développé suite à ce qu'ils avaient compris du modèle $S^{3m®}$. Une phase préalable de compréhension du modèle de maintenance, de développement d'une méthode adaptée et validée eut été plus propice pour les clients et aurait également permis la maintenance de $S^{3mAssess®}$ selon le modèle cyclique. Malheureusement, ces clients étaient pressés de recevoir des résultats et accordaient un temps limité à ces tâches empêchant toutes les validations requises par le modèle en spiral. Il faut en effet savoir qu'il s'agissait de multinationales qui mettaient l'accent sur la rapidité de la perception des résultats. Cependant, un minimum de préparation et d'analyse est nécessaire à la constitution d'un processus d'évaluation pertinent et efficace.
Ensuite, d'autres aspects étaient à considérer pour le choix du cycle de vie comme les difficultés techniques rencontrées par les clients pour installer l'outil sur leur machine personnelle, la limitation de l'équipe de maintenance à une seule personne limitant dans le temps certaines phases de cycle de vie, le retard dû à la non-prévision de l'acquisition d'un outil expérimental ne fonctionnant pas, etc.

Toutefois, le contexte du projet offrait l'avantage de la facilité d'interactions avec de nombreux intervenants. La maintenance de l'outil a principalement été réalisée pour trois intervenants : le Dr Alain April, la multinationale Texane Freescale et dans une moindre mesure pour IBM Australie. Des professeurs et experts du domaine ont également contribué à la validation de l'outil. Par la suite, une étude de cas appliquant l'outil et le modèle $S^{3m®}$ a été réalisée et explicitée dans le chapitre 7. Celle-ci fut réalisée pour la société Siemens en Belgique. Pour un résumé plus détaillé de tous les intervenants, voir la partie 9 *"Présentation des intervenants"* dans les annexes.
Il est à préciser enfin que divers modes de communications furent utilisés en fonction des distances géographiques : conférences téléphoniques, visioconférences, réunions et échanges de courriers électroniques.

Au final, l'intégration de ce contexte de projet a poussé à établir un cycle de vie de maintenance adapté. C'est ce que la section suivante va présenter.

6.1.2.1 Définition d'un cycle de vie de maintenance

Avant de contextualiser les problèmes et de décrire les solutions apportées, le schéma de la figure 6.1 définit le cycle de vie de maintenance suivi pour $S^{3mAssess®}$. L'expression *"cycle de vie de maintenance de* $S^{3mAssess®}$ *"* signifie "l'ensemble des phases de construction et d'opération employées au cours de la réalisation des trois révisions de $S^{3mAssess®}$".
A noter que l'expression *"processus de maintenance de* $S^{3mAssess®}$ *"* sera parfois employée dans ce mémoire de manière similaire, le terme "processus" étant à prendre dans ce cas d'une manière générale englobant l'ensemble des activités de maintenance de l'outil.

Afin de ne pas surcharger le schéma, les activités de la phase de construction d'une révision ne sont détaillées qu'une seule fois dans le bloc "Construire la Révision #2". Celles-ci restent les mêmes pour les trois révisions.

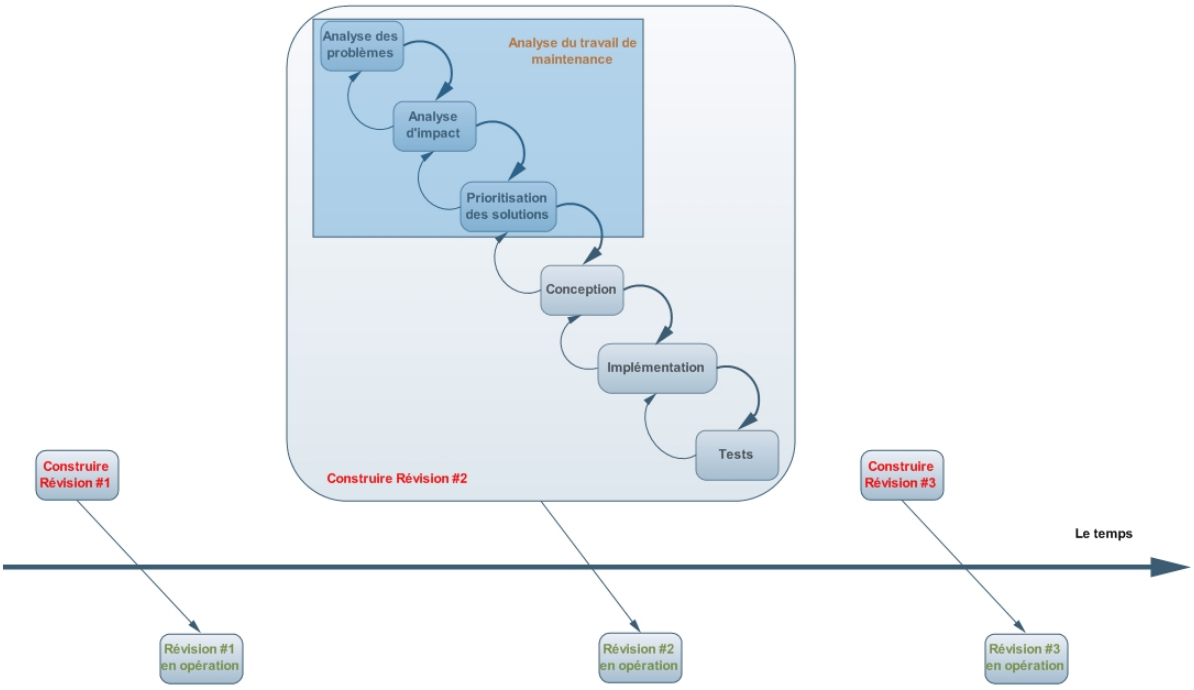


FIG. 6.1 – Cycle de vie de maintenance de $S^{3mAssess®}$

Le cycle de vie se compose de **trois révisions**. Une révision se décompose en une **phase de construc-**

tion et en une **phase d'opération**. Chaque phase de construction d'une révision est composée de **six activités** successives :

1. **Analyse des problèmes** : il s'agit d'analyser les feedbacks rendus par les clients à la suite de la phase d'opération de la révision précédente. Ensuite, il vient une analyse du code afin de déterminer d'où viennent concrètement les problèmes et quels en sont les impacts. Cette analyse du code peut être accompagnée d'une phase de rétro-ingénierie si la documentation existante ne couvre pas le problème en question.
2. **Analyse d'impact** : elle a pour buts d'établir un sommaire des requêtes de changements, d'établir une suggestion de prioritisation de ces requêtes, de prévoir quand la nouvelle révision pourra être livrée, d'évaluer les implications du changement proposé et ses effets sur le système ainsi que d'estimer la répartition de l'effort. Un gabarit d'analyse d'impact est donné en annexe, voir *Impact Analysis Template*, ainsi que l'ensemble des analyses d'impacts réalisées pour les trois révisions de $S^{3mAssess®}$, voir *Analyses d'impacts du logiciel* $S^{3mAssess®}$.
   Etant donné la limitation des ressources humaines de l'équipe de maintenance (réduite à une seule personne) et les contraintes de temps liées à la remise sur pied non prévue initialement de l'application, le nombre de nouvelles fonctionnalités à apporter et la volonté des clients de pratiquer rapidement une évaluation, cette activité fut partiellement réalisée de manière informelle. En effet, par révision, seules les cinq requêtes de maintenance les plus critiques furent l'objet d'une analyse d'impact formelle.
3. **Prioritisation des solutions à apporter** : suite à l'analyse d'impact, des décisions doivent être prises quant aux priorités de réalisation de chaque changement. Sur base des requêtes de modification proposées par les clients, de l'analyse d'impact et du niveau de criticité des problèmes, il s'agit de déterminer quels seront les changements à effectuer dans la prochaine révision et quels seront ceux à planifier dans une révision ultérieure. Ces décisions ont été prises de commun accord avec les clients et avec le maître de stage, Dr Alain April, afin de respecter les ententes de service en vigueur. Une discussion est donc entamée sur base des priorités définies dans l'analyse d'impact.
4. **Conception** : création des documents et schémas de conception des solutions.
5. **Implémentation** : codage des solutions spécifiées à l'étape précédente.
6. **Tests** : tester les solutions implémentées. Ces tests furent informels via des tests de cas particuliers et via les soumissions des futures révisions au Dr Alain April.

Il est à préciser que s'il est possible de remonter d'un niveau dans ces six activités successives de construction d'une révision, il est très rare de remonter davantage d'étapes. Ceci peut paraître rigide mais cela vient du fait qu'une grande importance est attachée aux trois premières activités du processus car elles analysent et définissent le travail de maintenance à effectuer en intégrant de nombreux critères comme le respect des ententes de services et l'analyse d'impact. En brisant ces critères, le projet de maintenance peut être mis à mal et le risque serait alors grand de ne jamais pouvoir terminer une révision. Il vaut mieux dans ce cas terminer le travail qui a été analysé en amont de la phase de construction et planifier les éventuels changements pour une révision future. En effet, après la mise en production de la révision, ces changements seraient alors naturellement considérés dans les trois premières étapes de la phase de construction de la révision suivante.

Pour terminer, la révision construite selon les étapes définies ci-dessus est mise en opération et donc rendue accessible au client. Le client peut dès lors déjà utiliser cette version, ce qui lui permet de devoir attendre moins de temps avant de pouvoir poursuivre ses propres activités d'évaluation de maturité. En contrepartie, l'équipe de maintenance s'attend éventuellement à recevoir de nouvelles Requêtes de Modification (RM) de la part du client. Ces requêtes de maintenance s'ajoutent alors aux éventuelles RM qui avaient été replanifiées à une révision ultérieure lors de l'étape *Prioritisation des solutions à apporter*.

6.1.2.2 Le cycle de vie en pratique et ses difficultés

Le cycle de vie présenté dans la sous-section précédente a été élaboré et adapté pragmatiquement. Néanmoins, certains problèmes ponctuels liés à l'environnement du projet de maintenance ont surgi impliquant quelques précisions à apporter. Certaines difficultés ont été rencontrées comme l'obligation de réaliser de manière partiellement informelle les analyses d'impact. Il est utile de préciser ici d'où venait cette nécessité de réalisation informelle et quels ont été les problèmes rencontrés lors la phase d'Opération :

1. **Etape des tests** : au début du projet de maintenance, c'est à dire après l'acquisition de $S^{3mAssess®}$, d'importantes défaillances dans le logiciel ont été découvertes. Ensuite, lors de l'étape de rétro-ingénierie, une architecture relativement complexe de persistance des données a également été découverte (voir la section 6.2.2.2). A cela, des difficultés supplémentaires liées à une documentation très peu fournie, à la limitation à une seule personne pour réaliser les requêtes de maintenance et à l'absence de contact avec les développeurs de l'application se sont révélés peu à peu dans l'étape d'analyse initiale du projet de maintenance.
   L'ensemble de ces impondérables ont retardé de plusieurs semaines le développement des nouvelles fonctionnalités pour $S^{3mAssess®}$. Il a notamment fallu créer une première révision permettant d'obtenir une base solide avant de commencer tout nouveau développement. Or, l'entente de service visait initialement uniquement au développement de nouvelles fonctionnalités et recherches pour l'amélioration du processus d'évaluation de maturité effectué avec $S^{3mAssess®}$. C'est pourquoi, les tests ont dû être réalisés informellement grâce à des tests de cas particuliers. Ensuite, des démonstrations des révisions furent proposées au maître de stage afin de s'assurer que les solutions apportées rencontraient bien les attentes. Ce n'est qu'après ces deux étapes qu'une révision fut proposée au client. Ce dernier exécutait alors la révision et en rendait un feedback.
   Ces phases informelles de tests et de validation furent suffisantes pour les deux premières révisions car la révision une visait uniquement à la remise sur pied de l'outil en se contentant de respecter les exigences originelles. Celles-ci avaient été obtenues grâce à la rétro-ingénierie de l'outil, grâce aux rencontres avec le maître de stage et grâce à la compréhension du domaine d'application via la lecture de documents et de livres pertinents. De plus, la première révision ne donna pas encore lieu à une mise en opération chez le client car de nouveaux problèmes, présentés dans le point suivant, ont surgis. A la place, une présentation à l'aide de capture d'écrans fut proposée.

   Pour la seconde révision, les problèmes de mise en opération furent résolus et les clients eurent la possibilité de tester directement l'outil. La nécessité de mise en opération fut cette fois beaucoup plus importante puisque de nouvelles fonctionnalités spécifiques avaient été développées sur base des feedbacks rendus à l'issue de la présentation de la première révision. Ces premiers feedbacks donnèrent lieu à de nouvelles exigences.

   Enfin, il faut préciser que le modèle de cycle de vie par phases donne naissance à des versions intermédiaires du produit logiciel. Ces versions n'ont pas pour but d'être exemptes de tout défaut mais plutôt de livrer rapidement un produit *d'un niveau raisonnable* au client. Cette contrainte de rapidité de livraison faisait en effet partie du projet de maintenance de $S^{3mAssess®}$.
   A noter que le terme "raisonnable" veut dire "qui permet d'effectuer correctement les opérations demandées sans défaut majeur". Au final, il s'est avéré que les clients n'avaient pas relevé d'erreur, laissant conclure à un niveau correct de réalisation des requêtes de maintenance.

2. **Etape d'Opération** : La mise en opération posa problème lors des deux premières révisions :

   - **Révision 1** : A l'issue de cette révision, il était prévu de fournir au client un code source compilé (sous forme de fichier jar à exécuter), les outils externes nécessaires à l'exécution de $S^{3mAssess®}$ ainsi qu'un manuel d'installation et de configuration du système. Cependant, il s'avéra que les

clients furent dans l'incapacité d'atteindre cet objectif par cause de difficultés techniques. Dans le même temps, il n'était pas possible de faire tourner l'outil sur un serveur de l'ETS endéans un délai raisonnable. La décision fut donc prise de présenter oralement l'application et d'en fournir un document expliquant ses principes et reprenant ses principaux écrans (voir *Guide d'introduction au logiciel* $S^{3mAssess®}$ en annexe).
Cette solution ne fut pas trop gênante car cette révision permettait surtout de remettre sur pied l'application et fut donc plutôt une révision en interne entre le mainteneur et le maître de stage. Le client ne devait pas être beaucoup plus sollicité qu'il ne l'a été.

- **Révision 2** : La mise en opération de cette révision était beaucoup plus importante car des fonctionnalités clientes furent cette fois développées. La décision fut de faire tourner le logiciel sur un serveur de l'ETS. L'équipe technique de l'école Montréalaise fut donc sollicitée afin d'obtenir un accès aux ressources informatiques. Il y eu cette fois un problème de délai dû à la structure bureaucratique de l'ETS. En effet, un mois et demi ont dû s'écouler avant d'obtenir les derniers droits d'accès requis au déploiement du logiciel sur l'infrastructure de l'ETS. Le dernier jour du mois et demi d'attente, une solution alternative fut de créer un serveur temporaire en Belgique afin d'héberger le logiciel en attendant d'obtenir les accès sur les ressources informatiques de l'ETS. Toutefois, hasard du calendrier, les derniers droits d'accès furent octroyés ce même jour. $S^{3mAssess®}$ fut donc directement déployé à l'ETS.

6.1.2.3 Critique du cycle de vie choisi

Avec le modèle de cycle de vie par phases, l'utilisateur peut rapidement utiliser les premières révisions/itérations, ce qui est intéressant dans le cas de ce projet étant donné les contraintes de temps fixées par les clients et par le contexte du projet. De cette manière, il est possible de raccourcir le temps d'un cycle afin d'obtenir une réaction plus rapide de l'utilisateur. L'idée fut donc de décomposer le logiciel en composantes par itérations dans un premier temps, c'est à dire selon un prototypage horizontal afin d'améliorer, enrichir et raffiner les fonctionnalités déjà présentes. Ce fut principalement le cas de la première révision et en partie pour la seconde révision.
Pour la troisième révision mais également en partie pour la seconde, la décomposition de la tâche fut plutôt par incréments, selon un prototypage vertical, où l'accent fut mis sur l'ajout des fonctionnalités. Ces décompositions définies à l'avance permettent de limiter dans le temps un cycle de production d'une révision.
Il est intéressant à présent de dresser une critique de l'application de ce modèle dans ce cas de maintenance :

Avantages	**Désavantages**
Développement en révisions moins complexes	Risque de mélanger raffinement et maintenance corrective
Les intégrations sont progressives	Nécessite une architecture du logiciel bien particulière
Permet un apprentissage progressif	Nécessite une planification bien documentée

TAB. 6.1 – Critique de l'application du modèle par phases pour la maintenance de $S^{3mAssess®}$

Le processus décomposait la tâche initiale complexe de maintenir l'outil en trois sous-tâches bien spécifiques, donnant chacune lieu à une révision, à la suite desquelles le client devait rendre un feedback. De plus, la nécessité d'une analyse préalable à la construction d'une révision, du nombre assez important d'étapes et la gestion de l'environnement externe aux tâches de maintenance poussait à gérer chaque révision comme un mini-projet. De ce point de vue, le cycle de vie par phases peut-être considéré comme

un ensemble de trois projets de petites à moyennes tailles, correspondant aux trois révisions de $S^{3mAssess®}$. De manière plus générale, en considérant une équipe de maintenance se composant d'un grand nombre de personnes et où la décision est prise de réaliser des révisions demandant plusieurs mois de travail, les révisions devraient être gérées selon les techniques de gestion de projet. Dans ce cas, il s'agit de **projets** de maintenance dont la complexité est comparable aux projets de développement logiciel à partir de zéro.

6.1.3 Méthodologie de maintenance de $S^{3mAssess®}$

La maintenance s'est réalisée selon trois révisions :
- Il s'agissait dans un premier temps de remettre en marche l'outil et d'apporter des corrections de haute et moyenne priorités. Ce premier objectif fut l'objet de la première révision et devait permettre de commencer à implémenter les nouvelles fonctionnalités sur une base solide et cohérente.
- Ensuite, de nouvelles fonctionnalités et corrections de priorités inférieures à celles de la révision précédente furent l'objet de la deuxième révision.
- Enfin, les dernières nouvelles fonctionnalités, RM correctives et perfectives, donnèrent lieu à la troisième révision.

Afin d'y voir plus clair, voici l'énumération des activités maîtresses de ces trois révisions :

1. **Révision 1** : Comprendre et avoir une vue globale du logiciel, comprendre la base de données, nettoyer les données incohérentes et ajouter des contraintes d'intégrité, résoudre les défaillances empêchant un fonctionnement correcte de l'application, maintenance préventive, maintenance corrective, correction d'erreurs provenant d'une mauvaise compréhension du domaine d'application, ajout de rétroaction pour l'utilisateur, encodage des 196 pratiques des trois premiers niveaux du modèle $S^{3m®}$, conception de documents de conception et du guide utilisateur.
2. **Révision 2** : Conception et mise à jour de documents de conception et d'aide pour l'utilisateur, amélioration des processus d'évaluation et d'analyse, réingénierie de la base de données, ajout de rétroaction, maintenance corrective, maintenance perfective, maintenance préventive, maintenance adaptative, amélioration de l'interface, augmentation de la stabilité.
3. **Révision 3** : Maintenance perfective et mise à jour des documents de conception et utilisateurs.

6.2 Réingénierie de $S^{3mAssess®}$

Cette section présente les principales requêtes de modification ayant fait l'objet d'une réingénierie. La phase de réingénierie de la base de données sera d'abord présentée puisqu'il convient généralement de commencer par là lors de la maintenance d'une application reposant sur un tel système de gestion de données. En effet, une application ne pourra jamais donner de résultats corrects en s'appuyant sur une base de données non fiable. Les diverses techniques de réingénierie qui ont été utilisées seront donc présentées ainsi que les solutions trouvées pour faire face aux problèmes qui ont surgis.
Ensuite, la réalisation de la maintenance de l'outil impliquait également une phase de réingénierie de l'application en elle-même. A partir du moment où celle-ci pouvait reposer sur une base de données ayant retrouvé un certain niveau de cohérence et de fiabilité, la phase de compréhension du logiciel via des techniques de rétro-ingénierie pouvait commencer. Il suivit alors son évolution grâce au bon niveau de compréhension acquis préalablement.

6.2.1 Réingénierie de la Base de Données

Cette sous-section vise à présenter comment il a été possible de rendre une base de données cohérente et fiable malgré des problèmes d'absence de phase de transition entre l'équipe de développement et l'équipe de maintenance, de quasi inexistence de documentation technique à jour, d'inaccessibilité des concepteurs, de présence en BD de données *partiellement* incohérentes et d'absence de contrainte d'intégrité.

Cette question a un enjeu particulièrement important dans le contexte de la maintenance étant donné qu'une grande majorité des logiciels actuels interagissent avec un système de persistance des données, en particulier avec une base de données relationnelle. Bon nombre d'organisations de maintenance faisant l'acquisition tardive d'un logiciel à maintenir se voient confrontées au problème de devoir maintenir correctement un logiciel en ayant des informations limitées, qui plus est concernant la structure de persistance des données.
Selon le moment d'implication de l'équipe de maintenance dans le processus de développement d'un logiciel (voir le chapitre 1) et des niveaux de maturité des processus de développement et de maintenance, la tâche sera plus ou moins ardue. Dans le cas idéal, voir utopique, où l'équipe de développement documente toute particularité du logiciel, où les processus de développement et de maintenance atteignent tous un niveau de maturité *en optimisation* (voir section 2.3), où l'équipe de maintenance est impliquée dès la conception du logiciel et est confondue avec l'équipe de développement, la phase finale de maintenance sera préparée au mieux. La connaissance du logiciel est au top et donc, les risques liés à des erreurs dues à une mauvaise compréhension de l'application ou liés à des retards très importants dans la réponse aux RMs (par cause de long temps de formation à l'application) seront fortement réduits. Dans ce cas, les problèmes évoqués ci-dessus ont peu de chance d'éclore.
Toutefois, la grande majorité des entreprises actuelles ne se situent pas à un niveau de maturité si élevé. Ainsi, il est possible de rencontrer toute sorte d'organisation ayant chacune leur façon d'opérer et donc étant à des stades de maturité différents. Dans ces cas, une partie, tous ou même encore d'autres problèmes que ceux précités surgissent.

Cette section montrera, sur base d'un cas concret, comment les problèmes ont pu être traités. Pour cela, des idées de solution ont été avancées. Il fallait avant tout comprendre la BD relationnelle, venant d'être acquise, en en retirant un maximum d'informations.
La figure 6.2 résume les deux processus de base de la rétro-ingénierie : l'Extraction des Structures de Données et la Conceptualisation des SD.

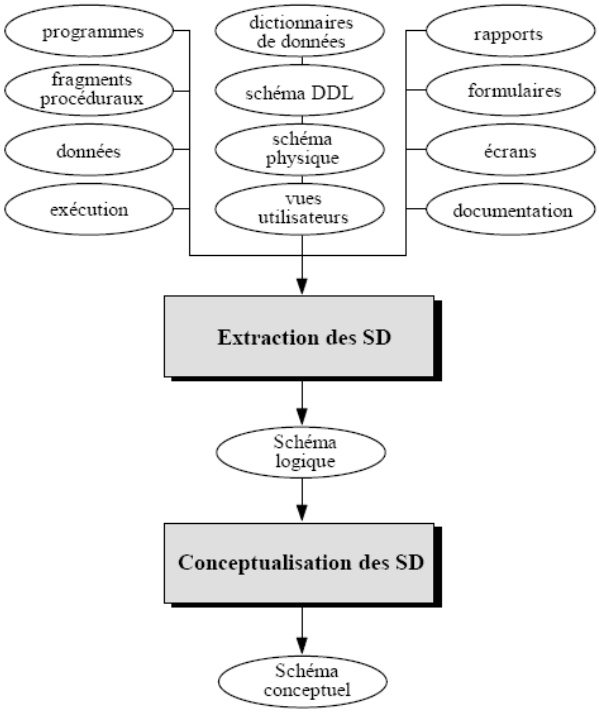


FIG. 6.2 – Les deux processus de base de la rétro-ingénierie (repris du cours [Hainaut, 2002])

Les deux processus de la figure 6.2 se décrivent comme suit :

- **L'extraction des Structures de Données** : a pour but de reconstruire un schéma logique complet dans lequel toutes les structures explicites et implicites et les propriétés sont documentées. La source principale des problèmes vient du fait que beaucoup de constructions et propriétés sont implicites, c'est à dire qu'elles ne sont pas explicitement déclarées, et sont contrôlées et gérées via des sections, dites "procédurales", des programmes. Retrouver ces structures implicites demande des analyses de codes DDL, d'extraire des structures de données explicites et d'utiliser des techniques d'élicitation de structures de données.
- **La conceptualisation des SD** : essaie de spécifier les structures sémantiques comme pour un schéma conceptuel. Alors que certaines constructions sont assez faciles à interpréter (par ex., une clé étrangère est l'implémentation d'une relation un-à-plusieurs), d'autres posent des problèmes complexes dû à l'utilisation d'implémentations difficiles et de techniques d'optimisation.

Alors que le premier processus a été réalisé au fil des sections suivantes, le deuxième processus est repris dans une section (6.2.1.6) à part entière.

Finalement, en rétro-ingénierie, l'une des activités fondamentales est de confirmer des hypothèses sur base de preuves et d'observations tangibles. Voici une liste des activités d'investigation :
- réunir la documentation (partielle et obsolète) disponible et l'analyser ;
- analyser les données pertinentes en BD ;
- analyser les noms ;
- se documenter sur le domaine d'application : ses contraintes, ses caractéristiques et son fonctionnement ;
- extraire des schémas des fichiers/bases de données à partir des codes DDL et du code source de l'application utilisant les données ;
- analyser le flux de données du logiciel ;
- exécuter le programme ;
- analyser l'utilisation de patterns ;
- conceptualiser les structures de données et les normaliser ;
- essayer de comprendre les structures et significations des schémas obtenus.

Ces idées vont être illustrées en pratique avec la présentation des principales RMs portant sur la BD.

6.2.1.1 Rétro-ingénierie de la BD et rectification des données d'initialisation

Un certain nombre de données étaient déjà présentes lors de l'acquisition de l'outil pour sa maintenance et son évolution. Malheureusement, étant donné l'absence de contrainte d'intégrité et le manque de cohérence des fonctions applicatives traitant les données, ces données s'avéraient partiellement incohérentes. Le but n'était pas de récupérer les données d'évaluation déjà encodées, étant donné que l'outil n'avait jamais été réellement confronté à des données cruciales des clients, mais de comprendre la signification de certaines colonnes en BD. En effet, certains noms de colonnes n'étant pas suffisamment explicites et en l'absence de clés étrangères, il était difficile de comprendre les données que devaient contenir certaines colonnes. Afin de fixer directement les idées sur l'état de cette BD à l'acquisition, voir la figure 6.3.

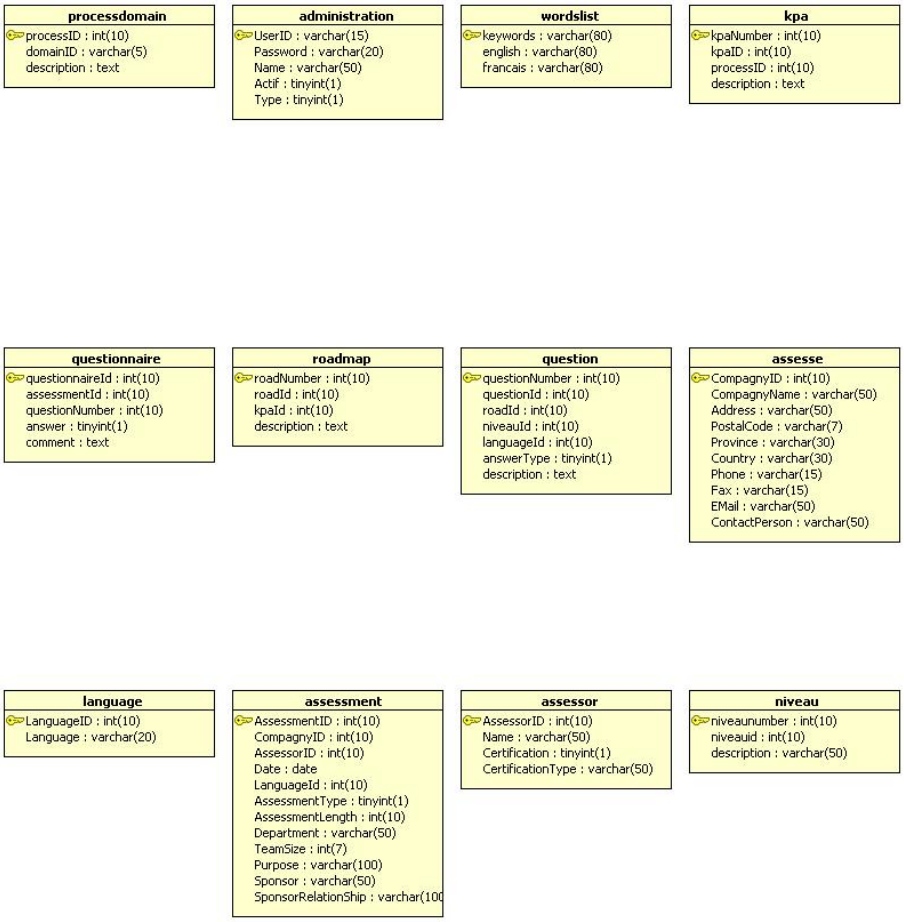


FIG. 6.3 – Schéma relationnel de la BD à son acquisition

Cette figure a pu être obtenue grâce à un outil de gestion de bases de données en extrayant un schéma relationnel sur base du code DDL de la BD. Le schéma permet d'avoir une vue globale sur la définition des tables et de voir principalement que les seules contraintes d'intégrité étaient des identifiants techniques. Une mise en application du code DDL sur un SGBD a permis de confirmer facilement cette absence de contrainte supplémentaire, comme des identifiants secondaires par exemple. En effet, nombre de SGBD offrent des options graphiques permettant de gérer simplement ce genre de contraintes. Sinon, il est évidemment possible de vérifier leur présence en passant en revue le code DDL.

L'un des deux buts de cette première RM était donc de corriger les données initiales permettant l'exécution de $S^{3mAssess®}$. Certaines données doivent en effet être constamment en BD, même lors de son initialisation. Ces sont les noms et identifiants des domaines $S^{3m®}$, les KPAs, les pratiques (reprises dans la table "Question"), les niveaux, etc.

Au final, cette étape d'analyse du code DDL, des données, du code applicatif et du domaine d'application a permis de comprendre les sémantiques des tables, des colonnes, des valeurs, de détecter des structures implicites[1] et des propriétés sur les valeurs à insérer, comme des intervalles par exemple.
Cela a aussi mené à faire une découverte assez étonnante : seules quelques pratiques du modèle $S^{3m®}$ avaient été encodées et seulement en français. Ceci était donc la preuve finale que l'outil n'avait jamais eu d'autre finalité que d'aider à la compréhension des principes du modèle, via par exemple des captures

[1]Leur existence dérive de quatre pratiques de conception, de programmation et d'exploitation : structures cachées, structures non-déclaratives, propriétés environnementales et spécifications perdues (voir le cours [Hainaut, 2002], chapitre *Rétro-ingénierie des données* pour plus de détails)

d'écran dans des présentations, sans toutefois être utilisable. L'encodage des pratiques des niveaux de maturité $S^{3m®}$ 0 à 2 allait donc passer en première priorité et devait être réalisé dès la première révision de l'outil.
Le résultat de ce travail d'investigation donna lieu à la création de documents techniques explicitant cette nouvelle compréhension acquise (voir notamment le schéma conceptuel de la sous-section 5.1.2) et à la création de contraintes d'intégrité référentielles (voir le schéma relationnel issu de la première révision, figure 6.4).

6.2.1.2 Ajout de clés étrangères

L'aboutissement de l'étape précédente permit de comprendre les structures de données en présence, une bonne partie de leur sémantique et ainsi de créer des contraintes d'intégrité référentielles. Implémenter des contraintes de ce type directement dans le SGBD offre les avantages non négligeables de garantir une cohérence des données, et donc des résultats de l'application, d'augmenter la maintenabilité de la BD et de l'application en facilitant sa compréhension et d'inclure de manière efficace et explicite les propriétés des données dans le code DDL.

L'existence de certaines colonnes faisait penser à l'existence de clés étrangères même si celles-ci n'étaient pourtant pas présentes explicitement dans le code de définition des tables. En analysant les données présentes, certains noms de colonnes ainsi que leur type, l'exécution de l'outil et en comprenant le fonctionnement du modèle $S^{3m®}$, il a été possible de recouper des sémantiques communes, de confirmer certaines hypothèses et donc d'augmenter la compréhension du schéma.
Ce travail d'investigation mena à la modification de la BD en incluant explicitement les clés étrangères dans la définition des tables. De cette manière, tout mainteneur pourra voir directement et facilement ces contraintes et la bonne application de ces dernières ne sera plus à la merci d'une erreur du programmeur. L'intégrité de la BD en est plus assurée. Le schéma de la figure 6.4 illustre le travail réalisé à l'issue de la première révision de l'outil.

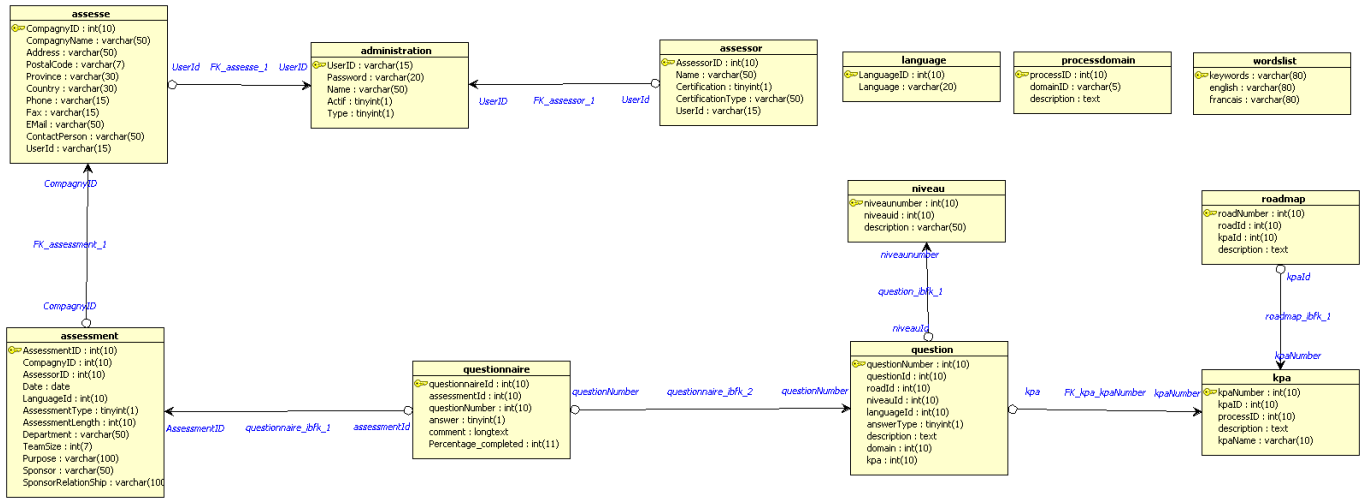


FIG. 6.4 – Schéma relationnel de la BD à l'issue de la première révision

6.2.1.3 Encodage de 196 pratiques

Suite aux deux étapes précédentes, il était possible d'encoder correctement les 196 pratiques que comptent les trois premiers niveaux de maturité $S^{3m®}$. La sémantique des tables et des colonnes était acquise et les clés étrangères étaient implémentées dans le SGBD. L'encodage pouvait donc suivre sans crainte d'encodage de mauvaises valeurs ou clés.

L'importance de cette étape n'était plus de comprendre et de protéger l'intégrité de la BD mais de rendre ses valeurs cohérentes afin de permettre une première utilisation correcte de l'outil $S^{3mAssess®}$.
Suite à cette étape, l'ajout de contraintes d'intégrités supplémentaires pouvait commencer sur une base saine avec le début de la deuxième révision de l'outil. C'est ce qui est présenté à partir de la section suivante.

6.2.1.4 Ajout de nouvelles contraintes d'intégrité

D'autres contraintes d'intégrité que celles ajoutées jusqu'ici ont dû être ajoutées afin de préserver l'intégrité de la base de données avec les nouvelles fonctionnalités implémentées dans l'outil. Certaines sont également nécessaires pour garantir les bonnes valeurs de retour de certaines requêtes. Il s'agit principalement d'identifiants secondaires. Ces identifiants secondaires sont :

- {CompagnyName, UserId} pour la table *assesse*
- {Name} pour la table *assessor*
- {assessmentId, questionNumber} pour la table *questionnaire*

En prenant l'exemple de la table questionnaire, l'ajout de l'identifiant secondaire {assessmentId, questionNumber} a permis de s'assurer qu'une ligne de cette table, c'est à dire l'évaluation d'une pratique par l'utilisateur, correspond à un numéro de question pour une évaluation bien déterminée. Autrement dit, il est impossible d'avoir deux réponses à une même question au sein d'une même évaluation. Cette contrainte assure le déterminisme de l'application.

Ensuite, d'autres propriétés ont été ajoutées comme une clé étrangère liant la table assessment à la table assesse ou le changement de type de la colonne "comment" de la table questionnaire. Ceci afin de répondre aux nouvelles exigences de l'application et pour corriger des défaillances.
En particulier, deux autres clés étrangères ont été ajoutées : celles liant la table administration aux tables assessor et assesse via la colonne UserId. Cet exemple illustre le développement en BD d'une nouvelle exigence. En effet, lors de cette deuxième révision, la décision fut prise de rendre l'outil accessible à tous via Internet. Cela a posé un nouveau problème quant à la confidentialité des données d'évaluation entre différents utilisateurs. Il est peu concevable que des entreprises concurrentes puissent avoir accès aux données internes d'autres entreprises. C'est pourquoi, les compagnies évaluées (celles de la table assesse) ne le sont que pour le compte d'un utilisateur enregistré. De cette manière, seul l'utilisateur ayant évalué une organisation peut effectivement avoir accès à ses résultats. Dans le même ordre d'idée, un évaluateur encodé ne l'est que pour le compte d'un utilisateur enregistré.
La figure 6.5 illustre tout ce qui vient d'être expliqué. Il s'agit du schéma relationnel issu de la troisième et dernière révision.

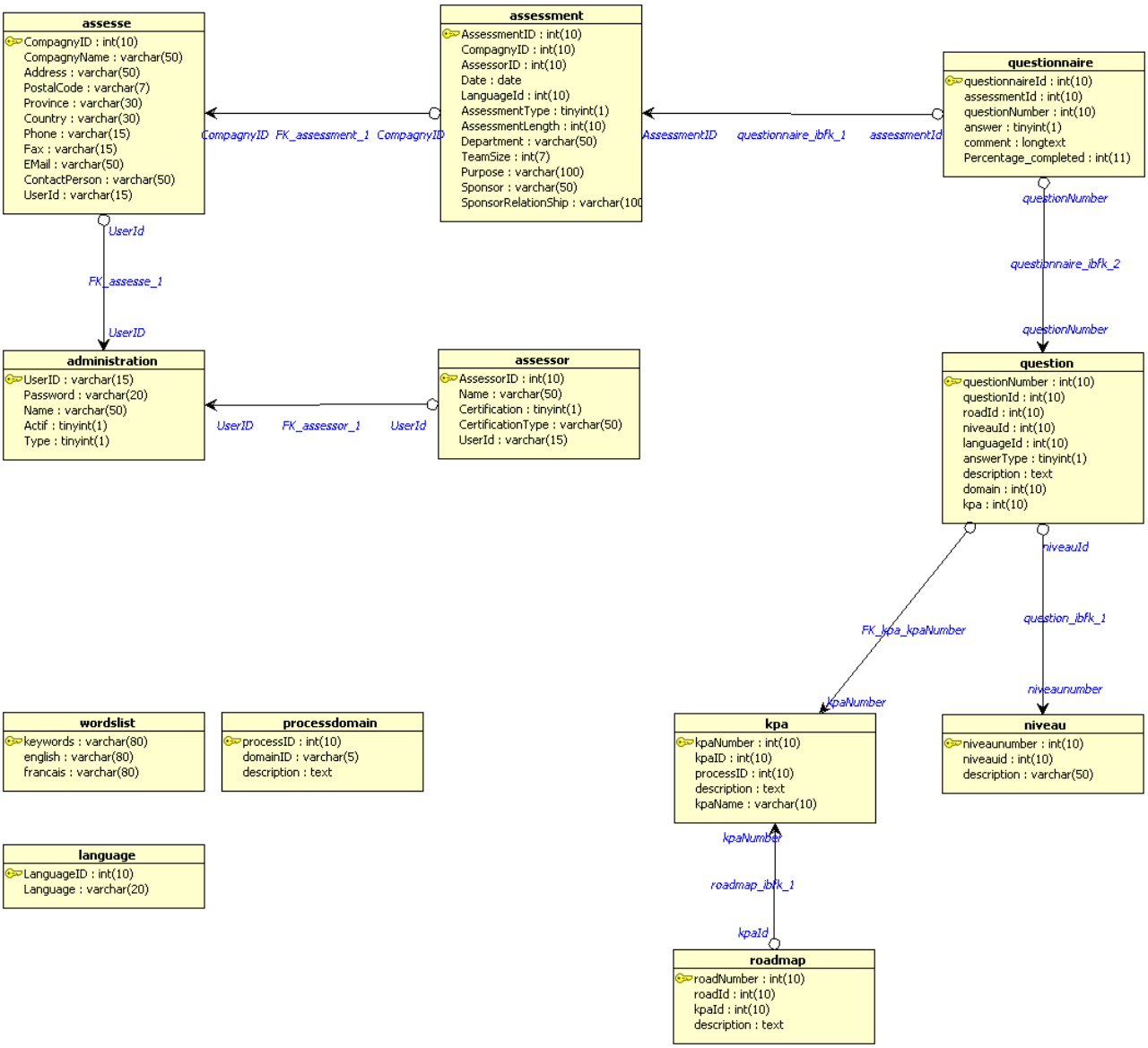


FIG. 6.5 – Schéma relationnel final de la BD

6.2.1.5 Ajout d'indices supplémentaires pour la performance

Les indices sont utilisés pour trouver très rapidement des lignes de résultat avec une valeur spécifique. Sans index, le SGBD doit lire successivement toutes les lignes, et à chaque fois, faire les comparaisons nécessaires pour extraire un résultat pertinent. Plus la table contient de lignes, plus c'est coûteux. Si la table dispose d'un index pour les colonnes utilisées, le SGBD peut alors trouver rapidement les positions des lignes dans le fichier de données, sans avoir à fouiller toute la table.
Concrètement, ces indices sont :

- {niveauId} pour la table *question*
- {assessmentId} pour la table *questionnaire*
- {questionNumber} pour la table *questionnaire*

Ces indices supplémentaires sont ajoutés pour des tables visant à contenir beaucoup d'enregistrements et pour des colonnes souvent reprises dans des clauses *where*. Ces indices s'ajoutent bien entendu aux indices

des clés identifiantes et des clés étrangères. Il s'agit d'une première optimisation de la structure de la BD à long terme.

6.2.1.6 Conceptualisation finale des structures de données

Une dernière sous-section pour présenter un dernier travail particulièrement intéressant concernant la BD. Comme cela a été expliqué précédemment, nombres de changements eurent lieu au cours des trois révisions. Ces changements ont modifié la BD et donc le schéma relationnel. Il y eut par conséquent recourt à une conceptualisation progressive au fur et à mesure des évolutions. Le résultat final de cette progression a déjà été donné dans la section 5.1.2, seule une version réduite sera donc reproduite ici.

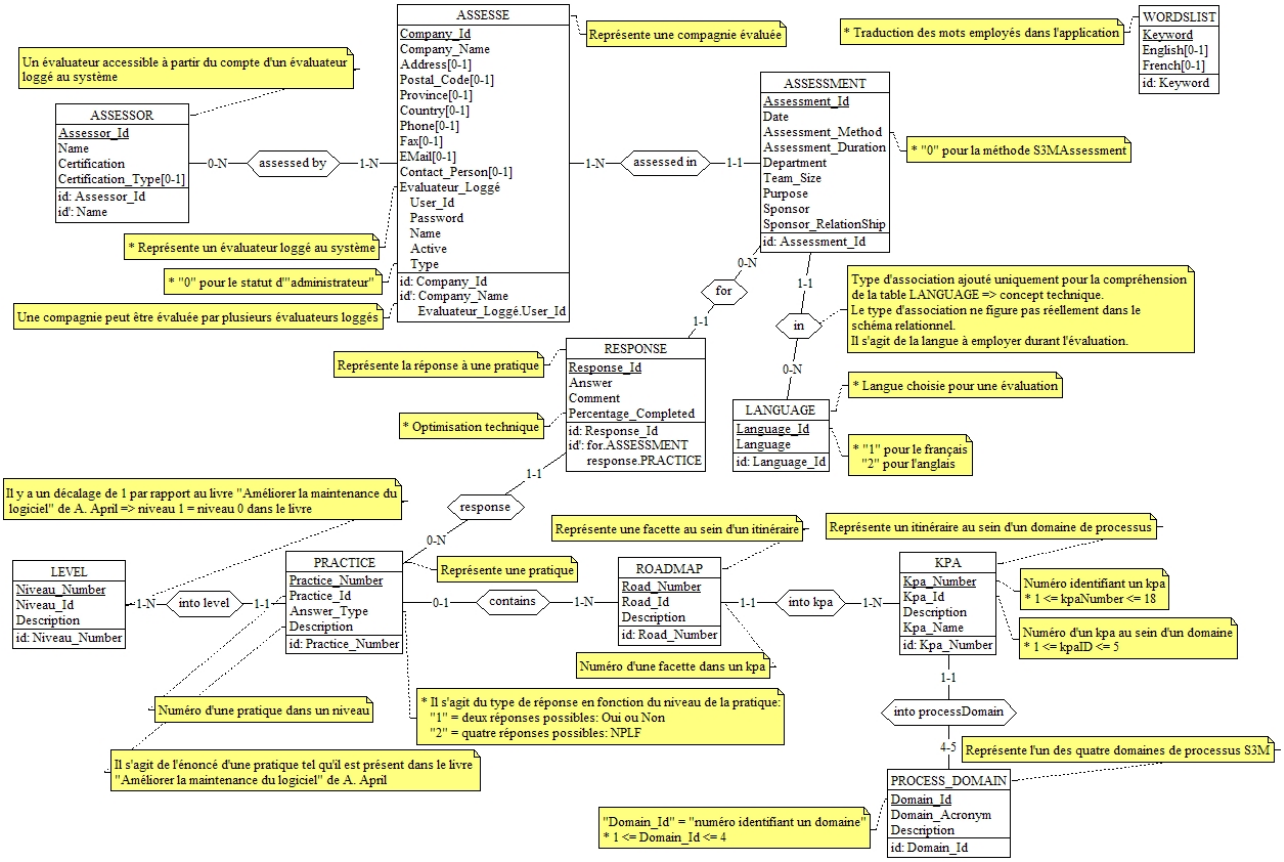


FIG. 6.6 – Schéma conceptuel de $S^{3mAssess®}$

Ce schéma final résulte des processus d'extraction des SD et de leur conceptualisation (voir le schéma repris à la figure 6.2). Le premier processus comprend des activités d'analyse de codes DDL, d'intégration physique, de raffinage du schéma et de nettoyage du schéma. La tâche la plus critique fut sans doute le raffinement du schéma étant donné qu'elle consistait à chercher des preuves de constructions implicites ou perdues à partir de diverses sources d'informations. Les informations se trouvaient dans des sections procédurales de l'outil, dans les procédures graphiques, dans des formulaires et rapports obsolètes, dans les valeurs en BD, en expérimentant, via des interviews (du maître de stage) et via la lecture de documents sur le domaine d'application (permettant de comprendre la logique du modèle $S^{3m®}$).

Le deuxième processus intéressant particulièrement cette section est donc la conceptualisation et suit la phase d'extraction venant d'être résumée. La figure 6.7 détaille cette phase.

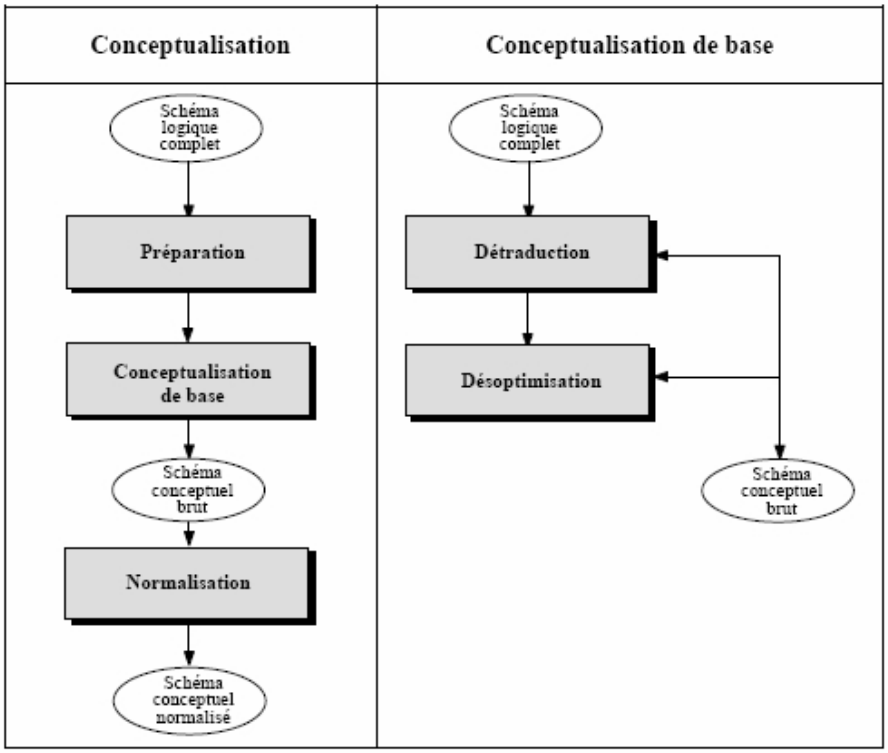


FIG. 6.7 – Conceptualisation des SD (repris du cours [Hainaut, 2002])

Cette seconde phase majeure vise à donner une interprétation conceptuelle du schéma logique issu du premier processus. Elle consiste par exemple à détecter et à transformer ou à supprimer des structures non-conceptuelles, des redondances, des optimisations techniques et des constructions dépendantes du système de management de la BD. Trois processus sont à dénombrer : la Préparation, la Conceptualisation de base et la Normalisation. La Conceptualisation de base se subdivise en deux sous-problèmes : la Détraduction et la Désoptimisation.

Voici un descriptif de ces processus appliqués au cas de $S^{3mAssess®}$:

– **Préparation** : il s'agit de supprimer les *structures de données obsolètes*, c'est à dire qui ne sont plus utilisées par le programme mais qui ont été laissées par les programmeurs successifs. Dans le cas de ce projet, il n'y en avait pas.
De plus, ce processus a également pour but de supprimer les *structures de données typiquement techniques*. Cependant, il a été décidé de laisser ces dernières dans le schéma conceptuel de la figure 6.6 afin de laisser ces informations aux futurs mainteneurs. Il faut savoir que l'accent a été mis, pour

ce projet, sur la maintenance de la partie maîtresse de l'outil (améliorer son processus d'évaluation et surtout ses résultats) mais qu'il existe également un menu "Administration" n'ayant pas fait l'objet d'un processus de maintenance dans le cadre de ce mémoire. Or, ce menu complémentaire utilise quelques structures de données complémentaires aux objets du domaine d'application. En ayant à la fois leur sémantique, via le schéma conceptuel, et leur implémentation actuelle, via le schéma relationnel, les futurs mainteneurs ont les cartes en main pour éventuellement les réimplémenter d'une autre manière. Il est à préciser que cela ne concerne au plus que trois ou quatre objets du schéma.
En addition aux deux phases suscitées, une phase typiquement cosmétique visait à améliorer les conventions de nommage et à restructurer certaines parties difficiles du schéma relationnel. Il s'agit de certains noms de tables et colonnes (par ex. : la table "QUESTIONNAIRE" devient "RESPONSE" ou la colonne "processID" devenant "Domain_Id" et "domainID" devenant "Domain_Acronym"). La différence sémantique peut d'ailleurs être assez importante puisqu'une pratique n'est pas une question (voir le glossaire de ce mémoire).
Autre exemple, une pratique est contenue dans une facette qui est contenue dans un itinéraire qui est enfin contenu dans un domaine de processus selon le schéma conceptuel. Dans le schéma relationnel, une pratique est contenue dans un itinéraire lié à des facettes et sans lien apparent avec un domaine de processus.

- **Conceptualisation de base** : il s'agit d'extraire tous les concepts sémantiques pertinents. Deux problèmes différents requérant différents raisonnements et méthodes doivent être résolus : *la détraduction de schéma* et la *désoptimisation de schéma*.
  - ***Détraduction* de schéma** : le schéma logique est la traduction techniques des constructions conceptuelles. Via ce processus, l'analyste identifie les traces de telles traductions et les remplace par leurs constructions conceptuelles originales.
    Ce travail de traduction a principalement eu lieu sur la traduction des clés étrangères (concept technique) en types d'associations (construction conceptuelle).
  - ***Désoptimisation* de schéma** : recherche de traces de constructions conçues à des fins d'optimisation. Trois familles principales de techniques d'optimisation sont à considérer : la dénormalisation, la redondance structurelle et la restructuration.
    Ce processus a eu lieu dans ce cas-ci non pas pour supprimer les quelques optimisations existantes mais pour les marquer d'un astérisque. Leur nombre d'occurence étant très bas, la lisibilité du schéma est préservée. Par exemple, le champ "Percentage_Completed" de la table RESPONSE a pour but de simplifier les calculs et les requêtes et a donc été marqué d'un astérisque.

- **Normalisation conceptuelle** : ce processus restructure le schéma conceptuel de base dans le but de lui donner les qualités attendues de tout schéma conceptuel final telles que l'expressivité, la simplicité, la minimalité, la lisibilité, la généricité et l'extensibilité. Par exemple, certains types d'entités ont été remplacés par des attributs et les noms ont été normalisés.

Cette sous-section clôt la phase de réingénierie de la Base de Données. La section suivante présente la réingénierie de l'application avec son architecture globale et de persistance et son évolution. La suite décrira donc pourquoi les changements en BD, tels que présentés ici, ont été nécessaires.

6.2.2 Réingénierie de l'architecture de l'application

Bon nombre de requêtes de modification de type correctives eurent lieu durant la réingénierie de l'application. Toutefois, elles ne seront pas détaillées ici car il est plus intéressant d'analyser et de critiquer l'architecture applicative du point de vue de la maintenance et d'ensuite présenter l'évolution de l'outil en vue de l'adapter au mieux au contexte réel des entreprises. Une liste complète recensant tout ce qui a été

réalisé au cours des trois révisions de $S^{3mAssess®}$ est disponible dans les annexes ; "*Description des trois révisions de* $S^{3mAssess®}$".
Il viendra donc tout d'abord une présentation et critique de l'architecture globale de $S^{3mAssess®}$ et ensuite de l'architecture de persistance des données. Une fois ces deux aspects importants abordés, les divers développements apportés visant à améliorer le processus d'évaluation et de prise de décision grâce à l'outil seront présentés.

6.2.2.1 Présentation et critique de l'architecture globale de $S^{3mAssess®}$

L'architecture employée dans $S^{3mAssess®}$ est dite "architecture deux tiers" car elle intègre chez le client à la fois la présentation et le traitement, lequel communique ensuite à distance avec un serveur de base de données (voir fig. 6.8). Le client interagit donc à distance pour obtenir des données et non des services d'application comme cela peut-être le cas dans une architecture 3-tiers séparant la présentation du traitement des données.

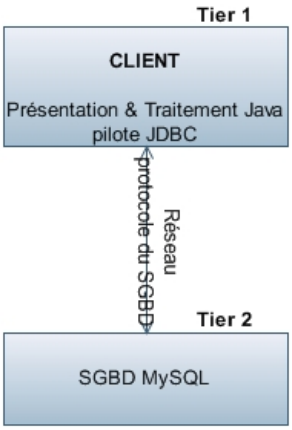


FIG. 6.8 – Architecture deux tiers de $S^{3mAssess®}$

Du point de vue de la maintenabilité, la découpe en deux tiers présente les avantages suivants :
- La base de données est stable et le moteur d'accès à celle-ci est centralisé. Des équipes de développement et de maintenance peuvent ainsi être spécialisées sur la couche de données de manière modulaire avec le reste de l'application.
- Le temps d'apprentissage du module des données est plus court.

Par contre, cette architecture présente également les limitations suivantes :
- Les coûts de développement et de maintenance s'élèvent avec le nombre d'utilisateurs, avec la complexité de l'application et avec la durée de vie de l'application. Une étude du groupe Gartner montre qu'une architecture 2-tiers s'avère moins couteuse qu'une architecture 3-tiers pour un projet impliquant de faibles valeurs pour ces variables. A l'inverse, au fur et à mesure que le projet prend de l'importance, une architecture 3-tiers s'avèrera moins couteuse, comme l'illustre la figure 6.9. L'expérience montre que le point de rencontre des deux courbes se situe entre 100 et 150 utilisateurs.

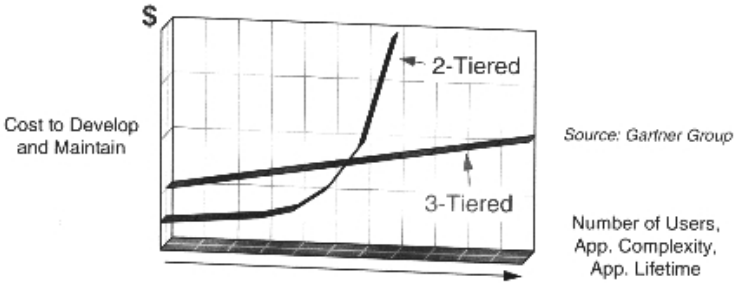


FIG. 6.9 – Comparaison des coûts de développement et de maintenance *(source : Gartner Group)*

- L'architecture 2-tiers se limite aussi par un manque de performance. Le client d'une telle architecture est dit "client lourd" étant donné qu'il lui revient la tâche d'effectuer les calculs de traitement. Si un serveur était dédié aux calculs et pouvait communiquer ensuite, en interne d'une organisation distante, avec un serveur de BD, comme cela peut-être le cas en 3-tiers, les performances de calculs pourraient être maîtrisées via la maîtrise du serveur dédié. Les limitations de performance dues au réseau en seraient également limitées. Le client ne devrait communiquer à distance qu'afin d'afficher des résultats. Tout le trafic réseau généré par les traitements de données pourrait être géré via un réseau interne. Un réseau Ethernet de 100mbits/sec limiterait déjà grandement les temps de latence chez le client. Par exemple, en considérant qu'il faille 200ms pour recevoir la réponse à une requête de BD entre un client européen et les Etats-Unis, il est évident que gérer ce genre de trafic en interne avec des temps de latence limités à quelques ms est beaucoup plus performant.

- Un autre désavantage du 2-tiers est un management d'application plus compliqué. Etant donné que la découpe crée un bloc de présentation/traitement de taille importante, il peut-être plus complexe pour les équipes de s'y retrouver, surtout si la découpe du code source n'est pas suffisamment fine. Le couplage entre classes peut-être beaucoup plus important qu'en 3-Tiers. Pour conséquence, cela peut rallonger les temps d'apprentissage et de traitement des RM par exemple.

- Enfin, une autre limitation concernant cette fois la sécurité peut être incriminée. En effet, le client communiquant directement avec un serveur de BD, des problèmes liés à la confidentialité et à l'intégrité des données peuvent survenir.

L'outil $S^{3mAssess®}$ n'étant encore que dans une phase expérimentale et non encore porté à être industrialisé, les limitations de l'architecture 2-tiers ne lui sont pas encore vraiment préjudiciables. En effet, ses équipes de maintenance restent très limitées, l'accent est encore principalement mis sur un développement des fonctionnalités les plus pertinentes et l'outil ne doit pas encore traiter simultanément plusieurs centaines de clients.
Au moment de l'acquisition de $S^{3mAssess®}$, c'est à dire au début du projet concernant ce mémoire, il était vital de commencer par restituer ses exigences initiales. De la maintenance corrective eut principalement lieu. Ensuite, des fonctionnalités ont été implémentées suite à des résultats de recherches (voir la sous-section 6.1.1 et les analyses d'impact en annexe). Avec l'augmentation de sa complexité, l'augmentation de ses utilisateurs et de sa durée de vie, c'est à dire avec le développement du logiciel et son adaptation à un contexte plus industriel, un changement d'architecture vers une plus fine modularité/granularité sera sans doute à envisager sérieusement. L'architecture devra s'adapter au contexte.

Au final, cette architecture peut-être critiquée pour avoir donc mélangé les couches Interface et Traitement. Ceci peut poser des problèmes de décomposabilité et de maintenabilité de l'application. Par exemple, il pourrait s'avérer utile dans le futur de réutiliser la couche Traitement avec une autre interface graphique. Par ailleurs, il peut être opportun de ne laisser à l'ordinateur client que la tâche de présentation, léguant

les calculs à un serveur distant. L'intégration des deux couches s'explique peut-être par le passé de l'outil dont la conception était basée sur une application monolithique à livrer au client. Avec le passage à une solution distante et avec l'industrialisation de l'outil, la découpe architecturale pourrait donc être revue.

Changer d'architecture en cours de maintenance n'est pas la meilleure solution. Il vaut mieux prévoir bien à l'avance quelle sera l'architecture la plus adaptée au contexte le plus probable du logiciel. Lorsqu'un changement architectural arrive dans l'organisation de maintenance en tant que requête de modification, il est très important d'analyser tout ce que cela va impliquer en tant que ressources, coût, effets de bord, risques, etc.
L'une des façons de faire peut-être via des analyses d'impact et via une planification des changements devant être avalisés au préalable par les membres de l'organisation de maintenance et ensuite validés par le client (contrat présentant les RMs, le temps que cela prendra, le coût, etc.). C'est d'ailleurs cette méthodologie qui a été développée à la section 6.1 et mis en oeuvre dans le cas du projet de maintenance de $S^{3mAssess®}$.

Après avoir présenté l'architecture globale de l'application, la section suivante va maintenant s'attarder sur une architecture bien particulière de persistance des données.

6.2.2.2 Principes d'une architecture de persistance basée sur une couche Objet-Relation

Une architecture particulière a été mise en oeuvre en ce qui concerne la gestion du stockage des objets java dans une base de données relationnelle. Cette architecture s'inspire du livre [Sperko, 2003] et a pour but de gérer la persistance des données de manière séparée et indépendante du code de l'application. De cette manière, les développeurs peuvent éviter de rechercher inlassablement les mêmes types de solutions, de s'attaquer aux mêmes problèmes et de répéter les mêmes types de tests et de débogage.
Un autre but est de pouvoir changer de système de persistance (fichier XML, base de données objets, Système de Gestion de Base de Données Oracle au lieu de MySQL, ...) sans devoir changer tout le code applicatif.
Toutefois, il faut également considérer l'architecture sous l'angle de la maintenabilité. Mettre en place des classes gérant de manière indépendante la persistance des objets entraine une augmentation générale du nombre de classes et entraîne la complexité de la conception de l'outil. De plus, implémenter une classe demande plus de temps, d'efforts et donc de coût que d'implémenter une méthode. C'est pourquoi, il est nécessaire de s'assurer que la mise en place d'une telle architecture est vraiment utile et nécessaire par rapport aux exigences initiales de l'application. Cette section donnera un coup de projecteur sur cet aspect et montrera l'utilité de la phase de transition, d'une documentation fournie et de la possibilité d'entrer en contact avec les développeurs.

Un mécanisme de stockage incluant de nombreuses classes a été mis en place pour rendre les objets java persistants (ce qui signifie que les objets existent même après que l'application ait terminé son exécution). La persistance est nécessaire dans le cas de $S^{3mAssess®}$ car l'outil est notamment amené à analyser et à comparer des évaluations exécutées à diverses dates. De plus, des évaluateurs, des entreprises, les pratiques du modèle $S^{3m®}$, etc. doivent être enregistrés et persister après l'arrêt de l'exécution de l'outil (qu'il soit volontaire ou non). C'est ainsi qu'il est nécessaire de stocker sur disque dur des objets tels la réponse à une pratique, un évaluateur, une entreprise, etc. dans des tables d'une base de données relationnelle dans le cas de $S^{3mAssess®}$. Ceci est différent de stocker un objet dans la mémoire du système car celle-ci est éphémère.
Afin de rencontrer ces objectifs de gestion indépendante de la persistance des objets, une couche de persistance a été implémentée entre la couche de Présentation et de Traitement et la couche de Données.

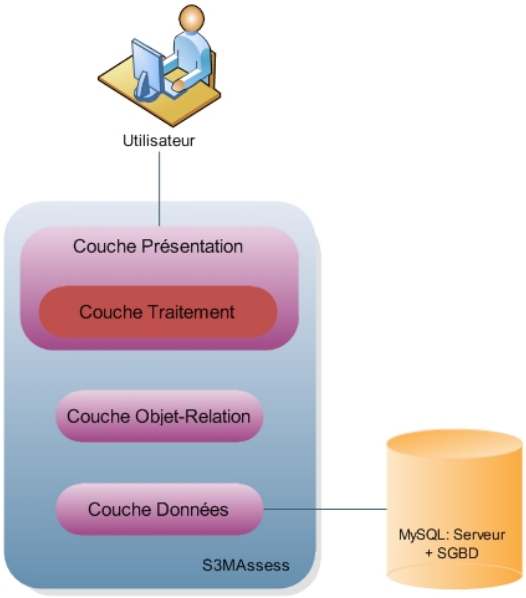


FIG. 6.10 – Les couches applicatives de $S^{3mAssess®}$

Cette couche "Objet-Relation" fournit une manière de lier l'attribut d'un objet Java à un champ d'une BD relationnelle, de lancer une requête sur un *objet* et de mettre à jour des objets. Le mot "objet" apparaît en italique dans la phrase précédente car l'idée maîtresse du livre [Sperko, 2003] est de stocker et de récupérer des objets de manière indépendante du reste de l'implémentation de l'application. Il s'agit donc de traiter des objets entiers. Or, dans $S^{3mAssess®}$, il est également souvent nécessaire d'aller ne rechercher dans la BD que certaines informations très spécifiques et donc, ne s'appliquant pas forcément à l'idée de les réencapsuler dans un objet. C'est l'une des raisons pour lesquelles la complexité de l'architecture n'est pas toujours justifiée dans le cas de cet outil. Une discussion traitant de la justification d'une telle architecture dans le cas de $S^{3mAssess®}$ sera proposée un peu plus loin dans cette section.

Une vue de haut niveau reprenant l'enregistrement d'un objet dans une base de données relationnelle à travers une couche objet-relation peut être représenté de la manière suivante :

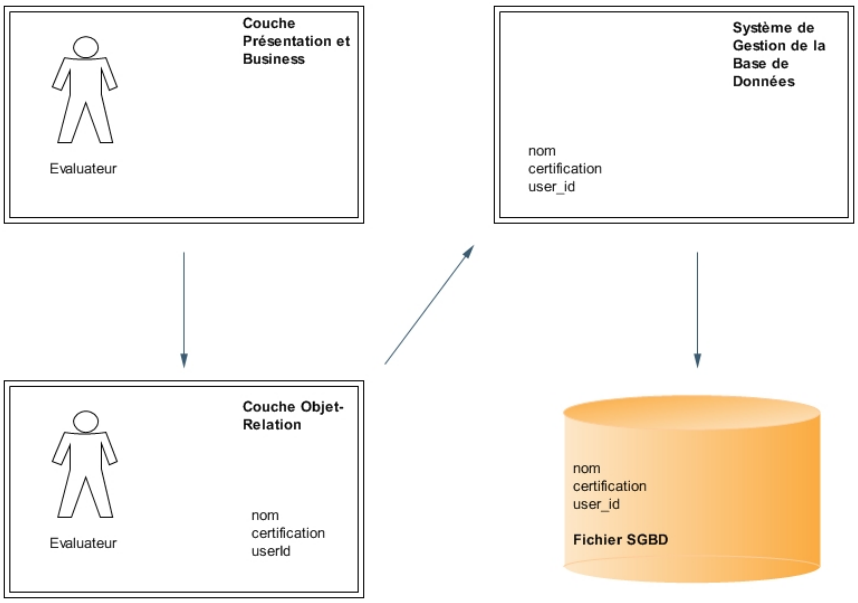


FIG. 6.11 – Enregistrement d'un objet "Evaluateur" dans une base de données relationnelle à travers une couche Objet-Relation (OR)

L'enregistrement représenté dans la figure ci-dessus est très similaire à l'enregistrement traditionnel d'objets dans une BD relationnelle à ceci près qu'il y a une pièce supplémentaire : la couche OR. Celle-ci offre beaucoup d'avantages des BD objet tout en permettant de garder les avantages d'une BD relationnelle. Ainsi dans ce diagramme, l'objet "Evaluateur" est fournit à la couche OR à partir de la couche Présentation et Business. La couche OR peut ou non mettre en cache l'objet mais elle doit surtout permettre son enregistrement en BD via un mappage des attributs de l'objet vers les champs de la BD. L'objet est alors enregistré dans la BD relationnelle en utilisant le langage des BD relationnelles : SQL (Structured Query Language).
L'idée est donc que le programmeur travaillant sur la couche Présentation et Traitement, ne manipule que des objets Java et ne s'occupe aucunement du code SQL interagissant concrètement avec la base de données.

6.2.2.3 Réalisation de la couche OR par des design patterns

Maintenant que les buts principaux et l'architecture générale de la persistance des objets via une couche OR ont été présentés, il faut préciser qu'il existe une multitude de solutions pour implémenter la couche OR. Pour cela, il existe ce qu'on appelle des *design patterns* qui sont des moyens de communication tout comme le sont les images et la conversation. Ce sont des documents contenant des instructions pour résoudre un problème spécifique dans un contexte spécifique. Des concepteurs logiciels dressent des patterns pour donner à d'autres concepteurs une voie ou une recette à suivre pour résoudre des problèmes de conception récurrents. En d'autres mots, les design patterns sont des cadres conceptuels qu'il est possible d'appliquer de manière appropriée sur certains aspects de son application ou système.

La couche de persistance des données dans $S^{3mAssess®}$ repose dès lors sur trois design patterns : le **Singleton Pattern**, le **Command Pattern** et l'**Event Pattern**. Ils ont été combiné tous les trois et sont utilisés en même temps d'une manière relativement complexe. Il n'est pas aisé de comprendre directement le fonctionnement d'une implémentation bâtie de manière simultanée sur trois voies de conception, d'autant plus lorsqu'il n'y a pas eu de phase de transition, qu'il n'y a pas de documentation claire à ce sujet et que les contacts avec les développeurs sont rendus à néant. Pourtant, le gain de temps et l'efficacité dans la compréhension induit par ces trois aspects permettent d'aider considérablement le mainteneur dans sa

tâche, comme il en est discuté plus loin dans cette section.

Pour comprendre comment tout ceci a été mis en oeuvre dans $S^{3mAssess®}$, un diagramme de classe reprenant les classes incriminées dans les différents patterns sera tout d'abord proposé. Ensuite, le lien entre ces classes et les classes théoriques appartenant aux différents patterns sera discuté. Enfin, la syntaxe d'un diagramme d'activité sera utilisée pour pouvoir retracer le plus clairement possible un exemple d'insertion dans la BD relationnelle à partir du code de la couche Présentation/Traitement.

Avant d'aller plus loin, il est à préciser que si le lecteur ne connait pas très bien le fonctionnement des trois design patterns utilisés dans la réalisation de la couche OR, il lui est loisible de se rendre à la partie 12 *"Détails théoriques sur les design patterns utilisés par la couche OR"* des annexes pour obtenir les détails théoriques.
Le paragraphe suivant va maintenant détailler, via la figure 6.12, comment les design patterns ont été mis en oeuvre pratiquement. L'architecture proposée implique cinq paquetages : smmm.db.command, smmm.db.table, smmm.db.connection, smmm.db.event et smmm.gui. Il est à noter que par soucis de concision, seules les classes pertinentes à la présentation de la couche OR sont illustrées.

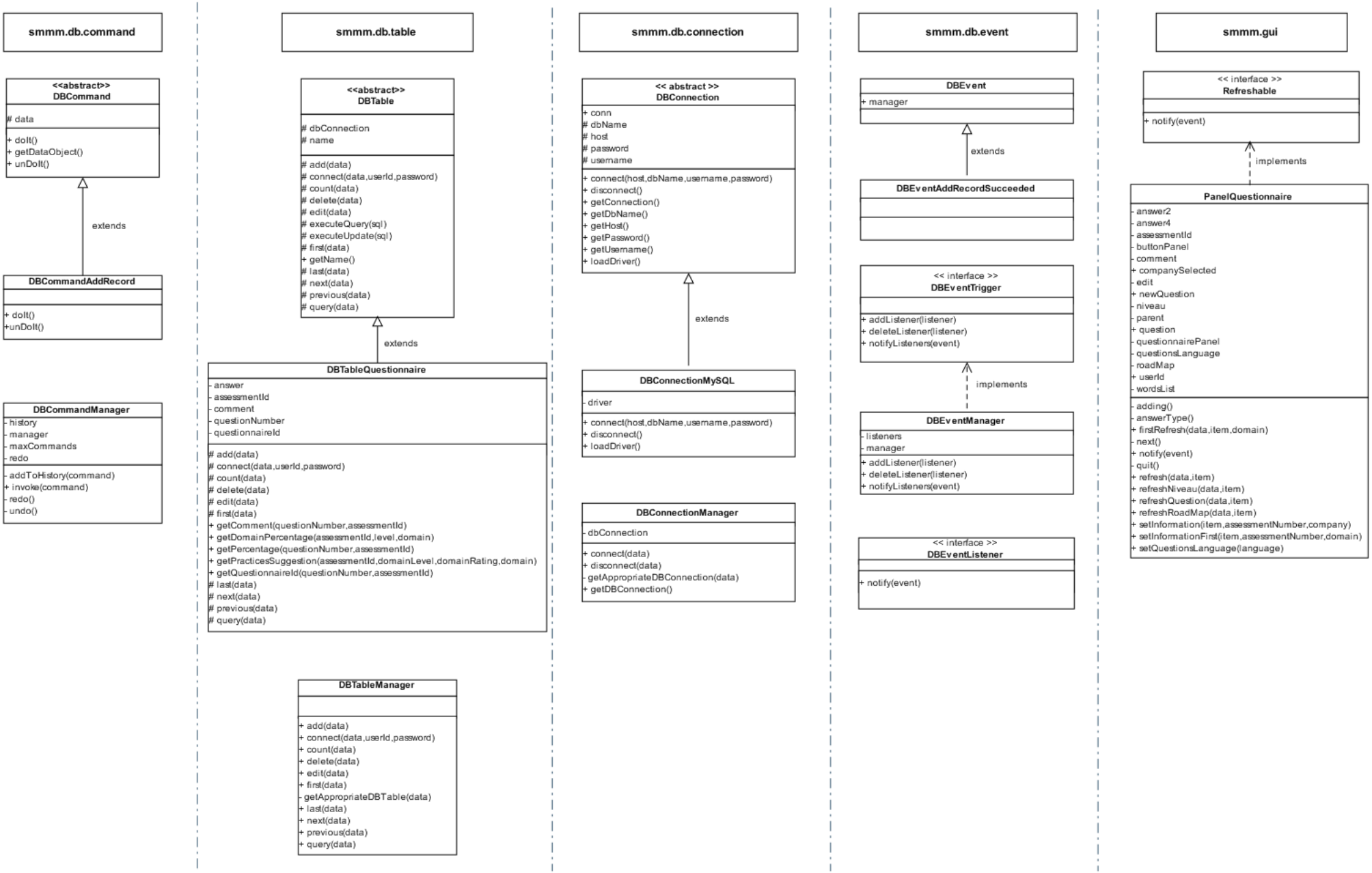


FIG. 6.12 – Diagramme de classes reprenant les classes incriminées par la couche OR

Le paquetage gérant les commandes se constitue d'une classe abstraite **DBCommand** à partir de laquelle plusieurs sous-classes héritent. Ces sous-classes représentent chacune un type de commande particulier. L'exemple d'une commande représentant l'ajout d'un enregistrement dans la base de données a été choisi dans la figure. C'est cet exemple de réalisation d'une insertion qui sera suivi tout au long de la présentation de l'architecture. La sous-classe a pour tâche de réaliser la commande envoyée par l'utilisateur et de générer un événement quant à la réussite de l'exécution de la commande.
DBCommand représente une commande basique à exécuter dans la BD et représente également une commande basique dans le Command Pattern. Cette classe doit être étendue par une sous-classe parce qu'une commande basique ne peut être exécutée : elle ne représente rien du point de vue d'une BD.
Enfin, DBCommandManager gère un historique des commandes et exécute certaines actions en fonction du type de la commande. Cette classe est utilisée pour le Command Pattern et respecte aussi le Singleton Pattern.

Le paquetage **smmm.db.table** est constitué d'une classe abstraite DBTable et de plusieurs sous-classes qui l'étendent. Chacune de ces sous-classes représente une table de la base de données relationnelle. L'exemple pris est DBTableQuestionnaire mappant ses attributs d'objet aux champs de la table 'questionnaire' et permettant d'effectuer diverses opérations sur cette table.
DBTable est utilisée quant à elle pour représenter une table générique de la BD. Tandis que DBTableManager est principalement utilisée afin d'obtenir la table appropriée comme ce sera illustré un peu plus loin.

Le paquetage **smmm.db.connection** se compose également d'une classe abstraite (DBConnection), d'une sous-classe (DBConnectionMySQL) et d'un manager (DBConnectionManager). DBConnection est utilisée pour un type particulier de BD. Elle évite les dépendances à des implémentations concrètes de connexion à une base de données. Ainsi, l'application peut se connecter de manière transparente à beaucoup de types différents de BD.
DBConnectionMySQL représente une connexion spécifique à une BD MySQL. Cette classe respecte le Singleton Pattern.
Enfin, DBConnectionManager est utilisé pour gérer la connexion à une BD. Il maintient la connexion à la BD de sorte que son accès soit disponible de partout dans l'application. Cette classe respecte aussi le Singleton Pattern.

Le paquetage **smmm.db.event** est un peu plus complexe que les autres de part le fait qu'il gère les événements, les receveurs – "listeners" – ainsi que les données générées par ces événements. Au-dessus de la hiérarchie de classes, une superclasse et deux interfaces sont présentes. La superclasse DBEvent représente un événement basique déclenché après qu'une DBCommand ait eu lieu sur la BD et récupère les enregistrements qui en sont générés. Cette classe est étendue par des sous-classes parce qu'un événement basique n'a pas de sens du point de vue de la base de donnée.
DBEvent représente un événement basique dans l'Event Pattern. Ses sous-classes représentent chacune un type d'événement particulier. Le diagramme de classes illustre l'exemple d'un DBEventAddRecordSucceeded qui représente un événement concret. Dans le cas de cette classe, la seule tâche lui incombant est de notifier les receveurs d'événements via le DBEventManager (ceci est fait dans le constructeur de la classe, lequel n'est pas représenté dans le diagramme). DBEventAddRecordSucceeded représente un événement notifiant la réussite de l'ajout d'un enregistrement dans la BD. Il représente un événement concret dans l'Event Pattern.
L'interface DBEventTrigger définit les méthodes qu'un DBEventTrigger doit offrir. Elle déclenche des DBEvents qui sont ensuite reçus par tous les DBEventListeners qui écoutent ce seul et unique DBEventTrigger. Cette interface représente un trigger dans l'Event Pattern.
C'est le DBEventManager qui implémente cette interface et qui est utilisé pour gérer les DBEvents et les déclencher pour tous les DBEventListeners enregistrés. Ses méthodes de gestion sont l'ajout, la suppression et la notification des récepteurs d'événements. DBEventManager est utilisé dans le cadre de l'Event

Pattern. Cette classe respecte le Singleton Pattern.
Enfin, l'interface DBEventListener définit les méthodes qu'un DBEventListener doit offrir. Un DBEventListener écoute un DBEventTrigger et est notifié quand un DBEvent est déclenché. Elle représente un listener dans l'Event Pattern. Les classes implémentant cette interface sont typiquement celles appartenant à smmm.gui.

Finalement, **smmm.gui** est simplement constitué d'une interface : Refreshable. Elle définit les méthodes qu'un JPanel (un panneau graphique appartenant au langage Java) Refreshable doit offrir, c'est-à-dire typiquement une méthode permettant au panneau Java de rafraîchir ses composants graphiques. Cette interface n'intervient pas directement dans l'architecture mise en place pour gérer la couche OR. Elle entraine le rafraîchissement de l'interface graphique lorsqu'elle reçoit la notification d'un événement relatif à la BD. Cette interface est implémentée par des classes représentant chacune un panneau Java constituant l'interface graphique. Le diagramme illustre l'exemple d'une sous-classe "PanelQuestionnaire" représentant un panneau permettant l'affichage de l'évaluation d'une pratique $S^{3m®}$. Pour rappel, c'est typiquement ce genre de sous-classe qui implémente l'interface smmm.db.event.DBEventListener présentée ci-dessus.

Avant de proposer un diagramme d'activité représentant un exemple de fonctionnement de la couche OR, il est utile de voir comment les design patterns ont été utilisés pratiquement dans $S^{3mAssess®}$. La figure théorique 9.4 a donc été adaptée de la manière suivante :

- La classe Invoker composée de commandes devient une classe singleton "DBCommandManager" permettant d'invoquer les commandes (via une méthode invoke(command)) au fur et à mesure de leur arrivée.
- La classe ConcreteCommand s'appelle DBCommandAddRecord et sa méthode execute() est implémentée par la méthode doIt().
- Le Receiver implémentant la méthode action() et étant déclenché par la classe ConcreteCommand est une sous-classe de la classe abstraite DBTable. Sa méthode action() est implémentée par l'une des méthodes représentant des actions qu'il est possible de faire sur une DBTable. Ce sont ces méthodes qui exécutent réellement le travail demandé par la commande générée par le client.
- Dans l'exemple de cette section, la classe Client est PanelQuestionnaire, Receiver est DBTableQuestionnaire et un objet Command est matérialisé par le constructeur new Questionnaire(...) (contenant les paramètres définissant l'action à réaliser sur la table questionnaire de la BD) encapsulé dans un objet DBCommandAddREcord, ce qui donne un appel du type :
  `"new DBCommandAddRecord(new Questionnaire(...))"`
- Le rôle de l'interface Command est quant à lui joué par la classe abstraite DBCommand.
- Enfin, en mélangeant ces concepts avec ceux appartenant à l'Event Pattern, un objet Commande pourra détenir des informations à propos de la commande grâce aux résultats collectés par les sous-classes de DBEvent mais également via les paramètres donnés au constructeur (dans le constructeur new Questionnaire(...) par exemple). Ces informations seront les paramètres définissant une commande et l'événement généré suite à l'exécution de cette dernière. Il est à préciser que les événements générés sont des DBEvent notifiant la réussite de l'achèvement d'une commande.

  Contrairement à ce que préconisait l'Event Pattern, les DBEventListeners font partie de la couche Présentation. Toutefois ils sont générés via la couche intermédiaire OR. La gestion des listeners se fait dans l'interface DBEventTrigger via le singleton DBEventManager l'implémentant.
  Lorsqu'un événement est généré, il n'y a pas de test à faire pour déterminer le listener à notifier étant donné qu'il n'y a qu'un seul type d'événement possible dans $S^{3mAssess®}$: des *DBEvent*. De plus, les listeners se désinscrivent après que leur commande ait été exécutée.

  Une autre différence avec l'Event Pattern est que les listeners n'héritent pas du composant qu'ils

écoutent (dans l'exemple, le composant est DBCommandAddRecord) mais de l'interface DBEventListener. Les listeners sont ainsi notifiés via le DBEventManager, lequel a été appelé par les objets Event générés.

Concernant la définition des listeners, elle est locale à l'endroit où ils sont utilisés étant donné que les panneaux implémentent l'interface DBEventListener qui est définie dans le paquetage définissant les événements (smmm.db.event). Autrement dit, l'interface DBEventListener est bien locale au paquetage smmm.db.event puisqu'elle y est définie. Par conséquent, les panneaux contiennent à la fois les commandes (qu'ils instancient) et les instructions liées à la notification d'un événement (dans la méthode notify(event)).

Afin d'y voir plus clair quant à la réalisation de la couche OR, voir la figure 6.13 illustrant le diagramme d'activité reprenant l'exemple de l'ajout d'un enregistrement dans la base de données relationnelle.

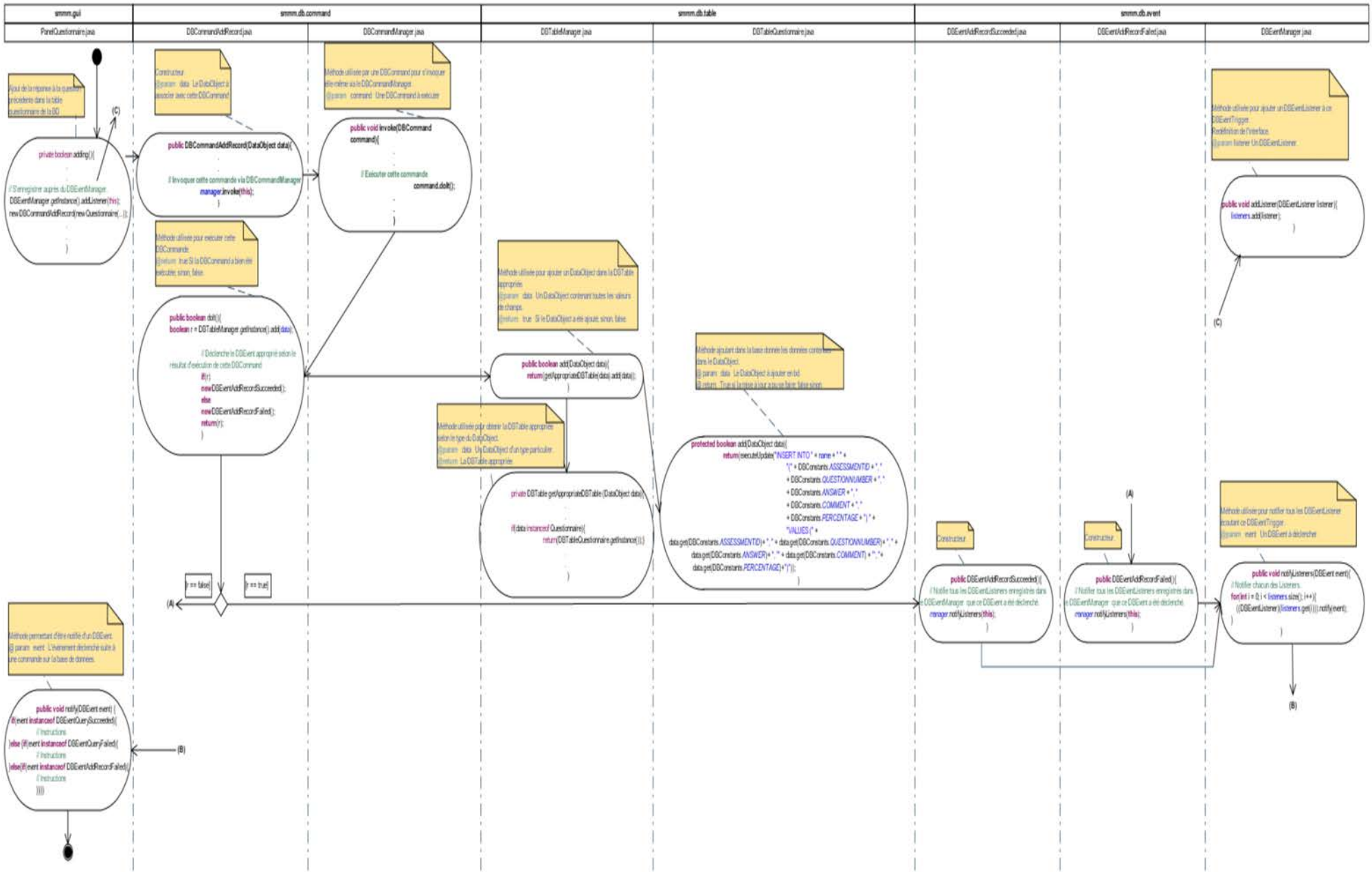


FIG. 6.13 – Diagramme d'activité utilisant la couche OR

6.2.2.4 Critique de la couche *Objet-Relation* du point de vue de la maintenance

Importance de l'encadrement du mainteneur En augmentant la complexité du design de part le nombre de classes, le nombre d'appels nécessaires à l'exécution d'une tâche, la structure conceptuelle, etc. la phase de transition entre les phases de développement et de maintenance prend tout son intérêt. Au lieu que le mainteneur ne doive passer des heures à disséquer le code, à essayer de réunir des bribes d'informations, à recréer lui-même des diagrammes de conception lui permettant de comprendre les idées et décisions prises par les développeurs et lui permettant de comprendre le fonctionnement du logiciel, une formation ou un entretien avec les développeurs, d'une durée minime comparativement au temps de compréhension nécessaire lorsqu'il est seul, permettrait un gain de temps et une efficacité énorme. La phase de transition devrait donc commencer très tôt dans le développement d'un logiciel en incluant une ou plusieurs personnes de l'équipe de maintenance dans l'équipe de développement et ce, dès la phase de conception. Ensuite, ces quelques personnes suivraient les étapes suivantes du développement du logiciel jusqu'à sa phase de maintenance. Elles auraient alors toutes les cartes en main pour assurer efficacement cette phase qui est, faut-il le rappeler, au final la plus longue dans la durée de vie d'un logiciel. Il incomberait ensuite, à ces personnes, la tâche de transmettre aux autres membres de l'équipe de maintenance toutes les connaissances acquises.

En plus de l'évidente importance de la phase de transition et d'une documentation (également dans le code) claire et détaillée, la possibilité d'entrer en contact avec les développeurs aide indéniablement le mainteneur. Tout cet ensemble lui permet d'entrer beaucoup plus rapidement et efficacement dans sa tâche de compréhension de l'application. Du matériel de compréhension lui étant ainsi disponible, un encadrement compétent dans son équipe de maintenance et des contacts avec les développeurs en cas de problèmes plus spécifiques assureraient ensemble la viabilité de la phase de maintenance. Egalement, le mainteneur se sentirait de cette manière beaucoup plus impliqué dans le développement du logiciel et son travail serait d'autant plus reconnu par les développeurs et clients. La complexité de sa tâche serait perçue par les développeurs et la transparence et efficacité de sa tâche perçue par le client. Ceci aide à répondre au mauvais sentiment général du mainteneur décrit dans [April et Abran, 2006].

Justification de cette architecture pour $S^{3mAssess®}$ Cette couche de persistance des données a pour but principal d'enregistrer des objets applicatifs dans une base de données relationnelle, de les modifier et ensuite de les récupérer et ce, d'une manière indépendante du code Business. Un avantage certain est que cela permet de pouvoir changer de base de données tout en n'ayant pas à devoir modifier le code Business. Il serait ainsi possible de passer de MySQL à Oracle sans devoir modifier du code existant mais juste en écrivant de nouvelles classes spécifiques à Oracle.
Ceci peut être intéressant mais une condition nécessaire à une telle architecture est de fournir aux prochains mainteneurs une documentation adéquate comme il en a été discuté à la section précédente. Or ceci n'était pas le cas dans le cadre de ce stage. L'un des buts principaux de la première révision de $S^{3mAssess®}$ consistait donc non seulement à remettre l'application sur pied mais surtout à comprendre l'architecture qui était en place.

Ensuite, le fait de pouvoir changer de base de données ne parait pas vraiment pertinent pour une application comme $S^{3mAssess®}$. En effet, celle-ci n'est pas sujette à devoir être prise en charge par d'autres entreprises et par la même occasion, à devoir s'adapter aux exigences d'un système informatique déjà en place.
De plus, MySQL est un SGBD suffisamment rapide, fiable et gratuit et convient par conséquent tout à fait à une application comme $S^{3mAssess®}$. Comparativement, Oracle est plus complet dans ses fonctionnalités mais s'avèrerait trop lourd dans ce cas-ci en termes de ressources informatiques nécessaires et en terme de budget.

Dès lors, l'ensemble du manque de documentation et d'encadrement ainsi qu'une nécessité assez limitée d'une architecture de cette complexité semble ne pas pouvoir se justifier. Ceci est d'autant plus vrai que la réduction de la complexité du logiciel est un des buts clés de la maintenance. Les diagrammes de classe et d'activité ont démontré la complexité de la gestion des requêtes de BD. A l'acquisition, les requêtes n'étaient d'ailleurs pas bien construites : leur construction générique ne fonctionnait pas. L'architecture du traitement des commandes de la BD n'est pas forcément propice à la maintenance car elle est compliquée de part le nombre de classes incriminées et en plus, ne génère pas d'exception SQL dans la façon où elle a été implémentée. Ce dernier point rend difficile le traitement des erreurs. Les performances moyennes peuvent aussi être amoindries de part le nombre d'appels aux méthodes java, même s'il est vrai qu'avec les ordinateurs actuels, ce critère devient négligeable. Une autre façon aurait été de procéder à l'appel direct d'une méthode implémentée dans la classe conceptualisant une table de BD.

Par contre, la présence de cette architecture se justifie plutôt par le fait qu'elle avait un but d'apprentissage pour des étudiants. Ensuite, l'aboutissement du stage réalisé dans le cadre de ce mémoire permettra aux futurs mainteneurs de débuter leur travail à partir d'un outil remis sur pied, évolué et documenté via ce mémoire et via la javadoc. Suite à cela, l'architecture proposée sera plus efficace du point de vue de la maintenance et le travail réalisé permettra de profiter des avantages de la couche OR tout en limitant ses inconvénients de maintenance.

6.2.2.5 Conclusion

Une fois les phases de réingénierie de la base de données et de rétro-ingénierie de l'application effectuées, la réalisation des requêtes de problème (RP) et des requêtes de modification (RM) pouvait commencer. En effet, ce sont ces deux phases qui assurent au mainteneur de pouvoir répondre à ces requêtes de manière efficace et fiable car il possède alors une bonne connaissance du système qu'il maintient. De plus, dans ce cas-ci de maintenance, il n'y eu aucun membre de l'équipe de maintenance présent lors du développement de $S^{3mAssess®}$, les contacts avec les développeurs étaient impossibles et la documentation disponible était très limitée, voire inexistante. Dans ces conditions, il est bien entendu très difficile de pouvoir considérer une connaissance complète du système même si elle atteint un niveau très acceptable à l'issue de ce qui vient d'être expliqué.

Pour rappel, la première révision de l'outil corrigeait les principales défaillances qui empêchaient une quelconque exécution correcte de $S^{3mAssess®}$. Les révisions 2 et 3 corrigeaient quant à elles les dernières défaillances et apportèrent surtout de nouvelles fonctionnalités pertinentes et directement adaptées au contexte réel des entreprises. C'est ce que la prochaine section va présenter.

6.2.3 Réingénierie de l'application : Evolution de l'outil

Les sections précédentes ont établi le contexte de $S^{3mAssess®}$, présenté la mise en place de son projet de maintenance et ont décrit la réingénierie de sa Base de Données et de son architecture. Une fois le processus de maintenance défini et l'outil corrigé, les questions de recherche pouvaient réellement commencer à être analysées, solutionnées et validées pour être ensuite intégrées à l'outil. La présente section va donc présenter pour quelles évolutions tout cela a été nécessaire. Ces évolutions ont bien entendu influencé les points cruciaux du projet et il est utile de voir comment elles peuvent répondre à diverses exigences contextuelles. Le détail technique de leur implémentation ne sera pas analysé ici car il est plus pertinent de voir exactement à quels besoins ces évolutions répondent et comment elles peuvent réellement aider une entreprise à améliorer ses processus de maintenance, à déterminer où elle se situe par rapport à la concurrence ou encore à évaluer la maturité d'une de ses filiales.
A cette fin, les évolutions seront présentées selon la démarche suivante :

1. Nom caractérisant l'évolution ;

2. Présentation du contexte dans lequel elle intervient et des besoins auxquels elle répond ;
3. Implémentation de la solution répondant à ces besoins.

6.2.3.1 Mise en ligne de l'outil

Contexte et besoins L'une des premières évolutions véritables de l'outil, sa correction n'étant pas une "évolution", fut sa mise en ligne. Ceci est la conséquence directe des difficultés techniques rencontrées par les clients pour installer et configurer le système permettant le fonctionnement de $S^{3mAssess®}$. En effet, il était initialement prévu de fournir aux clients non seulement les fichiers java compilés en un fichier exécutable "jar", mais également le fichier d'installation du serveur de BD "MySQL" ainsi que le fichier SQL permettant l'initialisation de la BD. Afin de les aider dans leur tâche, un fichier PDF détaillant les étapes de l'installation du serveur de BD et de sa configuration et présentant l'outil était également fourni (la version finale de ce guide utilisateur est accessible en annexe, *Guide d'introduction au logiciel* $S^{3mAssess®}$). Cependant, bien que les clients appartenaient à des organisations de maintenance de logiciels informatiques, cette solution ne s'avéra pas adaptée.

Solution implémentée La solution fut de permettre l'exécution du logiciel sur un serveur : l'utilisateur se connecte à une page web via l'adresse *s3massess.logti.etsmtl.ca* chargeant l'applet. L'utilisateur doit alors accepter, lors de sa première visite, un certificat $S^{3mAssess®}$ donnant droit à l'outil d'avoir accès au disque dur du client.
Le serveur de BD MySQL tourne sur un serveur LINUX situé à l'ETS de Montréal (voir la description du déploiement à la sous-section 5.3.4).
Cette solution offrait l'énorme avantage de simplifier la tâche de l'utilisateur au niveau de l'installation puisqu'elle était dès lors réduite à néant. Cela lui faisait gagner du temps et il pouvait donc commencer directement à utiliser l'outil.
Par contre, étant donné l'architecture 2-Tiers, si l'utilisateur ne se trouve pas suffisamment à proximité de Montréal, le temps de latence pour analyser une évaluation conséquente peut s'avérer très important. Cela est d'autant plus vrai lorsque l'utilisateur doit traverser un océan pour communiquer avec le serveur de Montréal puisqu'environ 200ms sont nécessaires pour l'aller-retour d'un requête. L'adaptation de l'architecture de l'outil à ce nouveau contexte est d'ailleurs l'un des travaux futurs à apporter à l'outil (voir le chapitre 9).
Quoiqu'il en soit, cette solution a été appréciée et a permis une première utilisation de l'application par les clients.

6.2.3.2 Améliorer le processus d'évaluation

Contexte et besoins Une connaissance pauvre des pratiques exemplaires et de leurs bénéfices pour la maintenance du logiciel peuvent mener à des investissements inefficaces et inappropriés. Par conséquent, cela peut :

1. créer de l'insatisfaction chez les clients ;
2. entraver l'optimisation des délais et des coûts de service ;
3. retarder le déploiement des améliorations, des innovations et des investissements dans les technologies de l'information.

Les entreprises veulent pouvoir effectuer des évaluations de la maturité de leurs processus de maintenance aussi vite que possible afin de lancer et de soutenir au plus vite un programme d'amélioration continue. Cette nécessité de rapidité d'évaluation des forces et faiblesses est d'autant plus vraie pour de petites organisations ayant des moyens limités.
L'outil $S^{3mAssess®}$ est basé sur le modèle d'évaluation des processus de maintenance : le $S^{3m®}$. Par conséquent, il est adapté aux activités dites "de petite maintenance". Ces activités ne se gèrent pas comme

des projets de grande envergure même si le $S^{3m®}$ fait appel à des techniques de gestion de projet dans la formulation de ses pratiques. Cela veut dire que le contexte d'utilisation de l'outil comportera peu de mainteneurs, le temps sera particulièrement compté, une évaluation ne devra donc pas demander trop d'effort et elle devra pouvoir être ciblée.

Pourtant, étant donné que des décisions d'amélioration importantes et coûteuses peuvent ensuite être prises, il est vital que l'évaluation soit pertinente, exacte et qu'elle puisse donner suite à une analyse complète facilitant au mieux la prise de décision. C'est de ce contexte d'antagonismes qu'est né l'un des objectifs premier de ce mémoire : réussir à concevoir une solution répondant à ces besoins de résultats rapides mais néanmoins complets et pertinents.
L'outil devra :

- aider à la compréhension du modèle $S^{3m®}$ et,
- supporter une évaluation en faisant gagner encore plus de temps que celui déjà épargné en utilisant le modèle $S^{3m®}$ plutôt qu'une méthode comme SCAMPI ([SEI, 2001b]) par exemple.

Solution implémentée En se basant sur le modèle $S^{3m®}$, l'application se réfère également indirectement aux recommandations de la norme ISO/IEC 15504. Toutefois se baser uniquement sur un modèle n'est pas suffisant pour être conforme à cette norme. C'est pourquoi, celle-ci a également fait l'objet d'une étude au chapitre 2. De cette manière, l'application tient compte de la description du contexte de l'évaluation (voir la section 2.2) et en particulier de la cotation des processus et de la création des rapports. En effet, une cotation est assignée et basée sur des données validées (par un utilisateur-évaluateur compétent) pour chaque processus et les résultats d'évaluation sont documentés et peuvent aisément être rapportés au sponsor de l'évaluation. Les recommandations sur les résultats seront exposées à la sous-section 6.2.3.4 tandis que ce paragraphe détaillera la cotation des processus.

Toutefois, avant de présenter le processus même d'évaluation, il faut tenir compte de l'importance des informations à réunir avant de commencer. Le processus d'évaluation est initié à la demande du sponsor de l'évaluation. La norme recommande que les inputs de l'évaluation définissent l'identité du sponsor de l'évaluation, son objectif, son étendue, les responsabilités, les critères de compétence, l'identité du/des modèles utilisé(s) pour l'évaluation, l'identité des évaluateurs et les éventuelles contraintes. Ainsi, avant l'utilisation de l'outil, il est nécessaire d'établir le contexte de l'organisation de maintenance, la limitation de l'étendue du/des projets(s) et la définition des processus de maintenance employés (voir l'exemple de l'étude de cas au chapitre 7). Le contexte du processus est important à prendre en compte car il influence le jugement d'une pratique et le degré de comparabilité entre les profils de processus.
Toutes ces informations préalables alliées à une formation minimale au modèle pour le(s) évaluateur(s) et alliées à l'utilisation de l'outil par un évaluateur compétent sont nécessaires à la garantie d'une bonne détermination de maturité des processus de maintenance de l'entreprise.
L'outil $S^{3mAssess®}$ permet de recueillir certaines de ces informations et de les lier aux évaluations effectuées. C'est ce qu'illustre la figure 6.14

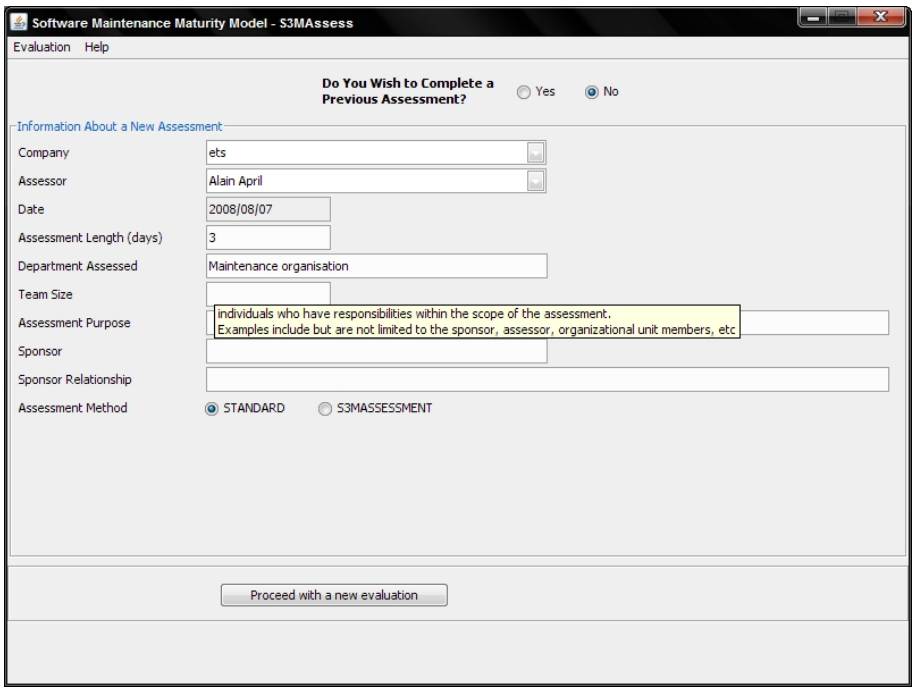


FIG. 6.14 – Données d'entrée à propos de la nouvelle évaluation

Cette fenêtre apparaît au tout début du processus d'une nouvelle évaluation. Des bulles d'aide ont été prévues pour définir les champs à compléter.
A noter que l'utilisateur a le choix entre deux méthodes d'évaluation possibles : la méthode standard et la méthode $S^{3m}Assessment$. Cette distinction sera détaillée lors de la sous-section 6.2.3.3.
Par ailleurs, pour répondre au besoin de réaliser une évaluation ciblée, la solution apportée a été de donner la possibilité de sélectionner le domaine et l'itinéraire à évaluer comme l'illustre la figure 6.15.

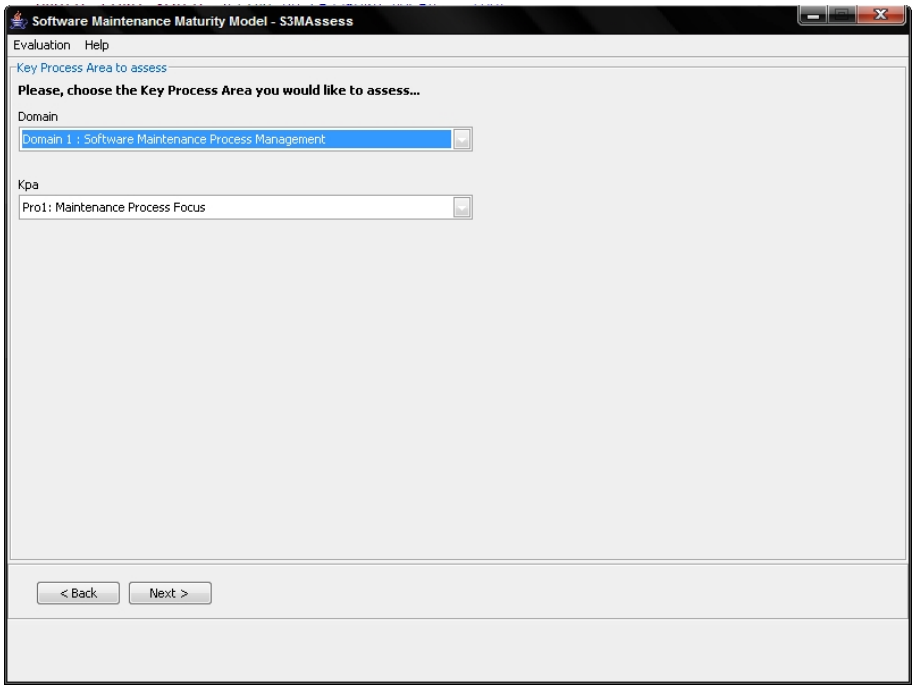


FIG. 6.15 – Choisir le domaine et l'itinéraire à évaluer

L'utilisateur peut choisir l'un des quatre domaines de processus définis par le $S^{3m®}$ et ensuite l'un des itinéraires afférant au domaine choisi.

De plus, afin d'offrir des résultats correctes et pertinents au contexte de l'organisation de maintenance, une option "Not Applicable" a été rendue disponible lors du processus d'évaluation des pratiques $S^{3m®}$ (voir fig. 6.16). En choisissant de ne pas évaluer une pratique, l'utilisateur ne compromet donc pas ses résultats finaux car la pratique est ignorée dans les calculs des moyennes.

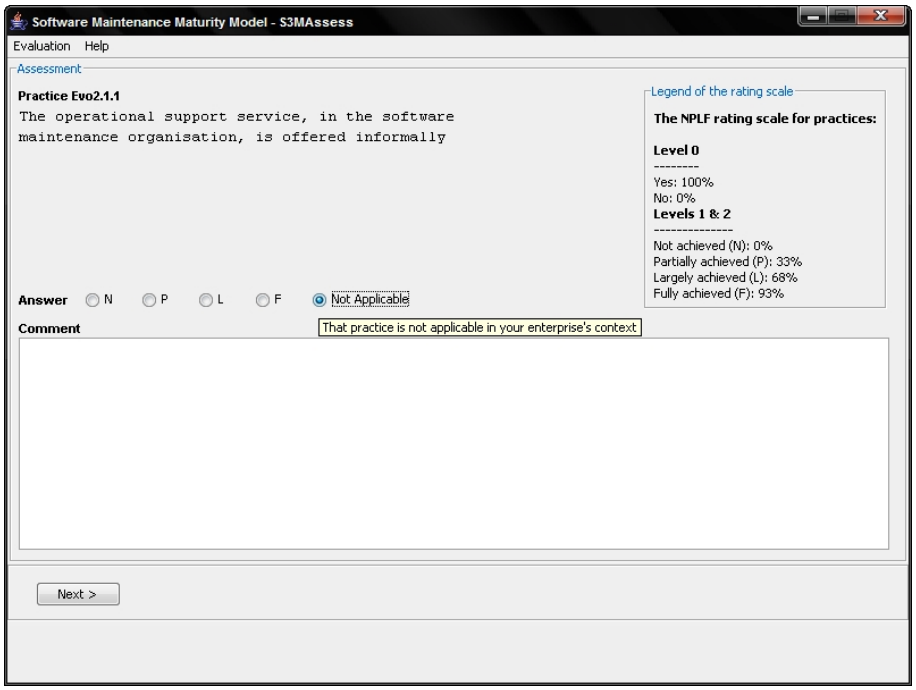


FIG. 6.16 – Evaluation d'une pratique $S^{3m®}$

A partir de la figure 6.16, il est également utile de préciser qu'un travail a été effectué, par rapport au prototype initial, quant à la rétroaction proposée à l'utilisateur. Ce dernier possède désormais les informations suivantes :

- l'identifiant de la pratique courante ;
- la légende de l'échelle de cotation définie comme suite (sur base de l'article de référence [Paquette *et al.*,]) : la valeur de 0% est assignée aux pratiques non accomplies (celles du niveau 0) et aux pratiques des niveaux supérieurs cotées "Not Achieved". Pour les autres niveaux de cotation (P, L et F), la valeur médiane de l'intervalle de la cotation (voir la section 2.3) est calculée et utilisée comme ceci :
  P : Partiellement atteint : (50% - 16%)/2 + 16% = 33%
  L : Largement atteint : (85% - 51%)/2 + 51% = 68%
  F : totalement (Fully) atteint : (100% - 86%)/2 + 86% = 93%

Cette échelle de cotation se base sur l'ISO 15504 car par exemple, la valeur de 86% représente la cotation minimum du niveau "Fully Achieved" selon cette norme. La médiane est à chaque fois calculée sur base des intervalles définis dans la norme. Cette échelle facilite le calcul des pourcentages et réduit la subjectivité (voir la signification des cotations à la section 2.3).

Il est aussi utile de préciser qu'un espace de commentaire est disponible offrant l'avantage de pouvoir traquer en détails toutes les pratiques lors de l'analyse finale. Ceci rencontre donc encore une autre recommandation de la norme ISO 15504.

Finalement, le processus d'évaluation se poursuit du premier niveau de maturité (0) jusqu'au troisième niveau (2). A la fin de l'évaluation de l'itinéraire sélectionné, un message apparait et donne les informations nécessaires à l'identification de l'évaluation courante. L'utilisateur a alors le choix d'évaluer un autre itinéraire (voir fig. 6.15) ou d'analyser l'évaluation qu'il vient de réaliser. L'application rencontre ainsi une autre recommandation ISO 15504 sur la réalisation d'une évaluation cohérente et fiable : le processus d'évaluation est guidé à travers l'utilisation de l'outil et fiabilisé via la correction de $S^{3mAssess®}$ et via la réduction de la subjectivité de l'évaluateur en lui proposant une échelle de cotation bien définie.

Par ailleurs, $S^{3mAssess®}$ permet aussi la centralisation des pratiques et des résultats d'évaluation via une base de données. Une BD est particulièrement utile ici car elle permet de stocker un grand volume d'informations et d'en extraire facilement des données recoupées. Dès lors, plus aucune feuille de papier n'est nécessaire et le processus d'évaluation peut être réalisé plus facilement.
Egalement, il est possible de projeter sur grand écran la succession des pratiques afin de réaliser l'évaluation dans un local comprenant plusieurs personnes. De cette manière, ces dernières peuvent se concentrer uniquement sur les réponses données et n'ont pas de feuilles à manipuler. Ensuite, les résultats sont directement disponibles automatiquement et peuvent être projetés.

Cette sous-section a donc présenté les principaux points (voir le guide utilisateur en annexe pour une liste plus exhaustive) de la partie "réalisation d'une évaluation" de l'outil. Cette première partie principale permet déjà de répondre à un bon nombre d'attentes des clients et du contexte de l'application. Les entrées pertinentes d'évaluation sont récupérées, toutes les pratiques sont centralisées et il est possible d'en évaluer seulement les plus pertinentes. Au final, un gain de temps évident est acquis par rapport à une évaluation "traditionnelle" sur papier.
Les parties suivantes présenteront en quoi les autres évolutions peuvent aussi répondre aux besoins formulés dans le contexte de l'évolution venant d'être présentée. En particulier, les évolutions concernant les résultats d'analyse répondront encore davantage aux besoins réels des entreprises.

6.2.3.3 Ajout d'une méthode d'évaluation supplémentaire

Contexte et besoins Lors de la section traitant de l'amélioration du processus d'évaluation, la figure 6.14 montrait qu'il était possible de choisir parmi deux méthodes d'évaluation :

- la méthode *Standard* permettant d'évaluer sans restriction l'itinéraire choisi ;
- la méthode S^{3m}*Assessment* stoppant l'évaluation d'un itinéraire si sa moyenne à un certain niveau de maturité est inférieure à 80%.

La première "méthode" permet simplement d'évaluer les pratiques à la guise de l'utilisateur. Cette première option est déduite du livre [April et Abran, 2006] même s'il ne définit pas réellement de méthode d'évaluation mais plutôt un suivi général de ce qu'il est possible de faire. Peu de contraintes donc concernant le processus d'évaluation, laissant une grande liberté à l'utilisateur mais pouvant également être risqué de part les conclusions finales qu'il pourrait tirer des résultats. Cette méthode serait plutôt à recommander pour un utilisateur averti.

Etant donné le manque de définition formelle d'une méthode d'évaluation, une méthode plus formelle et restrictive a été développée. L'objectif de cette présente évolution était d'analyser cette nouvelle méthode d'évaluation, d'en tirer des exigences pour l'outil et de valider son intégration à $S^{3mAssess®}$. Tout cela en préservant bien entendu les fonctionnalités déjà existantes et la stabilité de l'application. Le diagramme de la figure 6.17 illustre les étapes de cette méthode d'évaluation.

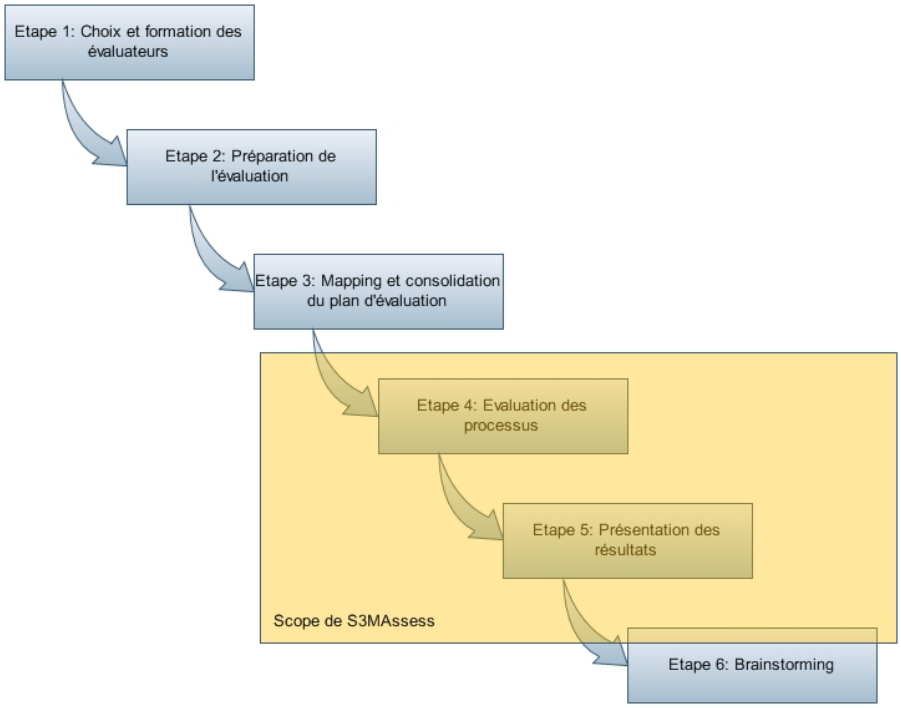


FIG. 6.17 – Les grandes étapes de la méthode $S^{3m}Assessment$

L'exigence principale qu'il a directement été possible de tirer de cette méthode est qu'il fallait bloquer le passage à l'évaluation des pratiques d'un niveau supérieur si les pratiques du niveau courant n'atteignaient pas le seuil de 80%. Le blocage de l'évaluation pour ce seuil permet d'éviter de consommer des ressources d'évaluation inutilement car il est recommandé d'obtenir un niveau de capacité *correct* (c'est à dire dont la moyenne égale au moins le seuil de 80%) pour un niveau inférieur avant de commencer à améliorer la capacité du niveau supérieur. Sinon, les éventuels efforts fournis en vue d'améliorer les niveaux supérieurs, en atteignant les objectifs spécifiques liés à ces niveaux, pourraient s'effondrer car les bases ne seraient pas suffisamment solides (voir notamment l'article [Adamonis *et al.*, 2005]). Les sous-sections 6.2.3.5 et 6.2.3.6 reviendront sur cette problématique.

Ensuite, la méthode propose deux types d'histogrammes de résultats comme l'illustrent les deux figures 6.18 et 6.19 issues de la présentation de cette méthode.

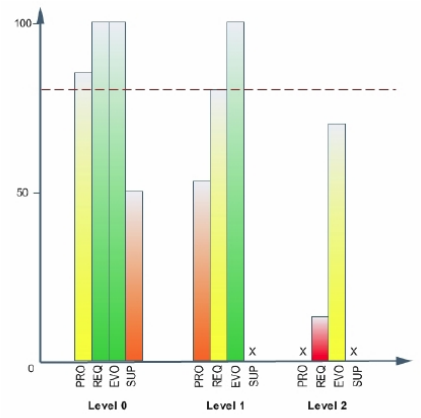


FIG. 6.18 – Exemple d'un histogramme général (*issu de la présentation de* $S^{3m}Assessment$)

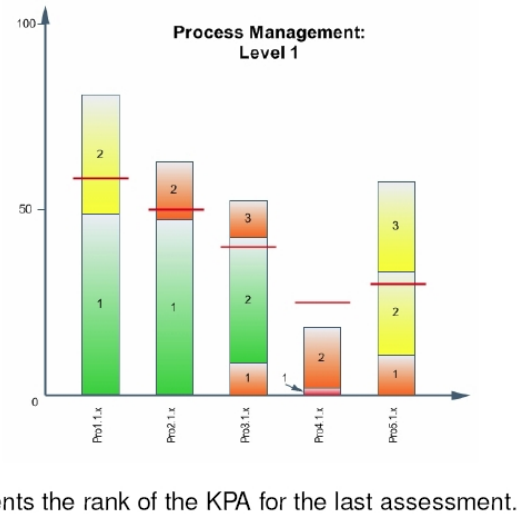


FIG. 6.19 – Exemple d'un histogramme pour un domaine et un niveau (*issu de la présentation de* $S^{3m}Assessment$)

Néanmoins, le paragraphe suivant montrera que ces deux histogrammes ont été adaptés et ne sont donc pas présents tel quel dans l'application. Ceci afin de rendre des résultats plus intuitifs et communs quelque soit la méthode d'évaluation choisie et afin de proposer ensuite la meilleure stratégie d'amélioration possible sur base des résultats, notamment graphiques.

Solution implémentée Tout d'abord, il faut situer le domaine d'intervention de l'outil $S^{3mAssess®}$ au sein de la méthode $S^{3m}Assessment$. La figure 6.17 illustrait, dans le paragraphe précédent, les différentes étapes de la méthode mais volontairement, aucun commentaire n'avait été fait à propos de l'encadré orangé nommé "Scope de $S^{3mAssess®}$" car il s'agit d'un élément de solution. En effet, cet encadré délimite l'étendue de l'application au sein de la méthode. Plus exactement, l'outil supporte totalement les étapes d'évaluation et de présentation des résultats mais également une partie de l'étape de brainstorming. Cette dernière étape vise principalement à réunir le haut management de l'entreprise, de l'organisation de main-

tenance (s'il s'agit d'une division interne à l'entreprise ou d'un sous-traitant) et quelques mainteneurs pour prendre des décisions d'amélioration concernant les processus actuels. La sous-section 6.2.3.5 expliquera dans quelle mesure l'outil intervient lors de cette dernière étape.

Suite au développement de cette nouvelle méthode d'évaluation, il s'agissait donc d'en prendre connaissance afin de déterminer sur quels points l'outil pouvait intervenir et afin d'en tirer des exigences à intégrer. La première évolution importante à intégrer était la modification du processus d'évaluation. Il fallait que l'outil puisse empêcher d'évaluer les pratiques d'un itinéraire au niveau deux si celles du niveau un n'atteignaient pas le seuil de 80%. La figure 6.20 illustre un exemple de l'implémentation de cette solution.

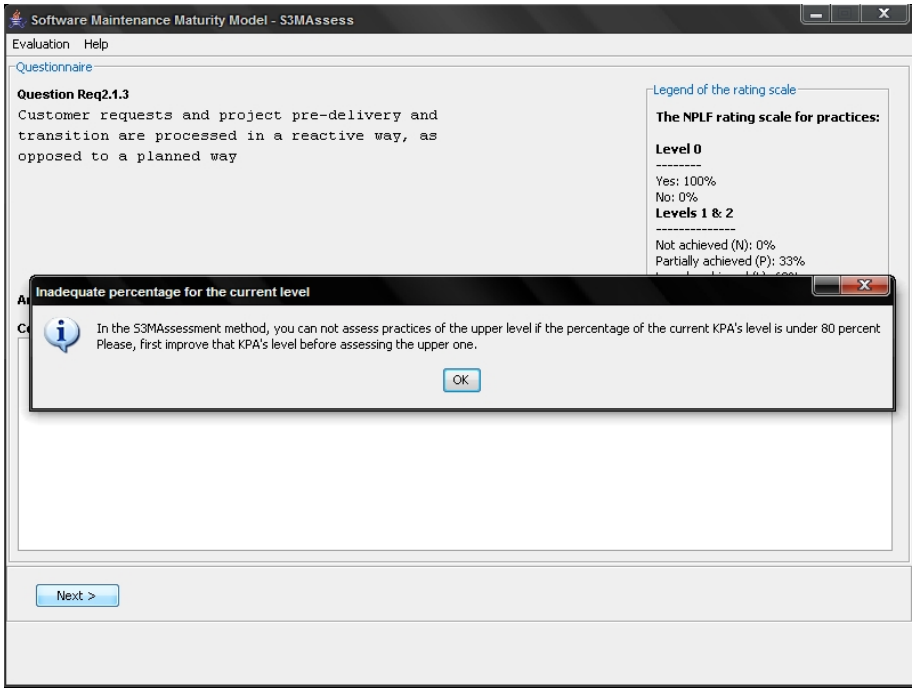


FIG. 6.20 – Exemple d'un processus d'évaluation bloqué par la méthode $S^{3m}Assessment$

L'enjeu était d'abord de pouvoir ajouter un nouveau procédé d'évaluation des pratiques sans pour autant compromettre l'existant de l'application.
Lorsque l'évaluation de l'itinéraire se trouve bloquée, l'utilisateur clique sur "ok" et revient sur l'écran de la figure 6.15.

Ensuite, concernant la représentation graphique, un premier histogramme (voir fig. 6.21) compare les quatre domaines de processus $S^{3m®}$ selon leurs trois niveaux de maturité $S^{3m®}$ d'une manière semblable à ce qu'illustrait la figure 6.18 :

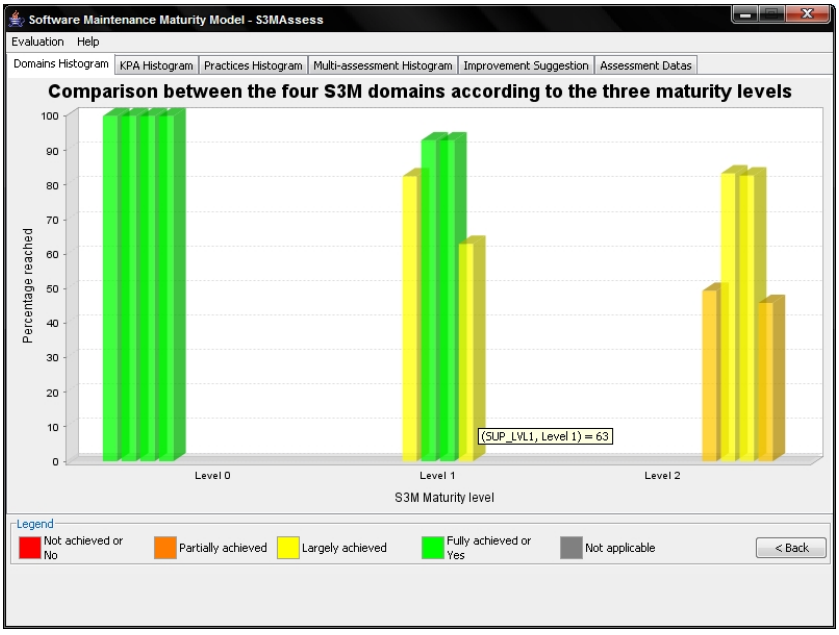


FIG. 6.21 – Comparaison des quatre domaines de processus $S^{3m®}$ dans $S^{3mAssess®}$

Avec ce graphique, l'utilisateur peut voir rapidement quels domaines devraient être améliorés et à quel niveau. Le but serait de commencer par améliorer les domaines les plus faibles à partir des niveaux inférieurs. La sous-section 6.2.3.5 détaillera une stratégie possible d'amélioration, sur base notamment des graphiques générés, ainsi que les détails mathématiques liés à ce diagramme.

En ce qui concerne le deuxième histogramme suggéré par la méthode, quelques changements ont été opérés afin de rendre le graphique plus lisible, plus accessible, plus intuitif et plus compréhensible :

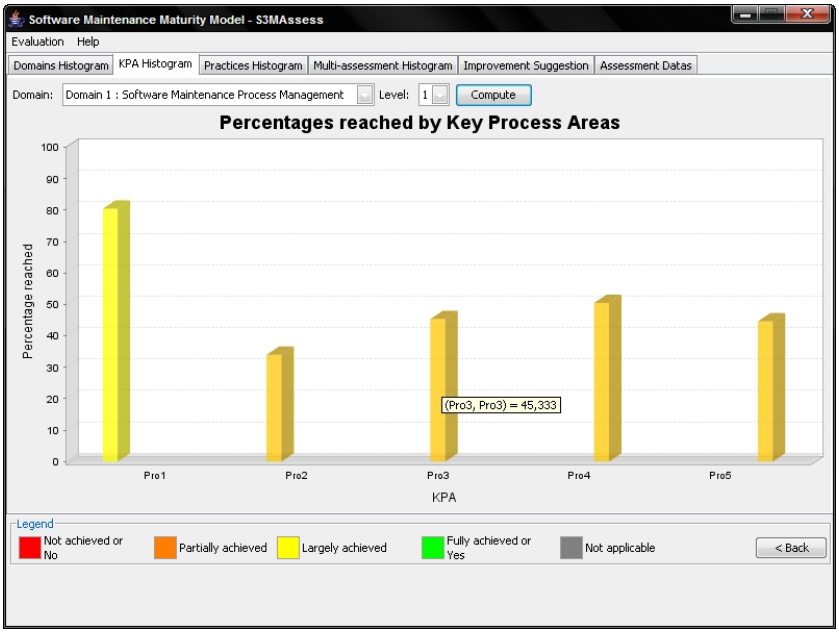


FIG. 6.22 – Comparaison des pourcentages atteints par les itinéraires dans $S^{3mAssess®}$

Ce diagramme permet d'obtenir les détails concernant les moyennes des itinéraires appartenant à un domaine et pour un niveau de maturité $S^{3m®}$ spécifique. En particulier, l'utilisateur peut connaître faci-

lement les itinéraires ayant une influence négative sur le pourcentage d'un domaine $S^{3m®}$. En plus d'offrir les avantages cités ci-dessus, ce diagramme permet d'intégrer la stratégie d'amélioration détaillée un peu plus loin dans ce chapitre.

Au final, le travail effectué quant à la manière de représenter les deux histogrammes suggérés par la méthode $S^{3m}Assessment$ permettra de faire ressortir des sous-ensembles de pratiques à améliorer. Ceci n'était pas solutionné par la méthode d'évaluation. Une étude a donc été menée afin de proposer à l'utilisateur une stratégie d'amélioration simple à suivre, complète et suggérant des sous-ensembles de pratiques faibles à améliorer en priorité. Ces sous-ensembles seront groupés par domaine et par niveau.
C'est dans ce contexte global d'amélioration que les graphiques ont été adaptés. Ils contribueront ainsi directement à l'un des objectifs premiers de l'outil consistant à être le plus adapté possible au contexte réel et pratique des entreprises.

6.2.3.4 Fournir des résultats complets et pertinents

Cette sous-section présente les objectifs de haut niveau concernant la problématique générale d'analyse des résultats d'évaluation. Les sous-sections suivantes détailleront par contre comment ces objectifs ont pu être concrètement rencontrés grâce à $S^{3mAssess®}$.

Contexte et besoins Une fois le processus d'évaluation achevé selon la méthode choisie, l'évaluateur veut pouvoir analyser les résultats de manière claire, rapide, cohérente et complète. L'un des gros avantages d'utiliser un logiciel pour effectuer une évaluation de processus se situe justement à ce niveau. Avec une évaluation faite sur papier, il faudrait rassembler les réponses et notes sur d'innombrables feuilles, calculer des moyennes à la main, dessiner des graphiques pour obtenir un visu plus explicite et/ou pour présenter les résultats au haut management, traiter des ensembles de nombreuses pratiques afin de les confronter et d'en tirer des pratiques faibles, etc. Il est donc évident que l'utilisation d'un logiciel automatisant ces tâches permet un énorme gain de temps, épargne des ressources nécessaires au processus d'évaluation et facilite l'analyse.
L'un des objectifs de ce mémoire était d'apporter une solution à cette problématique d'analyse finale devant extraire et proposer clairement le maximum d'informations en vue de démarrer un programme d'amélioration des processus de maintenance.

Solution implémentée Pour répondre à ces attentes, plusieurs solutions ont été étudiées, validées et intégrées à l'outil de support aux évaluations. Voici les principaux points d'intérêt de l'application à ce propos :

- Proposer des résultats graphiques directement après l'évaluation, automatiquement et selon plusieurs points de vue. Les résultats sont représentés d'une façon telle qu'une interprétation de leur signification et de leur valeur est rendue simple et directe. Différents graphiques sont proposés et sont paramétrables. De cette manière, l'utilisateur peut facilement avoir accès à l'information désirée quelque soit le niveau de détail recherché. Cela crée des vues de plus en plus détaillées pour la représentation graphique, allant des domaines jusqu'aux détails de chaque pratique.
  Il est ensuite également possible d'exporter tous les résultats, que ce soit les graphiques, les données d'évaluation, les suggestions d'amélioration, les moyennes, etc. L'utilisateur possède donc toutes les données nécessaires à la composition de rapports d'évaluation ou à la création de slides de présentation. Il s'agit encore là, bien évidemment, d'un énorme gain de temps et de facilité par rapport à une évaluation manuelle sur papier.
- La validité, l'utilisabilité et la comparabilité des résultats est également assurée. La validité est assurée par la correction du logiciel et, l'utilisabilité par la simplicité de l'interface et par la rétroaction proposée. Les résultats sont également comparables via la sélection de l'évaluation à analyser (ces

dernières sont toutes sauvegardées dans une BD de taille relativement importante) et via la fonction "multi-évaluation" (voir la sous-section 6.2.3.6).
- Les informations sont classées de manière à être utilisées ou à être distribuées de différentes façons. Lors de l'évaluation, les pratiques sont classées par domaines et itinéraires. Quant à l'analyse, la navigation dans l'outil se fait via des onglets proposant chacun une sémantique bien spécifique. De plus, il est possible de trier les pratiques lorsque l'utilisateur désire exporter les données d'évaluation dans une feuille de calcules.
- Les données sont conservées de manière sécurisée et confidentielle via la gestion de comptes utilisateurs.
- $S^{3mAssess®}$ a la capacité d'interagir avec d'autres outils via l'exportation des résultats dans une feuille de calculs.

Les particularités de l'outil présentées jusqu'ici lui permettent d'adhérer aux recommandations de la norme ISO 15504. Les détails seront donnés dans les sous-sections suivantes.

6.2.3.5 Aider l'entreprise à démarrer un processus d'amélioration

Contexte et besoins Il s'agit ici du sujet d'évolution le plus critique. L'application développée doit soutenir l'évaluateur tout au long de son processus d'évaluation afin qu'il puisse finalement en analyser les résultats. L'objectif d'une évaluation peut être de deux natures :
- Déterminer la capacité des processus de maintenance actuellement en place dans l'organisation de maintenance en vue de connaître ses forces/faiblesses ou celles d'une division de maintenance interne (c'est ce qui est appelé "auto-évaluation"), ou en vue d'évaluer un sous-traitant, une organisation externe ayant maintenu un logiciel que l'entreprise de l'utilisateur veut reprendre, ou encore un concurrent. Dans ces derniers cas, il s'agira d'une évaluation dite "indépendante" (voir la section 2.4.1).
- Améliorer des processus de maintenance. Grâce aux résultats d'analyse, l'entreprise peut déterminer ses faiblesses elle peut également vouloir démarrer un programme d'amélioration. Dès lors, comment passer des résultats d'évaluation à des idées claires d'amélioration ? Quels sont les pistes à suivre ? La méthodologie ? Les conséquences d'amélioration de certaines pratiques ?

A partir d'ici, l'utilité de l'outil prend une autre dimension. Il ne s'agit plus uniquement de systématiser un processus d'évaluation mais d'aider les entreprises à prendre des décisions pouvant avoir d'importantes conséquences. Dans ce cas, il faut donner le maximum d'informations pertinentes et fiables.

Il est également à noter que le but d'un modèle d'évaluation comme le CMMi ou le $S^{3m®}$ n'est pas de juger des personnes mais des processus. Les résultats ne devraient pas être détournés de leur objectif premier qui est de déterminer et/ou d'améliorer des processus. Cela incombe à l'utilisateur d'utiliser l'outil à bon escient. Malheureusement, il est difficile de contrôler comment le modèle et l'outil seront utilisés en pratique.

De plus, il est de la responsabilité de l'utilisateur d'utiliser ce modèle et cet outil afin d'atteindre ses objectifs d'affaire tout en tenant compte des coûts de l'organisation et des contraintes organisationnelles actuelles. Etant donné les incidences que l'amélioration d'une pratique peut avoir sur d'autres pratiques et les conséquences sur tout type de ressources, l'outil ne dictera jamais quels sont les processus à améliorer absolument et dans quel priorité. Pour cela, des discussions sont nécessaires sur base des contraintes pratiques et organisationnelles de l'entreprise mais également sur base des objectifs qu'elle s'est fixée. Par contre, il sera plutôt beaucoup plus pertinent d'attirer l'attention sur les principales faiblesses qui *devraient* être corrigées, de suggérer une méthode d'analyse des résultats permettant de déterminer soi-même où sont les faiblesses et de suggérer des sous-ensembles ordonnés de pratiques faibles. Le mot important ici

est donc la "*suggestion*".

Il ne serait pas sérieux d'affirmer d'avoir pu créer en quelques mois seulement un outil permettant à toute entreprise, ayant chacune son propre contexte, des objectifs, une philosophie, un budget, une organisation, etc. de dicter une voie obligatoire à suivre. D'autant plus que les évaluations ne doivent durer que quelques heures avec des ressources limitées.
Il faut aussi ajouter à cela qu'il peut subsister une certaine part de doute quant à la subjectivité des réponses données aux évaluations des pratiques, de la sincérité des participants à l'évaluation, de leur connaissance/expérience du modèle, de l'outil et de tous les processus évalués, de la compétence de l'évaluateur en chef, de la bonne définition du but de l'évaluation et de son contexte, etc.

L'outil devra pourtant limiter au maximum ces parts d'incertitudes, devra systématiser les tâches d'évaluation et d'analyse afin qu'elles soient *répétables* (voir les recommandations de la section traitant de la multi-évaluation) pour garantir la bonne définition de ce processus. Grâce à la bonne application du modèle S^{3m}®, de la méthode S^{3m}*Assessment* et donc grâce à un processus d'évaluation défini, répétable et documenté, l'outil pourra exprimer des résultats très précis et très critiques. L'outil sera alors un atout évident lors de l'analyse, lors de l'élaboration des rapports à remettre au sponsor de l'évaluation et finalement pour démarrer un brainstorming sur une base solide.
En voyant tout ce qu'implique l'amélioration de processus, il est compréhensible que l'outil "se limite" à la proposition d'une voie d'amélioration.

Un autre détail ayant son importance quant aux résultats est la façon de les représenter. Il existe dans le modèle CMMi deux types de représentations possibles : la représentation étagée et la représentation continue (voir [SEI, 2001a]). Le tableau 6.2 résume les caractéristiques de chacune d'elles.

Représentation continue	**Représentation étagée**
Permet une liberté explicite dans la sélection de l'ordre d'amélioration rencontrant au mieux les objectifs d'affaire de l'organisation et réduisant les risques	Permet aux organisations de suivre un chemin d'amélioration prédéfini et prouvé
Augmente la visibilité de la capacité atteinte dans chaque espace de processus individuel	Se concentre sur un ensemble de processus qui fournit à l'organisation une capacité spécifique caractérisée par chaque niveau de maturité
Permet des améliorations de différents processus réalisées à différentes allures	Résume les résultats d'amélioration en une forme simple : un seul numéro de niveau de maturité
Reflète une approche plus récente qui n'a pas encore de donnée démontrant qu'elle permet un retour sur investissement	Construit sur une expérience relativement longue qui inclut des études de cas démontrant un retour sur investissement

TAB. 6.2 – Comparaison des caractéristiques des représentations continues et étagées (adapté de [SEI, 2001a])

Une organisation ayant une bonne connaissance de ses objectifs d'affaire a très probablement établi un lien fort entre ses processus et ses objectifs d'affaire. La représentation continue peut être utile à ce genre d'organisation pour juger ses processus en déterminant jusqu'à quel point ils peuvent supporter et rencontrer ses objectifs d'affaire.
Si une organisation concentrée sur une ligne de produits décide d'améliorer les processus à travers l'organisation entière, elle pourrait être mieux servie par la représentation étagée. Ce type de représentation l'aidera à se concentrer sur les processus critiques qui devraient être améliorés.
La même organisation pourrait par contre opter pour l'amélioration des processus par ligne de produit.

Dans ce cas, elle pourrait choisir la représentation continue et différentes capacités pourraient être obtenues pour chaque ligne de produit.
Les deux approches sont valides. La considération la plus importante est de savoir quels objectifs d'affaires devraient être supportés par le programme d'amélioration et comment ceux-ci s'alignent avec les deux représentations.

Le modèle $S^{3m®}$, qui se base notamment sur le CMMi, opte plutôt pour la représentation continue. Son approche offre aux organisations la flexibilité de pouvoir choisir les processus sur lesquels elles veulent se concentrer. Toutefois, le modèle définit des ensembles de processus, appelés *itinéraires*, ce qui est habituellement une caractéristique de la représentation étagée puisqu'ils suggèrent un chemin d'amélioration à suivre. Il est tout à fait loisible pour une organisation d'évaluer tous les processus du modèle. A ceci, pourrait être attaché un niveau de maturité global grâce au calcul de la moyenne d'un domaine de processus pour un niveau de maturité $S^{3m®}$ donné. L'organisation pourrait ainsi rejoindre l'idée de la représentation étagée.
En fait, il n'existe pas réellement d'approche définie dans le modèle quant à la façon d'améliorer les processus. C'est pour combler ce manque que la solution implémentée et présentée dans le paragraphe suivant a été apportée via l'utilisation du logiciel de support.

Au final, même si l'approche initiale du modèle serait d'opter pour une représentation continue, l'utilisateur a encore le choix d'opter pour la représentation lui convenant le mieux. Cependant, étant donné la phase de mapping du diagramme de contexte de l'entreprise avec le diagramme théorique (voir la section 3.3 de la première partie du mémoire) proposé par le modèle et étant donné la classification des processus de maintenance propre à une entreprise, pouvant encore une fois diverger de celle proposée par le modèle, il sera très courant en pratique de voir les entreprises opter pour la représentation continue (voir un exemple avec l'étude de cas réalisée dans la partie trois du mémoire, chapitre 7). Cette dernière considération rejoint donc l'optique globale de la représentation continue suggérée par le modèle.

Solution implémentée Il a été mentionné ci-dessus que la représentation continue vise plutôt les entreprises qui ont bien défini leurs processus en rapport avec leurs objectifs d'affaire, qui ont une certaine expérience dans l'amélioration des processus ou, qui ont un besoin d'amélioration rapide d'un processus spécifique. Dans le cas où une entreprise ne répondant pas réellement à ces critères voudrait malgré tout opter pour l'approche continue, il serait intéressant de l'assister dans sa démarche. C'est précisément ce que vise la question de recherche matérialisée par ce paragraphe. La suite va décrire comment, sur base de tous les résultats d'évaluation, il est possible de suivre une voie menant à une idée plus précise d'amélioration.

La stratégie d'amélioration suit une démarche continue à travers une suite de graphiques paramétrables ayant chacun une sémantique propre. En voici un compte rendu :

1. Le premier graphique apparaissant dans l'analyse compare les quatre domaines $S^{3m®}$ selon les trois niveaux de maturité $S^{3m®}$. Les pourcentages atteints par chaque niveau de chacun des domaines sont calculés suivant la somme des moyennes des itinéraires divisée par le nombre d'itinéraires évalués pour ce domaine à ce niveau. Sous forme mathématique, cela donne les formules suivantes :

$$domainPercentage(d,l) = \frac{\sum_k kpaAssessedMean_{domain_d,level_l}}{\#kpaAssessed_{domain_d,level_l}} ,$$
avec $1 \leq d \leq 4$, $0 \leq l \leq 2$ et $1 \leq k \leq 18$.

et où,
$$kpaAssessedMean_{domain_d,level_l} = \frac{\sum valueOfPracticeAssessedAsPraticable_{kpa_k,level_l}}{\#practiceAssessedAsPraticable_{kpa_k,level_l}} ,$$
avec $0 \leq l \leq 2$ et $1 \leq k \leq 18$.

Le pourcentage moyen d'un domaine est donc calculé sur base des moyennes de ses itinéraires évalués et non en faisant la moyenne de toutes ses pratiques. Ceci se justifie par le fait qu'un itinéraire comportant davantage de pratiques qu'un autre ne doit pas pour autant être surpondéré. Tous les itinéraires ont le même poids.
A noter enfin que les pratiques évaluées comme "Non Applicable" ne sont pas considérées dans les calculs des moyennes.

La figure 6.23 montre un exemple du domaine *Support to Software Evolution Engineering* obtenant une moyenne de 63% pour ses itinéraires du niveau un.

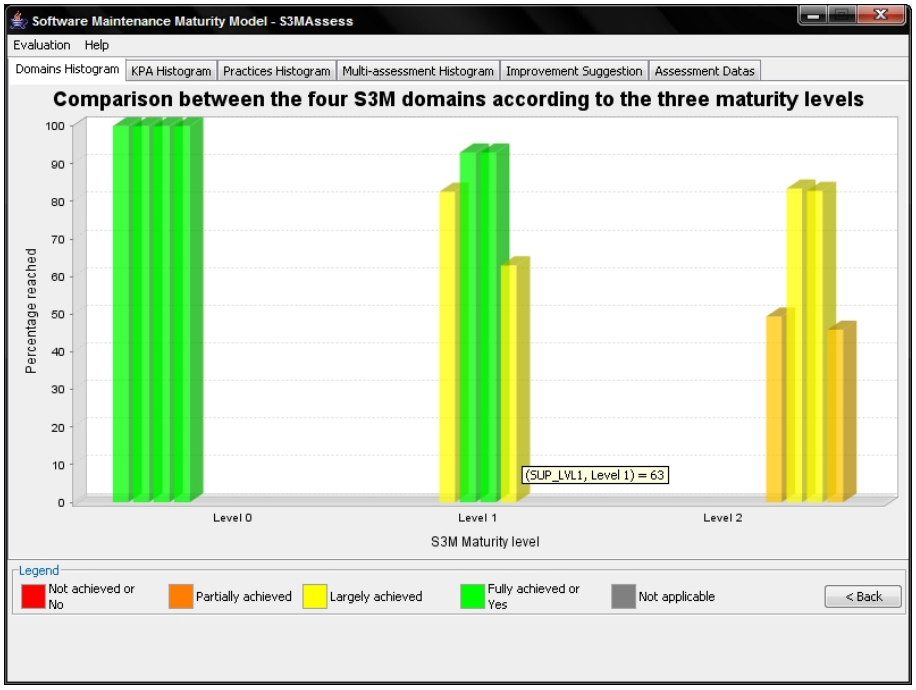


FIG. 6.23 – Comparaison des quatre domaines $S^{3m®}$ selon les trois niveaux de maturité $S^{3m®}$

Une politique intéressante liée à une représentation continue serait de commencer par obtenir un bon pourcentage (80% par exemple) pour un niveau avant d'améliorer le niveau supérieur. Grâce à ce résultat, l'utilisateur peut voir rapidement quels domaines devraient être améliorés en premier lieu et à quel niveau. Attention au fait que le seuil cité ici de 80% pour les niveaux d'un domaine n'est qu'un exemple de recommandation et n'a rien à voir avec le seuil de 80% définis pour les niveaux de chaque itinéraire venant de la méthode $S^{3m}Assessment$. Ce sont deux moyennes bien distinctes.

2. Lorsque les domaines les plus faibles ont été déterminés lors de l'étape précédente, l'utilisateur peut directement obtenir visuellement, grâce au diagramme de la figure 6.24, les itinéraires rendant un domaine faible. A cette fin, il peut sélectionner le domaine et le niveau de maturité à analyser.

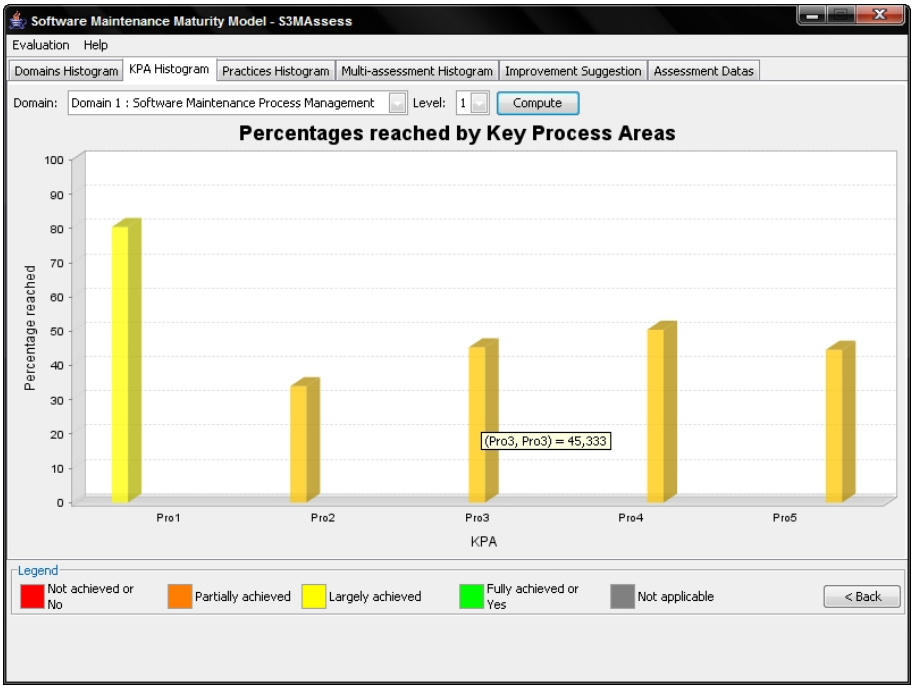


FIG. 6.24 – Pourcentages atteints par les itinéraires

A partir de là, l'utilisateur aura déterminé les niveaux les plus faibles des domaines et aura connaissance des itinéraires créant ces faiblesses. Ce qui l'intéresse désormais, c'est de voir quelles pratiques, issues de ces itinéraires faibles, ne sont pas bien réalisées et d'en obtenir des détails. Pour cela, il sélectionne l'onglet suivant dans l'application.

3. En sélectionnant le "profil des pratiques", l'utilisateur peut choisir un domaine à analyser dans le but d'obtenir les détails de toutes ses pratiques ordonnées par itinéraire et par niveau. Dans ce graphique, il retrouve des informations sur l'identifiant de la pratique, les commentaires laissés durant l'évaluation et la valeur donnée via un code couleur NPLF (pour **N**ot/**P**artially/**L**argely/**F**ully Achieved) intuitif et identique tout au long de l'outil. Avec ces détails, il obtient les pratiques faibles causant la faiblesse d'un itinéraire d'un domaine.

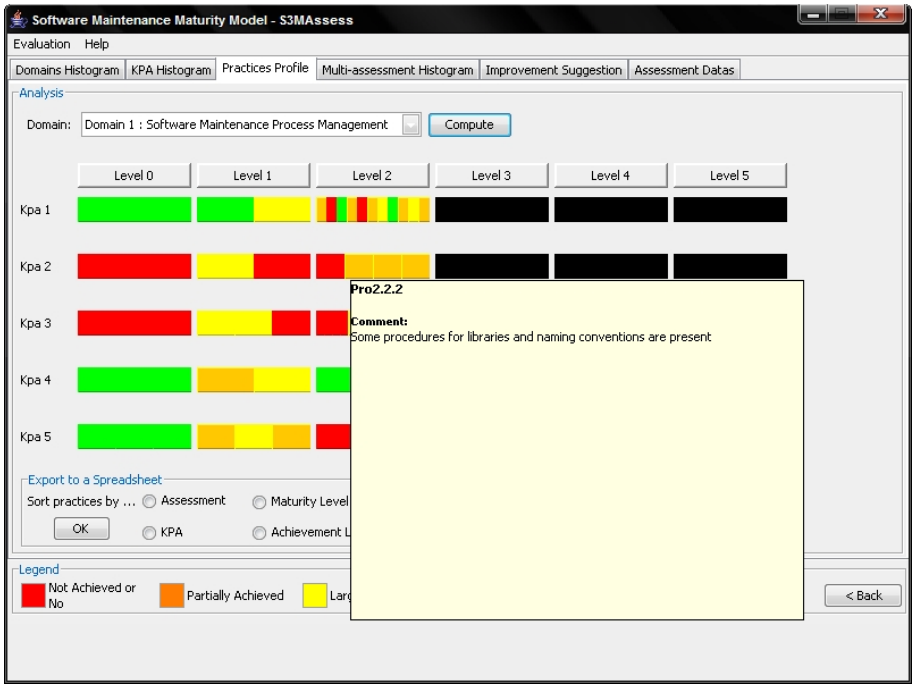


FIG. 6.25 – Représentation du profil de maturité : profil des pratiques

Il est également possible d'exporter toutes les informations d'évaluation et les résultats dans une feuille de calculs. L'utilisateur peut ordonner l'apparition des pratiques dans cette feuille selon l'ordre d'évaluation, par niveau de maturité, par itinéraire et par résultat. Cette possibilité lui facilite l'analyse et la rédaction des rapports finaux.

Pour rappel, voici l'échelle de conversion des réponses données à chaque pratique :
- Réponses du niveau zéro :
  - Yes : 100%
  - No : 0%

- Réponses des niveaux un et deux :
  - Not Achieved : 0%
  - Partially Achieved : 33%
  - Largely Achieved : 68%
  - Fully Achieved : 93%

4. L'onglet suivant reprend l'histogramme de la multi-évaluation. Etant donné que cette fonction fera l'objet d'une section à part entière, elle ne sera pas détaillée ici.

5. L'onglet *Improvement Suggestion* suggère des pratiques qui devraient être améliorées prochainement. Comme déjà précisé dans la description du contexte, le but ici n'est pas de définir des pratiques devant être absolument améliorées coûte que coûte mais suggère des pratiques faibles se situant en-dessous de la moyenne du niveau du domaine auquel elles appartiennent. Avec ces sous-ensembles de pratiques, l'utilisateur pourra commencer un brainstorming et décider en fonction du contexte, des objectifs, du budget, des ressources, etc. de l'entreprise quelles devront être au final les prochaines améliorations.

   La terminologie donnée aux pourcentages des domaines suit cette échelle :

- "Not Achieved" si le pourcentage est inclus dans [0 ;15]
- "Partially Achieved" si le pourcentage est inclus dans [16 ;50]
- "Largely Achieved" si le pourcentage est inclus dans [51 ;85]
- "Fully Achieved" si le pourcentage est inclus dans [86 ;100]

La politique à suivre est d'obtenir des niveaux uniformes au sein des quatre domaines $S^{3m®}$ sans avoir des pratiques très matures avec d'autres trop faibles. Les pratiques les plus faibles devraient être améliorées afin d'éviter de fortes instabilités dans le processus de maintenance de l'organisation. Pour cela, l'utilisateur devrait commencer par améliorer les pratiques des niveaux inférieurs jusqu'à ce qu'il obtienne de bons pourcentages (par exemple 80%) pour ces niveaux au sein d'un domaine. Ceci est bien évidemment vrai pour une approche étagée mais aussi pour la représentation continue puisque dans ce cas, l'utilisateur décide de se concentrer uniquement sur certains itinéraires de processus et qu'il lui faut donc garder de la stabilité dans ce choix.

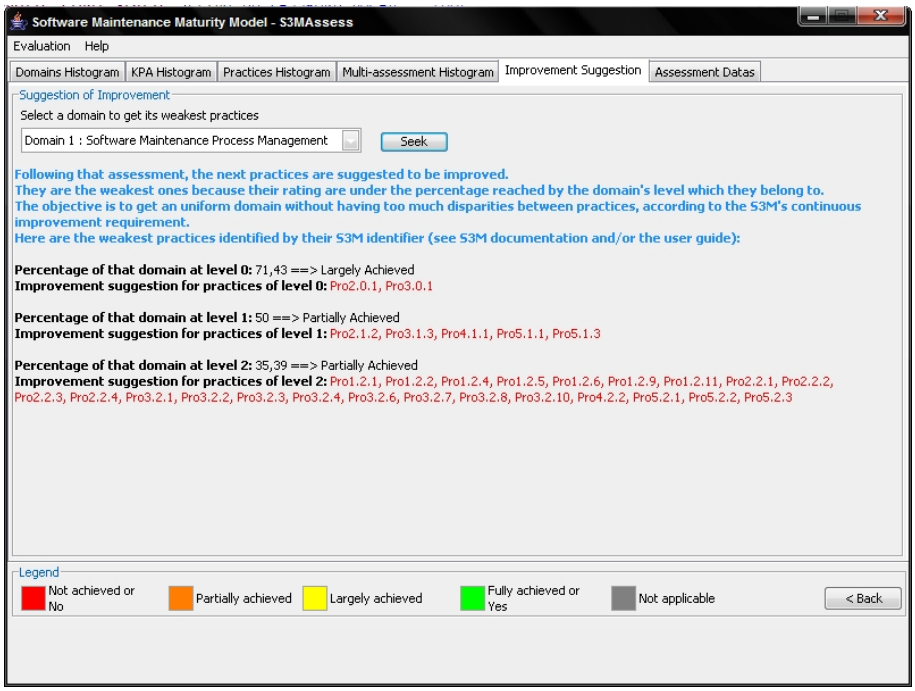


FIG. 6.26 – Suggestions d'amélioration

En résumé, cette analyse extrait un sous-ensemble de pratiques pour faciliter l'utilisateur dans sa décision d'amélioration. Ainsi, des sous-ensembles de pratiques faibles venant de différents itinéraires sont extraits afin de fournir un maximum de support. Avec tous les sous-ensembles de pratiques faibles, un brainstorming peut commencer sur une base solide et claire pour discuter des améliorations possibles.
De plus, il est aussi envisageable d'entrer ces sous-ensembles dans un autre outil de support basé sur un système d'aide à la décision. Il existe déjà un tel outil basé sur le modèle $S^{3m®}$: S3MExpert. Avec la génération de la feuille de calculs, le lien entre les deux outils est dès lors rendu possible.

6. Finalement, l'onglet *Assessment Datas'* donne accès aux informations générales de l'évaluation que l'utilisateur a entrées au début de son processus d'évaluation et aux détails de chaque pratique : son identifiant $S^{3m®}$, sa cotation, son pourcentage de réalisation et son commentaire. La figure 6.27 montre un exemple de détails.

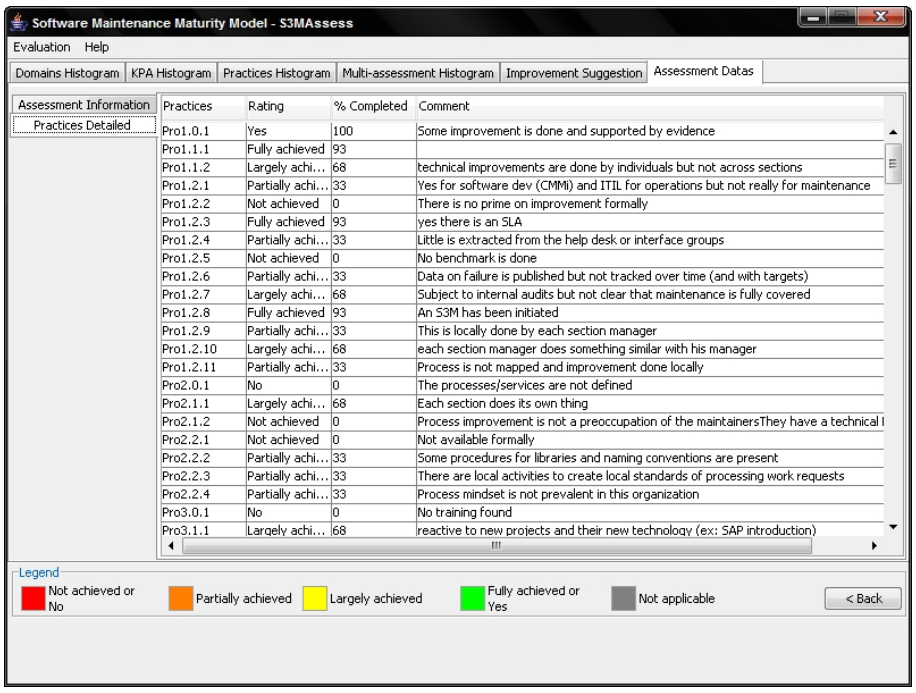


FIG. 6.27 – Données d'évaluation

En conclusion, la stratégie d'amélioration poursuit une logique selon trois graphiques principaux :
- Le premier compare les domaines des niveaux de maturité zéro à deux. Le but de l'amélioration est de commencer par les niveaux inférieurs afin d'obtenir une base solide du processus de maintenance. Avec ce graphique, on voit directement quel est la maturité des divers domaines et sur lesquels il faut se focaliser en premier lieu.
- Une fois les domaines les plus faibles identifiés, on peut observer quels itinéraires sont les plus faibles dans ces domaines via l'histogramme des itinéraires. Le mieux serait d'améliorer les itinéraires les plus faibles en commençant du niveau zéro jusqu'au niveau deux.
- Lorsque les itinéraires les plus faibles et les plus prioritaires à améliorer sont identifiés, le troisième graphique permet d'obtenir les détails des pratiques par domaine, par itinéraire et par niveau. Ainsi, une fois que le domaine et ses itinéraires sont jugés à améliorer, on analyse dans le profil des pratiques quelles en sont les pratiques les plus faibles et donc, sur lesquelles il faut d'abord se pencher.
- Afin d'aider à dégager les sous-ensembles de pratiques faibles, l'onglet "Improvement Suggestion" propose automatiquement des sous-ensembles de pratiques faibles par domaine et par niveau. L'utilisateur peut alors se pencher principalement sur ces pratiques, les confronter avec ses observations personnelles suite à ses analyses des graphiques et finalement débuter une séance de brainstorming en ayant déjà une bonne première idée des faiblesses du processus de maintenance à corriger au plus vite.

6.2.3.6 Supporter un processus multi-évaluation

Contexte et besoins Cette évolution matérialise la nécessité de gérer un historique d'évaluations et en particulier, d'analyser ce qui a changé entre deux évaluations successives. En effet, entre deux évaluations successives, des ressources ont été investies en vue de permettre une phase d'amélioration. Les questions lors de la seconde évaluation sont alors de savoir si l'entreprise a réellement gagné en maturité, sur quels processus et si l'amélioration d'un ensemble de processus n'a pas mené à l'affaiblissement ou à l'amélioration d'autres processus. Il faut en effet savoir qu'il existe une hiérarchie de dépendances entre certains niveaux de capacité des processus (voir [Adamonis *et al.*, 2005]).
L'étude de ces dépendances est particulièrement utile dans le cas de la représentation continue puis-

qu'aucun chemin d'amélioration n'est défini. En améliorant des processus qu'elle choisit par elle-même, l'entreprise risque d'être contrainte par d'autres processus dépendants. La capacité d'un processus particulier dépend d'un environnement de performance composé par d'autres processus. Un processus ne peut atteindre une capacité cible sans le support d'autres processus exécutés à un niveau adéquat.
Pour illustrer ce fait, voici un exemple simple repris de l'article [Adamonis *et al.*, 2005]. Pour atteindre un certain niveau de capacité pour les processus du cycle de vie primaire, une organisation a besoin de réaliser du management de projet (faisant partie des processus du cycle de vie organisationnel) et des processus de support (documentation, audit, revue, management de configuration, assurance qualité, résolution de problème). Par conséquent, si les processus du cycle de vie primaire atteignent le deuxième niveau de capacité (du modèle de référence de la partie deux de la norme ISO 15504, voir [ISO/IEC, 1998]), alors les processus du cycle de vie de l'Organisation et du cycle de vie de Support doivent être réalisés (par exemple au premier niveau de capacité dans la figure 6.28).

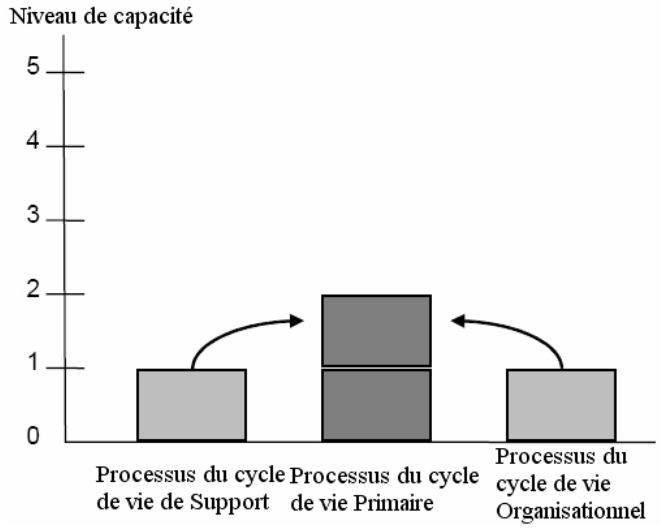


FIG. 6.28 – Processus gérés

Dans tout modèle de maturité, donc également dans le $S^{3m®}$, il existe des associations entre différents processus, entre des niveaux de capacité et des processus et, entre des caractéristiques de performance de pratique et des pratiques de base. Il faut bien être conscient que les niveaux de capacité des processus ne peuvent être atteints indépendamment des processus avoisinants. Il serait donc utile d'ajouter au modèle $S^{3m®}$ une hiérarchie des niveaux de capacité des processus. A défaut de son existence, la présente évolution visera à analyser les impacts des dépendances entre les pratiques. En attendant l'éventuel développement de cette hiérarchie, les organisations devront donc analyser elles-mêmes les dépendances d'une pratique avec d'autres pratiques, grâce à des évaluateurs expérimentés dans les modèles de processus ou grâce à des personnes ayant bonne connaissance du modèle $S^{3m®}$.

Par ailleurs, la section 2.4.3 définissait un cycle de multi-évaluation. Pour rappel, le diagramme de cette section définissait ce cycle de la manière suivante :

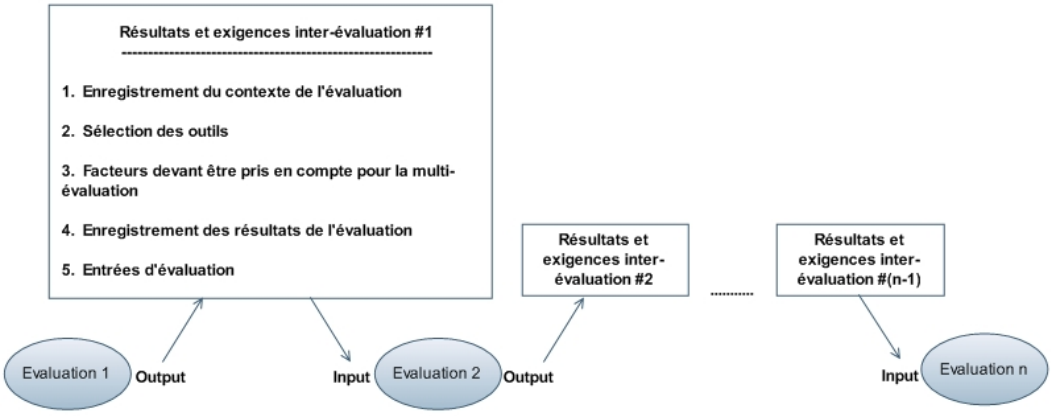


FIG. 6.29 – Le cycle de la multi-évaluation

Bien que placé dans la première partie de ce mémoire, ce diagramme fut en fait l'objet d'une recherche personnelle mais largement aidée par les recommandations de la norme ISO 15504. Ainsi, l'exigence inter-évaluation fut de fournir, en sortie de toute évaluation, la liste des cinq éléments suivants afin de définir un processus multi-évaluation répétable, documenté, géré, cohérent, fiable et complet :

1. L'enregistrement du contexte de l'évaluation ;
2. La sélection des outils ;
3. Les facteurs devant être pris en compte pour la multi-évaluation ;
4. l'enregistrement du résultat de l'évaluation ;
5. Des entrées d'évaluation.

Le paragraphe suivant détaillera comment l'outil permet de respecter ces exigences.

Finalement, l'important dans cette évolution est de prendre en compte le fait que dans un processus d'amélioration continue, des ressources sont investies et que des groupes de processus peuvent être dépendants. L'amélioration d'un processus peut donc mener à l'affaiblissement d'autres processus ou être tout simplement rendue impossible de part la nécessité d'atteinte d'un certain niveau de capacité des processus dépendants.

Il n'existe pas encore de diagramme exprimant la hiérarchie des dépendances entre les pratiques du modèle $S^{3m®}$. L'utilisateur doit donc en être conscient et les rechercher par lui-même. Afin de situer l'évolution de ses processus par rapport à la dernière évaluation, il a besoin d'analyser les "gaps", ou "variations", entre deux évaluations successives. Cette analyse est d'autant plus importante qu'il devra probablement justifier les ressources qui ont été investies. C'est pourquoi, la fonction de *multi-évaluation* présentée ici prend toute son importance dans une analyse des résultats d'évaluation.

Solution implémentée Ce paragraphe va donc présenter comment les exigences détaillées ci-dessus ont été solutionnées et implémentées au sein de $S^{3mAssess®}$. Tout d'abord, la fonctionnalité sera présentée. Ensuite, l'apport de l'outil par rapport aux exigences du cycle de la multi-évaluation, issues de la norme ISO 15504, sera détaillé.

La fonction de multi-évaluation se trouve dans l'onglet *Multi-assessment Histogram* et permet à l'utilisateur d'observer sa progression depuis la dernière évaluation réalisée pour un domaine, un itinéraire et

un niveau choisi. Pour cela, $S^{3mAssess®}$ gère un historique de toutes les évaluations réalisées. Grâce au graphique résultat, il est facile de voir quelle est la progression et les conséquences des ressources investies pour une amélioration depuis l'évaluation précédente.

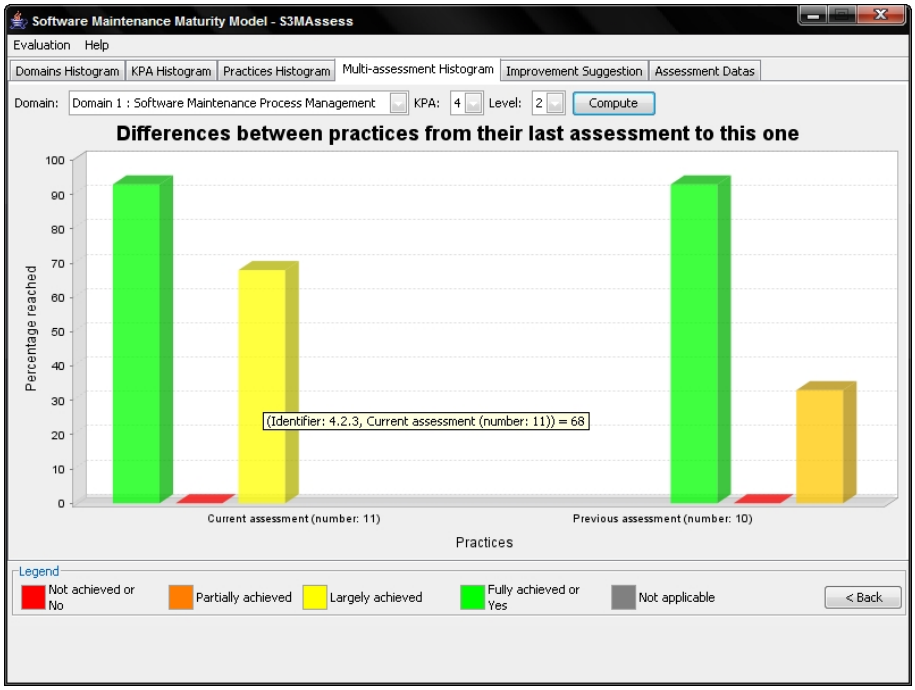


FIG. 6.30 – Histogramme de la multi-évaluation

La figure 6.30 illustre l'exemple de la progression d'une pratique identifiée par l'identifiant $S^{3m®}$ *Pro4.2.3* passant d'une réalisation partielle (33%) à une réalisation largement atteinte (68%) entre deux évaluations successives, c'est à dire de la dixième vers la onzième évaluation.
Il faut appuyer sur le fait que ces analyses de deltas ne sont pertinentes que pour deux évaluations successives, c'est à dire réalisées en temps t-1 et t. En effet, le but est d'observer les conséquences de l'allocation des ressources sur les pratiques pour l'amélioration venant d'être effectuée. Si la comparaison se fait entre plusieurs phases d'amélioration, il est impossible de connaître exactement les éléments ayant fait varier une pratique car trop de facteurs entrent en ligne de compte.

Après avoir présenté la fonctionnalité et son utilité, la suite détaille comment l'outil permet de rencontrer les exigences du cycle de la multi-évaluation basées sur la norme ISO 15504.

1. **L'enregistrement du contexte de l'évaluation**
   Le contexte est important car il influence l'évaluateur dans sa façon de coter les pratiques. Cette toute première exigence est d'abord rencontrée en dehors de l'outil lors principalement de la description du diagramme de contexte et lors de la classification des processus de maintenance de l'entreprise. Une fois cette discussion réalisée et ainsi documentée, l'utilisateur de $S^{3mAssess®}$ commence toute nouvelle évaluation par l'étape obligatoire de l'entrée des données générales sur l'évaluation. Ces informations reprennent le nom de la compagnie évaluée, le nom de l'évaluateur en chef, la date du jour, la durée de l'évaluation, le département évalué, la taille de l'équipe conduisant le processus d'évaluation, l'objectif de l'évaluation, son sponsor, la relation avec le sponsor et la méthode d'évaluation suivie. L'ensemble de ces données permettent de fixer les bases du contexte de l'évaluation et sont récupérables automatiquement à l'issue du processus.

2. **La sélection des outils**
   La norme ISO 15504 recommande que dans toute évaluation, les informations ont besoin d'être obtenues, enregistrées, sauvegardées, comparées, traitées, analysées, récupérées et présentées. Alors qu'en général, un processus d'évaluation documenté est supporté par divers outils pour atteindre ces différents objectifs, $S^{3mAssess®}$ permet de réaliser tous ces besoins à lui seul. Comme cette deuxième partie du mémoire l'a démontré, l'outil fait face au volume et à la complexité des informations d'évaluation. Il permet la réalisation d'évaluations cohérentes et fiables, de réduire la subjectivité de l'évaluateur et d'aider à s'assurer de la validité, de l'utilisabilité et de la comparabilité des résultats d'évaluation. L'application atteint principalement ces objectifs en rendant disponible à l'utilisateur le modèle $S^{3m®}$ et des indicateurs comme l'échelle de cotation des pratiques.
   L'outil de support peut-être utilisé selon les recommandations précisées au point deux de la section 2.4.3.
   De plus, il est recommandé que l'outil soit utilisé par une personne compétente. Il faut savoir que $S^{3mAssess®}$ ne peut-être utilisé que par des personnes en ayant fait la demande. Il serait donc tout à fait envisageable de n'accorder l'accès à l'outil qu'aux personnes ayant démontré leur compétence.

   Etant donné que $S^{3mAssess®}$ veut faire partie intégrante d'un processus d'évaluation, voici quelques exemples explicites illustrant concrètement son apport (listing basé sur les recommandations du point deux de la section 2.4.3, *Sélection des outils*) :
   - Il a la capacité de capturer et de maintenir des informations de support comme définis dans les entrées d'évaluation grâce aux informations demandées au début de toute nouvelle évaluation.
   - Il a la capacité de stocker et de récupérer les résultats d'évaluation pour une utilisation ultérieure dans un processus d'amélioration ou de détermination de capacité grâce notamment à la possibilité de continuer une ancienne évaluation (évolution non présentée dans cette section mais inclue dans le guide utilisateur en annexe).
   - L'outil met à disposition une ségrégation appropriée des différentes classes d'informations et des données permettant de les utiliser ou de les distribuer de différentes manières et il a la capacité d'être divisé par processus et par fonction. En effet, il réalise ces exigences grâce à des processus d'évaluation et d'analyse ordonnés, dont la structure en onglets de sa partie *Analyse*.
   - L'outil a la capacité de garder de manière sécurisée les informations obtenues afin de rencontrer des contraintes de confidentialité. Il a également la capacité de manipuler de multiples entrées d'évaluateurs différents. Grâce à la gestion de comptes utilisateurs, les profils de résultats restent confidentiels et sont donc accessibles uniquement par leurs auteurs.
   - Il a la capacité d'exécuter dynamiquement des étendues et des réductions afin de supporter des besoins culturels, organisationnels, de sponsor ou d'évaluations spécifiques. Ceci est atteint avec la mise à disposition du choix du domaine, de l'itinéraire et des pratiques à évaluer. Une fonction permettant de continuer une ancienne évaluation a également été implémentée.
   - Il a la capacité de proposer un contrôle de configuration adéquat de l'outil et des résultats. Cette recommandation est réalisée notamment via le choix de la méthode d'évaluation à suivre, via le paramétrage des différents graphiques offrant à chaque fois une vue différente et via la possibilité de trier l'ordre d'apparition des pratiques dans la feuille de calculs qu'il est possible de créer.
   - Il atteint des considérations de portabilité car il est écrit en java et car il est accessible via Internet à partir de n'importe quel ordinateur pourvu que celui-ci ait les autorisations d'accès nécessaires.
   - Il offre l'opportunité d'interagir avec d'autres outils grâce à ses possibilités d'exportation des résultats sous forme de feuille de calculs ou de fichiers images.

3. **Les facteurs devant être pris en compte pour la multi-évaluation**
   Ces facteurs sont décris dans le point trois de la section 2.4.3. Ils peuvent être rencontrés via une utilisation correcte de l'outil et via l'application correcte de la méthode S^{3m}*Assessment*. Il n'est pas vraiment pertinent d'en ajouter davantage à ce sujet.

4. **l'enregistrement du résultat de l'évaluation**
   Les résultats sont toujours enregistrés en base de données et peuvent l'être sous les diverses formes déjà présentées si l'utilisateur le souhaite. Ces enregistrements permettent la comparaison des résultats entre diverses évaluations. Il est envisageable de les comparer entre des organisations différentes ou de tracer l'évolution de la maturité au sein d'une même organisation. Ainsi, un historique et des analyses de deltas sont possibles et aident les entreprises dans leurs prises de décision suite à leurs évaluations successives.
   Il faut également préciser que les résultats devront généralement être documentés et rapportés au sponsor de l'évaluation. Dans cette optique, l'outil propose des ensembles de profils de processus comme résultat principal d'une évaluation. C'est d'ailleurs dû à cette importance que la création des résultats a été corrigée dès la première révision de l'outil (voir la *Description des trois révisions de* $S^{3mAssess®}$ en annexe).

   Les exigences sur les résultats ont été réalisées comme suit :

   (a) Les informations pertinentes permettant la compréhension des résultats de l'évaluation sont rédigées par l'utilisateur durant son processus d'évaluation et restituées par l'outil dans le profil des pratiques, dans la feuille de calculs exportable et dans les informations d'évaluation.

   (b) L'enregistrement d'une évaluation comprend : sa date, ses entrées, l'identification des preuves objectives via les commentaires, la méthode d'évaluation choisie, l'ensemble des profils des processus et les éventuelles informations additionnelles si l'utilisateur les a entrées.

5. **Des entrées d'évaluation**
   Les résultats d'évaluation doivent être cohérents entre eux et permettre la justification des cotations afin d'augmenter la probabilité d'obtenir des cotations répétables pour les processus évalués. Pour cela, des entrées d'évaluation doivent être respectées. Il a déjà été question de ces quelques points à travers la présentation des évolutions. Le lecteur est renvoyé au point cinq de la section 2.4.3 pour obtenir le détail complet de ces points qu'il faut respecter en utilisant correctement $S^{3mAssess®}$ et $S^{3m}Assessment$.

La présentation de la solution concrète et des exigences inter-évaluation clôt la description de la fonction de multi-évaluation et de la présentation des principales évolutions. Le dernier point de cette deuxième partie du mémoire va maintenant résumer et conclure ce qui a été présenté tout au long de cette exploration. Tandis que la troisième partie de ce mémoire a pour objectif d'expérimenter en cas réels les solutions qui ont été apportées.

Conclusion de la partie II

La première partie de ce mémoire résumait les bases théoriques nécessaires à l'accomplissement de la création d'un outil de support aux évaluations de maturité des processus de maintenance du logiciel. Alors que ces bases théoriques étaient nécessaires, encore fallait-il étudier quels étaient les besoins réels des organisations de maintenance et comment les outils, normes et modèles appartenant au domaine de la maintenance pourraient répondre concrètement à ces attentes. Cette première approche permit de dégager des exigences et de les recouper avec des bonnes pratiques reconnues internationalement. Elle permit par ailleurs de faire état d'une version expérimentale d'un outil de support déjà implémenté. Malheureusement, celui-ci ne s'avéra pas opérationnel et sa maintenance était rendue ardue par la découverte de divers manquement et d'une architecture bien particulière. Toutefois, étant également basé sur le modèle d'évaluation de la maturité des processus de la maintenance $S^{3m®}$, il a été décidé de reprendre cet outil et d'y opérer un processus de maintenance. L'un des sujets d'étude de ce mémoire était alors lancé avec une méthodologie de maintenance à définir et à mettre en oeuvre pour repartir d'un outil déjà existant. Ceci impliquait diverses problématiques typiquement liées à ce domaine d'activité. L'occasion d'expérimenter concrètement un processus de maintenance était donc saisie.
Comme cette deuxième partie a pu l'exprimer, un processus de maintenance n'est pas simple et il peut comporter de nombreux risques s'il n'est pas suffisamment mature. D'où l'importance de disposer d'outils permettant à des organisations d'évaluer ce type de processus et ainsi de limiter au final les risques encourus.

Afin de synthétiser les objectifs de cette partie et d'en avoir une idée plus claire, voici les principales questions de recherches auquel il fallait répondre :

1. Comment mener à bien un projet de maintenance incluant une phase relativement importante de réingénierie, lorsque la documentation est largement insuffisante, lorsque les développeurs ne sont plus joignables, lorsque les clients sont pressés d'obtenir un résultat et quand l'effectif de l'équipe de maintenance est très réduit ?
2. Comment rendre l'outil expérimental tout à fait compatible avec la norme ISO/IEC 15504 ?
3. Comment améliorer le processus d'évaluation utilisé dans $S^{3mAssess®}$?
4. Rechercher comment fournir une analyse complète et pertinente qui facilite la prise de décision quant à l'amélioration des processus de maintenance ou qui facilite la détermination de la capacité des processus de maintenance.
5. Intégrer une nouvelle méthode d'évaluation dans l'outil tout en préservant sa cohérence et sa stabilité.

L'ensemble de ces questions devait trouver réponse tout en se focalisant sur les besoins réels des entreprises.

Un premier objectif du projet fut d'arriver à mener à bien la maintenance de l'outil expérimental initial en ayant recours notamment à des techniques de réingénierie. Il s'agissait d'en apprendre suffisamment sur le domaine d'application afin de pouvoir reconstruire l'outil.
Il vint ensuite l'objectif d'améliorer l'outil directement en fonction des besoins en entreprise. Pour cela, il fallait intégrer les DDC pouvant avoir des priorités différentes et pouvant être incompatibles entre elles. Ces questions ont pu être résolues en mettant en place un processus de maintenance spécifique à ce contexte

(voir le chapitre 6).
Une étude a ensuite été menée quant à la façon d'améliorer les résultats d'évaluation. Cette dernière déboucha sur la proposition de divers graphiques représentants autant de vues différentes, sur la création d'une fonction multi-évaluation permettant la comparaison directe des pratiques entre leurs deux évaluations successives et illustrant ainsi leur évolution, sur la proposition automatique de sous-ensembles de pratiques faibles et enfin, sur l'accessibilité aisée à toutes les informations d'évaluation pouvant par ailleurs être exportées sous divers formats. Ces différentes solutions complètent le modèle car seule une représentation à quatre couleurs y est prévue pour exprimer les résultats d'une évaluation.
Ces recherches donnèrent également lieu à la suggestion d'une stratégie d'amélioration proposant de commencer par améliorer les faiblesses trop importantes se situant dans les premiers niveaux de maturité $S^{3m®}$ et de monter successivement dans les niveaux.

Après avoir réalisé ces divers objectifs, la phase suivante vise à expérimenter en cas réels les aboutissants et à auto-évaluer le processus de maintenance de l'outil en l'utilisant lui-même à cette fin. A l'issue de ces diverses expérimentations, des idées d'améliorations et de travaux futurs seront proposés.

Troisième partie

Expérimentation du modèle $S^{3m_{\circledR}}$ et de son outil de support aux évaluations, $S^{3mAssess_{\circledR}}$

Cette troisième et dernière partie du mémoire est structurée selon trois chapitres fondamentaux. Après avoir présenté en détails l'outil $S^{3mAssess®}$, analysé sa maintenance et ses fonctionnalités de recherche, il est intéressant de prendre de la hauteur et d'expérimenter le travail réalisé.
Pour cela, les chapitres suivants auront pour objectif de :

- analyser une étude de cas confrontant le modèle et l'outil à un contexte réel d'entreprise ;
- appliquer une auto-évaluation, selon le modèle $S^{3m®}$, du processus de maintenance suivi dans ce projet ;
- envisager des travaux futurs.

Chapitre 7

Etude de cas 1 : Application du modèle et de l'outil dans une entreprise de sous-traitance

Ce chapitre vise à étudier l'application de l'outil $S^{3mAssess®}$. Ceci permettra de tirer des conclusions quant à son applicabilité, mais aussi quant à l'applicabilité du modèle, dans un contexte réel d'une entreprise de sous-traitance voulant optimiser son retour sur investissement. Grâce à cela, des éléments supplémentaires d'améliorations seront proposés dans le chapitre 9.

7.1 Cadre de l'étude de cas

Cette étude de cas démarra avec des questions soulevées par une entreprise, Siemens en l'occurrence (voir la brève description de cette société dans la *présentation des intervenants* en annexe), quant au modèle $S^{3m®}$. Cette dernière se demandait en quoi il consistait exactement, quelle était son étendue, de quelle façon était-il applicable et dans quels contextes.

Pour répondre à ces interrogations, un premier rendez-vous eu lieu avec un chef de projet (voir la brève présentation de Mr Huvelle dans la *présentation des intervenants* en annexe) d'une équipe de maintenance chez Siemens. Il est à distinguer d'emblée l'entreprise générale Siemens de son équipe de projet de maintenance pour laquelle s'applique exclusivement cette étude de cas.
Le but de ce premier entretien était d'obtenir des précisions quant aux attentes du chef de projet de cette équipe, de lui présenter les objectifs généraux poursuivis par le $S^{3m®}$, de lui suggérer l'existence de l'outil de support $S^{3mAssess®}$ et de la méthode d'évaluation $S^{3m}Assessment$. Ainsi, ce premier échange visait à déterminer si la poursuite de l'entente serait opportune.

Des premiers détails importants ont donc émergé :
- Ce chef de projet s'occupe de projets de maintenance applicative et applique toutes les catégories de la maintenance ;
- Son idée initiale est d'évaluer la maturité des produits et du service des futurs clients de l'organisation de maintenance pour essayer de déterminer le risque lié à divers projets proposés à Siemens et ainsi savoir s'il peut s'engager dans un projet ;
- Il se demande s'il existe une check-list ou une synthèse pour qualifier la maturité d'un client. Un exemple type de question qu'il se pose avant l'acceptation d'un projet est : *Est-ce que le client en question a un service de ticketing fiable et complet pour gérer les incidents ?*

Il apparaissait dès lors que l'intérêt du modèle n'avait pas été tout à fait bien perçu mais qu'il pourrait

quand même lui apporter une certaine aide. En effet, le $S^{3m®}$ vise d'abord à évaluer la maturité d'un organisme de maintenance avec pour but final d'en déterminer la maturité de ses processus de maintenance ou de les améliorer.
Le cas typique de l'utilisation du $S^{3m®}$ est l'auto-évaluation, c'est à dire une entreprise voulant connaître ses faiblesses afin de les améliorer. Cela demande une évaluation pertinente, fiable et surtout *objective*. C'est là que l'utilisation du modèle peut-être compliquée pour répondre aux questions de ce chef de projet. La correction des résultats d'un processus d'évaluation repose sur l'évaluation objective des pratiques $S^{3m®}$ et sur la compétence des évaluateurs, que ce soit à propos du modèle, de l'outil, de leur expérience dans l'amélioration des processus ou encore de leurs connaissances concernant les processus évalués. De cette manière, une évaluation $S^{3m®}$ fiable repose en partie sur les réponses données par les membres de l'organisation évaluée. Etant donné que le modèle vise à des évaluations de courtes durées et demandant peu de ressources, au contraire du modèle CMMi, des évaluateurs externes compétents et expérimentés en $S^{3m®}$ seront toujours un minimum dépendants des réponses données par les membres de l'organisation évaluée. Or, il peut être assez naïf d'imaginer ces éventuels clients jouer réellement le jeu en pointant sur toutes leurs faiblesses et risquant ainsi de faire augmenter le prix de leur projet de maintenance.
Pour éviter ce problème, il faudrait imaginer que des évaluateurs $S^{3m®}$ externes puissent aller auditer pendant plusieurs jours, voir semaines, les processus de maintenance en place. Or, ce n'est pas ce qu'aspire actuellement le modèle.
Par conséquent, il était difficile d'apporter des réponses suffisamment sûres et fiables quant à la sélection des projets les moins risqués en utilisant le modèle $S^{3m®}$. Néanmoins, il permet de cibler des classifications de processus via sa répartition en quatre domaines et en dix-huit itinéraires et de suggérer des points importants à vérifier. Le chef de projet pourrait alors vérifier des points pertinents en fonction des diverses situations clientes qu'il rencontre.

Après avoir bien compris ce que le modèle pouvait offrir et les attentes du chef de projet, il a été décidé de poursuivre l'entente en fixant un deuxième rendez-vous. L'entente se poursuivrait alors avec l'application du modèle $S^{3m®}$ et de l'outil $S^{3mAssess®}$ sur les processus de maintenance internes à l'équipe de projet chez Siemens. Ceci permettrait de fixer concrètement les idées sur les possibilités offertes par le modèle, sur son fonctionnement et d'étudier interactivement les pratiques. Suite à cela, le chef de projet aurait une idée plus précise du modèle et recevrait, par la même occasion, des idées sur les éventuelles faiblesses des processus de maintenance liés au projet étudié.

Ceci constitue bien entendu une opportunité d'appliquer $S^{3mAssess®}$ dans un cas concret et réel afin de valider ses évolutions et d'en déduire des possibilités d'évolutions futures grâce à l'apport d'un intervenant supplémentaire. De même, ce cas permettra de dégager des idées d'évolution du modèle lui-même.

7.2 Mise en place de l'étude de cas

Deux réunions ont donc eu lieu avec le management de l'équipe de projet de maintenance chez Siemens. Une première permit de présenter les objectifs réels du modèle $S^{3m®}$, de l'outil de support $S^{3mAssess®}$ et de la méthode d'évaluation $S^{3m}Assessment$. Elle permit également de recueillir les attentes du client et de confronter les objectifs afin de s'assurer que les deux parties iraient bien dans le même sens.

Suite à ce premier entretien, un second entretien eu lieu et visait à présenter le modèle $S^{3m®}$, l'outil $S^{3mAssess®}$ et la méthode $S^{3m}Assessment$ plus en détails. Cette première étape du deuxième entretien était vitale pour la cotation pertinente des pratiques $S^{3m®}$.
Une fois cette première étape réalisée, un processus complet d'évaluation utilisant l'outil $S^{3mAssess®}$ a été opéré. Ce processus s'est déroulé à quatre personnes : le chef de projet initiant l'évaluation, un mainteneur faisant partie de l'équipe du projet évalué, le développeur de la méthode $S^{3m}Assessment$ et le développeur de l'outil de support aux évaluations, $S^{3mAssess®}$. La durée des processus d'évaluation et de présentation

des résultats a été d'environ quatre heures.

Ces réunions ont aidé à échanger des informations et à augmenter la compréhension des pratiques $S^{3m®}$. Une fois l'évaluation réalisée, les résultats ont présenté les forces et faiblesses et le niveau de maturité de l'organisation.

Par ailleurs, la méthodologie de notation des pratiques a suivi les exigences du modèle $S^{3m®}$. Cependant, il faut préciser que les résultats de recherche de ce mémoire, et donc des évolutions par rapport aux exigences initiales du modèle, ont été appliqués dès le processus d'évaluation puisque :

- les pratiques du premier niveau de maturité (0) ont été évaluées dans leur forme positive et non négative comme c'est le cas dans la version actuelle du modèle. Cela a permis plus d'intuitivité dans la cotation de ces pratiques spécifiques ;
- il était possible de ne pas coter certaines pratiques en sélectionnant l'option "not applicable" lorsque celles-ci n'étaient pas pertinentes au contexte de l'organisation de maintenance. Cela permet de ne pas fausser les résultats finaux et d'être encore plus proche du contexte réel de l'entreprise, et donc plus pertinent.
- pour faciliter le calcul des pourcentages et pour réduire la subjectivité, une échelle de cotation a été établie (voir la sous-section 6.2.3.2 pour plus de détails).

Finalement, toute la partie *Analyse* de l'outil a été présentée : les résultats, la façon d'utiliser l'application pour découvrir les faiblesses dans les processus évalués et, l'utilité des divers onglets. Ces résultats d'analyse de l'évaluation de Siemens seront détaillés plus loin dans ce chapitre.

7.3 Analyse du contexte

Le modèle $S^{3m®}$ et l'outil $S^{3mAssess®}$ ont donc été appliqués pour évaluer la maturité d'un projet de maintenance réalisé par la société *Siemens*. Leur client est la direction générale d'une administration européenne. Leur projet de maintenance consiste à fournir un support aux applications déjà existantes, de maintenir ces applications et de développer de nouvelles fonctionnalités.

Avant de commencer une évaluation de maturité, il était nécessaire d'établir le diagramme de contexte reflétant la situation concrète du projet ou plus exactement, les interfaces existantes ainsi que les liens les unissant. En effet, l'établissement du contexte est important vu qu'il influence la façon d'évaluer les pratiques du modèle $S^{3m®}$. Afin d'y voir plus clair et avant de commencer l'évaluation de la maturité de ce projet, voici donc ce diagramme :

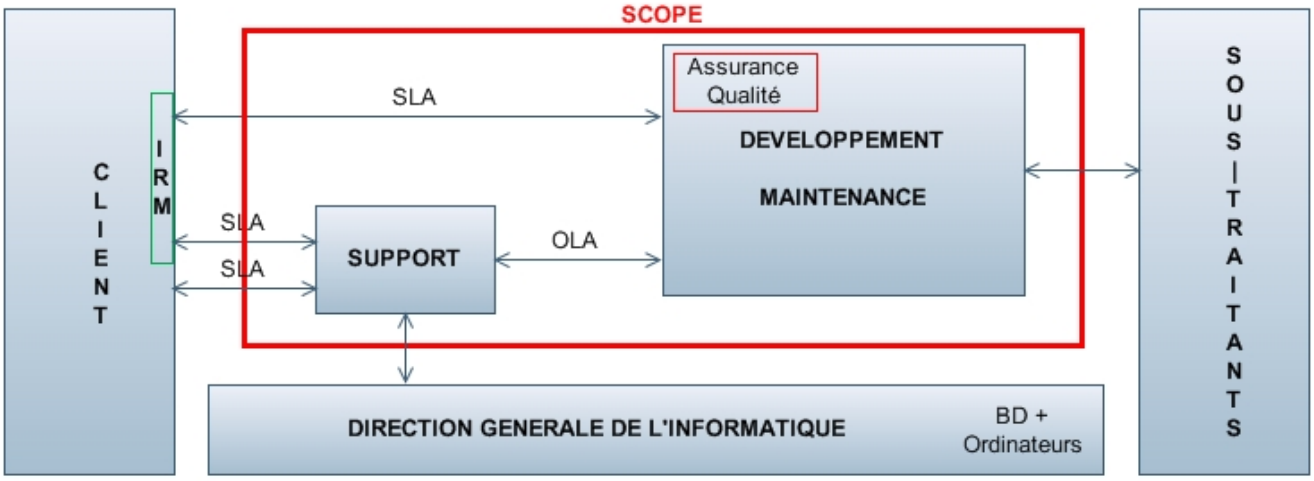


FIG. 7.1 – Diagramme de contexte du projet de Siemens

Cette illustration confronte le diagramme générique $S^{3m®}$ (voir figure 3.2) au contexte propre du projet de maintenance de Siemens. Le diagramme de contexte constitue un préalable à l'évaluation $S^{3m®}$ afin d'obtenir une idée du fonctionnement de l'organisation de maintenance et afin d'évaluer les pratiques du modèle en fonction du contexte spécifique de l'organisation.

L'interface du *Client* se subdivise en deux parties : 700 personnes étant des *clients-utilisateurs* directs des applications maintenues et 15-20 personnes faisant partie d'une équipe appelée *IRM* (pour Informatics Resources Management) qui gère l'unité informatique de l'organisation. Concrètement, les 700 utilisateurs relèguent leurs requêtes de maintenance concernant les applications, et/ou leurs demandes d'évolution, à l'équipe IRM qui les rassemble, les traite et les relaie ensuite, selon les cas, à l'équipe de support ou directement à l'équipe de développement et de maintenance internes à l'équipe de projet chez Siemens.
L'équipe IRM se compose d'interlocuteurs exclusifs avec Siemens utilisant un vocabulaire spécialisé d'informaticiens, vocabulaire que les clients-utilisateurs ne comprennent pas forcément. Ce sont également eux qui dressent le cahier des charges en vue de développer les nouvelles releases, patchs et versions.
Cette équipe est dirigée par un responsable allant à la rencontre des utilisateurs et obtenant des RM. Il lui est ensuite loisible de relayer ces RM directement à l'équipe de développement-maintenance sans passer par le Support.

L'interface *Support* est interne à Siemens bien qu'elle soit quasiment totalement composée d'employés externes. Elle a pour but d'essayer de corriger les requêtes de problème n'impliquant pas de modification du code applicatif. Par exemple, des problèmes liés à des valeurs erronées en base de données, des changements mineurs à apporter à celles-ci, etc.
Un autre objectif de cette interface est de répondre aux questions des clients et de les aider à utiliser les applications.
Dans le cas où l'équipe de support ne peut débloquer une situation ou si cela implique des changements trop importants, les RM sont relayées à l'équipe de développement-maintenance.

L'interface *Développement et Maintenance* est interne à Siemens et est composée à 80% d'employés internes. Elle représente en fait une seule et même équipe, les développeurs des applications jouant également le rôle de mainteneurs. Cela procure l'énorme avantage qu'une phase de transition des logiciels entre ces types d'équipes est inutile. Ici, les mainteneurs sont directement à même de répondre aux RM et possèdent évidemment une très bonne connaissance des applications. Il en résulte un gain de temps par

rapport à d'autres modèles d'organisation utilisant deux équipes distinctes et un risque d'erreur lié à la méconnaissance des applications diminué.
Ce dernier point est par ailleurs accentué par le mode de gestion des équipes de développement chez Siemens voulant que chacun des membres des équipes puisse travailler sur des parties différentes des applications. Cela limite le risque de dépendance de l'entreprise envers certaines personnes clés trop spécialisées.

Les personnes de l'équipe de développement-maintenance sont "interrompues" dans leur tâche de développement de nouvelles versions par des Requêtes de Modification ou incidents menant au développement de releases intermédiaires ou de patchs. Ces requêtes sont traitées selon leur urgence. Habituellement, les RM arrivent de manière standard regroupées dans des packages. Chaque package est alors analysé. Si les demandes clientes sont beaucoup trop importantes pour la prochaine release, elles sont refusées et reportées à la version suivante en accord avec le client. Sinon, elles sont développées et livrées selon les termes du contrat de service SLA.
Il existe également un plan d'assurance qualité visant à garantir un certain niveau de qualité dans les tâches effectuées. Par ailleurs, on s'y réfère aussi lors de la prioritisation des RM. La priorité donnée à ces dernières est finalement discutée et avalisée.
Une dernière précision peut être apportée quant aux types de maintenance réalisés dans ce projet : trois types sont principalement rencontrés à savoir la perfective (appelée "évolutive"), la corrective et l'adaptative. Quant à la maintenance préventive, elle est présente mais d'une manière mineure car l'équipe de projet chez Siemens n'a pas les pleins pouvoir de part les limites du projet définies par le contrat de service.

La définition des deux interfaces précédentes, Support et Développement-Maintenance, permet déjà de définir les limites du projet (ou "scope") de maintenance de Siemens. Ces limites sont importantes à prendre en compte dans le cadre d'une évaluation de maturité comme expliqué dans le chapitre 9.

Par ailleurs, il existe donc deux interactions possibles entre l'interface *Client* et l'interface *Développement et Maintenance* : soit directement d'une interface à l'autre, soit par le biais de l'interface *Support*. Il est à noter que les trois interactions possibles dans cette configuration sont définies selon des SLA's (Service Level Agreements, ententes de services), c'est à dire des contrats de services à respecter.

Il vient ensuite l'interface *Direction Générale de l'Informatique*. Celle-ci fait partie de la même organisation générale que le client et comprend les bases de données et ordinateurs du client. Il s'agit d'une équipe dirigeant le SI client et pouvant demander à Siemens de passer à une version supérieure d'une application. Cette interface communique exclusivement avec l'interface Support interne à Siemens.
Un exemple d'interaction peut être donné avec le processus de maintenance sur le Rajeunissement et la Retraite du logiciel (faisant partie des processus de support opérationnel de la maintenance, voir fig. 3.4). Ainsi, Siemens prend rarement la décision de stopper ou de rajeunir complètement une application. Cette décision vient de la Direction Générale de l'Informatique tandis que Siemens donne des informations et des recommandations. Toutefois, elle participe aussi au travail lorsqu'une décision est tombée.

Finalement, l'interface de Développement-Maintenance communique exclusivement avec l'interface des *Sous-Traitants*. Celle-ci se compose en fait principalement de fournisseurs d'outils comme Oracle, IBM Rational, Hibernate, etc.
Une sous-entité de Siemens (SDC) opérant en Pologne peut également être considérée comme un sous-traitant dans ce diagramme de contexte.

7.4 Classification des processus de la maintenance du logiciel

Afin de bien cerner le contexte du projet de maintenance de Siemens, il leur a aussi été demandé de confronter la classification générique des processus de la maintenance du logiciel issue du modèle $S^{3m®}$ (voir fig. 3.4) à leur propre classification. Il en est ressorti que leur classification est semblable à celle proposée par le modèle à ceci près que :

- Processus opérationnels : le processus de *Transition* n'existe pas chez eux étant donné que les développeurs sont également les mainteneurs.
  Au niveau des connaissances acquises par les membres des équipes, elles sont maintenues aussi équivalentes que possible grâce au roulement des affectations des membres aux différentes parties des applications.

- Processus de support opérationnel : le processus du *Rajeunissement et de la Retraite du logiciel*, bien qu'existant, n'est que très peu utilisé dans le contexte de ce projet de part l'existence de la Direction Générale de l'Informatique. Siemens officie dans ce projet en tant que sous-traitant de son client et ne peut donc pas prendre à lui seul ce genre de décision.

- Processus organisationnels : le processus des *Audits internes et d'assurance qualité* n'existe pas comme tel puisqu'il n'y a pas d'audit interne.

Au final, les processus de Transition, d'audit interne et de Rajeunissement et Retraite du logiciel n'étant pas, ou très peu utilisés, les pratiques $S^{3m®}$ y afférant pourront être évaluées à "Non Applicable" ou considérées dans une moindre mesure.

7.5 Résultats de l'évaluation

Après avoir défini le contexte du projet, ses limites contractuelles, les buts de l'évaluation et la méthode d'évaluation, le processus d'évaluation pu démarrer. Il faut préciser que l'évaluation s'est opérée sur toutes les pratiques $S^{3m®}$ hormis celles évincées pour cause d'absence de leur processus, c'est à dire celles liées aux processus mentionnés à la section 7.4. L'entreprise a ainsi choisi une évaluation traversant toute son organisation de maintenance sans se focaliser sur un ensemble particulier de processus.
Toutefois, ce choix vient surtout du fait que l'un des buts de l'évaluation était de comprendre le modèle et ses pratiques.

Cette section va donc analyser les résultats de l'évaluation de maturité effectuée sur le projet défini ci-dessus. Ceci permettra d'observer à partir d'un exemple réel comment il est possible d'extraire des pratiques faibles en se servant des résultats fournis par l'outil. Pour cela, l'ordre des onglets, tels qu'ils apparaissent dans la partie Analyse de l'application, sera suivi. Cette procédure illustrera donc par la même occasion la stratégie d'amélioration suggérée dans la partie précédente de ce mémoire.
Il est à préciser qu'afin d'être concis et pertinent, la démarche suivra l'exemple de la détection d'un ensemble de pratiques faibles, la détection des autres pratiques se faisant de manière semblable.

7.5.1 Histogramme des domaines $S^{3m®}$

Le processus d'analyse débute donc avec la comparaison des domaines de processus $S^{3m®}$. La figure 7.2 illustre le diagramme des résultats.

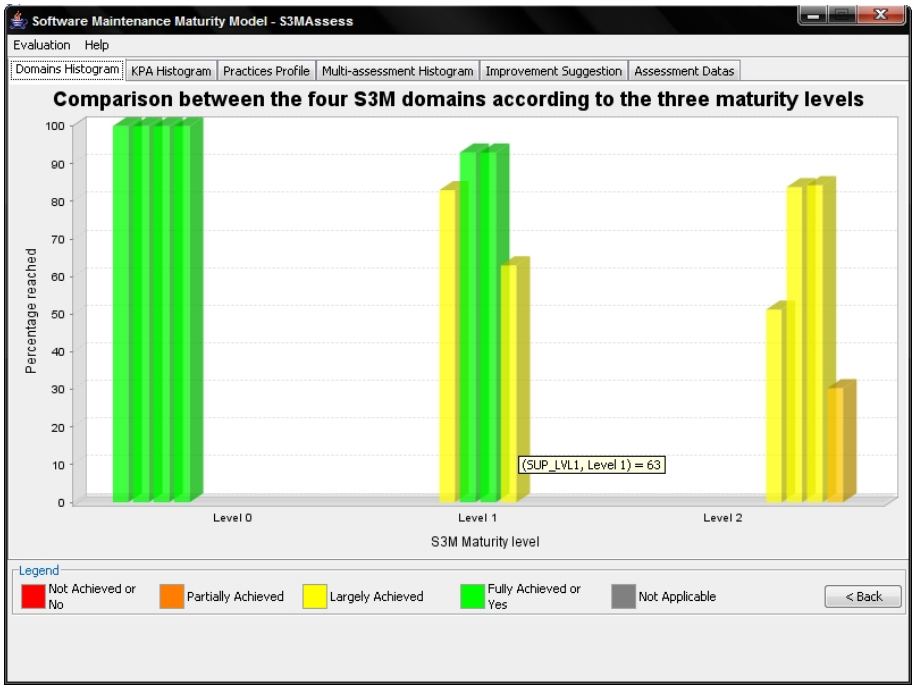


FIG. 7.2 – Histogramme des domaines $S^{3m®}$

Les quatre domaines de processus atteignent 100% pour le niveau de maturité zéro. Cela signifie que toutes les pratiques de ce niveau de maturité proposées pour l'ensemble des domaines existent bien dans l'équipe de projet chez Siemens ou ne sont pas applicables. Il n'y a donc pas de faiblesse à ce niveau.

Le niveau supérieur donne les moyennes suivantes :

1. Gestion du processus de la maintenance du logiciel : 83%
2. Gestion des requêtes de la maintenance du logiciel : 93%
3. Ingénierie de l'évolution du logiciel : 93%
4. Support à l'ingénierie de l'évolution du logiciel : 63%

Ceci veut dire que le premier domaine atteint largement les objectifs du niveau de maturité $S^{3m®}$ un, les deux domaines suivants atteignent complètement les objectifs de toutes les pratiques du niveau un (la valeur de 93% étant la valeur maximum octroyable aux niveaux de maturité supérieurs à zéro) tandis que le quatrième, celui du Support, n'atteint que 63% de moyenne pour ses pratiques de niveau un.
La stratégie d'amélioration suggèrerait donc de s'attarder en premier lieu sur le quatrième domaine au niveau un pour améliorer certaines de ses pratiques et ainsi obtenir une moyenne plus acceptable. En vue de préserver un certain équilibre dans le processus général de maintenance, il ne serait donc pas recommandé d'améliorer d'autres pratiques sans se soucier de celles-ci.

Le niveau de maturité deux exprime une certaine logique dans les résultats d'évaluation. En effet, les quatre domaines à ce niveau suivent le même profil de résultats qu'au niveau un : les deux domaines dominant sont les deuxième et troisième, suivis par le premier domaine et finalement, le quatrième domaine clôt la marche. C'est exactement la même configuration qu'au niveau un mais avec des objectifs de pratiques à chaque fois moins atteints. Ce fait peut donc témoigner d'une certaine rigueur et cohérence dans la cotation des pratiques.
Au niveau des résultats purement chiffrés, les domaines deux et trois atteignent toujours un niveau plus que raisonnable avec plus de 83% de moyenne. Par contre, le domaine du management des processus qui

était juste au delà des 80% à son niveau un, n'atteint plus que 51% au niveau deux.
Finalement et sans surprise, le domaine quatre déjà trop faible au niveau un, atteint très faiblement les objectifs du niveau de maturité deux.

En résumé, la suggestion de priorité des améliorations est la suivante :

1. Domaine 4, niveau 1 ;
2. Domaine 4, niveau 2 ;
3. Domaine 1, niveau 2.

Il n'y a pas de recommandation particulière à faire sur les autres configurations des domaines et niveaux car elles obtiennent de très bons pourcentages.
Attention toutefois à la façon d'améliorer ces pratiques. Etant donné les dépendances pouvant exister entre pratiques, il est recommandé d'effectuer une évaluation globale après chaque phase d'amélioration afin de s'assurer que des pratiques ne se sont pas dégradées risquant ainsi de devenir plus prioritaires qu'auparavant.

7.5.2 Histogramme des itinéraires

L'histogramme précédent suggérait prioritairement l'amélioration du domaine 4 au niveau 1. Cet exemple va être suivi ici. L'histogramme comparant les itinéraires permet d'afficher les moyennes des itinéraires d'un domaine à un niveau spécifié. De la même manière que pour les domaines, les itinéraires les plus faibles pourront être révélés.

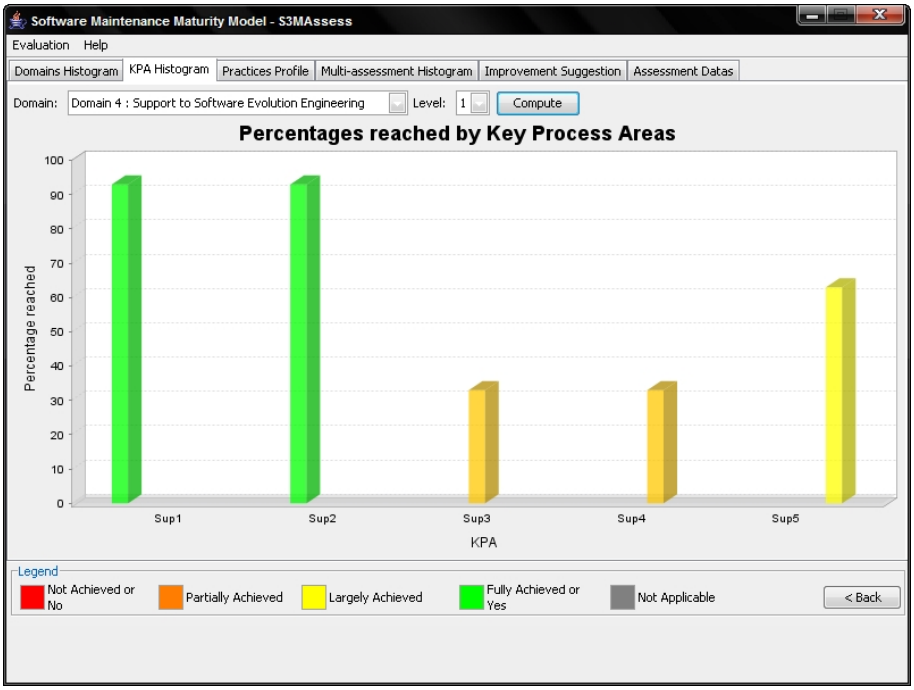


FIG. 7.3 – Histogramme des itinéraires S^{3m}®

Cinq itinéraires sont présents pour le domaine quatre, *Support à l'ingénierie de l'évolution du logiciel*, et obtiennent pour le niveau de maturité un :

Sup1 Gestion de la configuration et du changement : 93%
Sup2 Assurance qualité du processus, du service et du logiciel : 93%

Sup3 Mesure et analyse de la maintenance : 33%
Sup4 Analyse causale et résolution de problèmes : 33%
Sup5 Rajeunissement, migration et retraite du logiciel : 63%

Les deux premiers itinéraires atteignent complètement leurs objectifs avec la valeur maximale de 93%. Par contre, les itinéraires 3 et 4 n'atteignent que très faiblement leurs objectifs avec une moyenne de 33%. Finalement, le cinquième itinéraire atteint *assez* largement ses objectifs avec une moyenne de 63%.
Cela suggère donc que ce sont en premier lieu les itinéraires 3 et 4 qui abaissent la moyenne du quatrième domaine de processus et en second lieu, l'itinéraire 5. L'attention devrait donc se porter respectivement en priorité sur ces itinéraires.
Mais quelles sont exactement les pratiques faibles à améliorer et pourquoi sont-elles faibles ? La section suivante va répondre à cette question.

7.5.3 profil de secteur

Maintenant que les itinéraires les plus prioritaires ont été décelés, il s'agit de connaître les pratiques qu'ils comportent et surtout, de découvrir qu'elles sont les plus faibles et pourquoi le sont-elles. Toutefois, afin d'analyser ces résultats de manière plus pointue, il est utile de rappeler les objectifs poursuivis par le domaine 4 et par ses itinéraires 3, 4 et 5.

Le domaine traitant du Support à l'ingénierie d'évolution vise, comme son nom l'indique, des processus de support à l'ingénierie d'évolution de la maintenance du logiciel.
Les processus de support servent les processus opérationnels. Certains de ces processus sont gérés conjointement entre plusieurs unités organisationnelles, comme les développeurs, l'infrastructure et les opérations, car ils servent souvent tous les intervenants des Technologies de l'Information.

Quant aux itinéraires 3, 4 et 5, voici leurs objectifs :

Sup3, Mesure et analyse de la maintenance : Le but de la mesure et de l'analyse de la maintenance est de développer et de soutenir une capacité de mesure qui est utilisée pour supporter les besoins d'information de la direction de la maintenance du logiciel.

Sup4, Analyse causale et résolution de problèmes : L'analyse causale identifie les causes des défauts et autres problèmes et identifie les actions à prendre pour empêcher leur résurgence dans le futur. L'interdépendance entre les processus d'affaires et ses logiciels fait que, si les logiciels ont une défaillance, l'organisation entière est paralysée. La continuité du service et la disponibilité des logiciels nécessitent donc un processus de résolution de problèmes.

Sup5, Rajeunissement, migration et retraite du logiciel : L'itinéraire du rajeunissement du logiciel s'intéresse à la résolution de problèmes reliés au déclin d'un logiciel ou à un changement majeur dans son environnement (c'est à dire règles d'affaires, architecture, langage de programmation, etc.). On tentera donc d'augmenter soit sa qualité, soit son efficacité, soit sa disponibilité générale. Dans certains cas, on tente de réduire les coûts et le temps d'attente de sa maintenance.
Cet itinéraire comprend aussi la migration d'un logiciel dans un autre environnement technique. Ces activités sont considérées comme des activités de maintenance préventive.
On doit explorer les composants du logiciel pour essayer de le documenter, d'en dériver de l'information supplémentaire ou d'en changer la structure pour le rendre plus compréhensible et maintenable.

Pour plus de détails, voir [April et Abran, 2006].
A noter également qu'assez exceptionnellement, seules une ou deux pratiques $S^{3m®}$ ont été définies au

niveau de maturité un de ces itinéraires par l'auteur du modèle. Voici le profil de secteur du domaine quatre :

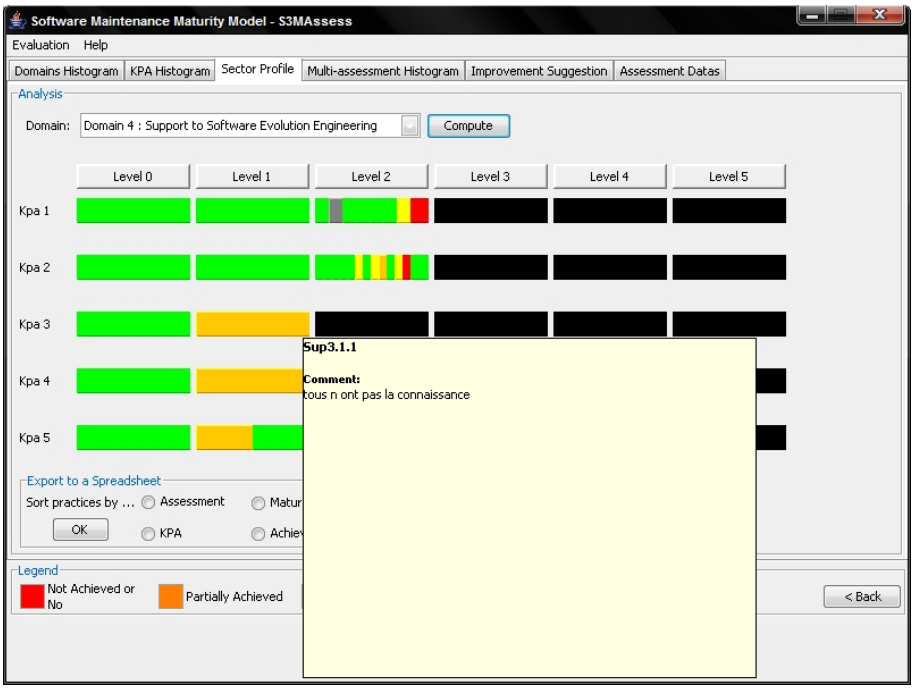


FIG. 7.4 – profil de secteur $S^{3m®}$ du domaine quatre

Cette fonctionnalité fournit plusieurs informations. En passant le pointeur de la souris sur les zones de couleurs, les bulles d'informations indiquent que les itinéraires (Kpa en anglais) 3 et 4 ne comportent chacun qu'une seule pratique pour le niveau de maturité un. Ce sont les pratiques **Sup3.1.1** et **Sup4.1.1**. Elles ont été évaluées à "Partiellement Atteint" et elles ont été commentées lors de l'évaluation :

Sup3.1.1 : "La mesure et l'analyse, dans l'organisation de maintenance, sont réalisées individuellement par chaque manager."
Commentaire : *tous n'ont pas la connaissance.*
La justification de cette cotation (33%) était que tous les managers n'ont pas les compétences nécessaires à la réalisation de mesures et d'analyse du processus de maintenance.

Sup4.1.1 : "L'organisation de maintenance reconnait la nécessité d'avoir un processus de résolution de problème et d'analyse de cause."
Commentaire : *Lessons Learned et Defect Prevention Process réalisés dans certains cas bien précis.*
Siemens définit deux approches allant dans le sens de la pratique : les Lessons Learned et les DPP.
Les Lessons Learned consistent en la réalisation de brainstormings dans une salle où les participants analysent tout ce qui n'a pas fonctionné.
Le DPP est l'analyse de tous les problèmes détectés dans une application. Ces problèmes sont principalement des défaillances et des erreurs de développement. Il s'agit de comprendre pourquoi ces problèmes n'ont pas été détectés dans les tests.
Cette pratique n'a donc obtenu que 33% car si l'organisation reconnait bien son utilité, de part la définition des deux processus expliqués ci-dessus, ceux-ci ne sont appliqués qu'assez rarement (exigence apparaissant dans le détail de la pratique).

Sup5.1.1 : "Dans l'organisation de maintenance du logiciel, le rajeunissement est réalisé de manière informelle, durant un changement spécifique, sans aucune planification préalable."

Commentaire : *Réalisé si cela n'a pas d'impact sur les coûts et sur le planning sinon il y a négociation avec le client (scope).*
Avant d'expliquer le résultat de cette pratique, il est à rappeler qu'elle est issue du cinquième itinéraire et qu'elle apparait donc seconde dans la priorité des améliorations (après Sup3.1.1 et Sup4.1.1). En effet, cet itinéraire obtenait 63% au niveau de maturité un contre 33% pour les itinéraires trois et quatre.
La pratique Sup5.1.1 recevait donc la cotation de 33% faisant ainsi chuter la moyenne de son itinéraire, l'autre pratique présente à ce niveau obtenant 93%. Elle obtient cette faible cotation car les mainteneurs ne décident pas réellement eux-mêmes si un logiciel doit être rajeuni ou non. Par exemple, il peut arriver qu'ils améliorent le niveau documentation du logiciel mais uniquement s'ils ont encore le temps de le faire et si cela n'impacte pas les coûts de maintenance. En fait, comme bon nombre d'organisations, Siemens définit avec son client l'étendue du projet pour lequel elle sera rémunérée. S'il n'est pas bénéfique pour Siemens de rajeunir, de quelque façon que ce soit, ou si cela n'est pas demandé par le client via un cahier des charges, il est très improbable que cette décision soit prise par le mainteneur.
Au final, la pratique a été évaluée à 33% car elle n'est pas réellement atteinte. Toutefois, si après l'étape de brainstorming à l'issue du processus d'analyse des résultats d'évaluation, la décision est prise qu'il n'est pas rentable d'améliorer cette pratique, il serait envisageable de l'évaluer à "Non Applicable" lors de la prochaine évaluation.

En conclusion, il serait recommandé de se pencher en priorité sur les pratiques Sup3.1.1 et Sup4.1.1 car elles sont principalement la cause de la faiblesse du domaine quatre au niveau un. Toutefois, cela ne signifie pas forcément que ce sont celles-là qui devront obligatoirement être améliorées. Leur amélioration doit être discutée lors du brainstorming pour déterminer l'investissement qu'elles représentent et si ce dernier est rentable et le plus judicieux. Cela dépend des objectifs de Siemens ainsi que de leur budget.
Cependant, ces deux pratiques ne semblent pas être les plus critiques par rapport à ces critères. A part une formation à payer aux managers afin de les former aux mesures et à l'analyse des processus de maintenance et, utiliser plus souvent les lessons learned et DPP, peu d'investissement économique est nécessaire. Cela consommera malgré tout du temps à réaliser.
Par contre, la pratique devant être la plus débattue est surement la Sup5.1.1 puisqu'il y a vraiment un choix à faire entre maintenir un logiciel plus maintenable mais entraînant un coût plus important de maintenance ou continuer dans l'état actuel des choses mais en risquant d'être de plus en plus lent dans la réponse aux requêtes de modifications pour cause de dégradation progressive de la maintenabilité du logiciel. Cette pratique est un exemple typique d'une pratique théorique n'étant pas forcément applicable dans un cas d'entreprise de sous-traitance où l'optimisation du ROI et le temps de maintenance sont très importants. Pratiquer de la réingénierie consomme beaucoup de ressources. Or, c'est difficilement envisageable d'autant plus que ce serait réalisé gratuitement et sans que le client n'en soit particulièrement conscient.
Cependant, il faut considérer le fait que cette pratique peut améliorer le processus de maintenance en augmentant la maintenabilité du logiciel. Par conséquent, un gain de temps pour les opérations de maintenance pour ce logiciel amélioré pourrait être observé dans le futur.
C'est pourquoi, un brainstorming sera certainement nécessaire afin d'évaluer si l'investissement porté sur l'amélioration de cette pratique pourra être rentable à terme.

A noter que les résultats complets de cette évaluation ont été exportés dans une feuille de calculs qui est disponible en annexe, voir la partie 10.

Finalement, étant donné qu'il s'agit de la première évaluation effectuée pour ce projet, il n'y a pas lieu d'utiliser la fonction de multi-évaluation. La section suivante va donc directement présenter les résultats de la fonction de suggestion automatique des résultats.

7.5.4 Suggestions d'amélioration

Cette fonctionnalité est disponible sous l'onglet *Improvement Suggestion* et permet d'obtenir automatiquement les sous-ensembles de pratiques faibles. Afin de vérifier si les résultats obtenus ci-dessus se confirment, voici les résultats retournés pour le domaine quatre.

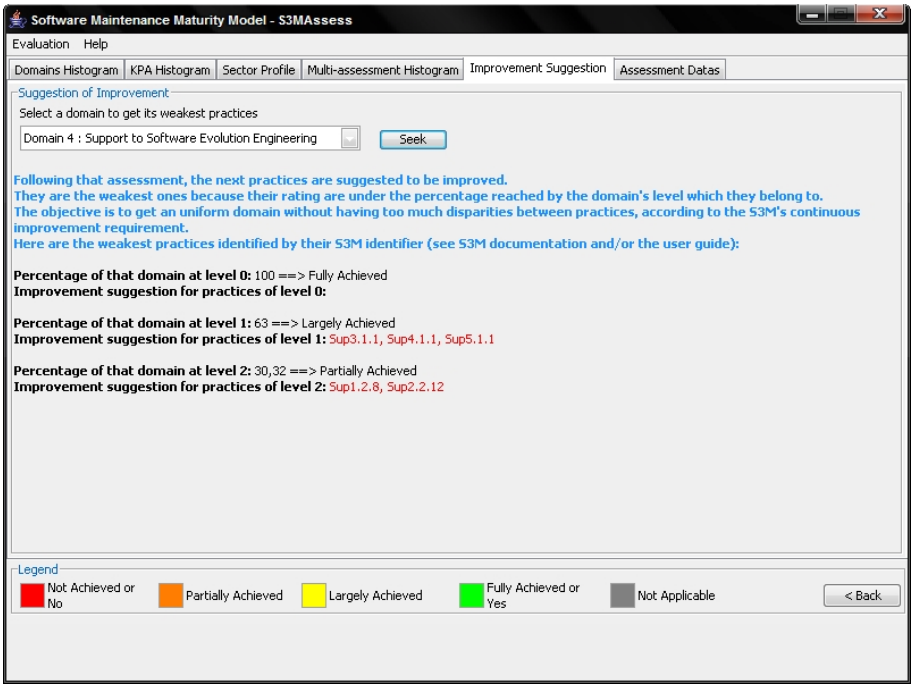


FIG. 7.5 – Suggestions d'amélioration

Les résultats ne donnent aucune indication quant aux pratiques du niveau zéro puisque celui-ci a été totalement atteint avec une moyenne de 100%.

Au niveau un, le sous-ensemble suggéré reprend les pratiques Sup3.1.1, Sup4.1.1 et Sup5.1.1 puisqu'elles ont été évaluées en-dessous de la moyenne de 63% obtenue par le domaine quatre au niveau un. Les résultats retournés par l'application confirment donc bien l'exemple retracé au fil des sections précédentes des pratiques à améliorer en priorité. Cela illustre la correction des résultats et la fonctionnalité de cet onglet.
L'utilisateur peut donc se limiter à établir la priorité des itinéraires à certains niveaux et obtenir directement leurs pratiques faibles parmi toutes les pratiques faibles proposées via cette fonctionnalité $S^{3mAssess®}$. Sinon, il peut se contenter d'accéder directement aux sous-ensembles suggérés par cette fonctionnalité et débattre de leur priorité d'amélioration lors du brainstorming.
Attention toutefois au fait que si l'utilisateur avait choisi la méthode $S^{3m}Assessment$ et voulait améliorer un niveau supérieur à un niveau n'ayant pas atteint le seuil de 80%, l'application ne donnerait pas d'indication concernant les pratiques de ce niveau supérieur car elles n'auraient pas été évaluées pour cause de blocage par la méthode d'évaluation.

Finalement, les pratiques suggérées pour le niveau deux du domaine quatre sont Sup1.2.8 et Sup2.2.12 car elles ont été évaluées en-dessous du pourcentage atteint par le domaine.
Profitons de cet exemple pour illustrer une particularité quant au calcul de la moyenne du domaine (voir les formules mathématiques à la section 6.2.3.5) puisque les 30% obtenus par le domaine ne représentent pas uniquement la moyenne des itinéraires ayant pu être évalués mais également le fait que des itinéraires n'ont pas pu être évalués au niveau deux pour cause de blocage par la méthode $S^{3m}Assessment$. N'ayant

pas de résultat d'évaluation au niveau deux, les itinéraires 3, 4 et 5 obtiennent 0% dans le calcul de la moyenne pour le niveau deux.
Le but est bien de représenter le fait qu'il ne serait pas logique qu'un domaine atteigne une moyenne faible pour ses itinéraires d'un niveau n, ces itinéraires étant tellement faibles que certains seraient bloqués au niveau $n+1$ par la méthode $S^{3m}Assessment$, et que le domaine obtienne malgré tout une moyenne élevée au niveau $n+1$ grâce à l'évaluation des itinéraires n'ayant pas été bloqués. La non évaluation des trois itinéraires bloqués constitue un manque pour le niveau deux et ce manque apparaît dans la faiblesse de la moyenne obtenue.

7.6 Conclusion

Cette étude de cas visait à confronter les attentes d'un chef de projet avec les possibilités offertes par le modèle $S^{3m®}$ et par son outil de support $S^{3mAssess®}$. Il s'est avéré qu'il n'était pas tout à fait possible de répondre directement aux attentes initiales qui étaient de pouvoir évaluer rapidement les risques des projets proposés. Toutefois, l'utilisation du modèle et de l'outil permettrait de mieux en comprendre les notions et d'éventuellement mieux définir et structurer les points importants à vérifier avant d'accepter un projet de maintenance.

A cette fin, une évaluation concrète d'un projet réel de l'organisation de maintenance fut réalisée. Ensuite, les résultats de cette évaluation furent présentés ainsi qu'une démarche permettant de trouver les faiblesses dans les processus employés dans le projet de Siemens. Finalement, un exemple de révélation de pratiques faibles a été suivi et justifié. Il est ainsi suggéré d'améliorer dans l'ordre les domaines et niveaux suivants : Domaine 4 au niveau 1, Domaine 4 au niveau 2 et Domaine 1 au niveau 2. Ceci donne un premier sous-ensemble de pratiques faibles à débattre lors du brainstorming : Sup3.1.1 ; Sup4.1.1 et Sup5.1.1.

Par ailleurs, l'étude de cas illustre bien la nécessité de l'étape de brainstorming. Il est difficile d'établir automatiquement un ordre d'amélioration des pratiques à respecter scrupuleusement sans même en débattre.
A la suite d'une première discussion concernant ces résultats, les participants au brainstorming pourraient se pencher sur la priorité suivante d'amélioration, c'est à dire le domaine quatre au niveau deux. Le procédé d'analyse de ce domaine est semblable à celui venant d'être exposé.
Ce domaine 4 au niveau 2 devrait toutefois être reconsidéré suite aux décisions prises quant aux pratiques faibles de son niveau 1. Le terme "reconsidéré" signifie "réévalué" (en utilisant éventuellement la fonction $S^{3mAssess®}$ de modification d'une évaluation) en tenant compte de l'approche exposée lors du brainstorming. Dans ce cas, il faudra être vigilant au fait que la méthode $S^{3m}Assessment$ a été choisie lors du processus d'évaluation impliquant la non-évaluation du niveau deux des trois itinéraires Sup3 à Sup5 puisqu'ils n'atteignaient pas le seuil des 80% au niveau un. Par conséquent, il serait risqué de prendre des décisions d'amélioration du niveau deux sans en connaître la maturité de toutes ses pratiques.

7.6.1 Suggestions d'amélioration du modèle

La réalisation de cette étude de cas a aussi permis de révéler certaines suggestions d'amélioration du modèle $S^{3m®}$:

- Dans ce cas-ci, le contrat du projet définit la manière de travailler. Il n'est pas toujours rentable d'améliorer toutes les pratiques prévues par le modèle. L'équipe de maintenance peut se trouver coincée par le contrat. Par exemple, l'obligation contractuelle de travailler avec un logiciel même si celui-ci n'est pas le meilleur sur le marché.
  Il est important de tenir compte de ces contraintes lors de l'évaluation des pratiques. Pour cela, il faudrait détailler comment évaluer certaines pratiques $S^{3m®}$ soumises à des contraintes contractuelles.

- Déterminer comment évaluer une pratique ayant un de ses éléments ne se produisant qu'une fois sur deux.
- Déterminer comment évaluer une pratique d'un niveau de cotation NPLF[1] si sa forme tend à répondre "oui" ou "non". Eventuellement reformuler ce genre de pratiques.
- Pouvoir utiliser le modèle afin de choisir les bonnes opportunités clientes. Cela pourrait passer par une sorte de "check-list" permettant de souligner les risques d'un projet. L'idée serait d'approcher au maximum la réalité pour en extraire les risques.

7.6.2 Suggestions d'amélioration de l'outil

Concernant l'outil, quelques suggestions surgirent également :

- Afficher les facettes $S^{3m®}$ pendant le processus d'évaluation pour interpréter plus facilement les pratiques.
- Insérer un bouton "Back" pour revenir sur une pratique déjà évaluée au cours de l'itinéraire courant.
- Afficher l'énoncé complet d'une pratique dans les bulles d'informations présentes lors de l'analyse des profils de secteur.

Ces suggestions d'amélioration du modèle et de l'outil seront reprises plus en détails au chapitre 9.

[1] C'est à dire à partir du niveau de maturité un

Chapitre 8

Etude de cas 2 : Application du modèle et de l'outil pour une auto-évaluation du projet

La structure de cette deuxième étude de cas suivra celle déjà proposée dans l'étude précédente. Toutefois, les objectifs entre les deux études sont différents. La première étude visait à expérimenter le modèle et l'outil dans le contexte d'une entreprise afin de déterminer ses pratiques faibles mais également afin de tirer des conclusions d'amélioration du modèle et de l'outil.
Cette seconde étude vise quant à elle à tirer des conclusions sur les processus de maintenance employés dans le cadre de ce projet. A l'issue de l'évaluation, des conclusions d'amélioration de certaines pratiques clés seront proposées mais également une discussion quant à l'adaptabilité de certaines pratiques $S^{3m®}$.

8.1 Cadre de l'étude de cas

Etant donné que les ressources de l'organisation de maintenance réalisant ce projet sont relativement limitées, une évaluation du type SCAMPI n'est pas du tout adaptée. Cette chapitre va donc profiter de cette configuration organisationnelle pour appliquer le modèle $S^{3m®}$. En effet, ce modèle est justement adapté à ce contexte spécifique. De même, l'outil de support permettra de réaliser cette auto-évaluation afin de déterminer la capacité du processus de maintenance et de voir comment il pourrait être amélioré. Les résultats finaux de cette auto-évaluation pourront aussi servir d'expérience pour les futurs mainteneurs de $S^{3mAssess®}$.

Par ailleurs, il n'est pas toujours évident et très critique de pratiquer une auto-évaluation de part le risque de manque d'objectivité. Cependant, il faut garder à l'esprit que le but final est d'améliorer le processus actuel de maintenance, la façon de travailler, d'éventuellement remettre en question la méthodologie établie, d'éviter de répéter les mêmes erreurs et d'aborder la prochaine phase de maintenance avec plus d'atouts. C'est pourquoi, cette auto-évaluation se voudra sincère et honnête.

Dernière précision, l'évaluation sera limitée au contexte de ce mémoire, c'est à dire qu'elle ne remontera pas jusqu'au début de l'existence de l'outil étant donné le peu d'informations disponibles. Les processus de maintenance évalués s'étendront donc de l'acquisition de l'outil au début de ce projet jusqu'à la création de sa troisième révision.

8.2 Mise en place de l'étude de cas

Le projet de maintenance de l'outil sera évalué par son auteur. Le modèle et l'outil lui-même seront utilisés afin de déterminer la maturité des processus employés lors de sa maintenance. L'ensemble des éléments exposés dans la partie II du mémoire ainsi que les détails présents en annexe serviront comme entrées au processus d'évaluation. Quant à la méthodologie de cotation, elle restera identique à celle de l'étude de cas précédente.

8.3 Analyse du contexte

Etant donné que le contexte de ce projet de maintenance a déjà été présenté en détails dans la partie précédente du mémoire, cette section se limitera à commenter le diagramme de contexte afin d'éviter des répétitions inutiles. Ce nouveau diagramme apporte en effet une vue différente de la réalisation de ce projet.

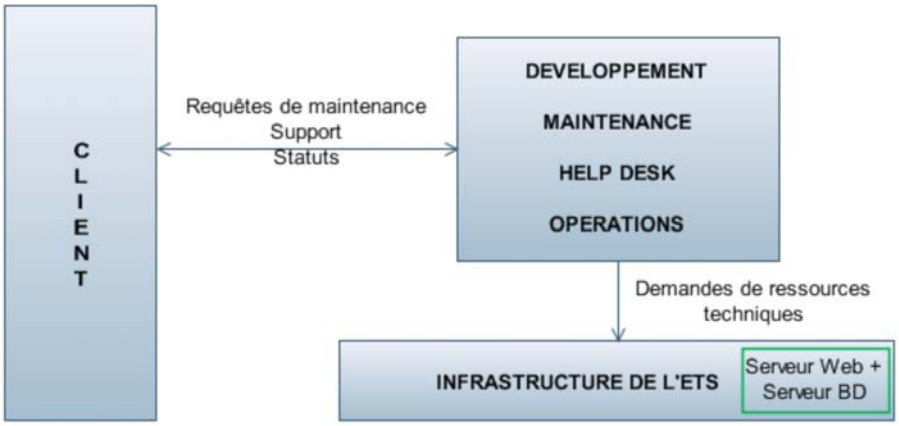


FIG. 8.1 – Diagramme de contexte du projet de maintenance de $S^{3mAssess®}$

Le diagramme de la figure 8.1 reflète directement la limitation de l'équipe de maintenance réduite à une seule personne de part le nombre réduit d'interfaces. En effet, les unités organisationnelles du développement, de la maintenance, du help desk et des opérations sont regroupées dans une seule et même interface.

Tout d'abord, l'interface du *Client* représente le maître de stage et les divers intervenants présentés à la partie 9 des annexes. Le maître de stage a bien entendu un rôle particulier puisqu'il ne se limite pas vraiment à un "simple" client. Il a aussi suggéré des idées dans la conception des solutions. Les autres intervenants sont de plusieurs types : des membres d'entreprises, professeurs, etc.
Cette interface a une seule relation dans ce projet, celle avec l'interface Développement/Maintenance/... . La nature de cette relation est un ensemble de requêtes de maintenance de tous types, de support et de notification des status d'avancement :

- Les quatre catégories de maintenance ont été rencontrées ;
- Un support, tant technique que d'encadrement pour la compréhension du $S^{3m®}$, était proposé aux clients ;
- Les changements étaient actés et notifiés aux clients.

L'interface de *Développement/Maintenance/Help Desk/Opérations* représente l'ensemble de ces quatre unités organisationnelles réalisé par une seule et même personne. Cela comprend des activités de développement d'évolutions, de maintenance répondant directement aux RMs, de Help Desk répondant aux

diverses demandes clientes et, de suivi de l'opération de l'outil. Cette dernière fonction comprend la mise en opération de l'outil, les interactions avec l'interface d'Infrastructure de l'ETS afin d'obtenir les ressources informatiques nécessaires et le suivi de l'opération de $S^{3mAssess®}$ comme les mises à jour des diverses révisions, le contrôle de la disponibilité de l'outil, etc.

Enfin, l'interface de l'*Infrastructure de l'ETS* comprend le personnel technique et le management de l'école allouant les autorisations d'accès aux ressources informatiques. Les ressources demandées dans le cadre de ce projet ont été un serveur de base de données ainsi qu'un serveur web.

8.4 Classification des processus de la maintenance du logiciel

Une fois le contexte du projet de maintenance établi, il s'agit de délimiter les processus de maintenance utilisés (à partir de la figure théorique 3.4). Cette limitation permettra de mieux cibler les pratiques pertinentes lors de l'évaluation $S^{3m®}$.
Voici donc par classes de processus, les processus non employés :

Processus opérationnels : Il n'y eu pas de *Transition* vu l'indisponibilité des développeurs initiaux lors de l'acquisition de $S^{3mAssess®}$ et vu que le développement et la maintenance ont ensuite appartenu à la même interface dans ce projet évalué.
A noter également que le processus de *Service de support opérationnel* inclus les aides fournies aux entreprises ayant participé à ce projet. Cela inclus la fourniture des documents de compréhension du modèle $S^{3m®}$ et les éclaircissements donnés via courriels, conférences téléphoniques, réunions, etc.

Processus de support opérationnel : Tous les processus ont été présents en considérant que la *Formation à la maintenance* est passée par la mise en place de la méthodologie de maintenance[1] et que la *Gestion de l'entente de service* passe par l'établissement des accords avec l'interface Client sur les produits à fournir (leur délimitation, le délai, etc.) via principalement les analyses d'impacts[2] (voir en annexe).

Processus organisationnels : Il n'y pas eu de *mesure organisationnelle*, la durée de développement des révisions faisant partie de la planification de la maintenance et les effectifs restant constants.
Ensuite, les processus d'*Audit interne* et d'*Assurance qualité* étaient également absents. Seul un certain niveau de documentation était maintenu mais il n'y avait pas vraiment de procédure définie.
Enfin, le processus d'*Achat et de gestion des ressources humaines* internes à l'équipe de maintenance n'était pas employé non plus.

8.5 Les résultats de l'évaluation

Pour évaluer les pratiques et analyser les résultats de leur évaluation de manière pertinente, il faut se rappeler que :
- l'"équipe" de maintenance se limitait à une seule personne ;
- l'acquisition de l'outil ne fut pas sans découverte de manques et de faiblesses importants,
- le contexte d'évolution de l'application exigeait une certaine rapidité de résultats comme cela a été présenté à la partie précédente lors de la définition du cycle de vie du processus de maintenance.

[1] la personne ayant mis en place cette méthodologie étant celle qui l'applique, il n'y a donc pas nécessité de former un quelconque membre à cette méthodologie.

[2] Les Analyses d'Impact sont un plan au niveau des requêtes. C'est très utile car cela permet de planifier le travail, d'établir un SLA avec les clients, de planifier les risques, de planifier la durée d'élaboration de chaque RM, d'évaluer à l'avance les impacts sur la performance, de décrire les interrelations des modifications/évolutions du logiciel, etc.

8.5.1 Histogramme des domaines $S^{3m®}$

Cette sous-section détermine les domaines faibles et établit une suggestion de priorité de leur amélioration.

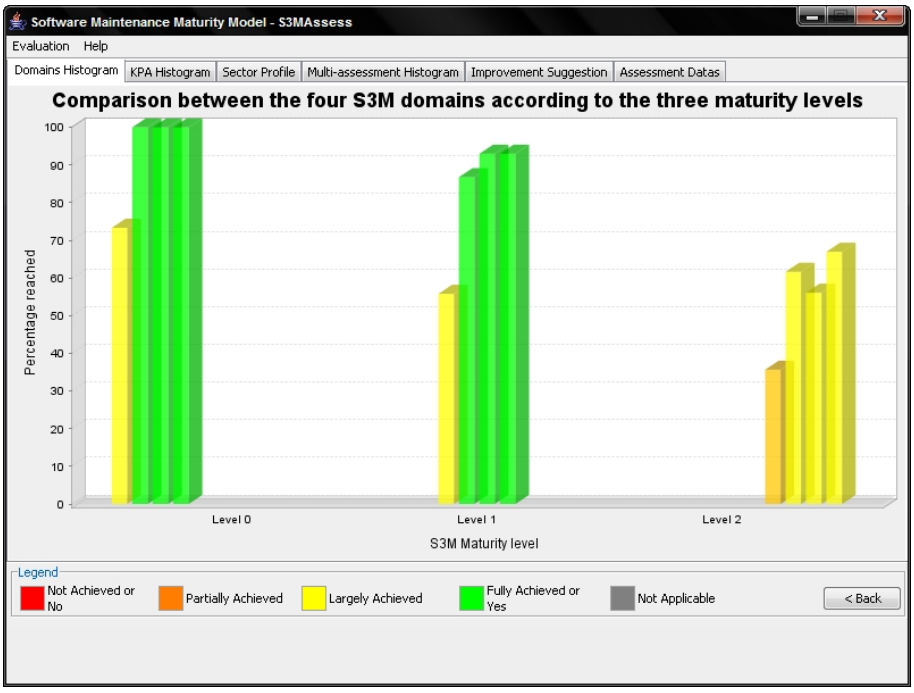


FIG. 8.2 – Histogramme des domaines du projet de maintenance de $S^{3mAssess®}$

En observant le graphique de la figure 8.2, deux caractéristiques apparaissent directement : une tendance forte se dessine autour d'une maturité générale atteignant le premier niveau de maturité $S^{3m®}$ et, le domaine traitant du management des processus de maintenance est le moins bien géré des quatre domaines $S^{3m®}$.
Dès lors, il serait recommandé d'améliorer les processus de maintenance selon l'ordre de priorité suivant :

1. Domaine 1, niveau 0 ;
2. Domaine 1, niveau 1 ;
3. *Domaine 1, niveau 2 ;
4. Domaine 3, niveau 2 ;
5. Domaine 2, niveau 2 ;
6. Domaine 4, niveau 2.

Cet ordre de priorité est fixé par rapport aux pourcentages atteints par les niveaux de maturité $S^{3m®}$ inférieurs. En effet, pour des domaines atteignant plus de 80% à un niveau de maturité $S^{3m®}$, l'ordre des priorités d'amélioration du niveau supérieur suit les pourcentages atteints dans ce niveau. Ainsi, les faibles pourcentages obtiennent une priorité supérieure. C'est la raison pour laquelle le domaine 1 au niveau 2 est marqué d'un astérisque car étant donné que ses niveaux 0 et 1 sont en-deçà des 80%, il serait recommandé de commencer par améliorer ce domaine dans ces niveaux puis de réeffectuer une évaluation afin de vérifier les effets des améliorations. La troisième position que le domaine 1 au niveau 2 obtient dans ce premier classement de priorités devrait donc à être vérifiée après les améliorations apportées à ce domaine et suite à une nouvelle évaluation.

Au final, deux premières conclusions de haut niveau pourraient être tirées sur base de ces premiers résultats.
La première est qu'il n'y aurait pas de collecte de données, d'étude spécifique, ni d'établissement de priorité d'amélioration et de déploiement. C'est ce que ce premier schéma peut laisser penser suite à la relative faiblesse du premier domaine. Les raisons exactes seront éclaircies à partir de la section suivante.
L'autre conclusion est que la maturité générale des processus de maintenance se situerait entre un niveau informel et un niveau formel où toutes les procédures sont définies à l'avance et documentées. La suite détaillera davantage la situation exacte.

8.5.2 Histogramme des itinéraires

Cette section va permettre de connaître les principaux itinéraires faisant baisser la moyenne des domaines. La présentation des comparaisons des itinéraires va suivre l'ordre de priorité proposé à la section précédente.

Itinéraires du domaine un aux niveaux de maturité zéro et un Ce sont donc tout d'abord les niveaux de maturité zéro et un du domaine $S^{3m®}$, traitant du management des processus de maintenance, qui seront analysés.

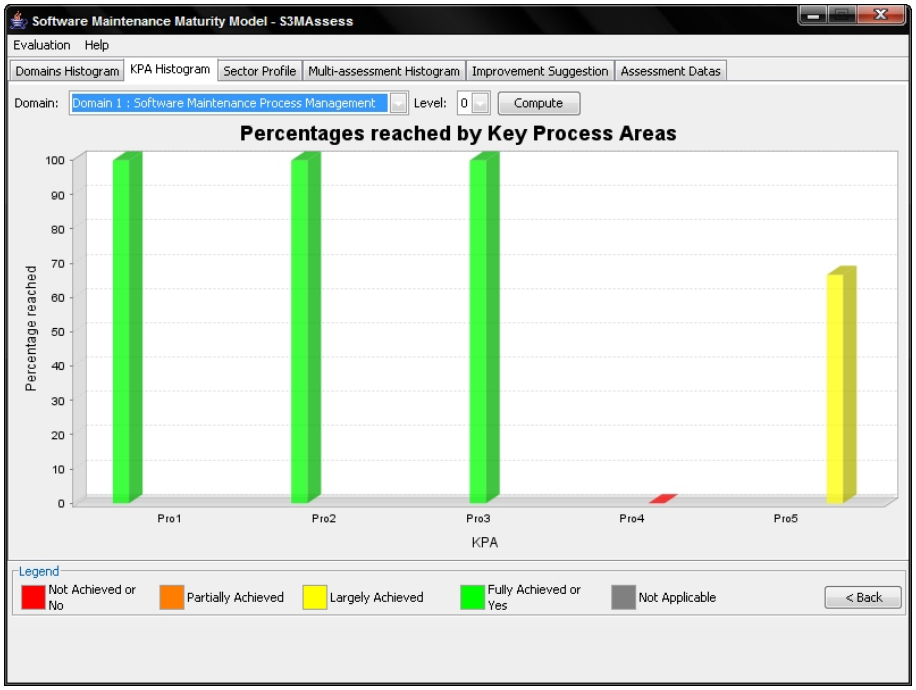


FIG. 8.3 – Histogramme des itinéraires du domaine 1 au niveau 0

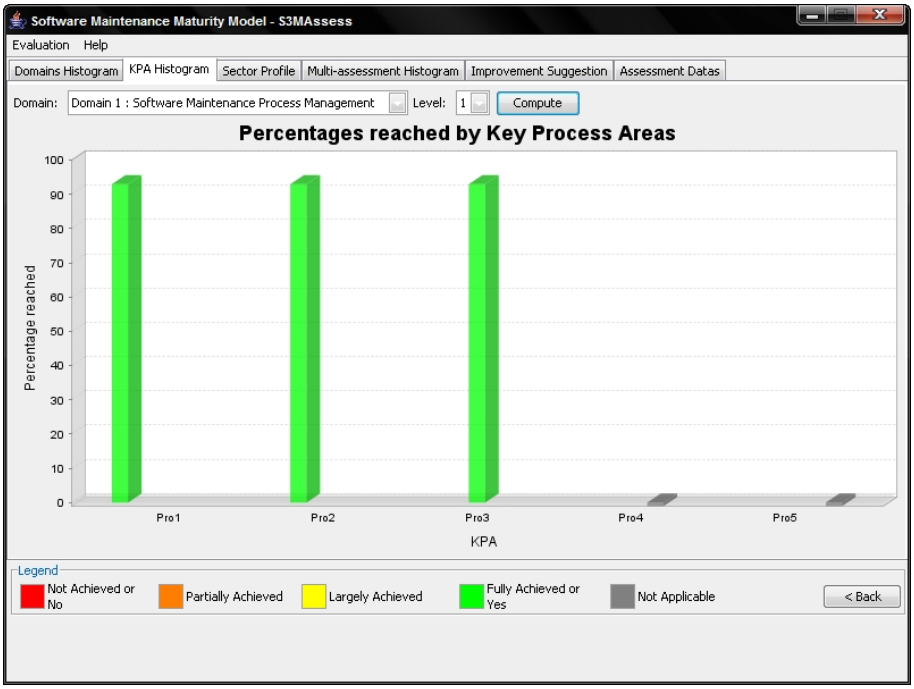


FIG. 8.4 – Histogramme des itinéraires du domaine 1 au niveau 1

Les deux figures 8.3 et 8.4 illustrent que les problèmes du premier domaine $S^{3m®}$ aux niveaux 0 et 1 viennent uniquement des itinéraires Pro4 et Pro5. Voici une explication à haut niveau de la faible cotation de ces itinéraires de compétence :

Pro4, Performance des processus de la maintenance : Il n'y a pas eu d'identification des processus et des activités clés de la maintenance devant faire l'objet d'une analyse de performance. Une méthodologie de maintenance a été avancée afin d'établir des objectifs à chaque étape mais cela n'a pas donné lieu à l'établissement d'un modèle de prédiction de la performance des processus.

Pro5, Innovation et déploiement : Si des améliorations ont été apportées lors de la conception de la méthodologie de maintenance, il n'y eut pas vraiment de mesure pour s'assurer que des bénéfices escomptés eurent été bel et bien récoltés. La modification de la méthodologie se faisait sur base des expériences observées mais il n'y eut pas de planification, de gestion de déploiement ni de mesure des effets d'amélioration.

Il est à préciser que ces deux itinéraires n'ont pas été évalués au deuxième niveau de maturité car ils n'atteignaient pas le seuil des 80% au premier niveau de maturité. La non-atteinte des objectifs proposés par ces deux itinéraires signifie que si des étapes ont été effectivement définies au sein d'une méthodologie de maintenance, ces dernières n'ont pas été mesurées par rapport à leur performance et les améliorations qu'elles apportaient n'ont pas été justifiées par des mesures.
Par rapport aux objectifs de haut niveau de ces itinéraires, il pourrait donc être opportun de mettre en place un système de mesures des étapes clés. Toutefois, il faut aussi se demander si de telles mesures sont réellement nécessaires dans le cadre d'une équipe de maintenance constituée d'une seule personne. Dans ce contexte, mettre en place des modèles de prédiction et des preuves de bénéfices mesurés par rapport à des investissements mesurés peut être assez lourd et risque d'augmenter considérablement le temps de création de chaque révision de l'outil. En effet, pendant que la seule ressource disponible effectue l'ensemble des mesures et documente les résultats, d'autres produits et services peuvent se dégrader comme le délai de création d'une révision, le service de support offert quotidiennement et au final, la qualité des produits et services car il n'y a plus de ressource disponible pour ces tâches. Le problème dans ce contexte est donc

la surcharge de travail.

La conclusion pour le contexte actuel du projet est donc que certaines mesures de processus pourraient être mises en place à l'avenir à condition que le bénéfice de leur mise en place soit plus important que leur investissement. Ces mesures pourraient s'avérer davantage pertinentes si les clients exigeaient un niveau de qualité spécifié dans un contrat pour s'assurer que chaque processus permette d'y répondre correctement.

Itinéraires du domaine un au niveau de maturité deux Il vient ensuite l'analyse du domaine un au niveau de maturité deux :

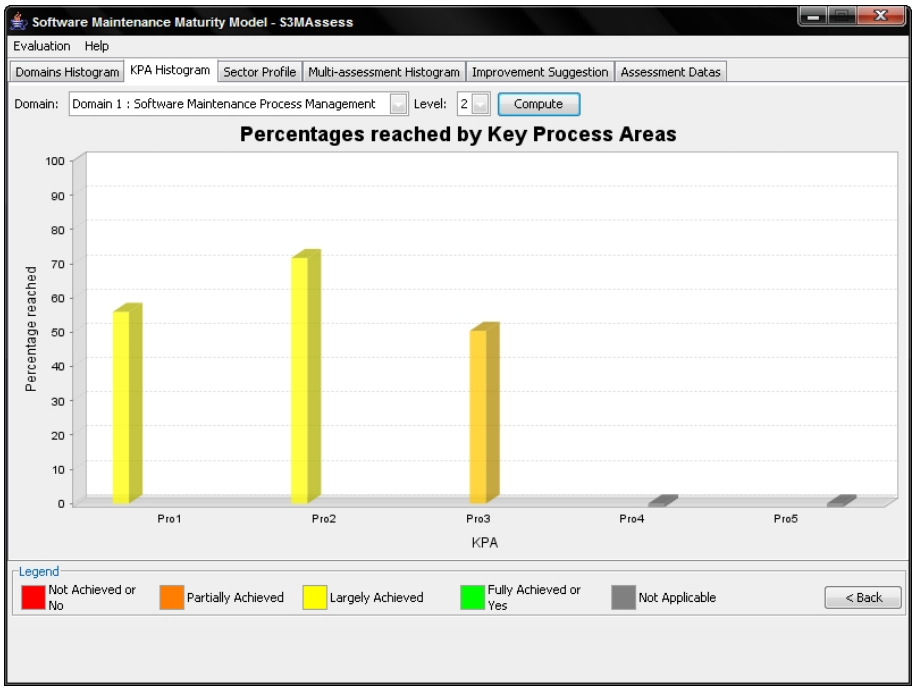


FIG. 8.5 – Histogramme des itinéraires du domaine 1 au niveau 2

Il se déduit de la figure 8.5 que la priorité des améliorations serait donc Pro3, Pro1 et Pro2 sous réserve des améliorations déjà apportées à ce premier domaine dans ses niveaux précédents. En considérant les deux premiers itinéraires de ce classement comme beaucoup plus faibles que le troisième, voici leurs conclusions à haut niveau :

Pro3, Formation des ressources de la maintenance : La formation du seul membre de l'équipe était plutôt, par définition, "informelle" et passait par la lecture de divers ouvrages de référence, normes internationales, discussions avec le maître de stage, etc. Il n'est pas vraiment adaptable à ce contexte d'une personne de formaliser un plan de formation.
Il n'y a pas eu de benchmarking pour mesurer la formation[3] et pas de formation des utilisateurs en dehors du guide utilisateur. Toutefois, la nature distante des relations avec les clients rend difficile une formation particulière à l'outil. De plus, les systèmes (l'outil, le mode de soumission des RMs, etc.) étant relativement simples à utiliser, une formation supplémentaire n'était pas vraiment nécessaire.

Pro1, Focalisation sur les processus de la maintenance : Des améliorations aux processus de maintenance ont été identifiées mais il n'y a pas eu réellement de donnée comparative.

[3] mais est-ce vraiment pertinent dans le cas d'un seul membre qui ne peut comparer sa formation avec d'autres membres ?

Un plan d'amélioration n'a pas été établi pour mettre en place la méthodologie de maintenance (voir sa description à la section 6.1) qui visait à améliorer les processus de maintenance.
Il n'y a pas eu d'audit interne et il s'agit ici de la première évaluation de maturité. Cela veut dire qu'il y a une perception de la nécessité d'évaluer la maturité des processus mais que d'éventuelles bonnes pratiques doivent encore être mises en place.

Les profils de secteur détailleront les pratiques incriminées dans ces itinéraires plus faibles. Toutefois, certains objectifs semblent encore une fois à la limite d'être adaptables au contexte de ce projet de maintenance. La formalisation d'un plan de formation et un benchmarking mesurant les performances de formation ne sont pas réellement pertinents si cela ne concerne qu'une seule personne, tout comme la formalisation d'un plan d'amélioration.
Les grandes idées derrière ces objectifs de niveau deux sont de faire prendre conscience à tous les membres d'une organisation de maintenance de ces intérêts de planification et de documentation, de les mettre en oeuvre d'une manière formalisée et de comparer les données afin de formaliser et de standardiser les meilleurs processus et activités.

Dans le cadre de ce projet de maintenance, bon nombre de pratiques du niveau deux ont malgré tout été évaluées afin de prendre conscience de la nécessité d'une certaine formalisation des processus de formation et d'amélioration. Cela permettra d'être prêt à améliorer ces points en cas de complexité augmentée du projet de maintenance de $S^{3mAssess®}$.

Itinéraires du domaine trois au niveau de maturité deux Le domaine trois au niveau deux n'obtenant que 56%, il est aussi utile d'en détailler la cause :

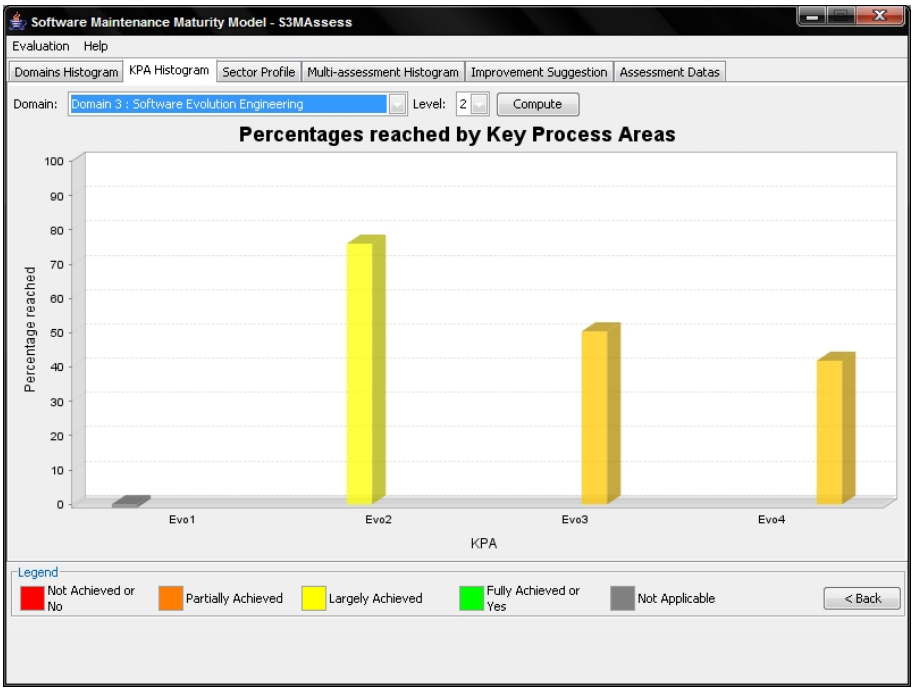


FIG. 8.6 – Histogramme des itinéraires du domaine 3 au niveau 2

L'itinéraire Evo2 obtient une moyenne très acceptable de 76%, les itinéraires Evo3 et Evo4 obtiennent par contre respectivement 50% et 42%. Voici donc quelques détails sur ces deux itinéraires plus faibles :

Evo3, Evolution/Correction du logiciel : La vérification que l'évolution et la correction du logiciel n'introduisait pas de défaut ni de défaillance se faisait de manière informelle et ne consistait donc pas en une planification formalisée avec des résultats documentés. La réalisation informelle de cet itinéraire explique sa faiblesse au deuxième niveau de maturité, lequel se veut plus formalisé. Les procédures de modification et de test furent en effet réalisées de manière informelle.
Par contre, la validation des analyses d'impact était formellement prévue dans une étape de la méthodologie de maintenance. Les résultats étaient évalués au début de toute phase de conception d'une révision de l'outil.
De même, un système de gestion de versions (SVN) des fichiers sources et des documents a été utilisé et permettait donc déjà de répondre à des besoins d'intégration même si cela ne s'est finalement pas avéré nécessaire car l'équipe est restée limitée à une seule personne. Ce système a néanmoins continué à être utilisé pour ses fonctions de sauvegarde et de régression.

Evo4, Vérification et validation : Une nouvelle fois, la faiblesse de cet itinéraire au niveau de maturité deux vient de sa réalisation informelle. Les activités de vérification et de validation sont bien planifiées mais leur procédure d'exécution est informelle.

En conclusion à ces deux derniers itinéraires, il y a conscience des bonnes pratiques à suivre, certaines activités ont été identifiées et planifiées mais leur réalisation reste néanmoins davantage à tendance informelle que formelle. Cela veut dire que les objectifs du niveau deux ne sont pas poussés à leur paroxysme dans la formalisation des définitions des procédures et dans la complétude de la documentation de leur réalisation et de leurs résultats.
Un tel niveau de maturité ne s'avérait pas forcément le plus pertinent dans le cadre de ce projet. Toutefois, le processus de maintenance gagnerait à formaliser davantage ses procédures de tests avec une complexité grandissante de l'outil et de la gestion du projet.

Pour résumé, l'ordre suggéré d'amélioration de ces itinéraires est donc le suivant :

1. Pro4, niveau 0 :
2. Pro5, niveau 0 ;
3. Pro3, niveau 2 ;
4. Pro1, niveau 2 ;
5. Evo4, niveau 2 ;
6. Evo3, niveau 2 ;

Les itinéraires Pro4 et Pro5 n'ayant pas été évalués au niveau un par cause de blocage par la méthode d'évaluation, il est recommandé de commencer par les améliorer au niveau zéro puis d'effectuer une nouvelle évaluation afin de déterminer leur capacité.
Les profils de secteur de la sous-section suivante détailleront les pratiques faibles et leurs possibilités d'améliorations.

8.5.3 profils de secteur

Cette sous-section propose une analyse des pratiques faibles classées par ordre de priorité d'amélioration des itinéraires. Pour cela, elle se base sur les résultats donnés à la section précédente et sur les figures 8.7 et 8.8.
Afin de suivre les recommandations données pour chaque pratique, il est recommandé de consulter le détail des pratiques dans le livre [April et Abran, 2006].

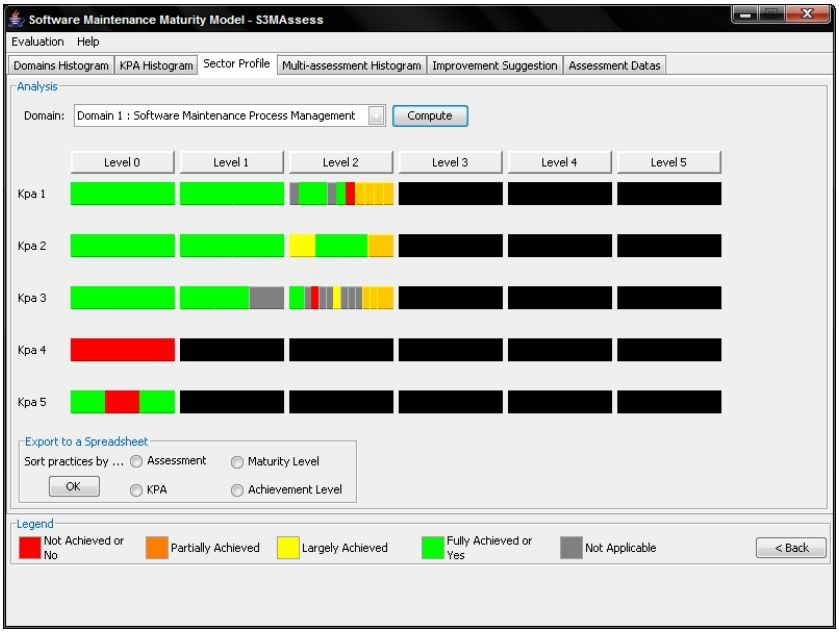


FIG. 8.7 – Secteur de profil des pratiques du domaine 1

Pro4, niveau 0, 0% : Cet itinéraire obtient 0% car une seule pratique existe à ce niveau et a été évaluée à "Non Atteint".

Pro4.0.1, Non Atteint : Il n'y a pas eu d'activité de mesure de performance des processus, services, produits et ressources. Ces mesures auraient été plus coûteuses en investissement qu'en bénéfice car elles ne sont pas tout à fait adaptées au contexte. Toutefois, elles pourraient être mises en place avec la complexité grandissante de l'outil et du projet.

Pro5, niveau 0, 67% :

Pro5.0.2, Non Atteint : Etant donné qu'il s'agit du niveau zéro, la réponse est exclusivement oui ou non. Des propositions d'amélioration et d'innovation ont bien entendu été étudiées tout au long de l'établissement de la méthodologie de maintenance mais il n'y a pas eu d'essais pilote suivant une procédure normalisée et mesurant et documentant tous les résultats. Les améliorations, après analyse informelle préalable, ont été apportées progressivement au cours du processus de maintenance.
Si le projet venait à se complexifier, il serait recommandé d'améliorer cette pratique en vue notamment de faire prendre conscience aux divers membres de l'équipe de maintenance de la nécessité des innovations et en vue d'être sûr d'employer les procédures les plus performantes.

Pro4 et Pro5, niveau 1, non évalués : Ces deux itinéraires n'ayant pas atteint le seuil de 80% au niveau inférieur, ils n'ont pas été évalués à ce niveau.

Pro3, niveau 2, 50% :

Pro3.2.4, Non Atteint : Il n'y a pas eu de benchmarking quant aux formations suivies par le seul membre de l'équipe. Le but réel de pratiquer une telle activité est de comparer les niveaux de formations entre unités organisationnelles et entre les différentes techniques de formation. Dans le contexte actuel, cela n'est donc pas vraiment adapté. Toutefois, la pratique a quand même été évaluée pour attirer l'attention sur la nécessité d'une formation performante.

Pro3.2.11, Partiellement Atteint : Il n'y a pas de procédure définie en ce qui concerne la mise à jour des produits de formation pour l'utilisateur. Toutefois, une seule personne effectuait toutes les mises à jour requises et suivait toujours le même formalisme.

Pro3.2.12, Partiellement Atteint : Les utilisateurs n'ont pas vraiment reçu de formation pour pouvoir utiliser le logiciel mais un guide utilisateur a été conçu et des démonstrations du logiciel eurent lieu. De plus, son utilisation reste assez simple si l'utilisateur comprend les concepts sous-jacents au modèle $S^{3m®}$.

Pro3.2.13, Partiellement Atteint : Les utilisateurs n'ont pas reçu de formation concernant les services qui leur étaient proposés. Toutefois, ces services étaient simples à utiliser. Ils pouvaient par exemple correspondre par email, par conférence téléphonique ou lors de réunion afin de communiquer leurs RMs. Il n'y avait pas de formulaire standardisé à remplir pour chaque RM.
Exiger un tel niveau de service serait pertinent si l'outil s'industrialisait et si son projet se complexifiait.

Pro1, niveau 2, 56% : Cet itinéraire est principalement sous-évalué à cause du fait que l'actuelle évaluation de maturité est la première effectuée.

Pro1.2.7, Non Atteint : Il n'y eut pas d'audit interne. Toutefois, ces audits internes doivent être réalisés par des mainteneurs seniors au sein d'une organisation de maintenance qu'ils connaissent bien. Ce n'est pas tout à fait possible dans le contexte actuel du projet. La pratique pourrait par contre être adaptée au contexte actuel en demandant à une personne expérimentée, le maître de stage par exemple, d'auditer le processus de maintenance en place. Cela a été partiellement réalisé aux cours des diverses réunions avec le maître de stage mais pas réellement de la manière préconisée par la pratique.
Par ailleurs, cette pratique sera utile à une éventuelle future équipe de maintenance travaillant sur ce projet.

Pro1.2.8, Partiellement Atteint : Il s'agit de la première évaluation de maturité effectuée. Les résultats sont analysés en vue d'améliorations futures.

Pro1.2.9, Partiellement Atteint : Il s'agit de la première évaluation de maturité et ce chapitre matérialise la liste des futures améliorations éventuelles.

Pro1.2.10, Partiellement Atteint : Même remarque que précédemment.

Pro1.2.11, Partiellement Atteint : Même remarque que précédemment.

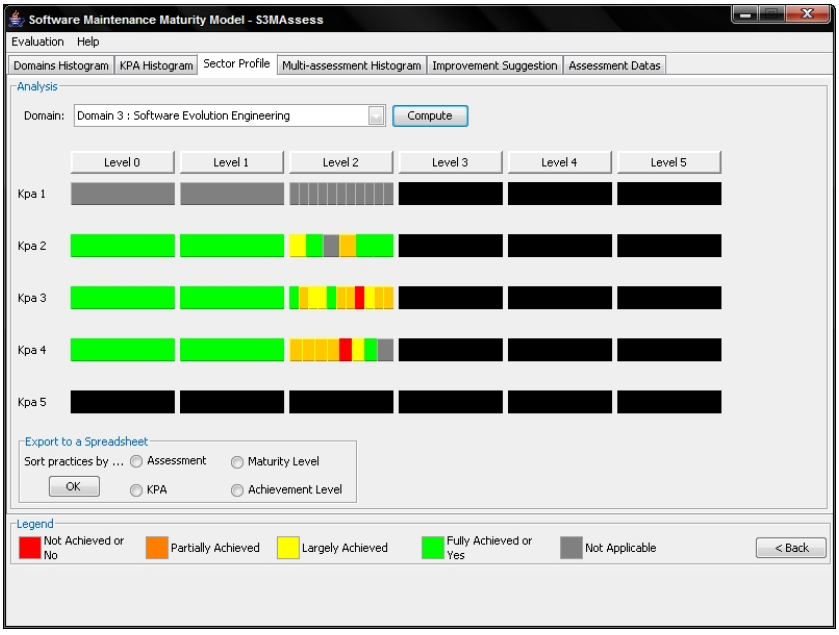


FIG. 8.8 – Secteur de profil des pratiques du domaine 3

A noter que l'itinéraire Evo1 a été évalué à "Non Applicable" étant donné l'absence de phase de transition dans le processus de maintenance.

Evo4, niveau 2, 42% :

Evo4.2.5, Non Atteint : Des tests de régression n'ont pas été systématiquement et formellement réalisés sur toutes les parties du logiciel affectées par des modifications. Seuls des tests informels de cas particuliers furent réalisés.

Evo4.2.1, Partiellement Atteint : Les tests ont été réalisés informellement. Il n'y avait pas de plan de test décrit formellement. Des démonstrations informelles eurent lieu (pour principalement valider l'outil par le maître de stage) et la mise en opération de chaque révision servait à chaque fois de validation par les clients.

Evo4.2.2, Partiellement Atteint : Vérifications et validations informelles lors de chaque révision de l'outil.

Evo4.2.3, Partiellement Atteint : Les revues n'étaient pas documentées : pas de procédure documentée et pas de document précis à remplir concernant les suggestions. Il y a soumission des produits aux clients et autres intervenants et ensuite, attente de leurs avis/suggestions. Les analyses d'impact sont les seules revues documentées lors des premières étapes du cycle de vie de maintenance.

Evo4.2.4, Partiellement Atteint : Les tests d'acceptation sont réalisés de manière informelle.

Evo3, niveau 2, 50% :

Evo3.2.8, Non Atteint : Il n'y avait pas de plan de test décrit, ni de résultat documenté. Les tests étaient informels.

Evo3.2.2, Partiellement Atteint : Pas de procédure documentée sur la façon d'enquêter sur les défaillances du logiciel mais certaines résolutions de problèmes donnèrent lieu à une meilleure documentation du code (via Javadoc).

Evo3.2.6, Partiellement Atteint : Seules les cinq plus importantes modifications ont leurs interrelations avec les procédures administratives, modules, processus et avec les données décrites dans

les analyses d'impact.

Evo3.2.7, Partiellement Atteint : Pas de plan de test formel.

Evo3.2.10, Partiellement Atteint : La documentation interne du logiciel est modifiée et mise à jour mais pas selon une procédure documentée. Toutefois, le même formalisme et la même façon de procéder ont été tenus par la seule et même personne affectée à ces tâches.

Evo3.2.11, Partiellement Atteint : Même remarque que précédemment pour la documentation externe du logiciel.

Ceci clôt le détail des pratiques faibles ordonnées par suggestions de priorité d'amélioration. Etant donné qu'il s'agit de la première évaluation de maturité de $S^{3mAssess®}$, il n'y a pas de comparaison possible avec une ancienne évaluation.
Ces résultats devraient être ensuite discutés avec le haut management (dans le contexte actuel, avec le maître de stage par exemple) afin de prendre les bonnes décisions d'amélioration.
La section suivante conclut ce chapitre en tirant les grandes conclusions quant aux résultats de cette première évaluation de maturité de $S^{3mAssess®}$.

8.6 Conclusion

En conclusion, étant donné le contexte du projet, les niveaux de maturité observés restent très acceptables. Les résultats ont tendance à se situer entre les niveaux de maturité $S^{3m®}$ un et deux, c'est à dire que les objectifs informels sont largement atteints et que les améliorations possibles se situent au niveau de l'atteinte des objectifs de formalisation proposés par le niveau deux. A ce niveau de maturité, les pratiques faibles ont souvent été évaluées à "Partiellement Atteint" et commentées comme étant "informelles", ce qui veut dire que certains objectifs sont atteints mais pas d'une manière formellement documentée.
En fait, beaucoup de pratiques ne sont pas vraiment adaptées au contexte actuel du projet de maintenance mais ont néanmoins été évaluées pour attirer l'attention sur leur nécessité d'autant plus importante si le projet se complexifie à l'avenir. Certaines d'entre elles donnent également des idées d'amélioration (par exemple, la formalisation des améliorations des processus de maintenance ou la formalisation de la formation) si pour autant elles sont adaptées au contexte actuel du projet de maintenance.

Dans le futur, si une équipe de maintenance aux ressources plus importantes faisait l'acquisition de l'outil, il pourrait être recommandé de prévoir des procédures plus documentées et plus formelles :

- des planifications de transition, de livraison et de mise en opération ;
- de coordination du travail entre les membres de l'équipe de maintenance ;
- prévoir des actions correctives sur les engagements de services/produits s'ils ne sont pas tenus ;
- Etablir des plans de test, en décrire les procédures et documenter les résultats (formalisation des tests) ;
- procéder à des analyses d'impact complètes ;
- formaliser les formations ;
- formaliser les améliorations apportées aux processus de maintenance (via un plan d'amélioration et via la documentation des procédures, des objectifs et des résultats), les mesurer pour les comparer et pour finalement standardiser les améliorations les plus performantes au sein de l'organisation de maintenance.

En partant d'un logiciel plus solide et mieux documenté et en étant plus nombreux, il sera plus adapté pour une organisation de suivre davantage de procédures documentées et formelles. Les ressources étant plus importantes, ce formalisme aidera dans la planification et dans la coordination du travail. De même, la qualité quant aux résultats des produits/services aura moins de risques de diminuer à cause d'une plus grande complexité de gestion des tâches.
De plus, suite à cette première évaluation de maturité de $S^{3mAssess®}$, les futurs mainteneurs auront main-

tenant une base d'amélioration possible et un référent pour leurs processus de maintenance.

Par contre, des améliorations pertinentes et applicables dès le contexte actuel pourraient être principalement la rédaction complète des analyses d'impact et la formalisation des tests. Etant donné que les longues phases de rétro-ingénierie et de correction des nombreuses défaillances sont achevées, davantage de temps pourrait être investi à la mise en place de ces améliorations.

Finalement, suite aux études de cas et aux diverses analyses, le chapitre suivant présente un ensemble d'améliorations possibles du modèle $S^{3m®}$ et de l'outil $S^{3mAssess®}$.

Chapitre 9

Travaux futurs

Ce chapitre reprend toutes les suggestions d'améliorations, tant du modèle que de l'outil de support, ayant pu surgir à la suite des expérimentations présentées dans cette partie mais également à la suite des recherches effectuées au cours de la partie précédente.

9.1 Possibilités d'améliorations du modèle $S^{3m®}$

Les recherches ainsi que les expérimentations effectuées au cours de ce mémoire ont permis de mettre en oeuvre le modèle de maturité. Suite à cela, certaines suggestions d'améliorations du modèle ont pu surgir. Il est intéressant dans le cadre de ce mémoire de revenir sur ces suggestions. En effet, l'un des buts fondamentaux du projet est de pouvoir contribuer à l'évolution de la maintenance du logiciel. Contribuer à l'amélioration du modèle contribue donc par la même occasion à l'atteinte de cet objectif.
De plus, l'outil étant principalement basé sur ce modèle, améliorer ce dernier suggérera par la suite d'implémenter ces améliorations au sein de $S^{3mAssess®}$.

- L'intégration des limites d'un projet dans le diagramme de contexte $S^{3m®}$ est importante dans le cadre de l'évaluation de maturité du processus de maintenance car il implique inexorablement des contraintes contractuelles et de retour-sur-investissement. Le projet étudié au chapitre 7 impliquait une société de sous-traitance et ne constituait donc pas le contexte d'une division IT oeuvrant en interne d'une société mère. Ces deux contextes peuvent être bien distincts de part leurs objectifs et leur fonctionnement. Si une division interne n'oeuvrant exclusivement que pour une société mère et comptant des centaines d'employés peut envisager de développer à son paroxysme chacune des pratiques proposées par le modèle $S^{3m®}$, une société d'outsourcing devra faire des choix d'amélioration sur certaines pratiques selon son propre contexte spécifique, selon des critères de temps et de ROI. A titre d'exemple, si le contexte d'une société d'outsourcing inclut de nombreuses personnes externes, il lui sera éventuellement moins intéressant de dépenser du temps et de l'argent à la formation de ces membres. En effet, cela pourrait revenir à former gratuitement les employés d'une société tierce. Dans ce cas, la société d'outsourcing cherchera plutôt à engager des employés déjà formés et compétents.
  Ce genre de différence importante liée au contexte doit absolument être pris en compte lorsqu'on pratique une évaluation de maturité selon $S^{3m®}$. Une idée d'amélioration du modèle pourrait être de développer un second modèle spécifiquement adapté au contexte moderne des entreprises IT d'outsourcing.
  Il est à préciser que cette réflexion ne concerne pas le retrait de certaines pratiques dû à l'absence d'interfaces dans le diagramme de contexte mais bien à une formulation plus adaptée de certaines pratiques. L'élaboration de pratiques typiquement adaptées au contexte de la sous-traitance permettrait une évaluation plus directe et plus pertinente.

- Il existe différents points de vue quant à l'amélioration de processus. Le modèle $S^{3m®}$ vise à améliorer les pratiques en vue d'obtenir des niveaux de maturité "plats", c'est à dire sans pratique très développée en présence d'autres ne l'étant quasiment pas. Il s'agit d'un objectif d'amélioration assez théorique. En pratique, de nombreuses entreprises ont intérêt à améliorer certaines pratiques par rapport à d'autres. Ces intérêts viennent de critères économiques, d'objectifs d'affaires, de disponibilité de ressources, etc.
  Formaliser absolument toutes les procédures, documenter les moindres aspects de ce qui touche au logiciel maintenu, chercher sans cesse à améliorer tous les processus en effectuant toutes sortes de mesures comparatives, planifier de nombreux essais pilotes, etc. mènera bien entendu à une organisation de maintenance très mature ayant un contrôle maximum sur tous les processus et produits/services. Cela permet également d'assurer au maximum certains niveaux de qualité. Toutefois, il faut bien tenir compte du fait qu'un tel développement de maturité demande des efforts d'investissements gigantesques.
  L'entreprise doit alors se poser les bonnes questions et déterminer si certaines améliorations ne sont pas au final moins bénéfiques pour son contexte. Il peut lui être plus adapté d'améliorer certains points très précis à la manière de la représentation continue proposée dans le CMMi.
  A cette fin, il pourrait être utile aux entreprises de préciser les investissements que demandent l'établissement de chaque pratique. C'est d'ailleurs l'une des raisons pour laquelle l'outil ne fait que suggérer des sous-ensembles de pratiques faibles sans imposer une priorité d'amélioration.
  De plus, préciser les interdépendances pouvant exister entre les pratiques $S^{3m®}$ peut également être utile. L'idée des interdépendances est le sujet du point suivant.

- Le modèle $S^{3m®}$ préconise d'établir des niveaux de maturité "plats", quels que soient les processus choisis pour une évaluation de maturité. Par ailleurs, un diagramme hiérarchique des dépendances entre niveaux de capacité des processus permet de prédéfinir un squelette d'amélioration des processus de maintenance[1]. Développer une hiérarchie de dépendances entre pratiques pourrait aider les organisations à faire les bons choix d'amélioration en ayant conscience de toutes les implications et de tous les effets de bord. Il serait donc utile d'ajouter au modèle $S^{3m®}$ une telle hiérarchie de dépendances.

- Définir des attributs de processus pour permettre une évaluation plus formelle et plus objective. Quelques problèmes ont parfois surgi quant à savoir quelle cotation donner à une pratique $S^{3m®}$. Un exemple de problème est : *Quelle cotation donner à une pratique lorsqu'un élément se produit une fois sur deux au cours d'un processus ? "Partiellement Atteint" ou "Largement Atteint" ?*
  L'idée d'indicateurs de capacité et de performance des processus proposée par la norme ISO 15504 (voir en particulier la partie quatre de [ISO/IEC, 1998]) pourrait être utile à l'amélioration du processus d'évaluation. Les évaluateurs auraient la capacité d'interpréter au mieux l'application des pratiques $S^{3m®}$ par l'organisation évaluée. De plus, les informations pertinentes seraient assurées d'être capturées pour une analyse future. Enfin, les résultats d'évaluation seraient encore plus représentatifs, fiables et répétables.
  Pour davantage d'explications sur les indicateurs, voir la partie *5. Utilisation des indicateurs* en annexe.

- Il serait utile d'exprimer les pratiques du niveau de maturité zéro sous forme positive afin de les évaluer plus intuitivement. Ainsi, l'évaluateur coterait une pratique à "oui" si la pratique est bien rencontrée dans l'organisation de maintenance.
  Cette recommandation a déjà été appliquée à l'outil $S^{3mAssess®}$.

- Certaines difficultés ont été rencontrées dans l'interprétation de certaines pratiques. Il n'était par-

[1] Pour plus de détails à ce sujet, voir l'article [Adamonis *et al.*, 2005].

fois pas intuitif d'évaluer une pratique à "Totalement Atteint" lorsque le processus implanté dans l'organisation rencontrait des objectifs encore plus poussés que ce qui était décrit.
Par exemple, la pratique Pro3.1.1 incite simplement à répondre "oui" ou "non" alors qu'elle fait partie du niveau de maturité un et qu'il faut donc la coter selon l'échelle NPLF.
Un autre exemple est lorsque les pratiques demandent l'accomplissement des objectifs à un niveau informel alors qu'ils sont réalisés dans l'organisation à un niveau formel (donc meilleur).
Par conséquent, la réexpression plus intuitive de ce genre de pratiques pourrait aider l'évaluateur dans sa tâche.

- Une dernière suggestion serait de pouvoir utiliser le modèle $S^{3m®}$ afin de faire un choix parmi les opportunités clientes de projet de maintenance. Cette suggestion intéressera particulièrement les entreprises d'outsourcing devant choisir un projet de maintenance. Ces entreprises veulent en effet limiter leurs risques en n'acceptant pas de projet comportant d'importantes défaillances, tant dans les applications à maintenir que dans les processus clients de maintenance.
  Une idée d'amélioration pourrait passer par une sorte de "check-list" permettant à l'organisation de maintenance de se faire rapidement une idée sur les risques d'un nouveau projet.

9.2 Travaux futurs à apporter à l'outil $S^{3mAssess®}$

Cette section recense la liste des améliorations qu'il serait possible d'apporter à l'outil dès maintenant. Comme précédemment, cette liste fait suite aux recherches et expérimentations effectuées au cours de ce mémoire. La future équipe de maintenance pourra donc se baser sur ces résultats afin de démarrer son projet.

- Envisager d'adapter l'architecture à l'environnement distant de l'outil. Il pourrait s'avérer utile de séparer la couche Présentation de la couche Traitement. Le client ne devrait effectuer que des tâches d'affichage tandis qu'un serveur distant de Traitement communiquerait localement (à l'ETS par exemple) avec le serveur de base de données. Cela permettrait d'augmenter la rapidité de traitement et la sécurité puisque le client n'accèderait plus directement à la base de données. Seuls des résultats chiffrés seraient renvoyés au client pour son interface graphique, lui évitant ainsi un nombre élevé de requêtes de base de données très coûteuses en temps. Une architecture 3-tiers pourrait contribuer à ces exigences.

- Déterminer automatiquement les pratiques afférentes à une interface non existante dans le diagramme de contexte de l'entreprise. Pour cela, il faudrait également donner la possibilité à l'utilisateur de renseigner à l'outil les interfaces présentes dans le diagramme de contexte de l'organisation évaluée. Ensuite, l'outil inscrirait automatiquement, lors du processus d'évaluation, un message du type "interface non existante" dans le champ de commentaire des pratiques concernées. Cela permettrait d'éviter le risque que l'évaluateur sous-évalue incorrectement une pratique s'il ne fait pas attention au fait qu'elle se réfère à une interface inexistante dans le contexte de l'organisation évaluée.

- Il pourrait être utile d'offrir la possibilité aux utilisateurs d'accéder aux détails des pratiques $S^{3m®}$ (c'est à dire aux explications supplémentaires au simple énoncé de la pratique) pendant le processus d'évaluation. De cette manière, plus aucun document ne serait nécessaire à la réalisation du processus d'évaluation, toutes les informations d'évaluation étant accessibles à partir de l'outil.

- Afficher les facettes du modèle $S^{3m®}$ lors de l'évaluation des pratiques afin de pouvoir mieux contextualiser ces dernières.

- Envisager la possibilité d'évaluer des facettes, et pas seulement des itinéraires entiers, pour offrir

encore plus de réduction et de concentration sur un groupe limité de pratiques.

- Réafficher l'énoncé des pratiques dans les bulles d'informations présentes dans l'interface *Sector Profile*. Cela facilitera la tâche d'analyse des résultats d'évaluation.

- Etablir un lien entre l'outil de support aux évaluations, $S^{3mAssess®}$, et l'outil de support d'aide aux décisions d'améliorations, S3MExpert. Cette tâche est désormais simplifiée grâce à la nouvelle fonctionnalité $S^{3mAssess®}$ d'exportation des résultats vers une feuille de calculs.

- Réaliser des tests unitaires formels, via JUnit par exemple, et intégrer une étape de tests formels dans la méthodologie de maintenance établie dans la partie précédente.

Conclusion générale du mémoire

La question principale de ce projet est de répondre de manière pertinente à des entreprises voulant évaluer la maturité de leurs processus de maintenance. Comment les aider dans cette tâche et comment les aider à analyser leurs résultats ?

Pour y répondre, des normes internationales, des modèles, des articles, des outils, etc. existent déjà. Toutefois, l'ensemble de ces connaissances et outils peuvent s'avérer très dispersés et leurs notions peuvent également être confuses pour une organisation de maintenance souhaitant s'évaluer rapidement. Cette dernière n'aura peut-être pas le temps d'examiner minutieusement ce qui existe et éprouvera certainement des difficultés à appliquer ces connaissances de manière fiable. C'est pourquoi, un modèle d'évaluation de la maturité des processus de la maintenance du logiciel, le $S^{3m®}$, a été développé. Ce modèle vise justement à aider les entreprises dans leurs démarches de détermination de maturité ou d'amélioration de leurs processus de maintenance.
Afin d'aider encore davantage les organisations de maintenance, le modèle peut être complété par un outil de support aux évaluations. Cet outil pourrait permettre d'automatiser un processus d'évaluation et surtout, de fournir instantanément des résultats d'analyses d'évaluations. Ainsi, avec une connaissance minimale du modèle $S^{3m®}$ et en étant méthodique, toute entreprise aurait l'occasion de répondre relativement facilement à ses interrogations et disposerait d'informations pertinentes sur ses forces, ses faiblesses et sur les améliorations envisageables. C'est l'enjeu premier de ce projet.

Afin de poser plus précisément l'ensemble des enjeux du projet, il est utile d'en dresser une liste :

1. Rechercher, rassembler, synthétiser et automatiser toutes les sources pertinentes permettant de répondre aux besoins d'évaluation de maturité des processus de la maintenance du logiciel et d'améliorer ces processus.
2. Déterminer comment mener à bien un projet de maintenance incluant une phase relativement importante de réingénierie, lorsque la documentation est largement insuffisante, lorsque les développeurs ne sont plus joignables, lorsque les clients sont pressés d'obtenir un résultat et quand l'effectif de l'équipe de maintenance est très réduit.
3. Déterminer comment rendre compatible l'outil expérimental avec la norme ISO/IEC 15504.
4. Déterminer comment améliorer le processus d'évaluation utilisé dans $S^{3mAssess®}$.
5. Rechercher comment fournir une analyse complète et pertinente qui facilite la prise de décision quant à l'amélioration des processus de maintenance ou qui facilite la détermination de la capacité des processus de maintenance.
6. Intégrer une nouvelle méthode d'évaluation dans l'outil tout en préservant sa cohérence et sa stabilité.
7. Déterminer comment valider l'ensemble des solutions proposées.
8. Pour l'ensemble de ces questions, toujours rester concentré sur les besoins réels des entreprises.

Pour répondre à l'ensemble de ces questions, une méthodologie de recherche à tout d'abord été établie et présentée dans l'introduction de ce mémoire. C'est alors qu'une structure en quatre parties a été avancée :

1. Une revue de la littérature résumant une base théorique couvrant l'ensemble des questions de recherche ;
2. L'analyse des problématiques soulevées et le développement des solutions ;
3. L'expérimentation des solutions afin de les valider et d'établir d'éventuelles perspectives de travaux futurs ;
4. Les annexes comportant tous les détails non nécessaires à la compréhension et au développement du raisonnement principal.

La première partie du mémoire définissait donc la maintenance du logiciel, résumait une norme de référence sur l'évaluation des processus du logiciel, synthétisait un modèle permettant d'évaluer des processus de maintenance en vue de les améliorer et, définissait des notions de réingénierie. A l'issue de la deuxième partie du mémoire, la présence de ces notions s'avérait effectivement justifiée. En effet, définir et délimiter exactement la maintenance du logiciel a été nécessaire pour pouvoir comprendre et répondre aux attentes des organismes de maintenance mais aussi pour accomplir le travail réalisé sur l'outil de support, $S^{3mAssess®}$.
Quant à la norme internationale ISO 15504, elle donna les armes nécessaires à la création de solutions pertinentes, complètes et fiables. L'outil de support aux évaluations devant être utilisé par des entreprises pouvant suivre des objectifs risqués, impliquant des allocations de ressources en vue d'améliorer leurs processus de maintenance, il était primordial que les processus d'évaluation et d'analyse des résultats soient pertinents et solides. Une attention toute particulière a donc été portée sur le suivi des recommandations de cette norme.
Ensuite, Le modèle $S^{3m®}$ constituait évidemment le coeur de ce mémoire car les processus d'évaluation et d'analyse de l'application sont directement basés sur ce modèle. L'outil implémente en fait les pratiques du modèle, sa cotation NPLF et est basé sur ses idées de résultats finaux. Bien entendu, ces points principaux ont été plus ou moins adaptés et/ou approfondis au cours de l'évolution de la recherche matérialisant la deuxième partie. D'ailleurs, la troisième partie du mémoire indiqua des idées d'évolution de l'outil, mais également du modèle, en vue de coller encore davantage à la réalité pratique et aux besoins des organisations.
Finalement, la maintenance de $S^{3mAssess®}$ a démontré qu'il était très difficile d'envisager un tel processus sans jamais avoir recours à des techniques de réingénierie. En effet, tout mainteneur doit un jour se confronter à l'appropriation d'un logiciel à maintenir et l'expérience a montré que peu de processus de maintenance sont réellement documentés et peuvent se targuer d'être à un niveau de maturité en optimisation, c'est à dire au cinquième et dernier niveau proposé par $S^{3m®}$. Par conséquent, un minimum de rétro-ingénierie s'avère souvent nécessaire.

La deuxième partie utilisait les bases théoriques de la première partie comme explicité ci-dessus. Elle visait également à présenter tant le domaine d'application de l'outil que son contexte technique et détaillait ses phases de maintenance et d'évolution. Une méthodologie de maintenance a ainsi été spécifiquement conçue pour répondre aux problématiques rencontrées lors du projet. La réingénierie de $S^{3mAssess®}$ a été analysée. L'architecture particulière de l'application a été disséquée et critiquée sous l'angle de la maintenabilité. Finalement, les nombreuses évolutions développées ont été décrites et justifiées selon les besoins réels des entreprises et selon les recommandations de la norme ISO 15504.

Finalement, la troisième partie du mémoire visait à valider les solutions conçues en les confrontant directement à un cas réel d'entreprise et en les utilisant elles-mêmes pour effectuer une auto-évaluation de maturité. Suite à ces deux dernières études mais également suite à l'ensemble des recherches effectuées, des propositions d'améliorations, tant du modèle $S^{3m®}$ que de son outil de support $S^{3mAssess®}$, ont été avancées.

A l'issue de ce projet, l'objectif final est d'avoir pu contribuer à l'avancée des recherches sur l'amélioration de la maintenance du logiciel. Plus spécifiquement, l'objectif est d'avoir pu développer un outil indispensable aux processus d'évaluation de la maturité des processus de la maintenance du logiciel et d'analyse des résultats d'évaluations. Cet outil se veut le plus adapté possible aux besoins de chaque entreprise et le plus compatible possible avec les recommandations de la norme internationale ISO 15504 tout en se basant bien entendu sur le modèle $S^{3m®}$.
Enfin, grâce aux activités de redocumentation du logiciel réalisées au cours de la deuxième partie du mémoire, à l'évaluation de maturité réalisée au chapitre 8 et aux listes de suggestions présentées au chapitre 9, la prochaine équipe de maintenance de $S^{3mAssess®}$ pourra débuter son projet sur des bases solides.

Bibliographie

[Abran et Nguyenkim, 1993] Abran, A. et Nguyenkim, H. (1993). *Measurement of the Maintenance Process From a Demand-Based Perspective*, pages 63–90. Journal of Software Maintenance : Research and Practice, Vol. 5, n°2.

[Adamonis *et al.*, 2005] Adamonis, A., Mitasiunas, A., Naujikas, I., Ragaisis, S. et Reingardtas, M. (2005). Dependencies of processes' capability levels. ISSN 1392 - 124X Information Technology and Control, Vol.34, number 2A.

[Agency, 2001a] Agency, C. C. . T. (2001a). *Information Technology Infrastructure Library.* Service Support 09/2000, London, HSMO Books.

[Agency, 2001b] Agency, C. C. . T. (2001b). *Information Technology Infrastructure Library (ITIL)*, pages 59–117, 211–292. Service Delivery 09/2000, Norwich, Controller of Her Majesty's Stationary Office.

[April, 2005] April, A. (2005). Sm^{mm} model to evaluate and improve the quality of the software maintenance process. Mémoire de D.E.A., École de Technologie Supérieure (ÉTS) Montréal.

[April et Abran, 2006] April, A. et Abran, A. (2006). *Améliorer la maintenance du logiciel.* Loze-Dion.

[Basili *et al.*, 1986] Basili, V., Selby, R. et Hutchens, D. (July 1986). *Experimentation in Software Engineering*, volume SE-12, pages 733–743. IEEE Trans. on Software Engineering.

[Bourque et Côté, 1991] Bourque, P. et Côté, V. (1991). *An experiment in Software Sizing with Structured Analysis Metrics*, pages 159–172. Journal of Systems and Software.

[Chikofsky et II, 1990] Chikofsky, E. J. et II, J. H. C. (1990). Reverse engineering and design recovery : A taxonomy. 7(1):13–17.

[Habra, 2006] Habra, N. (2006). *Ingénierie du Logiciel.* University of Namur.

[Hainaut, 2002] Hainaut, J.-L. (2002). *Ingénierie des bases de données - matières approfondies.* University of Namur.

[ISO/IEC, 1998] ISO/IEC (1998). Information technology : Software process assessment - part 1 to 9. Rapport technique, ISO/IEC 15504.

[ISO/IEC, 1999] ISO/IEC (1999). Software engineering - product quality - part 1 : Quality model. Rapport technique, ISO/IEC 9126.

[ISO/IEC, 2001] ISO/IEC (2001). Software engineering - product quality - part 2 to 4. Rapport technique, ISO/IEC 9126.

[ISO/IEC, 2005] ISO/IEC (2005). Fdis 14764 - software engineering – software life cycle processes – maintenance. Rapport technique, ISO/IEC 14764.

[Kajko-Mattsson, 2001] Kajko-Mattsson, M. (2001). Towards a business maintenance model. *In Proceedings of the IEEE International Conference on Software Maintenance*, pages 500–509.

[Kajko-Mattsson *et al.*, 2001] Kajko-Mattsson, M., Westblom, U. et Forssander, S. (2001). Taxonomy of problem management activities. *In Fifth European Conference on Maintenance and Re-Engineering*, pages 1–10.

[Lehman, 1980] Lehman, M. (1980). *On Understanding Laws, Evolution, and Conversation in the Large-Program Life Cycle.* The Journal of Systems and Software.

[malcolm baldrige national quality program, 2005] malcolm baldrige national quality PROGRAM (2005). Criteria for performance excellence. [En ligne] : <http://www.quality.nist.gov/>.

[Niessink *et al.*, 2002] NIESSINK, F., CLERK, V. et van VLIET, H. (2002). The it service capability maturity model. Release L2+3-0.3, [En ligne] : <http://www.itservicecmm.org>.

[Paquette *et al.*,] PAQUETTE, D.-A., APRIL, A. et ABRAN, A. Assessment results using the software maintenance maturity model (s^{3m}). Ecole de Technologie Supérieure.

[Paulk et Curtis, 1993] PAULK, M. et CURTIS, B. (1993). *CMM for Software v1.1.* Software Engineering Institute.

[Pigoski, 1996] PIGOSKI, T. M. (1996). *Practical software maintenance : best practices for managing your software investment.* Wiley computer publishing.

[Pigoski et April, 2004] PIGOSKI, T. M. et APRIL, A. (2004). *Guide to the Software Engineering Body of Knowledge - SWEBOK - 2004 Version*, chapitre Software maintenance. IEEE Computer Society Press.

[SEI, 2001a] SEI (2001a). *Capability Maturity Model Integration, V1.1.* Software Engineering Institute.

[SEI, 2001b] SEI (2001b). *Standard CMMI^{SM} Appraisal Method for Process Improvement (SCAMPI^{SM}), Version1.1 : Method Definition Document.* Carnegie Mellon.

[Sperko, 2003] SPERKO, R. (2003). *Java Persistence for Relational Databases.* Apress.

Quatrième partie

Annexes

1. Guide d'introduction au logiciel $S^{3m}Assess_{®}$

Multi-Assessment Support Tool: Software Maintenance Maturity Model $S^{3mAssess®}$

Cédric Di Tomaso
Student in Master of Computer Science
F.U.N.D.P. Belgium

Alain April
PhD

Ecole de Technologie Supérieure
cedric.ditomaso@gmail.com
alain.april@etsmtl.ca

1 Introduction

That article's purpose is a presentation of the $S^{3mAssess®}$ software and it will focus on its main functionalities. That tool allows to assess organizations according to the Software Maintenance Maturity Model $S^{3m®}$. That document will illustrate in some stages how to install the tool, how to use it and what are its current functionalities. The version presented here is the third revision from the version 0.02.

2 To install and to run the software

That section is only useful for the installable version of the tool in contrast with the distant web version.

1. First, extract the archive "S3m.zip"
2. Install the "mysql-5.0.45-win32.zip" and then the "mysql-gui-tools-5.0-r12-win32.msi" file situated in the tools directory. The first file is to install the database server and the second one is to install tools allowing to administrate the database (MySQL Administrator and MySQL Query Browser are the most important in our case)
3. During the installation, choose "localhost" as Server Host and "root" for the Username and also for the Password when requested. "Localhost" is to specify that the server will run on your personal computer and "root" will be used as Username and as Password to connect us to the database server within the MySQL Administrator tool or within the MySQL Query Browser tool.
   Following, the description of the main steps for the installation of the database server :
   (a) Choose the complete installation

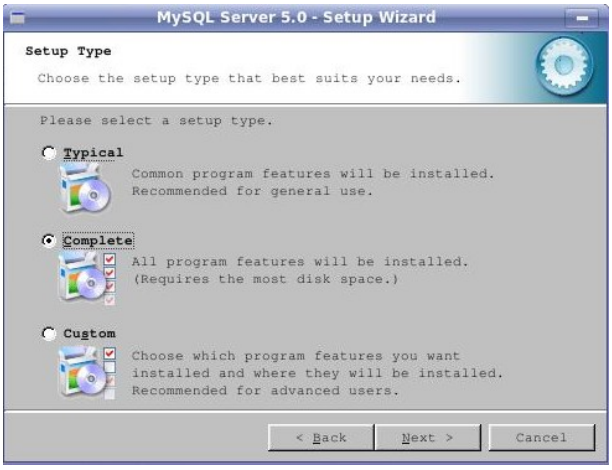


FIG. 1 – Choose "Complete"

(b) When the installation of the server is completed, let "Configure the MySQL Server now" selected

FIG. 2 – let "Configure the MySQL Server now" selected

(c) Choose "Standard Configuration"

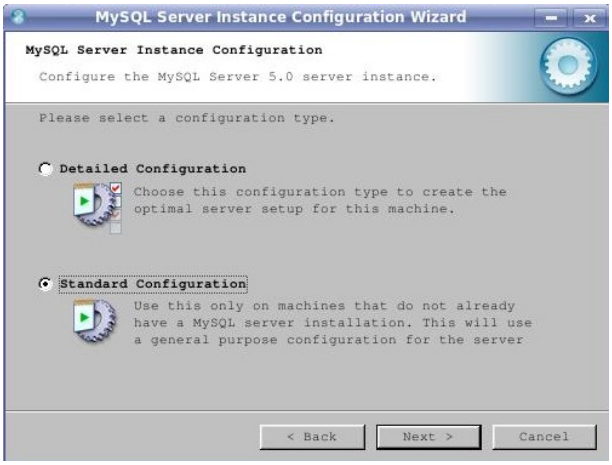


FIG. 3 – Choose "Standard Configuration"

(d) Select "Install As Windows Service" and select "MySQL5" as Service Name

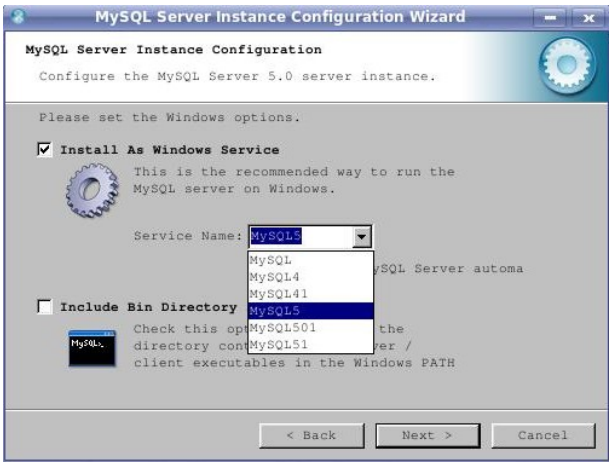


FIG. 4 – Install As Windows Service

(e) In the security options, choose "Modify Security Settings" and enter "root" two times

FIG. 5 – Modify Security Settings

(f) When the wizard is terminated, do not forget to install "mysql-gui-tools-5.0-r12-win32.msi" that will allow to execute the sql file of $S^{3mAssess®}$. If you have any trouble to connect to the database server or to run the MySQL service (you can verify the state of the MySQL service in MySQL Administrator), please see the MySQL documentation or ask to a technician to help you

4. When installations of the server and its tools are terminated, click on MySQL Query Browser from the MySQL directory located in the start menu of Windows. Enter "log790" in the Default Schema box

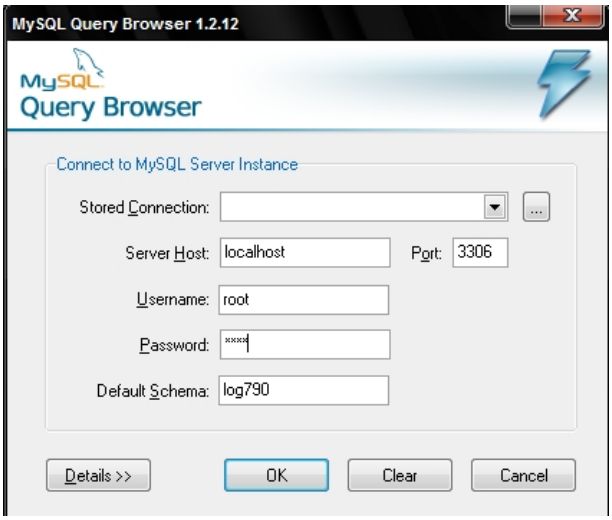


FIG. 6 – Login to MySQL Query Browser

5. In MySQL Query Browser, select File/Open Script ... and then select the sql file "smmm.sql" from the sql directory of the $S^{3mAssess®}$ archive you have extracted in the first step

6. Click on Execute

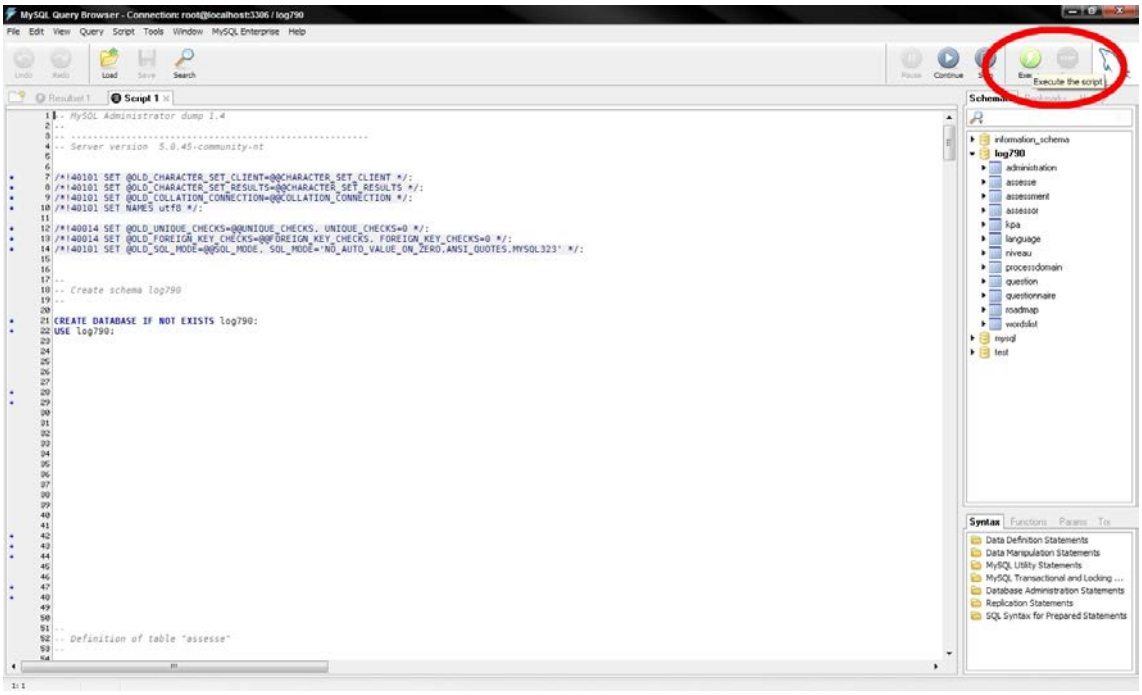


FIG. 7 – Execute "smmm.sql" in MySQL Query Browser

7. Now, the database is ready to operate and we can execute the assessment tool $S^{3mAssess}$®. To execute the tool, double-click on "S3m Assess.html" from the bin directory. In the starting of the application, an identification window is showed. Enter «guest» as Username and «guest» as Password

FIG. 8 – Identification window

3 Carry out an assessment

After the installation and the configuration steps are achieved, the user can create a new assessment or modify an old one in selecting the menu "Evaluation / Assessment". A first screen appears asking to choose to complete a previous assessment or not (see figure 9).

1. Click on "Evaluation / Assessment" from the main window
2. Click "Yes" if you want to complete a previous assessment, "No" if not

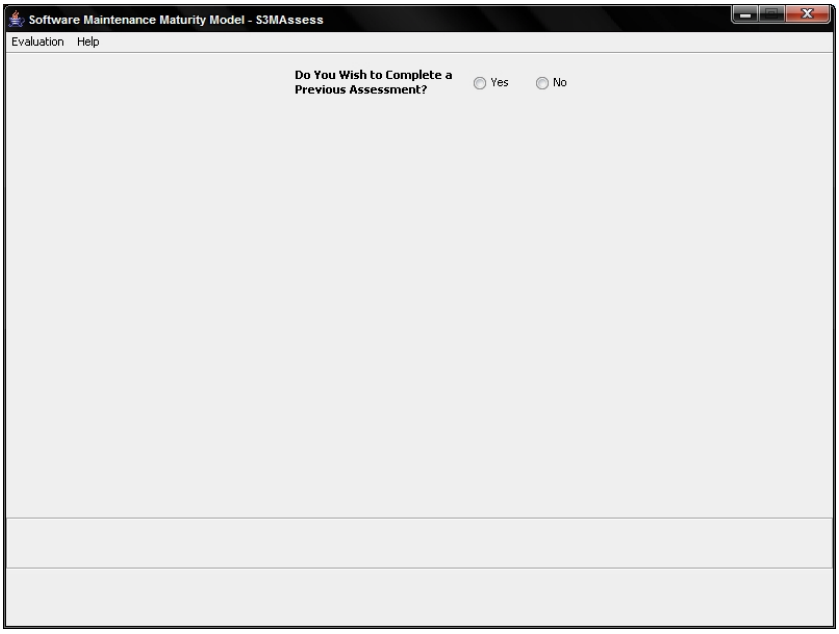


FIG. 9 – Choosing an assessment type

– Enter the information about the new evaluation if you clicked "No" while the previous step. See section 5 to have the definitions of information asked.

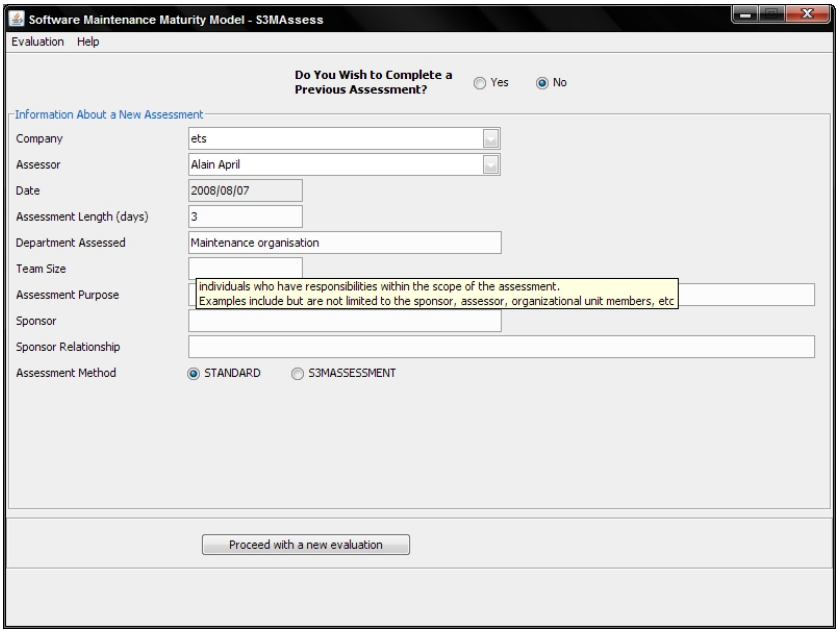


FIG. 10 – Input data's about the new assessment

Note that you have two assessment methods possible :

- The Standard method is used to assess completely the Key Process Area chosen. Because of its large liberty, that method would be recommended for advanced users ;
- The S^{3m}*Assessment* method will stop the assessment of a Key Process Area (KPA) if the mean level is under 80%. Advisable for users who do not know all particularities of the S^{3m}® or the mechanisms of an improvement program.

– Select the assessment to complete if you clicked "Yes" while the previous step

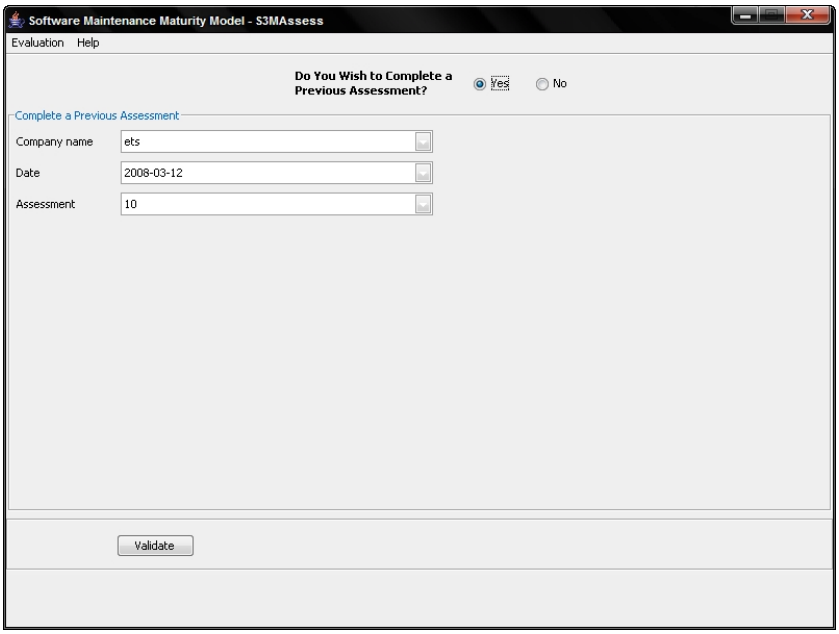


FIG. 11 – Complete a previous assessment

3. Choose one of the four domains available in the S^{3m}® and a Key Process Area (KPA) that relates to. You can also click on the Back button to select another old assessment to complete, to modify

the values about the current evaluation you are carrying out or to create a new one.

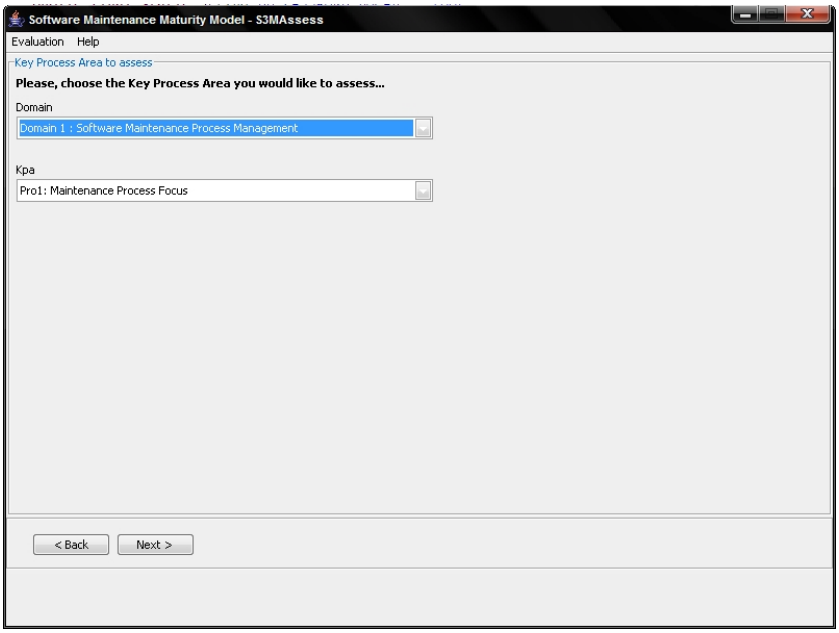


FIG. 12 – To choose the domain and the KPA to assess

4. Assess practices relative to the domain and to the KPA selected in the previous step. In the assessment process, practices are following from the level zero to the level two of the $S^{3m®}$ model.

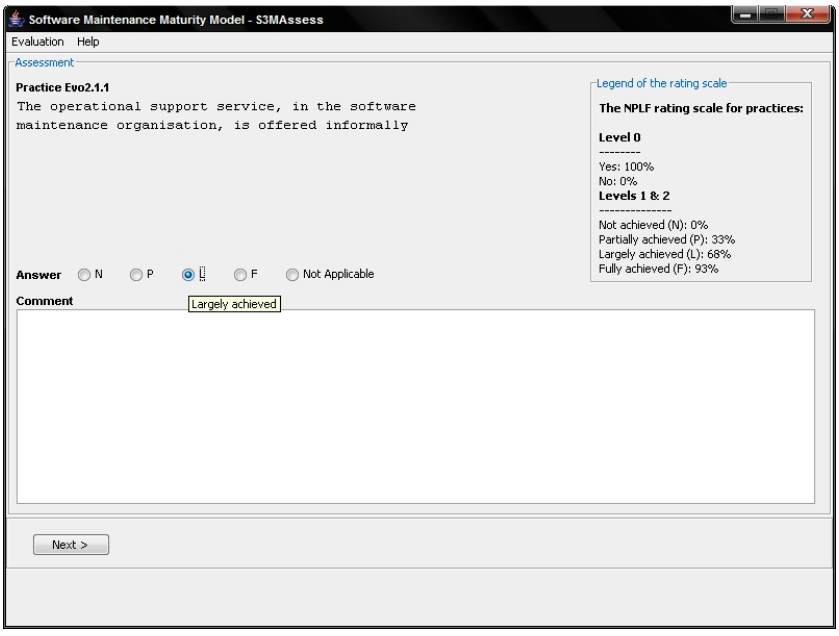


FIG. 13 – Example of a $S^{3m®}$ practice

4 To analyse an assessment

When questionnaires are filled in, the user can obtain a maturity profile. To do that, he selects the menu "Evaluation / Representation". Then, the application offers three lists : companies' names, dates and assessments. The user selects one company, one date and one assessment number that he has previously realized.

1. Click on "Evaluation / Representation" from the main window

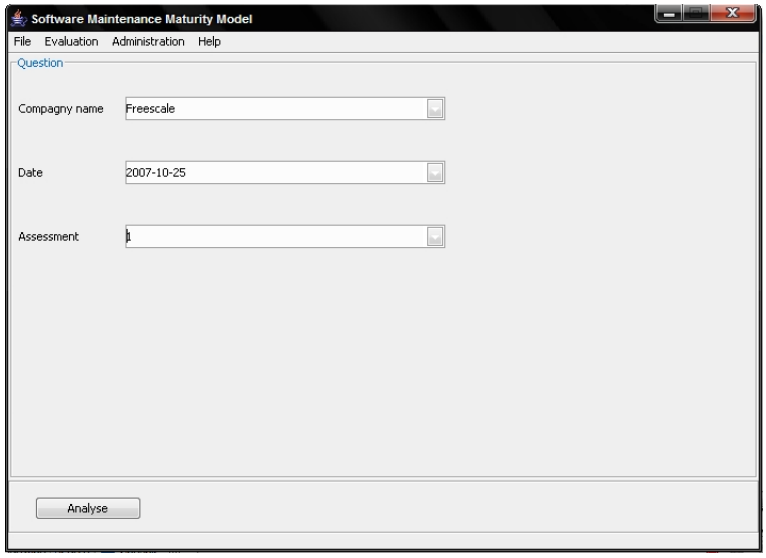


FIG. 14 – Selecting of a company, a date and an assessment number

2. After the user has chosen its assessment, he can click on "Analyse" to get many different graphical representations according to the **NPLF** (for **N**ot/**P**artially/**L**argely/**F**ully achieved) rating of the $S^{3m®}$ model. The succession of representations suggests an improvement policy as illustrated below :
   (a) The four $S^{3m®}$ domains are compared according to the three maturity levels. The percentages reached by each domain level are computed following the sum of kpa assessed means divided by the number of kpa's assessed for that domain in this level :

   $$domainPercentage(d,l) = \frac{\sum_k kpaAssessedMean_{domain_d,level_l}}{\#kpaAssessed_{domain_d,level_l}} ,$$
   with $1 \leq d \leq 4$, $0 \leq l \leq 2$ and $1 \leq k \leq 18$.

   and where
   $$kpaAssessedMean_{domain_d,level_l} = \frac{\sum valueOfPracticeAssessedAsPraticable_{kpa_k,level_l}}{\#practiceAssessedAsPraticable_{kpa_k,level_l}} ,$$
   with $0 \leq l \leq 2$, $1 \leq k \leq 18$.

   As we can see, the practices assessed as "Not applicable" are not considered while the means calculus. The figure 15 shows an example of the *Support to Software Evolution Engineering* domain getting a mean of 63% for its kpa's from the level one.

   An interesting policy linked to a continuous representation would be first to get a good percentage (80% for example) for a level before improving the upper level. With that output, the user can see quickly which domains should be first improved and at which level. Note that the 80% threshold for domains levels cited here is an example of recommendation and is not the 80% threshold for KPA's level from the S^{3m}*Assessment* method. This is two distinct means.

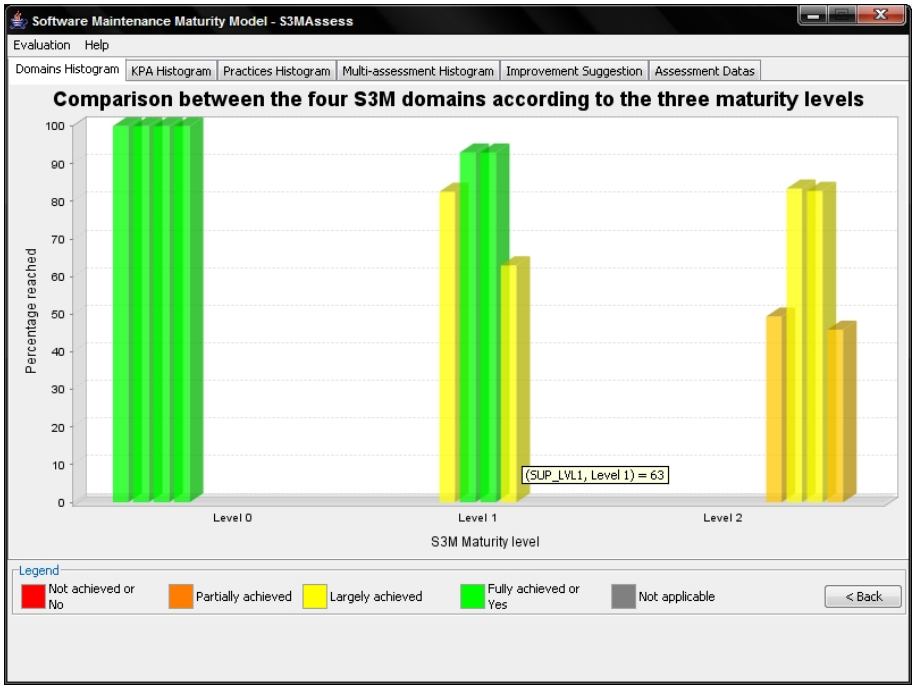


FIG. 15 – Comparison between the four $S^{3m®}$ domains according to the three $S^{3m®}$ maturity levels

(b) When the weakest domains have been determinated while the previous step, the user can see here because of which KPAs the domain is weak. For that, he selects a domain and a level for that domain. Finally, he clicks on "Compute" to see the result.

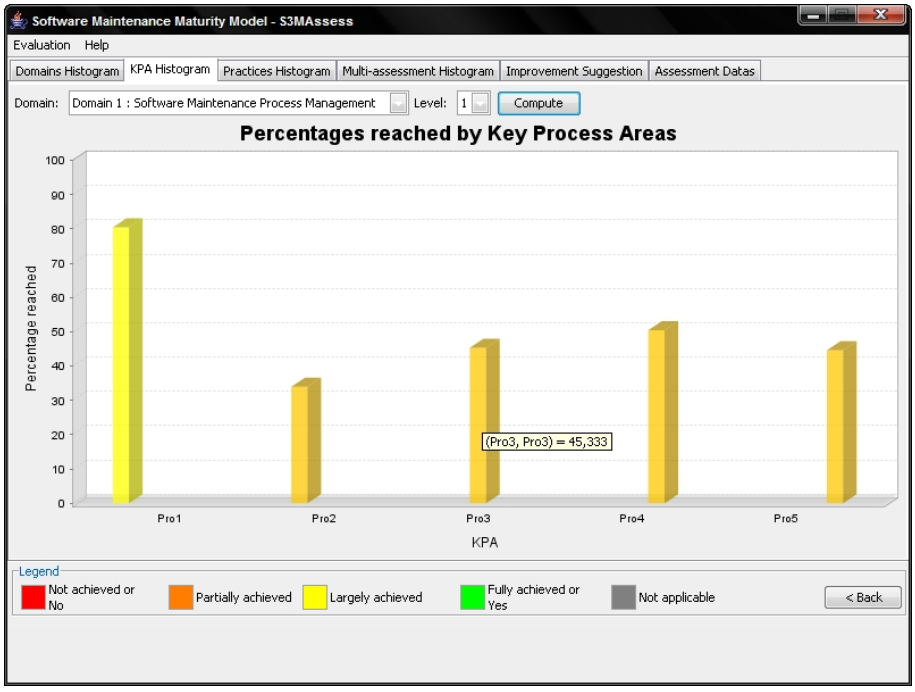


FIG. 16 – Percentages reached by KPAs

(c) In selecting the "Practices Histogram", the user can choose a domain to compute in order to get the details of which practices are the cause of the weakness of a domain.

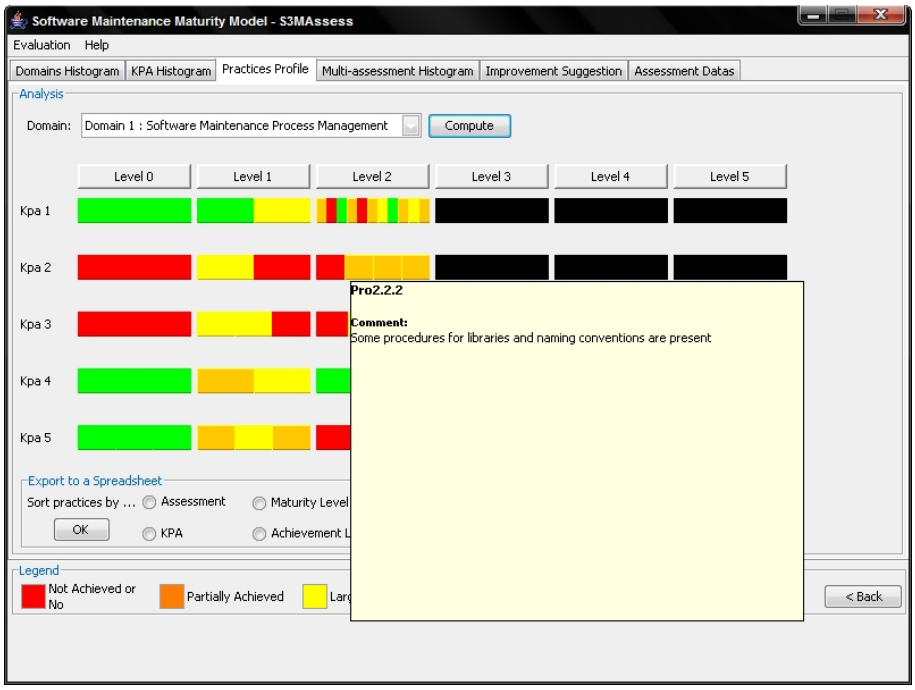


FIG. 17 – Representation of the maturity profile : Practices Profile

- It is also possible to export all assessment information and results in a spreadsheet. To do that, the user first selects a sort type for practices and then he clicks on the "OK" button in the "Export to a spreadsheet" frame (see fig. 17). Secondly, he can choose a location on his hard disk to save its assessment data's.

- Here is the scale of conversion of answers given to each practice :
  - Answers of the level 0 :
  - Yes : 100%
  - No : 0%
  - Answers of the levels 1 and 2 :
  - Not achieved : 0%
  - Partially achieved : 33%
  - Largely achieved : 68%
  - Fully achieved : 93%

- The user also has the possibility to see the practice identifier and its assessment comment when he lets the mouse pointer on a particular location of the graphic.

(d) The Multi-assessment Histogram tab allows the user to see what is its progression since the last assessment he carried out for the domain, the KPA and the level he has chosen to compute. Indeed, $S^{3mAssess}$® manages an historic of all assessments carried out. Thanks to that output, it is easy to see what is the progression and the consequences of resources invested to realize the improvement since the last time.

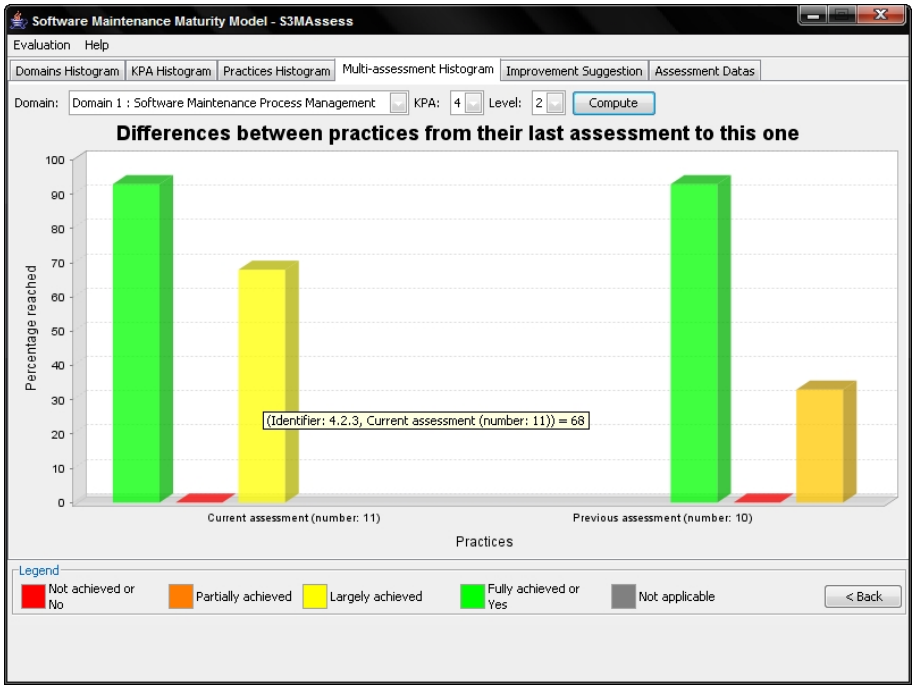


FIG. 18 – Multi-assessment Histogram

(e) The *Improvement Suggestion* tab suggests some practices that should be next improved. The goal here is to show which practices represent a weakness according to the percentage of its domain level.

The terminology given to domains' percentages follows that scale :
– "Not Achieved" if the percentage is included in [0 ;15]
– "Partially Achieved" if the percentage is included in [16 ;50]
– "Largely Achieved" if the percentage is included in [51 ;85]
– "Fully Achieved" if the percentage is included in [86 ;100]

The objective is to get uniform levels for the four $S^{3m®}$ domains without having high rated practices with very bad other practices in a same domain. Weakest practices should be improved to avoid future strong instabilities in the maintenance process of the organisation.
In order to respect the continuous improvement requirement, the user should first improve practices from lower levels until he reaches a good domain's level percentage (i.e. 80%).
Be careful, only practices assessed are presented in the output. If the user had chosen the S^{3m}*Assessment* method to carry out the assessment, practices of higher levels that could not be assessed because of a level not achieving the 80% level requirement are not showed. That is why the user has to improve weak practices of lower maturity levels before improving the highest ones.

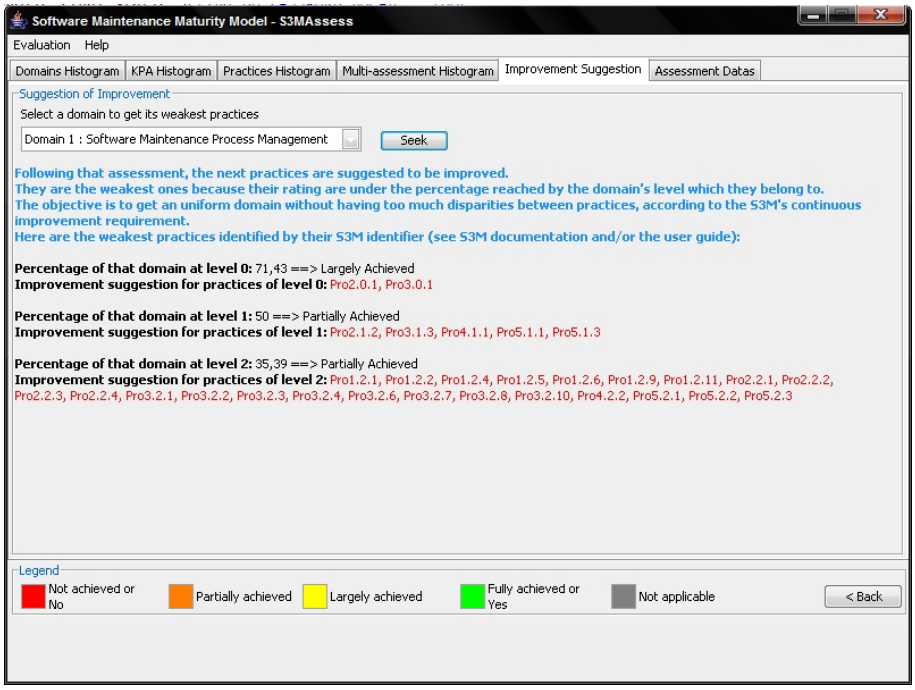


FIG. 19 – Improvement suggestion

So, that analysis extracts a subset of practices to facilitate the user in his improvement decision. In this way, subsets of weak practices coming from different KPA's are extracted to give the maximum support. With all subsets of weakest practices, a brainstorming to discuss possible improvements can begin with sound information.
Also, those subsets could be put as input in another support tool based on a Decision Support System. It already exists a tool for that purpose also based on the $S^{3m®}$ model : S3MExpert.

(f) In the *Assessment Data's'* tab, the user has an access to general assessment information he has entered in the beginning of the assessment process and to details of each practices (their $S^{3m®}$ identifier, their rating, their achievement percentage and their comment). The figure 20 shows an example of details and, in particular, an example of conversion of answer to percentage.

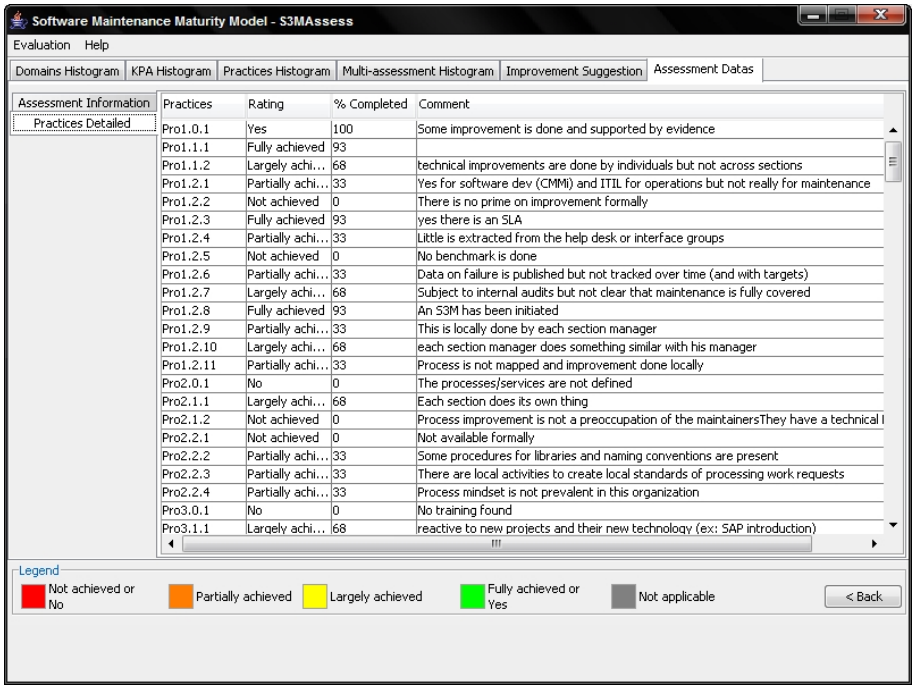


FIG. 20 – Assessment data's

5 Glossary

- **Assessment Length** : the duration in days of the assessment
- **Assessment Method** : the user has the choice of assessing all practices he wants without restriction (Standard method) or to follow the $S^{3m}Assessment$ method which prevents to assess practices of upper kpa's levels if a lower kpa's level does not attain the 80% limit
- **Assessment Purpose** : a statement which defines the reason for performing the assessment
- **Assessor** : the individual's name who will lead the assessment
- **Company** : the company's name to assess
- **Date** : the current date is automatically fulfilled
- **Department Assessed** : the department of maintenance to be assessed
- **Sponsor** : the individual, internal or external to the organization being assessed, who requires the assessment to be performed, and provides financial or other resources to carry it out
- **Sponsor Relationship** : details of the relationship between the sponsor and the assessment. For example, what are its provisions, financial, etc
- **Team Size** : individuals who have responsibilities within the scope of the assessment. Examples include but are not limited to the sponsor, assessor, organizational unit members, etc.

2. Modèle d'Analyse d'Impact

Impact Analysis Checklist for Requirements Changes

Implications of the Proposed Change

- ❑ Identify any existing requirements in the baseline that conflict with the proposed change.
- ❑ Identify any other pending requirement changes that conflict with the proposed change.
- ❑ What are the consequences of not making the change?
- ❑ What are possible adverse side effects or other risks of making the proposed change?
- ❑ Will the proposed change adversely affect performance requirements or other quality attributes?
- ❑ Will the change affect any system component that affects critical properties such as safety and security, or involve a product change that triggers recertification of any kind?
- ❑ Is the proposed change feasible within known technical constraints and current staff skills?
- ❑ Will the proposed change place unacceptable demands on any computer resources required for the development, test, or operating environments?
- ❑ Must any tools be acquired to implement and test the change?
- ❑ How will the proposed change affect the sequence, dependencies, effort, or duration of any tasks currently in the project plan?
- ❑ Will prototyping or other user input be required to verify the proposed change?
- ❑ How much effort that has already been invested in the project will be lost if this change is accepted?
- ❑ Will the proposed change cause an increase in product unit cost, such as by increasing third-party product licensing fees?
- ❑ Will the change affect any marketing, manufacturing, training, or customer support plans?

System Elements Affected by the Proposed Change

- ❑ Identify any user interface changes, additions, or deletions required.
- ❑ Identify any changes, additions, or deletions required in reports, databases, or data files.
- ❑ Identify the design components that must be created, modified, or deleted.
- ❑ Identify hardware components that must be added, altered, or deleted.
- ❑ Identify the source code files that must be created, modified, or deleted.
- ❑ Identify any changes required in build files.
- ❑ Identify existing unit, integration, system, and acceptance test cases that must be modified or deleted.
- ❑ Estimate the number of new unit, integration, system, and acceptance test cases that will be required.
- ❑ Identify any help screens, user manuals, training materials, or other documentation that must be created or modified.
- ❑ Identify any other systems, applications, libraries, or hardware components affected by the change.
- ❑ Identify any third party software that must be purchased.
- ❑ Identify any impact the proposed change will have on the project's software project management plan, software quality assurance plan, software configuration management plan, or other plans.
- ❑ Quantify any effects the proposed change will have on budgets of scarce resources, such as memory, processing power, network bandwidth, real-time schedule.
- ❑ Identify any impact the proposed change will have on fielded systems if the affected component is not perfectly backward compatible.

Effort Estimation for a Requirements Change

Effort (Labor Hours)	Task
_____	Update the SRS or requirements database with the new requirement
_____	Develop and evaluate prototype
_____	Create new design components
_____	Modify existing design components
_____	Develop new user interface components
_____	Modify existing user interface components
_____	Develop new user publications and help screens
_____	Modify existing user publications and help screens
_____	Develop new source code
_____	Modify existing source code
_____	Purchase and integrate third party software
_____	Identify, purchase, and integrate hardware components; qualify vendor
_____	Modify build files
_____	Develop new unit and integration tests
_____	Modify existing unit and integration tests
_____	Perform unit and integration testing after implementation
_____	Write new system and acceptance test cases
_____	Modify existing system and acceptance test cases
_____	Modify automated test drivers
_____	Perform regression testing at unit, integration, and system levels
_____	Develop new reports
_____	Modify existing reports
_____	Develop new database elements
_____	Modify existing database elements
_____	Develop new data files
_____	Modify existing data files
_____	Modify various project plans
_____	Update other documentation
_____	Update requirements traceability matrix
_____	Review modified work products
_____	Perform rework following reviews and testing
_____	Recertify product as being safe, secure, and compliant with standards.
_____	Other additional tasks
_____	**TOTAL ESTIMATED EFFORT**

Procedure:

1. Identify the subset of the above tasks that will have to be done.
2. Allocate resources to tasks.
3. Estimate effort required for pertinent tasks listed above, based on assigned resources.
4. Total the effort estimates.
5. Sequence tasks and identify predecessors.
6. Determine whether change is on the project's critical path.
7. Estimate schedule and cost impact.

Impact Analysis Report Template

Change Request ID: _____
Title: _____
Description: _____
_____
Analyst: _____
Date Prepared: _____

Prioritization Estimates:
- Relative Benefit: _____ (1-9)
- Relative Penalty: _____ (1-9)
- Relative Cost: _____ (1-9)
- Relative Risk: _____ (1-9)
- Calculated Priority: _____(relative to other pending requirements)

Estimated total effort: _____ labor hours
Estimated lost effort: _____ labor hours (from discarded work)
Estimated schedule impact: _____ days
Additional cost impact: _____ dollars
Quality impact: _____
_____

Other requirements affected: _____
_____
Other tasks affected: _____
_____
Integration issues: _____
Life cycle cost issues: _____
Other components to examine for possible changes: _____
_____

3. Analyses d'impacts du logiciel $S^{3mAssess_{®}}$

S3MAssess
Analyses d'impacts

Table des matières

Analyses d'impacts

1. Introduction

Ce document reprend les analyses d'impacts des principales requêtes de maintenance effectuées dans les Révisions 1,2 et 3 du logiciel S3MAssess. Celles-ci sont incrémentales et se basent sur la version d'acquisition de l'outil, c'est à dire la v0.02.

2. Analyse d'impacts de la Révision 1

2.1 Sommaire

Cette section comporte un tableau contenant un sommaire des cinq requêtes de maintenance les plus prioritaires.

L'outil n'était pas en opération lors de son acquisition. La réalisation de cette première étape vise à remettre sur pied l'application selon ses exigences initiales. Elle donnera principalement lieu à de la maintenance corrective dans un premier temps et à de la maintenance préventive dans un second temps.

Chaque requête de maintenance a été classifiée selon l'entente de service en vigueur. Le tableau ci-dessous présente un sommaire des requêtes avec leur numéro de référence, leur description et la catégorie de maintenance à laquelle elles se rapportent.

Numéro de référence	Description	Catégorie de maintenance
0000001	Succession des pratiques dans le mauvais sens	Corrective
0000002	Encodage des pratiques dans la base de données	Corrective
0000003	Correction des données incohérentes et ajout de contraintes d'intégrités dans la base de données	Préventive
0000004	Résultats d'évaluation invalides, partiellement inaccessibles et mal affichés	Corrective
0000005	Non mise à jour des dernières évaluations effectuées	Corrective

2.2 Recommandation

La présente section propose une priorisation des requêtes afin d'éviter des conflits lors dans l'implantation des diverses solutions.

2.2.1 Priorisation des problèmes

Afin d'éviter des conflits dans l'ordre d'implantation des solutions aux requêtes, l'ordre du tableau ci-dessous est suggéré.

Numéro de référence	Description	Priorité
0000005	Non mise à jour des dernières évaluations effectuées	1
0000004	Résultats d'évaluation invalides, partiellement inaccessibles et mal affichés	2
0000001	Succession des pratiques dans le mauvais sens	3
0000003	Correction des données incohérentes et ajout de contraintes d'intégrités dans la base de données	4
0000002	Encodage des pratiques dans la base de données	5

2.3 Livraison

Suite à l'analyse d'effort de la Révision 1, à l'entente convenue avec le maître de stage, Dr Alain April, et avec les clients, l'équipe de maintenance livrera la première révision à la mi-novembre (travail débutant le 21 octobre). Le tableau ci-dessous reprend une estimation de l'effort nécessaire pour la réalisation des cinq requêtes prioritaires.

Numéro de référence	Description	Effort (heures)
0000005	Non mise à jour des dernières évaluations effectuées	6
0000004	Résultats d'évaluation invalides, partiellement inaccessibles et mal affichés	18
0000001	Succession des pratiques dans le mauvais sens	13
0000003	Correction des données incohérentes et ajout de contraintes d'intégrités dans la base de données	7
0000002	Encodage des pratiques dans la base de données	18
	Total :	**62 heures**

2.4 Analyses d'impacts détaillées

2.4.1 0000005 - Non mise à jour des dernières évaluations effectuées

2.4.1.1 Implications du changement proposé

2.4.1.1.1 Conflit avec les spécifications actuelles

Aucune spécification n'entre en conflit avec le changement proposé.

2.4.1.1.2 Conflit avec les autres demandes de changement

Aucun conflit avec d'autres requêtes.

2.4.1.1.3 Conséquence de ne pas mettre en œuvre le changement

L'utilisateur ne sera pas en mesure d'analyser les dernières évaluations qu'il a faites. La perception d'utilité de l'outil pour celui-ci en serait par conséquent très réduite.

2.4.1.1.4 Risques d'effectuer le changement proposé et effets de bord

Assez peu de risque si toutefois la correction est testée. Sinon, l'utilisateur risque de ne plus pouvoir analyser aucune évaluation.

2.4.1.1.5 Impact sur les attributs de qualité et performance

Pas de changement sur la performance du logiciel mais impact positif sur sa qualité puisque si le

problème est résolu, son utilisabilité en est augmentée.

2.4.1.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun changement à ce niveau.

2.4.1.1.7 Compétence et Faisabilité technique

Aucun problème envisagé avec la compétence de l'équipe actuelle, la modification est tout à fait réalisable sans aucune nécessité de formation supplémentaire.

2.4.1.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

2.4.1.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun outil supplémentaire mais il serait envisageable d'implanter des tests unitaires afin d'augmenter la stabilité du logiciel S3MAssess.

2.4.1.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

2.4.1.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client.

2.4.1.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

2.4.1.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

2.4.1.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est corrective et n'ajoute aucune nouvelle fonctionnalité.

2.4.1.2 Éléments du système affectés par le changement proposé

2.4.1.2.1 Changements requis pour l'interface utilisateur

Aucun.

2.4.1.2.2 Changements requis aux rapports, bases de données ou fichiers de données

- Rectification des valeurs dans la base de données.

2.4.1.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Synchronisation d'une colonne entre deux tables de la base de données.

2.4.1.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

2.4.1.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Synchronisation des colonnes « assessmentId » entre les tables 'assessment' et 'questionnaire' de la base de données.
- La classe PanelQuestionnaire devra être modifiée pour mettre à jour les évaluations disponibles de l'objet PanelSettingAnalyze.
- Ajout d'un appel « refresh() » dans la méthode quit() de la classe PanelQuestionnaire.
- Création d'une méthode getInstance() dans la classe PanelSettingAnalyse.
- Ajout de l'attribut 'companySelected' dans la classe PanelAssessment.
- Modification de la méthode setInformation() dans PanelQuestionnaire.
- Ajout de l'appel 'refreshDate()' dans la méthode quit() de la classe PanelQuestionnaire.

2.4.1.2.6 Changements requis aux scripts de compilation

Aucun changement.

2.4.1.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

2.4.1.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

2.4.1.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Aucun changement.

2.4.1.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

2.4.1.2.11 Composant logiciel externe qui doit être acheté

Aucun.

2.4.1.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Comme expliqué ci-dessus, cette révision vise uniquement à corriger tous les problèmes. Le plan de projet initial doit prendre en compte l'intégralité de cette révision dans le sens où le commencement de l'implantation des nouvelles fonctionnalités sera décalé.

2.4.1.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet, pas de changement.

2.4.1.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

- Il va falloir migrer les changements de la base de données de maintenance vers la BD de production.

2.4.1.3 Estimation de l'effort pour 0000005 - Non mise à jour des dernières évaluations effectuées

Tâche	**Effort (heure)**
Analyse du problème	1
Modification du code source et de la base de données	4
Test informel	1
Total :	**6**

2.4.2 0000004 - Résultats d'évaluation invalides, partiellement inaccessibles et mal affichés

2.4.2.1 Implications du changement proposé

2.4.2.1.1 Conflit avec les spécifications actuelles

Le changement proposé permet de rencontrer les exigences initiales qui n'étaient pas respectées.

2.4.2.1.2 Conflit avec les autres demandes de changement

Aucun conflit avec d'autres requêtes.

2.4.2.1.3 Conséquence de ne pas mettre en œuvre le changement

Résultats totalement incohérents et sans aucune valeur d'analyse pour l'utilisateur.

2.4.2.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque de ne plus pouvoir afficher les résultats d'évaluation si la modification n'est pas opérée avec précaution et suivie de tests.

2.4.2.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent cohérents et l'utilité de l'application n'en est qu'augmentée. Pas d'impact sur les performances.

2.4.2.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun changement à ce niveau.

2.4.2.1.7 Compétence et Faisabilité technique

Formation nécessaire sur les API Java traitant des composants graphiques.

2.4.2.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement (Eclipse) peut être étendue par le plugin Jigloo GUI Editor.

2.4.2.1.9 Outils additionnels requis pour implémenter et tester le changement

Le plugin Jigloo GUI Editor peut être utilisé pour afficher et donc tester les changements graphiques. Il serait également envisageable d'implanter des tests unitaires afin d'augmenter la stabilité du logiciel S3MAssess.

2.4.2.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

2.4.2.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client.

2.4.2.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

2.4.2.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

2.4.2.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est corrective et n'ajoute aucune nouvelle fonctionnalité.

2.4.2.2 Éléments du système affectés par le changement proposé

2.4.2.2.1 Changements requis pour l'interface utilisateur

- Correction des couleurs selon le résultat.
- Remplissage des rectangles résultats dans l'interface Représentation. Il y a des blocs noirs dans les rectangles.

2.4.2.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Ajout d'une clé étrangère.

2.4.2.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Requêtes de BD corrigées.
- Contrainte d'intégrité.

2.4.2.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

2.4.2.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Correction de la requête BD dans la classe DbQueriesAnalyse : la méthode getAnswersList().
- Inversion dans la classe DBConstants de RATING_YES et RATING_NO: 1 et 0.
- Modification de la méthode showResults() dans la classe PanelResultAnalyse.
- Ajout d'une clé étrangère de la table 'question' vers la table 'kpa' pour permettre la jointure et d'avoir le 'kpaID'.

2.4.2.2.6 Changements requis aux scripts de compilation

Aucun changement.

2.4.2.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

2.4.2.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

2.4.2.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Aucun changement.

2.4.2.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

2.4.2.2.11 Composant logiciel externe qui doit être acheté

Aucun.

2.4.2.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Voir 2.4.1.2.12

2.4.2.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet, pas de changement.

2.4.2.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

- Il va falloir migrer les changements de la base de données de maintenance vers la BD de production.

2.4.2.3 Estimation de l'effort pour 0000004 - Résultats d'évaluation invalides, partiellement inaccessibles et mal affichés

Tâche	Effort (heure)
Analyse du problème	1
Modification du code source et de la base de données	15
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**18**

2.4.3 0000001 - Succession des pratiques dans le mauvais sens

2.4.3.1 Implications du changement proposé

2.4.3.1.1 Conflit avec les spécifications actuelles

Le changement proposé permet de rencontrer les exigences initiales qui n'étaient pas respectées.

2.4.3.1.2 Conflit avec les autres demandes de changement

Aucun conflit avec d'autres requêtes.

2.4.3.1.3 Conséquence de ne pas mettre en œuvre le changement

Difficulté accrue pour l'utilisateur d'évaluer un processus. Selon le modèle S3M, il vaut mieux évaluer à partir des premières questions d'ordre général et monter peu à peu dans les niveaux de maturité. De cette manière, l'évaluation sera plus pertinente et plus fiable.

2.4.3.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque de ne plus pouvoir effectuer d'évaluation si la modification n'est pas opérée avec précaution et suivie de tests.

2.4.3.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent plus cohérents et fiables. Pas d'impact sur les performances.

2.4.3.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun changement à ce niveau.

2.4.3.1.7 Compétence et Faisabilité technique

Aucun problème.

2.4.3.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

2.4.3.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun outil supplémentaire mais il serait envisageable d'implanter des tests unitaires afin d'augmenter la stabilité du logiciel S3MAssess.

2.4.3.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

2.4.3.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client.

2.4.3.1.12 Effort perdu si le changement est mis en œuvre

Aucun car le développement initial n'a pas été effectué par l'équipe de maintenance.

2.4.3.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

2.4.3.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est corrective (correction d'un défaut) et n'ajoute aucune nouvelle fonctionnalité.

2.4.3.2 Éléments du système affectés par le changement proposé

2.4.3.2.1 Changements requis pour l'interface utilisateur

Aucun.

2.4.3.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Aucun.

2.4.3.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

Requêtes de BD corrigées.

2.4.3.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

2.4.3.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Création de la méthode lastLevelQuestion(DataObject question) dans la classe DBTableQuestion.
- Réécriture de la méthode nextQuestion()

2.4.3.2.6 Changements requis aux scripts de compilation

Aucun changement.

2.4.3.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

2.4.3.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

2.4.3.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Modification du guide utilisateur.

2.4.3.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

2.4.3.2.11 Composant logiciel externe qui doit être acheté

Aucun.

2.4.3.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Voir 2.4.1.2.12

2.4.3.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet, pas de changement.

2.4.3.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun.

2.4.3.3 Estimation de l'effort pour 0000001 - Succession des pratiques dans le mauvais sens

Tâche	Effort (heure)
Analyse du problème	1
Modification du code source et de la base de données	10
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**13**

2.4.4 0000003 - Correction des données incohérentes et ajout de contraintes d'intégrités dans la base de données

2.4.4.1 Implications du changement proposé

2.4.4.1.1 Conflit avec les spécifications actuelles

Aucun.

2.4.4.1.2 Conflit avec les autres demandes de changement

Requête à réaliser avant la requête 0000002 pour éviter un mauvais encodage.

2.4.4.1.3 Conséquence de ne pas mettre en œuvre le changement

Impossibilité d'effectuer et d'analyser les évaluations de manière correcte par incohérence grandissante des données.

2.4.4.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque que la couche applicative présente des défaillances si le code est basé sur des données incohérentes (en cas de « hardcodage »).

2.4.4.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent cohérents et l'utilité de l'application n'en est qu'augmentée. Augmentation des performances grâce à l'utilisation d'index.

2.4.4.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Impact positif sur la fiabilité de l'application puisque prévention d'une incohérence des données.

2.4.4.1.7 Compétence et Faisabilité technique

Compétence acquise.

2.4.4.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête grâce notamment aux outils d'administration de MySQL.

2.4.4.1.9 Outils additionnels requis pour implémenter et tester le changement

Outils d'administration de MySQL (déjà installés).

2.4.4.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

2.4.4.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client.

2.4.4.1.12 Effort perdu si le changement est mis en œuvre

Aucun si cette requête est effectuée avant la requête 0000002.

2.4.4.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

2.4.4.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est corrective et préventive et n'ajoute aucune nouvelle fonctionnalité.

2.4.4.2 Éléments du système affectés par le changement proposé

2.4.4.2.1 Changements requis pour l'interface utilisateur

Aucun.

2.4.4.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Ajout de clés étrangères et vérification de la cohérence des données applicatives.

2.4.4.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

Les tables de la BD.

2.4.4.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Réingénierie de la BD à partir du serveur de BD.

2.4.4.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Fichier source SQL de la BD.

2.4.4.2.6 Changements requis aux scripts de compilation

Le fichier source SQL peut être considéré comme un script.

2.4.4.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Tester les fonctionnalités du logiciel utilisant la BD.

2.4.4.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

2.4.4.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Les schémas de BD.

2.4.4.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

La BD implantée sur le serveur de BD.

2.4.4.2.11 Composant logiciel externe qui doit être acheté

Aucun.

2.4.4.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Voir 2.4.1.2.12

2.4.4.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet, pas de changement.

2.4.4.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Il va falloir migrer les changements de la base de données de maintenance vers la BD de production.

2.4.4.3 Estimation de l'effort pour 0000003 - Correction des données incohérentes et ajout de contraintes d'intégrités dans la base de données

Tâche	**Effort (heure)**
Analyse du problème	2
Modification du code source et de la base de données	3
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**7**

2.4.5 0000002 - Encodage des pratiques dans la base de données

2.4.5.1 Implications du changement proposé

2.4.5.1.1 Conflit avec les spécifications actuelles

Le changement proposé permet de rencontrer les exigences initiales qui n'étaient pas respectées. Il est impossible d'effectuer une évaluation puisque seules 10 pratiques sont actuellement encodées.

2.4.5.1.2 Conflit avec les autres demandes de changement

Requête à réaliser après 0000003 pour éviter de rectifier les valeurs encodées.

2.4.5.1.3 Conséquence de ne pas mettre en œuvre le changement

Impossibilité d'effectuer une évaluation.

2.4.5.1.4 Risques d'effectuer le changement proposé et effets de bord

Aucun.

2.4.5.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque des évaluations complètes seront désormais possibles. Pas d'impact sur les performances.

2.4.5.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun changement à ce niveau.

2.4.5.1.7 Compétence et Faisabilité technique

Aucun problème, le texte des pratiques est fourni et les connaissances du langage SQL sont amplement suffisantes.

2.4.5.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

2.4.5.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun outil supplémentaire si ce n'est « MySQL Query Browser », déjà installé dès le début de la phase de maintenance de S3MAssess.

2.4.5.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

2.4.5.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client. Toutefois, interaction nécessaire avec le maître de stage, Dr Alain April, afin d'obtenir le texte des pratiques en anglais.

2.4.5.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

2.4.5.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

2.4.5.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est corrective (l'absence des pratiques est considérée comme un défaut) et n'ajoute aucune nouvelle fonctionnalité.

2.4.5.2 Éléments du système affectés par le changement proposé

2.4.5.2.1 Changements requis pour l'interface utilisateur

Aucun.

2.4.5.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Remplissage de la table 'question'.

2.4.5.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

Aucun.

2.4.5.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

2.4.5.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Le fichier SQL de génération de la base de données.

2.4.5.2.6 Changements requis aux scripts de compilation

Le fichier source SQL peut être considéré comme un script.

2.4.5.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Tests informels afin de s'assurer que les valeurs des colonnes encodées sont correctes et permettent une bonne succession d'affichage des pratiques.

2.4.5.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun.

2.4.5.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Aucun changement.

2.4.5.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

2.4.5.2.11 Composant logiciel externe qui doit être acheté

Aucun sauf peut-être MySQL à terme si S3MAssess devient industriel.

2.4.5.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Voir 2.4.1.2.12

2.4.5.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

L'encodage des pratiques prend très légèrement plus de place sur le serveur de BD.

2.4.5.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Il va falloir migrer les changements de la base de données de maintenance vers la BD de production.

2.4.5.3 Estimation de l'effort pour 0000002 – Encodage des pratiques dans la base de données

Tâche	**Effort (heure)**
Analyse du problème	1
Modification du code source et de la base de données	15
Test informel	2
Total :	**18**

3. Analyse d'impacts de la Révision 2

3.1 Sommaire

Cette section comporte un tableau contenant un sommaire des cinq requêtes de maintenance les plus prioritaires.

L'outil n'étant pas encore en opération suite à la révision une, la décision a été prise de développer les nouvelles fonctionnalités clientes avant la mise en opération. Cette deuxième révision a donc pour but d'implémenter ces nouvelles fonctionnalités et de corriger les problèmes de priorité inférieure à ceux déjà traités dans la révision 1. Elle donnera principalement lieu à de la maintenance perfective.

Chaque requête de maintenance a été classifiée selon l'entente de service en vigueur. Le tableau ci-dessous présente un sommaire des requêtes avec leur numéro de référence, leur description et la catégorie de maintenance à laquelle elles se rapportent.

Numéro de référence	Description	Catégorie de maintenance
0000006	Nouvelle fonctionnalité : exporter toutes les informations d'évaluation dans une feuille de calculs	Perfective
0000007	Nouvelle fonctionnalité : affichage de l'identifiant et du commentaire de chaque pratique dans le graphique résultat	Perfective
0000008	Nouvelle fonctionnalité : possibilité de trier les résultats dans la feuille de calcul	Perfective
0000009	Nouvelle fonctionnalité : ajout d'onglets dans la partie analyse permettant d'obtenir toutes les informations d'évaluation et les suggestions d'amélioration	Perfective
0000010	Confidentialité des informations entre les utilisateurs de S3MAssess depuis que l'application tourne sur un serveur web	Adaptative

3.2 Recommandation

La présente section propose une priorisation des requêtes afin d'éviter des conflits lors de l'implantation des diverses solutions.

3.2.1 Priorisation des problèmes

Afin d'éviter des conflits dans l'ordre d'implantation des solutions aux requêtes, l'ordre du tableau ci-dessous est suggéré.

Numéro de référence	Description	Priorité
0000010	Confidentialité des informations entre les utilisateurs de S3MAssess depuis que l'application tourne sur un serveur web	1
0000006	Nouvelle fonctionnalité : exporter toutes les informations d'évaluation dans une feuille de calculs	2
0000008	Nouvelle fonctionnalité : possibilité de trier les résultats dans la feuille de calculs	3
0000007	Nouvelle fonctionnalité : affichage de l'identifiant et du commentaire de chaque pratique dans le graphique résultat	4
0000009	Nouvelle fonctionnalité : ajout d'onglets dans la partie analyse permettant d'obtenir toutes les informations d'évaluation et les suggestions d'amélioration	5

3.3 Livraison

Suite à l'analyse d'effort de la Révision 2, à l'entente convenue avec le maître de stage, Dr Alain April, et avec les clients, l'équipe de maintenance livrera la deuxième révision à la fin décembre (travail débutant à la mi-novembre). Le tableau ci-dessous reprend une estimation de l'effort nécessaire pour la réalisation des cinq requêtes prioritaires.

Numéro de référence	Description	Effort (heures)
0000010	Confidentialité des informations entre les utilisateurs de S3MAssess depuis que l'application tourne sur un serveur web	11
0000006	Nouvelle fonctionnalité : exporter toutes les informations d'évaluation dans une feuille de calculs	19
0000008	Nouvelle fonctionnalité : possibilité de trier les résultats dans la feuille de calculs	9
0000007	Nouvelle fonctionnalité : affichage de l'identifiant et du commentaire de chaque pratique dans le graphique résultat	11
0000009	Nouvelle fonctionnalité : ajout d'onglets dans la partie analyse permettant d'obtenir toutes les informations d'évaluation et les suggestions d'amélioration	49
	Total :	**99 heures**

3.4 Analyses d'impacts détaillées

3.4.1 0000010 - Confidentialité des informations entre les utilisateurs de S3MAssess depuis que l'application tourne sur un serveur web

3.4.1.1 Implications du changement proposé

3.4.1.1.1 Conflit avec les spécifications actuelles

Aucun conflit.

3.4.1.1.2 Conflit avec les autres demandes de changement

Aucun conflit avec d'autres requêtes.

3.4.1.1.3 Conséquence de ne pas mettre en œuvre le changement

Tout utilisateur peut accéder aux informations de tout autre utilisateur. Non confidentialité des données.

3.4.1.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque de mélange d'informations confidentielles entre les utilisateurs si la modification n'est pas opérée avec précaution et suivie de tests.

3.4.1.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les données deviennent confidentielles. Pas d'impact sur les performances.

3.4.1.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

La sécurité (confidentialité) augmente.

3.4.1.1.7 Compétence et Faisabilité technique

Aucune formation supplémentaire n'est nécessaire.

3.4.1.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

3.4.1.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun outil supplémentaire mais il serait envisageable d'implanter des tests unitaires afin d'augmenter la stabilité du logiciel S3MAssess. Utilisation des outils MySQL.

3.4.1.1.10 Impact sur le plan de projet

Cette requête de première priorité n'est apparue que sur le tard mais elle reste réalisable pour cette révision.

3.4.1.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Les exigences de cette requête sont claires, aucun besoin d'interaction supplémentaire avec le client.

3.4.1.1.12 Effort perdu si le changement est mis en œuvre

Aucun puisque l'équipe de maintenance n'est pas celle de développement.

3.4.1.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation.

3.4.1.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Aucun impact car cette requête est adaptative et n'ajoute aucune nouvelle fonctionnalité.

3.4.1.2 Éléments du système affectés par le changement proposé

3.4.1.2.1 Changements requis pour l'interface utilisateur

Aucun.

3.4.1.2.2 Changements requis aux rapports, bases de données ou fichiers de données

- Modification des tables 'assessor' et 'administration'.
- Modification des requêtes de BD basées maintenant sur un UserId.

3.4.1.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- BD adaptée
- Requêtes de BD adaptées.

3.4.1.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

3.4.1.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Modification des requêtes dans le package smmm.db.table
- Modification du fichier SQL de génération de la BD.

3.4.1.2.6 Changements requis aux scripts de compilation

Modification du fichier SQL de génération de la BD.

3.4.1.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

3.4.1.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

3.4.1.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Modification du schéma relationnel de BD.

3.4.1.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement si ce n'est l'implantation de la nouvelle BD sur le serveur de BD.

3.4.1.2.11 Composant logiciel externe qui doit être acheté

Aucun.

3.4.1.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Cette requête tardive entre dans ce document d'analyse d'impact.

3.4.1.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet, pas de changement.

3.4.1.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

- Il va falloir migrer les changements de la base de données de maintenance vers la BD de production.

3.4.1.3 0000010 - Confidentialité des informations entre les utilisateurs de S3MAssess depuis que l'application tourne sur un serveur web

Tâche	Effort (heure)
Analyse du problème	2
Modification du code source et de la base de données	7
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**11**

3.4.2 0000006 - Nouvelle fonctionnalité : exporter toutes les informations d'évaluation dans une feuille de calculs

3.4.2.1 Implications du changement proposé

3.4.2.1.1 Conflit avec les spécifications actuelles

Pas de conflit, il s'agit d'une nouvelle exigence fonctionnelle.

3.4.2.1.2 Conflit avec les autres demandes de changement

Requête à réaliser avant la requête 0000008.

3.4.2.1.3 Conséquence de ne pas mettre en œuvre le changement

Exigence cliente non implémentée.

3.4.2.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'une mauvaise sortie des résultats d'évaluation si la modification n'est pas opérée avec précaution et suivie de tests.

3.4.2.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent accessibles en dehors de l'application et l'utilité de l'application n'en est qu'augmentée. Pas d'impact sur les performances globale du logiciel mais les requêtes BD nécessaires lors de la génération du fichier ont un impact négatif sur les performances pour ce cas précis.

3.4.2.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

L'utilisateur devra accepter un certificat permettant l'écriture de la feuille de calculs sur son disque dur.

3.4.2.1.7 Compétence et Faisabilité technique

Formation nécessaire sur l'API Java *JXL* permettant de traiter des fichiers de type feuille de calculs.

3.4.2.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

3.4.2.1.9 Outils additionnels requis pour implémenter et tester le changement

Utilisation du paquetage jxl.jar et d'un tableur pour afficher les résultats.

3.4.2.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

3.4.2.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Quelques éclaircissements sur l'exigence avec le client et avec le maître de stage sont nécessaires. Fonctionnalité soumise au maître de stage pour validation.

3.4.2.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

3.4.2.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation si utilisation d'un tableur gratuit.

3.4.2.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser. Une aide par bulle d'information est aussi prévue.

3.4.2.2 Éléments du système affectés par le changement proposé

3.4.2.2.1 Changements requis pour l'interface utilisateur

Modification de l'interface d'analyse par ajout d'un cadre permettant la génération de la feuille de calculs.

3.4.2.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation.

3.4.2.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Création de nouvelles méthodes Java.
- Intégration de la libraire *JXL*.

3.4.2.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

3.4.2.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Création de la méthode generateExcel() dans la classe PanelAnalyze.java.
- Création de méthodes spécifiques dans les classes gérant les tables de BD suivantes :
    - DBTableKpa.java
    - DBTableQuestionnaire.java
    - DBTableQuestion.java
    - DBTableAssessment.java
    - DBTableAssesse.java
    - DBTableAssessor.java

3.4.2.2.6 Changements requis aux scripts de compilation

Aucun changement.

3.4.2.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

3.4.2.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront

exécutés.

3.4.2.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation.

3.4.2.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

3.4.2.2.11 Composant logiciel externe qui doit être acheté

Aucun si utilisation d'un tableur gratuit.

3.4.2.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

3.4.2.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Un espace minime pour le stockage de la feuille de calculs sera nécessaire sur le disque dur de l'utilisateur et impact négatif sur les performances réseau de part les requêtes à la BD.

3.4.2.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

3.4.2.3 Estimation de l'effort pour 0000006 - Nouvelle fonctionnalité : exporter toutes les informations d'évaluation dans une feuille de calculs

Tâche	Effort (heure)
Analyse du problème	2
Modification du code source et de la base de données	15
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**19**

3.4.3 0000008 - Nouvelle fonctionnalité : possibilité de trier les résultats dans la feuille de calculs

3.4.3.1 Implications du changement proposé

3.4.3.1.1 Conflit avec les spécifications actuelles

Pas de conflit, il s'agit d'une nouvelle exigence fonctionnelle.

3.4.3.1.2 Conflit avec les autres demandes de changement

Requête à réaliser après la requête 0000006.

3.4.3.1.3 Conséquence de ne pas mettre en œuvre le changement

Exigence cliente non implémentée.

3.4.3.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'obtenir un ordre des pratiques rendant difficile l'analyse des résultats si la modification n'est pas opérée avec précaution et suivie de tests.

3.4.3.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent accessibles en dehors de l'application de manière triée et l'utilité de l'application n'en est qu'augmentée. Pas d'impact sur les performances (tri effectué dans la requête BD).

3.4.3.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

3.4.3.1.7 Compétence et Faisabilité technique

Aucune formation supplémentaire n'est nécessaire.

3.4.3.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

3.4.3.1.9 Outils additionnels requis pour implémenter et tester le changement

Un tableur.

3.4.3.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

3.4.3.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Quelques éclaircissements sur l'exigence avec le client et avec le maître de stage sont nécessaires. Fonctionnalité soumise au maître de stage pour validation.

3.4.3.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

3.4.3.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune augmentation si utilisation d'un tableur gratuit.

3.4.3.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser. Une aide par bulle d'information est aussi prévue.

3.4.3.2 Éléments du système affectés par le changement proposé

3.4.3.2.1 Changements requis pour l'interface utilisateur

Modification de l'interface d'analyse par ajout d'un cadre permettant le tri des pratiques pour la génération de la feuille de calculs.

3.4.3.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation.

3.4.3.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Modification de méthodes Java.
- Intégration de la libraire *JXL*.

3.4.3.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

3.4.3.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Modification de la méthode generateExcel() dans la classe PanelAnalyze.java.
- Modification de la méthode getAnswersList() dans la classe DbQueriesAnalyse.java.

3.4.3.2.6 Changements requis aux scripts de compilation

Aucun changement.

3.4.3.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

3.4.3.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

3.4.3.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation.

3.4.3.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

3.4.3.2.11 Composant logiciel externe qui doit être acheté

Aucun si utilisation d'un tableur gratuit.

3.4.3.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

3.4.3.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Un espace minime pour le stockage de la feuille de calculs sera nécessaire sur le disque dur de l'utilisateur et utilisation minime du réseau (une requête de BD).

3.4.3.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

3.4.3.3 Estimation de l'effort pour 0000008 - Nouvelle fonctionnalité : possibilité de trier les résultats dans la feuille de calculs

Tâche	Effort (heure)
Analyse du problème	2
Modification du code source et de la base de données	5
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**9**

3.4.4 0000007 - Nouvelle fonctionnalité : affichage de l'identifiant et du commentaire de chaque pratique dans le graphique résultat

3.4.4.1 Implications du changement proposé

3.4.4.1.1 Conflit avec les spécifications actuelles

Pas de conflit, il s'agit d'une nouvelle exigence fonctionnelle.

3.4.4.1.2 Conflit avec les autres demandes de changement

Pas de conflit.

3.4.4.1.3 Conséquence de ne pas mettre en œuvre le changement

Exigence cliente non implémentée et graphique résultat peu pertinent puisque impossibilité de savoir à quelles pratiques correspondent les résultats.

3.4.4.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'erreur de représentation du résultat d'une pratique si la modification n'est pas opérée avec précaution et suivie de tests. Cela engendrerait une analyse erronée.

3.4.4.1.5 Impact sur les attributs de qualité et performance

La qualité augmente puisque les résultats deviennent accessibles dynamiquement en passant la souris sur le graphique et l'utilité de l'application n'en est qu'augmentée. Impact négatif sur les performances (requêtes BD).

3.4.4.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

3.4.4.1.7 Compétence et Faisabilité technique

Aucune formation supplémentaire n'est nécessaire, la formation 2.4.2.1.7 est suffisante.

3.4.4.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête (suite à 2.4.2.1.8).

3.4.4.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

3.4.4.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

3.4.4.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Proposition de solutions au client et au maître de stage afin d'opter pour la meilleure solution. Fonctionnalité soumise au maître de stage pour validation.

3.4.4.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

3.4.4.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

3.4.4.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser, utilisation d'une info bulle par exemple.

3.4.4.2 Éléments du système affectés par le changement proposé

3.4.4.2.1 Changements requis pour l'interface utilisateur

Des info-bulles apparaîtront sur le graphique lors du passage de la souris au-dessus d'une zone.

3.4.4.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation.

3.4.4.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Création et modification de méthodes Java.

3.4.4.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

3.4.4.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

- Modification des classes PanelResultAnalyse.java, PanelAnalyze.java, DBTableKpa.java, DBTableQuestion.java et DBTableQuestionnaire.java.

3.4.4.2.6 Changements requis aux scripts de compilation

Aucun changement.

3.4.4.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

3.4.4.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

3.4.4.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation.

3.4.4.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

3.4.4.2.11 Composant logiciel externe qui doit être acheté

Aucun.

3.4.4.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

3.4.4.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Utilisation du réseau pour les requêtes à la BD.

3.4.4.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

3.4.4.3 Estimation de l'effort pour 0000007 - Nouvelle fonctionnalité : affichage de l'identifiant et du commentaire de chaque pratique dans le graphique résultat

Tâche	Effort (heure)
Analyse du problème	2
Modification du code source et de la base de données	7
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**11**

3.4.5 0000009 - Nouvelle fonctionnalité : ajout d'onglets dans la partie analyse permettant d'obtenir toutes les informations d'évaluation et les suggestions d'amélioration

3.4.5.1 Implications du changement proposé

3.4.5.1.1 Conflit avec les spécifications actuelles

Pas de conflit, il s'agit d'une nouvelle exigence fonctionnelle.

3.4.5.1.2 Conflit avec les autres demandes de changement

Pas de conflit.

3.4.5.1.3 Conséquence de ne pas mettre en œuvre le changement

Exigence cliente non implémentée et diminution de l'utilité de l'outil.

3.4.5.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'erreur sur la suggestion d'amélioration des pratiques si la modification n'est pas opérée avec précaution et suivie de tests. Cela engendrerait une analyse erronée et pourrait amener l'utilisateur à faire de mauvais choix d'amélioration.

3.4.5.1.5 Impact sur les attributs de qualité et performance

La qualité augmente grâce à une facilité accrue de l'accès aux diverses informations pertinentes d'évaluation et l'utilité de l'application n'en est qu'augmentée. Impact négatif sur les performances dû aux requêtes BD.

3.4.5.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

3.4.5.1.7 Compétence et Faisabilité technique

Formation nécessaire sur la gestion d'onglets et l'affichage de tableaux en Java.

3.4.5.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

3.4.5.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

3.4.5.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

3.4.5.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Discussion avec le maître de stage afin de trouver la meilleure solution de suggestion d'amélioration des pratiques. Fonctionnalité soumise au maître de stage pour validation.

3.4.5.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

3.4.5.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

3.4.5.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser et facile à comprendre.

3.4.5.2 Éléments du système affectés par le changement proposé

3.4.5.2.1 Changements requis pour l'interface utilisateur

Ajout de deux onglets : *Assessment Datas* et *Improvement Suggestion*.
Le premier onglet reprendra les données générales d'évaluation entrées par l'utilisateur au début de son processus d'évaluation. Il comprendra également les détails d'évaluation de chaque pratique avec: son identifiant, sa cotation, son pourcentage d'accomplissement et son commentaire.
L'onglet « Improvement Suggestion » affichera, par domaine et par niveau, les pratiques en-deçà du niveau moyen de ce domaine.

3.4.5.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation.

3.4.5.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Création et modification de méthodes Java.

3.4.5.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

3.4.5.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification des classes PanelAnalyze.java, DBTableQuestionnaire.java, DBTableKpa.java, DBTableQuestionnaire.java, DBTableQuestion.java, DBTableAssessment.java, DBTableAssesse.java, DBTableAssessor.java via principalement les méthodes PanelAnalyze(Refreshable parent), getRatingName(Float result), generateDataTable(Collection answersList) et suggest().

3.4.5.2.6 Changements requis aux scripts de compilation

Aucun changement.

3.4.5.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

3.4.5.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

3.4.5.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation.

3.4.5.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

3.4.5.2.11 Composant logiciel externe qui doit être acheté

Aucun.

3.4.5.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

3.4.5.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Utilisation du réseau pour les requêtes à la BD.

3.4.5.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

3.4.5.3 Estimation de l'effort pour 0000009 - Nouvelle fonctionnalité : ajout d'onglets dans la partie analyse permettant d'obtenir toutes les informations d'évaluation et les suggestions d'amélioration

Tâche	Effort (heure)
Analyse du problème	5
Modification du code source et de la base de données	40
Modification des documents (design et produits livrables)	1
Test informel	3
Total :	**49**

4. Analyse d'impacts de la Révision 3

4.1 Sommaire

Cette section comporte un tableau contenant un sommaire des cinq requêtes de maintenance les plus prioritaires pour cette révision.

A l'issue de la deuxième révision, l'outil fût lancé en opération suite à quoi des feedbacks ont été reçus (voir la partie *Révision 3* en annexes). De ces feedbacks ressortent diverses nouvelles demandes de maintenance. Ces nouvelles demandes s'ajoutent aux requêtes issues des révisions précédentes et non encore traitées. Les cinq principales requêtes de maintenance prises en compte pour cette troisième révision sont reprises dans le tableau ci-dessous. Il s'agit essentiellement de requêtes perfectives.

Chaque requête de maintenance a été classifiée selon l'entente de service en vigueur. Le tableau ci-dessous présente un sommaire des requêtes avec leur numéro de référence, leur description et la catégorie de maintenance à laquelle elles se rapportent.

Numéro de référence	Description	Catégorie de maintenance
0000011	Correction du crash de l'application qui survenait après certaines sélections d'analyse	Corrective
0000012	Nouvelle fonctionnalité : ajout de la méthode *S3MAssessment*	Perfective
0000013	Nouvelle fonctionnalité : ajout de nouveaux graphes d'analyse : *Domains Histogram, KPA Histogram and Multi-assessment Histogram*	Perfective
0000014	Présentation des pratiques du niveau 0 sous forme intuitive positive: "Yes" obtient un score de 100% et "No" un score de 0%. Correction des données utilisateurs déjà encodées en base de données	Perfective
0000015	Nouvelle fonctionnalité : ajout de la possibilité de modifier une ancienne évaluation	Perfective

4.2 Recommandation

La présente section propose une priorisation des requêtes afin d'éviter des conflits lors de l'implantation des diverses solutions.

4.2.1 Priorisation des problèmes

L'ordre du tableau au point 4.1 sera suivi pour la réalisation de cette troisième révision.

4.3 Livraison

Suite à l'analyse d'effort de la Révision 3 (le tableau ci-dessous), à l'entente convenue avec le maître de stage, Dr Alain April, et avec les clients, la troisième révision sera mise en production à la fin juin (travail débutant à la mi-février). Le tableau ci-dessous reprend une estimation de l'effort nécessaire pour la réalisation des cinq requêtes prioritaires.

Numéro de référence	Description	Effort (heures)
0000011	Correction du crash de l'application qui survenait après certaines sélections d'analyse	11
0000012	Nouvelle fonctionnalité : ajout de la méthode *S3MAssessment*	23
0000013	Nouvelle fonctionnalité : ajout de nouveaux graphes d'analyse : *Domains Histogram, KPA Histogram and Multi-assessment Histogram*	23
0000014	Présentation des pratiques du niveau 0 sous forme intuitive positive: "Yes" obtient un score de 100% et "No" un score de 0%. Correction des données utilisateurs déjà encodées en base de données	6
0000015	Nouvelle fonctionnalité : ajout de la possibilité de modifier une ancienne évaluation	22
	Total :	**85 heures**

4.4 Analyses d'impacts détaillées

4.4.1 0000011 - Correction du crash de l'application qui survient après certaines sélections d'analyse

4.4.1.1 Implications du changement proposé

4.4.1.1.1 Conflit avec les spécifications actuelles

Pas de conflit.

4.4.1.1.2 Conflit avec les autres demandes de changement

Pas de conflit si on corrige cette défaillance avant de poursuivre l'évolution de l'outil.

4.4.1.1.3 Conséquence de ne pas mettre en œuvre le changement

Le client ne peut pas accéder à certaines de ses données d'évaluation bien qu'elles soient bel et bien stockées en base de données.

4.4.1.1.4 Risques d'effectuer le changement proposé et effets de bord

Aucun, il s'agit de corriger une défaillance bien déterminée.

4.4.1.1.5 Impact sur les attributs de qualité et performance

La qualité augmente et les performances sont inchangées.

4.4.1.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

4.4.1.1.7 Compétence et Faisabilité technique

Aucune supplémentaire.

4.4.1.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

4.4.1.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

4.4.1.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

4.4.1.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Tests en interne et validation avec clients.

4.4.1.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

4.4.1.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

4.4.1.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

aucun.

4.4.1.2 Éléments du système affectés par le changement proposé

4.4.1.2.1 Changements requis pour l'interface utilisateur

Aucun.

4.4.1.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Aucun.

4.4.1.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Modification de requêtes de BD.
- Modification de méthodes Java.

4.4.1.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

4.4.1.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification de méthodes issues des classes PanelSettingAnalyze.java, DBTableAssesse.java, DBTableAssessment.java.

4.4.1.2.6 Changements requis aux scripts de compilation

Aucun changement.

4.4.1.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

4.4.1.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

4.4.1.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Aucun.

4.4.1.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

4.4.1.2.11 Composant logiciel externe qui doit être acheté

Aucun.

4.4.1.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

4.4.1.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Pas d'effet supplémentaire.

4.4.1.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

4.4.1.3 Estimation de l'effort pour 0000011 - Correction du crash de l'application qui survient après certaines sélections d'analyse

Tâche	Effort (heure)
Analyse du problème	1
Modification du code source et de la base de données	9
Modification des documents (design et produits livrables)	0

Test informel	1
Total :	**11**

4.4.2 0000012 - Nouvelle fonctionnalité : ajout de la méthode *S3MAssessment*

4.4.2.1 Implications du changement proposé

4.4.2.1.1 Conflit avec les spécifications actuelles

Il s'agit de nouvelles spécifications à intégrer de manière indépendante par rapport aux spécifications actuelles.

4.4.2.1.2 Conflit avec les autres demandes de changement

Pas de conflit.

4.4.2.1.3 Conséquence de ne pas mettre en œuvre le changement

Manque d'une opportunité d'augmenter l'utilité de l'outil.

4.4.2.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'erreur dans le processus d'évaluation si l'intégration n'est pas opérée avec précaution et suivie de tests. Cela engendrerait une analyse erronée et pourrait amener l'utilisateur à faire de mauvais choix d'amélioration.

4.4.2.1.5 Impact sur les attributs de qualité et performance

Pas d'impact, ces attributs restent les même par rapport à la méthode d'évaluation déjà implémentée.

4.4.2.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

4.4.2.1.7 Compétence et Faisabilité technique

Aucune supplémentaire si ce n'est sur l'accord quant à la bonne implémentation de la méthode.

4.4.2.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

4.4.2.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

4.4.2.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

4.4.2.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Discussion avec le maître de stage et avec l'auteur de la méthode S3MAssessment. Fonctionnalité soumise au maître de stage et à divers intervenants pour validation.

4.4.2.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

4.4.2.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

4.4.2.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser et facile à comprendre.

4.4.2.2 Éléments du système affectés par le changement proposé

4.4.2.2.1 Changements requis pour l'interface utilisateur

Ajout d'un bouton de sélection de la méthode d'évaluation dans le panneau « Information about the assessment » et ajout de l'information quant à la méthode choisie dans le tableau « Assessment Information ».

4.4.2.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation et utilisation de la colonne *AssessmentType* de la table *Assessment*.

4.4.2.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Création et modification de méthodes Java.

4.4.2.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

4.4.2.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification des classes PanelAssessment.java, PanelSettingQuestion.java, PanelQuestionnaire.java, DBTableQuestionnaire.java et DBTableAssessment.java via principalement les méthodes isSMAssessmentMethod(assessmentId), endAssessment(), select(), next(), isUnder80() et setSMAssessmentMethod(smAssessmentMethod).

4.4.2.2.6 Changements requis aux scripts de compilation

Aucun changement.

4.4.2.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

4.4.2.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

4.4.2.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation.

4.4.2.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

4.4.2.2.11 Composant logiciel externe qui doit être acheté

Aucun.

4.4.2.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

4.4.2.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Utilisation du réseau pour quelques requêtes BD initiales.

4.4.2.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

4.4.2.3 Estimation de l'effort pour 0000012 - Nouvelle fonctionnalité : ajout de la méthode *S3MAssessment*

Tâche	Effort (heure)
Analyse du problème	3
Modification du code source et de la base de données	16
Modification des documents (design et produits livrables)	1
Test informel	3
Total :	**23**

4.4.3 0000013 - Nouvelle fonctionnalité : ajout de nouveaux graphes d'analyse : *Domains Histogram, KPA Histogram and Multi-assessment Histogram*

4.4.3.1 Implications du changement proposé

4.4.3.1.1 Conflit avec les spécifications actuelles

Pas de conflit.

4.4.3.1.2 Conflit avec les autres demandes de changement

Pas de conflit mais faire attention à la requête 0000015 car une précédente évaluation peut-être modifiée (problème déjà réglé par la manière d'implémenter, il s'agit de s'assurer de la mise à jour des

données des graphiques).

4.4.3.1.3 Conséquence de ne pas mettre en œuvre le changement

Le client ne peut pas accéder aux nouvelles représentations des données et outils de comparaison, ce qui lui rendra plus ardue sa tâche finale d'amélioration de processus.

4.4.3.1.4 Risques d'effectuer le changement proposé et effets de bord

De mauvais résultats peuvent entrainer de mauvais choix d'amélioration de processus de la part de l'utilisateur.

4.4.3.1.5 Impact sur les attributs de qualité et performance

Les performances seront moins bonnes dû aux nouveaux calculs. Toutefois, une réorganisation des workflows visant à ne calculer qu'à la demande explicite de l'utilisateur (bouton *Compute*) permet de contrebalancer ce problème.

4.4.3.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun.

4.4.3.1.7 Compétence et Faisabilité technique

Parfaire la connaissance de la gestion des graphiques et interfaces Java.

4.4.3.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

4.4.3.1.9 Outils additionnels requis pour implémenter et tester le changement

Jigloo (déjà disponible dans l'environnement).

4.4.3.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

4.4.3.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Tests en interne et validation avec clients.

4.4.3.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

4.4.3.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

4.4.3.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Mettre à jour le guide utilisateur pour former la clientèle et répondre à leurs questions.

4.4.3.2 Éléments du système affectés par le changement proposé

4.4.3.2.1 Changements requis pour l'interface utilisateur

Quatre ajouts/modifications d'onglets dans la partie *Representation* : *Domains Histogram, KPA Histogram, Practices Histogram et Multi-assessment Histogram.*

4.4.3.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mettre à jour le guide utilisateur.

4.4.3.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Ajout et modification de requêtes de BD.
- Ajout de méthodes Java.

4.4.3.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

4.4.3.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification de méthodes issues des classes PanelAnalyze.java, DBTableKpa.java et DBTableQuestionnaire.java. Il s'agit principalement des méthodes drawDomainsGraph(), drawKpaGraph(), createKpaChart(), multiAssessment() et createMultiAssessChart().

4.4.3.2.6 Changements requis aux scripts de compilation

Aucun changement.

4.4.3.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

4.4.3.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

4.4.3.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

User guide.

4.4.3.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

4.4.3.2.11 Composant logiciel externe qui doit être acheté

Aucun.

4.4.3.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

4.4.3.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Pas d'effet supplémentaire.

4.4.3.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact.

4.4.3.3 Estimation de l'effort pour 0000013 – Nouvelle fonctionnalité : ajout de nouveaux graphes d'analyse : *Domains Histogram, KPA Histogram and Multi-assessment Histogram*

Tâche	Effort (heure)
Analyse du problème	2
Modification du code source et de la base de données	18
Modification des documents (design et produits livrables)	2
Test informel	1
Total :	**23**

4.4.4 0000014 - Présentation des pratiques du niveau 0 sous forme intuitive positive: "Yes" obtient un score de 100% et "No" un score de 0%. Correction des données utilisateurs déjà encodées en base de données

4.4.4.1 Implications du changement proposé

4.4.4.1.1 Conflit avec les spécifications actuelles

Les pratiques du premier niveau – *0* – sont présentées dans le modèle S3M sous forme négative menant ainsi l'utilisateur à devoir évaluer une pratique négativement si l'exigence de la pratique est bien vérifiée. Ceci rend l'évaluation non intuitive.

4.4.4.1.2 Conflit avec les autres demandes de changement

Pas de conflit si la RM 0000013 permet la mise à jour automatique des graphiques.

4.4.4.1.3 Conséquence de ne pas mettre en œuvre le changement

Manque d'intuitivité de l'outil.

4.4.4.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'erreur dans le processus d'évaluation si les pratiques de niveau 0 ne sont pas correctement formulées et si le renversement de signification des réponses données n'est pas correctement pris en compte. Cela engendrerait une analyse erronée et pourrait amener l'utilisateur à faire de mauvais choix d'amélioration.

4.4.4.1.5 Impact sur les attributs de qualité et performance

Pas d'impact.

4.4.4.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Valider la nouvelle formulation positive des pratiques.

4.4.4.1.7 Compétence et Faisabilité technique

Aucune supplémentaire.

4.4.4.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

4.4.4.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

4.4.4.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

4.4.4.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Validation des pratiques par le maître de stage qui également l'auteur des pratiques initiales. Fonctionnalité soumise au maître de stage et à divers intervenants pour validation.

4.4.4.1.12 Effort perdu si le changement est mis en œuvre

L'effort lié à l'encodage des pratiques de niveau 0 et celui lié aux calculs basés sur les valeurs de ces pratiques.

4.4.4.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

4.4.4.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser et facile à comprendre. L'outil gagnera en intuitivité.

4.4.4.2 Éléments du système affectés par le changement proposé

4.4.4.2.1 Changements requis pour l'interface utilisateur

La valeur en pourcentage liée aux réponses « yes » ou « no » est inversée.

4.4.4.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation et modification des valeurs présentes en BD concernant l'évaluation de ces pratiques (de sorte que les évaluations passées soient toujours valides).

4.4.4.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Modification de méthodes Java.

4.4.4.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

4.4.4.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification de la classe PanelQuestionnaire.java via la méthode adding().

4.4.4.2.6 Changements requis aux scripts de compilation

Aucun changement.

4.4.4.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

4.4.4.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

4.4.4.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour de la partie d'écran d'aide présente dans l'écran d'évaluation des pratiques.

4.4.4.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

4.4.4.2.11 Composant logiciel externe qui doit être acheté

Aucun.

4.4.4.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

4.4.4.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Aucun effet.

4.4.4.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact car BD centralisée.

4.4.4.3 Estimation de l'effort pour 0000014 – Présentation des pratiques du niveau 0 sous forme intuitive positive: "Yes" obtient un score de 100% et "No" un score de 0%. Correction des données utilisateurs déjà encodées en base de données

Tâche	Effort (heure)
Analyse du problème	1
Modification du code source et de la base de données	3
Modification des documents (design et produits livrables)	1
Test informel	1
Total :	**6**

4.4.5 0000015 - Nouvelle fonctionnalité : ajout de la possibilité de modifier une ancienne évaluation

4.4.5.1 Implications du changement proposé

4.4.5.1.1 Conflit avec les spécifications actuelles

Il était initialement prévu d'évaluer en une seule et unique fois l'ensemble des processus.

4.4.5.1.2 Conflit avec les autres demandes de changement

Pas de conflit.

4.4.5.1.3 Conséquence de ne pas mettre en œuvre le changement

Manque d'une opportunité d'augmenter l'utilité et l'utilisabilité de l'outil.

4.4.5.1.4 Risques d'effectuer le changement proposé et effets de bord

Risque d'erreur dans le processus d'évaluation si l'intégration n'est pas opérée avec précaution et suivie de tests. Cela engendrerait une analyse erronée et pourrait amener l'utilisateur à faire de mauvais choix d'amélioration.

4.4.5.1.5 Impact sur les attributs de qualité et performance

Pas d'impact, ces attributs restent les même par rapport à la méthode d'évaluation déjà implémentée.

4.4.5.1.6 Impact sur les aspects critiques (ex : sécurité, certification)

Aucun, seul l'utilisateur identifié par l'outil peut modifier une ancienne évaluation.

4.4.5.1.7 Compétence et Faisabilité technique

Aucune supplémentaire.

4.4.5.1.8 Impact sur les ressources informatiques (développement, test, opération)

L'architecture de développement actuel répond amplement aux exigences de développement de cette requête.

4.4.5.1.9 Outils additionnels requis pour implémenter et tester le changement

Aucun.

4.4.5.1.10 Impact sur le plan de projet

La présente requête n'a pas vraiment d'impact sur les échéanciers car elle est planifiée.

4.4.5.1.11 Prototypage et implication de l'utilisateur pour fin de vérification

Discussion avec le maître de stage. Fonctionnalité soumise au maître de stage et à divers intervenants pertinents pour validation.

4.4.5.1.12 Effort perdu si le changement est mis en œuvre

Aucun.

4.4.5.1.13 Augmentation du coût du produit ou des redevances à un produit d'un tiers externe

Aucune.

4.4.5.1.14 Impact sur la mise en marché, la production, la formation ou le service à la clientèle

Impact positif pour la mise sur le marché car il s'agit d'une nouvelle fonctionnalité. La fonctionnalité sera implémentée de manière à ce qu'elle soit simple à utiliser et facile à comprendre.

4.4.5.2 Éléments du système affectés par le changement proposé

4.4.5.2.1 Changements requis pour l'interface utilisateur

Ajout d'une question posée à l'utilisateur lorsqu'il sélectionne le menu *Assessment* : « Do you wish to complete a previous assessment ? ». Si sa réponse est oui, un nouveau panneau lui est proposé afin qu'il sélectionne l'évaluation qu'il veut modifier. Sinon, le panneau de création d'une nouvelle évaluation lui est proposé.

4.4.5.2.2 Changements requis aux rapports, bases de données ou fichiers de données

Mise à jour du guide d'utilisation.

4.4.5.2.3 Composantes du design qui doivent être créées, modifiées ou supprimées

- Création de nouvelles requêtes de BD.
- Création et modification de méthodes Java.

4.4.5.2.4 Composants matériels qui doivent être créés, modifiés ou supprimés

Aucune modification n'est requise.

4.4.5.2.5 Fichiers sources qui doivent être créés, modifiés ou supprimés

Modification des classes PanelQuestionnaire.java et DBTableQuestionnaire.java via principalement les méthodes adding() et getQuestionnaireId(int questionNumber, int assessmentId).

4.4.5.2.6 Changements requis aux scripts de compilation

Aucun changement.

4.4.5.2.7 Identifier les cas de tests (unitaires, intégration, système et acceptation) qui doivent être modifiés ou supprimés

Aucun changement car le projet n'a pas d'architecture de tests.

4.4.5.2.8 Estimer les nombres de nouveaux tests (unitaires, intégration, système et d'acceptation) qui devront être créés

Aucun car le projet n'a pas d'architecture de test. Seulement des tests informels seront exécutés.

4.4.5.2.9 Écrans d'aide, manuels d'utilisateur, manuel de formation ou autre documentation qui devront être créées ou modifiés

Mise à jour du guide d'utilisation et de l'écran du menu « Assessment ».

4.4.5.2.10 Autre système, application, librairie ou composant matériel affecté par le changement proposé

Aucun changement.

4.4.5.2.11 Composant logiciel externe qui doit être acheté

Aucun.

4.4.5.2.12 Impact qu'aura le changement proposé sur les plans de projet, d'assurance qualité, de contrôle de configuration ou tout autre plan

Le plan de projet prévoit cette requête de maintenance.

4.4.5.2.13 Quantifier tout effet qu'aura le changement proposé sur le budget des ressources limitées, comme la mémoire, temps de CPU, bande passante du réseau, plage de temps réel

Utilisation supplémentaire mineure du réseau pour quelques requêtes BD.

4.4.5.2.14 Impact qu'aura le changement proposé sur les systèmes déjà en place si la composante affectée n'est pas totalement compatible vers l'arrière

Aucun impact car BD centralisée.

4.4.5.3 Estimation de l'effort pour 0000015 - Nouvelle fonctionnalité : ajout de la possibilité de modifier une ancienne évaluation

Tâche	**Effort (heure)**
Analyse du problème	3
Modification du code source et de la base de données	16
Modification des documents (design et produits livrables)	1
Test informel	2
Total :	**22**

5. Annexes

Cette partie présente les demandes formulées et les commentaires pour chaque révision.

5.1 Révision 1

Sujet: Mémoire
De: Alain April <Alain.April@etsmtl.ca>
Date: Sun, 07 Oct 2007 12:54:18 -0400
Pour :: Cédric Di Tomaso cedric.di-tomaso.1@ens.etsmtl.ca

```
j'aimerais que vous preniez le temps de regarder le travail de session #16
qui porterait sur:
- faire fonctionner le logiciel S3m à l'école (contacter Patrice Dion dès son
retour de vacances)
- faire la démonstration à Freescale et à IBM Australie par la suite
- faire les modifications fonctionnelles pour Freescale (si-besoin)
- faire des améliorations (ré-ingénierie)
-
Vous auriez à vous assurer que la méthode d'évaluation qui serait développée
par
Vincent est bel et bien supportée par l'outil S3m.
Alain April
```

Sujet: Re: s3massess en ligne
De: Alain April <Alain.April@etsmtl.ca>
Date: Mon, 22 Oct 2007 17:55:10 -0400
Pour :: Cédric Di Tomaso <cedric.di-tomaso.1@ens.etsmtl.ca>

```
Bonsoir,
Pour l'aspect complétude des pratiques: J'ai toutes les pratiques en Anglais
dans
un format word. Il serait possible pour vous d'effectuer un petit programme de
chargement automatique pour remplir la BD avec toutes les pratiques en
Anglais.
Les autres problèmes de l'outils sont reliés au fait que les exigences des
versions
passées ont été assez simple !
Nous n'avons pas plus de documentation que ce qui est disponible en ce moment.
Bien sur ce logiciel est encore considéré comme un prototype et en faire
l'avancement peut faire partie de votre mémoire si nous obtenons des
spécifications
pour son amélioration. Les spécifications doivent provenir de:
1) vos travaux de recherche concernant des logiciels similaires
2) des problèmes que vous identifiez avec le logiciel actuel
3) les fonctionnalités que Freescale aurait besoin pour l'utiliser dans
l'entreprise

Cordialement,
Alain April
```

5.2 Révision 2

Sujet: RE: Presentation of the S3m Assess tool
De: "Milam Caroline" <Caroline.Milam@freescale.com>
Date: Tue, 6 Nov 2007 16:56:33 -0700
Pour :: Cédric Di Tomaso <cedric.di-tomaso.1@ens.etsmtl.ca>
Copie à :: "Alain April" <alain.april@etsmtl.ca>
Cedric,

I prefer to try out this tool locally, so please provide a zip file when convenient.
Here are some initial requirements feedback:
1. Provide a link from the results graphic to the responses for that Key Process Area (what were the questions, what was the comment)
2. Provide a report of the comments associated with various filters including: KPA, Maturity Level, Achievement Level
3. Integrate with utility to provide suggestions on how to improve based on assessment results - wpuld be ideal to have as part of suggestion #1.
4. In general, ability to output results to excel is always desirable...for example, would need to be able to paste a results graphic into a management presentation.
Regards,
Caroline Milam
ATX MES/UI Domain Mgr
512.895.6640
Freescale Semiconductor
General Business Information
Freescale Internal Use Only
Freescale Confidential Proprietary

Sujet: Re: Fw: Final agreement - Signed by IBM and ETS
De: Laxman N Vasan <vasan@au1.ibm.com>
Date: Fri, 30 Nov 2007 14:45:31 +1100
Pour :: Cédric Di Tomaso <cedric.ditomaso@gmail.com>
Cedric, I must be technically challenged !! I could not install and get the MYSQL database working. Do you have any further instruction that can help me or any thoughts on what I may have done wrong ? I got the installation completed but the configuration part failed. Apprecitae your help..thanks
Laxman Vasan
IBM Global Business Services
FF42, 55 Coonara Avenue, West Pennant Hills,
NSW 2125, Australia
Phone 61-2- 93544265, Mobile 61-412-282569,
Fax 61-2-93547738
Internet vasan@au1.ibm.com

5.3 Révision 3

Sujet: Re: Demande de feedback
De: "Jean-Marc Desharnais" <desharnaisjm@gmail.com>
Date: Thu, 21 Feb 2008 10:23:31 -0500
Pour :: "Cédric Di Tomaso" <cedric.ditomaso@gmail.com>
Copie à :: "Alain April" <alain.april@etsmtl.ca>
Précision sur ma réponse précédente. J'ai vu qu'il y avait des recommandations du modèle voir 4.3.1 etc. Il faut être plus explicite dans les recommandations ou encore mettre plus de texte. En ce qui concerne les problèmes potentiels je crois qu'une revue de la littérature peut aider. Il n'est pas nécessaire que ce soit
fait pour tout maintenant, mais en réalisant quelques modules ça peut préparer d'autres chercheurs à aller
plus loin.

Le 21/02/08, **Jean-Marc Desharnais** <desharnaisjm@gmail.com> a écrit :
Bonjour,
Le but de l'exercice devrait être expliqué. Il n'est pas clair. Évidemment je le connais puisque je connais bien le travail d'Alain April. Pour une personne extérieure ce serait difficile à suivre.

J'ai regardé les résultats à partir de l'option "analyze". Le graphique est utile pour voir d'un coup d'oeil les résultats. De plus on peut aller d'un domaine à un autre pour voir les différents résultats. J'ai regardé aussi la partie interprétation. Je crois qu'il pourrait y avoir amélioration de ce côté. Il faudrait aller plus loin que juste des constatations, ce qui représenterait un réel intérêt pour l'utilisateur. Dans le contexte d'un système qui offre une certaine expertise c'est dans cette partie du logiciel que se trouverait toute l'expertise. Il pourrait y avoir des recommandation sur ce qui doit être fait pour améliorer la situation, ou des commentaires sur les problèmes potentiels que la situation peut créer. Évidemment il faudra que tu consulte un expert ou que tu fouilles dans la littérature (ex: dans le livre d'Alain).

Sujet: RE: [Fwd: Demande de feedback]
De: <Alain.Abran@etsmtl.ca>
Date: Sun, 2 Mar 2008 04:49:06 -0500
Pour :: <cedric.ditomaso@gmail.com>

Bonjour M. Di Tomasso,
J'ai pu accéder maintenant à votre logiciel.
Très intéressant, et il sera certainement très utile pour les évaluations, et après
pour les recommendations.
Voici quelques commentaires mineurs:
Pour les questions de niveau 0, la formulation de la question est de forme 'négation'
ce qui rend difficile (non intuitif) de répondre correctement.
(également à l'affichage des analyses le 'not acheived or Yes' ainsi que le 'fully acheived or No' est non intuitif.
Pour les codes de réponse (N, P, L, F) il y aurait peut-être à mettre une échelle qui
indique les significations, comme pour la présentation des analyses. (mais pas obligatoire).
Lorsque l'on passe d'une question à l'autre, la réponse précédente reste indiquée dans
le rond. Je préférerais qu'elle soit remise à vide (opinion personnelle).
Tout le reste me semble très bien.
Félicitations pour votre travail.
Alain
Alain Abran
Marie Curie International Fellow 2007-2008
Universidad del Pais Vasco, San Sebastian - Donostia, Spain
Professeur - Département: génie logiciel & technologies de l'information
Professor - Department of Software Engineering and Information Technologies
École de technologie supérieure - Université du Québec
Montréal, Canada H3W 1T8

Sujet: Re: [Fwd: Demande de feedback]
De: Alain April <Alain.April@etsmtl.ca>
Date: Sun, 02 Mar 2008 12:09:23 -0500
Pour :: Cédric Di Tomaso <cedric.ditomaso@gmail.com>

Cédric,
Je crois qu'il faudrait reformuler les pratiques de niveau 0.
Ce n'est pas la première fois que le commentaire fait surface !
Ce serait une belle contribution de votre étude.
Ce ne sera pas un gros travail et vous pourriez décider de l'implanter tout de suite dans l'outil ?
Cordialement,
Prof. Dr. Ing. Alain April
École de Technologie Supérieure

1100 Notre-Dame West , Office A-3482
Montréal, Québec, Canada, H3C 1K3
(514) 396-8682

Sujet: Re: Minutes of our meeting today
De: Alain April <Alain.April@etsmtl.ca>
Date: Fri, 28 Mar 2008 23:12:12 -0400
Pour :: Milam Caroline <Caroline.Milam@freescale.com>
Copie à :: Cédric Di Tomaso <cedric.ditomaso@gmail.com>, Vincent Lebrun <vincent.lebrun@student.fundp.ac.be>, Molyneaux Harold <H.Molyneaux@freescale.com>,
naji.habra@info.fundp.ac.be
Hello All,
Although the meeting was difficult because of audio conferencing issues with the Belgium toll-free
access point we were able to discuss a number of
items in two seperate meetings. First with Caroline and Harold and after with Cédric and Vincent
over SKYPE.
Action: 0.1 - Either we use another software that includes IP voice or Caroline identifies valid
Belgium toll-free numbers for next conference call.
Highlights/Minutes of the Freescale-S3M conference call held on April 28th, 2008 from 3:00 to
4:30 PM
Participants: Caroline Milam - Freescale USA
Harold Molyneaux - Freescale USA
Dr. Alain April - ETS University Montreal
Cédric Di Tomaso - FUNDP University Belgium
Vincent Lebrun - FUNDP University Belgium
Cédric Di Tomaso related topics:
2) The S3MAssess is a support tool (still an improving prototype) for the S3M assessment process.
Included is a number of issues reported by Caroline:
a) When conducting an assessment and stopping it is closed. It cannot be reopened to continue the assessment;
Re: Minutes of our meeting today
2 sur 3 1/07/2008 15:07
b) Comments are added to individual practice assessment in text and cannot be used for
report (or graph) outputs;
c) Rating is produced but recommendations seems un-usable in some cases (or hard to understand);
d) The current on-line version was developed because Caroline could not install S3MAssess earlier (difficult to configure the underlying softwares). Now the Freescale

firewall is blocking the 'on-line' Java application for security reasons. Dr. April recommended the use of VMWARE as no configuration is needed and it is easy to send an already configured 'Virtual Machine'. Freescale has not normalized VMWARE yet. Caroline has asked to try to install the S3MAssess locally again as a possibility. Cédric will send the source code of the most recent version and Dr. April will find a student to support Caroline in her install. Caroline will try to obtain the permission from Freescale to use VMWARE on one of her workstations in the mean time.

Actions: 2.1 - Cédric will describe the assessment process in tiers and possibly develop the option to continue assesssments;

2.2 - Caroline offered to explain (by drafting an output report/graph) to show a potential use of the comments for reporting (maturity and recommendations) on Freescale current processes. Cédric will review this new functionality for future improvement of S3MAssess

2.3 - Caroline offered to send a snapshot of the resulting screen and would like to understand the use of the recommendations. Cédric to look into this question;

2.4 - Cédric will send a copy of the source code of the most recent version of S3MAssess to Caroline (and install script/documentation if possible)

2.5 - Dr. April to identify a student to support Caroline in installing S3MAssess locally

2.6 - Caroline to ask for VMWARE permission use on one workstation

Cédric and Dr. April can follow-up on this topic directly with Caroline.

More general discussion/topics

3) Caroline renewed the notice that the FSL diagram should not be published outside of Freescale. Dr. April confirmed that the Freescale NDA signed applies in this case and that this diagram was never released or shown to anyone. Any thesis or publication (while there is currently no publication planned from this work) would be submitted to Caroline previous to publication.

4) UT at Austin possible collaboration has not lead to any contact yet. Caroline has tried through her organization and Dr. April through UT contacts and management. As of now it is unclear if there is a research lab or a reasercher in software engineering process improvement or software maintenance process at UT.

5) Dr. April informed Caroline and Harold of the recent publishing of the CMMi for Services which he has worked and experimented with in the Canadian defense industry recently which takes on ITIL best practices. Although it is not as specific as the S3M it can become popular in the USA and should at least be known.

6) Possible Consulting agreement. Caroline has looked at the possibilities and is looking into S3M yearly Gold membership which includes access to all the S3M documentation

and softwares plus included remote consulting (for 2 days/year). Dr. April explained that support to
specific S3M practices is also available if needed (like measurement from Dr. Abran work). Caroline explained that they may look into the Help-Desk process to prepare for an
upcoming 'Lean & Kaizen' assessment. This could be an example of a specific
Re: Minutes of our meeting today
3 sur 3 1/07/2008 15:07
work-item to discuss.
Actions: 3.1 - Cédric and Vincent will make sure that this diagram is generalized and does not
use FSL-specific labels. For example, we should not include "CSC" but perhaps "Outsourced IT Infrastructure Organization" or similar.
3.2 - Cédric and Vincent will send the new diagram (that is presented in their thesis) to Caroline for approval
4.1 - Caroline to follow-up internally with her management and Dr. April to continue to try to find a collaborator at UT at Austin to support Freescale locally.
6.1 - S3M Gold memberships will be available after the new Wiley book publication and the launch of the S3M website in a few months. Caroline to see the interest.
6.2 - Caroline and Harold to see if help needed from S3M in their Help-Desk process improvement effort.
As always we enjoy putting our students to 'real-situations' challenges so keep up the good work ! Vincent
and Cédric plan to finish their thesis this summer so
we are starting to document the results of this project as part of their thesis to be finalized this summer.
Next conference call (if technical issues are resolved) should be planned for May 7th, 2008 earlier if
possible (because of the advanced time in Belgium). I would
like **Dr. Naji Habra** from Belgium to join in that conferece call to see what great work the students have
been able to do progressively.
Best Regards and health,
Prof. Dr. Ing. Alain April
École de Technologie Supérieure
1100 Notre-Dame West , Office A-3482
Montréal, Québec, Canada, H3C 1K3
(514) 396-8682

4. Sélection et utilisation d'un modèle compatible

Lorsqu'une évaluation est effectuée, les pratiques observées dans l'unité organisationnelle sont comparées à celles définies dans un modèle de bonnes pratiques pour déterminer la performance de leurs résultats via le niveau de leurs attributs de capacité. Pour cela, le modèle choisi, qui est conforme au modèle de référence, doit contenir des descriptions de ces pratiques et des indicateurs de performance de tel sorte que les jugements sur la capacité peuvent être fiables et cohérents.
Un des critères de sélection d'un modèle est qu'il soit compatible avec le modèle de référence définis à la partie deux de [ISO/IEC, 1998]. La compatibilité est essentielle pour fournir un bon degré de comparabilité entre les résultats des différentes évaluations en les rendant uniformes. Le modèle devra aussi être basé sur une bonne ingénierie logicielle, sur de bons principes de management de processus et être adapté à l'évaluation de la capacité de processus logiciel. Les modèles n'ayant pas été spécifiquement développés à cette fin peuvent donner des résultats non fiables. Leur aptitude à répondre à ces attentes doit donc être validée avant leur sélection.
Un autre critère est le « mapping » :

Un modèle doit fournir un mapping explicite des éléments fondamentaux du modèle vers les processus et attributs de processus du modèle de référence.
Le mapping devra être complet, clair, non-ambigu et devra justifier la déclaration de couverture de l'étendue de l'évaluation. [ISO/IEC TR 15504-2, 7.5]

Il est essentiel que l'évaluateur ait accès aux détails du mapping.
Citons un dernier critère de sélection avec la « traduction » :

Un modèle compatible doit fournir un mécanisme formel et vérifiable pour convertir les données obtenues à partir du modèle compatible vers les cotations des attributs de processus et ce, pour chaque processus, directement ou indirectement évalué, du modèle de référence. [ISO/IEC TR 15504-2, 7.5]

Le résultat d'une évaluation d'un processus est un ensemble de profiles du processus. Un profil de processus est un ensemble de neuf cotations, une pour chaque attribut du processus. Les résultats d'évaluation de n'importe quel modèle compatible doivent pouvoir être converti dans cette forme, de sorte qu'il existe une base commune pour la comparaison. Ajoutons enfin que le mécanisme de traduction peut être manuel ou informatisé. Si un modèle fournit explicitement les résultats dans le format prescrit dans ISO/IEC TR 15504-2, un mécanisme de traduction n'est alors pas nécessaire.

En conclusion, les considérations principales dans la sélection d'un modèle, en partant du fait que celui-ci est compatible avec le modèle de référence, sont ses aptitudes face au contexte de l'évaluation. Quand une organisation souhaite conduire une évaluation dans un domaine qui n'est pas représentatif de son domaine normal, il est d'autant plus important que le modèle choisis soit adapté. Un modèle fournit les définitions de base des processus et des attributs de processus qui sont des points de référence pour les jugements sur la performance des processus. L'utilisation d'un seul modèle pendant l'évaluation est donc essentielle. Il s'ensuit par conséquent qu'un évaluateur compétent doit très bien connaître les spécificités du modèle utilisé ; sa structure, ses éléments de base et sa relation envers le modèle de référence.

5. Utilisation des indicateurs

L'utilisation d'indicateurs appropriés est un composant clé des exigences de l'ISO 15504. Les indicateurs de performance ou de capacité de processus forment une base objective à partir de laquelle les jugements pour la cotation des attributs de processus peuvent être fondés. C'est cette base objective pour le jugement qui est le fondement de toute comparaison de résultats d'évaluation. En effet, les résultats ont la forme d'un ensemble de profiles de processus illustrant les cotations de chacun des neufs attributs de processus pour chaque processus évalué mais ils ne montrent pas pourquoi une cotation particulière a été donnée. Les indicateurs aident à identifier ce qui est présent ou absent d'un processus ou d'un produit et ils fournissent un guide à l'évaluateur lorsque celui-ci assigne une cotation à un processus ou à un attribut.
Les exigences sur les indicateurs sont :

- un modèle compatible doit contenir un ensemble compréhensible d'indicateurs couvrant les deux dimensions du modèle de référence ;
- les indicateurs doivent être utilisés pendant l'évaluation en support au jugement des évaluateurs concernant la cotation des attributs de processus ;
- la preuve basée sur les indicateurs doit être enregistrée et maintenue.

Il y a donc deux types d'indicateurs : de performance des processus et de capacité des processus. Les indicateurs de performance permettent de guider l'évaluateur en lui indiquant comment juger si un processus rencontre son objectif comme il est défini dans le modèle de référence. Ces indicateurs sont des pratiques exécutées au sein d'un processus spécifique. La performance des pratiques pertinentes fournit une première approximation sur la rencontre de l'objectif d'un processus. La seconde indication est l'existence de produits à partir de l'exécution des pratiques. L'exécution apparente de la pratique ne fournit pas à elle seule la preuve d'une implémentation suffisante. La preuve supplémentaire que l'exécution de la pratique rencontre l'objectif du processus est obtenue à partir de l'existence de produits appropriés et de caractéristiques de produits appropriées.
Les indicateurs de capacité sont associés à chaque attribut de processus des niveaux de capacité deux à cinq. De manière similaire aux indicateurs de performance des processus, ils aident les évaluateurs à juger la réalisation de la capacité décrite par les attributs de processus. Ils aident aussi à identifier la capacité de l'organisation à gérer un processus efficacement.
La figure 9.1 illustre comment les indicateurs de performance et de capacité sont combinés pour supporter la cotation des neufs attributs de processus au sein des niveaux de capacité un à cinq.

Les indicateurs de performance permettent de donner une cotation au seul attribut du niveau de capacité 1, lequel exprime jusqu'à quel point les pratiques et produits du processus rencontrent l'objectif du processus.
Les indicateurs de capacité permettent quant à eux de donner une cotation aux huit autres attributs du processus appartenant aux niveaux de capacité deux à cinq, lesquels mesurent les aspects de la capacité de gestion du processus.

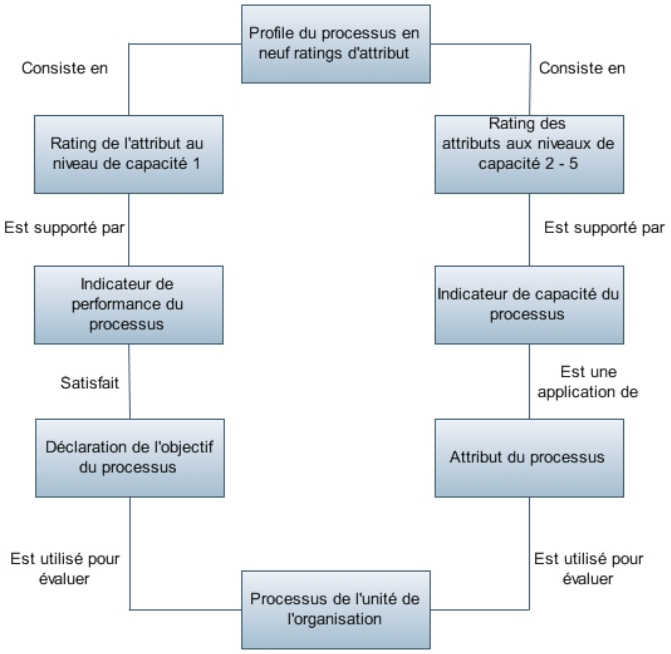
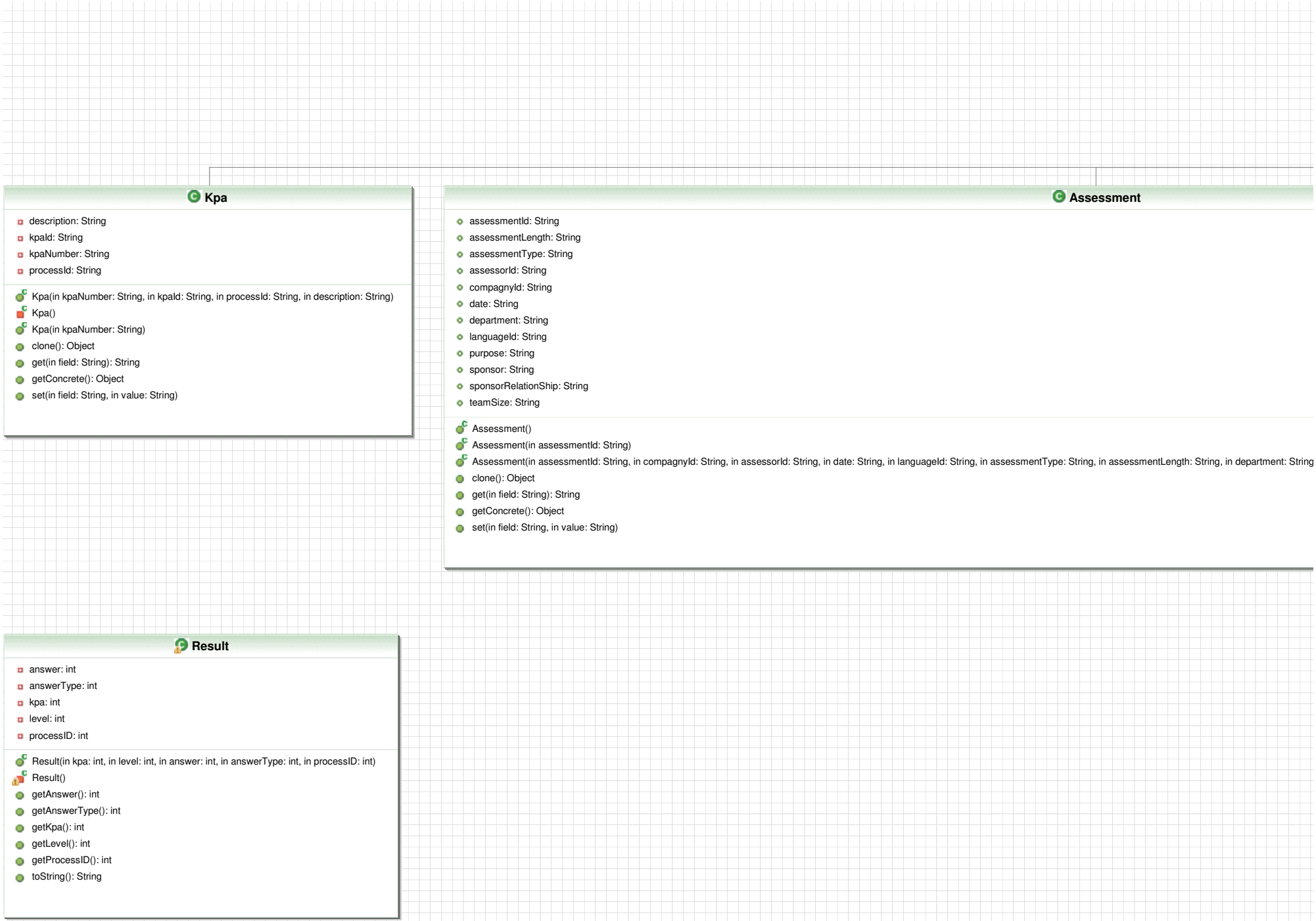
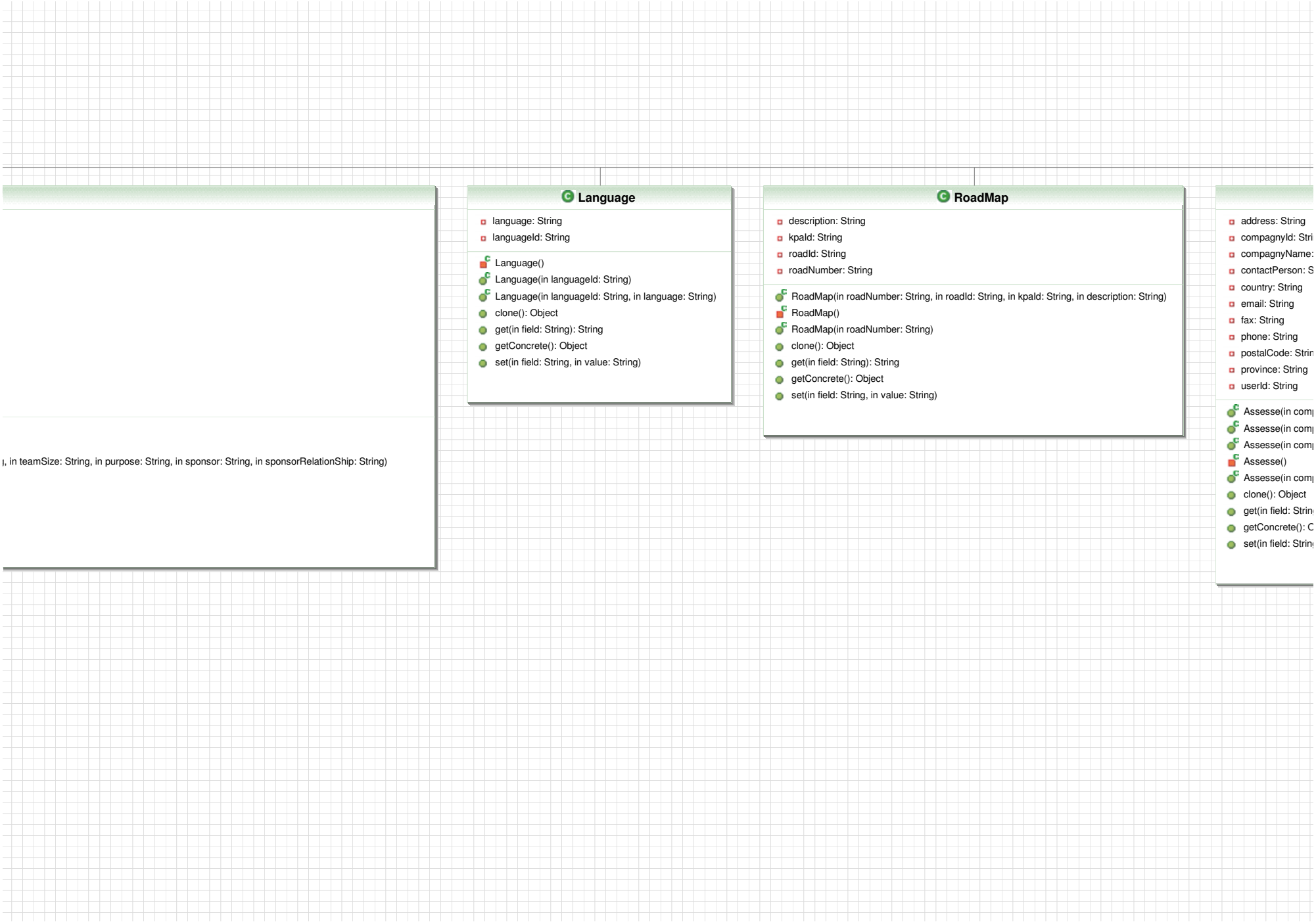


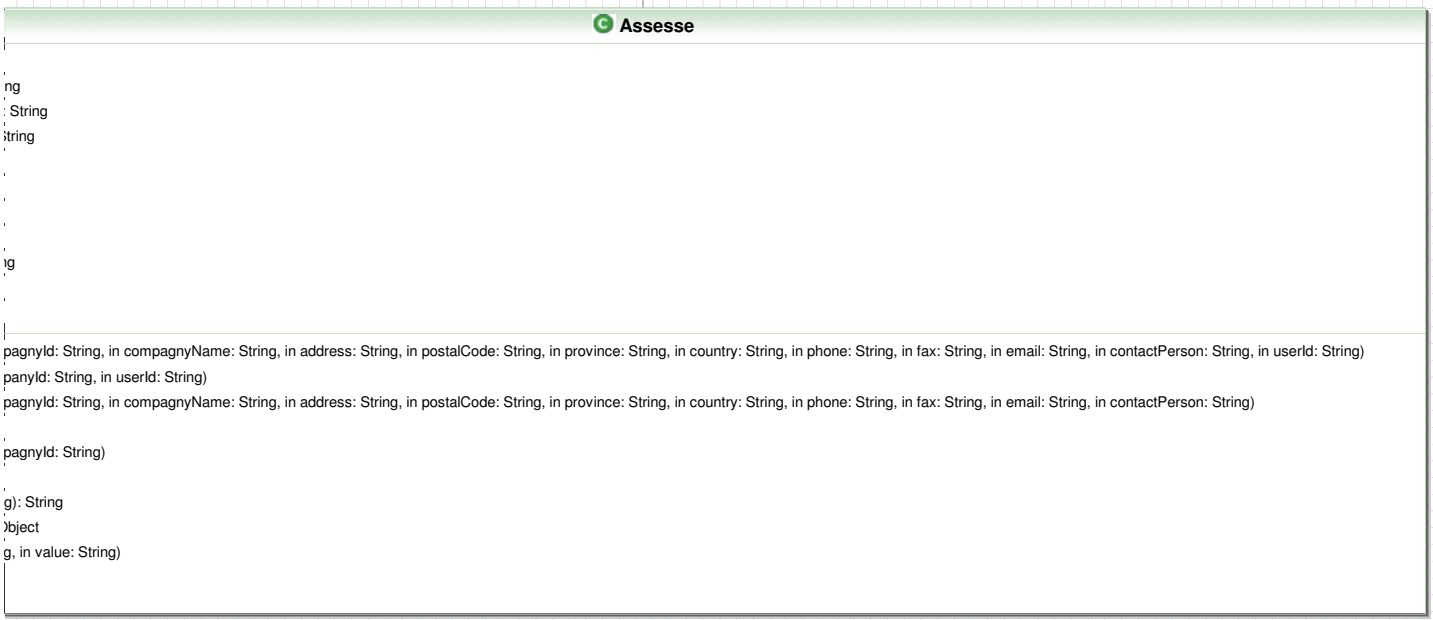
FIG. 9.1 – Détermination des cotations (*adapté de [ISO/IEC, 1998]*)

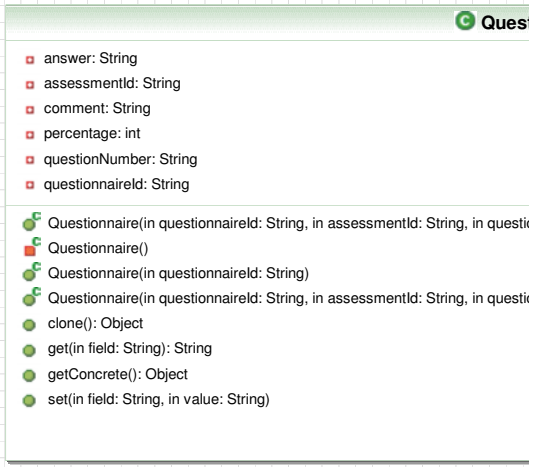
6. Diagramme de classes détaillé de $\mathcal{S}^{3}mAssess_{®}$

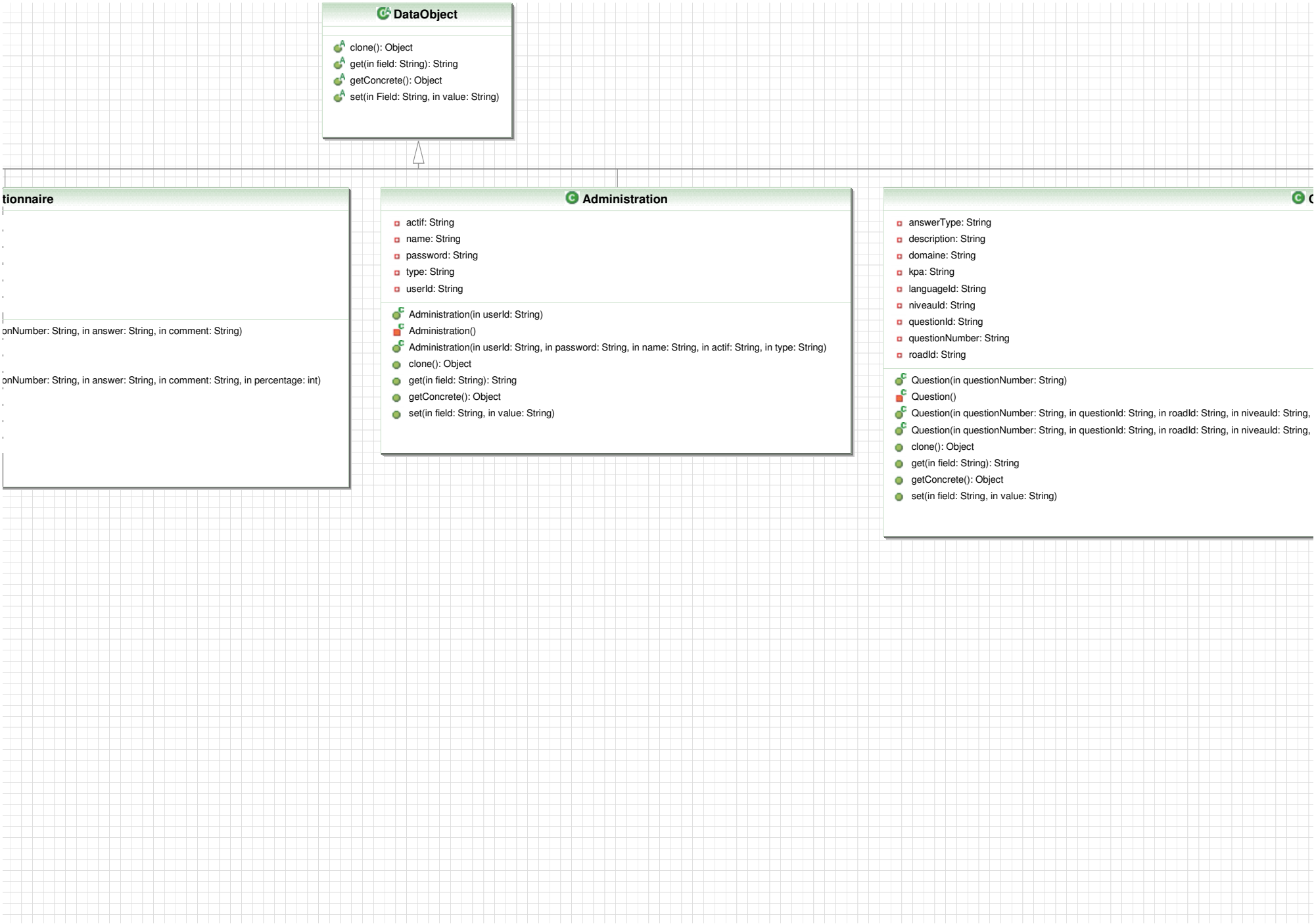

Main
start()

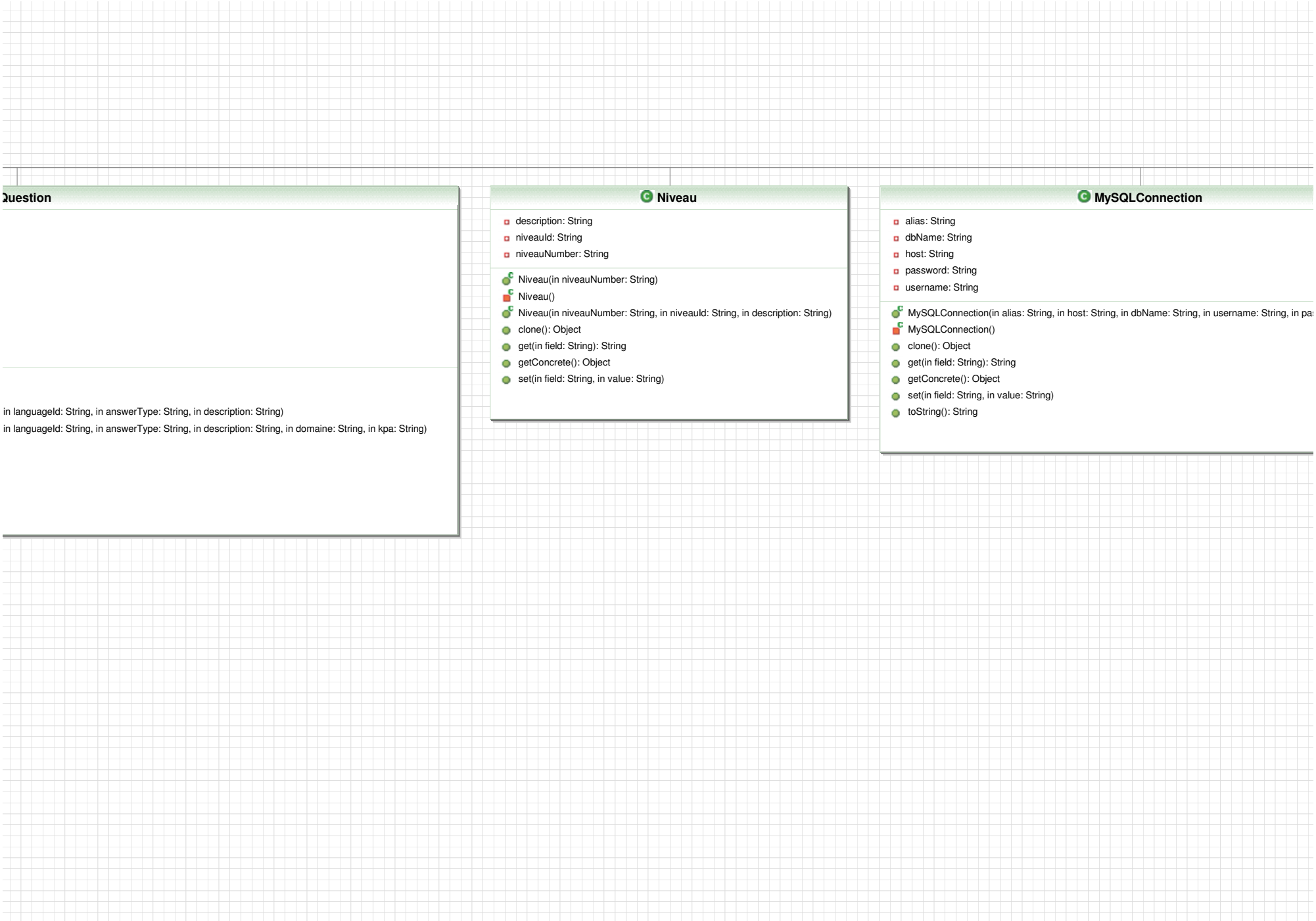

Kpa
description: String
kpaId: String
kpaNumber: String
processId: String
Kpa(in kpaNumber: String, in kpaId: String, in processId: String, in description: String)
Kpa()
Kpa(in kpaNumber: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
Assessment
assessmentId: String
assessmentLength: String
assessmentType: String
assessorId: String
compagnyId: String
date: String
department: String
languageId: String
purpose: String
sponsor: String
sponsorRelationShip: String
teamSize: String
Assessment()
Assessment(in assessmentId: String)
Assessment(in assessmentId: String, in compagnyId: String, in assessorId: String, in date: String, in languageId: String, in assessmentType: String, in assessmentLength: String, in department: String
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
Result
answer: int
answerType: int
kpa: int
level: int
processID: int
Result(in kpa: int, in level: int, in answer: int, in answerType: int, in processID: int)
Result()
getAnswer(): int
getAnswerType(): int
getKpa(): int
getLevel(): int
getProcessID(): int
toString(): String

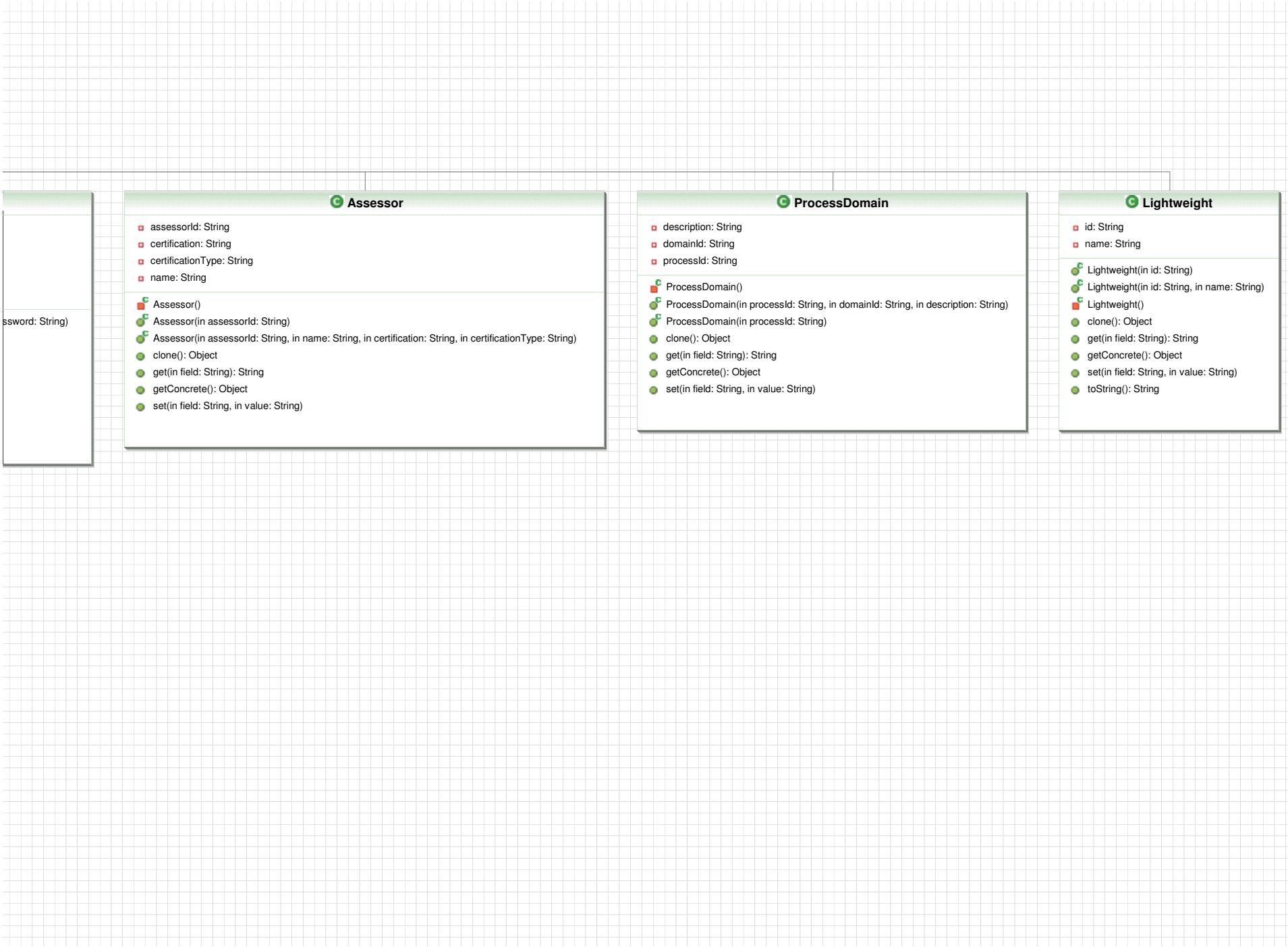

, in teamSize: String, in purpose: String, in sponsor: String, in sponsorRelationShip: String)
Language
language: String
languageId: String
Language()
Language(in languageId: String)
Language(in languageId: String, in language: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
RoadMap
description: String
kpaId: String
roadId: String
roadNumber: String
RoadMap(in roadNumber: String, in roadId: String, in kpaId: String, in description: String)
RoadMap()
RoadMap(in roadNumber: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
address: String
compagnyId: Stri
compagnyName:
contactPerson: S
country: String
email: String
fax: String
phone: String
postalCode: Strin
province: String
userId: String
Assesse(in comp
Assesse(in comp
Assesse(in comp
Assesse()
Assesse(in comp
clone(): Object
get(in field: Strin
getConcrete(): C
set(in field: Strin


Assesse
ng
String
tring
g
pagnyId: String, in compagnyName: String, in address: String, in postalCode: String, in province: String, in country: String, in phone: String, in fax: String, in email: String, in contactPerson: String, in userId: String)
panyId: String, in userId: String)
pagnyId: String, in compagnyName: String, in address: String, in postalCode: String, in province: String, in country: String, in phone: String, in fax: String, in email: String, in contactPerson: String)
pagnyId: String)
g): String
bject
g, in value: String)

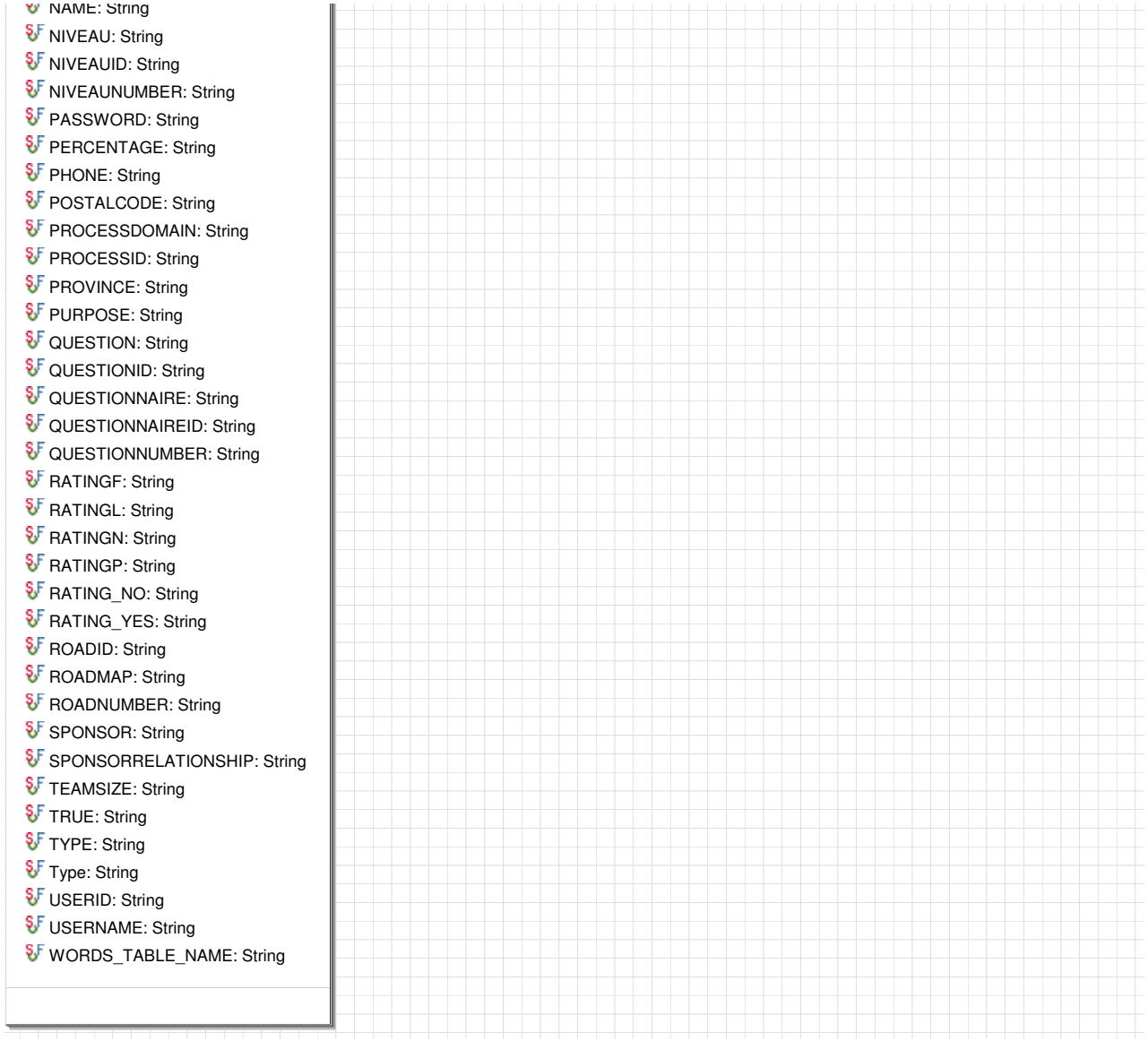

Ques
answer: String
assessmentId: String
comment: String
percentage: int
questionNumber: String
questionnaireId: String
Questionnaire(in questionnaireId: String, in assessmentId: String, in questi
Questionnaire()
Questionnaire(in questionnaireId: String)
Questionnaire(in questionnaireId: String, in assessmentId: String, in questi
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)

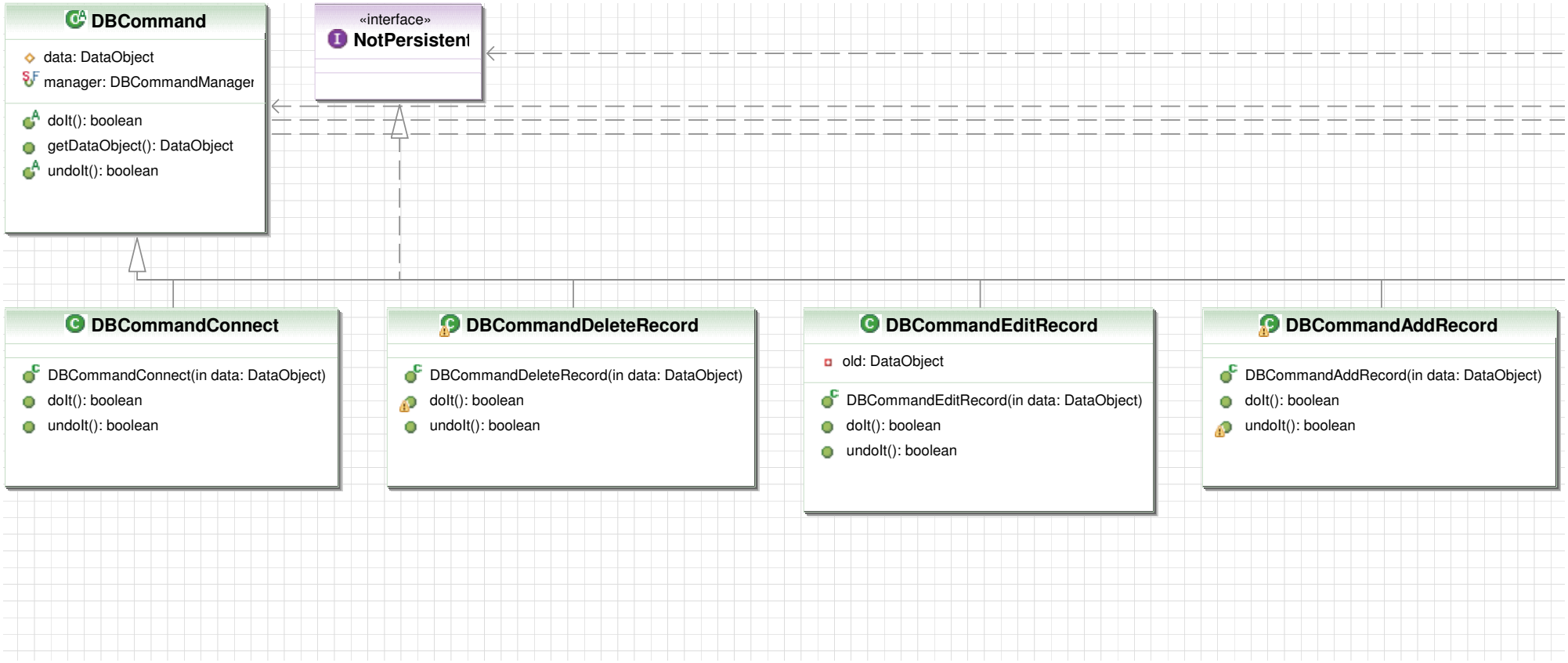

DataObject
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in Field: String, in value: String)
tionnaire
onNumber: String, in answer: String, in comment: String)
onNumber: String, in answer: String, in comment: String, in percentage: int)
Administration
actif: String
name: String
password: String
type: String
userId: String
Administration(in userId: String)
Administration()
Administration(in userId: String, in password: String, in name: String, in actif: String, in type: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
answerType: String
description: String
domaine: String
kpa: String
languageId: String
niveauId: String
questionId: String
questionNumber: String
roadId: String
Question(in questionNumber: String)
Question()
Question(in questionNumber: String, in questionId: String, in roadId: String, in niveauId: String,
Question(in questionNumber: String, in questionId: String, in roadId: String, in niveauId: String,
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)

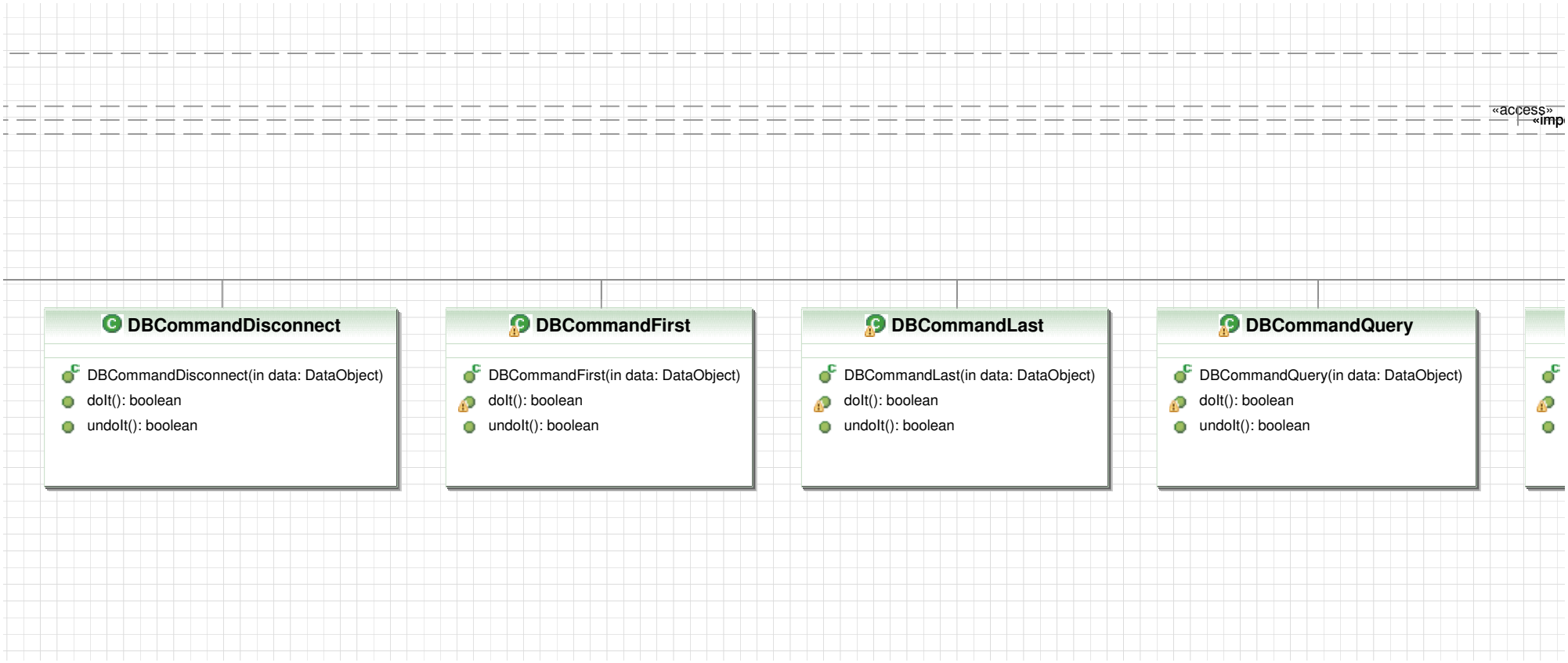

Question
in languageId: String, in answerType: String, in description: String)
in languageId: String, in answerType: String, in description: String, in domaine: String, in kpa: String)
Niveau
description: String
niveauId: String
niveauNumber: String
Niveau(in niveauNumber: String)
Niveau()
Niveau(in niveauNumber: String, in niveauId: String, in description: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
MySQLConnection
alias: String
dbName: String
host: String
password: String
username: String
MySQLConnection(in alias: String, in host: String, in dbName: String, in username: String, in pa
MySQLConnection()
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
toString(): String

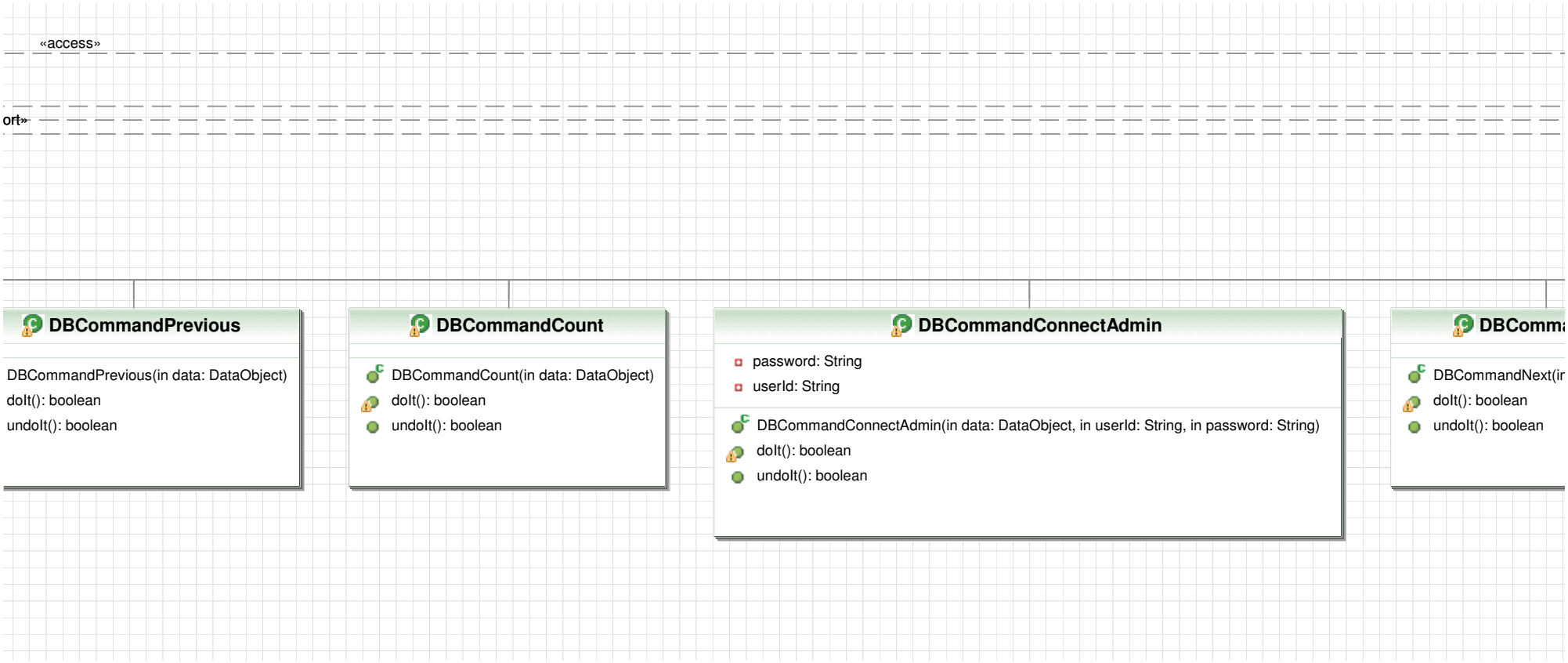

ssword: String)
Assessor
assessorId: String
certification: String
certificationType: String
name: String
Assessor()
Assessor(in assessorId: String)
Assessor(in assessorId: String, in name: String, in certification: String, in certificationType: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
ProcessDomain
description: String
domainId: String
processId: String
ProcessDomain()
ProcessDomain(in processId: String, in domainId: String, in description: String)
ProcessDomain(in processId: String)
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
Lightweight
id: String
name: String
Lightweight(in id: String)
Lightweight(in id: String, in name: String)
Lightweight()
clone(): Object
get(in field: String): String
getConcrete(): Object
set(in field: String, in value: String)
toString(): String

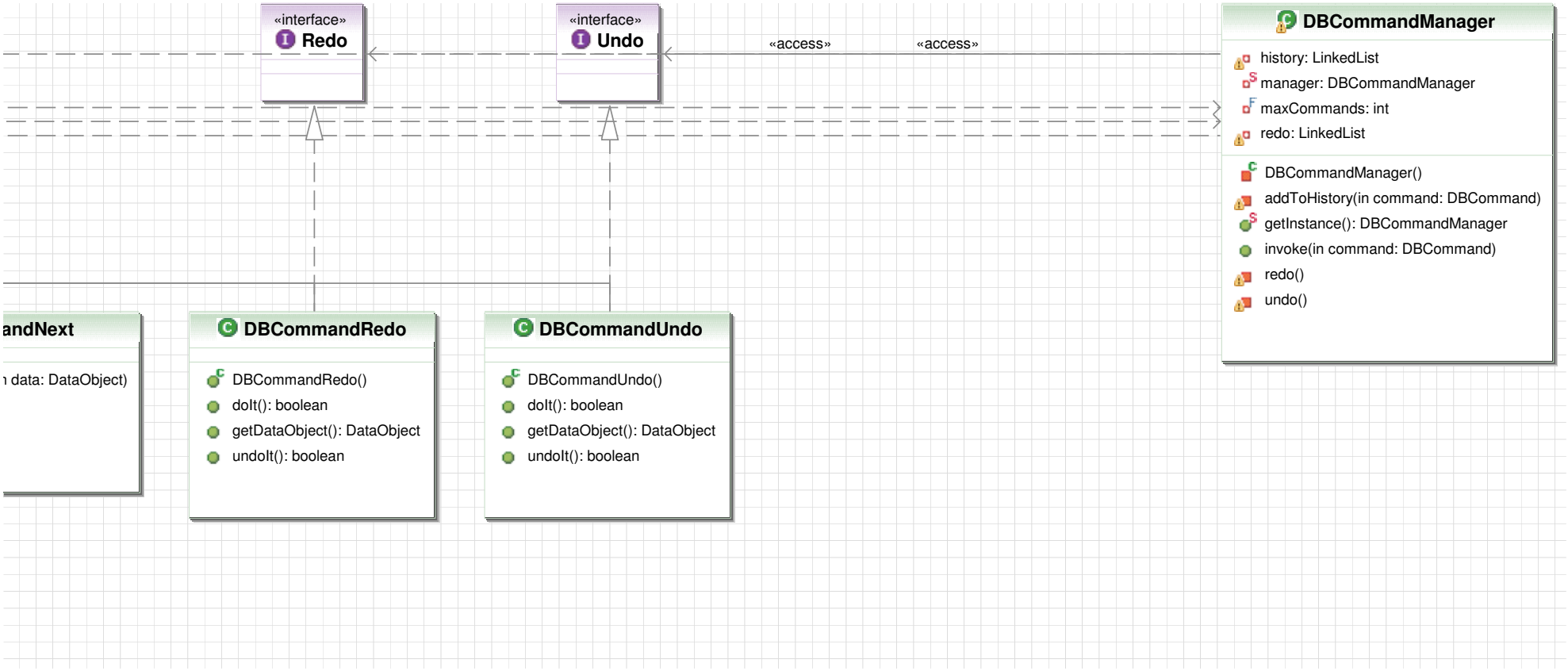

DBConstants
ACTIF: String
ADDRESS: String
ADMINISTRATION: String
ALIAS: String
ANALYZESHOW: String
ANSWER: String
ANSWERTYPE: String
ASSESSE: String
ASSESSMENT: String
ASSESSMENTID: String
ASSESSMENTLANGUAGE: String
ASSESSMENTLENGTH: String
ASSESSMENTTYPE: String
ASSESSOR: String
ASSESSORID: String
CERTIFICATION: String
CERTIFICATIONTYPE: String
COMMENT: String
COMPAGNYID: String
COMPAGNYNAME: String
CONTACTPERSON: String
COUNTRY: String
DATE: String
DBNAME: String
DEPARTMENT: String
DESCRIPTION: String
DOMAIN: String
DOMAINE: String
DOMAINID: String
EMAIL: String
EMPTY: String
FALSE: String
FAX: String
HOST: String
ID: String
KEYWORDS_FIELD_NAME: String
KPA: String
KPAID: String
KPANUMBER: String
LANGUAGE: String
LANGUAGEID: String
LANGUAGES_TABLE_NAME: Strin
LANGUAGE_FIELD_NAME: String


NAME: String
NIVEAU: String
NIVEAUID: String
NIVEAUNUMBER: String
PASSWORD: String
PERCENTAGE: String
PHONE: String
POSTALCODE: String
PROCESSDOMAIN: String
PROCESSID: String
PROVINCE: String
PURPOSE: String
QUESTION: String
QUESTIONID: String
QUESTIONNAIRE: String
QUESTIONNAIREID: String
QUESTIONNUMBER: String
RATINGF: String
RATINGL: String
RATINGN: String
RATINGP: String
RATING_NO: String
RATING_YES: String
ROADID: String
ROADMAP: String
ROADNUMBER: String
SPONSOR: String
SPONSORRELATIONSHIP: String
TEAMSIZE: String
TRUE: String
TYPE: String
Type: String
USERID: String
USERNAME: String
WORDS_TABLE_NAME: String

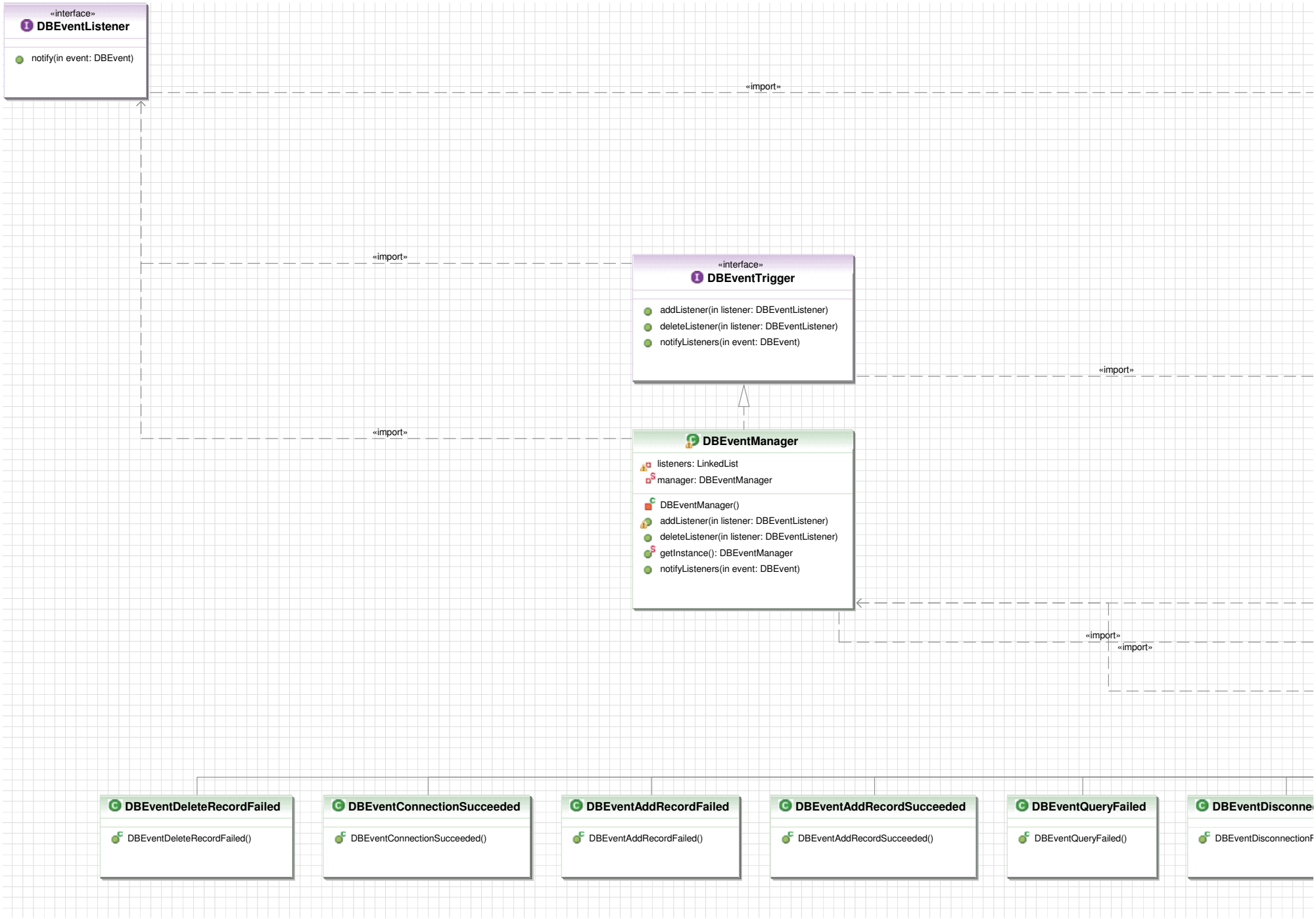

DBCommand
data: DataObject
manager: DBCommandManager
doIt(): boolean
getDataObject(): DataObject
undoIt(): boolean
«interface»
NotPersistent
DBCommandConnect
DBCommandConnect(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandDeleteRecord
DBCommandDeleteRecord(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandEditRecord
old: DataObject
DBCommandEditRecord(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandAddRecord
DBCommandAddRecord(in data: DataObject)
doIt(): boolean
undoIt(): boolean

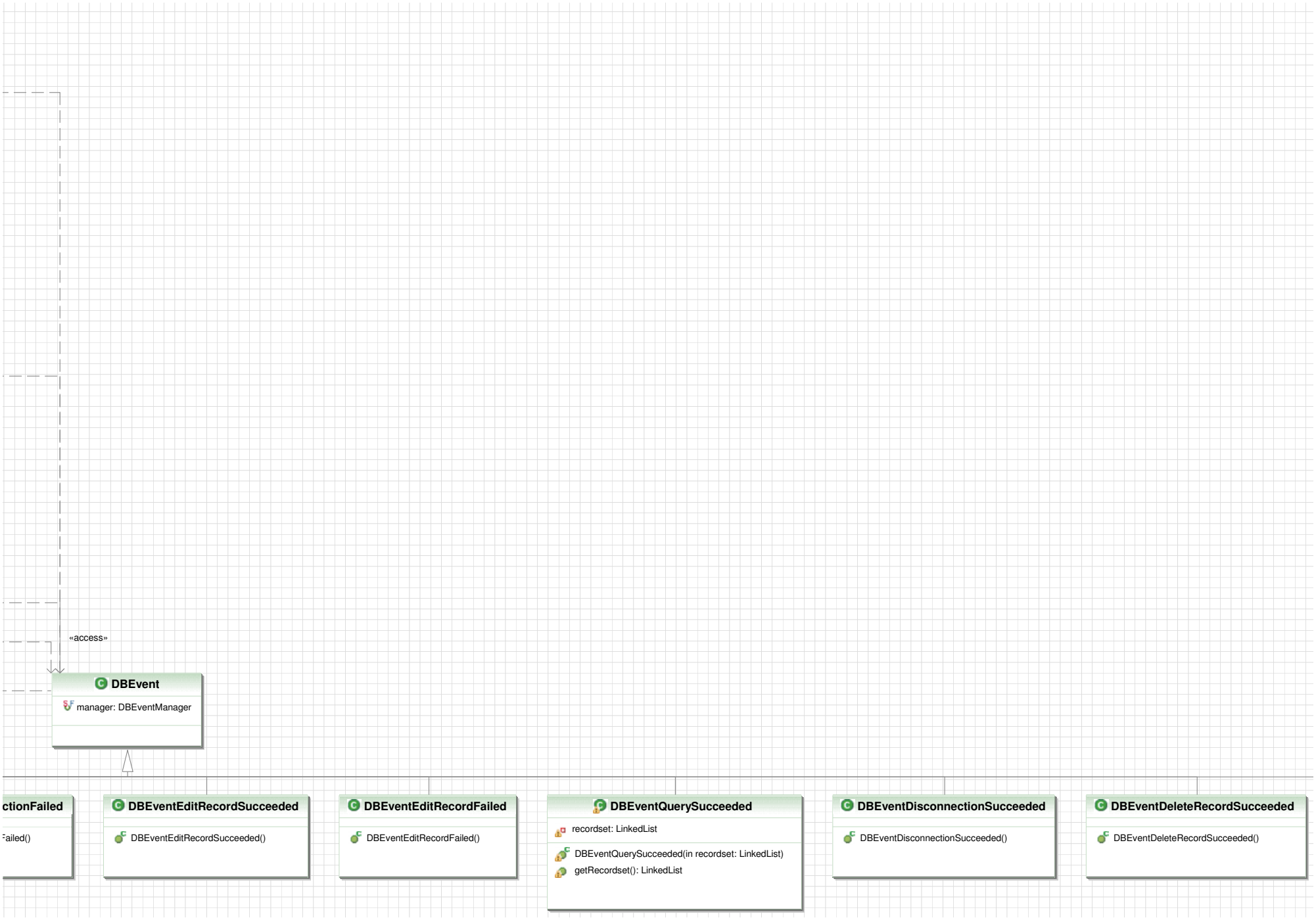

«access»
«imp
DBCommandDisconnect
DBCommandDisconnect(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandFirst
DBCommandFirst(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandLast
DBCommandLast(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandQuery
DBCommandQuery(in data: DataObject)
doIt(): boolean
undoIt(): boolean


«access»
ort»
DBCommandPrevious
DBCommandPrevious(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandCount
DBCommandCount(in data: DataObject)
doIt(): boolean
undoIt(): boolean
DBCommandConnectAdmin
password: String
userId: String
DBCommandConnectAdmin(in data: DataObject, in userId: String, in password: String)
doIt(): boolean
undoIt(): boolean
DBComma
DBCommandNext(ir
doIt(): boolean
undoIt(): boolean

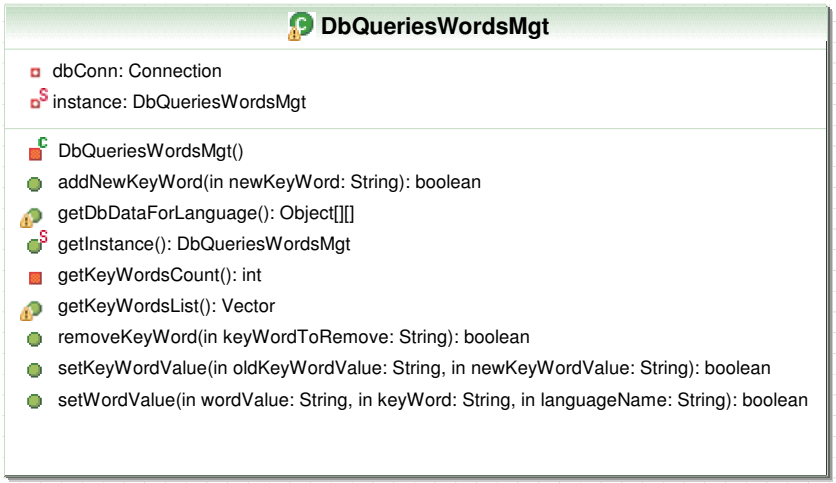

«interface»
Redo
«interface»
Undo
«access»
«access»
DBCommandManager
history: LinkedList
manager: DBCommandManager
maxCommands: int
redo: LinkedList
DBCommandManager()
addToHistory(in command: DBCommand)
getInstance(): DBCommandManager
invoke(in command: DBCommand)
redo()
undo()
andNext
data: DataObject)
DBCommandRedo
DBCommandRedo()
doIt(): boolean
getDataObject(): DataObject
undoIt(): boolean
DBCommandUndo
DBCommandUndo()
doIt(): boolean
getDataObject(): DataObject
undoIt(): boolean

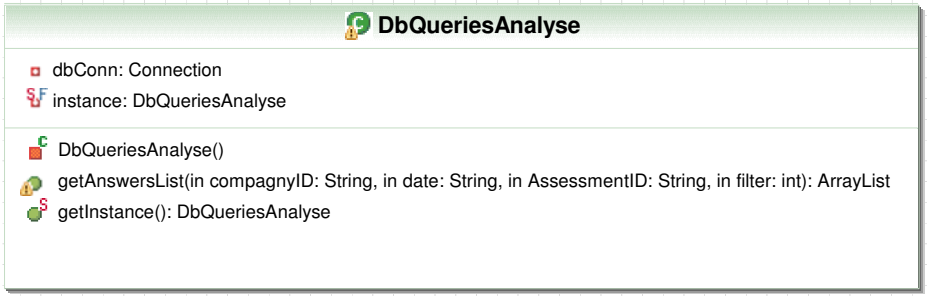

DBConnectionManager
dbConnection: DBConnection
manager: DBConnectionManager
DBConnectionManager()
connect(in data: DataObject): boolean
disconnect(in data: DataObject): boolean
getAppropriateDBConnection(in data: DataObject): DBConnection
getDBConnection(): DBConnection
getInstance(): DBConnectionManager
«import»
«access»
DBConnection
conn: Connection
dbName: String
host: String
password: String
username: String
connect(in host: String, in dbName: String, in username: String, in password: String): boolean
disconnect(): boolean
getConnection(): Connection
getDbName(): String
getHost(): String
getPassword(): String
getUsername(): String
loadDriver(): boolean
«import»
DBConnectionMySQL
driver: DBConnectionMySQL
DBConnectionMySQL()
connect(in host: String, in dbName: String, in username: String, in password: String): boolean
disconnect(): boolean
getInstance(): DBConnection
loadDriver(): boolean
DBConnectionConstants
APPLET_CLOSE_URL: String
APPLET_CONTEXT: AppletContext
DB_DRIVER: String
DB_LOGIN: String
DB_LOGIN_LOCAL: String
DB_PASSWORD: String
DB_PASSWORD_LOCAL: String
DB_SERVER_PATH: String
DB_SERVER_PATH_LOCAL: Strin
ISADMINCONNECTED: boolean
ISLOCALRUN: boolean
ISUSAGECONNECTED: boolean
DBConnectionConstants()

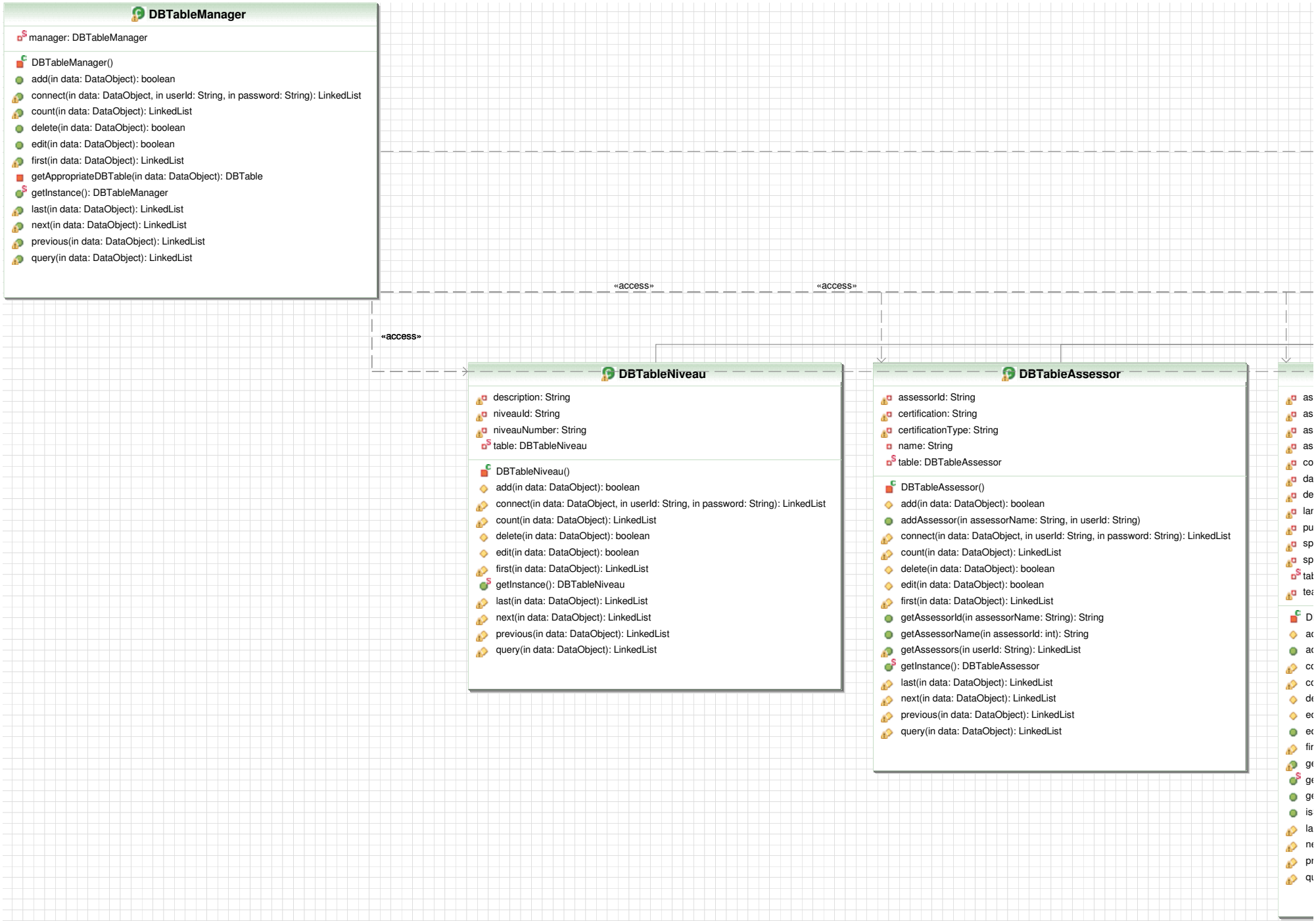

«interface»
DBEventListener
notify(in event: DBEvent)
«import»
«import»
«interface»
DBEventTrigger
addListener(in listener: DBEventListener)
deleteListener(in listener: DBEventListener)
notifyListeners(in event: DBEvent)
«import»
«import»
DBEventManager
listeners: LinkedList
manager: DBEventManager
DBEventManager()
addListener(in listener: DBEventListener)
deleteListener(in listener: DBEventListener)
getInstance(): DBEventManager
notifyListeners(in event: DBEvent)
«import»
«import»
DBEventDeleteRecordFailed
DBEventDeleteRecordFailed()
DBEventConnectionSucceeded
DBEventConnectionSucceeded()
DBEventAddRecordFailed
DBEventAddRecordFailed()
DBEventAddRecordSucceeded
DBEventAddRecordSucceeded()
DBEventQueryFailed
DBEventQueryFailed()

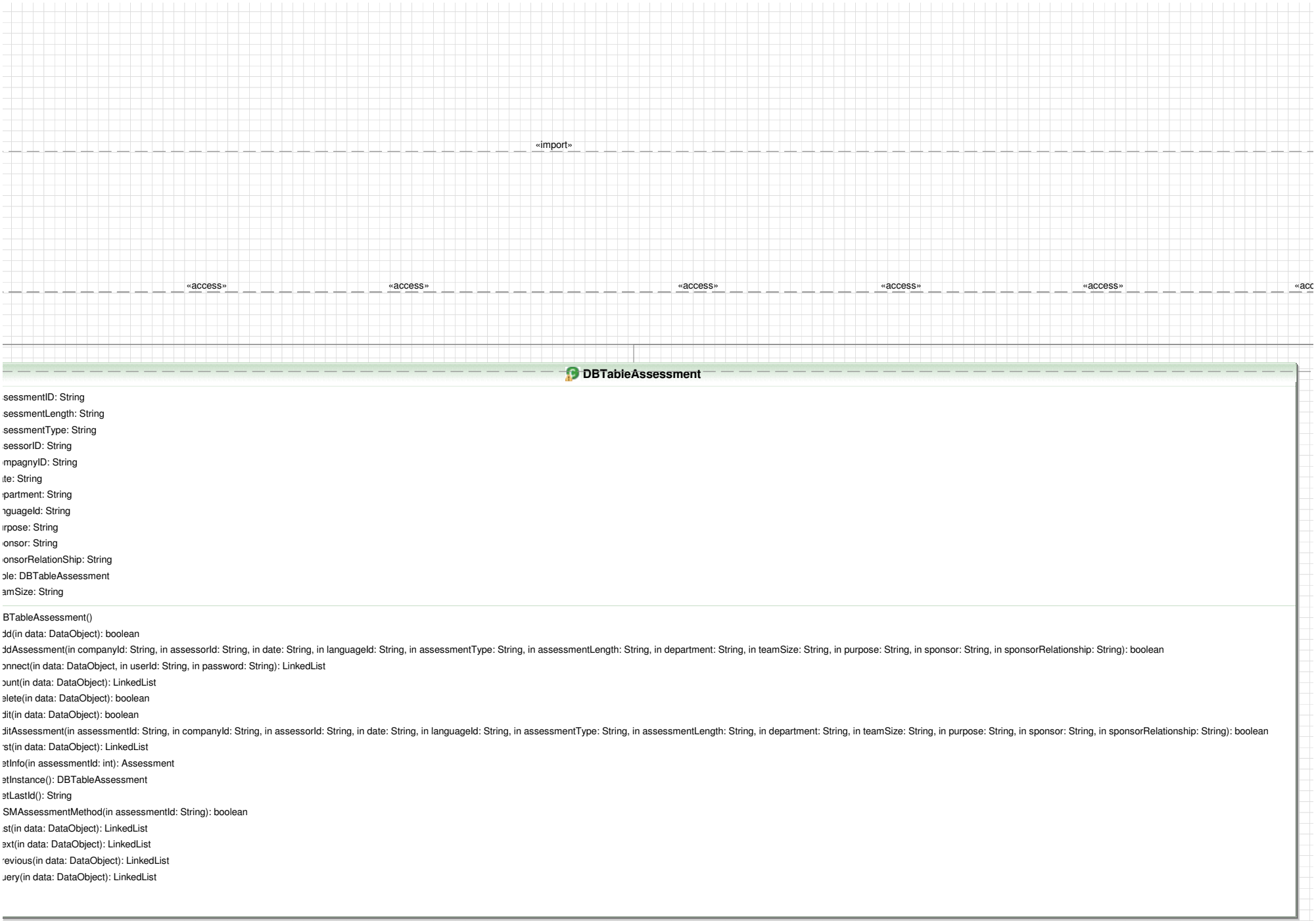

«access»
DBEvent
manager: DBEventManager
ctionFailed
ailed()
DBEventEditRecordSucceeded
DBEventEditRecordSucceeded()
DBEventEditRecordFailed
DBEventEditRecordFailed()
DBEventQuerySucceeded
recordset: LinkedList
DBEventQuerySucceeded(in recordset: LinkedList)
getRecordset(): LinkedList
DBEventDisconnectionSucceeded
DBEventDisconnectionSucceeded()
DBEventDeleteRecordSucceeded
DBEventDeleteRecordSucceeded()

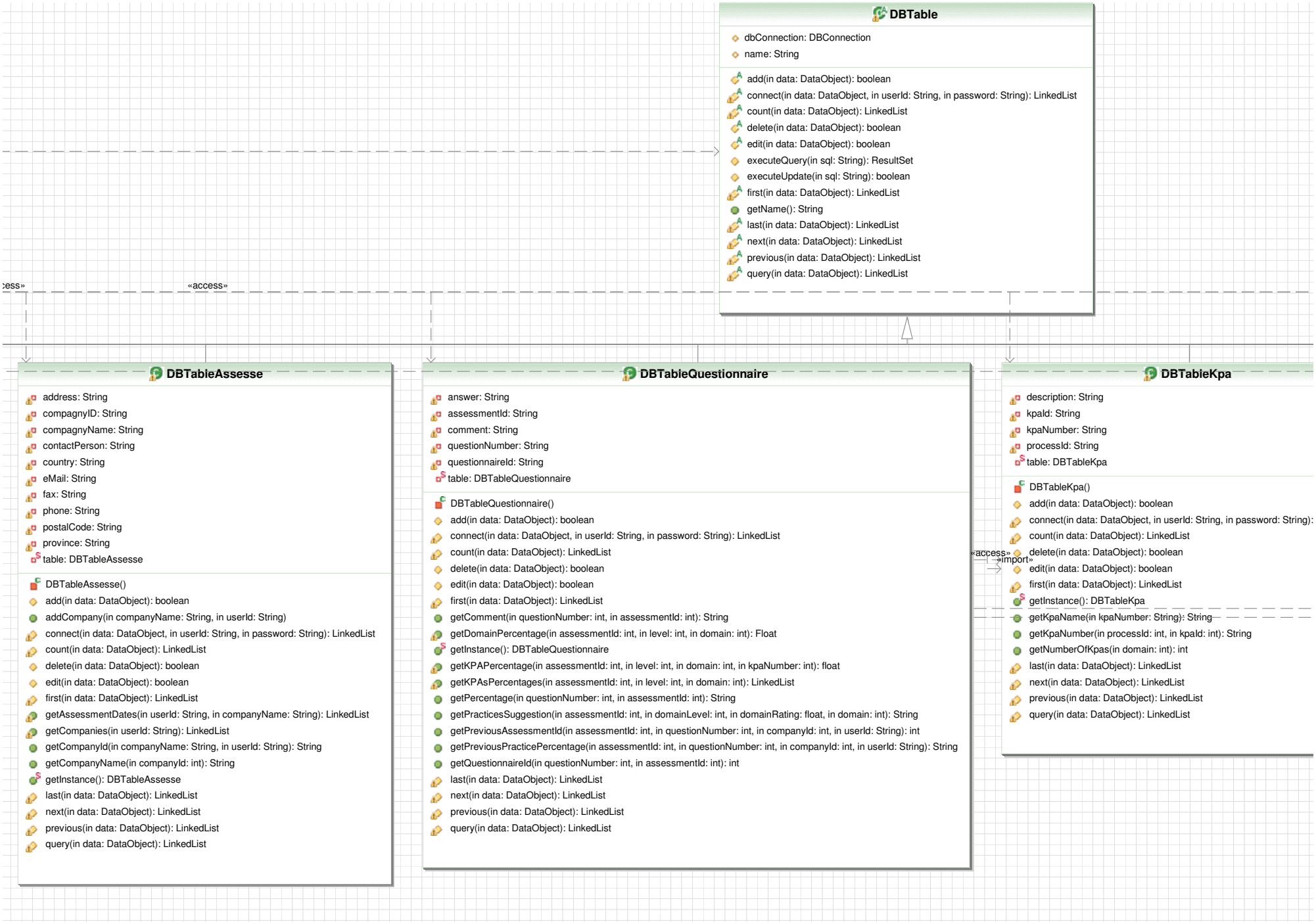
DBEventConnectionFailed
DBEventConnectionFailed()

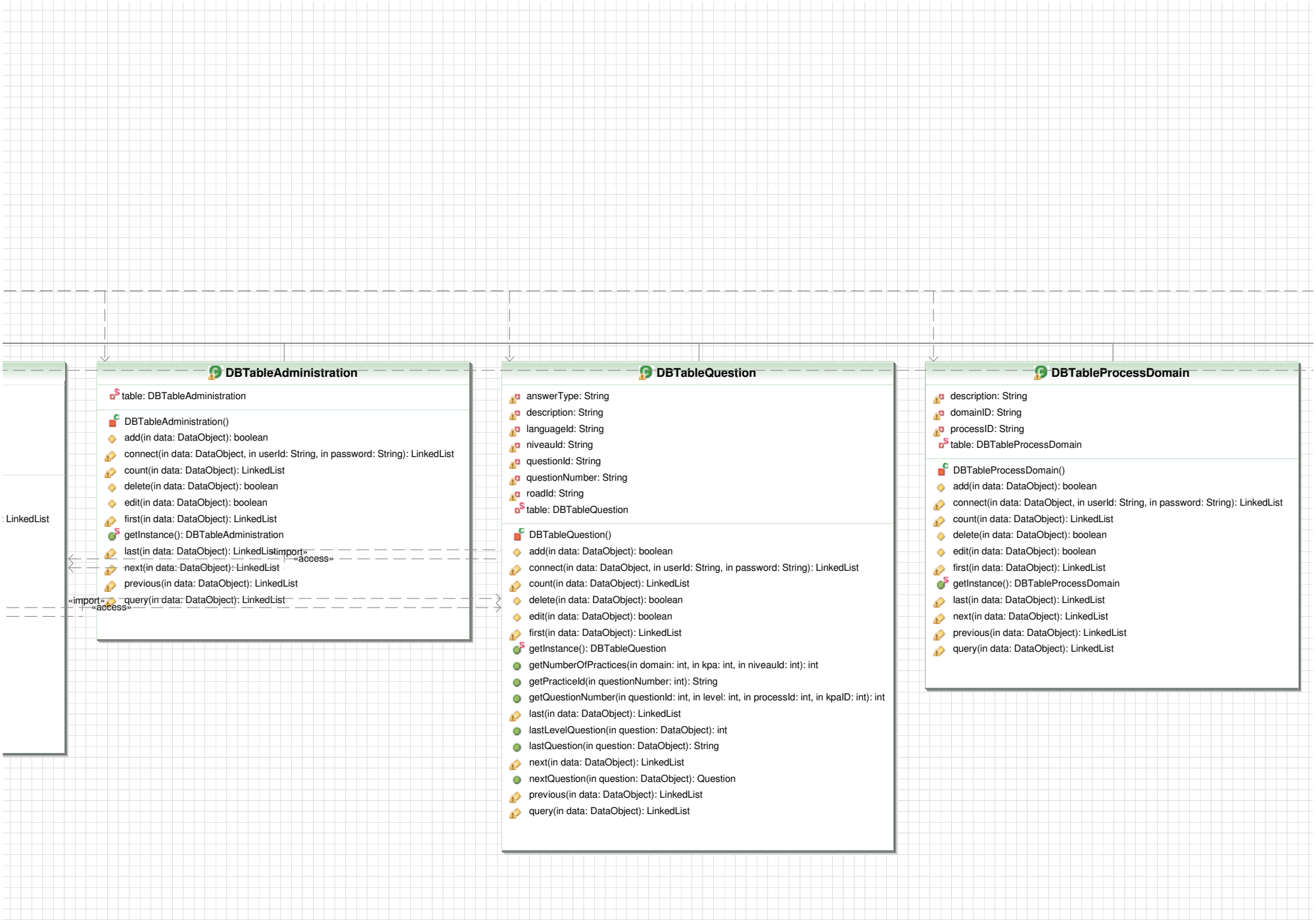

DbQueriesWordsMgt
dbConn: Connection
instance: DbQueriesWordsMgt
DbQueriesWordsMgt()
addNewKeyWord(in newKeyWord: String): boolean
getDbDataForLanguage(): Object[][]
getInstance(): DbQueriesWordsMgt
getKeyWordsCount(): int
getKeyWordsList(): Vector
removeKeyWord(in keyWordToRemove: String): boolean
setKeyWordValue(in oldKeyWordValue: String, in newKeyWordValue: String): boolean
setWordValue(in wordValue: String, in keyWord: String, in languageName: String): boolean


DbQueriesAnalyse
dbConn: Connection
instance: DbQueriesAnalyse
DbQueriesAnalyse()
getAnswersList(in compagnyID: String, in date: String, in AssessmentID: String, in filter: int): ArrayList
getInstance(): DbQueriesAnalyse

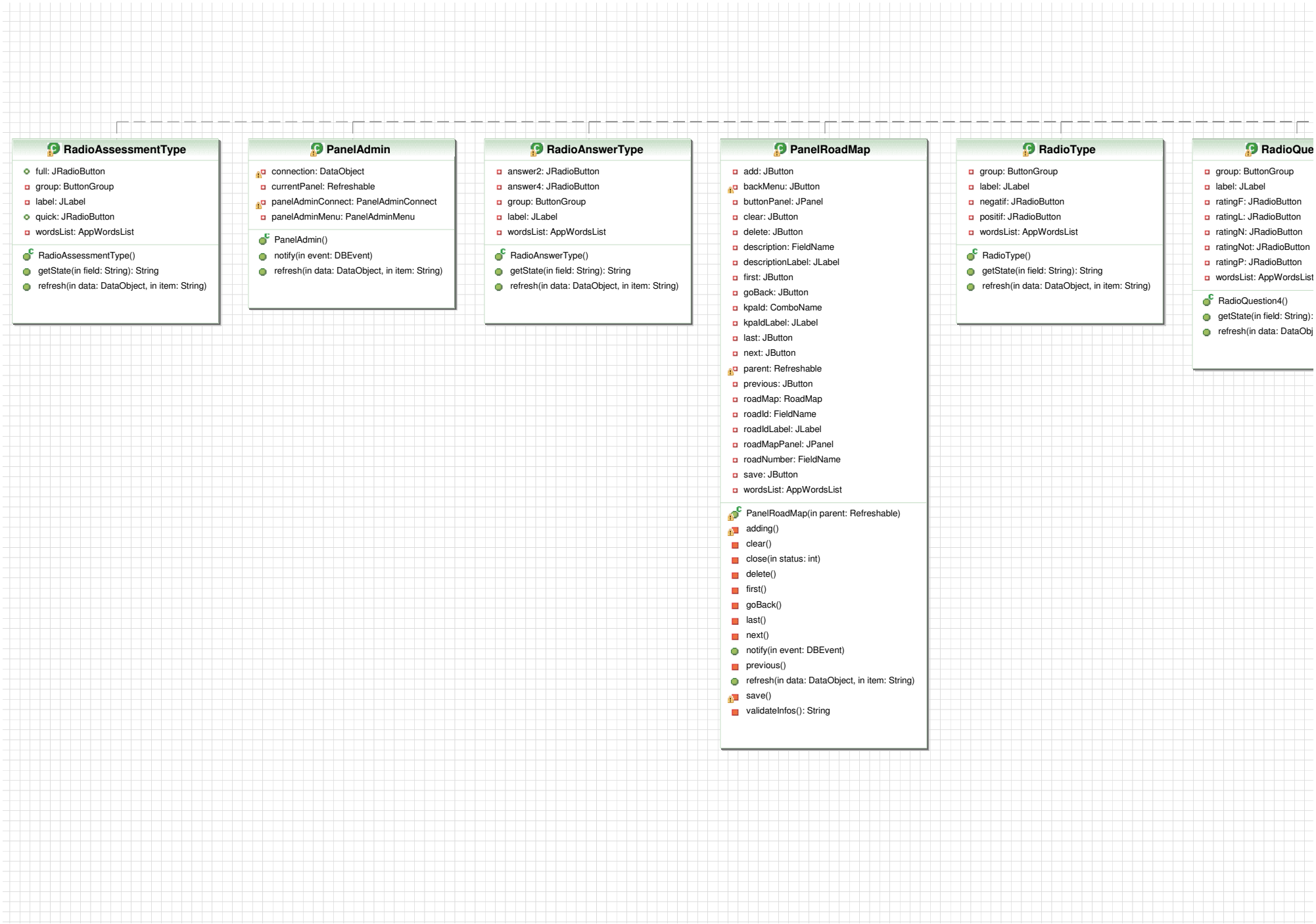

DBTableManager
manager: DBTableManager
DBTableManager()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getAppropriateDBTable(in data: DataObject): DBTable
getInstance(): DBTableManager
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
«access»
«access»
«access»
DBTableNiveau
description: String
niveauId: String
niveauNumber: String
table: DBTableNiveau
DBTableNiveau()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableNiveau
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
DBTableAssessor
assessorId: String
certification: String
certificationType: String
name: String
table: DBTableAssessor
DBTableAssessor()
add(in data: DataObject): boolean
addAssessor(in assessorName: String, in userId: String)
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getAssessorId(in assessorName: String): String
getAssessorName(in assessorId: int): String
getAssessors(in userId: String): LinkedList
getInstance(): DBTableAssessor
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList

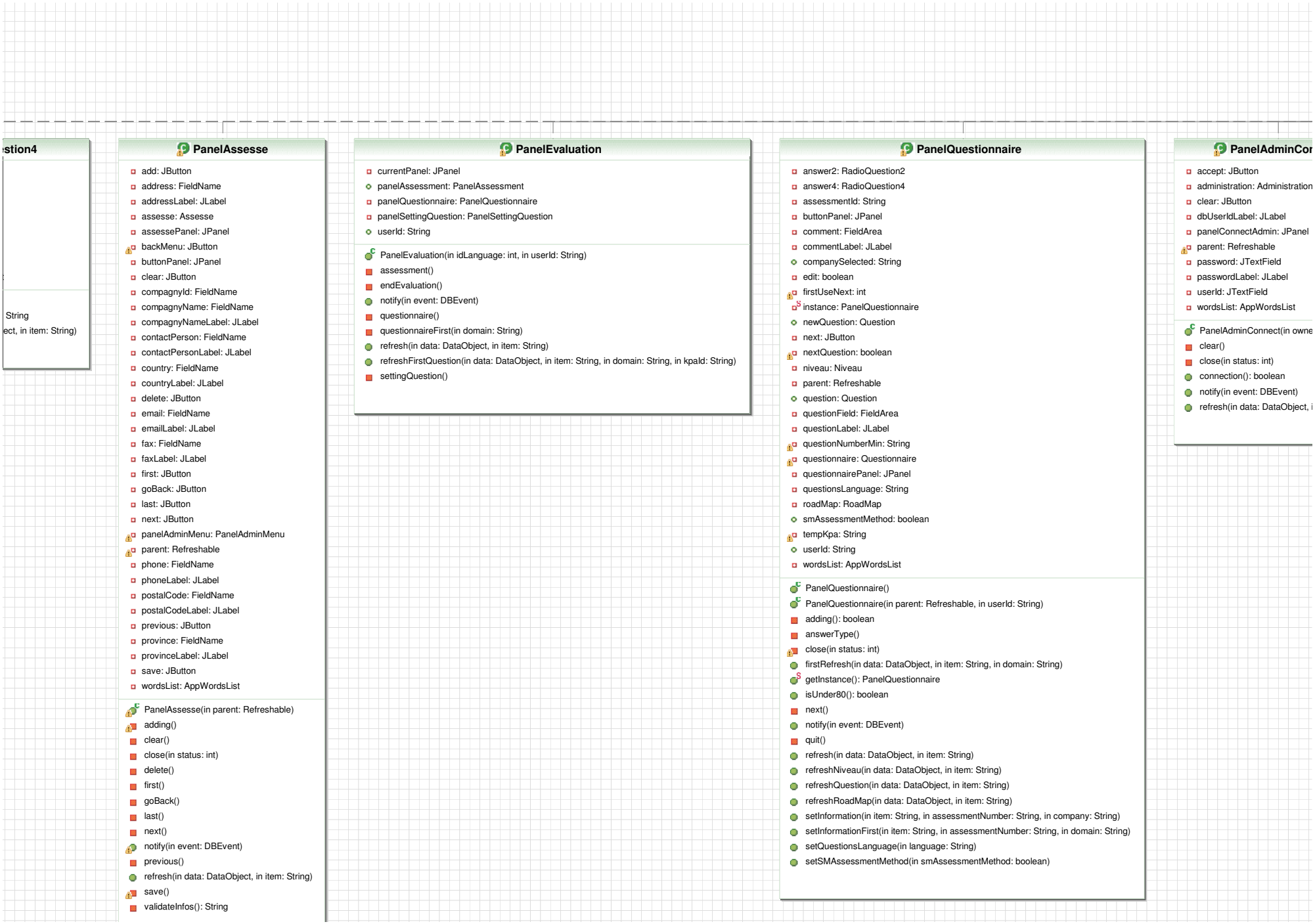

«import»
«access»
«access»
«access»
«access»
«access»
«acc
DBTableAssessment
sessmentID: String
sessmentLength: String
sessmentType: String
sessorID: String
mpagnyID: String
te: String
partment: String
guageId: String
rpose: String
onsor: String
onsorRelationShip: String
ole: DBTableAssessment
amSize: String
BTableAssessment()
dd(in data: DataObject): boolean
ddAssessment(in companyId: String, in assessorId: String, in date: String, in languageId: String, in assessmentType: String, in assessmentLength: String, in department: String, in teamSize: String, in purpose: String, in sponsor: String, in sponsorRelationship: String): boolean
onnect(in data: DataObject, in userId: String, in password: String): LinkedList
ount(in data: DataObject): LinkedList
elete(in data: DataObject): boolean
dit(in data: DataObject): boolean
ditAssessment(in assessmentId: String, in companyId: String, in assessorId: String, in date: String, in languageId: String, in assessmentType: String, in assessmentLength: String, in department: String, in teamSize: String, in purpose: String, in sponsor: String, in sponsorRelationship: String): boolean
st(in data: DataObject): LinkedList
etInfo(in assessmentId: int): Assessment
etInstance(): DBTableAssessment
etLastId(): String
SMAssessmentMethod(in assessmentId: String): boolean
st(in data: DataObject): LinkedList
ext(in data: DataObject): LinkedList
evious(in data: DataObject): LinkedList
uery(in data: DataObject): LinkedList

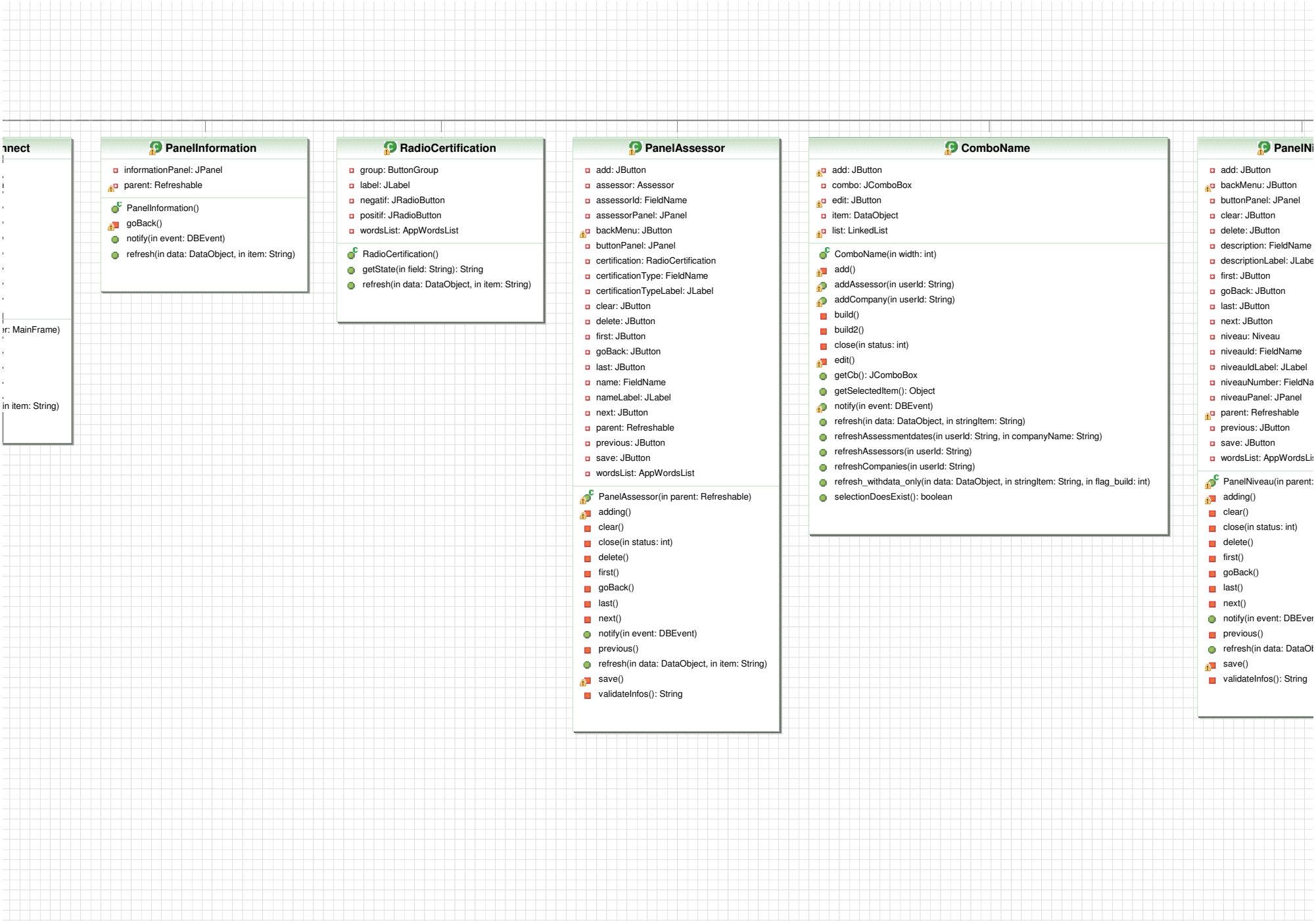

DBTable
dbConnection: DBConnection
name: String
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
executeQuery(in sql: String): ResultSet
executeUpdate(in sql: String): boolean
first(in data: DataObject): LinkedList
getName(): String
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
«access»
«access»
DBTableAssesse
address: String
compagnyID: String
compagnyName: String
contactPerson: String
country: String
eMail: String
fax: String
phone: String
postalCode: String
province: String
table: DBTableAssesse
DBTableAssesse()
add(in data: DataObject): boolean
addCompany(in companyName: String, in userId: String)
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getAssessmentDates(in userId: String, in companyName: String): LinkedList
getCompanies(in userId: String): LinkedList
getCompanyId(in companyName: String, in userId: String): String
getCompanyName(in companyId: int): String
getInstance(): DBTableAssesse
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
DBTableQuestionnaire
answer: String
assessmentId: String
comment: String
questionNumber: String
questionnaireId: String
table: DBTableQuestionnaire
DBTableQuestionnaire()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getComment(in questionNumber: int, in assessmentId: int): String
getDomainPercentage(in assessmentId: int, in level: int, in domain: int): Float
getInstance(): DBTableQuestionnaire
getKPAPercentage(in assessmentId: int, in level: int, in domain: int, in kpaNumber: int): float
getKPAsPercentages(in assessmentId: int, in level: int, in domain: int): LinkedList
getPercentage(in questionNumber: int, in assessmentId: int): String
getPracticesSuggestion(in assessmentId: int, in domainLevel: int, in domainRating: float, in domain: int): String
getPreviousAssessmentId(in assessmentId: int, in questionNumber: int, in companyId: int, in userId: String): int
getPreviousPracticePercentage(in assessmentId: int, in questionNumber: int, in companyId: int, in userId: String): String
getQuestionnaireId(in questionNumber: int, in assessmentId: int): int
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
«access»
«import»
DBTableKpa
description: String
kpaId: String
kpaNumber: String
processId: String
table: DBTableKpa
DBTableKpa()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String):
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableKpa
getKpaName(in kpaNumber: String): String
getKpaNumber(in processId: int, in kpaId: int): String
getNumberOfKpas(in domain: int): int
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList

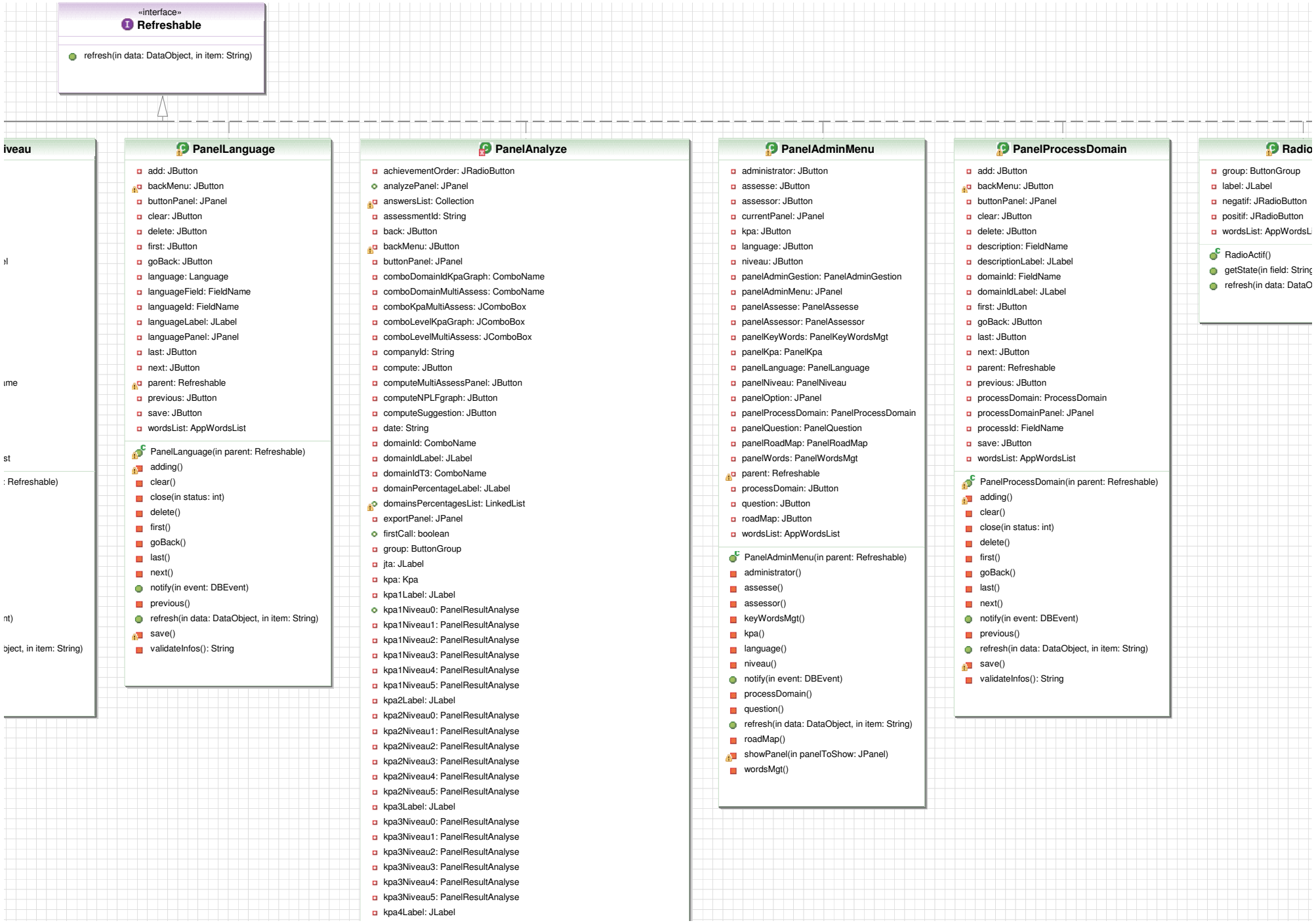

LinkedList
«import»
«access»
«import»
«access»
DBTableAdministration
table: DBTableAdministration
DBTableAdministration()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableAdministration
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
DBTableQuestion
answerType: String
description: String
languageId: String
niveauId: String
questionId: String
questionNumber: String
roadId: String
table: DBTableQuestion
DBTableQuestion()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableQuestion
getNumberOfPractices(in domain: int, in kpa: int, in niveauId: int): int
getPracticeId(in questionNumber: int): String
getQuestionNumber(in questionId: int, in level: int, in processId: int, in kpaID: int): int
last(in data: DataObject): LinkedList
lastLevelQuestion(in question: DataObject): int
lastQuestion(in question: DataObject): String
next(in data: DataObject): LinkedList
nextQuestion(in question: DataObject): Question
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
DBTableProcessDomain
description: String
domainID: String
processID: String
table: DBTableProcessDomain
DBTableProcessDomain()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableProcessDomain
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList

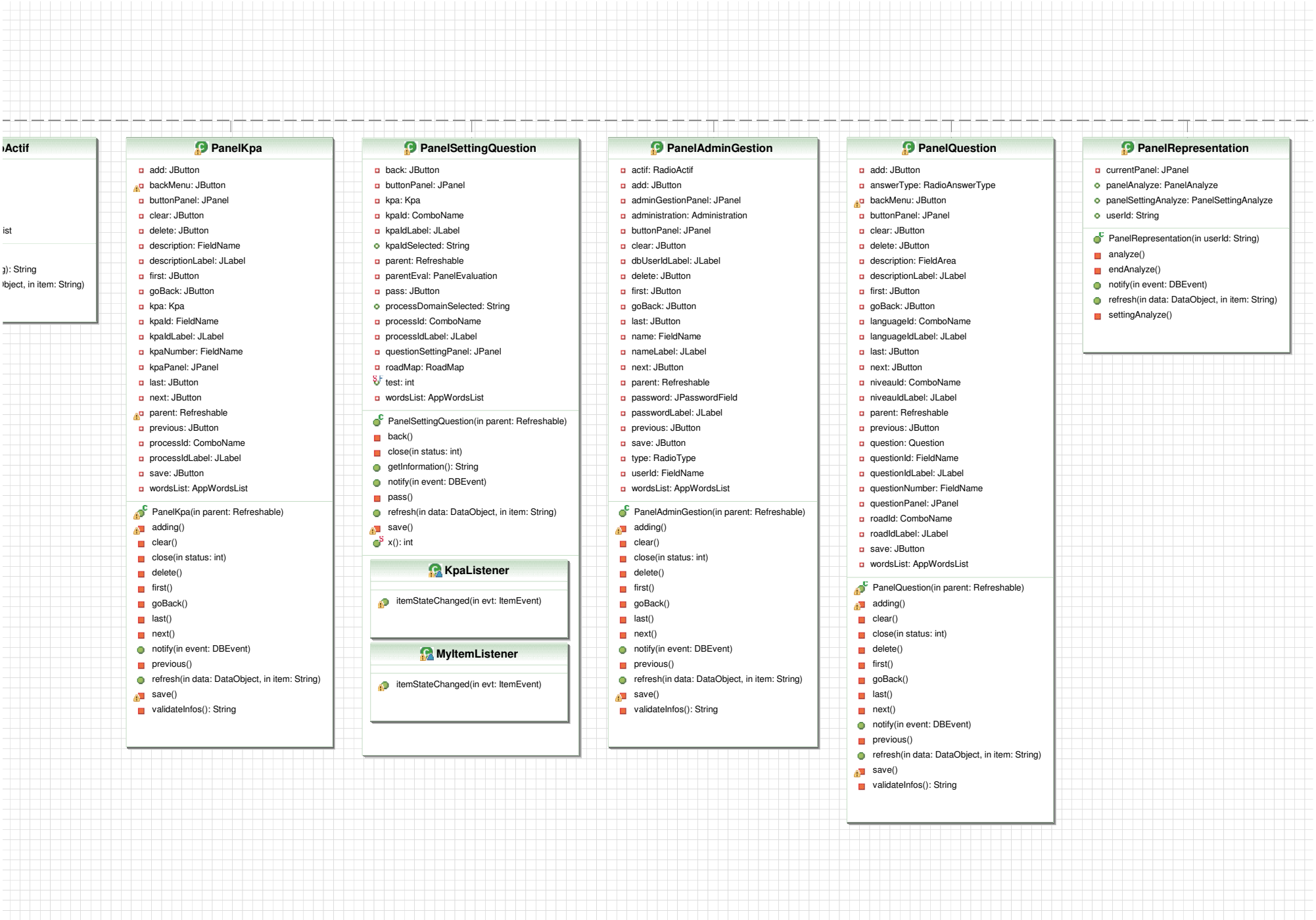

DBTableRoadMap
description: String
kpaId: String
roadId: String
roadNumber: String
table: DBTableRoadMap
DBTableRoadMap()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableRoadMap
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList
DBTableLanguage
language: String
languageId: String
table: DBTableLanguage
DBTableLanguage()
add(in data: DataObject): boolean
connect(in data: DataObject, in userId: String, in password: String): LinkedList
count(in data: DataObject): LinkedList
delete(in data: DataObject): boolean
edit(in data: DataObject): boolean
first(in data: DataObject): LinkedList
getInstance(): DBTableLanguage
getLanguagesList(): Vector
last(in data: DataObject): LinkedList
next(in data: DataObject): LinkedList
previous(in data: DataObject): LinkedList
query(in data: DataObject): LinkedList

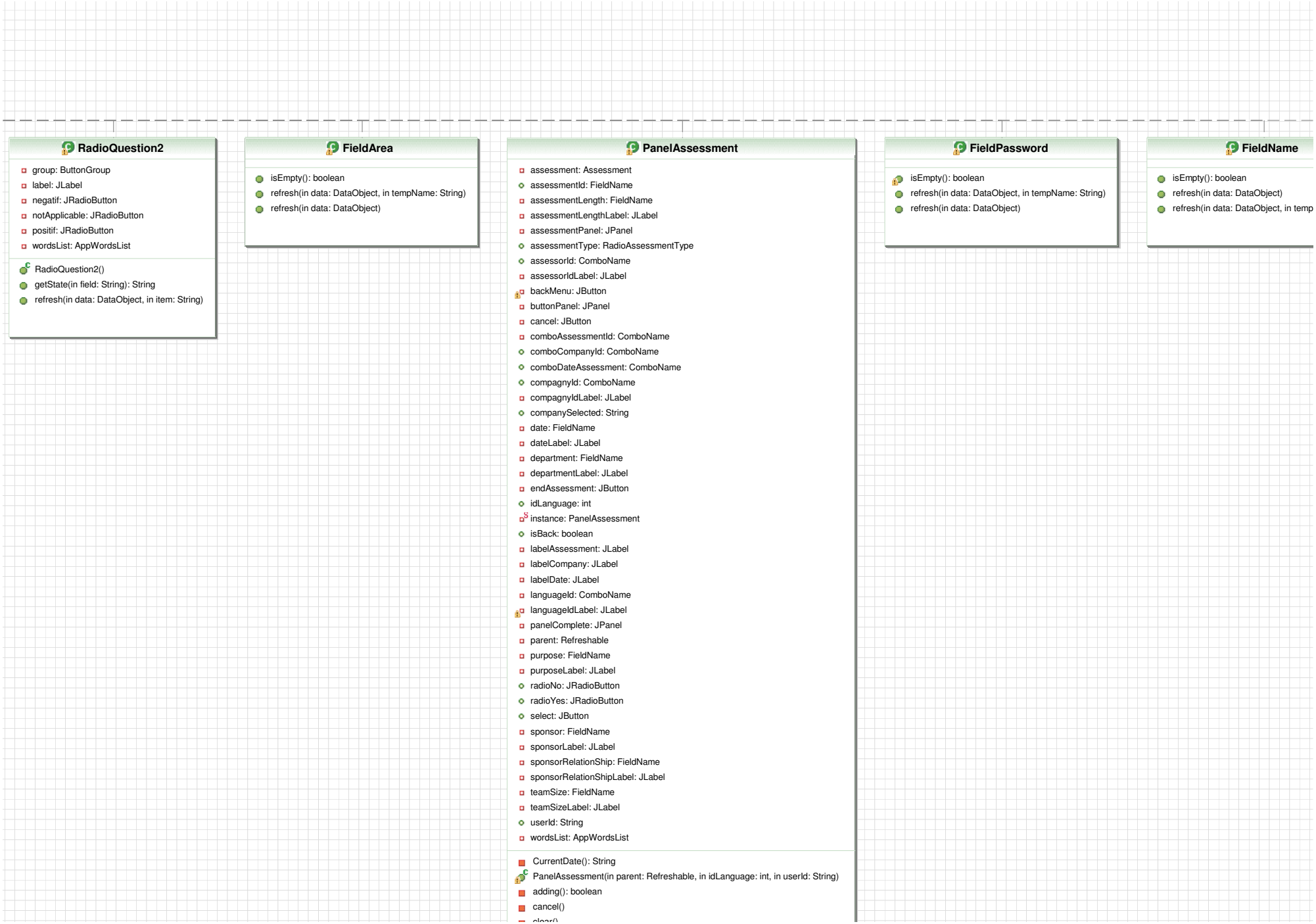

RadioAssessmentType
full: JRadioButton
group: ButtonGroup
label: JLabel
quick: JRadioButton
wordsList: AppWordsList
RadioAssessmentType()
getState(in field: String): String
refresh(in data: DataObject, in item: String)
PanelAdmin
connection: DataObject
currentPanel: Refreshable
panelAdminConnect: PanelAdminConnect
panelAdminMenu: PanelAdminMenu
PanelAdmin()
notify(in event: DBEvent)
refresh(in data: DataObject, in item: String)
RadioAnswerType
answer2: JRadioButton
answer4: JRadioButton
group: ButtonGroup
label: JLabel
wordsList: AppWordsList
RadioAnswerType()
getState(in field: String): String
refresh(in data: DataObject, in item: String)
PanelRoadMap
add: JButton
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
description: FieldName
descriptionLabel: JLabel
first: JButton
goBack: JButton
kpaId: ComboName
kpaIdLabel: JLabel
last: JButton
next: JButton
parent: Refreshable
previous: JButton
roadMap: RoadMap
roadId: FieldName
roadIdLabel: JLabel
roadMapPanel: JPanel
roadNumber: FieldName
save: JButton
wordsList: AppWordsList
PanelRoadMap(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
RadioType
group: ButtonGroup
label: JLabel
negatif: JRadioButton
positif: JRadioButton
wordsList: AppWordsList
RadioType()
getState(in field: String): String
refresh(in data: DataObject, in item: String)
RadioQue
group: ButtonGroup
label: JLabel
ratingF: JRadioButton
ratingL: JRadioButton
ratingN: JRadioButton
ratingNot: JRadioButton
ratingP: JRadioButton
wordsList: AppWordsList
RadioQuestion4()
getState(in field: String):
refresh(in data: DataObj

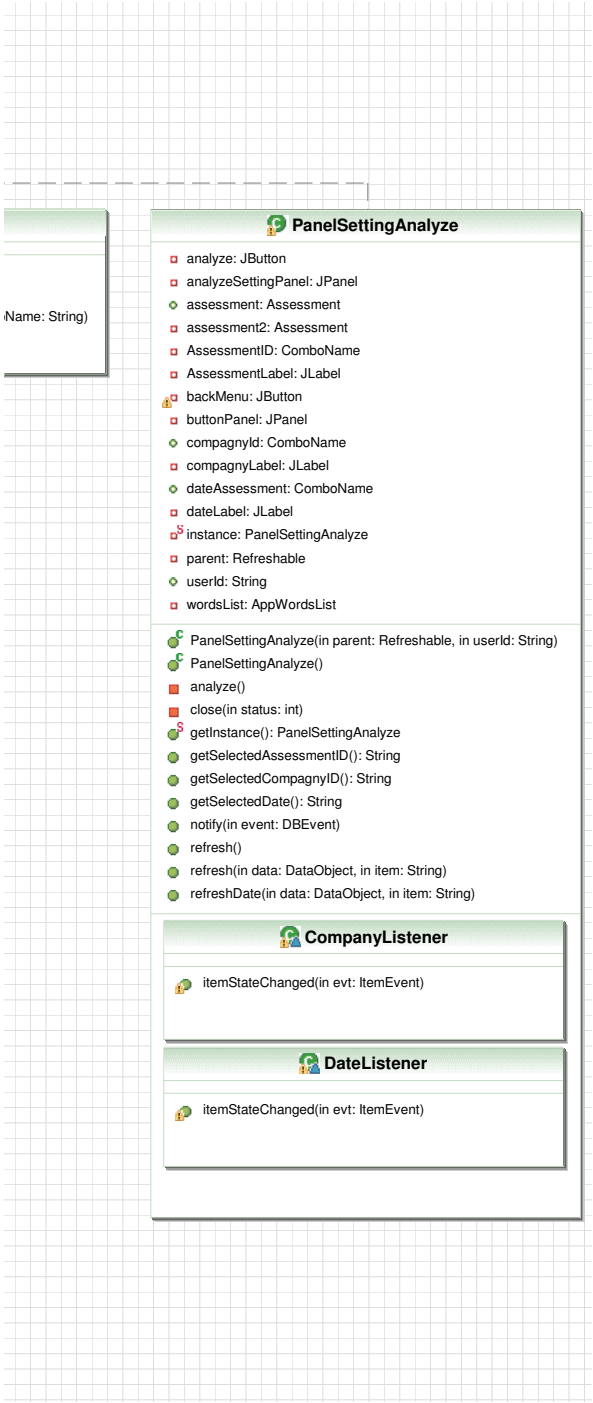

stion4
String
ect, in item: String)
PanelAssesse
add: JButton
address: FieldName
addressLabel: JLabel
assesse: Assesse
assessePanel: JPanel
backMenu: JButton
buttonPanel: JPanel
clear: JButton
compagnyId: FieldName
compagnyName: FieldName
compagnyNameLabel: JLabel
contactPerson: FieldName
contactPersonLabel: JLabel
country: FieldName
countryLabel: JLabel
delete: JButton
email: FieldName
emailLabel: JLabel
fax: FieldName
faxLabel: JLabel
first: JButton
goBack: JButton
last: JButton
next: JButton
panelAdminMenu: PanelAdminMenu
parent: Refreshable
phone: FieldName
phoneLabel: JLabel
postalCode: FieldName
postalCodeLabel: JLabel
previous: JButton
province: FieldName
provinceLabel: JLabel
save: JButton
wordsList: AppWordsList
PanelAssesse(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
PanelEvaluation
currentPanel: JPanel
panelAssessment: PanelAssessment
panelQuestionnaire: PanelQuestionnaire
panelSettingQuestion: PanelSettingQuestion
userId: String
PanelEvaluation(in idLanguage: int, in userId: String)
assessment()
endEvaluation()
notify(in event: DBEvent)
questionnaire()
questionnaireFirst(in domain: String)
refresh(in data: DataObject, in item: String)
refreshFirstQuestion(in data: DataObject, in item: String, in domain: String, in kpaId: String)
settingQuestion()
PanelQuestionnaire
answer2: RadioQuestion2
answer4: RadioQuestion4
assessmentId: String
buttonPanel: JPanel
comment: FieldArea
commentLabel: JLabel
companySelected: String
edit: boolean
firstUseNext: int
instance: PanelQuestionnaire
newQuestion: Question
next: JButton
nextQuestion: boolean
niveau: Niveau
parent: Refreshable
question: Question
questionField: FieldArea
questionLabel: JLabel
questionNumberMin: String
questionnaire: Questionnaire
questionnairePanel: JPanel
questionsLanguage: String
roadMap: RoadMap
smAssessmentMethod: boolean
tempKpa: String
userId: String
wordsList: AppWordsList
PanelQuestionnaire()
PanelQuestionnaire(in parent: Refreshable, in userId: String)
adding(): boolean
answerType()
close(in status: int)
firstRefresh(in data: DataObject, in item: String, in domain: String)
getInstance(): PanelQuestionnaire
isUnder80(): boolean
next()
notify(in event: DBEvent)
quit()
refresh(in data: DataObject, in item: String)
refreshNiveau(in data: DataObject, in item: String)
refreshQuestion(in data: DataObject, in item: String)
refreshRoadMap(in data: DataObject, in item: String)
setInformation(in item: String, in assessmentNumber: String, in company: String)
setInformationFirst(in item: String, in assessmentNumber: String, in domain: String)
setQuestionsLanguage(in language: String)
setSMAssessmentMethod(in smAssessmentMethod: boolean)
PanelAdminCor
accept: JButton
administration: Administration
clear: JButton
dbUserIdLabel: JLabel
panelConnectAdmin: JPanel
parent: Refreshable
password: JTextField
passwordLabel: JLabel
userId: JTextField
wordsList: AppWordsList
PanelAdminConnect(in owne
clear()
close(in status: int)
connection(): boolean
notify(in event: DBEvent)
refresh(in data: DataObject,

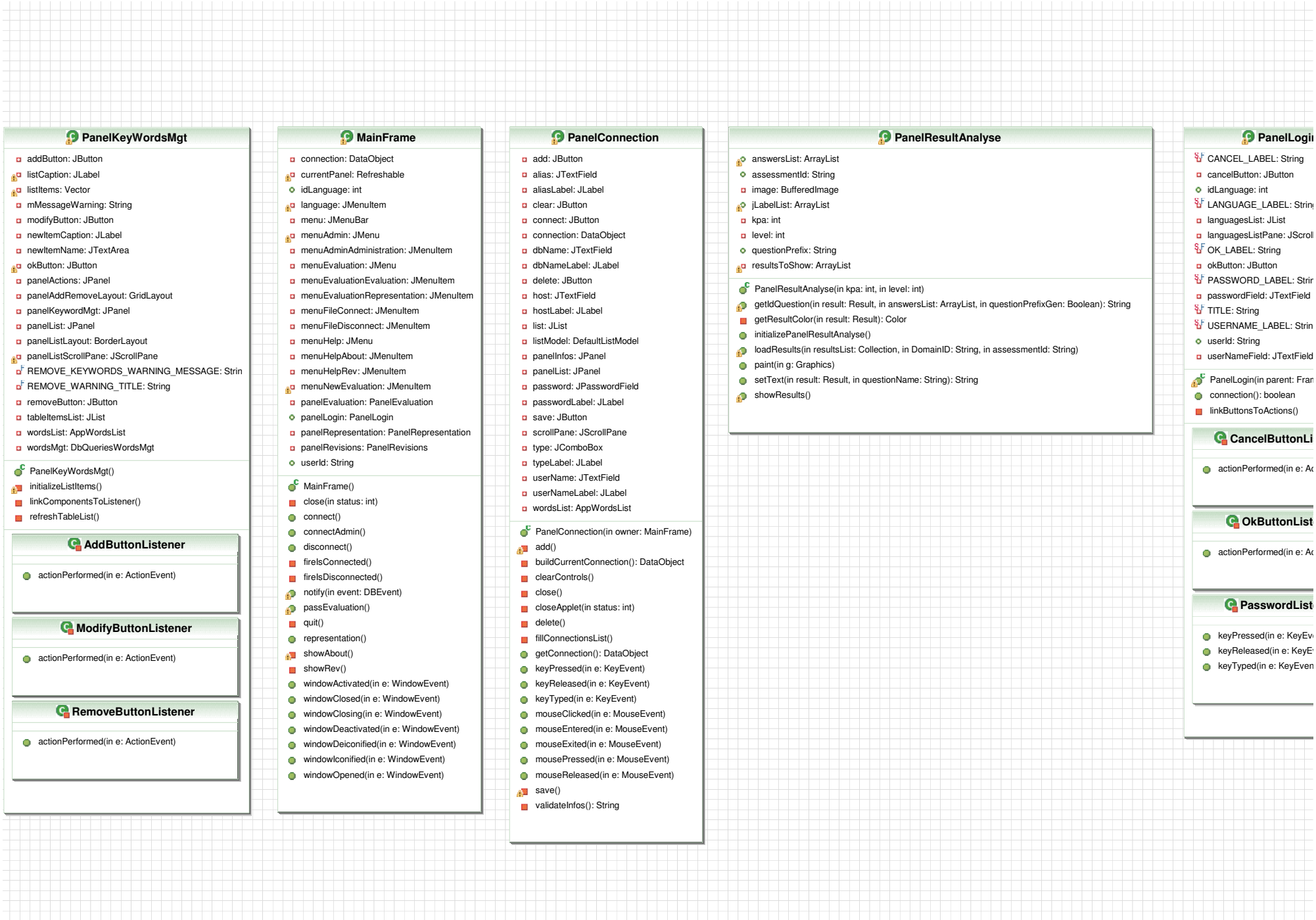

nect
r: MainFrame)
in item: String)
PanelInformation
informationPanel: JPanel
parent: Refreshable
PanelInformation()
goBack()
notify(in event: DBEvent)
refresh(in data: DataObject, in item: String)
RadioCertification
group: ButtonGroup
label: JLabel
negatif: JRadioButton
positif: JRadioButton
wordsList: AppWordsList
RadioCertification()
getState(in field: String): String
refresh(in data: DataObject, in item: String)
PanelAssessor
add: JButton
assessor: Assessor
assessorId: FieldName
assessorPanel: JPanel
backMenu: JButton
buttonPanel: JPanel
certification: RadioCertification
certificationType: FieldName
certificationTypeLabel: JLabel
clear: JButton
delete: JButton
first: JButton
goBack: JButton
last: JButton
name: FieldName
nameLabel: JLabel
next: JButton
parent: Refreshable
previous: JButton
save: JButton
wordsList: AppWordsList
PanelAssessor(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
ComboName
add: JButton
combo: JComboBox
edit: JButton
item: DataObject
list: LinkedList
ComboName(in width: int)
add()
addAssessor(in userId: String)
addCompany(in userId: String)
build()
build2()
close(in status: int)
edit()
getCb(): JComboBox
getSelectedItem(): Object
notify(in event: DBEvent)
refresh(in data: DataObject, in stringItem: String)
refreshAssessmentdates(in userId: String, in companyName: String)
refreshAssessors(in userId: String)
refreshCompanies(in userId: String)
refresh_withdata_only(in data: DataObject, in stringItem: String, in flag_build: int)
selectionDoesExist(): boolean
PanelNi
add: JButton
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
description: FieldName
descriptionLabel: JLabe
first: JButton
goBack: JButton
last: JButton
next: JButton
niveau: Niveau
niveauId: FieldName
niveauIdLabel: JLabel
niveauNumber: FieldNa
niveauPanel: JPanel
parent: Refreshable
previous: JButton
save: JButton
wordsList: AppWordsLi
PanelNiveau(in parent:
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEver
previous()
refresh(in data: DataOb
save()
validateInfos(): String

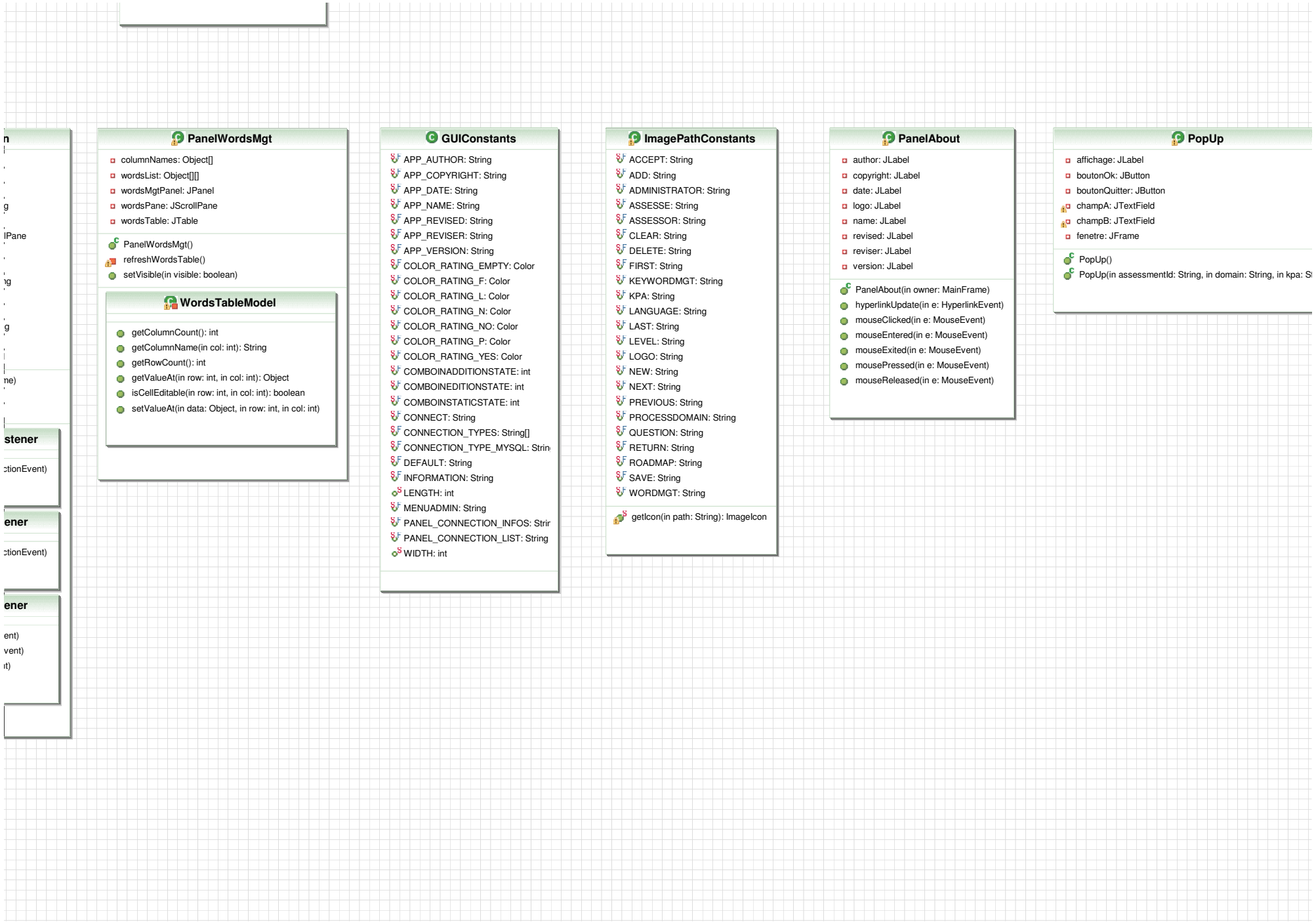

«interface»
Refreshable
refresh(in data: DataObject, in item: String)
PanelLanguage
add: JButton
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
first: JButton
goBack: JButton
language: Language
languageField: FieldName
languageId: FieldName
languageLabel: JLabel
languagePanel: JPanel
last: JButton
next: JButton
parent: Refreshable
previous: JButton
save: JButton
wordsList: AppWordsList
PanelLanguage(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
PanelAnalyze
achievementOrder: JRadioButton
analyzePanel: JPanel
answersList: Collection
assessmentId: String
back: JButton
backMenu: JButton
buttonPanel: JPanel
comboDomainIdKpaGraph: ComboName
comboDomainMultiAssess: ComboName
comboKpaMultiAssess: JComboBox
comboLevelKpaGraph: JComboBox
comboLevelMultiAssess: JComboBox
companyId: String
compute: JButton
computeMultiAssessPanel: JButton
computeNPLFgraph: JButton
computeSuggestion: JButton
date: String
domainId: ComboName
domainIdLabel: JLabel
domainIdT3: ComboName
domainPercentageLabel: JLabel
domainsPercentagesList: LinkedList
exportPanel: JPanel
firstCall: boolean
group: ButtonGroup
jta: JLabel
kpa: Kpa
kpa1Label: JLabel
kpa1Niveau0: PanelResultAnalyse
kpa1Niveau1: PanelResultAnalyse
kpa1Niveau2: PanelResultAnalyse
kpa1Niveau3: PanelResultAnalyse
kpa1Niveau4: PanelResultAnalyse
kpa1Niveau5: PanelResultAnalyse
kpa2Label: JLabel
kpa2Niveau0: PanelResultAnalyse
kpa2Niveau1: PanelResultAnalyse
kpa2Niveau2: PanelResultAnalyse
kpa2Niveau3: PanelResultAnalyse
kpa2Niveau4: PanelResultAnalyse
kpa2Niveau5: PanelResultAnalyse
kpa3Label: JLabel
kpa3Niveau0: PanelResultAnalyse
kpa3Niveau1: PanelResultAnalyse
kpa3Niveau2: PanelResultAnalyse
kpa3Niveau3: PanelResultAnalyse
kpa3Niveau4: PanelResultAnalyse
kpa3Niveau5: PanelResultAnalyse
kpa4Label: JLabel
PanelAdminMenu
administrator: JButton
assesse: JButton
assessor: JButton
currentPanel: JPanel
kpa: JButton
language: JButton
niveau: JButton
panelAdminGestion: PanelAdminGestion
panelAdminMenu: JPanel
panelAssesse: PanelAssesse
panelAssessor: PanelAssessor
panelKeyWords: PanelKeyWordsMgt
panelKpa: PanelKpa
panelLanguage: PanelLanguage
panelNiveau: PanelNiveau
panelOption: JPanel
panelProcessDomain: PanelProcessDomain
panelQuestion: PanelQuestion
panelRoadMap: PanelRoadMap
panelWords: PanelWordsMgt
parent: Refreshable
processDomain: JButton
question: JButton
roadMap: JButton
wordsList: AppWordsList
PanelAdminMenu(in parent: Refreshable)
administrator()
assesse()
assessor()
keyWordsMgt()
kpa()
language()
niveau()
notify(in event: DBEvent)
processDomain()
question()
refresh(in data: DataObject, in item: String)
roadMap()
showPanel(in panelToShow: JPanel)
wordsMgt()
PanelProcessDomain
add: JButton
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
description: FieldName
descriptionLabel: JLabel
domainId: FieldName
domainIdLabel: JLabel
first: JButton
goBack: JButton
last: JButton
next: JButton
parent: Refreshable
previous: JButton
processDomain: ProcessDomain
processDomainPanel: JPanel
processId: FieldName
save: JButton
wordsList: AppWordsList
PanelProcessDomain(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
group: ButtonGroup
label: JLabel
negatif: JRadioButton
positif: JRadioButton
wordsList: AppWordsL
RadioActif()
getState(in field: String
refresh(in data: DataO

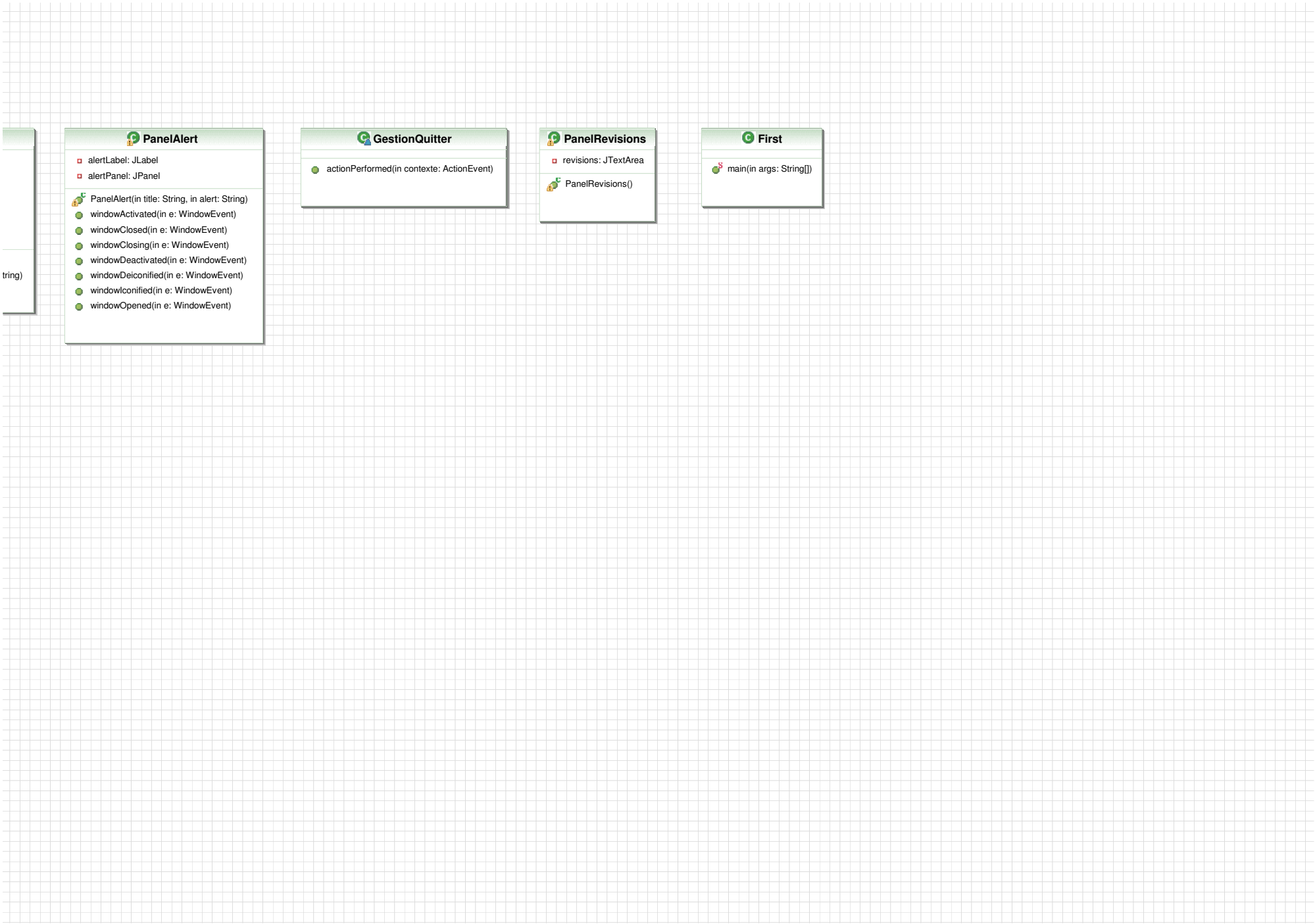

Actif
ist
): String
bject, in item: String)
PanelKpa
add: JButton
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
description: FieldName
descriptionLabel: JLabel
first: JButton
goBack: JButton
kpa: Kpa
kpaId: FieldName
kpaIdLabel: JLabel
kpaNumber: FieldName
kpaPanel: JPanel
last: JButton
next: JButton
parent: Refreshable
previous: JButton
processId: ComboName
processIdLabel: JLabel
save: JButton
wordsList: AppWordsList
PanelKpa(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
PanelSettingQuestion
back: JButton
buttonPanel: JPanel
kpa: Kpa
kpaId: ComboName
kpaIdLabel: JLabel
kpaIdSelected: String
parent: Refreshable
parentEval: PanelEvaluation
pass: JButton
processDomainSelected: String
processId: ComboName
processIdLabel: JLabel
questionSettingPanel: JPanel
roadMap: RoadMap
test: int
wordsList: AppWordsList
PanelSettingQuestion(in parent: Refreshable)
back()
close(in status: int)
getInformation(): String
notify(in event: DBEvent)
pass()
refresh(in data: DataObject, in item: String)
save()
x(): int
KpaListener
itemStateChanged(in evt: ItemEvent)
MyItemListener
itemStateChanged(in evt: ItemEvent)
PanelAdminGestion
actif: RadioActif
add: JButton
adminGestionPanel: JPanel
administration: Administration
buttonPanel: JPanel
clear: JButton
dbUserIdLabel: JLabel
delete: JButton
first: JButton
goBack: JButton
last: JButton
name: FieldName
nameLabel: JLabel
next: JButton
parent: Refreshable
password: JPasswordField
passwordLabel: JLabel
previous: JButton
save: JButton
type: RadioType
userId: FieldName
wordsList: AppWordsList
PanelAdminGestion(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
PanelQuestion
add: JButton
answerType: RadioAnswerType
backMenu: JButton
buttonPanel: JPanel
clear: JButton
delete: JButton
description: FieldArea
descriptionLabel: JLabel
first: JButton
goBack: JButton
languageId: ComboName
languageIdLabel: JLabel
last: JButton
next: JButton
niveauId: ComboName
niveauIdLabel: JLabel
parent: Refreshable
previous: JButton
question: Question
questionId: FieldName
questionIdLabel: JLabel
questionNumber: FieldName
questionPanel: JPanel
roadId: ComboName
roadIdLabel: JLabel
save: JButton
wordsList: AppWordsList
PanelQuestion(in parent: Refreshable)
adding()
clear()
close(in status: int)
delete()
first()
goBack()
last()
next()
notify(in event: DBEvent)
previous()
refresh(in data: DataObject, in item: String)
save()
validateInfos(): String
PanelRepresentation
currentPanel: JPanel
panelAnalyze: PanelAnalyze
panelSettingAnalyze: PanelSettingAnalyze
userId: String
PanelRepresentation(in userId: String)
analyze()
endAnalyze()
notify(in event: DBEvent)
refresh(in data: DataObject, in item: String)
settingAnalyze()


RadioQuestion2
group: ButtonGroup
label: JLabel
negatif: JRadioButton
notApplicable: JRadioButton
positif: JRadioButton
wordsList: AppWordsList
RadioQuestion2()
getState(in field: String): String
refresh(in data: DataObject, in item: String)
FieldArea
isEmpty(): boolean
refresh(in data: DataObject, in tempName: String)
refresh(in data: DataObject)
PanelAssessment
assessment: Assessment
assessmentId: FieldName
assessmentLength: FieldName
assessmentLengthLabel: JLabel
assessmentPanel: JPanel
assessmentType: RadioAssessmentType
assessorId: ComboName
assessorIdLabel: JLabel
backMenu: JButton
buttonPanel: JPanel
cancel: JButton
comboAssessmentId: ComboName
comboCompanyId: ComboName
comboDateAssessment: ComboName
compagnyId: ComboName
compagnyIdLabel: JLabel
companySelected: String
date: FieldName
dateLabel: JLabel
department: FieldName
departmentLabel: JLabel
endAssessment: JButton
idLanguage: int
instance: PanelAssessment
isBack: boolean
labelAssessment: JLabel
labelCompany: JLabel
labelDate: JLabel
languageId: ComboName
languageIdLabel: JLabel
panelComplete: JPanel
parent: Refreshable
purpose: FieldName
purposeLabel: JLabel
radioNo: JRadioButton
radioYes: JRadioButton
select: JButton
sponsor: FieldName
sponsorLabel: JLabel
sponsorRelationShip: FieldName
sponsorRelationShipLabel: JLabel
teamSize: FieldName
teamSizeLabel: JLabel
userId: String
wordsList: AppWordsList
CurrentDate(): String
PanelAssessment(in parent: Refreshable, in idLanguage: int, in userId: String)
adding(): boolean
cancel()
FieldPassword
isEmpty(): boolean
refresh(in data: DataObject, in tempName: String)
refresh(in data: DataObject)
FieldName
isEmpty(): boolean
refresh(in data: DataObject)
refresh(in data: DataObject, in temp


Name: String)
PanelSettingAnalyze
analyze: JButton
analyzeSettingPanel: JPanel
assessment: Assessment
assessment2: Assessment
AssessmentID: ComboName
AssessmentLabel: JLabel
backMenu: JButton
buttonPanel: JPanel
compagnyId: ComboName
compagnyLabel: JLabel
dateAssessment: ComboName
dateLabel: JLabel
instance: PanelSettingAnalyze
parent: Refreshable
userId: String
wordsList: AppWordsList
PanelSettingAnalyze(in parent: Refreshable, in userId: String)
PanelSettingAnalyze()
analyze()
close(in status: int)
getInstance(): PanelSettingAnalyze
getSelectedAssessmentID(): String
getSelectedCompagnyID(): String
getSelectedDate(): String
notify(in event: DBEvent)
refresh()
refresh(in data: DataObject, in item: String)
refreshDate(in data: DataObject, in item: String)
CompanyListener
itemStateChanged(in evt: ItemEvent)
DateListener
itemStateChanged(in evt: ItemEvent)

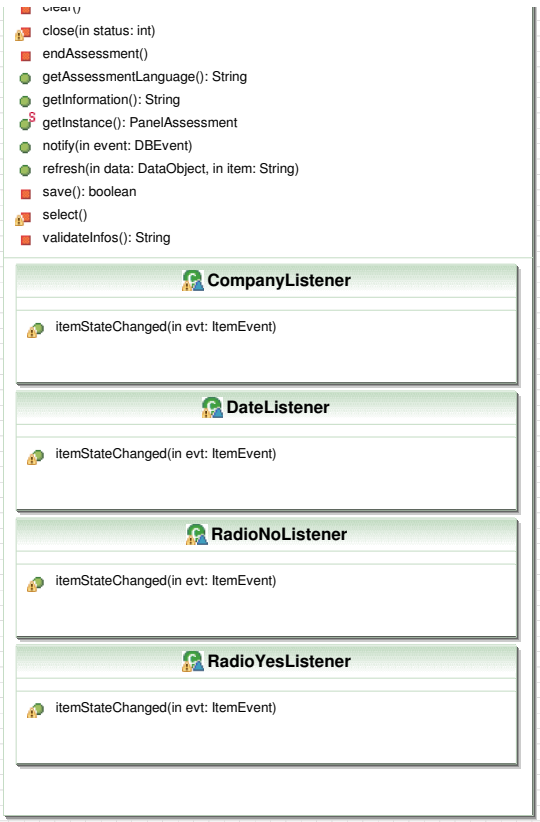

PanelKeyWordsMgt
addButton: JButton
listCaption: JLabel
listItems: Vector
mMessageWarning: String
modifyButton: JButton
newItemCaption: JLabel
newItemName: JTextArea
okButton: JButton
panelActions: JPanel
panelAddRemoveLayout: GridLayout
panelKeywordMgt: JPanel
panelList: JPanel
panelListLayout: BorderLayout
panelListScrollPane: JScrollPane
REMOVE_KEYWORDS_WARNING_MESSAGE: Strin
REMOVE_WARNING_TITLE: String
removeButton: JButton
tableItemsList: JList
wordsList: AppWordsList
wordsMgt: DbQueriesWordsMgt
PanelKeyWordsMgt()
initializeListItems()
linkComponentsToListener()
refreshTableList()
AddButtonListener
actionPerformed(in e: ActionEvent)
ModifyButtonListener
actionPerformed(in e: ActionEvent)
RemoveButtonListener
actionPerformed(in e: ActionEvent)
MainFrame
connection: DataObject
currentPanel: Refreshable
idLanguage: int
language: JMenuItem
menu: JMenuBar
menuAdmin: JMenu
menuAdminAdministration: JMenuItem
menuEvaluation: JMenu
menuEvaluationEvaluation: JMenuItem
menuEvaluationRepresentation: JMenuItem
menuFileConnect: JMenuItem
menuFileDisconnect: JMenuItem
menuHelp: JMenu
menuHelpAbout: JMenuItem
menuHelpRev: JMenuItem
menuNewEvaluation: JMenuItem
panelEvaluation: PanelEvaluation
panelLogin: PanelLogin
panelRepresentation: PanelRepresentation
panelRevisions: PanelRevisions
userId: String
MainFrame()
close(in status: int)
connect()
connectAdmin()
disconnect()
fireIsConnected()
fireIsDisconnected()
notify(in event: DBEvent)
passEvaluation()
quit()
representation()
showAbout()
showRev()
windowActivated(in e: WindowEvent)
windowClosed(in e: WindowEvent)
windowClosing(in e: WindowEvent)
windowDeactivated(in e: WindowEvent)
windowDeiconified(in e: WindowEvent)
windowIconified(in e: WindowEvent)
windowOpened(in e: WindowEvent)
PanelConnection
add: JButton
alias: JTextField
aliasLabel: JLabel
clear: JButton
connect: JButton
connection: DataObject
dbName: JTextField
dbNameLabel: JLabel
delete: JButton
host: JTextField
hostLabel: JLabel
list: JList
listModel: DefaultListModel
panelInfos: JPanel
panelList: JPanel
password: JPasswordField
passwordLabel: JLabel
save: JButton
scrollPane: JScrollPane
type: JComboBox
typeLabel: JLabel
userName: JTextField
userNameLabel: JLabel
wordsList: AppWordsList
PanelConnection(in owner: MainFrame)
add()
buildCurrentConnection(): DataObject
clearControls()
close()
closeApplet(in status: int)
delete()
fillConnectionsList()
getConnection(): DataObject
keyPressed(in e: KeyEvent)
keyReleased(in e: KeyEvent)
keyTyped(in e: KeyEvent)
mouseClicked(in e: MouseEvent)
mouseEntered(in e: MouseEvent)
mouseExited(in e: MouseEvent)
mousePressed(in e: MouseEvent)
mouseReleased(in e: MouseEvent)
save()
validateInfos(): String
PanelResultAnalyse
answersList: ArrayList
assessmentId: String
image: BufferedImage
jLabelList: ArrayList
kpa: int
level: int
questionPrefix: String
resultsToShow: ArrayList
PanelResultAnalyse(in kpa: int, in level: int)
getIdQuestion(in result: Result, in answersList: ArrayList, in questionPrefixGen: Boolean): String
getResultColor(in result: Result): Color
initializePanelResultAnalyse()
loadResults(in resultsList: Collection, in DomainID: String, in assessmentId: String)
paint(in g: Graphics)
setText(in result: Result, in questionName: String): String
showResults()
PanelLogi
CANCEL_LABEL: String
cancelButton: JButton
idLanguage: int
LANGUAGE_LABEL: Strin
languagesList: JList
languagesListPane: JScrol
OK_LABEL: String
okButton: JButton
PASSWORD_LABEL: Strir
passwordField: JTextField
TITLE: String
USERNAME_LABEL: Strin
userId: String
userNameField: JTextField
PanelLogin(in parent: Fran
connection(): boolean
linkButtonsToActions()
CancelButtonLi
actionPerformed(in e: Ac
OkButtonList
actionPerformed(in e: Ac
PasswordList
keyPressed(in e: KeyEv
keyReleased(in e: KeyE
keyTyped(in e: KeyEven

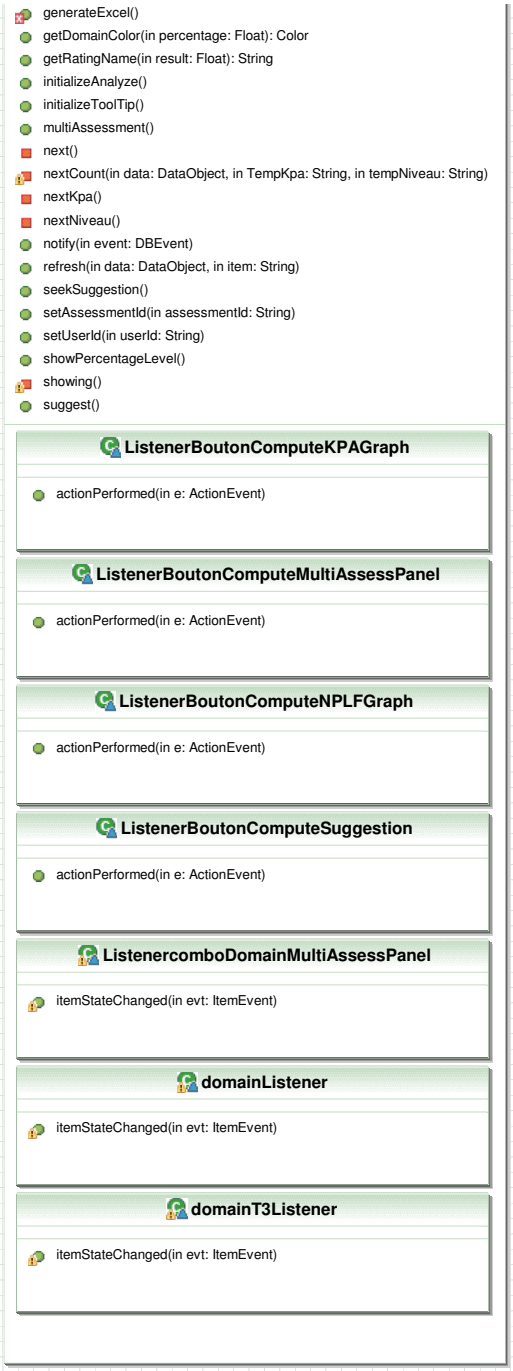

PanelWordsMgt
columnNames: Object[]
wordsList: Object[][]
wordsMgtPanel: JPanel
wordsPane: JScrollPane
wordsTable: JTable
PanelWordsMgt()
refreshWordsTable()
setVisible(in visible: boolean)
WordsTableModel
getColumnCount(): int
getColumnName(in col: int): String
getRowCount(): int
getValueAt(in row: int, in col: int): Object
isCellEditable(in row: int, in col: int): boolean
setValueAt(in data: Object, in row: int, in col: int)
GUIConstants
APP_AUTHOR: String
APP_COPYRIGHT: String
APP_DATE: String
APP_NAME: String
APP_REVISED: String
APP_REVISER: String
APP_VERSION: String
COLOR_RATING_EMPTY: Color
COLOR_RATING_F: Color
COLOR_RATING_L: Color
COLOR_RATING_N: Color
COLOR_RATING_NO: Color
COLOR_RATING_P: Color
COLOR_RATING_YES: Color
COMBOINADDITIONSTATE: int
COMBOINEDITIONSTATE: int
COMBOINSTATICSTATE: int
CONNECT: String
CONNECTION_TYPES: String[]
CONNECTION_TYPE_MYSQL: Strin
DEFAULT: String
INFORMATION: String
LENGTH: int
MENUADMIN: String
PANEL_CONNECTION_INFOS: Strir
PANEL_CONNECTION_LIST: String
WIDTH: int
ImagePathConstants
ACCEPT: String
ADD: String
ADMINISTRATOR: String
ASSESSE: String
ASSESSOR: String
CLEAR: String
DELETE: String
FIRST: String
KEYWORDMGT: String
KPA: String
LANGUAGE: String
LAST: String
LEVEL: String
LOGO: String
NEW: String
NEXT: String
PREVIOUS: String
PROCESSDOMAIN: String
QUESTION: String
RETURN: String
ROADMAP: String
SAVE: String
WORDMGT: String
getIcon(in path: String): ImageIcon
PanelAbout
author: JLabel
copyright: JLabel
date: JLabel
logo: JLabel
name: JLabel
revised: JLabel
reviser: JLabel
version: JLabel
PanelAbout(in owner: MainFrame)
hyperlinkUpdate(in e: HyperlinkEvent)
mouseClicked(in e: MouseEvent)
mouseEntered(in e: MouseEvent)
mouseExited(in e: MouseEvent)
mousePressed(in e: MouseEvent)
mouseReleased(in e: MouseEvent)
PopUp
affichage: JLabel
boutonOk: JButton
boutonQuitter: JButton
champA: JTextField
champB: JTextField
fenetre: JFrame
PopUp()
PopUp(in assessmentId: String, in domain: String, in kpa: S


tring)
PanelAlert
alertLabel: JLabel
alertPanel: JPanel
PanelAlert(in title: String, in alert: String)
windowActivated(in e: WindowEvent)
windowClosed(in e: WindowEvent)
windowClosing(in e: WindowEvent)
windowDeactivated(in e: WindowEvent)
windowDeiconified(in e: WindowEvent)
windowIconified(in e: WindowEvent)
windowOpened(in e: WindowEvent)
GestionQuitter
actionPerformed(in contexte: ActionEvent)
PanelRevisions
revisions: JTextArea
PanelRevisions()
First
main(in args: String[])

- kpa4Niveau0: PanelResultAnalyse
- kpa4Niveau1: PanelResultAnalyse
- kpa4Niveau2: PanelResultAnalyse
- kpa4Niveau3: PanelResultAnalyse
- kpa4Niveau4: PanelResultAnalyse
- kpa4Niveau5: PanelResultAnalyse
- kpa5Label: JLabel
- kpa5Niveau0: PanelResultAnalyse
- kpa5Niveau1: PanelResultAnalyse
- kpa5Niveau2: PanelResultAnalyse
- kpa5Niveau3: PanelResultAnalyse
- kpa5Niveau4: PanelResultAnalyse
- kpa5Niveau5: PanelResultAnalyse
- kpaOrder: JRadioButton
- MAXKPA: String
- MAXNIVEAU: String
- maturityOrder: JRadioButton
- niveau: Niveau
- niveau0Label: JLabel
- niveau1Label: JLabel
- niveau2Label: JLabel
- niveau3Label: JLabel
- niveau4Label: JLabel
- niveau5Label: JLabel
- panelChoiceMultiAssess: JPanel
- panelDomainGraph: JPanel
- panelLeftT2AssessInfo: JPanel
- panelLeftT2Practices: JPanel
- panelMultiAssessment: JPanel
- panelT2: JPanel
- panelT3: JPanel
- panelkpaGraph: JPanel
- parent: Refreshable
- questionnaire: Questionnaire
- ratingF: FieldName
- ratingFLabel: JLabel
- ratingL: FieldName
- ratingLLabel: JLabel
- ratingN: FieldName
- ratingNLabel: JLabel
- ratingNot: FieldName
- ratingNotLabel: JLabel
- ratingP: FieldName
- ratingPLabel: JLabel
- sortingLabel: JLabel
- toExcel: JButton
- unOrdered: JRadioButton
- userId: String
- wordsList: AppWordsList

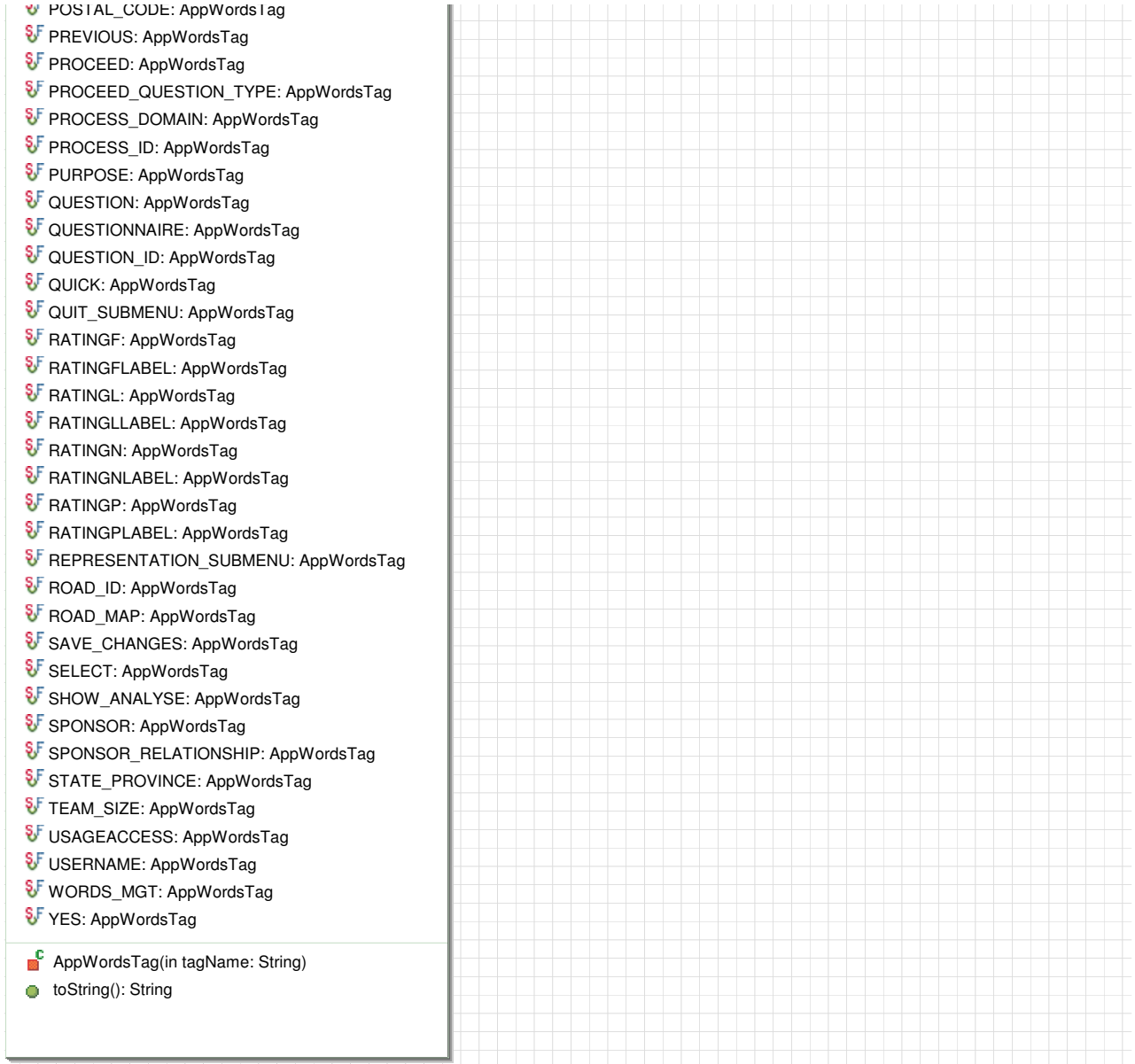
- PanelAnalyze(in parent: Refreshable)
- analyse(in compagnyID: String, in date: String, in AssessmentID: String)
- back()
- calculateDomainsPercentages()
- close(in status: int)
- createKPAChart()
- createMultiAssessChart()
- drawDomainsGraph()
- drawKPAGraph()
- drawNPLFGraph()
- generateDataTable(in answersList: Collection)


clear()
close(in status: int)
endAssessment()
getAssessmentLanguage(): String
getInformation(): String
getInstance(): PanelAssessment
notify(in event: DBEvent)
refresh(in data: DataObject, in item: String)
save(): boolean
select()
validateInfos(): String
CompanyListener
itemStateChanged(in evt: ItemEvent)
DateListener
itemStateChanged(in evt: ItemEvent)
RadioNoListener
itemStateChanged(in evt: ItemEvent)
RadioYesListener
itemStateChanged(in evt: ItemEvent)


generateExcel()
getDomainColor(in percentage: Float): Color
getRatingName(in result: Float): String
initializeAnalyze()
initializeToolTip()
multiAssessment()
next()
nextCount(in data: DataObject, in TempKpa: String, in tempNiveau: String)
nextKpa()
nextNiveau()
notify(in event: DBEvent)
refresh(in data: DataObject, in item: String)
seekSuggestion()
setAssessmentId(in assessmentId: String)
setUserId(in userId: String)
showPercentageLevel()
showing()
suggest()
ListenerBoutonComputeKPAGraph
actionPerformed(in e: ActionEvent)
ListenerBoutonComputeMultiAssessPanel
actionPerformed(in e: ActionEvent)
ListenerBoutonComputeNPLFGraph
actionPerformed(in e: ActionEvent)
ListenerBoutonComputeSuggestion
actionPerformed(in e: ActionEvent)
ListenercomboDomainMultiAssessPanel
itemStateChanged(in evt: ItemEvent)
domainListener
itemStateChanged(in evt: ItemEvent)
domainT3Listener
itemStateChanged(in evt: ItemEvent)


AppWordsTag
ABOUT_SUBMENU: AppWordsTag
ADD: AppWordsTag
ADDRESS: AppWordsTag
ADMINISTRATION_MENU: AppWordsTag
ADMINISTRATION_SUBMENU: AppWordsTag
ADMINISTRATOR: AppWordsTag
ANALYSE: AppWordsTag
ANALYSE_SHOW: AppWordsTag
ANALYZE: AppWordsTag
ANSWER2: AppWordsTag
ANSWER4: AppWordsTag
ANSWER_TYPE: AppWordsTag
ASSESSE: AppWordsTag
ASSESSMENT: AppWordsTag
ASSESSMENT_LENGTH: AppWordsTag
ASSESSOR: AppWordsTag
ASSESSOR_ID: AppWordsTag
BACK: AppWordsTag
BACK_TO_ASSESSEMENT: AppWordsTag
CERTIFICATION: AppWordsTag
CERTIFICATION_TYPE: AppWordsTag
CLEAR: AppWordsTag
COMMENT: AppWordsTag
COMPAGNY_ID: AppWordsTag
COMPAGNY_NAME: AppWordsTag
CONNECT: AppWordsTag
CONNECT_AS_ADMIN: AppWordsTag
CONNECT_DB_SUBMENU: AppWordsTag
CONNECT_TO_DATABASE: AppWordsTag
CONTACT_PERSON: AppWordsTag
COUNT: AppWordsTag
COUNTRY: AppWordsTag
DATE: AppWordsTag
DELETE: AppWordsTag
DEPARTMENT: AppWordsTag
DESCRIPTION: AppWordsTag
DISCONNECT_DB_SUBMENU: AppWordsTag
DOMAIN: AppWordsTag
DOMAIN_ID: AppWordsTag
EMAIL: AppWordsTag
END: AppWordsTag
END_ASSESSEMENT: AppWordsTag
ERR_ASSESSORID_DOESNT_EXIST: AppWordsTa
AppWordsList
instance: AppWordsList
language: String
wordsList: HashMap
AppWordsList()
getInstance(): AppWordsList
getWord(in keyWord: AppWordsTag): String
initializeLanguageWordsList()
setLanguage(in language: String)

ERR_COMPAGNYID_DOESNT_EXIST: AppWordsT
ERR_LANGUAGEID_DOESNT_EXIST: AppWordsTa
ERR_NO_ALIAS: AppWordsTag
ERR_NO_ASSESSMENTLENGTH: AppWordsTag
ERR_NO_DATE: AppWordsTag
ERR_NO_DBNAME: AppWordsTag
ERR_NO_DEPARTMENT: AppWordsTag
ERR_NO_HOST: AppWordsTag
ERR_NO_NAME: AppWordsTag
ERR_NO_PURPOSE: AppWordsTag
ERR_NO_SPONSOR: AppWordsTag
ERR_NO_SPONSORRELATIONSHIP: AppWordsTa
ERR_NO_TEAMSIZE: AppWordsTag
ERR_NO_USERNAME: AppWordsTag
ERR_TITLE: AppWordsTag
EVALUATION_MENU: AppWordsTag
EVALUATION_SUBMENU: AppWordsTag
FAX_NUMBER: AppWordsTag
FILE_MENU: AppWordsTag
FIRST: AppWordsTag
FULL: AppWordsTag
FULLACCESS: AppWordsTag
HELP_MENU: AppWordsTag
INFORMATION: AppWordsTag
KEYWORDS_MGT: AppWordsTag
KEYWORDS_NAME: AppWordsTag
KPA: AppWordsTag
KPA_ID: AppWordsTag
keyWords: String
LANGUAGE: AppWordsTag
LANGUAGE_ID: AppWordsTag
LAST: AppWordsTag
LEGEND: AppWordsTag
LEVEL: AppWordsTag
LEVEL_ID: AppWordsTag
MODIFY: AppWordsTag
NAME: AppWordsTag
NEXT: AppWordsTag
NEXT_QUESTION: AppWordsTag
NO: AppWordsTag
OK: AppWordsTag
PASS: AppWordsTag
PASSWORD: AppWordsTag
PASS_TO_QUESTION: AppWordsTag
PHONE_NUMBER: AppWordsTag


POSTAL_CODE: AppWordsTag
PREVIOUS: AppWordsTag
PROCEED: AppWordsTag
PROCEED_QUESTION_TYPE: AppWordsTag
PROCESS_DOMAIN: AppWordsTag
PROCESS_ID: AppWordsTag
PURPOSE: AppWordsTag
QUESTION: AppWordsTag
QUESTIONNAIRE: AppWordsTag
QUESTION_ID: AppWordsTag
QUICK: AppWordsTag
QUIT_SUBMENU: AppWordsTag
RATINGF: AppWordsTag
RATINGFLABEL: AppWordsTag
RATINGL: AppWordsTag
RATINGLLABEL: AppWordsTag
RATINGN: AppWordsTag
RATINGNLABEL: AppWordsTag
RATINGP: AppWordsTag
RATINGPLABEL: AppWordsTag
REPRESENTATION_SUBMENU: AppWordsTag
ROAD_ID: AppWordsTag
ROAD_MAP: AppWordsTag
SAVE_CHANGES: AppWordsTag
SELECT: AppWordsTag
SHOW_ANALYSE: AppWordsTag
SPONSOR: AppWordsTag
SPONSOR_RELATIONSHIP: AppWordsTag
STATE_PROVINCE: AppWordsTag
TEAM_SIZE: AppWordsTag
USAGEACCESS: AppWordsTag
USERNAME: AppWordsTag
WORDS_MGT: AppWordsTag
YES: AppWordsTag
AppWordsTag(in tagName: String)
toString(): String

7. Enjeux économiques de la rétro-ingénierie

La rétro-ingénierie peut avoir une incidence non négligeable sur les coûts d'un logiciel. En effet, les coûts induits par les pratiques de la maintenance du logiciel peuvent atteindre entre 50 et 90 pourcents du coût total du cycle de vie d'un logiciel. Etant donné que la rétro-ingénierie peut aider considérablement la phase de maintenance en réduisant le temps de compréhension d'un logiciel et en limitant les risques engendrés par une mauvaise compréhension, développer et utiliser des méthodes efficaces s'avère nécessaire. C'est pourquoi Walt Scacchi de l'université de la Californie du Sud fit remarquer que l'ingénierie en avant et la rétro-ingénierie n'ont pas de préoccupations distinctes et qu'elles devraient donc être vues comme une opportunité de convergence, de complémentarité et de développement d'outils et techniques disponibles pour l'ingénieur logiciel moderne. Ainsi, il y a fort à parier que les technologies d'ingénierie du logiciel seront de plus en plus utilisables pour à la fois l'ingénierie en avant et la rétro-ingénierie. Cette dernière considération confirme bel et bien l'idée qui était présentée à la section 1.4, c'est à dire que les mainteneurs devraient être inclus dès la phase de conception. De cette manière, les deux sens d'ingénierie se trouvent complémentaires et convergeants vers un but commun : bien comprendre le logiciel afin de le développer efficacement et d'en réduire les coûts.

8. Description des trois révisions de $\mathcal{S}^{3}mAssess_{®}$

```
Revisions of S3MAssess carried out by Cedric Di Tomaso
<cedric.ditomaso@gmail.com>

Legend: "=>" = "issue corrected thanks to"

Revision #1
###########
+ Displaying of an error popup when a bad log of the user occurs
+ Language selection only at user log because that is unuseful
to have the possibility to have French questions later
+ Bigger space to display practices
+ Bigger space to display the four NPLF responses
+ It was impossible to select last assessments in the
Representation menu => update in real time
+ Results displayed at wrong locations => database query corrected
+ Incorrect displaying of results => correction of colours
according to results
+ In the representation interface: LEGEND: from N to F
+ Filling of results rectangles in Representation interface:
there were black spaces
+ The application crashed when the user clicked on the "next"
button without having selected a domain and a KPA
+ It was impossible to display results of the second domain
+ The Date Combo does not have many times the same date anymore
in the Representation interface
+ Correction of the displaying order of practices while the
questionnaire
+ Succession of practices in the questionnaire: do not go down
in levels but go up (from level 0 to 2). To begin with lower
levels of questions is easier, more pertinent and more reliable
to carry out an assessment
+ The rating for the level 1 of practices is NPLF and not "Yes
or No"
+ Deletion of the "File" menu because it was impossible to quit
the applet running in a browser
+ Displaying of an information popup at the end of a kpa
assessment. The user can know which possibilities are offered at
that time and what is the assessment identifier, which is
necessary to analyse it
+ Encoding in the database of all English practices (196
practices). A complete assessment following the first three
levels of the S3M model can now be carried out
+ Cleaning of the database: initial data's are corrected
(identifiers, model values, ...)
+ Adding of foreign keys to preserve the database integrity
+ Correction of some buttons' names
+ Adding of some columns in the database: "domain" and "kpa" in
the 'question' table

Revision #2
###########
```

+ Creation of an installation guide in a pdf file
+ Display of the practice identifier in the questionnaire interface
+ When a first practice was not assessed, the user could not see a black segment in the graphic (mistake in the output) => the user has to answer something to practices (via a popup)
+ Adding of some columns in the database
+ Adding of new integrity constraints to preserve the database integrity
+ New functionality: the user can export all assessment information and results (practice's percentage, domain's level percentage, suggestion of practices that should be improved, ...) in a spreadsheet
+ Display of the identifier and the comment related to its practice in the graphic output via a tooltip text
+ Deletion of the admin menu (the code is just hidden for later maintenance activity)
+ Correction of the back button in the setting question interface: now we can modify the assessment information and then proceed with the assessment without crashing the application and in keeping the integrity of the database
+ Opportunity to create a new assessment at any moment
+ Prevention to edit Combos in the Setting Analyse, Setting Question, Analyze and Assessment interfaces because that was unuseful and that crashed the application
+ Possibility to add a new company and a new assessor in the Assessment interface
+ Deletion of the Assessment Type selection in the Assessment interface: unuseful
+ Possibility to skip over a practice from the assessment of a KPA (via the "Not applicable" option)
+ Improvement of the interface: application words, components' positions and tool tips to help the user
+ Adding of the functionality to sort the results for the spreadsheet
+ Increasing of the application window size to 800x500
+ Display of each domain's level percentage in the Analysis interface
+ New tabs in the analysis containing assessment information and improvement suggestions
+ Adding of the capability to scroll comments entered in the questionnaire interface. Before, the text was hidden
+ Display of graphic results in light 3D: easiest to read the results
+ Adding of a type control and a length control for fields in the Assessment Information interface. We could pass to the Assessment interface without recording the assessment information into the database
+ Adding of buttons "Cancel" and "Validate" to cancel or modify assessment information
+ Adding of the possibility to modify a KPA already assessed

while the assessment procedure. Before, new changes were ignored
+ Confidentiality of information between logged users since the application is running on a web server
+ Adding of a "Revisions" menu that shows that file
+ Creation of a S3MAssess certificate that the user has to accept to be able of running the applet. That certificate allows S3MAssess to write the spreadsheet containing assessment results on the user's hard disk

Revision #3
###########
+ Fixing of Java exceptions that occurred when the database was reinitialized
+ Improvement of the tool start and of the analysis start by optimizing workflows and interactions with the database
+ Opportunity to download the user guide from the About interface
+ Adding of an "NPLF" rating scale on the interface while the procedure of assessment
+ Fixing of the application crash that occurred after some analysis selections
+ Adding of the S3MAssessment method
+ Adding of new analysis graphs: Domains Histogram, KPA Histogram and Multi-assessment Histogram
+ Fixing of output issues: reinitialization of graphs
+ Practices of level 0 are now presented in the positive form and "Yes" is rated at 100%, "No" is rated at 0% + correction in the user's data's already encoded in the database
+ Adding of the possibility to select and modify a previous assessment
+ Fixing of errors due to special characters entered by the user and improvement of the security at the same time
+ Improvement of the feedback given to the user

9. Présentation des intervenants

Le chapitre 5 présentait $S^{3mAssess®}$, notamment à qui il s'adresse en priorité et à quelles fins. Au cours du stage effectué à Montréal, la maintenance de l'outil a principalement été réalisée pour trois intervenants principaux : Dr Alain April, la multinationale Freescale et dans une moindre mesure pour IBM Australie. Les modes de communications utilisés furent via : conférence téléphonique, réunion, échange de courriers électronique.

Une brève présentation des intervenants ayant influencé l'évolution de $S^{3mAssess®}$ est donnée ici :

- Dr Alain ABRAN : actuellement professeur titulaire au département de génie électrique de l'École de Technologie Supérieure (E.T.S.) du réseau de l'Université du Québec. Il est le directeur du Laboratoire accrédité de recherche en génie logiciel – GÉLOG de l'ETS et il a dirigé plusieurs travaux de recherche pour l'industrie privée, pour des associations professionnelles et pour divers ministères aux niveaux provincial et fédéral. Le Dr. Abran a obtenu son doctorat de l'École Polytechnique de Montréal et est diplômé de l'Université d'Ottawa en informatique, en génie électrique (M.Sc.) et en administration (M.Sc.Gestion). Il est aussi le co-éditeur exécutif du projet sur le corpus des connaissances en génie logiciel (SWEBOK - Software Engineering Body of Knowledge). Le Dr. Abran représente également le Canada sur plusieurs groupes de travail ISO dans le domaine des normes internationales en génie logiciel. De plus, plusieurs des résultats des travaux du Dr. Abran sont devenus par la suite des documents officiels ISO soit comme normes internationales, soit comme rapports techniques officiels publiés par ISO.
  Le programme de mesures qu'il a conçu et implanté dans le secteur de la maintenance des logiciels au Montréal Trust a gagné en 1993 un prix "Best of the Best" du Quality Assurance Institute aux États Unis.

- Dr Alain APRIL : il s'agit du maître de stage. Alain April est Professeur de Génie Logiciel et un membre du Laboratoire de Recherche en Génie Logiciel à l'École de Technologie Supérieure (ETS) - Université du Québec. Professeur April a plus de 25 années d'expérience en industrie dans les domaines des systèmes d'informations, de la qualité et du génie logiciel.

- Dr Jean-Marc DESHARNAIS : œuvre dans le domaine de l'informatique depuis plus de 30 ans. Il a développé une expertise en informatique (analyste et chef de projet), puis en génie logiciel (consultation et recherches). Depuis 1987, J.M. Desharnais développe une expertise en gestion de projet et plus particulièrement en mesure du logiciel.

- Christian HUVELLE : Manager IT, Chef de projet pour la compagnie Siemens à Jambes (Namur). Il a été un contact déterminant en ce qui concerne l'étude de cas présentée dans la partie III.

- Caroline MILAM : Domain Manager pour la société Freescale au Texas.

- SIEMENS Belgique : est un des leaders dans les secteurs de l'Information et des Communications, de l'Industrie, de l'Energie, de la Lumière, du Transport, de la Conception des Technologies et des Solutions Médicales. Site web : http://www.siemens.be

- Laxman VASAN attaché aux services business globaux pour IBM en Australie.

10. Exportation des résultats de l'évaluation de Siemens

ASSESSMENT INFORMATION

Assessment Number	19
Company	Siemens
Assessor	Christian Huvelle
Date	2008-07-15
Assessment Length (days)	1
Department	Organisation de maintenance de Siemens pour un projet de la Commission Européenne
Team Size	4
Purpose	Projet de maintenance pour la direction générale d'une administration européenne
Sponsor	Chef de projet au sein de l'organisation de maintenance de Siemens
Sponsor Relationship	Comprendre le fonctionnement du modèle S3M tout en évaluant ses processus de maintenance

PRACTICES DETAILED

PRACTICES (sorted by Asse	RATING	% CO	COMMENT
Pro1.0.1	Yes	100	
Pro1.1.1	Largely achieved	68	discussions
Pro1.1.2	Largely achieved	68	
Pro2.0.1	Yes	100	
Pro2.1.1	Fully achieved	93	
Pro2.1.2	Fully achieved	93	
Pro2.2.1	Fully achieved	93	
Pro2.2.2	Fully achieved	93	
Pro2.2.3	Fully achieved	93	
Pro2.2.4	Largely achieved	68	QP + validationpostes ou ce n est pas fait
Pro3.0.1	Yes	100	transition n existe pas
Pro3.1.1	Fully achieved	93	
Pro3.1.2	Largely achieved	68	cas d une personne seule sur une application specifique
Pro3.1.3	Fully achieved	93	
Pro3.2.1	Fully achieved	93	
Pro3.2.2	Fully achieved	93	
Pro3.2.3	Largely achieved	68	pas applique aux externes
Pro3.2.4	Largely achieved	68	pas pour les externes
Pro3.2.5	Largely achieved	68	pas externes
Pro3.2.6	Largely achieved	68	schedules pas universels
Pro3.2.7	Largely achieved	68	pour les externes ce n est pas maitrise
Pro3.2.8	Largely achieved	68	
Pro3.2.9	NOT APPLICABLE		pas de transition
Pro3.2.10	NOT APPLICABLE		idem
Pro3.2.11	NOT APPLICABLE		par IIRM, hors scope
Pro3.2.12	NOT APPLICABLE		idem
Pro3.2.13	Fully achieved	93	
Pro4.0.1	Yes	100	
Pro4.1.1	Fully achieved	93	
Pro4.1.2	Fully achieved	93	
Pro4.2.1	Fully achieved	93	
Pro4.2.2	Fully achieved	93	
Pro4.2.3	Fully achieved	93	
Pro5.0.1	Yes	100	
Pro5.0.2	Yes	100	
Pro5.0.3	Yes	100	
Pro5.1.1	Largely achieved	68	
Pro5.1.2	Largely achieved	68	
Pro5.1.3	Fully achieved	93	
Req1.0.1	Yes	100	

Req1.1.1	Fully achieved	93	
Req1.1.2	Fully achieved	93	
Req1.2.1	Fully achieved	93	
Req1.2.2	Fully achieved	93	
Req1.2.3	Fully achieved	93	
Req1.2.4	Fully achieved	93	
Req2.0.1	Yes	100	
Req2.1.1	Fully achieved	93	
Req2.1.2	Fully achieved	93	
Req2.1.3	Fully achieved	93	
Req2.2.1	Fully achieved	93	
Req2.2.2	Fully achieved	93	
Req2.2.3	Fully achieved	93	
Req2.2.4	Fully achieved	93	
Req2.2.5	Partially achieved	33	plutot reactif que proactifDIGIT pousse aux changements car contexte contractuel
Req2.2.6	NOT APPLICABLE		Facette Transition
Req2.2.7	NOT APPLICABLE		
Req2.2.8	NOT APPLICABLE		
Req2.2.9	NOT APPLICABLE		
Req2.2.10	NOT APPLICABLE		
Req2.2.11	NOT APPLICABLE		
Req2.2.12	NOT APPLICABLE		pas de disaster recovery
Req2.2.13	Fully achieved	93	
Req2.2.14	Not achieved	0	pas teste
Req2.2.15	Fully achieved	93	
Req2.2.16	Fully achieved	93	
Req2.2.17	Fully achieved	93	
Req2.2.18	Fully achieved	93	
Req2.2.19	Fully achieved	93	
Req2.2.20	Fully achieved	93	
Req3.0.1	Yes	100	
Req3.1.1	Fully achieved	93	
Req3.2.1	Fully achieved	93	
Req3.2.2	Fully achieved	93	
Req3.2.3	Partially achieved	33	pas hebdomadaire, mensuel
Req3.2.4	Largely achieved	68	pas pour tous les projets ou personnes impliquees
Req3.2.5	NOT APPLICABLE		
Req3.2.6	Largely achieved	68	pas toujours renegocie
Req4.0.1	Yes	100	
Req4.1.1	Fully achieved	93	
Req4.2.1	NOT APPLICABLE		pas le choix: cahier des charges a respecter
Req4.2.2	Fully achieved	93	
Req4.2.3	NOT APPLICABLE		idem Req4.2.1
Req4.2.4	Fully achieved	93	
Req4.2.5	Fully achieved	93	
Req4.2.6	Fully achieved	93	
Req4.2.7	Fully achieved	93	
Req4.2.8	Largely achieved	68	pas pour toutes les applications
Req4.2.9	Fully achieved	93	
Req4.2.10	NOT APPLICABLE		pas dans le contexte
Req4.2.11	NOT APPLICABLE		contexte d outsourcing
Req4.2.12	Fully achieved	93	
Evo1.0.1	NOT APPLICABLE		
Evo1.1.1	NOT APPLICABLE		
Evo1.2.1	NOT APPLICABLE		
Evo1.2.2	NOT APPLICABLE		

Evo1.2.3	NOT APPLICABLE		
Evo1.2.4	NOT APPLICABLE		
Evo1.2.5	NOT APPLICABLE		
Evo1.2.6	NOT APPLICABLE		
Evo1.2.7	NOT APPLICABLE		
Evo1.2.8	NOT APPLICABLE		
Evo1.2.9	NOT APPLICABLE		
Evo1.2.10	NOT APPLICABLE		
Evo1.2.11	NOT APPLICABLE		
Evo2.0.1	Yes	100	
Evo2.1.1	Fully achieved	93	
Evo2.2.1	Fully achieved	93	
Evo2.2.2	Fully achieved	93	
Evo2.2.3	Fully achieved	93	
Evo2.2.4	Fully achieved	93	
Evo2.2.5	Fully achieved	93	
Evo2.2.6	Fully achieved	93	
Evo3.0.1	Yes	100	
Evo3.1.1	Fully achieved	93	
Evo3.2.1	Fully achieved	93	
Evo3.2.2	Fully achieved	93	
Evo3.2.3	Fully achieved	93	
Evo3.2.4	Fully achieved	93	
Evo3.2.5	Fully achieved	93	
Evo3.2.6	Largely achieved	68	pas toujours pour bug fixing
Evo3.2.7	Largely achieved	68	toutes les applications ne sont pas couvertespas de tests systematiques pour toutes nouvelles prc
Evo3.2.8	Fully achieved	93	
Evo3.2.9	Partially achieved	33	tard: apres les essais
Evo3.2.10	Partially achieved	33	pas pour toutes les applications
Evo3.2.11	Fully achieved	93	
Evo4.0.1	Yes	100	
Evo4.1.1	Fully achieved	93	
Evo4.2.1	Fully achieved	93	
Evo4.2.2	Partially achieved	33	documentation review standard mais pas de code review systematique
Evo4.2.3	Fully achieved	93	
Evo4.2.4	Fully achieved	93	
Evo4.2.5	Largely achieved	68	pas pour toutes les parties pour raison de ROI
Evo4.2.6	Fully achieved	93	
Evo4.2.7	Fully achieved	93	
Evo4.2.8	Fully achieved	93	
Sup1.0.1	Yes	100	
Sup1.1.1	Fully achieved	93	
Sup1.2.1	Fully achieved	93	
Sup1.2.2	NOT APPLICABLE		pas de processus de transition
Sup1.2.3	Fully achieved	93	
Sup1.2.4	Fully achieved	93	
Sup1.2.5	Fully achieved	93	
Sup1.2.6	Fully achieved	93	
Sup1.2.7	Largely achieved	68	pas toutes les applications (pas de lien auto)
Sup1.2.8	Not achieved	0	
Sup2.0.1	Yes	100	
Sup2.1.1	Fully achieved	93	
Sup2.2.1	Fully achieved	93	
Sup2.2.2	Fully achieved	93	
Sup2.2.3	Fully achieved	93	
Sup2.2.4	Fully achieved	93	

Sup2.2.5	Fully achieved	93	
Sup2.2.6	Largely achieved	68	pas regulierement
Sup2.2.7	Fully achieved	93	
Sup2.2.8	Largely achieved	68	pas pour tout
Sup2.2.9	Partially achieved	33	uniquement dans le cas de developpement d importantes fonctionnalites
Sup2.2.10	Fully achieved	93	
Sup2.2.11	Largely achieved	68	produit mais pas forcement presente
Sup2.2.12	Not achieved	0	rien de prevu periodiquement
Sup2.2.13	Fully achieved	93	
Sup3.0.1	Yes	100	
Sup3.1.1	Partially achieved	33	tous n ont pas la connaissance
Sup4.0.1	Yes	100	
Sup4.1.1	Partially achieved	33	lessons learned et Defect Prevention Process realise dans certains cas bien precis
Sup5.0.1	Yes	100	limites contractuelles importantes
Sup5.1.1	Partially achieved	33	si pas d impact sur les couts et planning sinon negociation avec le client (scope)
Sup5.1.2	Fully achieved	93	

IMPROVEMENT SUGGESTION: Domain level percentage ==> Rating terminology ==> Practices suggestion

Domain 1 Level 0	100 ==> Fully Achieved ==>
Domain 1 Level 1	83 ==> Largely Achieved ==> Pro1.1.1, Pro1.1.2, Pro3.1.2, Pro5.1.1, Pro5.1.2
Domain 1 Level 2	51,22 ==> Largely Achieved ==>
Domain 2 Level 0	100 ==> Fully Achieved ==>
Domain 2 Level 1	93 ==> Fully Achieved ==>
Domain 2 Level 2	83,78 ==> Largely Achieved ==> Req2.2.5, Req2.2.14, Req3.2.3, Req3.2.4, Req3.2.6, Req4.2.8
Domain 3 Level 0	100 ==> Fully Achieved ==>
Domain 3 Level 1	93 ==> Fully Achieved ==>
Domain 3 Level 2	84,31 ==> Largely Achieved ==> Evo3.2.6, Evo3.2.7, Evo3.2.9, Evo3.2.10, Evo4.2.2, Evo4.2.5
Domain 4 Level 0	100 ==> Fully Achieved ==>
Domain 4 Level 1	63 ==> Largely Achieved ==> Sup3.1.1, Sup4.1.1, Sup5.1.1
Domain 4 Level 2	30,32 ==> Partially Achieved ==> Sup1.2.8, Sup2.2.12

11. Technologies employées

La langage Java

Une autre qualité de Java est qu'il permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur. Côté client, les applets sont à l'origine de la notoriété du langage. C'est surtout côté serveur que Java s'est imposé dans le milieu de l'entreprise grâce aux servlets, le pendant serveur des applets, et plus récemment les JSP (JavaServer Pages) qui peuvent se substituer à PHP, ASP et ASP.NET.
Les applications Java peuvent être exécutées sur tous les systèmes d'exploitation pour lesquels a été développé une plate-forme Java, dont le nom technique est *JRE* (*Java Runtime Environment* – Environnement d'exécution Java). Cette dernière est constituée d'une *JVM* (*Java Virtual Machine* - Machine Virtuelle Java), qui est un programme interprétant le code Java et le convertissant en code compréhensible pour la machine sur laquelle il s'exécute. C'est la garantie de portabilité qui a fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, très difficile pour les gros systèmes.

Applet Java

A la différence d'un code html ne présentant qu'une interface de présentation d'informations, une applet est capable de les traiter. De cette manière, les communications réseau et le volume de travaux - ou *charge* - imposés au serveur diminue permettant d'accroître la fluidité de l'exécution en évitant de longs temps de latence. Les ressources mobilisées pour le traitement sont dès lors celles du poste client étant donné qu'une applet est un fichier Java compilé en un fichier *Class* (un langage compréhensible par un ordinateur) référencé par le fichier html auquel le client se connecte via une URL. Un autre aspect positif par rapport à un code html est qu'une applet améliore l'ergonomie de l'application via les nombreuses possibilités que les interfaces graphiques (issues de Java dans ce cas-ci) proposent. Pour cela, tous les navigateurs Internet abritent de nos jours un environnement d'exécution : le Java Runtime Environment. C'est grâce à cet environnement d'exécution intégré au navigateur qu'un client peut exécuter du code Java et ce, sans devoir installer l'application en elle-même. Plus précisément, une applet est un code de programme que le serveur a expédié au client lorsque celui-ci a accédé à l'URL de l'application. Dès cet instant, l'applet $S^{3mAssess®}$ assure le traitement des données, leur présentation et l'interaction entre l'utilisateur et le serveur.

Cependant, il est nécessaire de mentionner le fait que dans le cas de $S^{3mAssess®}$, l'utilisateur devra au préalable accepter un certificat, que le serveur lui proposera, s'il veut pouvoir utiliser l'outil. En effet, une applet ne possède que les droits du navigateur lui permettant au plus l'accès aux ressources fournies par ce dernier. Or, ces droits sont insuffisants pour des instructions d'Entrée/Sortie tel l'accès à un fichier sur le disque dur. Ce type d'instruction est notamment présent dans l'application de part la possibilité de créer une feuille de calcul suite à une évaluation de maturité. Cette règle de sécurité préalable est donc modifiée grâce à l'utilisation d'un certificat permettant d'obtenir l'accord explicite de l'utilisateur afin d'activer une nouvelle règle de sécurité.

Enfin, le fait d'apposer une signature numérique à l'applet (nécessaire à la création du certificat) offre aussi à son utilisateur l'assurance que ce qu'il a téléchargé provient bien de l'auteur et n'a pas été modifié. Toutefois, pour en avoir une assurance encore plus grande, il serait nécessaire qu'une autorité de certification, VeriSign par exemple, signe ce certificat. Cela pourrait-être envisagé dans une version future de l'outil si celui-ci s'industrialisait.

Une base de données relationnelle

A la différence d'une base de données objet, une BD relationnelle ne doit pas charger tous les objets d'un certains type lorsque seule une information bien spécifique est requise.
Un autre problème d'une BD objet est la façon dans laquelle elle marque les objets comme *persistant*.

Certaines BD objet ajoutent du code additionnel dans un objet ou dans son code compilé. Bien que ce code peut ne pas affecter la capacité d'exécuter ces objets dans d'autres JVMs, il y a le problème que cela viole l'idéal Java suivant : "*Ecrire une fois, exécuter partout*".
Enfin, il existe également le problème que peu d'entreprises et peu de développeur utilisent de nos jours ce type de BD. Cela induit des problèmes de portabilité d'une part et de maintenance d'autre part car il est plus difficile de former une équipe apte à gérer efficacement son processus de maintenance avec une BD que peu maîtrisent.

Face à ces considérations, une base de données relationnelle représente une technologie plus mature. Elle a été développée dès les années 70 par de puissantes sociétés comme IBM, Oracle ou même Microsoft. Ces années d'élaboration lui permettent aujourd'hui d'offrir un certain niveau de performance et de stabilité. Ainsi, les données sont groupées en fonction de leur lien et d'une façon permettant leur accès rapide. Il est possible d'y accéder à partir des tables de données constituant la base de données ou à partir de techniques plus avancées comme les *vues* (permettant de regrouper logiquement des tables stockées physiquement afin de traiter plus facilement et rapidement les données). Au niveau de la stabilité, les données sont enregistrées sur le disque dur jusqu'à ce qu'il en soit décidé autrement et il est possible d'utiliser ce qu'on appelle des *transactions*. Ces dernières sont une caractéristique que les bases de données relationnelle les plus sérieuses supportent. C'est une façon de grouper des travaux afin de s'assurer qu'ils ont été entièrement complétés ou alors pas du tout. De cette manière, la base de données n'entre jamais dans un état incohérent.

Pour terminer cette section, il est à préciser que si une application comme $S^{3mAssess®}$ veut communiquer avec une telle BD, elle doit utiliser un driver. Dans ce cas-ci, le driver utilisé est *JDBC* qui est une API (une interface de programmation proposant des services) permettant une communication cohérente et facile avec une base de données à partir de Java. De plus, avec JDBC il est possible d'écrire du code Java pouvant fonctionner avec de multiples bases de données, ce qui peut être utile en phase de maintenance.

Accès à distance

Accès distant VPN

Une infrastructure VPN est nécessaire car elle permet aux deux entités de communiquer de manière sécurisée malgré le fait qu'elles sont éloignées géographiquement et que leurs données échangées passent par des infrastructures publiques et appartenant à plusieurs opérateurs.
Cela fonctionne de la manière suivante : après qu'un algorithme de cryptage ait été choisi, le client VPN chiffre ses données et les envoie au serveur VPN qui les déchiffre ensuite selon l'algorithme choisi si la vérification de ce client est satisfaisante. Un réseau privé virtuel vise à apporter certains éléments essentiels dans la transmission de données : l'authentification (et donc l'identification) des interlocuteurs, l'intégrité des données (le chiffrement vise à les rendre inutilisables par quelqu'un d'autre que le destinataire) ou encore la cohérence des données transmises. Ce système VPN permet donc d'obtenir une liaison sécurisée à moindre coût, si ce n'est la mise en oeuvre des équipements terminaux. En contrepartie il ne permet pas d'assurer une qualité de service comparable à une ligne louée dans la mesure où le réseau physique est public et donc non garanti.

Un bon compromis consiste à utiliser Internet comme support de transmission en utilisant un protocole d'"encapsulation" (en anglais tunneling), c'est-à-dire encapsulant les données à transmettre de façon chiffrée.
Ce réseau est dit virtuel car il relie deux réseaux "physiques" (réseaux locaux) par une liaison non fiable (Internet), et privé car seuls les ordinateurs des réseaux locaux de part et d'autre du VPN peuvent "voir" les données.

Le protocole de tunnelisation (tunneling) permet aux données passant d'une extrémité du VPN à l'autre d'être sécurisées par des algorithmes de cryptographie. Ici, ces algorithmes sont : 56-bit DES pour le chiffrement et HMAC-MD5 pour l'authentification.

Zone démilitarisée DMZ

Lorsque certaines machines d'un réseau interne ont besoin d'être accessibles de l'extérieur (comme c'est le cas du serveur HTTP et MySQL), il est souvent nécessaire de créer une nouvelle interface vers un réseau à part, accessible aussi bien du réseau interne que de l'extérieur, sans pour autant risquer de compromettre la sécurité de l'organisation. On parle ainsi de "zone démilitarisée" (notée DMZ pour DeMilitarized Zone) pour désigner cette zone isolée hébergeant des applications mises à disposition du public. La DMZ fait ainsi office de "zone tampon" entre le réseau à protéger et le réseau hostile.

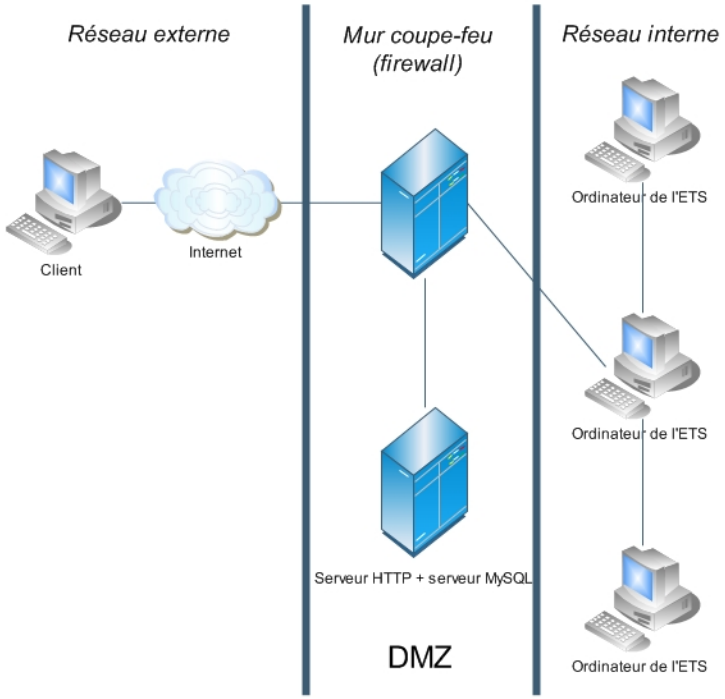


FIG. 9.2 – Zone démilitarisée DMZ à l'ETS

La politique de sécurité est la suivante :
- Trafic du réseau externe vers la DMZ autorisé ;
- Trafic du réseau externe vers le réseau interne interdit ;
- Trafic du réseau interne vers la DMZ autorisé ;
- Trafic du réseau interne vers le réseau externe autorisé ;
- Trafic de la DMZ vers le réseau interne interdit ;
- Trafic de la DMZ vers le réseau externe refusé.

La zone DMZ possède donc un niveau de sécurité intermédiaire mais son niveau de sécurisation n'est pas suffisant pour y stocker les données critiques d'une entreprise par exemple. Dans le cas de $S^{3mAssess®}$, le niveau de sécurité proposé par la DMZ est pour le moment suffisant mais il serait envisageable de faire tourner le serveur de base de données, contenant les informations de toutes les sociétés, au sein du réseau interne de l'ETS si l'outil venait à grandir davantage. Cela permettrait de garantir un maximum de sécurité concernant par exemple les données d'évaluation des processus de maintenance de sociétés concurrentes.

12. Détails théoriques sur les design patterns utilisés par la couche OR

Singleton Pattern

Tout d'abord, le Singleton Pattern est un design pattern utilisé pour restreindre l'*instanciation* d'une classe à un objet. L'instanciation est l'action d'instancier, de créer un objet. Elle est réalisée par la composition de deux opérations : l'allocation et l'initialisation. L'allocation consiste à réserver un espace mémoire au nouvel objet. L'initialisation consiste à fixer l'état du nouvel objet. Cette opération fait par exemple appel à l'un des constructeurs Java de la classe de l'objet à créer.
Le fait de restreindre est utile quand exactement un objet est nécessaire pour coordonner des actions à travers le système. Cela est même parfois généralisé à des systèmes qui opèrent plus efficacement quand seulement un ou quelques objets existent.

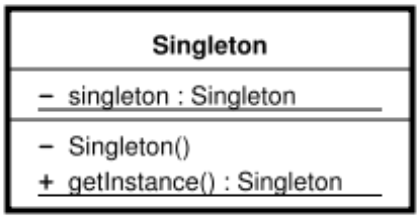


FIG. 9.3 – Diagramme de classe d'un singleton

Le pattern singleton est implémenté en créant une classe avec une méthode qui crée une nouvelle instance de cette classe à la condition qu'une instance ne soit pas déjà créée. Si une instance existe déjà, la méthode retourne simplement une référence à cet objet.

Command Pattern

Un deuxième pattern utilisé est le Command Pattern. Il s'agit d'un design pattern dans lequel les objets sont utilisés pour représenter des actions. Un objet **Commande** encapsule une action ainsi que ses paramètres. Voici quelques propriétés propres à ce pattern :

- Un objet Commande est adapté au stockage temporaire des paramètres des procédures. Il peut être utilisé pour l'assemblage des paramètres pour un appel de fonction et pour permettre l'utilisation ultérieure d'une commande.
- Une classe est un endroit propice pour collecter du code et des données liés à une commande. Un objet commande peut détenir des informations à propos de la commande, tels son nom ou quel utilisateur l'a lancée et répondre à des questions comme combien de temps va-t-elle probablement durer.
- Traiter les commandes comme des objets permet des structures de données contenant de multiples commandes. Ces commandes pourraient ensuite être gérées via une file d'attente.
- Traiter les commandes comme des objets permet également de supporter des opérations réversibles, si toutefois les objets sont enregistrés (dans un stack par exemple).

La structure théorique de ce design pattern est la suivante est illustré à la figure 9.4.
Dans la littérature, la terminologie employée peut parfois être incohérente et donc amener de la confusion. Ceci est dû au fait qu'il est possible d'implémenter ce pattern de différentes manières et qu'il existe plusieurs combinaisons possibles de ce pattern avec d'autres. Toutefois, la terminologie suivante sera utilisée ici afin d'exprimer le plus clairement possible la figure 9.4 :

- Client : la source de la commande. Par exemple : un bouton pressé par l'utilisateur.

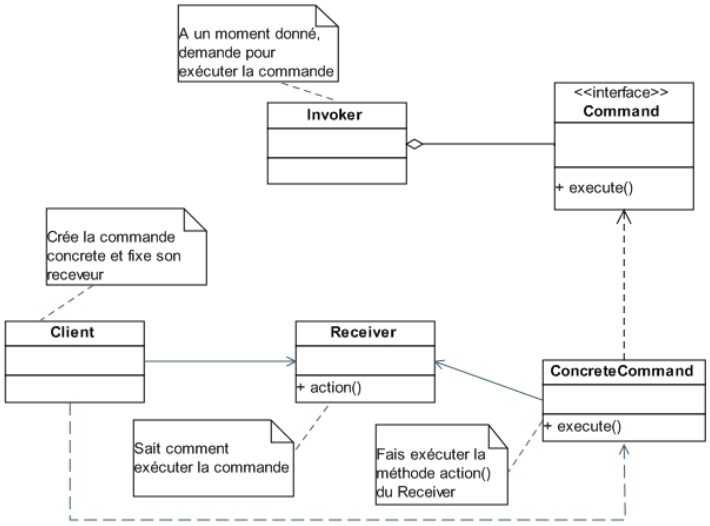


FIG. 9.4 – Structure du Command Pattern

- Invoker : un objet singleton qui possède des informations à propos de la commande.
  Le but de cette classe est de gérer un tel scénario : un objet Client appelle la méthode execute() de l'objet Commande. Ensuite, l'objet Commande notifie les objets Client appropriés quand la commande a généré quelque chose. L'Invoker peut gérer une liste de commandes.
- Receiver ou receveur de la commande : l'objet qui doit traiter réellement des données en fonction de la commande qu'il a reçue.
- L'objet Commande concrète : objet passant de la source de la commande au code de l'objet cible exécutant le travail. La commande peut passer par des objets et méthodes intermédiaires.

Event Pattern

Le troisième et dernier pattern utilisé est l'Event Pattern. Dans la structure *Event*, les "Listeners" ou "Ecouteurs" des événements peuvent être considérés comme des objets Commande responsables d'exécuter les commandes dans l'application dès que l'utilisateur manipule l'interface graphique. Les listeners/commandes font partie des fonctionnalités de l'application mais pas de l'interface. En d'autres mots, ils sont les contrôleurs formant une couche entre la couche Présentation et la couche Business.
Le but ici est d'éviter un contrôle non orienté objet des méthodes. Il faut pouvoir ajouter des éléments d'interface graphique sans devoir modifier du code existant. Les objets Commande (les listeners) qui exécutent les instructions des utilisateurs doivent être associés au modèle de l'application et ne pas faire partie de la structure de la couche Présentation.

Un composant par Listener

Lorsqu'un récepteur – listener – d'événements est attribué à un seul composant (par exemple graphique) à écouter, les méthodes de la classe Listener ne traiteront les événements que pour ce composant. Donc, dans le cas où il y a deux composants, il faut avoir deux listeners différents, un pour chaque composant. Utiliser des listeners séparés signifie qu'il n'est pas nécessaire d'effectuer des tests booléens pour déterminer ce qu'il devrait être fait dans chaque listener. Cela a deux avantages. Le premier est que cela permet d'exécuter le code un peu plus rapidement. Le second est que cela améliore la maintenabilité et l'extensibilité du code. Il n'est dès lors plus nécessaire de modifier des méthodes existantes pour ajouter un nouveau composant.
Il peut être précisé que les listeners n'ont pas besoin d'être nommés. Il suffit de les créer comme des

arguments dans les méthodes d'ajout de listeners.

Listeners héritiers

Dans le cas où une ou plusieurs classes réceptrices d'événements doivent être écrites, celles-ci devraient hériter de la classe qui contient le composant qu'elles écoutent. Cela permet de garder la définition du récepteur locale à l'endroit où se trouve le composant à écouter et d'éviter de devoir élargir la visibilité du composant. La classe Listener obtient ainsi un accès aux autres parties des classes avoisinantes (qui appartiennent au même paquetage) qui peuvent être nécessaires pour traiter l'événement. En retour, le nombre de paramètres nécessaires est limité quand l'objet récepteur est construit. Toutefois, le couplage entre les classes avoisinantes et la classe réceptrice se voit augmenté par la même occasion. Dès lors, si une classe voisine est révisée, il pourrait être nécessaire de réviser également la sous-classe réceptrice héritant de la classe contenant le composant à écouter.

Cependant, ce pattern a quelques conséquences négatives. Il n'est réellement utile que quand il n'y a qu'une seule classe implémentant l'interface graphique et quand le nombre d'événements possibles est limité. S'il y a de multiples interfaces graphiques pour la même application, il peut être difficile de partager les commandes à exécuter suite à un événement étant donné que les commandes sont au sein des composants graphiques. Si le composant doit répondre à de multiples événements, alors ces classes deviennent lourdes et peu maniables en encapsulant beaucoup trop en un endroit qui devrait probablement être séparé. En séparant les commandes des composants, les commandes pourraient être partagées avec d'autres composants devenus ainsi plus légers et donc plus maniables.