MASTER EN SCIENCES INFORMATIQUES

Contribution à l'acquisition de spécifications formelles pour un problème d'investissement

Martens, Arnaud

Award date:
1992

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

CONTRIBUTION À L'ACQUISITION DE SPECIFICATIONS FORMELLES POUR UN PROBLEME D'INVESTISSEMENT

ARNAUD MARTENS

Promoteur :

Professeur E. Dubois

Mémoire présenté en vue
de l'obtention du titre
de Licencié et Maître
en Informatique

Année académique1991-1992

RESUME

Ce mémoire a pour but d'illustrer l'étape d'ingénierie des besoins dans le cycle de développement d'un outil informatique support à un problème d'investissement. Ce document se concentre sur la phase d'élicitation des besoins et de la modélisation. La phase d'élicitation a été effectuée au travers de l'étude de différentes interfaces permettant de mieux comprendre la nature des interactions prenant place entre l'investisseur et son logiciel. La modélisation proprement dite est exprimée dans un langage formel de spécification appelé GLIDER. La structuration de cette spécification est faite selon le paradigme orienté objet.

ABSTRACT

This thesis describes the Requirements Engineering step in the context of the life cycle of the devellopment of a data processing tools supporting an investmnent problem. This paper focuses on the elicitation and modelling phases. The elicitation phase has been performed through the evaluation of several interfaces witch allowed to understand the nature of interactions between user and sofware. The modelling phase has been handled using a formal language specification called GLIDER. The structuring of this specification follows on Object-Oriented paradigm.

REMERCIEMENTS

J'aimerais remercier ici les personnes qui m'ont apporté leur aide pour la bonne réalisation de ce mémoire et l'énorme ouverture d'esprit que cela apporte.

Bien que des médailles de remerciement pourraient être distribuées tel que l'ordre du grand remerciement, je voudrais remercier Mlle. K. Becker, le Professeur.E.Dubois, M. Ph. Du Bois, le Professeur M. Guillaume, le Professeur H. Leroy, M. S. Tourneur, le Professeur G. Schepens ainsi que mes parents pour l' aide très précieuse apportée à la résolution du problème posé.

Merci également aux étudiants, habitués du pool de Macintosh du 2ième, qui ont su créer une ambiance stimulante et "anti-glucose" indispensable au maintien de notre équilibre psycho-neuro-physio-informatique lors de nos longues journées de travail acharné.

TABLE DES MATIERES

Introduction..... 1

Chapitre 1. Les différentes vues d'un problème d'investissement 3

1.1. Délimitation du problème de l'investissement vu de manière philosophique..... 3
1.2. Rappel théorique de la théorie de l'investissement..... 5
1.3. Processus intellectuel de l'investisseur de type gestionnaire..... 6
1.4. Les différentes visions pour résoudre un problème d'investissement 8

Chapitre 2. le processus de conception..... 25

2.1. Introduction..... 25
2.2. LeProcessus de conception d'un point de vue théorique. 26
2.3. Processus de conception d'un point de vue pratique..... 29

Chapitre 3. Spécification du problème de l'investissement 38

3. 1. Présentation théorique des spécifications. 38
3. 2. Présentation pratique des spécifications. 41
3.2.1. Définition de la structure de données 41
3.2.2. Spécification du problème de l'investissement 43
TABLEAU 45
CORPSTABLEAU..... 60
LIGNE 99
LIGNE-DONNEES 100
LIGNE DE VALEUR 100
LIGNE DE TEMPS 104
EXPRESSSION..... 105
LIGNE DE FORMULE..... 108
LIGNE FINANCIERE 109
LIGNE AMORTISSEMENT DEGRESSIF..... 111

LIGNE RESIDUELLE 114
LIGNE IMPOT 115
LIGNE AMORTISSEMENT LINEAIRE 115
LIGNE TIR 117
LIGNE BENEFICE IMPOSABLE 119
3.2.3. Glossaire des spécifications 121

Conclusion 127
Bibliographie 130

Introduction.

Ce mémoire traite d'une application supportant l'acquisition et la spécification d'un problème d'investissement orienté objet. Cependant, le développement de l'application informatique est limité à l'étape d'ingénierie des besoins (étape préliminaire du cycle de vie de développement) et l'implémentation n'est pas considérée (étape d'ingénierie de logiciel "software engineering").

L'ingénierie des besoins [Dub91b] a pour but de déterminer les besoins du client aussi précisément que possible et cela à différents niveaux tel que les fonctionnalités attendues de son système, les délais de développement etc; le tout constituant le cahier des charges. L'activité d'ingénierie des besoins peut se décomposer en 4 phases qui forment un processus interactif et répétitif. Ces 4 phases sont : l' élicitation, la modélisation, l'analyse, la validation.

-La phase d'élicitation consiste en la collecte d'informations sur le problème posé par le client.

-La phase de modélisation consiste à définir une architecture et spécifier le problème.

-La phase d'analyse met à jour les problèmes posés par ce type de modélisation. Cette phase n'est pas traitée dans le mémoire mais brièvement abordée en conclusion.

-La phase de validation vérifie si les spécifications correspondent avec le besoin du client. Si ce n'est pas le cas, une nouvelle phase d'élicitation recommence et ainsi de suite. Cette phase n'est pas traitée dans ce mémoire.

Ces quatres phases sont regroupées en deux activités :l'acquisition [Fic87], [Dub91b] et la spécification. L'acquisition regroupe l'élicitation et la validation. L'activité de spécification regroupe la modélisation et l'analyse. La première activité privilégie les rapports avec les clients et la deuxième activité privilégie le concepteur qui élabore un modèle qui sera accepté ou infirmé par la suite. Le document présenté se concentre à l'élicitation et la modélisation.

La phase d'élicitation peut prendre plusieurs formes : interviews, discussions, étude de documentations, observations, etc. A l'heure actuelle, un des problèmes est de disposer d'une méthode permettant de très bien

cibler ce que le client veut. Une méthode généralement utilisée est l'établissement d'un questionnaire "check-list" pour une famille de problèmes particuliers et de sélectionner les questions pertinentes pour le problème en question. Cette étape est un passage très difficile car il faut comprendre non seulement le problème du client (par exemple; "le calcul de la valeur actuelle nette") mais également son contexte (dans l'exemple, le domaine de l'investissement). Seule l'acquisition de cette connaissance va permettre par la suite un dialogue constructif entre le(s) client et le(s) développeur(s) de logiciel. Dans le cadre de ce mémoire, l'acquisition des connaissances est effectuée pour un problème d'investissement. Le processus d'acquisition est basé sur l'expérience déjà obtenue dans le domaine, des interviews pour délimiter les fonctionnalités désirées et une interface à partir desquelles les principaux concepts vont être dégagés. Le chapitre 1 présente les différentes interfaces.

La phase de modélisation consiste ensuite à élaborer la spécification. Le concepteur établit d'abord une architecture qui est dérivée à partir de la phase d'élicitation. Par la suite, il modélise en utilisant un langage de spécification qui peut être informel, rigoureux ou formel. Spécifier un problème en informatique : c'est le poser aussi précisément que possible en s'interdisant de penser prématurément à sa solution [Mey80]. Dans le cadre de ce mémoire, l'utilisation d'un langage formel orienté objet a été choisi:
-"langage formel" signifie qu'il s'agit d'un langage non ambigu auquel correspond une et une seule interprétation et, auquel sont associées des règles de déductions basées sur la logique mathématique,
-"orienté objet " indique que le paradigme de conception suivi est analogue à celui de la programmation orienté objet.
Le chapitre 2 du mémoire établit une architecture par rapport aux informations récoltées lors de l'étape d'acquisition et contribue à la mise en évidence des objets permettant par la suite d'écrire et de structurer les spécifications. Le troisième chapitre traite de façon pratique la manière de spécifier un problème d'investissement en utilisant le langage GLIDER [Dub91a].

Chapitre 1. Les différentes vues d'un problème d'investissement

1. 1 Délimitation du problème de l'investissement vu de manière philosophique.

L'investissement [Qui87] est une dépense actuelle de l'entreprise consentie dans le but d'obtenir des recettes futures. Il traduit une notion de croissance, de diversification ou de rationalisation. Les micro-économistes se rendirent vite compte que chaque définition avait son contraire. Ce fut le mot désinvestissement qui fut choisi. Le désinvestissement est une recette ou une dépense actuelle destinée à réduire les dépenses futures. Il désigne communément une idée d'abandon ou de recul mais son contenu réel en élargit la portée pour introduire la notion de reconversion ou de réorientation.

Une étude plus approfondie de la définition est utile :

- "dépense actuelle" ne signifie pas que ce soit de l'argent. Si l'investisseur décide de faire du troc, cela peut être considéré comme un dépense actuelle.
- "entreprise " doit être pris dans son sens le plus large. L'individu est lui même une entreprise qui possède un ensemble d'actifs tels que son corps,sa richesse pécuniaire, sa richesse intellectuelle etc. . et son passif caractérisé par ses handicaps personnels, pécuniaires, sportifs etc. .
_ "obtenir des recettes futures " n'implique pas que l'investisseur recevra une somme monétaire. Par exemple, si l'état décide de construire une autoroute, la dépense monétaire a bien lieu mais son remboursement se fait par une meilleure qualité du réseau. En général, l'Etat pratique une politique de l'enjeu. [deD90].

Comme la signification du mot investissement s'interprète de plusieurs façons, une limitation s'impose quant à la théorie. L'investisseur est de type financier. Sa logique de raisonnement, tout en n'ignorant pas la nécessité des autres impératifs, impose la supériorité d'un objectif précis : la rentabilité. Celle-ci possède trois propriétés :

-elle est une mesure de performance et de la validité des décisions,
- elle tend à privilégier les aspirations d'un groupe de personnes : l'espérance de rendement maximal formulée par les actionnaires, propriétaires du capital,
- elle est indispensable pour la satisfaction des aspirations des autres groupes de partenaires à la vie de l'entreprise.

La finance d'entreprise a ainsi consacré le rôle de l'objectif financier. Celui-ci consiste à maximiser la valeur de la société en maximisant sa rentabilité. Cette valeur peut être exprimée par le prix que les investisseurs sont disposés à payer pour acquérir un titre de capital de l'entreprise; par exemple,sur les marchés financiers,ce prix correspond à la cotation boursière des actions,laquelle reflète normalement une série de dividendes attendus augmentés de la valeur de revente estimée du titre au terme d'une période de détention. Cette formulation indique clairement que le raisonnement rationnel de l'investisseur repose sur des anticipations qui introduisent de même la notion de risque.

Il apparaît en conséquence que les décisions d'investir, de financer et de désinvestir dans l'entreprise, en voulant répondre à l' objectif financier, sont justifiées par la recherche de la meilleure combinaison rentabilité / risque susceptible de maintenir ou d' attirer la confiance des actionnaires de même que celle du personnel et des autres individus liés à l'entreprise. Tous les projets d 'investissements de financements et de désinvestissements qui sont de nature à contribuer à la rentabilité doivent être appréciés sur la base de cette logique.

Les fondements philosophiques étant posés, une solution technique doit être élaborée. Pour répondre à cette question il faudra s'intéresser à la théorie de l'investissement et au processus intellectuel qui amène l' investisseur à élaborer une solution d'un point de vue strictement technique.
Son processus intellectuel étant maîtrisé, il nous faut déterminer quel support informatique lui convient le mieux à partir d'un ensemble de logiciels se trouvant sur le marché ou qui pourraient l'être. Chaque

logiciel a ses avantages et ses désavantages par rapport au processus intellectuel de l'investisseur du type gestionnaire.

1. 2. Rappel théorique de la théorie de l'investissement.

Pour évaluer si un investissement est valable, deux formules sont utilisées : la valeur actuelle nette et le taux interne de rentabilité.

1. 2. 1. La valeur actuelle nette (VAN).

La méthode de la valeur actuelle nette offre plusieurs avantages; ainsi elle prend en compte la valeur et le moment de réalisation des flux et elle fournit une base valable d'évaluation des projets d'investissements en actualisant les flux nets de liquidité au coût du capital. Sa formulation est la suivante :

$$VAN = \sum_{t=0}^{n} \frac{F_t}{(1+k)^t}$$

1. 1 Formule de la VAN

K: taux d'actualisation
n: durée de vie estimée
F_t: flux net de liquidité en t (il est égal au capital investi et il inclut la valeur récupérable)

Si le résultat trouvé pour la VAN est positif, cela signifie que le projet d'investissement est acceptable et que sa rentabilité est au moins égale au taux d'actualisation.

Deux remarques s'imposent:
-la valeur actuelle nette est un critère d'acceptation (ou de rejet) mais non directement un critère de classement du projet car la VAN ne

précise pas le montant de l'investissement. Si l'investisseur désire classer ses projets, il déterminera la VAN par franc engagé,

- la formule de la valeur actuelle nette implique ipso facto le réinvestissement des flux au coût du capital. l'expression suivante est équivalente :

$$-1000 + \frac{1000}{(1.1)^1} + \frac{1000}{(1.1)^2} = \frac{-1000 * (1,1)^2}{(1.1)^1} + \frac{1000 * (1,1)^1}{(1.1)^2} + 1000$$

1. 2. 2. Taux de rentabilité interne (TIR)

Le taux de rentabilité interne est le taux d'actualisation pour lequel la valeur actuelle est nulle. Cela revient à déterminer le taux qui, si les capitaux nécessaires étaient empruntés à ce taux, annulerait le résultat de l'opération. Sa formulation s'établit comme suit :

$$TIR = \sum_{t=0}^{n} \frac{F_t}{(1+k)^t} = 0$$

1. 2 Formule du TIR

Notons que pour une durée de vie donnée, le taux de rentabilité interne est un taux que l'on perçoit en moyenne sur toute une période. Ce qui ne signifie pas que le financier touche chaque année le même rendement.

1.3. Processus intellectuel de l'investisseur de type gestionnaire.

Pour déterminer le processus intellectuel de l'investisseur,il est intéressant de cibler son niveau de connaissance. L'utilisateur de ce produit est un licencié en économie ayant goût pour la finance. Celui-ci utilisera uniquement les fonctions essentielles pour résoudre son

problème d'investissement. Ayant appris à l'université une manière standard de résoudre ce genre de problème, il est intéressant d'étudier le présent schéma intellectuel.

1.3.1. Principe.

L'utilisateur ayant un problème d'investissement à résoudre va prendre une feuille de papier organisée en lignes et colonnes et résout son problème en cinq étapes si le problème est considéré d'un point de vue séquentiel. Explicitons les :

- étape no 1 : dans la première colonne, il écrit tous les libellés les uns à la suite des autres, ce qui permet de visualiser l'ordre des opérations,

-étape no 2 : il écrit dans la première ligne, une référence à la période considérée,

-étape no 3 : l'utilisateur va écrire à partir de la deuxième colonne et la deuxième ligne toutes les valeurs qui lui semblent valables et ceci dans les limites du tableau,

-étape no 4 : à l'aide d'une formule basée sur les lignes précédentes, il va créer une ligne de résultat,

-étape no5 : dès que la ligne de résultat est créée, il doit déterminer le TIR et la VAN qui se trouvent sur une autre ligne.

1.3.2. Visualisation graphique.

libellé	ligne de temps
libellé	saisir des valeurs
libellé	calculer un résultat
libellé	déterminer VAN, TIR

1. 3 Visualisation graphique du processus intellectuel

1.3.3. Commentaire

L'investisseur, bien que dans un premier temps, choisit un raisonnement séquentiel pour élaborer son problème, sa manière de travailler va changer par la suite quand il étudie son problème plus en profondeur. Cela veut dire qu'à n'importe quel moment, il peut élaborer des retours en arrière sur une étape bien particulière.

1. 4. Les différentes visions pour résoudre un problème d'investissement

Nous avons distingué cinq visions pour effectuer un problème d'investissement. Citons les:

-la vision cellule,
-la vision logique-donnée,
-la vision des micro-économistes,
-la vision du gestionnaire,
-la vision tableur séparé pour chaque application.

Pour chaque vision, une définition, une critique générale et une critique du point de vue du gestionnaire sont présentées.

1.4.1. La vision cellule

A) Définition du tableur

Un tableur [Lal90] est un logiciel divisé en cellules, dans lequel des données peuvent être introduites. Mais le tableur ne s'arrête pas là; il fait aussi usage de la puissance de calcul de l'ordinateur. En plus des données, il mémorise des relations entre ces données sous la forme d'expressions mathématiques. Ces relations sont recalculées automatiquement chaque fois qu' une donnée est modifiée.

B) Critique du tableur

A l'heure actuelle, un certain nombre de critiques sont faites au tableur. D' après une étude expérimentale [Cou87] basée sur le type de problème concernant l'utilisation d'un tableur, plusieurs conclusions ont été faites. Explicitons les :
-la plupart des gens ayant utilisé le tableur ont été très confiants dans le résultat alors que la réalité enseignait un tout autre point de vue,
-les erreurs de type orthographique ne sont pas inclues dans l'étude,
-les chercheurs ont constaté que le nombre d'erreurs provenait surtout de l'écriture de formule,
- il existe un ensemble d'erreurs classiques telles que saisir une fausse donnée,effectuer un mauvais arrondi mal fait, protéger une cellule protégée alors qu'elle ne devait pas l'être.

En conclusion, si le nombre d'erreurs doit diminuer, l'interface doit être améliorée en vue de réduire le fossé qui existe entre la manière de penser et la manière de le réaliser.

C) Critique du tableur par le gestionnaire

Quand l'investisseur établi un problème d'investissement, il se fait qu' il a un certain art de manipuler les données du problème avec son raisonnement. Ce point de vue se retrouve dans le tableur.
Le tableur pose un ensemble de problèmes du fait qu'il offre trop de fonctionnalités dont le gestionnaire ne trouve guère d'utilité (ex: cos,

mod). Comme le tableur est un outil qui permet de tout faire, cela implique que l'utilisateur doit se plier à un certains nombre de concepts ne correspondant pas à son raisonnement intellectuel et amenant un temps d' apprentissage plus long.
Parmi un des concepts utilisés et souvent mal compris lors des problèmes d'investissements figure le concept de cellule. S'il est vrai que l'utilisateur rentre valeur par valeur, la manière de saisir des informations est différente. En effet les problèmes d'investissements se résolvent en terme de lignes. L'utilisateur travaille avec des lignes de valeurs, des lignes d'impôts etc.

1.4.2. La vision logique-donnée

A) Principe

Le but de cette vision est de séparer l'aspect conception et l'aspect donnée. Le logiciel FMP (processus de modèle financier) [Sch80] adopte cette vue. Lorsque l'investisseur établit un plan d'investissement, il doit d'abord écrire son modèle avant de saisir les données. Ecrire son modèle signifie que l''utilisateur attribue un statut aux lignes et aux colonnes. Dans ce cas le concepteur raisonne en terme de lignes et non de cellules. Une ligne contient plusieurs cellules. Prenons un exemple: supposons que trois lignes sont nécessaires pour construire un modèle. Cela s'écrit de la manière suivante:

L1 = UNITE VENDUES = INPUT
L2 = PRIX UNITAIRE = INPUT
L3 = CHIFFRE D' AFFAIRE = L1 * L2

1 2 3

1: numéro de ligne
2: libellé de la ligne
3: statut de la ligne (INPUT signifie donnée)

Le modèle étant réalisé, l' investisseur sait qu'il possède un tableau à trois lignes dans lequel deux lignes sont consacrées à des données et une troisième ligne est une ligne de résultat sur laquelle l'utilisateur ne

peut effectuer aucune action. Par la suite, l'utilisateur va saisir ses données. Quand l'investisseur a fini, il le signale et le tableau se calcule automatiquement. A partir de ce moment, l' investisseur peut établir des rapports qui correspondent aux lignes du tableau souhaité. Cette option est très intéressante pour la consolidation des comptes.

B) Critique de la vision logique / donnée.

Le très grand défaut de ce logiciel provient d'un temps d'apprentissage extrêmement long pour bien maîtriser ce type de produit.
Par contre, le fait de séparer la logique d'un problème et les données permet de savoir si les personnes maîtrisent parfaitement la théorie qu'ils essayent de modéliser.

C) Critique de la vision logique /donnée par l'investisseur.

Le concept de ligne est très intéressant à utiliser car cela correspond bien à la philosophie du gestionnaire. Cependant la séparation du mode de conception et du mode de donnée ne correspond pas du tout à sa manière de penser. A nouveau, ce type de produit est axé sur un nombre d'applications beaucoup plus élevé puisque le produit s'adresse à un ensemble de problèmes financiers classiques tels que la centralisation des comptes, l'élaboration des comptes, les problèmes de gestion de personnel etc. .

1.4.3. La vision du micro-économiste

En général, les microéconomistes résolvent souvent leur problème sous une forme arborescente.

Bien que le produit ne soit pas sur le marché, une première interface a été construite pour expliquer comment fonctionnerait ce logiciel [Bec91].

A) Explication de fonctionnement

Au départ l'utilisateur à un ensemble d'option possible(fig 1. 4).

CF-TABLE	TIME-PERIOD
ITEM	
CAPITAL-COST	
DISCOUNTED-CASHFLOW	

1. 4 Menu initial

Option CF-TABLE

L'utilisation de cette option permet de saisir les valeurs à condition que les items soient écrit dans le tableau (fig 1. 5).

CF-TABLE	90	91	92	93	94	95	96	97
sales								
salaries								
raw								
variable cost								
maintenance								
energy								
results								
taxes								
profit								

1. 5 C f-table

Option ITEM

Cette option ouvre une fenêtre dans laquelle 5 subdivisions sont représentées: ITEM, AGGREGATED ITEM, un arbre, PLACE-IN-ROW, PLACE-DEDUCED-EVAL-EXPRESSIONS (fig 1. 8).

Option CAPITAL-COST

Cette option permet de saisir le taux d'intéret que l'investisseur souhaite. Cela permettra de calculer la valeur actuelle nette.

Option DISCOUNTED-CASHFLOW

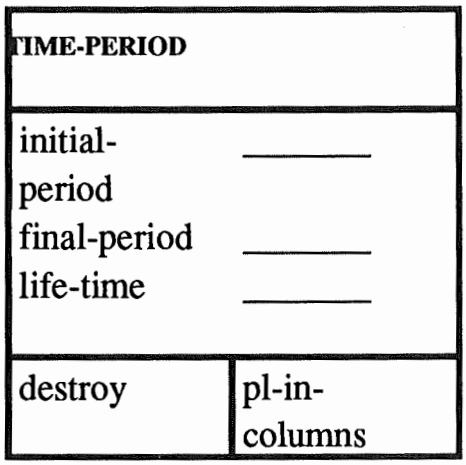
Cette permet de calculer la VAN (fig 1. 6).

CF-TABLE	90	91	92	93	94	95	96	97
sales								
salaries								
raw								
variable cost								
maintenance								
energy								
results								
taxes								
profit								
investment	**EVALUATION-EXPRESSION**							
cash-flow	scope: [90. . 97]							
capital cost	equation: i=0. . 8, cash flow/(1+cap. cost)**							
discounted -- cash flow	->							

1. 6 Discounted cash flow

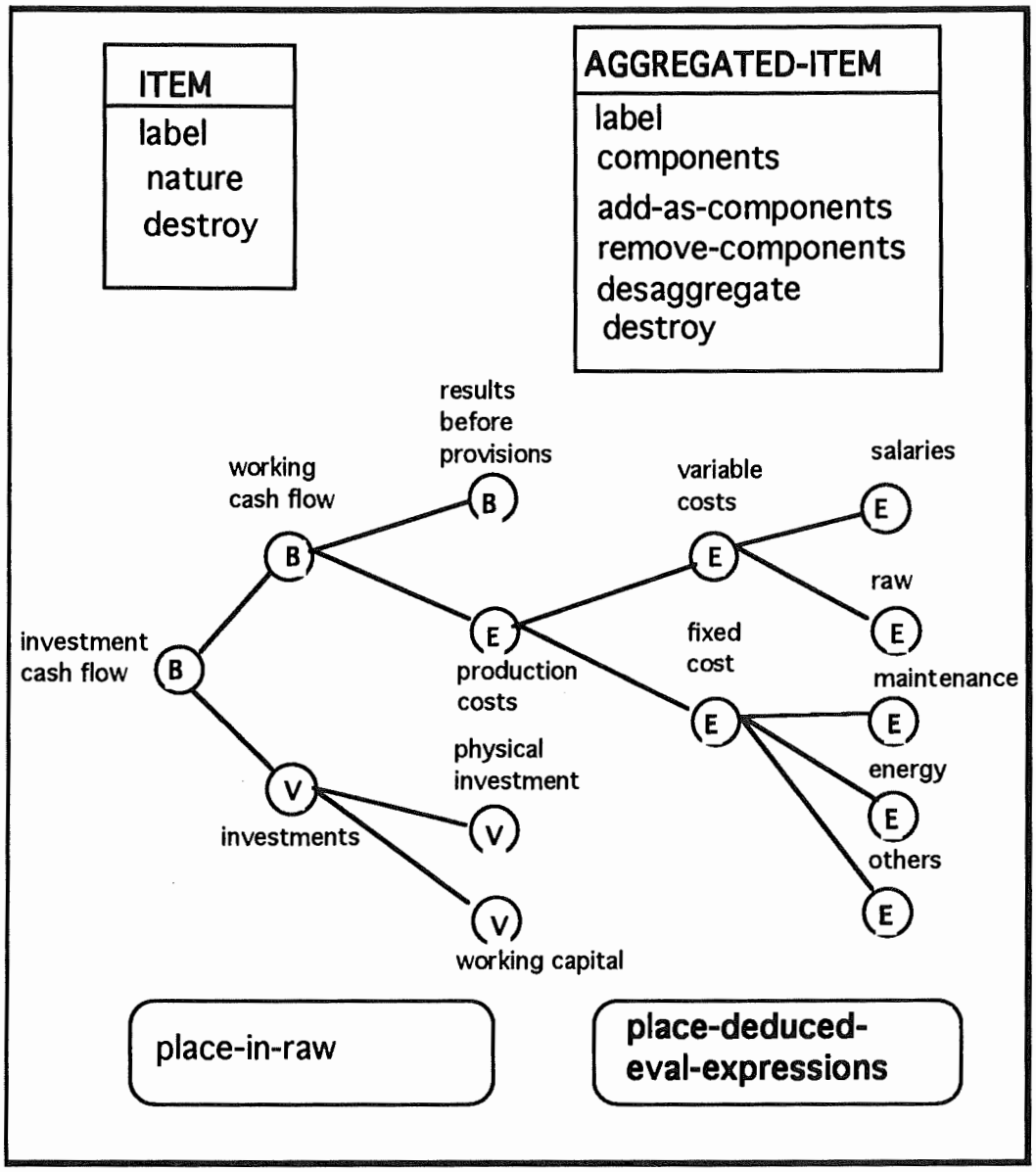
Option TIME-PERIOD

Cette option ouvre une boîte de dialogue dans laquelle l'utilisateur donne la période initiale, la période finale, le temps de vie. Deux sous options sont possibles : le placement de la ligne de temps dans le tableau ou la destruction des données entrées (fig 1. 7).



1. 7 Time period

L'option ITEM.



1. 8 Item

L' option ITEM.

Un item est un libellé ne dépendant pas d'autres libellés. Sur l'item, un certain nombre d'opérations sont possibles tel que saisir un item, détruire un item. Les exemples d'items créés sont : salaries, raw material, maintenance,working-capital (fig 1. 8).

L'option AGGREGATED-ITEM.

Un agrégat d'items est un item réunissant un ensemble d'items. Par exemple: variable cost, production cost sont des agrégats d'items. Les opérations utiles sur un item agrégé sont : donner un nom à l'agrégat, donner les composantes de l'agrégat (par ex: les composantes de variable cost sont salaries et raw material), ajouter une composante à l'agrégat, déplacer une composante d'un agrégat à un autre agrégat, détruire un agrégat (fig 1. 8).

L'option PLACE-IN-ROW.

Quand l'ensemble des agrégats ont été placés dans la fenêtre, l'option place-in row permet de placer les items et les agrégats dans le tableau suivant la position occupée dans la hiérarchie de l'arbre (fig 1. 8).

L'option PLACE -DEDUCED-VAL-EXPRESSIONS

CF-TABLE	90	91	92	93	94	95	96	97
raw	**scope : variable cost**							
salaries	**equation: salaries +raw**							
variable-cost	variable cost:90 = salaries:90 + raw90							
maintenance	variable cost:91 = salaries:91 + raw91							
energy	variable cost:92 = salaries:92 + raw92							
others	variable cost:93 = salaries:93 + raw93							
fixed cost	variable cost:94 = salaries:94 + raw94							
production	variable cost:95 = salaries:95 + raw95							
results	variable cost:96 = salaries:96 + raw96							
taxes	variable cost:97 = salaries:97 + raw97							

1. 9 Place-deduced-val-expressions

L'utilisateur se positionne sur l'élément de l' arbre auquel il veut attribuer une formule. Par la suite, il clique sur place deduced eval-expression. Le tableau de cash flow lui est présenté plus une fenêtre demandant d'écrire l'expression
(l'exemple pris concerne variable cost (fig 1. 9)).

B) Critique de cette vision.

Certaines constatations sont à faire. Travailler sous forme d'arbre paraît intéressant à condition que le nombre de libellés ne dépasse pas une certaine taille ou que le logiciel incorpore une fonction de réduction des caractères au fur et à mesure que l' espace du tableau se remplit. La question de lier les graphes attachés à la feuille de calcul semble aussi une question très pertinente et non traitée à l'heure actuelle pour tous les logiciels de ce type. Il est très compliqué à l'heure actuelle de manier la feuille de calcul avec l'interface graphique [Din91] (ex : EXEL, LOTUS). Une dernière remarque concernerait le choix des objets et des opérations. La question est de répondre si les objets utilisés sont les bons ? Une étude plus psychologique du milieu serait à faire. Bref le domaine est loin d'être exploré.

C) Critique de cette vision par le gestionnaire

L'approche arborescente ne correspond pas du tout à la vision du gestionnaire. Il a toujours l'habitude de travailler en lignes et colonnes car l'enseignement lui a appris cette habitude. Maintenant si l'enseignement change sa manière d'expliciter sa théorie, alors peut-être cela correspondrait à une bonne représentation mentale. Quant au fossé logique/donnée, il est important car l' utilisateur doit d'abord rentrer son modèle avant de rentrer ses données.

1.4.4. La vision du gestionnaire

La vision proposée ici est la vision developpée dans le cadre de ce mémoire qui est spécifiée par la suite.

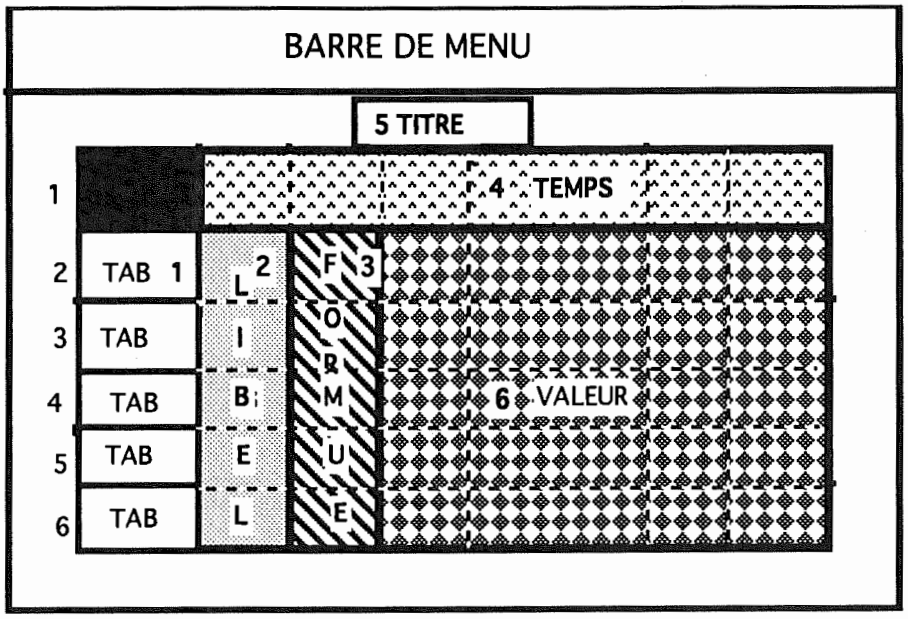
A) Principe.

En reprenant sa démarche intellectuelle discutée au point.1.3 un concept important apparaît : la découpe du tableau en lignes. Quand l'investisseur doit résoudre son problème, il exprime clairement son besoin en terme de lignes. Par exemple l'utilisateur désire une ligne de produit suivie d'une

ligne de charge suivie d'une ligne d'amortissement, suivie d'une ligne de résultat brut, suivie d'une ligne d'impôt, suivi d'une ligne de résultat net.

Certaines lignes comme la ligne impôts, amortissements, résultats sont des lignes de type formule. D'autres lignes sont de type valeur telle que la ligne de charge et produit. Cette constatation étant faite, cela permet de faciliter le travail lors de la spécification du problème.

Comme la logique de fonctionnement a été décrite d'une manière assez succincte du fait que la vision logique/donnée en parle aussi, une interface de la feuille de calcul va être proposée (fig 1. 10).



1. 10 Vision générale de la feuille de calcul

Première constatation: la barre de menu doit être réduite à son maximum. Seules quelques opérations n'ayant rien à voir avec le tableau doivent se situer dans la barre de menus.

Deuxième constatation : une des grandes idées maîtresses réside dans le fait qu' à chaque cellule du tableau correspond un menu caché et qui apparaît quand l'individu demande le contenu de cette case.

Troisième constatation : chaque colonne du tableau appartient à une famille bien particulière des six menus. Citons les:

-1: menu tableau,
-2 : menu libellé,
-3 : menu formule,
-4 : menu temps,
-5: menu titre,
-6: menu valeur.

Menu tableau.

Si l'individu se positionne sur la case ou il est possible d'effectuer des opérations sur un tableau, voici les opérations qui lui sont offertes (fig 1. 11)

TABLEAU
Supprimer une ligne dans un tableau
Etendre une ligne dans un tableau
Limiter le nombre de lignes du tableau
Insérer une ligne dans un tableau
Quitter le tableau

1. 11 Menu tableau

L'option supprimer une ligne du tableau.

Cette instruction permet de supprimer une ligne automatiquement sans passer des paramètres comme cela se fait avec EXEL ou LOTUS et le tableau se recalcule automatiquement.

L'option étendre une ligne dans un tableau.

Cette instruction permet d'ajouter une ligne automatiquement sans passer des paramètres.

L'option limiter le nombre de lignes dans le tableau.

Cette instruction permet à l'utilisateur de supprimer toutes les lignes supérieures par rapport à sa position dans le tableau.

L'option insérer une ligne dans un tableau.

Cette instruction permet à l'utilisateur de créer une ligne à la fin du tableau.

L'option quitter le tableau.

Cette instruction permet de sortir de la feuille de calculs.

Le menu libellé

Dans la colonne deux, l' utilisateur peut écrire du texte mais s'il le désire, il peut demander le menu de cette case qui lui fournit un dictionnaire de mots dans lequel il peut choisir un mot qui s'inscrit automatiquement dans la case où il est positionné. Deux autres opérations lui sont fournies pour lui permettre de gérer le dictionnaire de mots. Ce sont les opérations supprimer un mot du dictionnaire et ajouter un mot dans le dictionnaire (fig 1. 12).

LIBELLE
Amortissement linéaire Amortissement dégressif VAN TIR Résultat Impôt
Supprimer un mot
Ajouter un mot

1. 12 Menu libellé

Le menu formule.

Dans la colonne trois et à partir de la ligne deux, l'utilisateur a le choix entre trois manières de raisonnement. Lors de l'initialisation du tableau, toutes les lignes sont destinées à loger des valeurs. Toutes les cases formules ont le mot valeur. Ces trois modes de raisonnement sont :
-soit la ligne est différente d'une ligne de valeur, mais elle est l'agrégation d'un ensemble d'autres lignes. L'utilisateur écrira sa formule soit en fonction de chiffres (L2 + L3) soit en fonction des libellés. Lors des spécifications du problème, l'hypothèse d'écrire des formules sous forme de libellé est supprimée. Les deux représentations sont directement inscrites dans la case formule et la ligne est grisée, ce qui veut dire qu'aucune valeur ne peut être rentrée,

-soit c'est une formule préétablie. Dans cette hypothèse, l'utilisateur demande le menu formule (fig 1. 13) de la case dans lequel il choisira la formule souhaitée et saisit les paramètres demandés. Quand il a fini, la case formule contient le nom de la formule utilisée et la ligne est grisée.

-Soit c'est une ligne valeur et l'utilisateur peut saisir toutes les valeurs qu'il souhaite et la case formule contient le mot valeurs.

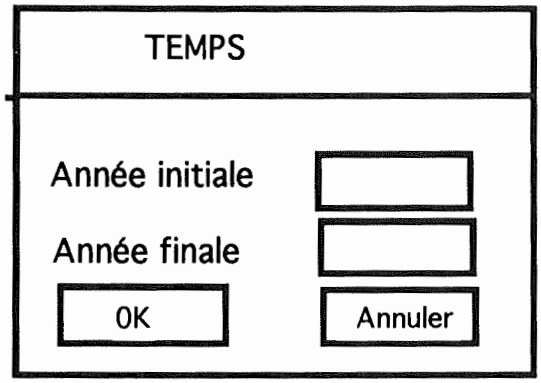
Si l'utilisateur décide de passer d'une ligne formule à une ligne valeur, il se positionne sur la case formule, demande la boîte et choisit la ligne de valeur. Cela a pour effet de détruire l'ancienne ligne et de créer une ligne valeur.

Formule
Amortissement linéaire Amortissement dégressif Impôt TIR VAN
Ajouter une formule Supprimer une formule

1. 13 Menu formule

Le menu période.

Par défaut les années sont limitées dans le temps. Si l'utilisateur décide de changer les limites de la période, il se positionne sur la ligne temps et demande la boîte de dialogue (fig 1. 14) qui lui demande l'année initiale, l'année finale. Quand l'utilisateur touche OK, le tableau est remis à jour.



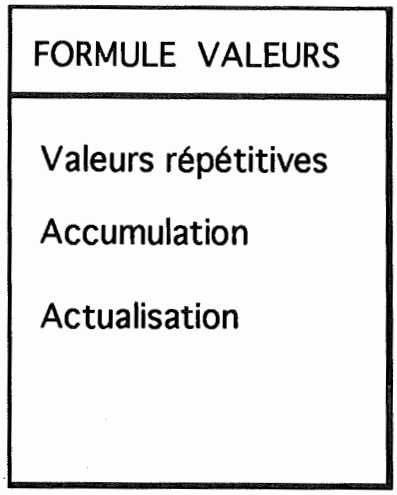
1. 14 Menu temps

Le menu titre de la feuille

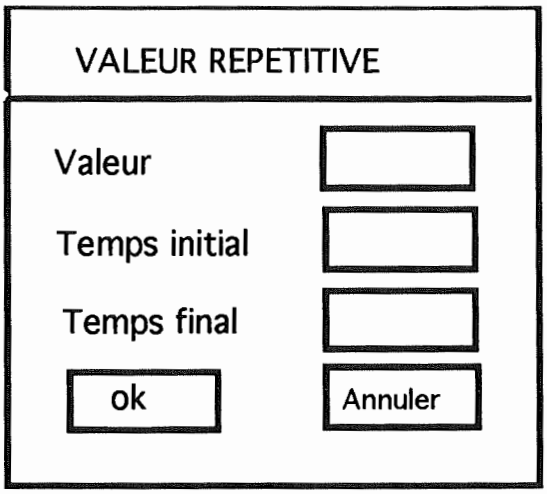
Il s'agit de l'ensemble des opérations qui permettent de nommer, sauver, supprimer un document.

Le menu valeurs.

L'utilisateur a le choix entre deux options :
-soit il inscrit directement les valeurs à condition que ce soit une ligne de valeurs,
-soit il demande sur la case où il est situé: le menu formule valeur (fig 1. 15) qui consiste par exemple à écrire une ligne de valeur répétitive (fig 1. 16). Le fait d'effectuer des formules dans le menu valeurs montre qu'il n'existe aucune liaison avec les autres lignes.



1. 15 Menu valeur



1. 16 Boite de dialogue : valeur répétitive

B) Critique de cette vision

A nouveau, c'est une première ébauche de solution qui est loin d' être satisfaisante et qui demande à être explorée davantage. Cette petite interface tient en compte les principales sources d'erreurs que l'utilisateur effectuait avec un tableur de type normal et cela tant au point de vue maniabilité que fiabilité du résultat.

Ce type d'application réduit terriblement la puissance d'un tableur normal. Quant à la liaison feuille de calcul et interface graphique, cela demande sérieuse réflection. L'utilisation d'un browser ne serait peut-être pas à négliger. Cela apporterait une toute autre vision du tableur quant à sa maniabilité.

Mais le grand problème qui se pose lors d' un problème d'investissement est de déterminer quelle liberté d'action lui laisse-t-on. Pourquoi ? Prenons l'exemple des libellés : il serait tout à fait normal de lier le libellé avec une ligne de formule.
Or, que constate-t-on dans la réalité ? La plupart des personnes préfèrent donner une autre appellation. Par conséquent dans les choix des hypothèses nous pris le concept de libellé à part.

Un autre problème surgit dans cette même famille. Il s'agit des problèmes de cohérence. Par exemple : quand un utilisateur demande une ligne d'amortissement, a-t-il le droit par après de détruire les lignes qui lui servent de référence au calcul ? Ici la réponse tombe sous le sens : c'est non. Mais au fur et à mesure que l'on rajoutera de nouvelles formules, jusqu'à quel point faut-il garder la cohérence?. Devons nous aller vers une automatisation complète d' un problème d'investissement ou pas ?. Par

exemple : si un utilisateur demande la formule "bénéfice imposable ", automatiquement la ligne impôt et résultat net serait créée. Faudrait-il créer deux modes de travail ?

C) Critique du gestionnaire.

Une des grandes critiques que l'utilisateur peut faire au problème d'investissement vu en ligne est la suivante : à une ligne correspond un seul type d'opération et cela est inacceptable. Par exemple: sur une ligne, l'utilisateur désire accumuler deux valeurs différentes sur deux périodes différentes. Il est obligé de créer trois lignes : deux lignes d'accumulations et une ligne de résultat faisant la somme des deux précédentes. Une des solutions est d'introduire le concept de colonnes mais le risque de retomber sur la vue d'un tableur classique est grand.

Chapitre 2. : le processus de conception

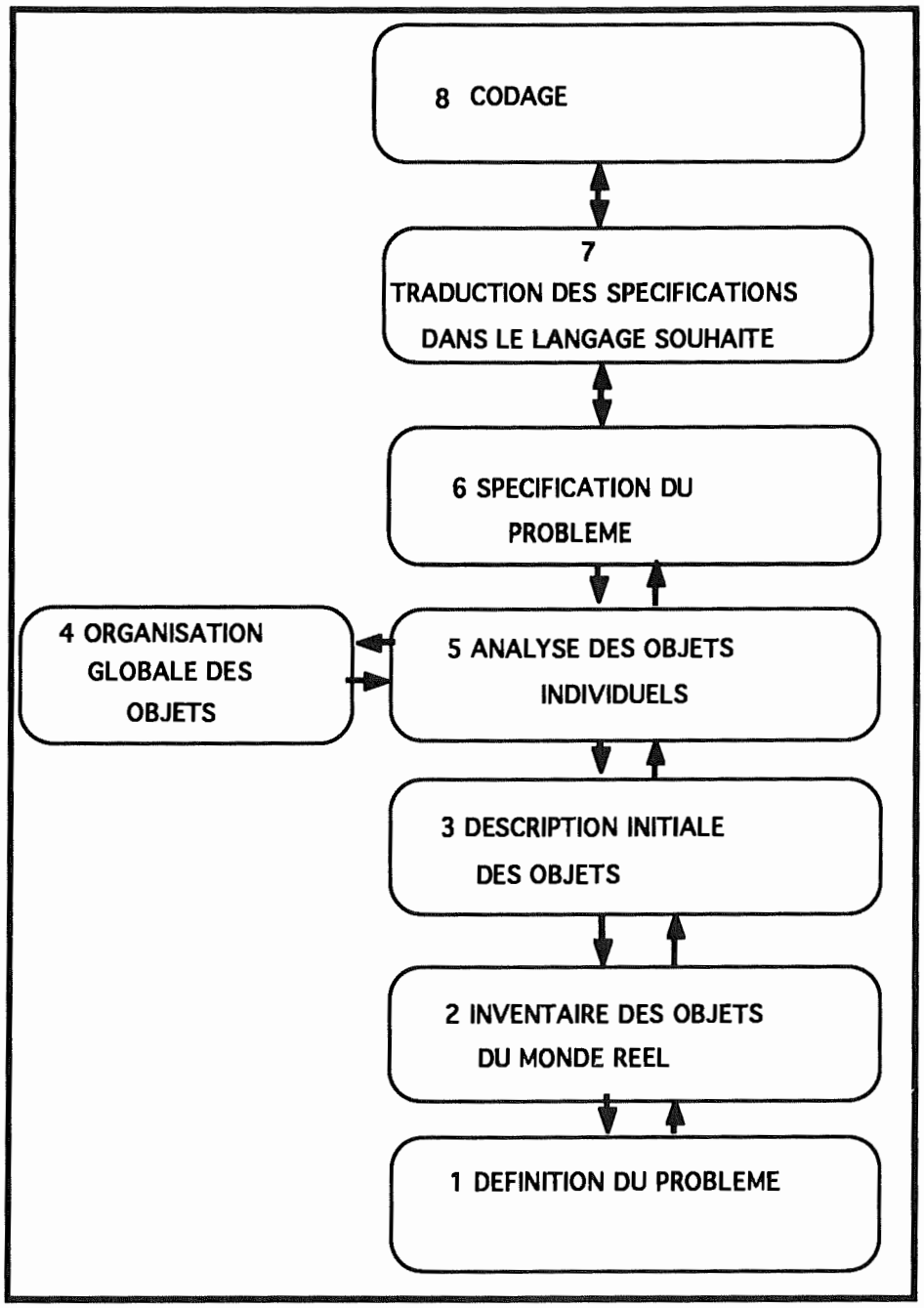
2.1. Introduction.

Le processus de conception se base sur la vision du gestionnaire présentée au point 1. 4. 4. du chapitre 1. Le problème pour le concepteur est de déterminer son plan de travail en incluant certaines qualités souhaitées telles que:
- l'utilisation d'un langage formel de spécification (GLIDER),
- une structuration de la spécification en suivant le paradigme "orientée objet",
- le développement du concept de ligne.

Face à ces objectifs, la méthode proposée est d'exposer de manière théorique la manière dont cela va être conçu. Par la suite, la méthode théorique est appliquée au problème de l'investissement.

2.2. LeProcessus de conception d'un point de vue théorique.

La figure (Fig 2. 1) montre les huit étapes d'un processus conceptuel orienté objet [Rol92].



2. 1 Processus de conception

Remarque : les étapes de 1 à 6 représentent à l'ingénierie des besoins et les étapes 7 et 8 correspondent à l'ingénierie de conception.

2. 2.1. Définition du problème.

Le concepteur va définir le but à atteindre en effectuant une étude approfondie de ce que le client veut mais cela sans celui-ci.

2.2.2. Inventaire des objets du monde réel.

Cette étape consiste à identifier tous les objets de base que le concepteur va utiliser pour réussir sa conception. Identifier tous les objets de base est une question d'intuition et de perception du problème. Seules les étapes ultérieures permettent de dire si les objets utilisés sont les bons. En général, les objets sont identifiés après une réflection méthodologique [Boo90], [Coa89], [Kar90].

2.2.3. Description initiale des objets.

Cette étape permet de donner une définition de chaque objet en français. Cette définition amène dans le futur le concepteur à définir d'une manière plus rigoureuse des objets. S'il est possible de dégager une propriété très générale du problème, le concepteur le notifie [Joh88].

2.2.4. Organisation globale des objets.

Cette étape permet de considérer l'ensemble des objets pour définir une structure globale. Les schémas d'objets sont organisés hiérarchiquement en appliquant une démarche bottom-up. Cette organisation évite des redondances de spécifications. Elle constitue aussi un mécanisme d'abstraction permettant d'étudier les schémas d'objets à différents niveaux de détail. En général, le concepteur présente son architecture orientée objet sous forme d' un schéma qu' il commente [Kar90].

2.2.5. Analyse des objets individuels.

Sur chaque objet, l'utilisateur définit les principales opérations de base [Reb90] qu'il désire avoir. Une remarque est à faire quant à ce terme. Une opération de base peut faire référence à d'autres opérations. Comme la définition des opérations de base n'est en général pas assez précise, le concepteur ne peut pas déterminer l'ensemble des opérations. Ce n'est qu'au stade de la spécification que le concepteur peut déterminer d'ajouter des opérations.

2.2.6. Spécification du problème.

L'analyse générale étant effectuée, le concepteur spécifie le problème en utilisant un langage de spécification (ex : GLIDER [Dub91]). Cela lui permet d'avoir une vue plus formelle du problème. A cette étape, le concepteur crée des opérations supplémentaires. L'aspect théorique des spécifications est traité au chapitre 3.

2.2.7. Traduction des spécifications dans le langage souhaité.

A cette étape, le concepteur choisi le langage dans lequel il veut programmer. Il remodifie toutes les spécifications en fonction du langage. Un des grands problèmes réside dans le fait que le langage de spécification ne peut avoir aucun lien avec un langage de programmation (par ex : l'utilisation du langage GLIDER pour faire du COBOL). Le langage GLIDER est très propice pour programmer en C++ ou EIFFEL [Mey90] car les modifications sont assez faibles [Pig91] Les différents langages de programmation orienté objet sont : C++, EIFEL, TRELLIS [O'b85], SIMULA, SMALLTALK [Gol85], TURBO PASCAL 6. 0.

2.2.8. Codage.

Cette étape vise à coder et tester le programme.

2.3. Processus de conception d'un point de vue pratique

2.3.1. Définition du problème

A partir de l'étude d'une interface avec l'utilisateur, deux points importants ont été dégagés:
-la nature des objets qui sont des lignes,
-les principales fonctionnalités.

Cette partie s'attache à déterminer les principales fonctionnalités que l'investisseur peut exécuter à partir d'une feuille de calcul. Les principales fonctionnalités pratiques que l'utilisateur demande sont : pouvoir écrire des valeurs, déterminer un TIR, déterminer une VAN, déterminer un amortissement linéaire, déterminer un amortissement dégressif, déterminer son bénéfice imposable, déterminer les impôts, accumuler des valeurs, déterminer la valeur résiduelle d'une ligne d'amortissements, effectuer des accroissements, actualiser une ligne, écrire des formules simples en fonction des autres lignes de la feuille de calcul.

Il est à noter que les graphiques que l'investisseur peut faire à partir d'une feuille de calcul ne sont pas représentés car l' objectif se fixe aux principales fonctionnalités du système. L'objectif essentiel est de déterminer tous les types de lignes dont l'investisseur a besoin pour résoudre le problème. Avant d'aller plus loin, un certain nombre d'hypothèses ont été élaborées pour respecter une certaine logique de fonctionnement lors de la réalisation du problème d'investissement. Cinq hypothèses fortes ont été choisies:

- l' utilisateur est sensé connaître la théorie,
- les opérations offertes reflètent la logique la plus probable pour résoudre un problème d'investissement. Par exemple: l'utilisateur peut décider qu'une ligne résultat peut être du type $((L11+L4)*L5)^{34}$. Dans les problèmes d'investissements, ce type d'opération complexe n'existe pas. L'investisseur se contente d'établir une ligne résultat avec + et - ou * et /. Cela veut dire que L5= (L1+L3)*3 n'est pas réalisable, mais le même résultat peut s'obtenir en créant une ligne L4=(L1+L3) et puis une ligne faisant référence à l'ancienne L5 = L4*3,
-l'utilisateur ne peut utiliser plusieurs types d'opérations sur une même ligne. Par exemple: sur la ligne 32, il veut accumuler 1000 f de 1970 à

1980 et répéter une valeur entre 1981 et 1990. Pour s'en sortir l'utilisateur devra créer trois lignes : deux lignes d'accumulations et une ligne résultat faisant la somme des deux. précédentes,
- l'utilisateur ne peut référencer que des lignes supérieures lorsqu'il écrit des formules,
- dans un premier stade ce mémoire se limite à la formule de la VAN classique et du TIR. Pour raison de délimitation du projet, certaines fonctionnalités ne sont pas abordées telles que : la payback période, la valeur actuelle nette par franc engagé, les représentations graphiques associées, une VAN calculée à partir du premier janvier et non à partir du 31 décembre etc.

2.3.2. Inventaire des objets du monde réel

La détermination des objets du monde réel s'est élaborée à partir de plusieurs inteviews. Plusieurs types de lignes ont été choisies. Malheureusement ce n'est pas une liste exhaustive car le but de ce mémoire consiste à démontrer l'intérêt de l'ingénierie des besoins.

Les lignes les plus souvent utilisées sont : une ligne de temps, une ligne de valeurs, une ligne de résultats, une ligne d'amortissements linéaires, une ligne d'amortissements dégressifs, une ligne de bénéfices imposables, une ligne d'impôts, une ligne VAN, une ligne de valeurs répétitives, une ligne d'accumulations, une ligne d'accroissements, une ligne de pertes reportées, un tableau, une ligne de valeurs résiduelles.

L'identification des différents objets de base peut paraître limpide; cependant, ce n'est guère une tâche simple car il faut toujours avoir à l'esprit que tous les objets créés doivent être réutilisables. Par la suite, certains objets ont été agrégés car ils possèdent la même structure de données et des opérations communes. Est-ce une bonne solution ? La réutilisation et l'augmentation du nombre d'opérations par objet sont des éléments de réponse pour déterminer si le choix est correct.

2.3.3. Description initiale des objets

Cette étape vise à définir en français les objets de base. Les objets de base sont les objets élémentaires pour l'application. Cela correspond au deux derniers niveaux de la figure 2. 2 qui présente une approche décomposition descendante des objets du problème.

A) Ligne de temps

Définition : objet qui représente une suite d'années comprises dans une période délimitée.

B) Ligne de valeur.

Définition : objet qui représente une suite de chiffres et chaque chiffre est associé à une année donnée.

C) Ligne de résultat.

Définition : objet qui est composé d'une ligne de valeur et d'une expression. Cette expression est de quatre types: la sommation ou négation (l1 + L2 - L3), la division (L1/L2), la multiplication (L1 * L2) entre lignes.

D) Ligne de formule.

Définition : objet regroupant un ensemble d'objets associés à un ligne de formule pouvant être spécialisée en lignes : d'amortissement linéaire, dégressif, d'impôt, financière, bénéfice imposable, accroissement, valeur résiduelle.

E) Ligne financière.

Définition: objet qui est composé : d'un numéro de ligne qui fait référence à la ligne avec laquelle l'utilisateur décide de travailler, d'une ligne de valeur qui contient le résultat de l'opération demandée, d'un taux d'intérêt, du temps initial pour commencer l'opération et du temps final pour terminer l'opération.

F) Ligne résiduelle.

Définition: objet qui est composé : d'un numéro de ligne qui fait référence à une ligne résultat ou une ligne de valeur, d'un numéro de ligne qui fait référence à une ligne d'amortissement dégressif ou linéaire, une ligne de valeur destinée à recevoir le résultat.

G) Ligne d'amortissement linéaire.

Définition: objet qui est composé d'une ligne de valeur, d'un numéro de ligne qui fait référence à la ligne des biens à amortir et un autre numéro de ligne qui correspond à la période d'amortissement autorisée pour chaque bien. L'utilisation de ces deux lignes va permettre de créer une ligne d'amortissement linéaire dont le résultat se trouve dans la ligne de valeur.

H) Ligne d'amortissement dégressif.

Définition: objet qui est composé d'une ligne de valeur, d'une période de temps unique d'amortissement et d'un numéro de ligne qui fait référence à la ligne des biens à amortir. La ligne de valeur contient le résultat de l'amortissement dégressif.

I) Ligne impôt.

Définition: objet qui est composé du taux d'imposition imposé par la loi, d'une référence à la ligne qui doit être imposée et une ligne de valeur qui contient le résultat de l'imposition.

J) Ligne bénéfice imposable.

Définition: objet composé d'un numéro de ligne qui fait référence à la ligne dont l'utilisateur désire calculer le bénéfice imposable et la ligne de valeur qui contient le résultat.

K) Ligne résiduelle.

Définition: objet composé d'un numéro de ligne qui fait référence à la ligne des biens à amortir, d'un numéro de ligne de référence qui fait référence à la ligne d'amortissement et d'une ligne de valeur qui contient la valeur résiduelle à la dernière année.

L) Ligne TIR.

Définition: objet composé d'un numéro de ligne qui fait référence à la ligne dont l'utilisateur désire calculer le TIR et la ligne de valeur qui contient le résultat.

2.3.4. Organisation globale des objets

Cette étape vise à établir un schéma hiérarchique qui permet de voir comment les objets vont interagir entre eux. La figure 2. 2 représente l' architecture définie. Un ensemble d'objets vont être créés pour établir des agrégats cohérents et qui permettent par la suite de mieux spécifier le problème.

A) Ligne-donnée.

Définition: objet composé des principales lignes de travail et du type de ligne avec laquelle l'utilisateur travail. Ces lignes sont : ligne de valeur, de temps, de résultat, de formule.

B) Ligne.

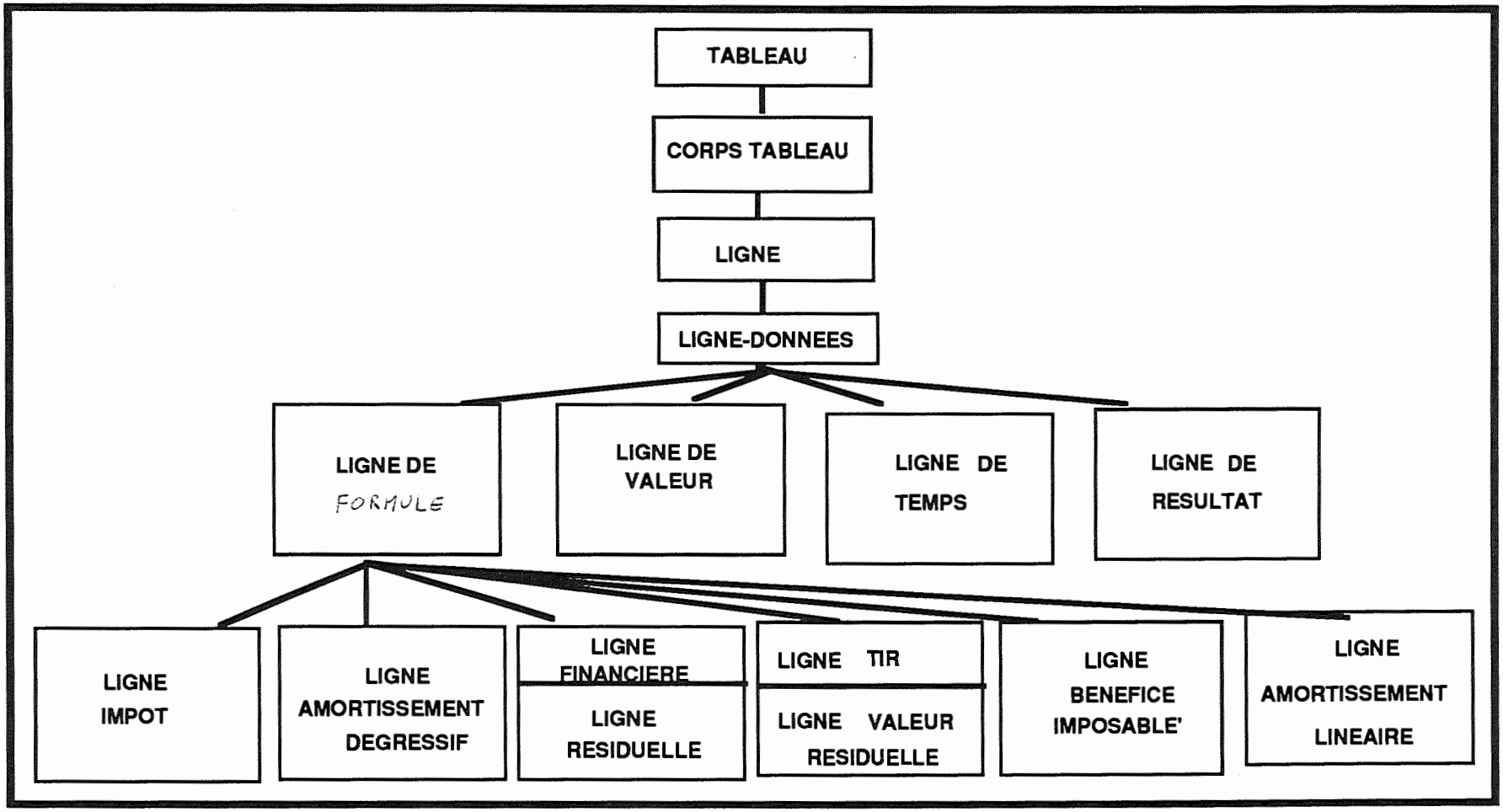
Définition : objet qui est composé d' un libellé et d'une ligne-donnée.

C) Corps tableau.

Définition: objet qui est composé d'une suite de lignes.

D) Tableau.

Définition: objet qui est composé d'un corps tableau et d'un nom.



2.2 Architecture globale

2.3.5. Analyse individuelle des objets .

Le but de cette étape est de décrire l'ensemble des opérations finales de chaque objet. Deux types d'opérations sont utilisés : les opérations intermédiaires et les opérations finales. Les opérations de type intermédiaire sont des opérations où aucune action n'est effectuée dans l'objet. Les opérations de type final sont des opérations réellement effectuées sur l'objet.

A) L'objet tableau.

L'ensemble des opérations se trouvant dans cet objet a pour but de citer les principales fonctionnalités demandées. Ce qui veut dire que le rôle de chaque fonctionnalité est minime mais elle offre l'avantage d'une plus grande clarté dans le raisonnement. La seule opération finale qui est faite sur cet objet est de créer un tableau. Quant à la définition de chaque opération, elle se situe au niveau des spécifications.

B) L'objet corps tableau.

C'est autour de l'objet corps tableau que tout le système fonctionne. L'ensemble des opérations a pour but de clarifier les spécifications de l'objet tableau et de travailler sur l'ensemble des lignes du tableau.

Les opérations finales sont: limiter la taille du corps, remodifier le corps, supprimer le corps, recalculer le corps, créer un corps , créer une ligne d'impôts, calculer le résultat, créer une ligne d'amortissements, créer une ligne VAN, créer une ligne d'actualisations, créer une ligne d'accumulations, créer une ligne de bénéfices imposables, créer une ligne d'accroissements, créer une ligne de valeurs, créer une ligne d'amortissement dégressifs, créer une ligne TIR, créer une ligne de résultats, créer une ligne de valeurs répétitives.

C) L'objet ligne.

A l'origine, cet objet contenait des opérations de type intermédiaire et final. Les opérations intermédiaires ont été tranférées dans un autre objet qui est l'objet ligne donnée et cela dans un but de clarté. Les opérations de type action sont : mise à blanc, écrire libellé.

D) L'objet ligne donnée.

Cet objet ne contient que des opérations intermédiaires.

E) L'objet ligne de valeurs.

Les opérations sont toutes de type final. Les différentes opérations que l'utilisateur peut effectuer sont: mise à zéro de la ligne, insérer des valeurs, vérifier si c'est une ligne de zéro, insérer des valeurs répétitives, accumuler une valeur, effectuer un accroissement dans la ligne.

Insérer une valeur répétitive, accumuler une valeur, effectuer un accroissement, auraient pu être mis dans des objets distincts. Le choix de cette philosophie trouve son origine dans la non-recaculabilité de cette ligne car la ligne n'utilise pas d'autres lignes. Est-ce un bon choix ? La conclusion du mémoire en parle.

F) L'objet ligne de temps.

Les opérations sont toutes de type final. Les différentes opérations que l'utilisateur peut effectuer sont : initialiser le temps, modifier le temps de départ, modifier le temps final, tester le temps.

G) L'objet ligne formule.

Cet objet ne contient que des opérations de type intermédiaire.

H) L'objet ligne résultat.

Les opérations restent à définir.

I) L'objet ligne amortissement linéaire.

Cet objet ne contient qu'une opération finale, c'est l'opération : calcul de l'amortissement linéaire.

J) L'objet ligne amortissement dégressif.

Cet objet ne contient qu'une opération finale; c'est l'opération : calcul de l'amortissement dégressif.

K) L'objet ligne bénéfice imposable.

Cet objet ne contient qu'une opération finale; c'est l'opération : calcul du bénéfice imposable.

L) L'objet ligne impôt.

Cet objet ne contient qu'une opération finale; c'est l'opération : calcul ligne impôt.

M) L'objet ligne TIR.

Cet objet ne contient qu'une opération finale; c'est l'opération : conserver le TIR.

N) L'objet ligne résiduelle.

Cet objet possède une opération finale : calcul résiduel de la ligne.

O) L'objet ligne financière.

Cet objet contient trois opérations finales. Elles se nomment: actualiser une valeur, actualiser une ligne de valeur, valeur actuelle nette d'une ligne.

Chapitre 3. Spécification du problème de l'investissement

Ce chapitre se divise principalement en deux parties. La première traite des spécifications d'un point de vue théorique et situe le langage GLIDER par rapport aux autres langages de spécification en énonçant les principales caractéristiques de ce langage. La deuxième partie, la plus conséquente, applique GLIDER au domaine des problèmes de l'investissement.

3.1. Présentation théorique des spécifications.

L'introduction traitait de l'ingénierie des besoins et de conception. Dans l'ingénierie des besoins, quatres phases ont été mises à jour : l'élicitation, la modélisation, l'analyse, la validation. C'est pendant la phase de modélisation que le problème est spécifié.

Le fait de spécifier un problème évite de penser directement en terme de programme. Le concepteur peut se concentrer pleinement sur le travail de modélisation sans avoir à faire des choix d'implémentation. Une bonne spécification doit être aussi générale que possible.

Pour spécifier un problème, plusieurs choix s'offrent au concepteur. Il peut spécifier soit de manière informelle, rigoureuse ou formelle
Spécifier de manière informelle, c'est s'exprimer en langage naturel. Le désavantage majeur provient de l'ambiguïté du langage naturel, c-à-d qu'une même spécification admet différentes interprétations.
Spécifier de manière rigoureuse, c'est exprimer le problème de manière structurée, cela permet de guider le développement mais cela ne règle pas totalement le problème de l'ambigüité.
Spécifier de manière formelle, c'est exprimer un problème au moyen d'un formalisme mathématique dont la sémantique est univoque (p.ex.: logique du premier ordre).

3.1.1. Avantages et inconvénients de spécifier de manière formelle.

Ecrire des spécifications de manière formelle rend possible l'utilisation d'outils permettant des contrôles automatiques, la conversion de spécification dans un autre formalisme, la génération de prototype.
Le checking est une action de contrôle.
Potentiellement, pour tout langage formel, on peut bâtir des outils qui vérifient la cohérence et cela à différents niveaux. Par exemple, à un niveau élémentaire, l'analyse se ramène à vérifier la syntaxe. Un contrôle du typage peut compléter le premier type d'analyse. Enfin, des outils plus élaborés pourront détecter des incohérences entre morceaux de spécification et déduire les propriétés d'une spécification.
Les outils de conversion [Fic87] ont pour but de traduire les spécifications en langage naturel structuré ou sous forme de représentation graphique. Ceci est fait pour pallier à l'illisibilité des spécifications formelles et intervient au niveau de la phase de validation.
Les outils de prototype servent à produire de manière semi-automatique une maquette du futur logiciel ou d'une partie de celui-ci.

Spécifier de manière formelle est intéressant pour de grosses applications. L'exemple le plus connu est : "The CICS experience". Il s'agit d'un système d'exploitation qui a été spécifié en langage Z (un autre langage de spécification). Spécifier de manière formelle est particulièrement utile pour des applications à temps partagé [Dub90]. Un des exemples les plus classique fût de spécifier les applications de vol du programme A7 [Hen80] (mieux connu sous le nom de l'avion: YF16). Enfin, la disponibilité d'outils de contrôle rend les spécifications formelles indispensables à des domaines où le fonctionnement des logiciels doit être sévèrement garanti (logiciels de gestion de satellites ou de missiles, logiciels de contrôle de centrale nucléaire,...).

Un langage de spécification peut être caractérisé par trois propriétés: son expressivité,ses mécanismes de structuration et son aspect formel. L'expressivité est la facilité d'exprimer les concepts que l'on veut formaliser, c-à-d la facilité avec laquelle le concepteur peut exprimer les concepts du monde réel en utilisant les concepts du langage .
Les mécanismes de structuration sont les constructions du langage permettant d'organiser les spécifications.

L'aspect formel est le fait d'avoir des règles d'interprétation qui garantissent l'absence d'ambiguïté.

3.1.2. Le langage GLIDER.

Le langage GLIDER [Dub91a], [Fun] est un langage de spécification formel. GLIDER possède un bon pouvoir d'expression et dispose de mécanismes de structuration puissants. Une spécification GLIDER est organisée en un ensemble de "clusters". Les clusters peuvent être liés par deux mécanismes de structuration : l'héritage et la généricité.
L'héritage dans GLIDER permet d'utiliser la réutilisation de propriétés définies au sein d'un cluster à partir d'un autre cluster . Cet héritage est simple et il autorise le renomage.
La généricité dans GLIDER permet de paramétrer les clusters. Un cluster paramétré est un cluster dont l'interprétation dépend du ou des paramètre(s) effectif(s) spécifié(s) à l'instanciation.

Les clusters sont des boîtes contenant une structure de données et des services offert sur la structure de données. La structure de données est définie au moyen d' une expression de type. Une expression de type est constituée sur base de types simples prédéfinis (BOOLEAN, TIME, DURATION, REAL, INT, STRING) groupés au moyen de constructeurs de types prédéfinis (CARTESIAN PRODUCT, SEQUENCE, SET, BAG).

Les services offerts sur la structure de données (les opérations) s'expriment au moyen de la logique du premier ordre typé.

L'expressivité est favorisée par l'existence de types et d'opérations prédéfinis. Prenons deux cas appliqués au problème de l'investissement:
-dans les problèmes d'investissement les observations associées à l'état du système varient avec le temps. En GLIDER, la notion OBJECT a été introduite pour répondre à ce besoin,
- dans les problèmes d'investissement le concept de ligne est employé. Cela se traduit dans le langage GLIDER par l'utilisation du type constructeur SEQUENCE.

GLIDER supporte la notion de spécifications incomplètes (c'est à dire qu'une interprétation peut être donnée même à une spécification

non-terminée). GLIDER permet l'expression des contraintes temporelles avec temps explicite telles que "cette propriété est vraie pendant trois minutes".

GLIDER est assez riche, donc complexe à utiliser; par conséquent, avant de s'attaquer à un nouveau problème, il serait bon de définir une série de clusters génériques typiques du domaine d'application qui permettraient de spécifier le problème en termes plus proches de l'application. Une fois définis, ces clusters pourront être réutilisés à souhait.
Cependant, la gestion d'une bibliothèque de composants réutilisables pose des problèmes à deux niveaux: au niveau du choix des composants à réutiliser dans le cadre d'une application donnée et au niveau de la manière d'adapter les composants au contexte dans lequel le concepteur va les utiliser.

3. 2. Présentation pratique des spécifications.

Cette partie est divisée en trois parties. La première définit la structure de donnée de manière formelle, la deuxième concerne la résolution, la troisième est un glossaire reprenant toutes les spécifications.

3. 2.1. Définition de la structure de données.

TABLEAU is CP [Tabi : CORPSTABLEAU, Nom: STRING]

CORPSTABLEAU is SEQ [LIGNE]

LIGNE is CP [Libellé : STRING, Données : LIGNE-DONNEES]

LIGNE-DONNEES is CP [Choix : UNION [LIGNE DE VALEUR,
LIGNE DE TEMPS,
LIGNE DE RESULTAT,
LIGNE DE FORMULE]
Typeligne : STRING]

LIGNE DE VALEUR is OBJECT [Chiffre *]

LIGNE DE TEMPS is SEQ [TIME]

LIGNE DE FORMULE is
UNION [LIGNE FINANCIERE, LIGNE AMORTISSEMENT LINEAIRE, LIGNE IMPOT,LIGNE BENEFICE IMPOSABLE, LIGNE AMORTISSEMENT DEGRESSIF,LIGNE TIR, LIGNE RESIDUELLE].

LIGNE DE RESULTAT is
CP [Res : LIGNE DE VALEUR, Expression : EXPRESSION]

EXPESSION is UNION [SOMME, DIVISION, PRODUIT]

LIGNE AMORTISSEMENT LINEAIRE is
[Bien : INTEGER, Durée : INTEGER, Am :LIGNE DE VALEUR]

LIGNE FINANCIERE is
CP [Tempsinit : INTEGER, Tempsfin : INTEGER, Taux : INTEGER,
Fin : LIGNE DE VALEUR, Lignederéférence : INTEGER]

LIGNE IMPOT is
CP [Tauximposé : REAL, Imp : LIGNE DE VALEUR,
Numligneréférence : INTEGER]

LIGNE BENEFICE IMPOSABLE is
[Ben : LIGNE DE VALEUR, Noligneréférence : INTEGER]

LIGNE TIR is
CP [Valtir : LIGNE DE VALEUR, Numéro : INTEGER,
Tempsinit: INTEGER, Tempsfinal : INTEGER]

LIGNE AMORTISSEMENT DEGRESSIF is
CP [Deg : LIGNE DE VALEUR, Temps : INTEGER, Bien : INTEGER]

DIVISION is
CP [Dividende : INTEGER, Diviseur : INTEGER Signe : SIGNE]

SOMME is
SEQ [CP [Signe : SIGNE, Terme : INTEGER]]

PRODUIT is
SEQ [CP [Signe : SIGNE, Terme : INTEGER]

LIGNE RESIDUELLE is
CP [Res : LIGNE DE VALEUR, Numérolignevalres : INTEGER,
Numéroligneamo : INTEGER]

3. 2. 2. Spécification du problème de l'investissement.

Remarque: les trois dernières lettres des fonctions correspondent au cluster utilisé. Voici les abréviations ainsi que les clusters associés.

tab	TABLEAU
cta	CORPSTABLEAU
lig	LIGNE
ldo	LIGNE-DONNEES
lva	LIGNE DE VALEUR
lte	LIGNE DE TEMPS
lfo	L I G N E D E FORMULE
lre	L I G N E D E RESULTAT
exp	EXPRESSION
lal	LIGNE AMORTISSEMENT LINEAIRE
lfa	LIGNE FINANCIERE
lim	LIGNE IMPOT
lbi	LIGNE BENEFICE IMPOSABLE
lti	LIGNE TIR
lad	LIGNE AMORTISSEMENT DEGRESSIF
div	DIVISION
som	SOMME
pro	PRODUIT
lre	LIGNE RESIDUELLE

TABLEAU
is CP [Tabi : CORPS-TABLEAU, Nom : STRING]

But : dans ce cluster, toutes les fonctionnalités demandées sur les lignes sont répertoriées.

Exportation: tout

Accumulertab(v,taux,temps1,temps2,noligne,t) = t'

REAL X REAL X TIME XTIME X INTEGERX TABLEAU
--> TABLEAU

But: opération intermédiaire qui, à partir , d'un numéro de ligne(noligne), d'un tableau(t), permet à partir d'une valeur donnée(v) de l'accumuler entre la période de temps considérée(temps1, temps2) à un taux donné(taux). Cette opération fournit un nouveau tableau(t').

Pré: $v > 0 \wedge taux > 0 \wedge 0 < temps1 < temps2 \wedge noligne > 0$

Post : Tabi(t)=z ∧ Créationd'uneligneacumulationcta(z,noligne)=z'
∧Corpstableauaccumulationcta(z',v,temps1,temps2,noligne,taux)=z"
∧ Calculertableaucta(z")= z''' ∧ Tabi(t') = z''' ∧ Nom(t') = Nom(t)

Variables: z,z',z",z''': CORPSTABLEAU

Accroissementvaleurtab(v,accrois,temps1,temps2,noligne,t)=t'

REAL X REAL X TIME X TIME X INTEGER X TABLEAU
--> TABLEAU

But:opération intermédiaire qui, à partir d'un tableau(t), permet à partir, d'un numéro de ligne(noligne), d'une valeur donnée(v) de l'accroître dans la période de temps considérée.(temps1, temps2) à un taux donné(accrois). Cette opération fournit un nouveau tableau(t').

Pré :v> 0 ∧ accrois > 0 ∧ 0<temps1<temps2 ∧ noligne > 0

Post :Tabi(t) = z ∧ Créationd'uneligneaccroissementcta (z,noligne)=z'
∧ Insérеraccroissementcta(z',v,accrois,temps1,temps2,noligne)=z"
∧ Calculertableaucta(z")= z"' ∧ Tabi(t')=z"'∧ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Actualisationtab(nlcash,nlact,débpér,finpér,taux,t)=t'

INT X INT X TIME X TIME X REAL X TABLEAU -->TABLEAU

But:opération intermédiaire qui à partir d'un tableau(t), d'un numéro de ligne de cash(nlcash), d'un numéro de ligne actualisées(nlact) permet d'actualiser une ligne dans la période de temps considérée.(débpér1, débper2) à un taux donné(taux). Cette opération fournit un nouveau tableau(t').

Pré: 1≤nlcash≤Length(Tabi(t)) ∧ 1≤nlact≤Length(Tabi(t))∧taux >0
∧ 0 < dépér < finpér

Post : Tabi(t)=z ∧ Créationd'uneligneactualiséecta(z,nlact)=z'
∧ Calculactualisationcta(nlcash,nlact,débpér,finpér,taux,z')=z"
∧ Calculertableaucta(z")=z"'∧ Tabi(t')=z"' ∧ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Amortissementdégressiftab(pér,nlamdeg,nl,t)=t'

INT X INT X INT X TABLEAU ---> TABLEAU

But: opération intermédiaire qui permet à partir d'un tableau(t) de calculer un amortissement dégressif. Le nombre d'années est donné(pér) ainsi que le numéro de la ligne de l'amortissement(nlamdeg) et le numéro de la ligne de biens à amortir(nl). Cette opération donne un nouveau tableau(t').

Pré: pér > 0 ∧ 1≤nlamdeg≤Length(Tabi(t)) ∧ 1≤n1≤Length(Tabi(t))

Post Tabi(t)=z
∧ Créationd'uneligneamortissementdégressifcta(z,nlamdeg)=z'
∧ Calculamortdégressifcta(z',pér,nl,nlamdeg)=z"
∧ Calculertableaucta(z")=z"' ∧ Tabi(t')=z"'∧ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Amortissementlinéairetab(nd,nv,nal,t)=t'

INTEGER X INTEGER X INTEGER X TABLEAU --> TABLEAU

But: opération intermédiaire permettant d'effectuer un amortissement linéaire à partir d'un tableau(t), de deux lignes. Une qui est la ligne de temps(nd) et la seconde qui est la ligne des biens à amortir.(nv) Pour calculer l'amortissement , le numéro de la ligne amortissement linéaire doit être donné(nal). Cette opération donne lieu à un nouveau tableau(t').

Pré : $1 \leq nd \leq Length(Tabi(t)) \wedge 1 \leq nv \leq Length(Tabi(t))$
$\wedge 1 \leq nal \leq Length(Tabi(t))$

Post: Tabi(t)=z
$\wedge$ Créationd'uneligneamortissementlinéairecta(z,nval)=z'
$\wedge$ Calculamortlinéairecta(z',nd,nv,nal)=z"
$\wedge$ Calculertableaucta(z") =z'" $\wedge$ Tabi(t')=z"' $\wedge$ Nom(t')=Nom(t)

Variables: z,z',z",z'": CORPSTABLEAU

Avoirlignevaleurtab(nl,t)=t'

INTEGER X TABLEAU ---> TABLEAU

But: opération intermédiaire qui permet de travailler sur une ligne de valeurs dont le numéro est demandé(nl) ainsi que le tableau(t). Cette opération fournit un nouveau tableau(t').

Pré: $1 \leq nl \leq Length(Tabi(t))$

Post :Tabi(t)=z $\wedge$ Créationd'unelignevaleurcta(z,nl)=z'
$\wedge$ Mizeàzérodelalignecta(z',n)=z"
$\wedge$ Calculertableaucta (z")=z'"$\wedge$ Tabi(t')=z'" $\wedge$ Nom(t')=Nom(t)

Variables: z,z',z",z'": CORPSTABLEAU

Bénéficeimposabletab(nlres,nlbenimp,t)=t'

INTEGER X INTEGER X TABLEAU ---> TABLEAU

But : Opération intermédiaire qui permet de calculer une ligne de bénéfices imposables dont le numéro est donné(nlbenimp) à partir d'un tableau(t), d'une ligne de résultats dont le numéro de la ligne est donné(nlres). Cette opération fournit un nouveau tableau(t').

Pré : $1 \leq nlres \leq Length(Tabi(t)) \wedge 1 \leq nlbenimp \leq Length(Tabi(t))$

Post:Tabi(t)=z
$\wedge$ Créationd'unelignebénéficeimposablecta(z,nlbenimp)=z'
$\wedge$ Calculbénéficeimposablecta(nlres,llbenimp,z')=z"
$\wedge$ Calculertableaucta(z")=z"'$\wedge$ Tabi(t')=z"'$\wedge$ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Créertableautab (nom) = t

STRING -> TABLEAU

But : opération finale qui crée un tableau(t) vide avec un titre(nom).

Pré : /

Post : Nom(t) = nom $\wedge$ Tabi(t') = Initialisationcta ()

Etendreunelignedansuntableautab(t)=t'

TABLEAU --> TABLEAU

But : opération intermédiaire qui vise à rajouter une ligne à la fin du tableau(t) et qui donne un nouveau tableau(t').

Pré : Nom (t) ≠ " "

Post: Tabi (t) = z ∧ Tabi(t') =z'∧ Etendrelignecta (z)=z'
∧ Nom(t')=Nom(t)

Variables : z, z': CORPSTABLEAU

Impôttab(nlbenimp,nlimp,taux,t)=t'

INT X INT X REAL X TABLEAU X --> TABLEAU

But : opération intermédiaire permettant de calculer une ligne d'impôts dont le numéro est donné(nlimp) à partir d'un tableau(t), d'une ligne de bénéfices imposables(nlbenimp), d'un taux(taux). Cette opération fournit un nouveau tableau(t').

Pré : 1≤nlbenimp≤Length(Tabi(t)) ∧ 1≤nlimp≤Length(Tabi(t))
∧ taux > 0

Post Tabi(t)=z∧ Créationd'uneligneimpôtcta(z,nlimp)=z'
∧ Calculimpôtcta(nllimp, nlbenimp,taux,z')=z"
∧ Calculertableaucta(z")=z"'
∧ Tabi(t')=z"' ∧ Nom(t')=Nom(t')

Variables: z,z',z",z"' : CORPSTABLEAU

Inscrirelibellétab(t, nom,nl) = t'

TABLEAU X STRING X INTEGER

But : opération intermédiaire permettant d'écrire un libellé(nom) dont le numéro de ligne est référencé(nl) dans le tableau(t). Cette opération fournit un nouveau tableau.(t').

Pré: $1 \leq nl \leq Length(Tabi(t)) \wedge nom \neq$ '...'

Post:Tabi(t) = z $\wedge$ Nom(t')=Nom(t)
$\wedge$Tabi(t')=z
except
(z[nl] $\wedge$ Ecrirelibellélig(nom,z[nl]))

Variable: z: CORPSTABLEAU

Insérerlignedansuntableautab(t,x) = t'

TABLEAU X INTEGER --> TABLEAU

But: opération intermédiaire qui permet d'insérer une ligne dont le numéro est référencé(x) dans un tableau(t). Cette opération fournit un nouveau tableau(t').

Pré : Nom(t) $\neq$ " " $\wedge$ 1< x <Lenght(Tabi(t))

Post: Tabi(t) = z $\wedge$ Remodifiercorpscta(x,z) = z'
$\wedge$ Calculertableaucta(z')= z"
$\wedge$ Tabi(t')= z" $\wedge$ Nom(t')=Nom(t)

Variables: z,z',z": CORPSTABLEAU

Limitertailletableautab (t,x)=t'

TABLEAU X INTEGER --> TABLEAU

But : Opération intermédiaire qui a pour but que toutes les lignes supérieures à un numéro de ligne référencée(x) d'un tableau(t) soient supprimées.Cette opération fournit un nouveau tableau(t')

Pré : 1< x < Length (Tabi(t))

Post : Tabi (t) = z ∧ Limitertaillecta(z,x) = z' ∧ Tabi(t') = z'
∧ Nom(t')=Nom(t)

Variables: z,z' : CORPSTABLEAU

Modifierlesdeuxtempstab(t,i,j)=t'

TABLEAU X TIME X TIME ---> TABLEAU

But : opération intermédiaire qui permet de modifier le temps initial(i) et le temps final(j) d'un tableau(t). Cette opération fournit un nouveau tableau(t').

Pré : 0< i <j ∧ Isoflignedetemps(Choix(Données(Tabi(t)[1])))
∧Vérificationtableauzérocta(Tabi(t))

Post : Tabi(t)=z∧ Choix(Données(z[1]))=x
∧ z=z' except
(Choix(Données(z'[1]))=x'∧
Modificationintervalledetempslte(x,i,j)=x')
∧Tabi(t')=z'

Variables x,x': LIGNE

Modifiervaleurtab(t,n,année1,année2)= t'

TABLEAU X INTEGER X TIME XTIME-->TABLEAU

But: opération intermédiaire qui permet de supprimer des valeurs dans un intervalle de temps(année1,année2) dont le numéro de ligne (n) d'un tableau(t) est réferencé. Cette opération fournit un nouveau tableau(t').

Pré : 1< n< Length(Tabi(t)) ∧ 0 < année1 <année2

Post : Saisirvaleurtab(t,n,année1,année2) = t'

Réinitialisationdutableautab (t) = t'

TABLEAU --> TABLEAU

But : opération intermédiaire qui réinitialise le tableau(t) et qui donne un nouveau tableau(t').

Pré : Nom (t) ≠ " " ∧ ∃ Tabi (t)

Post : Initialisationcta()=z' ∧ Nom(t') = " " ∧ Tabi(t')=z'

Variables: z': CORPSTABLEAU
Nom(t): STRING

Résiduelletab(nlam,nlbien,nlresi,t)=t'

INT X INT X INT X TABLEAU ---> TABLEAU

But: opération intermédiaire permettant de produire une ligne donnant la valeur résiduelle du tableau(t) dont le numéro de la ligne est précisé(nlresi) à partir d'une ligne amortissements dont le numéro de la ligne est précisé(nlam) et une ligne de valeurs dont le numéro est précisé(nlbien). Cette opération fournit un nouveau tableau(t').

Pré : $1 \leq nlam \leq Length(Tabi(t)) \wedge 1 \leq nlbien \leq Length(Tabi(t))$
$\wedge 1 \leq nlresi \leq Length(Tabi(t))$

Post:Tabi(t)=z $\wedge$ Créationd'unelignerésiduellecta(z,nlresi)=z'
$\wedge$ Calculrésiduellecta(z',nlam,nlresi,nlbien)=z"
$\wedge$ Calculertableaucta(z")=z"'
$\wedge$ Tabi(t')=z"' $\wedge$ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Résultattab(expression,nlres,t)=t'

EXPRESSION X INTEGER X TABLEAU--> TABLEAU

But: Opération qui fournit un nouveau tableau(t') à partir d'une expression(expression),d'un numéro de ligne de résultats(nlres),d'un tableau(t). Opération intermédiaire qui permet de calculer une ligne de résultats.

Pré : $1 < nlres \leq Lenght(Tabi(t))$

Post :Tabi(t) = z $\wedge$ Créationd'unelignerésultatcta (z,nlres)=z'
$\wedge$ Calculrésultatcta(expression,nlres,z')=z"
$\wedge$ Calculertableaucta(z")= z"' $\wedge$ Tabi(t')=z"' $\wedge$ Nom(t')=Nom(t)

Variables: z,z',z",z"': CORPSTABLEAU

Saisirvaleurtab(t,n,année1,année2) = t'

TABLEAU X INTEGER X TIME X TIME --> TABLEAU

But opération intermédiaire qui permet de saisir des valeurs.à partir d'un numéro de ligne(n), d'un tableau(t), d'un intervalle de temps(année1,année2). Cette opération fournit un nouveau tableau(t').

Pré : n>0 Tabi(t) = z ∧ 1≤ n ≤Length(Tabi(t))
∧ Isoflignedevaleur (Choix(Données(z[n])))
∧Typeligne(Données(z[n])) = 'valeur' ∧ 0 < année1 < année2

Post : Insérervaleurcta(z,n,année1,année2)=z'
∧ Calculertableaucta(z')=z"
∧ Tabi(t')=z" ∧ Nom(t')=Nom(t)

Variables: z,z',z": CORPSTABLEAU

Supprimerlignedutableautab(x,t) = t'

INTEGER X TABLEAU --> TABLEAU

But : opération intermédiaire qui permet de supprimer une ligne du tableau(t) à partir d'un numéro de ligne de données(x) Cette opération fournit un nouveau tableau(t').

Pré : Nom(t) ≠ " " ∧ 1≤x≤Length(Tabi(t))

Post : Supprimercorpscta(z,x) = z' ∧ Calculertableaucta(z') = z"
∧ Tabi(t') = z" ∧ Tabi(t) = z ∧ Nom(t')=Nom(t)

Variables: z,z': CORPSTABLEAU

Supprimerunlibellétab(t,n) = t'

TABLEAU X INTEGER --> TABLEAU

But: opération intermédiaire qui permet de supprimer le libellé de la ligne à partir d'un numéro de ligne(n) du tableau(t). Cette opération fournit un nouveau tableau(t').

Pré : 1< n ≤ Length(Tabi(t))

Post: Tabi(t) = z ∧ Nom(t')=Nom(t)
∧ Libellé(Tabi(t)[n])=" ".

Variables : z,z' : CORPSTABLEAU

Supprimervaleurtab(t,n,année1,année2)= t'

TABLEAU X INTEGER -->TABLEAU

But: opération intermédiaire qui permet de supprimer des valeurs d'une ligne d'un tableau(t) dont le numéro de la ligne(n), l'intervalle de temps(année1,année2) est donné Cette opération fournit un nouveau tableau(t').

Pré :1< n ≤ Length(Tabi(t)) ∧ 0 < année1 < année2

Post : Saisirvaleurtab(t,n,année1,année2) = t'

Tirtab(nltir,nlres,t,débpér,finpér)=t'

INTEGER X INTEGER X TABLEAU X TIME X TIME
---> TABLEAU

But: opération intermédiaire permettant de calculer le taux interne de rentabilité à partir d'un numéro de ligne de résultats(nlres), d'un numéro de ligne tir(nltir),du tableau(t), de la période d'intervalle(débpér,finpér). Cette opération fournit un nouveau tableau(t').

Pré: Tabi(t)=z ∧ 1≤nltir≤Length(Tabi(t)) ∧ 1≤nlres≤Length(Tabi(t))
0 < déppér < finpér

Post: Créationd'unelignetircta(z,nltir)=z'
∧ Calcultirct(z',nltir,nlres,déppér,finpér)=z"
∧ Calculertableaucta(z")=z"'
∧ Tabi(t')=z"' ∧ Nom(t')=Nom(t)

Variables : z,z',z",z"':CORPSTABLEAU

Valeurrépétitivetab(t,v,temps1,temps2,n) = t'

TABLEAU X REAL X TIME X TIME X INTEGER --> TABLEAU

But : opération intermédiaire qui permet à une valeur donnée(v) de se répéter dans un intervalle de temps(temps1,temps2) dans une ligne donnée du tableau(t) qui est référencée par un numéro(n). Cette opération fournit un nouveau tableau(t').

Pré: Tabi(t)= z ∧ 1≤n≤Length(Tabi(t))
∧ 0< temps1 <temps2
∧ Testertempslte(Choix(Données(z[1])), temps1,temps2)

Post Créationd'unelignevaleurrépétitivecta(z,n)=z'
∧ Insérervaleurlrépcta(v,temps1,temps2,n,z')=z"
∧ Calculertableaucta(z")=z"' ∧ Tabi(t') = z"'.∧ Nom(t')=Nom(t)

Variables: n: INTEGER;
ld: LIGNE - DONNEES
z,z' : CORPSTABLEAU

Vantab(t,débpé,finpé,taux,nolignecashflow,nolvalactunette)=t'

TABLEAU X TIME X TIME X REAL X INTEGER X INTEGER
-->
TABLEAU

But: opération intermédiaire permettant de donner la valeur actuelle nette d' un tableau(t) dont le numéro de ligne(novalactunette), le numéro de ligne de cash flows(nolignecashflow), l' intervalle de temps(débpé,finpé), le taux(taux) sont donnés . Cette opération donne lieu à un nouveau tableau(t').

Pré: Tabi(t)=z ∧ 1≤ nolignecashflow ≤Length(Tabi(t))
∧ 1≤nolvalactunette≤Length(Tabi(t)) ∧ taux > 0 ∧ 0< déppé< finpé

Post : Créationd'unelignevancta(z,nolvalactuelle)=z'
∧Calculvancta(z',débpé,finpé,taux,nolignecashflow,nolvalactuelle) =z"
∧ Calculertableaucta(z") = z"' ∧ Tabi(t') = z"' ∧ Nom(t')=Nom(t)

Variables : z,z',z",z"': CORPSTABLEAU

CORPSTABLEAU

is SEQ [LIGNE]

But : cluster pemettant d'effectuer une série d'opérations d'actions sur le corps du tableau et permettant également une série d'opérations intermédiaires.

Invariants :

1)Isoflignedetemps(Choix(Données(z[1]))

2) $\forall i,j : 1 \leq i,j \leq$ Lengtht(z)

$\Rightarrow$ (Length(Choix(Données(z_i)) = Length(Choix(Données(z_j)

$\wedge \exists !$ Isoflignedetemps(Choix(Données(z[i]))

$\wedge \exists !$ Isofligned'impôt(Choix(Donnéesz[j]))

$\wedge \exists !$ Isoflignebénéficeimposable(Choix(Donnéesz[j])))

3) $\forall i,j\ 1 \leq i,j \leq$ Length(z), $\forall k\ 1 \leq k \leq$ Length(t)

$\Rightarrow$ Time(Choix(Données(z[i], [k])))

= Time(Choix(Données (z[j], [k])))

4)$\forall i,j\ 1 \leq i,j \leq$ Length(z)$\Rightarrow$ Length z[i]=Length z[j]

5)$\forall i,j\ 1 \leq i \leq$ Length(z), $\forall k\ 1 \leq k \leq$ Length(t)

$\Rightarrow$ Time(Choix(Données(z[i], [k])))

= First(Choix(Données(z[k])))

Variables: z: CORPSTABLEAU

i,j,k: INTEGER

Exportation: tout exepté Vérificationtableauzérocta, Blanclibellécta, Initialiserlignecta, Initialisertempscta.

Blanclibellécta(z)=z'

CORPSTABLEAU --> CORPSTABLEAU

But : opération intermédiaire qui met tous les libellés à blanc du corpstableau(z').à partir d'un corpstableau(z)

Pré: $\exists$ z

Post: $\forall$ i : $2 \leq i \leq$ Length(z) $\Rightarrow$

Libellé(z'[i])=Miseàblanclibellélig(z [i])

Variable: i : INTEGER

Calculactualisationcta(nlcash,nlact,débpér,finpér,taux,z)=z'

INT X INT X INT X TIME X TIME X REAL X CORPSTABLEAU
--> CORPSTABLEAU

But opération intermédiaire permettant de calculer une ligne actualisée à partir d'un numéro (nlcash) de ligne de valeurs ou de ligne de résultats.et d'un numéro de ligne valeur actuelle(nlact) d'un corpstableau(z). La période(dépér,finpér) de temps dans laquelle la valeur actuelle doit être inscrite, ainsi que le coût du capital(taux). Cette opération donne un nouveau corpstableau(z').

Pré: $1 \leq nlact \leq Length(z) \wedge 1 \leq nlcash \leq Length(z)$
$\wedge$ Choix(Données(z[nlact]))=lact
$\wedge$ Choix(Données(z[nlcash]))=lval
$\wedge$ Isoflignedeformule(lact)
$\wedge$ (Isoflignedeformule(lact) $\Rightarrow$Isoflignefinancière(lact))
$\wedge$ (Isoflignedevaleur(lval) $\vee$ Isoflignederésultat(lval))
$\wedge$ taux > 0 $\wedge$ 0 < débpér<finpér

Post: Choix(Données(z' [nact]))=lact' $\wedge$ lv=<lval,Res(lva)>
z' = z except
(Taux(lact')=taux
$\wedge$ Tempsfin(lact')=finpér $\wedge$Tempsinit(lact')=déppér
$\wedge$ Fin(lact')=Actualiséligne1fi(lv,taux,débpér,finpér))
$\wedge$ Lignederéférence(lact')=nlcash)

Variables: lact,lval,lact': LIGNE-DONNEES
lv: LIGNE DE VALEUR

Calculamortdégressifcta(z,pér,nl,nlamdeg)=z'

CORPSTABLEAU X INT X INT X INT --> CORPSTABLEAU

But: opération intermédiaire permettant de calculer l'amortissement dégressif à partir d'un corpstableau(z), d'un numéro(nl) de ligne de valeurs ou de résultats, d'un numéro de ligne de l'amortissement dégressif(nlamdeg) et une durée(pér) unique de déduction. Un nouveau corpstableau(z') est produit.

Pré: $1 \leq nl \leq Length(z) \wedge 1 \leq nlamdegl \leq Length(z)$
$\wedge$ (Isoflignedevaleur(lv) $\vee$Isoflignerésultat(lv))
$\wedge$ Isoflignedeformule(lad)
$\wedge$
(Isoflignedeformule(lad)$\Rightarrow$Isofligneamortissementdégressif(lad))
$\wedge$ Choix(Données(z[nlamdegl]))=lad $\wedge$ Choix(Données(z[nl]))=lv
$\wedge$ pér > 0

Post :Choix(Données(z'[nlamdeg]))=lad' $\wedge$ lv1=<lv,Res(lv)>
$\wedge$ z' = z except
(Deg(lad')=Calculamortdégressiflad(lv1,Deg(lad),pér)
$\wedge$ Bien(lad')=nv $\wedge$ Temps(lad')=pér
$\wedge$ Temps(lad')=pér)

Variables: lad,lv,lad': LIGNE-DONNEES
lv1: LIGNE DE VALEUR

Calcullamortineairecta(z,nd,nv,nal) =z'

CORPSTABLEAU X INT X INT X INT -->CORPSTABLEAU

But: opération intermédiaire permettant de calculer l'amortissement linéaire(nal) à partir d'un corpstableau(z) de deux numéros de lignes: le numéro(nd,nv) correspondant à la ligne de durée de l'amortissement et la ligne correspondant à la ligne de biens à amortir. Cette opération fournit un nouveau corpstableau(z').

Pré:1≤nd≤Length(z) ∧ 1≤nv≤Length(z)
∧ 1≤nal≤Length(z) ∧ Islignedevaleur(Choix(données(z[nd])))
∧ Isoflignedevaleur(Choix(Données(z[nv])))
∧ Isoflignedeformule(lal)
∧ Choix(Données(nlal))=lal
∧ (Isoflignedeformule(lal)⇒Isofligneamortissementlinéaire(lal)

Post : Choix(Données(z[nv]))=lvv∧ Choix(Données(z[nd]))=lvd
∧ Choix(Données(z' [nal]))=lal'
z=z'except
(Am(lal')=Calculamortlinéairelal(lvv,lvd)
∧ Bien(lal')=nv ∧ Durée(lal')=nd)

Variables: lal,lal':LIGNE-DONNEES
lvv,lvd:LIGNE DE VALEUR

Calculbénéficeimposablecta(nlres,nlbenimp,z)=z'

INT X INT X CORPSTABLEAU --> CORPSTABLEAU

But: opération intermédiaire qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne bénéfices imposables(nlbenimp), d'un numéro de ligne résultats(nlbenimp), d'un corpstableau(z).

Pré: $\wedge$ 1≤nlres≤Length(z) $\wedge$ 1≤nlbenimp≤Length(z)
$\wedge$ Choix(Données(z[nlbenimp]))=lbi
$\wedge$ Choix(données(z[nlres]))=lr $\wedge$ Isoflignederésultat(lr)
$\wedge$ Isoflignedeformule(lbi)
$\wedge$ (Isoflignedeformule(lbi)⇒Isoflignebénéficeimposable(lbi)

Post: Choix(Donnée(z'[nlbenimp])=lbi'
z=z' except
(Ben(lbi') = Calculbénéficeimposablelbi(Res(lr))
$\wedge$ Nolignerréférence(lbi') = nlres)

Variables: lr,lbi,lbi': LIGNE-DONNEES

Calculimpôtcta(nlimp,nlbenimp,taux,z)=z'

INT X INT X REAL X CORPSTABLEAU --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne bénéfices imposables(nlbenimp), d'un numéro de ligne impôts(nlimp), d'un corpstableau(z), d'un taux(taux). Opération intermédiaire permettant de calculer les impôts.

Pré: $1 \leq nlimp \leq Length(z) \wedge 1 \leq nbenimp \leq Length(z)$
$\wedge$ Choix(Donnée(z[nlbenimp]))=lbi
$\wedge$ Choix(données(z[nlimp]))=li $\wedge$ Isoflignedeformule(li)
$\wedge$ Isoflignedeformule(lbi)
$\wedge$ (Isoflignedeformule(li)$\Rightarrow$Isofligneimpôt(li))
$\wedge$ (Isoflignedeformule(lbi)$\Rightarrow$Isoflignebénéficeimposable(lbi)
$\wedge$ taux > 0
Post:Choix(Donnée(z'[nlimp])=li'
z=z' except
Taux(li')=taux $\wedge$ Imp(li')=Calculimpôtlim(taux, Ben(lbi))
$\wedge$ Numligneréférence(li')=nlbenimp

Variables: li,lbi,li': LIGNE-DONNEES

Calculrésiduellecta(z,nlam,nlresi,nlbien)=z'

CORPSTABLEAU X INT X INT X INT --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne amortissements(nlam), d'un numéro de ligne résiduelles(nlimp), d'un corpstableau(z), d'un numéro de ligne de valeurs(nlbien). Opération intermédiaire qui permet de calculer une ligne de valeurs résiduelles à partir d'un numéro de ligne amortissements et un numéro de ligne résultats ou valeurs.

Pré: $1 \leq nlbien \leq Length(z) \wedge 1 \leq nlresi \leq Length(z)$
$\wedge\ 1 \leq nlam \leq Length(z)$
$\wedge$ Choix(Données(z[nlam])=la
$\wedge$ Choix(Données(z[nlbien])=lb $\wedge$ Choix(Données(z[nlresi])=lr
$\wedge$ Isoflignedeformule(la)
$\wedge$ ((Isoflignedeformule(la)$\Rightarrow$Isofligneamortissementlinéaire(la)
$\vee$
(Isoflignedeformule(la)$\Rightarrow$Isofligneamortissementdégressif(la))
$\wedge$ (Isoflignederésultat(lb)$\vee$ Isoflignedevaleur(lb))
$\wedge$ (Isoflignedeformule(lr)$\Rightarrow$Isoflignerésiduelle(lr))
$\wedge$ Isoflignedeformule(lr)

Post: lvbien=<lb,Res(lb)> $\wedge$ lvam=<Am(la),Deg(la)>
$\wedge$ Choix(Données(z'[nlresi]))=lr'
$\wedge$ z=z'except
(Resi(lr')=Calculrésiduellelre(lvbien,lvam)
$\wedge$ Numéroligneval res(lr')=nlbien
$\wedge$ Numéroligneamo(lr')=nlam)

Variables:la,lb,lr,lr': LIGNE-DONNEES
lvbien,lvam: LIGNE DE VALEUR

Calculrésultatcta(expression,nlres,z)=z'

EXPRESSION X INTEGER X CORPSTABLEAU
--> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne de résultats(nlres), d'une expression(expression), d'un corpstableau(z). Opération finale qui calcule une ligne de résultats.

Pré: $1 \leq nlres \leq Length(z) \wedge Choix(Données(z[nlres])) = lr$
$\wedge$ Isoflignederésultat(lr)$\wedge$Testerexpressionexp(expression,nlres)

Post:...

Calcultircta(z,nltir,nlres,déppér,finpér)=z'

CORPSTABLEAU X INT X INT XTIME X TIME

--

>CORPSTABLEAU

But opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne de résultats(nlres), d'un numéro de ligne tir(nltir), d'un intervalle de temps(déppér,finpér), d'un corpstableau(z). Opération intermédiaire permettant de calculer le taux interne de rentabilité à partir d'un numéro de ligne de résultats. La période de temps de la valeur actuelle doit être précisée.

Pré:1≤nltir≤Length(z) ∧ 1≤nlres≤Length(z)
∧ Choix(Données(z[nltir]))=ltir ∧ Choix(Données(z[nlres])))=lres
∧ Isoflignedeformule(ltir)
∧ (Isoflignedeformule(ltir) ⇒Isoflignetir(ltir))
∧ Isoflignederésultat(lrés) ∧ 0 < déppér < finpér

Post: Choix(Données(z'[i])=ltir'
z=z'except
(Numéro(ltir')=nlres
∧ Valtir(ltir')=Calcultirlti(Res(lres),déppér,finpér)
∧ Tempsinit(ltir')=déppér ∧ Tempsfinal(ltir')=finpér)

Variables: lres,ltir,ltir':LIGNE-DONNEES

Calculvancta(z,débpé,finpé,taux,nlcash,nllact) =z'

CORPSTABLEAU X TIME X TIME X REAL X INT X INT
--> CORPSTABLEAU

But opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne de résultats(nlcash), d'un numéro de ligne financière(nlact), d'un intervalle de temps(déppé,finpé), d'un taux(taux), d'un corpstableau(z). Opération intermédiaire permettant de calculer une ligne VAN à partir d'un numéro de ligne de valeurs ou de ligne de résultats. La période de temps, dans laquelle la valeur actuelle doit être inscrite, doit être précisée ainsi que le coût du capital.

Pré: $1 \leq nlact \leq Length(z) \wedge 1 \leq nl \leq Length(z)$
$\wedge$ Choix(Données(z[nlact]))=lact
$\wedge$ Choix(Données(z[nlcash]))=lval
$\wedge$ Isoflignedeformule(lact)
$\wedge$ (Isoflignedeformule(lact) $\Rightarrow$Isoflignefinancière(lact))
$\wedge$ (Isoflignedevaleur(lval) $\vee$ Isoflignederésultat(lval))
$\wedge$ Typeligne(Données(z[nlact]))=" valeur actuelle "
$\wedge$ taux>0 $\wedge$ 0< débpé < finpé
Post:Choix(Données(z'[nlact])=lact' $\wedge$ lv=<lval,Res(lva)>
$\wedge$ Actualiséligneldfi(lv,taux,déppé,finpé)=Fin(lact)
z=z' except
(Lignederéférence(lact')=nlcash
$\wedge$ Fin(lact')=Valeuractuellelignelfi(débpé,périfin,Fin(lact))
$\wedge$Tempsinit(lact')=débpé $\wedge$ Tempsfin(lact')=périfin
$\wedge$Taux(lact')=taux)

Variables: lact,lact',lval: LIGNE-DONNEES
lv: LIGNE DE VALEUR

Calculertableaucta(z)=z'

CORPSTABLEAU -->CORPSTABLEAU

But :opération finale qui permet de calculer le corpstableau(z') à partir d'un corpstableau(z').

Pré: /

Post: ∀i: 2 ≤ i ≤ Length(z[i])
⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒
Isofligneamortissementlinéaire(tl))
⇒(nd=Durée(tl) ∧ nv=Bien(tl)
∧ Calculamortlinéairecta(z,nd,nv,i)=z')]

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isoflignefinancière(tl))
⇒(Typeligne(Données(z[i])="VAN"
⇒(nolcash=Ligneréférence(tl)
∧ taux=Taux(tl)
∧ tempsinit= Tempsinit(tl)
∧ tempsfin=Tempsfin(tl)
∧Calculvancta(z,tempsinit,tempsfin,taux,nolcash,i)
=z'))

⇒(Typeligne(Donnée(z[i])="ligne actualisée"
⇒(nolcash=Ligneréférence(tl)
∧ taux=Taux(tl)
∧ tempsinit= Tempsinit(tl)
∧ tempsfin=Tempsfin(tl)
∧ Calculactualisationcta(nolcash,i,tempsfin,tempsinit,
taux,z)=z'))]

Suite calculertableaucta(z)=z'

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isofligneimpôt(tl))
⇒(nlbenimp=Numéroligneréférence(tl)
∧ taux=Tauximposé(tl)
∧ Calculimpôtcta(i,nlbeimp,taux,z)=z')]

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isoflignebénéficeimposable(tl))
⇒(nlbenimp=Noligneréférence(tl)
∧ Calculbénéficeimposablecta(i,nlbenimp,z)=z')]

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒
Isofligneamortissementdégressif(tl))
⇒(temps=Temps(tl)
∧ noligne=Bien(tl)
∧ Calculamortdégressifcta(z,temps,noligne,i)=z')]

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isoflignetir(tl))
⇒(nlres=Numéro(tl)
∧ tempsinit=Tempsinit(tl)
∧ tempsfin=Tempsfinal(tl)
∧ Calcultircta(z,i,nlres,tempsinit,tempsfin)=z')]

Suite calculertableaucta(z)=z'

⇒ Choix(Données(z[i])=tl
∧ [Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isoflignerésiduelle(tl))
⇒(nlam=Numéroligne1(tl)
∧ nlval=Numéroligne2(tl)
∧ Calcultrésiduellecta(z,nlam,i,nlval)=z')]

⇒ Choix(Données(z[i])=tl
∧ [Isoflignederésultat(tl).
⇒(expression=Expression(tl)
∧ Calculrésultatcta(expression,i,z)=z')]

Variables: nd, nv, i, nolcash, nlbenimp, temps,
noligne, nlres, nlam, nlva :INT
taux :REAL
tempsinit, tempsfin :TIME

Corpstableauaccumulationcta(z,v,temps1,temps2,noligne,taux)=z'

CORPSTABLEAU X REAL X TIME X TIME X INT X REAL
--> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne de résultats(nlres), d'un numéro de ligne tir(nltir), d'un intervalle de temps(déppér,finpér), d'un corpstableau(z). Opération intermédiaire qui permet d'accumuler une valeur(v) pendant une période donnée.

Pré: temps1 < temps2 ∧ z[1]=lt ∧ temps1 > te1
∧ temps2 < te2 ∧ 1< noligne < Length(z)
∧Isoflignedevaleur(Choix(Données(z[noligne])))
∧Choix(Données(lt))[1]=te1
∧ Choix(Données(lt))[Length[lt]]=te2

Post: Choix(Données(z'[noligne])=l'∧ Isoflignedevaleur(l')
∧ Choix(Données(z[noligne]))=l
z=z' except
Calculligneaccumulationlva(l,temps1,temps2,taux,,v)=l'

Variables lt : LIGNE
l',l : LIGNE DE VALEUR
te1,te2:TIME

Créationd'uneligneaccroissementcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne d'accroissements.

Pré: 1≤nl≤Length(z)
∧ créationlignepermise(z,nl)

Post: z=z"except
z"[nl]=l'
∧ Isoflignedevaleur(Choix(Données(l')))
∧ Typeligne(Données(l'))=" valeur accroissement ")

Variables: l' : LIGNE
z": CORPSTABLEAU

Créationd'uneligneaccumulationcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne d'accumulations.

Pré: 1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z"except
z"[nl]=l'
∧ Isoflignedevaleur(Choix(Données(l')))
∧ Typeligne(Données(l'))="valeur accumulée")

Variables: l' : LIGNE
z": CORPSTABLEAU

Créationd'uneligneactualisationcta(z,nl)=z''

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne d'actualisations.

Pré: 1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post:
z=z"except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignefinancière(tl))
∧ Typeligne(Données(l'))="ligne actualisée")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'uneligneamortissementdégressifcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne d'amortissements dégressifs.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z"[nl]=l'
∧ Choix(Données(l')) = tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isofligneamortissementdégressif(tl))
∧ Typeligne(Données(l'))="amortissement dégressif")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'uneligneamortissementlinéairecta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne d'amortissements linéaires.

Pré: 1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z'' except
(z"[nl]=l'
∧ Choix(Données(l')) = tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl) ⇒ Isofligneamortissementlinéaire(tl))
∧ Typeligne(Données(l')) = "amortissement linéaire")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignebénéficeimposablecta(z,nl)=z''

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne de bénéfices imposables.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z'' except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignebénéficeimposable(tl))
∧ Typeligne(Données(l'))=" bénéfice imposable ")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'uneligneimpôtcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne d'impôts.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z" [nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isofligned'impôt(tl))
∧ Typeligne(Données(l'))=" impôt")

Variables: l': LIGNE
tl: LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignerésiduellecta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne résiduelle.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post:z=z" except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignerésiduelle(tl))
∧ Typeligne(Données(l'))=" ligne résiduelle")

Variables: l': LIGNE
tl: LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignerésultatcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne de résultats.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignederésultat(tl)
∧ Typeligne(Données(l'))=" résultat")

Variables: l': LIGNE
tl: LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignetircta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), un corpstableau(z). Opération finale qui crée une ligne de taux interne de rentabilité

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignetir(tl))
∧ Typeligne(Données(l'))=" TIR")

Variables: l': LIGNE
tl: LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignevaleurcta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne de valeurs.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedevaleur(tl)
∧ Typeligne(Données(l'))="valeur ")

Variables: l': LIGNE
tl: LIGNE-DONNEES
z" CORPSTABLEAU

Créationd'unelignevaleurrépétitivecta(z,nl)=z”

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne de valeur répétitive.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post:z=z” except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedevaleur(tl)
∧ Typeligne(Données(l'))="valeur répétitive ")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationd'unelignevancta(z,nl)=z"

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui crée une ligne de VAN.

Pré:1≤nl≤Length(z)
∧ Créationlignepermisecta(z,nl)

Post: z=z" except
(z"[nl]=l'
∧ Choix(Données(l'))=tl
∧ Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignefinancière(tl))
∧ Typeligne(Données(l'))="valeuractuellenette")

Variables: l' : LIGNE
tl : LIGNE-DONNEES
z": CORPSTABLEAU

Créationlignepermisecta(z,nl)

CORPSTABLEAU XINTEGER --> BOOLEAN

But :fonction booléenne composé d'un numéro de ligne (nl), d'un corpstableau(z). Opération finale qui vérifie si le numéro de la ligne n'est pas lié aux autres lignes.

Pré: $1 \leq nl \leq Length(z)$

Post: Choix(Donnée(z[i])=tl
$\wedge \neg \exists i : 1 \leq i \leq Length(z), i \neq nl$
| (Isoflignedeformule(tl)
$\wedge$ (Isoflignedeformule(tl)$\Rightarrow$Isofligneamortissementlinéaire(tl))
$\Rightarrow$(Bien(tl)=nl $\vee$ Durée(tl)=nl))
| (Isoflignedeformule(tl)
$\wedge$ (Isoflignedeformule(tl)$\Rightarrow$Isoflignefinancière(tl))
$\Rightarrow$(Lignederéférence(tl)=nl))
| (Isoflignedeformule(tl)
$\wedge$ (Isoflignedeformule(tl)$\Rightarrow$Isofligneimpôt(tl))
$\Rightarrow$(Numligneréférence(tl)=nl))
| (Isoflignedeformule(tl)
$\wedge$ (Isoflignedeformule(tl)$\Rightarrow$Isoflignebénéficeimposable(tl))
$\Rightarrow$(Noligneréférence(tl)=nl))
| (Isoflignedeformule(tl)
$\wedge$
(Isoflignedeformule(tl)$\Rightarrow$Isofligneamortissementdégressif(tl))
$\Rightarrow$ (Temps(tl)=nl $\vee$ Bien(tl)=nl))
| (Isoflignedeformule(tl)
$\wedge$ (Isoflignedeformule(tl)$\Rightarrow$Isoflignetir(tl))
$\Rightarrow$ (Numéro(tl)=nl))

Suite créationlignepermise(z,nl)

| (Isoflignedeformule(tl)
∧ (Isoflignedeformule(tl)⇒Isoflignerésiduelle(tl))
⇒(Numéroligne1(tl)=nl ∨ Numéroligne2(tl))

Variables tl :LIGNE-DONNEES
i: INTEGER
l1:LIGNE

remarque: la ligne résultat n'est pas testée car on suppose que l'utilisateur est conscient de ce qu'il fait

Créertableaucta()=z'

--> CORPSTABLEAU

But: opération finale qui crée un corpstableau(z') de 3000 lignes à partir d'un corpstableau(z).

Pré: /

Post: Length(z')=3000

Etendrelignecta(z)=z"

CORPSTABLEAU --> CORPSTABLEAU

But:opération qui fournit un nouveau corpstableau(z'') à partir d'un corpstableau(z). Opération intermédiaire et finale qui permet de rajouter une ligne en fin du corpstableau.

Pré: /

Post: Append(z,l)=z' ∧ Length(z')=i
∧Choix(Données(z"[i]))=Miseàzérolv(Choix(Données(z[nl])))
∧Libellé(z"[i])=Miseàblanclibellélig(z'[i])

Variable: l: LIGNE

Initialisationcta()=z"",

--> CORPSTABLEAU

But: opération intermédiaire qui initialise un corpstableau(z''') à vide.

Pré: /

Post : Créertableaucta()=z' ∧ Blanclibellécta(z"')=z''''
Initialiserlignecta(z")=z"' ∧ Initialisertempscta(z')=z"

Variables: z',z",z"',z"": CORPSTABLEAU

Initialiserlignecta(z)=z'

CORPSTABLEAU --> CORPSTABLEAU

But : Opération intermédiaire qui permet de mettre toutes les lignes à zéro à partir d'un corpstableau(z) et qui fournit un nouveau corpstableau(z').

Pré : $\exists$ z

Post : $\forall i : 2 \leq i \leq 3000$
$\Rightarrow$ Choix(Données(z'[i]))=Miseàzérolv(Choix(Données(z[i])))

Variable: i: INTEGER

Initialisertempscta(z)=z'

CORPSTABLEAU --> CORPSTABLEAU

But : Opération intermédiaire qui à partir d'un corpstableau(z) initialise le temps d'un nouveau corpstableau(z') entre 0 et 3000.

Pré : $\exists$ z

Post: z=z' except
Initlte(Choix(Données(z[1])))=Choix(Données(z'[1]))

Variable: l: LIGNE

Insèreraccroissementcta(z,v,accrois,temps1,temps2,noligne)=z'

CORPSTABLEAU X REAL X REAL X TIME X TIME X INT
-->CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un numéro de ligne (noligne), un intervalle de temps(temps1,temps2),un corpstableau(z), une valeur(v). Opération intermédiaire qui permet d'accroître(accrois) une valeur durant une période déterminée.

Pré: temps1 < temps2 ∧ z[1]=lt ∧ temps1> te1 ∧ temps2 < te2
∧ 1< noligne <Length(z)
∧ Isoflignedevaleur(Choix(Données(z'[noligne])))
∧ accrois > 0
∧ Choix(Données(lt))[1]=te1
∧ Choix(Données(lt))[Length[lt]]=te2

Post: Choix(Données(z'[noligne]))=l
z=z' except
(Choix(Données(z'[noligne]))=l' ∧ Isoflignedevaleur(l')
∧ Calculaccroissementlignelva(l,temps1,temps2,accrois,v)=l')

Variables: l,l': LIGNE-DONNEES
lt : LIGNE
te1,te2:TIME

Insérervaleurcta(z,n,année1,année2)=z'

CORPSTABLEAU X INTEGER X TIME X TIME
--> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un intervalle de temps(année1,année2) , d'un corpstableau(z),d'un numéro de ligne(n). Opération intermédiaire permettant d'insérer des valeurs dans une ligne de valeurs.

Pré: Isoflignedevaleur(Choix(Données(z[n])))
∧ z[1]=lt ∧ te1<année1<année2<te2
∧Choix(Données(lt))[1]=te1
∧ Choix(Données(lt))[Length[lt]]=te2

Post: z=z' except
(Insérervallva(année1,année2,Choix(Données(z[n])))
=Choix(Données(z'[n]))
∧ Isoflignedevaleur(Choix(Données(z' [n]))))

Variables: lt : LIGNE
te1,te2 : TIME

Insérervaleurrepcta(v,temps1,temps2,n,z)=z'

REAL X TIME XTIME X INT X CORPSTABLEAU
--> CORPSTABLEAU

But: opération intermédiaire qui permet de répéter une valeur(v) dans un intervalle(temps1, temps2) d'un ligne dont le numéro de la ligne(n) du corpstableau(z) est défini. Cette opération fournit un nouveau corpstableau(z').

Pré:1 < n < Length(z) ∧Isoflignedevaleur(Choix(Données(z[n])))
∧ te1< temps1< temps2<te2 ∧ z[1]=lt
∧Typeligne(Choix(Données(z[n])))='valeur répétitive'
∧Choix(Données(lt))[1]=te1
∧ Choix(Données(lt))[Length[lt]]=te2

Post:Choix(Données(z[n]))=lv
z=z'
except
(Choix(Données(z'[n]))=Insérervaleurrépétitivelva(v,temps1,temps2,lv)
∧Isoflignedevaleur(Choix(Données(z'[n]))))

Variable: lv: LIGNE DE VALEUR
te1,te2: TIME
lt: LIGNE

Limitertaillecta(z,x)=z'

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un corpstableau(z), d'un numéro de ligne(x). Opération finale qui limite la taille du tableau à partir du numéro de ligne donné.

Sign : x: numéro de la ligne à partir duquel on limite

Pré :$\forall$ i : x $\leq$ i $\leq$ Length (z) $\Rightarrow$ Isoflignevaleur(Données(z[i])

Post : Length(z') = x $\forall$ i: 1 $\leq$ i $\leq$ Length(z') $\Rightarrow$ z[i]' = z[i]

Variable: i: INTEGER

Miseàzérodelalignescta(z,nl)=z'

CORPSTABLEAU X INT --> CORPSTABLEAU

But: opération intermédiaire qui permet de mettre des zéros dans une ligne d'un corpstableau(z') à partir d'un corpstableau(z), numéro de ligne(nl).

Pré: 1 < nl < Length(z[nl]) $\wedge$ Isoflignedevaleurs(Choix(Données(z[nl]))

Post: z=z' except
(Choix(Donnée(z'[nl]))=Miseàzérolv(Choix(Données(z[nl])))
$\wedge$ Isoflignedevaleur(Choix(Données(z'[n]))))

Remodifiercorpscta(z,x)=z'

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un corpstableau(z), d'un numéro de ligne(x). Opération finale qui insère une ligne dans un tableau. L'insertion implique des modifications dans les paramètres associés à certaines lignes.

Pré : x < Length(z) x > 1

Post :Insert(z,x)=z'
∧ z=z' except
Miseàzérolva(Choix(Données(z[x])))=Choix(Données(z' [x]))
∧ ∀i : x ≤ i ≤ length(z') (Choix(Données(z[i]))=tl *type ligne*
∧ Choix(Données(z' [i]))=tl)'
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isofligneamortissementlinéaire(tl))
⇒(Bien(tl) Durée(tl) > x ⇒(Bien(tl')= Bien(tl)+1
∧ Durée(tl')=Durée(tl)+1))]
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignefinancière(tl))
⇒(Ligneréférence(tl) > x
⇒ Ligneréférence(tl')= Ligneréférence(tl)+1)]

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isofligneimpôt(tl))
⇒(Numlignereréférence(tl) > x
⇒ Numligneréférence(tl')= Numligneréférence(tl)+1)]
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignebénéficeimposable(tl))
⇒(Noligneréference(tl) > x
⇒ Noligneréférence(tl')= Noligneréférence(tl)+1)]

Suite Remodifiercorpscta(z,x)=z'

⇒ [Isoflignedeformule (tl)
∧(Isoflignedeformule(tl)
⇒Isofligneamortissementdégressif(tl))
⇒(Bien(tl) > x ⇒ Bien(tl')= Bien(tl)+1]

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignetir(tl))
⇒(Numéro(tl) > x ⇒ Numéro(tl')= Numéro(tl)+1]

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignerésiduel(tl))
⇒(Numérolignevalres(tl) Numéroligneamo(tl) > x
⇒ (Numérolignevalres(tl')= Numérovalres(tl)+1
∧ Numéroligneamo(tl') = Numéroligneamo(tl)+1)]

⇒ [Isoflignerésultat (tl)
⇒ Modifierpositivementexp(Expression(tl),x)]

Variable: tl: LIGNE-DONNEES

Supprimercorpscta(z,x)=z'

CORPSTABLEAU X INTEGER --> CORPSTABLEAU

But: opération qui fournit un nouveau corpstableau(z') à partir d'un corpstableau(z), d'un numéro de ligne(x). Opération finale qui supprime une ligne dans un tableau. La suppression implique des modifications dans les paramètres associés à certaines lignes.

Pré : x < Length(z) x > 1

Post :Remove(z,x)=z' ∧
z=z' except
∧ Miseàzérolva(Choix(Données(z[x])))=Choix(Données(z' [x]))
∀ i : x ≤ i ≤ length(z') : (Choix(Données(z[i]))=tl *type ligne*
∧ Choix(Données(z'[i]))=tl')
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isofligneamortissementlinéaire(tl))
⇒(Bien(tl) Durée(tl) < x ⇒(Bien(tl')= Bien(tl) - 1
∧ Durée(tl')=Durée(tl) - 1))]
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignefinancière(tl))
⇒(Ligneréférence(tl) < x
⇒ Ligneréférence(tl)= Ligneréférence(tl) -1)]
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isofligneimpôt(tl))
⇒(Numligneréférence(tl) < x
⇒ Numligneréférence(tl')= Numligneréférence(tl)-1)]
⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignebénéficeimposable(tl))
⇒(Noligneréférence(tl) < x
⇒ Noligneréférence(tl')= Noligneréférence(tl) - 1)]

Suite supprimercorpscta(z,x)=z'

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isofligneamortissementdégrssif(tl))
⇒(Bien(tl) < x ⇒ Bien(tl')= Bien(tl) - 1]

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignetir(tl))
⇒(Numéro(tl) < x ⇒ Numéro(tl')= Numéro(tl) -1]

⇒ [Isoflignedeformule (tl)
∧ (Isoflignedeformule(tl) ⇒Isoflignerésiduel(tl))
⇒(Numérolignevalres(tl) Numéroligneamo(tl) < x
⇒ (Numérolignevalres(tl')= Numérovalres(tl) -1
∧ Numéroligneamo(tl') = Numéroligneamo(tl) -1)]

⇒ [Isoflignederésultat (tl)
⇒Modifiernégativementexp(Expression(tl),x)]

Variable: tl:LIGNE-DONNEES

Vérificationtableauzérocta(z)

CORPSTABLEAU --> BOOLEAN

But: opération intermédiaire permettant de vérifier si toutes les lignes du corpstableau(z) sont remplies de zéros.

Pré:/

Post : ∀ t: 2 ≤ t ≤ Length(z) ⇒ Isoflignedevaleur(Choix(Données(z[t]))
∧ Vérifierzérolva(Choix(Données(z[t]))

Variable: t:INTEGER

LIGNE

is CP [Libellé: STRING, Données : LIGNE-DONNEES]

But: cluster qui permet d'effectuer des opérations finales sur les différents types de lignes

Exportation: tout

Ecrirelibellélig(nom,l)

STRING X LIGNE -->BOOLEAN

But:opération finale qui permet d'écrire un libellé(nom) dans une ligne(l).

Pré: Isnotlignedetemps(Choix(Données(l))

Post: Libellé(l)=nom

Miseàblanclibellélig(l)=l'

LIGNE-->LIGNE

But: opération finale qui met les libellés de la ligne(l') à blanc à partir de la ligne(l).

Pré:

Post: Libellé(l')= " "

LIGNE-DONNEES
is CP [Choix: UNION [LIGNE DE VALEUR,
LIGNE DE TEMPS
LIGNE DE RESULTAT
LIGNE DE FORMULE]
Typeligne : STRING]

But: cluster qui permet des opérations sur les différents types de lignes.

Invariants:
1) Typeligne(ld)=("valeuraccroissement " ∨"accumulation")
⇒∀ i,j 1≤ i,j ≤ Length(Choix(ld)) ∧ Isoflignedevaleur(Choix(ld))
⇒
(∀ k:1<k<i ⇒Chiffre(Choix(ld)[k])=0
∧ ∀ k:j<k<Length(Choix[ld]) ⇒ Chiffre(Choix(ld)[k])=0
∧ ∀ k:i<k<j ⇒ Chiffre(Choix(ld)[k]) > Chiffre(Choix(ld)[k-1]))
2) Typeligne(ld)=("valeur répétitive")
⇒∀ i,j 1≤ i,j ≤ Length(Choix(ld)) ∧ Isoflignedevaleur(Choix(ld))
⇒
(∀ k:1<k<i ⇒Chiffre(Choix(ld)[k])=0
∧ ∀ k:j<k<Length(Choix[ld]) ⇒ Chiffre(Choix(ld)[k])=0
∧ ∀ k:i<k<j ⇒ Chiffre(Choix(ld)[k]) = Chiffre(Choix(ld)[k-1]))

Variables: ld: LIGNE-DONNEES
i,j,k: INTEGER

Exportation: tout

LIGNE DE VALEUR
is SEQ [CP[Chiffre : REAL, Time :TIME]]
is OBJECT [Chiffre*: REAL]

But : Cluster qui est une suite de valeurs associées au temps sur lequel une série d'opérations peuvent être effectuées.

Exportation: tout

Calcullacroissementlignelva(l,temps1,temps2,accrois,v)=l'

LIGNE DE VALEUR X TIME X TIME X REAL X REAL
--> LIGNE DE VALEUR

But:opération finale qui accroît dans une ligne(l') une valeur(v) pendant une période donnée(temps1,temps2) à un facteur d'accroissement donné(accrois) dans une ligne donnée(l).

pré:: temps1 < temps2 ∧ temps1> Time(l[1])
∧ temps2<Time(l[Length(l)]
∧ Chiffre(l at temps1)= v ∧ Isoflignedevaleur(l) ∧ accrois>0

Post:∀t : temps1< t ≤temps2
⇒(Chiffre(l' at t) = Chiffre(l at (t-1))* accrois

Variable: t: TIME

Calculligneaccumulationlva(l,temps1,temps2,taux,v)=l'

LIGNE DE VALEUR X TIME X TIME X REAL X REAL
--> LIGNE DE VALEUR

But: Opération finale qui accumule une valeur(v) pendant une période donnée(temps1,temps2) à un taux donné(taux) d'une ligne donnée(l). Cette opération fournit une nouvelle ligne(l').

pré:: temps1 < temps2 ∧ temps1> Time(l[1])
∧ temps2<Time(l[Length[l]])
∧ Isoflignedevaleur(l)∧ taux > 0

Post:∀t : temps1< t< temps2 ⇒(Chiffre(l' at t) = v* (intérêt)$^{temps2-t}$
∧ intérêt = (taux: / 100) + 1

Variables: intérêt: REAL
t: TIME

Insérervaleurrépétitivelva(v,temps1,temps2,lv)=lv'

REAL X TIME X TIME X LIGNE DE VALEUR
--> LIGNE DE VALEUR

But: opération finale qui insère la même valeur(v) à travers l'intervalle de temps(temps1,temps2) souhaité de la ligne de valeurs(l) et qui fournit une autre ligne de valeurs(lv')..

Pré : temps1 < temps2 ∧ temps1>Time(lv[1])
∧ temps2<Time(lv[Length[lv]])
∧ Isoflignedevaleur(lv)]

Post: ∀ t: temps1< t< temps2 ∧ ∀ t:Chiffre(lv' at t) = v

Variable: t: TIME

Insérervallva(lv,débuttemps,fintemps)=lv'

LIGNE DE VALEUR X TIME X TIME -->LIGNE DE VALEUR

But: opération finale qui permet d'inscrire des valeurs dans une ligne de valeurs(lv') dans un intervalle de temps(temps1,temps2) défini. Cette opération se fait à partir d'une ligne de valeurs(lv).

Pré: débuttemps < fintemps ∧ débuttemps> Time(lv[1])
∧ fintemps<Time(lv[Length[lv]])
∧ Isoflignedevaleur(lv)

Post: ∀t : débuttemps<t<fintemps : Chiffre(lv' at t)=i

Variables: t: TIME
i: REAL

Miseàzérolva(lv)=lv'

LIGNE DE VALEUR --> LIGNE DE VALEUR

But: opération finale qui met la ligne de valeurs(lv') à zéro à partir d'une ligne de valeurs(lv).

Pré: Isoflignedevaleur(lv)

Post: ∀t:Time(lv[1]) < t < Time(lv[Length[lv]]) : Chiffre(lv' at t) =0

Variable: t:TIME

Vérifierzérolva(l)

LIGNE DE VALEUR --> BOOLEAN

But: opération finale qui s'assure si la ligne(l) ne contient que des zéros.

Pré: Isoflignedevaleur(l)

Post: $\forall$t: Time(l[1]) $\leq$ t$\leq$ Time(l[Length[l]]) $\wedge$ $\forall$ t:=(Chiffre(l at t), 0)

Variable: t :TIME

LIGNE DE TEMPS
is SEQ [TIME]

But : cluster qui est une suite d'années et sur lequel une série d'opérations peuvent être effectuées.

Invariants: $\forall$ lt 1 < i $\leq$ Length(lt) $\Rightarrow$(lt[i] >lt[i-1]
$\wedge$ lt[i] = lt[i-1] + 1)
Variables: lt: LIGNE DE TEMPS
i: INTEGER

Exportation: tout

Initlte(x)=x'

LIGNE DE TEMPS --> LIGNE DE TEMPS

But: opération finale qui écrit à partir d'une ligne de temps(x) une ligne de temps(x') allant de zéro à 3000.

Pré:/

Post:$\forall$ i : 1< i <3000 $\Rightarrow$ x ' [i]=x[i-1] +1

Variable: i:TIME

Modificationintervalledetempslte(x,i,j)=x'

LIGNE DE TEMPS X TIME X TIME--> LIGNE DE TEMPS

But: opération finale qui modifie l' intervalle de temps(i,j) d'une ligne de temps(x) et fournit une autre ligne de temps(x')

Pré:Isoflignedetemps(x) $\wedge$ $0 < i < j$

Post: $x'[1] = i \wedge Length(x') = j-i \wedge x[Lengtht(x')] = j$
$\wedge \forall t:: 2 < t < Length(x) \Rightarrow x'[t] = x[t-1] +1$

Variable: t:TIME

Testertempslte(x,i,a)

LIGNE DE TEMPS X TIME X TIME -->BOOLEAN

But:opération finale qui permet de tester si les bornes de temps(i,a) données sont bien comprises dans l'intervalle de la ligne de temps(x).

Pré: $i < a \wedge i > 0$

Post: $i \geq First(x) \wedge a \leq Last(x)$

EXPRESSSION
is UNION[SOMME, DIVISION,PRODUIT]

But: cluster qui traite les expressions.

Exportation: tout

Modifierpositivementexpressionexp(exp,x)=exp'

EXPRESSION X INTEGER--> EXPRESSION

But: opération finale qui modifie le numéro(x) de la ligne de l'expression(exp) de plus un si le numéro est supérieur au numéro de la ligne de référence. Cette opération fournit une nouvelle expression(exp').

Pré: x>0

Post: Isofdivision(exp)
⇒ (Diviseur(exp)>x ⇒ Diviseur(exp')=Diviseur(exp)+1
∧ Dividente(exp) >x) ⇒
Dividente(exp')=Dividente(exp)+1)
Isofsomme(exp)
⇒∀i 1< i < Length(exp) :
(Terme(exp[i])>nlres)⇒Terme(exp''[i])=Terme[exp] + 1)
Isofproduit(expression)
⇒∀i 1< i < Length(exp):
(Terme(exp[i])>nlres)⇒Terme(exp'[i]=Terme[exp] + 1)

Variable: i :INTEGER

Modifiernégativementexpressionexp(exp,x)=exp'

EXPRESSION X INTEGER --> EXPRESSION

But: opération finale qui modifie le numéro(x) de la ligne de l'expression(exp) de moins un si le numéro est supérieur au numéro de la ligne de référence. Cette opération fournit une nouvelle expression(exp').

Pré: x>0

Post: Isofdivision(exp)
⇒ (Diviseur(exp) > x ⇒ Diviseur(exp')=Diviseur(exp)-1
∧ (Dividende(exp) > x)
⇒ Dividende(exp')=Dividende(exp)-1)
Isofsomme(exp)
⇒∀i 1< i < Length(exp) :
(Terme(exp[i])>nlres)⇒Terme(exp' [i])=Terme(exp[i]) - 1)
Isofproduit(expression)
⇒∀i 1< i < Length(exp):
(Terme(exp[i])>nlres)⇒Terme(exp' [i]=Terme[exp] - 1)

Variable: i: INTEGER

Testerexpressionexp(expression,nlres)

EXPRESSION X INTEGER --> BOOLEAN

But: opération finale qui teste si l'expression (expression) est correcte à partir du numéro de la ligne résultat(nlres).

Pré: nlres>0

Post: Isofdivision(expression)
⇒ (Signe(expression)= / ∧ Diviseur(expression)<nlres
∧ 2<Dividende(expression) <nlres)
Isofsomme(expression)
⇒∀i 1< i < Length(expression) :
(Signe(expression[i])=<+,->
∧ 1<Terme(expression[i])<nlres)
Isofproduit(expression)
⇒∀i 1< i < Length(expression):
(Signe(expression[i])=* ∧ 1<Terme(expression[i])<nlres)

Variable: i: INTEGER

LIGNE DE FORMULE
is UNION [LIGNE FINANCIERE,
LIGNE AMORTISSEMENT
LINEAIRE
LIGNE IMPOT
LIGNE BENEFICE IMPOSABLE
LIGNE AMORTISSEMENT
DEGRESSIF
LIGNE TIR
LIGNE RESIDUELLE]

But : cluster qui permet d'offrir un ensemble de clusters financiers sur lesquels des opérations financières sont permises.

LIGNE FINANCIERE
is CP [Tempsinit :INTEGER, Tempsfin: INTEGER,
Taux : INTEGER, Fin : LIGNE DE VALEUR,
Lignederéférence : INTEGER]

But: cluster qui permet d'effectuer des opérations sur des lignes VAN et des lignes actualisées.

Invariants: $\forall$ lf : ti = Tempsinit(lf) $\wedge$ tf=Tempsfin(lf)
$\forall$ i : (i < ti $\Rightarrow$ Chiffre(Fin(lf[i]))=0
$\wedge$ i > tf $\Rightarrow$ Chiffre(Fin(lf[i]))=0)

Variables: lf: LIGNE FINANCIERE
i: INTEGER
Exportation: tout

Actualiséunevaleurlfi(x,i,intérêt)=y

REAL X INTEGER X REAL --> REAL

But: opération finale qui calcule une valeur actuelle(y) à partir d'un intérêt(intérêt), d'une valeur(x), d'un exposant(i).

Pré: /

Post: $y = \frac{x}{(\text{interêt})^i}$

Actualiserlignelfi(l,taux,pérdép,pérfin)=l'

LIGNE DE VALEUR X REAL X TIME X TIME
--> LIGNE DE VALEUR

But: opération finale qui calcule la valeur actuelle d'une ligne(l') à partir d'un taux(taux), d'un intervalle de temps(pérdép,pérfin), d'une ligne de valeurs(l).

Pré: pérdép< pérfin ∧ pérdép> Time(l[1])
∧ pérfin<Time(l[Length[l]]) ∧ taux>0
∧Isoflignedevaleur(l)

Post: intérêt = $\frac{taux}{100} + 1$
∧∀ i : pérdép< i < pérfin
⇒ (Actualiser une valeur (Chiffre(l at i),intervalle,intérêt)=y
∧ Chiffre(l' at i)=y ∧ intervalle = i-pérdép

Variable: i:TIME

Valeuractuellelignelfi(pérdép,pérfin,lact)=lact'

TIME X TIME X LIGNE DE VALEUR --> LIGNE DE VALEUR

But: opération finale qui calcule la valeur actuelle de la ligne(lact') à partir d'un intervalle de temps défini(pérdép,pérfin), d'une ligne de valeurs(lact') .

Pré: pérdép< pérfin ∧ pérdép> Time(lact[1])
∧ pérfin<Time(lact[Length[lact]]) ∧ Isoflignedevaleur(lact)

Post:∀ i : pérdép< i ≤ pérfin
⇒ Chiffre(lact' at i)= Chiffre(lact at i) + Chiffre(lact' at (i-1))
i=pérdép ⇒ Chiffre(lact' at i)= Chiffre(lact at i)

Variable: i:TIME

LIGNE AMORTISSEMENT DEGRESSIF
is CP [Deg : LIGNE DE VALEUR, Temps : INTEGER,
Bien : INTEGER]

But: cluster qui permet d'effectuer des opérations sur une ligne d'amortissement dégressif. Cette ligne est composée d'une ligne de valeur qui va recevoir le résultat du calcul, une période d'amortissement unique et un numéro de référence à la ligne des biens à amortir.

Exportation: Calculamortdégressiflad

Calculamortdégressiflad(lvl,lvl2,pér)=lad

LIGNE DE VALEUR X LIGNE DE VALEUR X INTEGER
-->LIGNE DE VALEUR

But: opération intermédiaire qui permet de calculer l'amortissement dégressif à partir de deux lignes de valeurs(lvl,lv2) et d'une période d'amortissement donné(pér) et qui fournit une nouvelle ligne de valeurs(lad).

Pré : pér > 0 ∧ Isoflignedevaleurs(lvl) ∧ Isoflignedevaleurs(lvl2)

Post: ∀t : Time(lv1[1]) ≤ t ≤ Time(lv1[Length[lvl]])
⇒ ((Chiffre(lvl at t) = x ∧ x ≠ 0)
⇒Calculamortissementdégressiflad(t,x,lvl2,pér)=lad)

Variables: t :TIME
x: REAL

Calculamortissementdégressiflad(t,x,l,pér)=l'

TIME X REAL X LIGNEDE VALEUR X INT
-->LIGNE DE VALEUR

But: cette opération finale effectue un calcul d'amortissement dégressif(l'). Le taux d'amortissement utilisé est égal au double du taux linéaire. Il est appliqué au solde non encore amorti. Dès que l'amortissement ainsi calculé devient inférieur à l'amortissement linéaire du montant initial, on applique à nouveau l'amortissement linéaire. Opération produisant une ligne de valeur(l') à partir d'une année considérée(t), d'une valeur(x), d'une ligne de valeurs(l), d'une péride d'amortissement(pér).

Pré: t>Time(l[1]) ∧ pér>0 ∧ Isoflignedevaleur(l)

Post: ∀i : t ≤ i ≤ Time(l[Length[l]])
⇒ (x [i-1]> amdeg ⇒ (x[i]= x[i-1] - amdeg
∧ Chiffre(l' at i)= Chiffre (l at i) + amdeg
⇒ (x [i-1]< amdeg ⇒ (x[i]= x[i-1] - amlin
∧ Chiffre(l' at i)= Chiffre (l at i) + amlin
⇒ (x [i-1]≤ 0 ⇒ Chiffre(l' at i)= Chiffre (l at i))
∧ amlin $=\frac{x[t]}{pér}$ ∧ amdeg$=\frac{(x[t]* 2)}{pér}$

Variables: t: année ou commence l'amortissement: TIME
pér: nombre d'années à amortir:INT
x: valeur à amortir : real
i : indice : TIME
amlin: valeur de l'amortissement linéaire : REAL
amdeg: valeur de l'amortissement dégressif : REAL

LIGNE RESIDUELLE

is CP[Resi:: LIGNE DE VALEUR,
Numérolignevalres: INTEGER
Numéroligneamo: INTEGER]

But: Cluster qui permet d'effectuer une série d'opérations sur une ligne résiduelle. Cette ligne est composée d'une ligne de valeur, d'un numéro de ligne qui fait référence à la ligne de valeurs ou de résultats et un numéro de ligne qui fait référence à une ligne d'amortissement.

Exportation: tout

Calculrésiduellelre(l1,l2)=l3

LIGNE DE VALEUR X LIGNE DE VALEUR -->LIGNE DE VALEUR

But opération finale qui calcule une ligne de valeurs(l3) de la ligne résiduelle à partir de deux lignes de valeurs(l1, l2).

Pré :Isoflignedevaleur(l1) $\wedge$ Isoflignedevaleur(l2)

Post : Time (l1[Lenght[l1]])=j

$$\wedge \left(y= \sum_{i=lt[1]}^{j} \text{Chiffre(l2 at i)} - \text{Chiffre(l1 at i)}\right.$$

$\wedge$ Chiffre(l3 at i)=0)

$\wedge$ Chiffre(l3 at lt[Length(lt)])=y

Variables: y: REAL
i: TIME

LIGNE IMPOT
is CP [Tauximposé:REAL, Imp : LIGNE DE VALEUR
Numligneréférence : INTEGER]

But : cluster qui permet d'effectuer une série d'opérations Ce cluster est composé d'un taux imposé par la loi, une ligne de valeurs et le numéro de la ligne de référence du bénéfice imposable.

Exportation: tout

Calculimpôtlim(taux,lv)=l

REAL X LIGNE DE VALEUR --> LIGNE DE VALEUR

But opération finale qui calcule une ligne de valeurs(l) de la ligne d'impôts à partir d'un taux(taux), d'une ligne de valeurs(lv).

Pré : taux > 0 $\wedge$ Isoflignedevaleur(lv)

Post: $\forall$t : Time(lv[1]) $\leq$ t $\leq$ Time(lv[Length[lv]])
$\Rightarrow$ (Chiffre(l at t))= Chiffre(lv at t) * taux

Variable: t: TIME

LIGNE AMORTISSEMENT LINEAIRE
is CP [Bien: INTEGER, Durée: INTEGER
Am : LIGNE DEVALEUR]

But : cluster qui permet d'effectuer un ensemble d'opérations d'amortissement. Ce cluster possède un numéro de référence à la ligne de biens, un numéro de référence à la ligne correspondant à la durée d'amortissement et une ligne de valeur qui conservent les valeurs.

Exportation: Calculamortlinéairelal

Amortissementlal(lvv,ld,i,j)=v.

LIGNE DE VALEUR X LIGNE DE VALEUR X TIME X TIME
-->REAL

But: opération qui calcule la valeur(v) d'un amortissement linéaire à une position déterminée(i,j) en fonction de deux lignes de valeurs(lvv,ld).

Pré: $i \leq j \wedge$ Isoflignedevaleur(lvv) $\wedge$ Isoflignedevaleur(ld)

Post: $j > i +$ Chiffre(ld(i)) $-1 \Rightarrow v = 0$

$j \leq i +$ Chiffre(ld at i) $-1 \Rightarrow v = \frac{\text{Chiffre(lb at i)}}{\text{Chiffre(ld at i)}}$

Variables: i : période d'achat du bien
j : période d'amortissement du bien

Amorttotalelal(lvv,ld,k,j)=v

LIGNE DE VALEUR X LIGNE DE VALEUR X TIME X TIME
-->REAL

But: Opération intermédiaire qui calcule la valeur(v) d'un amortissement linéaire d' un segment(k) de la ligne de valeurs(lvv) à partir d'une autre ligne de valeurs(ld).

Pré: Isoflignedevaleur(lvv) $\wedge$ Isoflignedevaleur(ld) $\wedge$ k>0

Post: $\sum_{i=\text{Time(ld[1])}}^{k}$ Amortissementlal(lvv,ld,i,k)=v

Variable: i,:TIME

Calculamortlinéairelal(lvv,ld)=lal

LIGNE DE VALEUR X LIGNE DE VALEUR--> LIGNE DE VALEUR

But : opération intermédiaire qui permet de calculer l'amortissement linéaire. sur toute la ligne de valeur(lal) de l'amortissement linéaire à partir de deux lignes de valeurs(lvv,ld).

Pré: Isoflignedevaleur(lvv) $\wedge$ Isoflignedevaleur(ld)

Post : $\forall k$: Time(ld[1]) $\leq k \leq$ Time(ld[Length[ld]])
Chiffre(lal at k)= Amorttotal(lvv,ld,k)

Variable: k:TIME

LIGNE TIR
is CP[Valtir: LIGNE DE VALEUR, Numéro : INTEGER]

But: Cluster qui est composé d'une ligne de valeurs et d'un numéro de ligne de référence.

Exportation: Calcultirlti

Calcultirlti(lv,dépér,finpér)=ltir

LIGNE DE VALEUR X TIME X TIME --> LIGNE DE VALEUR

But: opération finale qui permet de calculer le taux interne de rentabilité d'une ligne de valeurs(ltir) de la ligne TIR à partir d'une ligne de valeurs(lv), d'un intervalle de temps(dépér,finpér).

Pré: finpér>dépér ∧ dépér > Time(lv[1])
∧ finpér<Time(lv[Length[lv]])
∧Isoflignedevaleur(lv)

Post: Vanltir(lv,dépér,finpér,r)=0
∧ Chiffre(ltir at Time(lv[1]))=r
∀ i :Time(lv[2]) ≤ i ≤ Time(lv[Length[lv]]): Chiffre(ltir at i)=0

Variable: r: REAL

Vanltir(lv,dépér,finpér,r)=v

LIGNE DE VALEUR X TIME X TIME X REAL --> REAL

But opération finale qui calcule la VAN(v) à partir d'une ligne de valeurs(lv), d'un intervalle de temps(dépér,finpér).

Pré:. finpér>dépér ∧ dépér >Time(lv[1]) ∧ finpér<Time(lv[Length[lv]])
∧Isoflignedevaleur(lv)

Post: $\sum_{i=dépér}^{finpér} \frac{Chiffre(lv\ at\ i)}{(i+r)^{i-dépér+1}} = 0$

LIGNE BENEFICE IMPOSABLE
is CP [Ben : LIGNE DE VALEUR,
Nolignerèférence : INTEGER]

But: Cluster composé d'une ligne de valeurs et d'un numéro de ligne qui fait référence à une ligne de résultats.

Exportation: tout

Calculbénéficeimposablelbi(lv)=l

LIGNE DE VALEUR --> LIGNE DE VALEUR

But: fournir une ligne de valeur(l) du bénéfice imposable correcte à partir d'une ligne de valeurs(lv). Le calcul est effectué sur base du résultat obtenu sur la colonne précédente. Un solde positif obtenu après report éventuel des pertes antérieures est reporté sur sur la période suivante. Le bénéfice imposable étant fixé à zéro.

Pré: Isoflignedevaleur(lv) $\wedge$ j=lt[1]

Post: Chiffre(lv[1])$\geq$ l$\Rightarrow$Chiffre(l' [0])=0
Chiffre(lv[1])$\leq$ l$\Rightarrow$Chiffre(l' [0])=Chiffre(lv[1])
Chiffre(lv[1])$\geq$ l$\Rightarrow$Chiffre(l [0])=Chiffre(lv[1])
Chiffre(lv[1])$\geq$ l$\Rightarrow$Chiffre(l [0])=0

$\forall$ 1 < i $\leq$ Length(lv)$\Rightarrow$
Chiffre(l[i-1])$\geq$ 0$\Rightarrow$Chiffre(l' [i])=0
Chiffre(l[i-1])= 0$\Rightarrow$Chiffre(l' [i])= Chiffre(l'[i-1]) + Chiffre(lv[i-1])
Chiffre(lv[i]) + Chiffre(l' [i]) < 0 $\Rightarrow$ Chiffre(l [i])=0
Chiffre(lv[i]) + Chiffre(l' [i]) $\geq$ 0
$\Rightarrow$ Chiffre(l[i])=Chiffre(l' [i])+Chiffre(lv[i])

Variables: l' : LIGNE DE VALEUR
i: TIME

3. 2. 3Glossaire des spécifications

A) Glossaire par cluster en ordre hiérarchique

TABLEAU 45
Accumulertab(v,taux,temps1,temps2,noligne,t) = t' 45
Accroissementvaleurtab(v,accrois,temps1,temps2,noligne,t)= t' 46
Actualisationtab(nlcash,nlact,débpér,finpér,taux,t)=t' 46
Amortissementdégressiftab(pér,nlamdeg,nl,t)=t' 47
Amortissementlinéairetab(nd,nv,nal,t)=t' 48
Avoirlignevaleurtab(nl,t)=t' 48
Bénéficeimposabletab(nlres,nlbenimp,t)=t' 49
Créertableautab (nom) = t 49
Etendreunelignedansuntableautab(t)=t' 50
Impôttab(nlbenimp,nlimp,taux,t)=t' 50
Inscrirelibellétab(t, nom,nl) = t' 51
Insérerlignedansuntableautab(t,x) = t' 51
Limitertailletableautab (t,x)=t' 52
Modifierlesdeuxtempstab(t,i,j)=t' 52
Modifiervaleurtab(t,n,année1,année2)= t' 53
Réinitialisationdutableautab (t) = t' 53
Résiduelletab(nlam,nlbien,nlresi,t)=t' 54
Résultattab(expression,nlres,t)=t' 54
Saisirvaleurtab(t,n,année1,année2) = t' 55
Supprimerlignedutableautab(x,t) = t' 55
Supprimerunlibellétab(t,n) = t' 56
Supprimervaleurtab(t,n,année1,année2)= t' 56
Tirtab(nltir,nlres,t,débpér,finpér)=t' 57
Valeurrépétitivetab(t,v,temps1,temps2,n) = t' 58
Vantab(t,débpé,finpé,taux,nolignecashflow,nolvalactunette)= t' 59
CORPSTABLEAU 60
Blanclibellécta(z)=z' 61
Calculactualisationcta(nlcash,nlact,débpér,finpér,taux,z)=z' 62

Calculamortdégressifcta(z,pér,nl,nlamdeg)=z' 63
Calcullamortineairecta(z,nd,nv,nal) =z' 64
Calculbénéficeimposablecta(nlres,nlbenimp,z)=z' 65
Calculimpôtcta(nlimp,nlbenimp,taux,z)=z' 66
Calculrésiduellecta(z,nlam,nlresi,nlbien)=z' 67
Calculrésultatcta(expression,nlres,z)=z' 68
Calcultircta(z,nltir,nlres,déppér,finpér)=z' 69
Calculvancta(z,débpé,finpé,taux,nlcash,nllact) =z' 70
Calculertableaucta(z)=z' 71
Corpstableauaccumulationcta(z,v,temps1,temps2,noligne,tau
x)=z' 74
Créationd'uneligneaccroissementcta(z,nl)=z" 75
Créationd'uneligneaccumulationcta(z,nl)=z" 75
Créationd'uneligneactualisationcta(z,nl)=z'' 76
Créationd'uneligneamortissementdégressifcta(z,nl)=z" 77
Créationd'uneligneamortissementlinéairecta(z,nl)=z" 78
Créationd'unelignebénéficeimposablecta(z,nl)=z" 79
Créationd'uneligneimpôtcta(z,nl)=z" 80
Créationd'unelignerésiduellecta(z,nl)=z" 81
Créationd'unelignerésultatcta(z,nl)=z" 82
Créationd'unelignetircta(z,nl)=z" 83
Créationd'unelignevaleurcta(z,nl)=z" 84
Créationd'unelignevaleurrépétitivecta(z,nl)=z" 85
Créationd'unelignevancta(z,nl)=z" 86
Créationlignepermisecta(z,nl) 87
Créertableaucta()=z' 88
Etendrelignecta(z)=z" 89
Initialisationcta()=z'''' 89
Initialiserlignecta(z)=z' 90
Initialisertempscta(z)=z' 90
Insérerаccroissementcta(z,v,accrois,temps1,temps2,noligne)=
z' 91
Insérervaleurcta(z,n,année1,année2)=z' 92
Insérervaleurrepcta(v,temps1,temps2,n,z)=z' 93
Limitertaillecta(z,x)=z' 94
Miseàzérodelalignescta(z,nl)=z' 94
Remodifiercorpscta(z,x)=z' 95
Supprimercorpscta(z,x)=z' 97

Vérificationtableauzérocta(z)..... 98
LIGNE..... 99
Ecrirelibellélig(nom,l)..... 99
Miseàblanclibellélig(l)=l' 99
LIGNE-DONNEES..... 100
LIGNE DE VALEUR..... 100
Calcullacroissementlignelva(l,temps1,temps2,accrois,v)=l' 101
Calculligneaccumulationlva(l,temps1,temps2,taux,v)=l'..... 102
Insérervaleurrépétitivelva(v,temps1,temps2,lv)=lv' 102
Insérervallva(lv,débuttemps,fintemps)=lv' 103
Miseàzérolva(lv)=lv' 103
Vérifierzérolva(l) 104
LIGNE DE TEMPS 104
Initlte(x)=x' 104
Modificationintervalledetempslte(x,i,j)=x' 105
Testertempslte(x,i,a) 105
EXPRESSSION 105
Modifierpositivementexpressionexp(exp,x)=exp'..... 106
Modifiernégativementexpressionexp(exp,x)=exp'..... 107
Testerexpressionexp(expression,nlres) 108
LIGNE DE FORMULE 108
LIGNE FINANCIERE..... 109
Actualiséunevaleurlfi(x,i,intérêt)=y..... 109
Actualiserlignelfi(l,taux,pérdép,pérfin)=l' 110
Valeuractuellelignelfi(pérdép,pérfin,lact)=lact'..... 111
LIGNE AMORTISSEMENT DEGRESSIF 111
Calculamortdégressiflad(lvl,lvl2,pér)=lad..... 112
Calculamortissementdégressiflad(t,x,l,pér)=l' 113
LIGNE RESIDUELLE 114
Calculrésiduellelre(l1,l2)=l3 114
LIGNE IMPOT..... 115
Calculimpôtlim(taux,lv)=l 115
LIGNE AMORTISSEMENT LINEAIRE..... 115
Amortissementlal(lvv,ld,i,j)=v..... 116
Amorttotalelal(lvv,ld,k,j)=v..... 116
Calculamortlinéairelal(lvv,ld)=lal..... 117
LIGNE TIR 117

Calcultirlti(lv,dépér,finpér)=ltir 118
Vanltir(lv,dépér,finpér,r)=v 118
LIGNE BENEFICE IMPOSABLE 119
Calculbéneficeimposablelbi(lv)=l 120

B) Glossaire des fonctions par ordre alphabétique

Accroissementvaleurtab(v,accrois,temps1,temps2,noligne,t)=
t' 46
Accumulertab(v,taux,temps1,temps2,noligne,t) = t' 45
Actualisationtab(nlcash,nlact,débpér,finpér,taux,t)=t' 46
Actualiserlignelfi(l,taux,pérdép,pérfin)=l' 110
Actualiséunevaleurlfi(x,i,intérêt)=y 109
Amortissementdégressiftab(pér,nlamdeg,nl,t)=t' 47
Amortissementlal(lvv,ld,i,j)=v 116
Amortissementlinéairetab(nd,nv,nal,t)=t' 48
Amorttotalelal(lvv,ld,k,j)=v 116
Avoirlignevaleurtab(nl,t)=t' 48
Bénéficeimposabletab(nlres,nlbenimp,t)=t' 49
Blanclibellécta(z)=z' 61
Calculactualisationcta(nlcash,nlact,débpér,finpér,taux,z)=z'
..... 62
Calculamortdégressifcta(z,pér,nl,nlamdeg)=z' 63
Calculamortdégressiflad(lvl,lvl2,pér)=lad 112
Calculamortissementdégressiflad(t,x,l,pér)=l' 113
Calculamortlinéairelal(lvv,ld)=lal 117
Calculbénéficeimposablecta(nlres,nlbenimp,z)=z' 65
Calculbénéficeimposablelbi(lv)=l 120
Calculertableaucta(z)=z' 71
Calculimpôtcta(nlimp,nlbenimp,taux,z)=z' 66
Calculimpôtlim(taux,lv)=l 115
Calcullacroissementlignelva(l,temps1,temps2,accrois,v)=l'
..... 101
Calcullamortineairecta(z,nd,nv,nal) =z' 64
Calculligneaccumulationlva(l,temps1,temps2,taux,v)=l' 102
Calculrésiduellecta(z,nlam,nlresi,nlbien)=z' 67

Calculrésiduellelre(l1,l2)=l3 114
Calculrésultatcta(expression,nlres,z)=z' 68
Calcultircta(z,nltir,nlres,déppér,finpér)=z' 69
Calcultirlti(lv,dépér,finpér)=ltir 118
Calculvancta(z,débpé,finpé,taux,nlcash,nllact) =z' 70
Corpstableauaccumulationcta(z,v,temps1,temps2,noligne,tau
x)=z' 74
Créationd'uneligneaccroissementcta(z,nl)=z" 75
Créationd'uneligneaccumulationcta(z,nl)=z" 75
Créationd'uneligneactualisationcta(z,nl)=z'' 76
Créationd'uneligneamortissementdégressifcta(z,nl)=z" 77
Créationd'uneligneamortissementlinéairecta(z,nl)=z" 78
Créationd'unelignebénéficeimposablecta(z,nl)=z" 79
Créationd'uneligneimpôtcta(z,nl)=z" 80
Créationd'unelignerésiduellecta(z,nl)=z" 81
Créationd'unelignerésultatcta(z,nl)=z" 82
Créationd'unelignetircta(z,nl)=z" 83
Créationd'unelignevaleurcta(z,nl)=z" 84
Créationd'unelignevaleurrépétitivecta(z,nl)=z" 85
Créationd'unelignevancta(z,nl)=z" 86
Créationlignepermisecta(z,nl) 87
Créertableaucta()=z' 88
Créertableautab (nom) = t 49
Ecrirelibellélig(nom,l) 99
Etendrelignecta(z)=z" 89
Etendreunelignedansuntableautab(t)=t' 50
Impôttab(nlbenimp,nlimp,taux,t)=t' 50
Initialisationcta()=z'''' 89
Initialiserlignecta(z)=z' 90
Initialisertempscta(z)=z' 90
Initlte(x)=x' 104
Inscrirelibellétab(t, nom,nl) = t' 51
Insérerаccroissementcta(z,v,accrois,temps1,temps2,noligne)=
z' 91
Insérerlignedansuntableautab(t,x) = t' 51
Insérervaleurcta(z,n,année1,année2)=z' 92
Insérervaleurrepcta(v,temps1,temps2,n,z)=z' 93
Insérervaleurrépétitivelva(v,temps1,temps2,lv)=lv' 102

Insérervallva(lv,débuttemps,fintemps)=lv' 103
Limitertaillecta(z,x)=z' 94
Limitertailletableautab (t,x)=t' 52
Miseàblanclibellélig(l)=l' 99
Miseàzérodelalignescta(z,nl)=z' 94
Miseàzérolva(lv)=lv' 103
Modificationintervalledetempslte(x,i,j)=x' 105
Modifierlesdeuxtempstab(t,i,j)=t' 52
Modifiernégativementexpressionexp(exp,x)=exp' 107
Modifierpositivementexpressionexp(exp,x)=exp' 106
Modifiervaleurtab(t,n,année1,année2)= t' 53
Réinitialisationdutableautab (t) = t' 53
Remodifiercorpscta(z,x)=z' 95
Résiduelletab(nlam,nlbien,nlresi,t)=t' 54
Résultattab(expression,nlres,t)=t' 54
Saisirvaleurtab(t,n,année1,année2) = t' 55
Supprimercorpscta(z,x)=z' 97
Supprimerlignedutableautab(x,t) = t' 55
Supprimerunlibellétab(t,n) = t' 56
Supprimervaleurtab(t,n,année1,année2)= t' 56
Testerexpressionexp(expression,nlres) 108
Testertempslte(x,i,a) 105
Tirtab(nltir,nlres,t,débpér,finpér)=t' 57
Valeuractuellelignelfi(pérdép,pérfin,lact)=lact' 111
Valeurrépétitivetab(t,v,temps1,temps2,n) = t' 58
Vanltir(lv,dépér,finpér,r)=v 118
Vantab(t,débpé,finpé,taux,nolignecashflow,nolvalactunette)=
t' 59
Vérificationtableauzérocta(z) 98
Vérifierzérolva(l) 104

Conclusion

Ce mémoire a pour but de montrer l'importance de l'ingénierie des besoins. Parmi les phases représentatives de cette activité, ce document se concentre sur les phases d'élicitation et de modélisation et est organisé autour de trois chapitres. Le premier traite des différentes interfaces qui sont possibles pour traiter un problème d'investissement et correspond à la phase d'élicitation. Le deuxième chapitre traite d'une architecture possible de la spécification suivant le paradigme orienté objet et le troisième chapitre est la spécification du problème de l'investissement suivant le langage formel GLIDER. Ces deux derniers chapitres correspondent à la phase de modélisation.

Dans le premier chapitre, une discussion globale sur le problème de l'investissement et des différentes interfaces possibles est proposée. Ce qui est intéressant de remarquer, c'est l'évolution vers laquelle nous nous dirigeons. Si au départ un logiciel pouvait servir pour différentes applications tel que FMP, LOTUS,EXEL, la nouvelle tendance voudrait limiter cette tendance à des logiciels à portée beaucoup plus limité, mais avec une interface qui correspond réellement au problème traité. L'énorme désavantage de cette vision est le coût. Est-ce qu'une entreprise est prête à payer 50 applications différentes avec des interfaces bien adaptées plutôt que d'utiliser un produit qui permet de tout faire mais dont l'interface est très mauvaise.

Le deuxième chapitre discute de la manière dont l'architecture a été choisie. L'architecture est de type "orientée objet". Ces avantages sont la réutilisation, une division par module plus claire. Pour concevoir de façon orienté objet, une méthode est proposée. Face au problème de l'investissement le concept principal identifié est le concept de ligne associé au temps. L'objet ligne est spécialisé en types de lignes plus particulières telles: une ligne impôt, une ligne de valeurs etc. .

Le troisième chapitre vise à spécifier le problème de l'investissement en utilisant le langage GLIDER qui est un langage formel de spécification reposant sur le paradigme objet. La spécification en langage GLIDER est organisée en un ensemble de modules qui possèdent une structure de données et des opérations associées à la structure de données. L'utilisation de ce langage s'est avérée intéressante au point de vue :

-de l'expressivité offerte: la notion d'"OBJECT"(observations associées au temps) est introduite,
-des mécanismes de structurations: l'héritage et la généricité en sont les piliers,
-formalisme:les services associés à la structure de données s'expriment par des formules logiques de premier ordre typé.

Pour conclure, indiquons trois aspects qui pourraient être davantage étudiés par la suite: l'apport de la théorie de l'investissement, l'implémentation, l'intérêt de la réutilisation.

Parmi les apports de la théorie de l'investissement, nous nous sommes limités aux concepts simples de la théorie à savoir le TIR et la VAN. Cependant la théorie de l'investissement regroupe bien d'autres concepts tel que : la "pay-back période", les investissements de type internationaux comprenant l'inflation et les conversions de devises, le taux interne de rentabilité par franc engagé, le classement des investissements, la comparaison des investissements à sommes différentes et cela sur périodes différentes, l'utilisation de paramètres difficilement quantifiables pour le choix d'un investissement (analyse multicritère), le choix de sa période est aussi utile car dans le calcul de la VAN nous supposons que la VAN se calcule à partir du 31 décembre et non à partir du premier janvier. De plus, les choix qui ont été faits supposent une semi-automatisation des problèmes de l'investissement. Par exemple, lors du calcul de la VAN, une série de cash flow sont supposés positifs. S'ils sont négatifs, il faut les accumuler jusque la dernière période et puis actualiser. Ce type de raisonnement est laissé à l'utilisateur car, en fait, cela a une incidence mineure. Un autre domaine qui n'est pas exploré est la détermination du coût du capital de manière plus complexe.

D'un point de vue implémentation, l' analyse de l'interface devrait être davantage finalisée, en particulier la gestion globale d'un problème d'investissement devrait être effectué à partir d'un browser Cela faciliterait énormément les liaisons entre la feuille de calcul et les feuilles graphiques associées. Maintenant si l'utilisateur désire faire un graphique, la solution la plus efficace serait de reprendre l'interface du gestionnaire décrit au chapitre 1 et d'insérer l'option graphique dans le menu formule. Si un utilisateur se trouve dans une ligne de TIR et qu'il sélectionne "graphique" du menu formule, les différents graphiques en fonction du type de ligne lui sont proposées automatiquement (p.ex.:évolution du taux d'intérêt en

fonction des VAN). S'il choisit ce graphique, il est affiché. Proposer une interface est un point mais à nouveau il faut toujours avoir à l' esprit de ce que coûte toutes ces idées.

Pour déterminer si des spécifications sont bonnes, il faut s'assurer de son taux de réutilisation [Joh88],[Fer89]. Deux questions sont soulevées: est-ce que la structure de données permet la réutilisation ? Est-ce-que les services offerts dans la structure de données permettent la réutilisation. A la première question, il importe de noter que la structure de données est très générale et par conséquent très réutilisable. Prenons une exemple : la structure de données LIGNE AMORTISSEMENT LINEAIRE est un produit cartésien de type INTEGER,INTEGER,LIGNE DE VALEUR. Ce type de structure est une structure qui se trouve dans beaucoup de contextes tels que les statistiques, les problèmes économiques ou chaque observation est associée au temps et ou l'on désire conserver une ou deux valeurs de référence. Quant aux services offerts, il faut les voir suivant la manière dont ils vont être utilisés. Par exemple, dans le problème de l'investissement, l'opération valeur actuelle a été spécifiée en deux étapes. La première actualise la ligne et la seconde effectue l'accumulation des valeurs. Pour un statisticien le mot VAN ne lui dit rien, mais le mot accumulation lui est connu. Par conséquent, la seule modification qu' il doit apporter est le renomage de l'opération.

Bibliographie

[Bec91] Becker K., Bodart F., " Reusable Object-Oriented Specification for Decision Support Systems, In : IFIP-WG-8. 1 Working Conference on the Object-Oriented Approach in Information Systems, Quebec City, Proceedings North-Holland p 137-155, Oct 28-31, 1991.

[Dub90] Dubois E., Hagelstein J., Rifaut A., " ERAE: a Formal Language for Expressing and Structuring Real-Time Requirements ", Philips Research Labotory, Manuscript 353, Louvain-la-Neuve, June 1990.

[Dub91a] Dubois E., Du Bois Ph., Rifaut A., Wodon P., " Glider User Manual", Esprit Project Icarus 2537, Task-spec-Func-028-R, Namur-Brussel, July 1991.

[Dub91b] Dubois E., Hagelstein J., Rifaut A., " A Formal Language for the Requirements Engineering of Computer Systems. ", In " From Natural Language Processing to Logic for Expert Systems ", A. Thayse(ed), Wiley, NY, 1991.

[Dub92] Dubois E., Du Bois P., Rifaut A., " A wide -Spectrum Formal Language for Sructuring and Requirements of Composite Systems ", In Proc., 4th International Conference CAISE 92 LNCS 593, Springer-Verlag, 1992.

[Boo90] Booch G., " Object Oriented Design with Application", Benjamin Cummings, Redwood, CA(USA), 1990.

[Coa89] Coad P., Yourdon E., "Object Oriented Analysis", Prentice-Hill, New-jersey, 1989.

[Cou87] Could J. D., " An Experimental Study Of People Creating Spreadsheets", ACM, Transactions on Office Systems, Vol 5, No3, July 1987.

[Cou89] Couvreur J. P., " La décision d'investir et la politique de l'entreprise", Entreprise Moderne d'Edition, Paris, 1989.

[deD90] de Donnea F. X., "Méthode d'évaluation des projets publics ", Ciaco, Louvain-la -Neuve, 1990.

[Din91] Dinnematin S., "IMPROV-LOTUS réinvente le tableur ", SVM, mars 1991.

[Fer89] Fergini M. B., Pernici B., " Recast: a tool for Reusing Requirements", Politechnico di Milano, Milan, December 1989.

[Fic87] Fickas S., "Automating The Analysis Process : an Example", Computer Science Departement, In Proc. 4th International Workshop on Software Specification and Design, Monterey, CA (USA), 1987.

[Fic87] Fickas S., Collins S., Olivier S., " Problem Acquisition in Software Analysis : a Preliminary study ", Tech. Rep. 01-97403, Computer Science Departement, University of Oregon, Eugène, Augustus 87.

[Fun] FUN-Namur (B), INRIA/CRIN-Nancy(F)., PRLB-Brussels(B), "First Version of the Icarus Product and Process Language for the specification of the system Functionalities", Esprit Project Icarus 2537, TaskSpec -Func-016-R.

[Hen80].Heninger K.L., "Specifying Software Requirements for complex Systems : New techniques and their applications", I.E.E.E. Tron.Soft., Eng, Vol. SE-6, 1, 1980, P2-13.

[Gol85] Goldberg A., Robson D., "Smalltalk -80 : the Language and its Implementations ", Addison-Wesley Publishing company, London, 1985.

[Gui92] Guillaume M., " Projet Expan ", Faculté Universitaire de la Paix, département économique, Namur, 1992.

[Joh88] Johnson R. E., Foote B, "Designing Reusable Classes ", Journal of Object-Oriented -Programming, Vol 1, No 2, June 1988.

[Kar90] Karson T. " Understanding Object-Oriented an Unifying Paradigm ", Communication of the ACM, September 1990, Vol 33, No 9.

[Lal90] Laloux D., "Exel facile", Marabout, Alleur (Belgique), 1990.

[Mey80] Meyer B., "Sur le formalisme des spécifications", Globule, Bulletin du groupe de travail "Génie logiciel " de l'AFCET, 1980.

[Mey90] Meyer B., "Conception et programmation par objet", InterEdition, Paris, 1990.

[O'b85] O' Brien P.,"Trellis Object-Based Environmemt Version 1. 1", Digital Equipment Corporation, Hudson -Massachussets, 1985.

[pig91] Pigot T., " Programmation Orienté -Objet ", Info PC, No 75, Octobre 1991.

[Qui87] Quintart A., Zisswiller R., " Investissement et désinvestissement de l'entreprise : pratique et méthodes", Dalloz Gestion, Paris, 1987.

[Reb90] Rebecca J., Johnson R., "Current Research in Object-oriented Design", Communication of the ACM, Vol 33, No 9, September 1990.

[Rol92] Rolland C., "Conception de système d'information orienté objet ", note de cours de la chaire Francqui, Institut d'Informatique, Namur, 1992.

[Sch80] Schepens G., "Financial Model Processor", Faculté Universitaire de la Paix, département économique, Namur, 1980.