**MASTER EN SCIENCES INFORMATIQUES**

# La Problématique de la Gestion d'Identité dans les Grandes Organisations

Desart, Thomas

*Award date:*
2016

*Awarding institution:*
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2015–2016

# La Problématique de la Gestion d'Identité dans les Grandes Organisations

Thomas Desart


Maître de stage : Denis Zampunieris

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

# Résumé


## Résumé

Nous possédons tous de multiples identités au travers des différents services que nous utilisons quotidiennement. Ces identités sont bien évidemment stockées sur des serveurs, plus précisément des serveurs d'identité. La gestion des identités est devenue en quelques années un réel défi pour les organisations qui ne cessent de grandir et dont le nombre d'utilisateurs ne cesse d'augmenter. La maintenance des serveurs d'identité dans les grandes organisations n'est pas sans poser quelques problématiques et est devenue une tâche qui se complexifie au fil du temps. Ce mémoire aborde certaines de ces problématiques liées à la maintenance d'un serveur d'identité. Nous proposons également un outil d'aide à la gestion d'un tel serveur sous forme d'un prototype grâce à un nouveau mode de calcul appelé "Proactive Computing". L'outil proposé offre un mode de gestion plus aisé pour l'administrateur facilitant ses tâches quotidiennes.



**Mots clés :** Gestion de l'identité, Proactive Computing, Automatisation, Sécurité



## Abstract

We all have multiple identities across different services that we use on a daily basis. These identities are obviously stored on servers, specifically on identity servers. Identity management has become in recent years a challenge for organisations that are growing and in which the number of users is only increasing. Maintenance of identity servers in large organisations is not without raising any problems and is a task that becomes more complex over time. This thesis presents some of these issues related to the maintenance of an identity server. We also propose a prototype for supporting the management of such servers based on a new calculation method called "Proactive Computing". The proposed tool facilitates the daily identity management tasks to be done by the administrator.



**Keywords :** Identity management, Proactive Computing, Automation, Security

# Avant-propos

Ce mémoire est l'aboutissement du travail effectué lors de mon stage de trois mois à l'Université du Luxembourg. Plus précisément il s'est déroulé au sein de la faculté des Sciences, de la Technologie et de la Communicationdans le laboratoire des systèmes logiciels avancés dénommé le LASSY. Ce stage a été supervisé par le Professeur Denis Zampunieris, que je tiens particulièrement à remercier pour son suivi et sa disponibilité durant mon séjour à Luxembourg. Je tiens à remercier également l'ensemble de son équipe ainsi que les autres membres du laboratoire qui ont tous contribué au bon déroulement de ces trois mois de stage.

Je tiens aussi à remercier mon promoteur, le Professeur Jean-Noël Colin pour ses riches et nécessaires commentaires lors de l'élaboration et la rédaction de mon mémoire, ainsi que pour sa disponibilité.

J'adresse également mes remerciements à tous les professeurs et doctorants rencontrés lors de mon parcours universitaire pour les connaissances et conseils qu'ils m'ont apportés d'une façon ou d'une autre. Indirectement, ils ont tous énormément contribué à l'élaboration de ce mémoire.

Je pense aussi à tous les scientifiques et les auteurs qui ont développé, recherché et écrit le contenu des articles qui sont à la base de mon mémoire.

Enfin, je remercie également ma famille pour leurs nombreuses relectures et leur patience, ainsi que mes amis pour leur soutien durant toutes mes études et plus particulièrement durant la réalisation de ce travail.

J'ai également pour terminer, une pensée émue pour mon grand-père qui nous a quitté très récemment et à qui j'aimerais dédier ce travail.

# Table des matières

# Table des figures

# Acronymes

**AAS** Assurance Assessment Scheme. 30
**AD** Active Directory. 46, 66, 68–70, 74, 75, 79, 87, 89, 90, 96, 99
**AL** Assurance Level. 29, 30
**AQR** Assessor Qualification & Requirements. 30

**CSP** Credential Service Provider. 29, 30

**DN** Distinguish Name. 71, 72, 85, 87, 91

**IAF** Identity Assurance Framework. 11, 29, 30
**IdP** Identity Provider. 22, 23, 26, 29, 30

**PE** Proactive Engine. 9, 53, 62–64, 66, 96
**PKI** Public Key Infrastructure. 35

**RBAC** Role-Based Access Control. 11, 44–47, 51, 52
**RRS** Rule-running system. 59, 60

**SAC** Service Assessment Criteria. 29, 30
**SP** Service Provider. 23, 29, 30
**SSO** Single Sign-On. 7, 8, 11, 17, 18, 25, 26, 33, 34, 36, 40, 41, 43, 49

**URI** Uniform Ressource Identifier. 20

# Introduction

La plupart des services que nous utilisons au travers des réseaux ou en ligne demandent une authentification de notre identité avant de nous accorder l'accès aux services qu'ils proposent. Le moyen d'authentification le plus répandu que nous connaissons tous et que nous utilisons quotidiennement prend la forme d'un couple nom d'utilisateur et mot de passe. Pour que l'on puisse utiliser ces services autant de fois que souhaité, les organisations stockent notre identité sur des serveurs dédiés. La gestion de ces identités n'est pas une chose aisée pour celles-ci. La maintenance de ces serveurs destinés au stockage des identités peut se révéler être une véritable épreuve pour les administrateurs à cause du nombre élevé et toujours croissant d'identités présentes sur ces serveurs. Ce nombre élevé d'identités ne fait que complexifier les mécanismes de maintenance qui demande énormément de temps. Une organisation négligeant la gestion des identités peut se voir confrontée à de réels problèmes touchant notamment sa sécurité informatique, ainsi que la protection de la vie privée de ses utilisateurs.

Nous avons dès lors voulu dans ce mémoire présenté la problématique de la gestion d'identité dans les grandes entreprises. Nous estimons que cette problématique est en passe de devenir un problème de plus en plus conséquent et complexe dans le monde informatique des organisations notamment au niveau des tâches de maintenance que cela représente. Le nombre croissant d'utilisateurs et de services applicatifs offerts ne font qu'augmenter la complexité et les coûts de maintenance liés à ces systèmes d'identité.

Afin de faire face à cette croissance constante, nous avons imaginé qu'il ne serait peut-être pas inutile de réfléchir à la possibilité d'automatiser en partie ou complètement les tâches de maintenance que l'on peut retrouver dans la gestion d'un système d'identité. Cet ajout d'autonomie aux systèmes existants permettrait premièrement de dégager du temps pour les administrateurs afin qu'ils puissent se concentrer pleinement sur d'autres problèmes plus complexes. Deuxièmement, cela permettrait de réduire les coûts liés à la maintenance tout en augmentant la fiabilité des systèmes d'informations de l'organisation. Nous présentons effectivement une solution autonome ne nécessiterant pas ou peu d'interactions humaines pour fonctionner.

Cette solution autonome a été réalisée grâce à un nouveau mode de calcul qu'est le Proactive Computing. Le Proactive Computing, consiste en une nouvelle façon de voir l'informatique proposant des solutions liées à l'explosion du nombre d'ordinateurs personnels et embarqués. En effet, l'informatique est de plus en plus omniprésente dans notre environnement. Le nombre d'équipements informatiques a plus que dépassé le nombre d'individus que compte la planète. Il est dès lors important de trouver comment tous ces équipement peuvent être exploités sans nécessiter d'interactions de notre part. Le Proactive Computing

apporte des solutions à ce problème en invitant l'individu à sortir de la boucle de contrôle et en devenant le superviseur de tous ces équipements. Dans ce cas de figure, le superviseur n'intéragit donc plus que de manière limitée en cas de décisions critiques.

Ce mémoire est la suite logique du travail effectué lors du stage réalisé durant le premier quadrimestre de l'année académique 2015-2016 à l'Université du Luxembourg. Le but de ce stage était de proposer un outil autonome d'aide à la gestion d'un Active Directory en utilisant un moteur proactif précédemment implémenté à l'Université.

Ce mémoire est structuré autour de quatre chapitres. Le premier chapitre est dédié à la définition des différents concepts que contient la gestion d'identité ou identity management. Parmi les concepts, nous étudierons bien sûr la définition du mot identité. Nous découvrions ensuite les différents acteurs de la gestion d'identité ainsi que le cycle de vie normal qu'une identité peut avoir dans une organisation. Nous verrons aussi qu'il existe différents modèles qui permettent de mettre en place la gestion d'identité. Ces modèles sont tous différents et présentent tous des avantages et inconvénients. Ce premier chapitre se terminera par l'énoncé des différentes exigences formulables par les parties prenantes à la gestion d'identité.

Le deuxième chapitre, quant à lui est réservé aux problématiques liées à la gestion d'un serveur d'identité. Durant le stage effectué, nous avons identifié trois problèmes représentatifs majeurs auxquels nous avons essayé d'apporter une solution. Ces problèmes peuvent paraître bénins au premier abord, mais ce sont des problèmes qui engendrent des tâches extrêmement chronophages pour les administrateurs de ce type de serveurs. Ces problèmes peuvent notamment être résolus en fournissant une assitance partiellement ou complètement automatisé à l'administrateur du système.

Le troisième chapitre est lui consacré à une présentation du mode de calcul appellé Proactive Computing et choisi pour l'implémentation des solutions apportées aux problèmes invoqués dans le deuxième chapitre. Ce chapitre débutera par la définition des concepts clés. Nous verrons d'où vient l'idée du Proactive Computing soutenu par deux grandes firmes bien connues ayant proposé leurs idées et réflexions par rapport à ce nouveau système. Ce chapitre contiendra aussi la description du moteur proactif utilisé dans le prototype réalisé et pour lequel nous avons spécifiquement conçu quatre scénarios proactifs qui sont les composants principaux de la solution proposée. Enfin, ce chapitre répondra à la question suivante : le Proactive Computing est-il un bon moyen pour concevoir des systèmes d'informations autonomes tels que des systèmes de gestion de l'identité.

Le dernier et quatrième chapitre, vous présentera la solution que nous avons imaginée et implémentée tentant de résoudre les problèmes évoqués dans le chapitre deux. Cette solution utilise le mode de calcul proactif en se servant du moteur proactif, qui sont présentés dans le chapitre trois. Au total, vous découvrirez les quatre scénarios proactifs qui composent cette solution. Les détails de leur implémentation y seront décrits. Pour clore ce dernier chapitre, nous proposons une évaluation de la solution présentée.

# Chapitre 1

# Gestion de l'identité

## Sommaire

## 1.1 Définitions

### 1.1.1 Identité

Nous allons commencer ce chapitre par une exploration en profondeur du concept d'identité, notion clé de la gestion d'identité ou identity management. La compréhension de ce mot est clée pour saisir toute l'importance de l'identity management. Le terme identité est un terme complexe et nous allons d'abord le définir de manière générique pour ensuite le spécifier dans le domaine qui nous intéresse. La définition du dictionnaire Larousse [2] pour le mot identité est la suivante :

> *« Caractère permanent et fondamental de quelqu'un, d'un groupe, qui fait son individualité, sa singularité »*

Cette première définition nous apprend que l'identité serait donc un moyen possible d'identifier de manière unique une personne ou un groupe à l'aide de caractéristiques qui lui sont propres. Sur base de cette définition générale du mot identité, définissons maintenant son sens à un niveau plus informatique.

Dans l'article « Trust Requirements in Identity Management » [19], l'identité est définie comme suit :

*« Un ensemble de caractéristiques propres par lesquelles une personne ou une organisation est connue ou reconnue »*

Ces caractéristiques peuvent être définies, comme le nom, l'adresse, la nationalité, ou peuvent être innées comme les empreintes digitales. Pour l'identité d'une organisation, les caractéristiques sont acquises. Cette définition étend celle présentée précédemment rattachée aux personnes aux concepts des organisations. De plus, elle nous indique que l'identité serait un moyen de distinguer une personne ou une organisation spécifique. Cette deuxième définition est certes plus précise, mais n'est pas encore satisfaisante.

Afin de trouver une définition satisfaisante, nous nous penchons vers la définition fournie par les standards existants dans le cadre de travail pour la gestion de l'identité, notamment le standard international ISO/IEC 24760-1 [16], basé sur la recommandation UIT-T Y.2720 [17] rédigée par l'Union Internationale des Télécommunications qui étend la définition d'identité à

*« L'information utilisée pour représenter une entité dans un système d'information et de communication »*

Une entité représente une personne physique ou morale (organisation, entreprise...), une ressource (un objet tel qu'un matériel informatique, un système d'information ou de communication) ou un groupe d'entités individuelles.

Une entité peut posséder plusieurs identités numériques. Chaque identité est alors relative à au moins un domaine de travail. Ainsi, un individu peut présenter, par exemple, des informations publiques le concernant dans le cadre de son activité professionnelle, ce qui représentera une identité mais aussi d'autres informations personnelles le présentant dans son contexte familial, ce qui désignera une autre identité.

Une identité peut être utilisée dans plusieurs domaines. Conjointement, dans un même domaine, une entité peut être incarnée par plusieurs identités. De plus, plusieurs identités d'une même entité peuvent partager les mêmes caractéristiques, ce qui implique que les identités peuvent ne pas être uniques dans un même domaine. La figure 1.1 illustre ces propos.

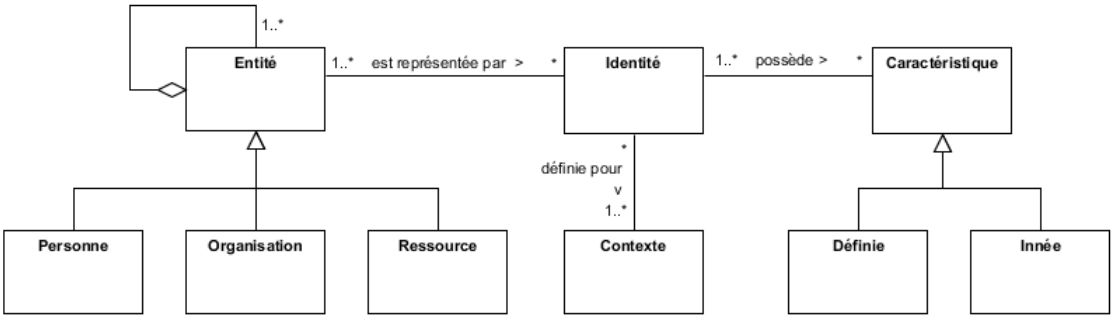


FIGURE 1.1 – Diagramme de classe du concept d'identité

Selon [16] et [17] l'information qui compose l'identité, consisterait en trois types de données différentes : un identifiant, un ou des justificatif(s) d'identité (en anglais credentials) et des attributs. Ces propos sont illustrés à la figure 1.2

- Identifiants : une série de chiffres, de caractères et de symboles, ou toute autre forme de données utilisées pour identifier un sujet dans un contexte particulier. Les identifiants peuvent être référencés par temps ou par espace. Par exemple, un Uniform Ressource Identifier (URI) est globalement unique au fil du temps. Les pseudonymes sont temporaires et effectifs seulement pour un service spécifique. L'identifiant peut être un attribut. Dans ce cas, la valeur d'un identifiant ne peut pas être utilisée par plusieurs identités. De ce fait, il est généralement utilisé dans le processus d'identification qui est responsable de la reconnaissance de l'identité dans un contexte. De même qu'une entité peut posséder plusieurs identités pour présenter des informations en fonction des contextes, une identité peut être représentée par plusieurs identifiants. Quelques exemples d'identifiants sont les noms de compte utilisateur, les numéros de passeport, les numéros de téléphone, les trigrammes et les URI.
- Justificatifs d'identité : un ensemble de données qui fournit des preuves pour attester d'une partie ou de l'entièreté d'une identité. Un justificatif d'identité peut être généré à partir d'un ou plusieurs autres justificatifs. Ces justificatifs sont évidemment connus seulement par l'identité. Nous pouvons prendre comme exemple les mots de passe, les certificats digitaux, les empreintes digitales...
- Attributs : un ensemble de données qui décrivent les caractéristiques d'une identité. Les données incluent les informations fondamentales permettant d'identifier une identité (par exemple : nom, prénom, adresse et date de naissance), ses préférences et les informations générées résultant de ses activités. Chaque attribut est défini par un type, une valeur et un contexte. Il peut éventuellement avoir un nom qui peut être utilisé pour le référencer. Nous pouvons prendre comme exemple pour un attribut, le nom, le prénom, l'adresse, l'âge, le genre, les rôles, les titres, les affiliations et la réputation... Un attribut peut être certifié par un organisme de confiance et alors appelé « claim » ou certificat en français [10].

Nous pouvons déduire de cette définition du concept d'identité au sens informatique que l'identité est pour la plupart du temps conceptualisable sous forme de structures et de processus [7]. La structure définit l'identité comme un ensemble d'attributs caractérisant une personne. Les processus quant à eux définissent l'ensemble des traitements touchant à l'identification de la personne, à la déduction d'informations complémentaires sur cette personne et l'exploitation de ces informations.

### 1.1.2 Gestion de l'identité

La gestion d'identité porte donc simplement sur la gestion de toutes les informations concernant une entité. Toutefois selon la norme ISO/IEC 24760-1 [16] cela va plus loin. La gestion d'identité est définie comme suit :

> *« Les processus et règles utilisées dans la gestion du cycle de vie des identités, les structures de données caractérisant la notion d'identité dans un domaine particulier»*

La gestion d'identité aurait donc une fonction plus complexe que ce que nous aurions pu

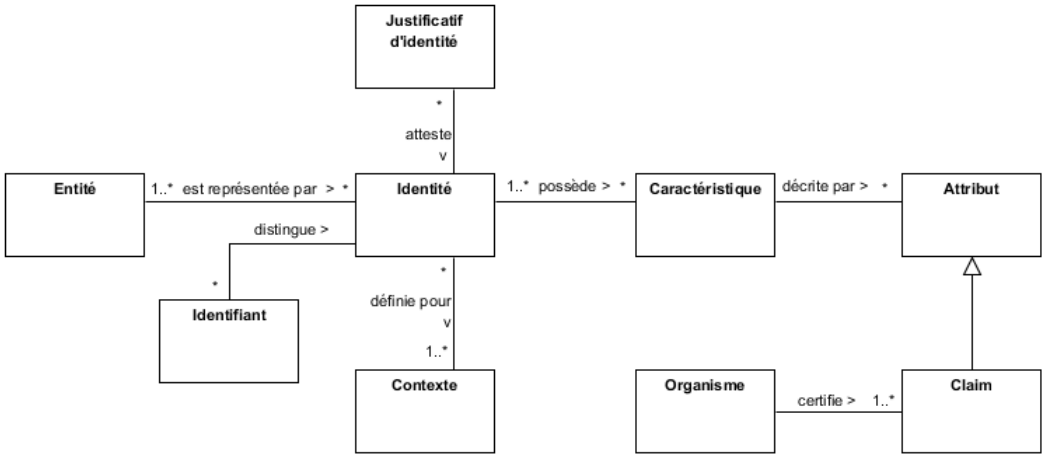


FIGURE 1.2 – Diagramme de classe des concepts d'identifiant et d'attribut

croire. Cette fonction est de maintenir l'intégrité des identités à travers leur cycle de vie dans le but de rendre disponibles les identités et leurs données relatives à des services applicatifs de manière sécurisée tout en respectant la protection de la vie privée [7]. Cette définition suscite donc des questions de sécurité et de vie privée. Les différentes parties impliquées et le détail du cycle de vie d'une identité seront détaillés plus tard dans ce chapitre respectivement aux points 1.2 et 1.3.

### 1.1.3 Fédération d'identité

Une fédération d'identité est un accord entre plusieurs domaines d'identités qui spécifie comment les différentes parties prenantes peuvent échanger des informations relatives aux identités. L'accord définit les protocoles utilisés, les formats des données et les procédures de protection et d'audit. L'identité ainsi fédérée pourra être utilisée dans le regroupement des différents domaines.

## 1.2 Parties prenantes

Il y a plusieurs intermédiaires qui ont un intérêt dans la gestion d'identité et les solutions qui implémentent de tels systèmes se doivent de prendre en compte les exigences de chaque partie [7] et [10]. Nous pouvons organiser ces parties en différents groupes présentés dans les sous-sections suivantes.

### 1.2.1 Sujets

Les sujets sont typiquement les individus dont les attributs d'identité ont été stockés dans un certain but. Il y a un grand nombre d'attributs possibles, que nous pouvons classer comme suit :

- **Les attributs qui se basent sur des documents légaux :** ce sont des attributs issus de documents administratifs, par exemple un numéro de registre national. Du fait que les documents administratifs sont considérés comme étant les documents les plus forts en matière d'identification, les attributs provenant de ces documents doivent être fortement protégés.
- **Les attributs démographiques :** ce sont des attributs contenant l'âge, le genre, le pays de résidence, l'adresse du domicile. Ces informations nécessitent également d'être protégées, car une combinaison de ces informations avec d'autres peut conduire à l'identification de certains individus causant ainsi un problème de violation de la loi relative au respect de la vie privée.
- **Les attributs financiers :** ce sont des attributs provenant d'institutions financières comme un numéro de carte bancaire ou de compte. Ces informations sont des attributs largement répandus et très souvent la cible de tentatives de vol. Ils doivent aussi être également fortement protégés.
- **Les attributs biométriques :** ces attributs incluent différentes caractéristiques physiques d'un individu, telles que les empreintes digitales ou de l'iris. Ils représentent un moyen puissant de vérifier l'identité d'un individu. Leur utilisation est encore controversée et peut poser des problèmes lors du processus de reconnaissance.
- **Les attributs transactionnels :** ce sont des attributs très dynamiques caractérisant les interactions que le sujet opère sur internet notamment. Les reçus électroniques représentent un exemple intéressant d'un tel type d'attributs. Ils sont utiles dans le sens que cela permet d'offrir au sujet un service amélioré. Ils doivent être adéquatement protégés, car ils peuvent révéler des informations privées concernant les utilisateurs comme les habitudes de consommation.

La respect de la législation en matière de protection de la vie privée et de mauvaises manipulations de ces données constituent les exigences principales de la part des sujets ou utilisateurs concernés. La mauvaise utilisation de ces attributs pourrait potentiellement porter préjudice aux individus. L'augmentation du nombre de cas constat de vol d'identité impose la mise en place de solutions recourant à des mécanismes hautement sécurisés.

### 1.2.2 Fournisseurs d'identités

Le fournisseur d'identités ou Identity Provider (IdP) en anglais est l'organisme qui met à disposition des identités pour des sujets. Il effectue quatre tâches :

1. Générer et assigner des attributs d'identité spécifiques à un sujet
2. Faire le lien entre un attribut d'identité d'un sujet et les autres attributs d'identité de ce sujet
3. Générer les certificats d'authentification relatifs aux attributs d'identité
4. Fournir les justificatifs contenant les attributs d'identité ainsi que les certificats d'identité

Un IdP peut lier la valeur des attributs qu'il fournit avec des attributs fournis par d'autres IdP. Une justification émise par un IdP ne contient pas nécessairement que des attributs fournis par cet IdP, mais peut aussi contenir des attributs fournis par d'autres IdP. Par exemple, un loueur de voitures vous demandera votre carte d'identité comme preuve de votre identité.

Les Identity Providers doivent pouvoir accorder leur confiance à d'autres IdPs dans le cas où ils utilisent des justificatifs émis par ces derniers. Une des exigences émises par les IdPs porte sur la mise en place de processus d'assurance d'identité (identity assurance processes). Un processus d'assurance d'identité permet simplement d'accorder un certain degré de confiance à tel ou tel attribut.

### 1.2.3 Fournisseurs de services

Les fournisseurs de services ou Service Provider (SP) ou encore relying parties en anglais sont des intervenants qui en contrepartie de fournir l'accès à des services ou des ressources, exigent l'authentification de l'identité fournie par un fournisseur d'identité. La plus grosse exigence de ces intervenants est de pouvoir déterminer le niveau de confiance qu'ils peuvent accorder aux justificatifs d'identité qu'ils reçoivent. Certains services requièrent évidemment un niveau d'assurance plus élevé (fort) que d'autres. C'est pourquoi l'existence d'un processus d'assurance d'identité est également primordial pour ces intervenants. Une deuxième exigence émise par les fournisseurs de services est qu'ils doivent avoir la possibilité de vérifier certains attributs auprès des émetteurs ; une infrastructure de vérification doit donc être mise en place. Les SPs doivent également se conformer aux lois et normes en vigueur afin de prévenir d'éventuels vols d'identité, d'où la mise en place de processus de vérification de conformité.

### 1.2.4 Organismes de contrôle

Les organismes de contrôle sont typiquement les forces de l'ordre ou des organismes de vérification qui doivent avoir accès aux identités. Par exemple, l'accès à des journaux de transactions impliquant l'utilisation d'une identité ou d'autres données peut être exigé dans le but de mener des investigations. L'exigence principale de ces organismes est l'auditabilité et l'existence de tels supports permet de detécter des fraudes et anomalies éventuelles.

### 1.2.5 Relations entre les acteurs

Les relations qui existent entre les différents acteurs sont présentées dans la figure 1.3. Il est important de noter qu'un intervenant peut exercer plusieurs rôles. Un sujet peut être son propre fournisseur d'identité ou interagir directement avec un IdP ou un fournisseur de services. Mais il peut aussi y avoir des échanges entre IdP et SP sans besoin d'interactions du sujet, par exemple une publicité personnalisée en fonction des actions du sujet sur un site web.

Conventionnellement, chaque fournisseur de services implémente sa propre gestion de l'identité, ce qui implique que les sujets recoivent des noms d'utilisateurs et des mots de passe pour chaque service utilisé. L'introduction des fournisseurs d'identité et de services a pour but d'encourager les sujets à confier la gestion de leurs identités aux IdPs. Les sujets pourraient alors n'utiliser qu'un seul compte acquis auprès du fournisseur d'identité. Dans ce modèle, les sujets ne sont pas les seuls bénéficiaires, les fournisseurs de services sont également gagnants. En effet, déléguer leur gestion d'identité à un IdP leur permettrait de mieux se concentrer sur l'amélioration de leurs services principaux. Les fournisseurs d'identité ne font sens que s'ils peuvent travailler en collaboration avec les fournisseurs de services. Cette répartition permettrait de distribuer ainsi les coûts et les IdPs pourraient de cette manière fournir des services encore plus avancés et développés. L'implémentation de tels services n'est évidemment pas sans risque pour la sécurité et le respect de la vie privée.

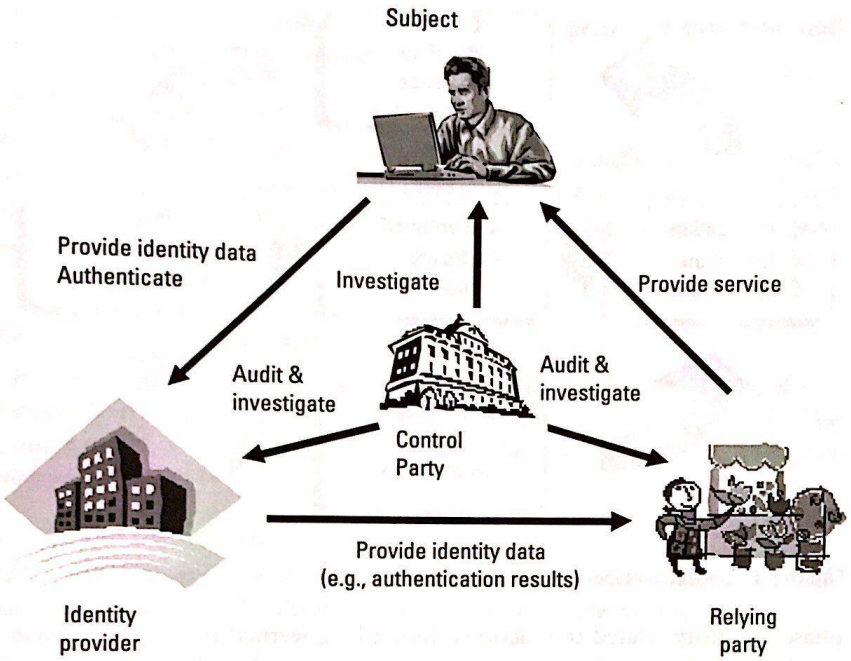


FIGURE 1.3 – Les 4 intervenants de la gestion d'identité [7]

## 1.3 Cycle de vie de l'identité

Les identités doivent être maintenues durant leur cycle de vie. Comme l'indique la définition de la gestion de l'identité à la sous-section 1.1.2, la gestion de l'identité n'est pas uniquement constituée de fonctions destinées aux utilisateurs, mais concerne également tous les aspects de gestion de l'identité, à savoir de la création à la suppression de celle-ci. Le cycle de vie de l'identité soutenu par une gouvernance consiste en quatre phases : la création, l'utilisation, la modification et la révocation de l'identité, comme illustré à la figure 1.4. Durant les différentes phases, les transactions d'identité doivent être gouvernées de manière cohérente. Les sous-sections suivantes vont présenter les différentes phases ainsi que la gouvernance assurant la cohérence.

### 1.3.1 Création de l'identité

La création d'une identité suit les trois étapes suivantes : l'attestation des attributs (attribute proofing), l'émission du justificatif d'identité (credential issuance) et la formation de l'identité (identity formation).

#### 1.3.1.1 Attestation des attributs

Les autorités agrées se portent garantes de la validité des attributs d'identité fournis aux destinataires de ceux-ci. Par exemple, la date de naissance est validée par les autorités locales. Certaines transactions vont requérir un attribut qui a été attesté conforme tandis que d'autres vont simplement faire confiance en la bonne foi du sujet. Le contexte de l'attestation devrait être disponible pour les destinataires afin qu'ils puissent déterminer le niveau

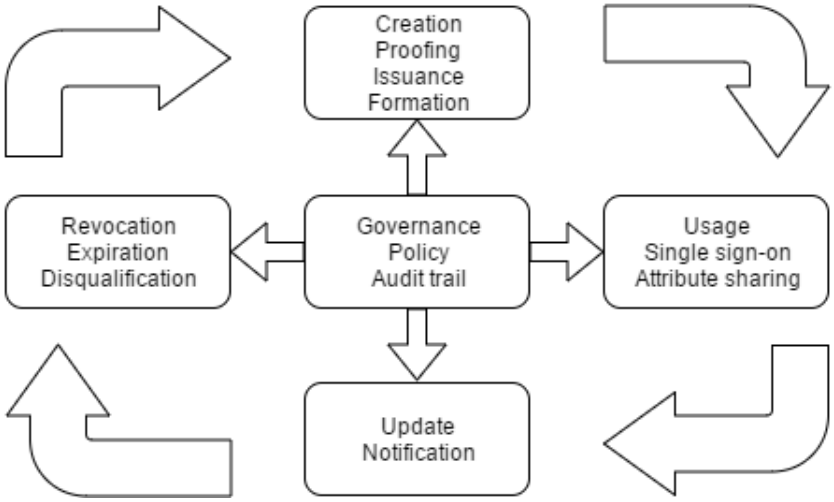


FIGURE 1.4 – Le cycle de vie de l'identité [7]

confiance à accorder à l'attribut.

#### 1.3.1.2 Émission du justificatif d'identité

Après que les attributs ont été attestés conformes, les justificatifs sont émis par les autorités (par exemple : une carte d'identité) ou par les sujets que les attributs référencent (par exemple : le choix d'un mot de passe). Les justificatifs peuvent eux prendre plusieurs formes telles qu'un certificat digital, un mot de passe, une empreinte digitale,... traduisant ainsi un niveau de confiance différent. Le contexte de l'émission du justificatif devrait être à disposition des destinataires afin qu'ils puissent déterminer le niveau de confiance à accorder au justificatif. Le contexte inclut l'émetteur du justificatif, sa date d'émission et sa date d'expiration.

#### 1.3.1.3 Formation de l'identité

Les identités sont constituées d'attributs prouvés, de justificatifs d'identités et d'identifiants assignés par des tierces parties ou les sujets eux-mêmes. Les identifiants choisis par les sujets doivent être faciles à retenir. L'espace reservé aux noms d' identifiants doit être suffisamment large pour éviter les conflits.

### 1.3.2 Utilisation de l'identité

Les identités digitales précédemment créées peuvent être utilisées pour activer ou enrichir des services. Nous allons décrire trois fonctions généralement utilisées par les services qui utilisent les identités : la communication fiable (trusted communication), le Single Sign-On (SSO) et le partage d'attributs (attributes sharing).

#### 1.3.2.1 Communication fiable

Des identités dignes de confiance sont fondamentales pour permettre des communications fiables. Dans ce genre de communication, les émetteurs et destinataires des messages

doivent être capables de découvrir, de distinguer et d'authentifier les identités du côté opposé de manière fiable. De telles communications sont nécessaires afin de permettre des transactions d'identité fiables également. Les mécanismes permettant la découverte d'identité doivent être variables et sécurisés, suite au grand nombre d'émetteurs et receveurs se trouvant sur internet. L'authentification pour les communications et les transactions doit être bien coordonnées. L'authentification mutuelle est essentielle pour éviter des attaques potentielles.

#### 1.3.2.2 Single Sign-On

Le Single Sign-On est une relation d'identité qui permet au sujet de réutiliser son authentification pour accéder à plus d'un service. L'utilisateur peut accéder à plusieurs services sans avoir besoin de s'authentifier auprès de chaque service une fois que la transaction de SSO est mise en place. L'utilisateur ne doit donc plus le besoin de maintenir un compte pour chaque service utilisé. Ce mécanisme permet de simplifier l'authentification de l'utilisateur à plusieurs services en réduisant les efforts de maintenance à fournir par les utilisateurs et fournisseurs de services.

#### 1.3.2.3 Partage d'attributs

Le partage d'attributs est une transaction qui permet aux fournisseurs de services et d'identité de partager les attributs de sujets. Les attributs sont essentiels à la personnalisation des expériences du sujet, mais celles-ci sont dispersées au travers de plusieurs fournisseurs de services provoquant redondance et inconsistance. Le partage d'attributs permet d'éviter ce phénomène en et maintenant l'intégrité des attributs dispersés dans différents services.

### 1.3.3 Modification de l'identité

Les données d'identité sont constamment mises à jour lors de leur cycle de vie. Certains justificatifs peuvent expirer et d'autres se créer. Certains attributs peuvent changer au fil du temps. Les données d'identité doivent donc être mises à jour en temps opportun afin de maintenir leur intégrité. Les changements doivent être notifiés aux parties qui stockent les données comme les Identity Provider afin que les dernières données valides soient utilisées. L'historique des changements peut être conservé afin d'être inclus et utilisé dans les journaux d'audit. Les identifiants doivent être conçus pour ne pas devoir être modifiés au cours du cycle de vie.

### 1.3.4 Révocation de l'identité

Les identités et les justificatifs doivent être révoqués s'ils deviennent obsolètes ou invalides. La révocation est très importante pour assurer la validité de l'authentification et des autorisations basées sur les données d'identité. La révocation doit être rapidement partagée parmi les destinataires des données d'identité. L'historique des révocations doit être stocké pour que cela puisse être inclus et utilisé dans les journaux d'audit.

### 1.3.5 Gouvernance de l'identité

Durant les phases décrites précédemment, les transactions d'identité doivent être gouvernées par des procédures compréhensives et stockées de manière fiable. La gouvernance

de l'identité doit être intégrée dans l'organisation du contrôle interne et doit être planifiée au même titre que le reste des activités de l'organisation. La gouvernance de l'identité est la base pour la mise en place des processus de conformité de l'organisation par rapport au respect des réglementations en matière de protection de la vie privée notamment.

#### 1.3.5.1 Procédures d'identité

Les procédures d'identités portent sur l'authentification et l'autorisation. Les procédures d'authentification définissent le niveau de confiance requis pour une certaine transaction. Les procédures d'autorisation définissent les conditions dans lesquelles le sujet est autorisé à accéder à certains services ou données. La procédure d'identité est un concept générique qui n'est pas restreint à la gouvernance d'identité, mais également définie à d'autres fins comme les procédures relatives à la protection de la vie privée entre les utilisateurs et les fournisseurs de services. Il est important que ces procédures soient exprimées et gérées de manière à ce que l'utilisateur puisse facilement les comprendre.

#### 1.3.5.2 Journaux d'audit

Tout le long du cycle de vie de l'identité, des traces doivent être conservées. Les traces incluent de l'information détaillée sur chaque transaction impliquant des identités afin de mitiger toute répudiation potentielle. Les traces étant elles-mêmes des données d'identité, elles doivent être protégées de manière sécurisée et respectueuses de la vie privée.

### 1.3.6 Cycle de vie de l'identité selon la norme ISO/IEC 24760

Le cycle de vie de l'identité selon la norme ISO/IEC 24760 [16] représenté à la figure 1.5 est une adaptation des phases présentées dans les sous-sections précédentes 1.3.1 à 1.3.5. Selon la norme ISO/IEC 24760, l'identité peut se retrouver dans cinq états différents : unknown, established, archived, active et suspended.

La signification de ces cinq états est explicitée ci-dessous.

- **Unknown :** Aucune information n'est présente dans le système et ne peut être utilisée pour identifier une entité.
- **Established :** Les informations d'identité exigées ont été vérifiées pendant le processus d'inscription. Des informations supplémentaires, par exemple un identifiant de référence a été produit, et les informations ont été enregistrées.
- **Archived :** Les informations d'identité pour une entité sont toujours présentes dans le système, bien que l'entité n'est plus active dans le domaine. Les informations archivées ne sont pas disponibles pour reconnaître l'entité sauf pendant le processus de ré-inscription. Quand l'entité se ré-inscrit, les informations archivées peuvent être utilisées pour établir une nouvelle identité pour l'entité, qui peut inclure certaines des informations archivées.
- **Active :** Les informations d'identité sont présentes dans le système, ce qui permet à l'entité d'interagir avec des services et d'utiliser les ressources disponibles dans un domaine d'application.
- **Suspended :** L'identité est présente dans le système dans le seul but d'indiquer son interdiction d'utiliser les ressources disponibles.

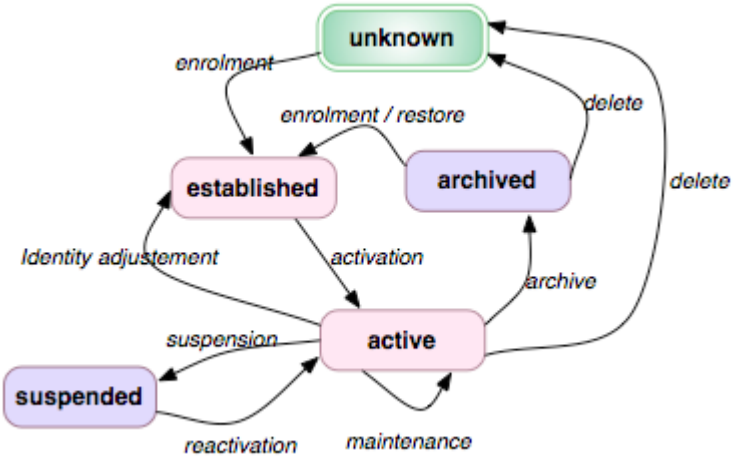


FIGURE 1.5 – Le cycle de vie de l'identité selon la norme ISO/IEC 24760 [**?**]

Les transitions entre les différents états possibles sont explicitées ci-dessous.

- **Enrolment :** Correspond à la phase de création de l'identité présentée ci-dessus à la sous-section 1.3.1.
- **Activation :** Correspond à la phase d'utilisation de l'identité présentée ci-dessus à la sous-section 1.3.2.
- **Maintenance :** Correspond à la phase de modification de l'identité présentée ci-dessus à la sous-section 1.3.3.
- **Identity adjustment :** Correspond à une phase de modification de l'identité où les nouvelles informations provoquent la ré-inscription de l'identité.
- **Suspension :** Phase de modification de l'identité la rendant temporairement indisponible pour l'utilisation. La suspension peut être réalisée en enlevant des droits d'accès exprimés dans les informations d'identité stockées.
- **Reactivation :** Phase de modification qui annule la suspension.
- **Delete :** Correspond à la phase de révocation de l'identité présentée ci-dessus à la sous-section 1.3.4.
- **Archive :** Suppression partielle d'informations d'identité du système pour une entité. Ces informations sont seulement disponibles pour le traitement statistique et peuvent seulement être accédées avec des informations supplémentaires fournies par l'entité.
- **Enrolment/restore :** Processus d'inscription, où certaines des informations d'identité utilisées comme la preuve d'identité sont obtenues du système.

La norme ISO/IEC 24700 indique également que la gestion d'identité inclut la gouvernance, les règles de gestion, les processus, les données, les technologies et les standards permettant notamment :

- d'authentifier les identités
- d'établir la provenance des informations des identités
- d'établir le lien entre les informations sur les identités et les entités
- de maintenir à jour les informations sur les identités

- d'assurer l'intégrité des informations sur les identités
- de fournir les justificatifs d'identité et les services pour faciliter l'authentification d'une entité
- d'ajuster les risques de sécurité liés au vol d'information.

Ces différents aspects font partie intégrante de la gouvernance mise en place pour contrôler l'entièreté du cycle de vie qui a été présenté à la sous-section 1.3.5

## 1.4 Assurance d'identité

L'assurance d'identité joue un rôle clé dans la gestion d'une fédération d'identité. Les fournisseurs de services doivent pouvoir connaître le niveau de confiance des informations d'identité fournies par les fournisseurs d'identité afin d'assurer des transactions fiables. Comme discuté à la section 1.3, le niveau d'assurance des informations d'identité est déterminé par la manière dont a été maintenue leur intégrité tout au long du cycle de vie. Les cadres de travail pour l'assurance d'identité ont été largement discutés par les organisations émettant les standards, dont un des standards est l'Identity Assurance Framework (IAF) développé par Liberty Alliance [6]. Dans cette section, nous allons expliciter la notion d'assurance d'identité comme défini par l'IAF.

L'IAF défini un ensemble de critères et directives qui permet d'évaluer l'assurance des informations d'identité impliquées dans des transactions d'identité. L'IAF a pour but de faciliter l'acceptation mutuelle entre les individus et les organisations en les aidant à déterminer les niveaux d'assurance. Parmi les nombreuses parties incluses dans l'IAF, les niveaux d'assurance (Assurance Level (AL)) et les critères d'évaluation des services (Service Assessment Criteria (SAC)) sont importants pour les fournisseurs de services, car ils permettent de déterminer à quel point les SP peuvent avoir confiance dans les informations fournies par les fournisseurs d'identité. Les niveaux d'assurance définissent quels sont les différents niveaux possibles et les SAC définissent ce qui doit être atteint pour obtenir chaque niveau d'assurance.

Dans l'IAF il y a quatre types de participants : les sujets, les Credential Service Provider (CSP) (ou IdP), les fournisseurs de services (SP), les assesseurs (assessor)et les accréditeurs (accreditor) comme illustré à la figure 1.6. Un sujet est une personne ou un groupe de personnes dont les identités représentent un intérêt, et qui peut déterminer quel IdP utiliser sur base du niveau d'assurance offert et du statut d'accréditation des assesseurs déterminant les niveaux d'assurance. Un CSP est une entité qui crée, fourni, maintient les justificatifs d'identité des sujets et fournit l'identité partielle des tiers. Un CSP peut être un IdP ou une entité qui fournit des justificatifs qui peuvent être utilisés par les IdPs (ou jouer les deux rôles en tant qu'IdP et en tant que fournisseur de justificatifs pour d'autre IdPs). Un SP est un fournisseur de services qui se repose sur les IdPs pour l'information d'identité. Les SPs négocient et se mettent d'accord avec les CSP sur les niveaux d'assurance à utiliser lors des transactions d'identité. Un assesseur est une entité qui évalue et détermine les niveaux d'assurance d'identité des justificatifs fournis par les CSP conformément aux SAC. Un accréditeur est une entité qui accrédite les assesseurs sur base des compétences d'évaluation et les exigences de celui-ci.

La définition des concepts clés de l'assurance d'identité est présentées ci dessous à la

figure 1.6.

- **Assurance Levels ALs** : Ils indiquent le niveau d'assurance associé à un justificatif mesuré par la nature et la qualité des technologies déployées, des processus exécutés et des procédures imposées. Les niveaux d'assurance s'étalent du plus faible (niveau 1) au plus fort (niveau 4). Les niveaux d'assurance sont résumés dans le tableau 1.7.
- **Service Assessment Criteria (SAC)** : Ils définissent la conformité des CSPs d'un point de vue opérationnel et organisationnel. Les SAC sont conçus pour déterminer objectivement le respect ou non des critères. Un ensemble de critères est défini pour chaque niveau d'assurance.
- **Assurance Assessment Scheme (AAS)** : Il décrit en détail les opérations de l'entièreté des programmes d'évaluation et de certification afin de fonctionner comme un cadre de travail utile et complet. Par exemple, l'AAS décrit l'application, la revue de suivi de conformité et les processus de révocation.
- **Assessor Qualification & Requirements (AQR)** : Ils définissent les exigences que les assesseurs des ALs doivent satisfaire afin d'être accrédités.

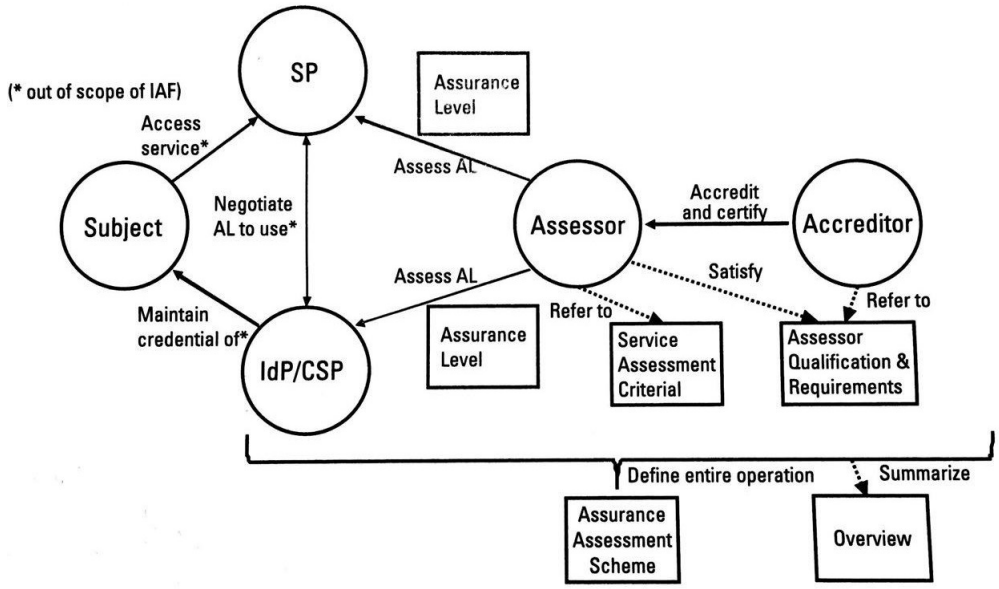


FIGURE 1.6 – Une vue d'ensemble de l'IAF [7]

Le déploiement de l'IAF doit être entièrement planifié et doit faire l'objet d'un commun accord entre IdPs, SPs, CSPs et assesseurs. Il est important de fixer des règles sur lesquelles les SPs peuvent se baser pour déterminer le niveau d'assurance requis pour chacun de leurs services. Le niveau d'assurance à utiliser doit être déterminé en fonction des dégâts potentiellement causés par l'échec des transactions ciblées. Par exemple, les transactions concernant la vente de médicaments contrôlés requièrent le niveau quatre, car son échec peut causer du tort à la santé des personnes. A l'opposé, l'inscription sur un site d'actualité en ligne requiert le niveau un, car les dégâts causés dans l'inscription sont à priori limités.

| Niveau | Description | Exemple d'implémentation | Exemple d'utilisation |
|---|---|---|---|
| 1 | Peu ou pas de confiance dans la validité de l'affirmation de l'identité | Numéros d'identification personnels (code PIN) | Enregistrement sur un site d'actualité |
| 2 | Confiance moyenne dans la validité de l'affirmation de l'identité | Simple authentification en ligne (nom d'utilisateur et mot de passe sur une connexion cryptée) | Changement d'adresse du bénéficiaire du service |
| 3 | Forte confiance dans la validité de l'affirmation de l'identité | Authentification multiple via des éléments logiciels (une combinaison de pin et de certificats électroniques) | Accès en ligne à un compte en banque |
| 4 | Très forte confiance dans la validité de l'affirmation de l'identité | Authentification multiple via des éléments hardware (carte à puce protégée par une authentification d'empreinte digitale) | Distribution de médicaments contrôlés |

FIGURE 1.7 – Niveaux d'assurance d'identité

## 1.5 Modèles de gestion de l'identité

Il existe trois modèles de gestion de l'identité communément utilisés [19] : le modèle de gestion de l'identité isolée, fédérée et centralisée. Ces trois modèles sont détaillés dans les sous-sections suivantes 1.5.1 à 1.5.2.

### 1.5.1 Gestion de l'identité isolée

Le modèle de gestion de l'identité isolée est le modèle de gestion le plus communément utilisé. Le modèle laisse les fournisseurs de services agir en tant fournisseurs de justificatifs d'identité et d'identifiant [20]. Les fournisseurs de services contrôlent l'espace des noms pour un service spécifique et allouent des identifiants aux utilisateurs. Un utilisateur reçoit un identifiant différent pour chaque service avec lequel il traite. En plus, chaque utilisateur possède un justificatif tel qu'un mot de passe associé à son identifiant. Les utilisateurs peuvent être autorisés à définir leur propre identifiant pour autant que ceux-ci soient uniques dans l'espace des noms. Ce modèle est illustré à la figure 1.8.

Cette approche fournit une gestion d'identité simple du point de vue des fournisseurs de services mais est problématique pour les utilisateurs. Le nombre de fournisseurs de services avec qui ils traitent est en constante augmentation. Les utilisateurs oublient régulièrement les mots de passe rarement utilisés des fournisseurs de services. Les mots de passe oubliés ou la crainte de les oublier créent une barrière significative au niveau de l'utilisation, ce qui résulte que les services ne peuvent pas atteindre leur total potentiel [19]. Pour les services sensibles, où la récupération du mot de passe doit être hautement sécurisée, les mots de

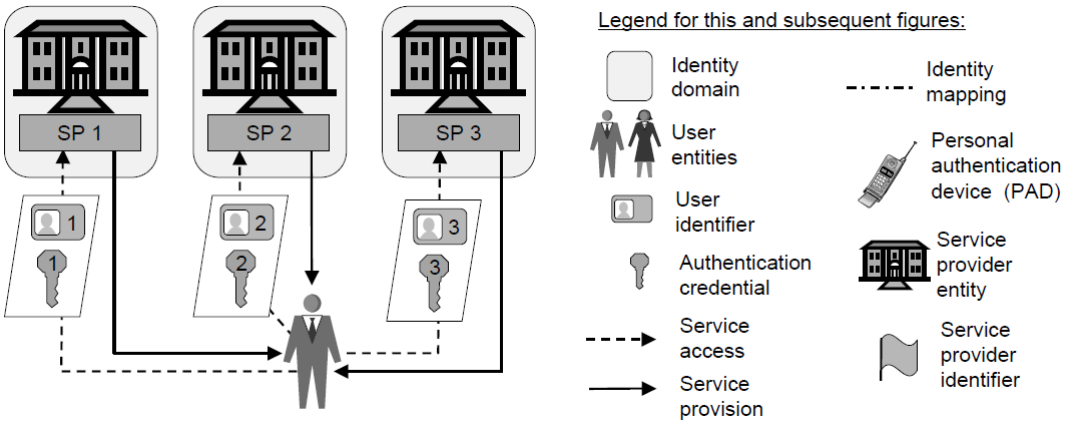


FIGURE 1.8 – Gestion de l'identité isolée [20]

passe oubliés peuvent aussi augmenter les coûts de disponibilité du service [21]. Les utilisateurs voient alors leur nombre d'identités se multiplier et cela cause une certaine fatigue des utilisateurs voulant accéder à des services fournis par des fournisseurs différents. Ceci viole le principe de facilité d'utilisation de sécurité et est illustré par les demandes de connexion multiples provenant de l'utilisateur.

La simplicité de ce modèle le rend relativement simple à comprendre et résout les problèmes de confiance impliqués. La complexité de la confiance est grandement simplifiée lorsque la même entité agit en même temps en tant que fournisseurs d'identifiants, justificatifs et services. L'utilisateur et le fournisseur de services doivent seulement se faire confiance pour un petit ensemble de besoins. Le niveau d'assurance, explicité en section 1.4, est déterminé par le fournisseur de services en fonction de son évaluation des risques et de la sensibilité des services offerts.

#### 1.5.1.1 Confiance de l'utilisateur vis-à-vis du fournisseur de services

L'utilisateur émet deux exigences de confiances vis-à-vis des fournisseurs de service.
**T1 :** Le fournisseur de service protège la vie privée du client.
**T2 :** Le fournisseur de service a implémenté des procédures et mécanismes satisfaisants d'inscription et d'authentification (du point de vue du client).

#### 1.5.1.2 Confiance du fournisseur de services vis-à-vis de l'utilisateur

Le fournisseur de service émet une seule exigence de confiance vis-à-vis de l'utilisateur.
**T3 :** L'utilisateur gère ses justificatifs d'identité avec le soin adéquat.
Cette exigence sera d'ailleurs exigée dans tous les modèles présentés.

### 1.5.2 Gestion de l'identité fédérée

Le modèle de gestion de l'identité fédérée tente de résoudre les inefficacités identifiées dans le modèle précédent. La fédération d'identité est définie comme un ensemble d'accords, standards et technologies permettant à un groupe de fournisseurs de services de reconnaître

les identifiants provenant d'autres fournisseurs de services appartenant à la fédération [20]. L'idée de base est de lier les différents identifiants, et ainsi les identités associées, entre une multitude de fournisseurs de services. Pour faire simple, on réalise un mapping de toutes les identités que l'utilisateur possède chez les fournisseurs de services de la fédération. Et ainsi permettre à l'utilisateur de n'utiliser plus qu'un seul identifiant pour s'authentifier auprès d'un service, et ainsi être directement identifié et authentifié sur les autres services de la fédération. C'est en fait une solution de Single Sign-On (SSO) (déjà décrite dans la sous-section 1.3.2.2), un identifiant isolé appartenant à une fédération devient un seul identifiant fédéré. Ce modèle est illustré à la figure 1.9.

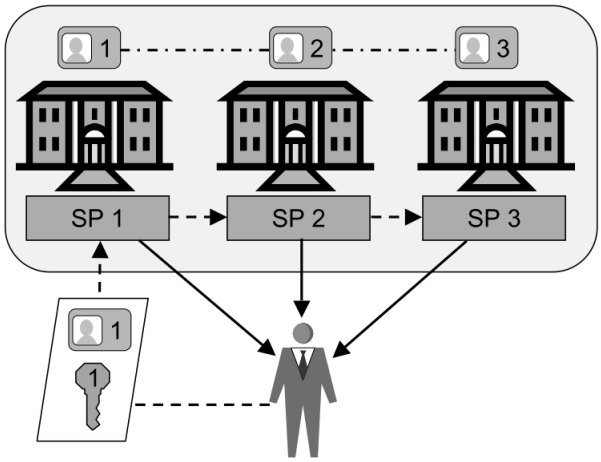


FIGURE 1.9 – Gestion de l'identité fédérée [20]

La fédération des identifiants isolés donne l'illusion à l'utilisateur qu'il n'y a plus qu'un seul identifiant qui est nécessaire pour s'authentifier auprès des services réunis dans la fédération. L'utilisateur peut toujours garder des identifiants séparés pour chaque fournisseur de services, mais il n'a pas nécessairement besoin de les connaitre ou de les posséder. Un identifiant et un justificatif sont suffisants pour accéder à tous les services de la fédération. Il reste tout de même un problème potentiel de gestion de multiples identités pour les utilisateurs, même s'il ne les utilise pas toutes. La fédération d'identité prend tout sont sens quand l'utilisateur a le désir de ne gérer qu'un seul couple d'identifiant et de justificatif [20].

Le désavantage du modèle de gestion de l'identité fédérée est qu'il crée de la complexité légale et technique. D'un point vue légal, car la question de la vie privée se pose, car les fournisseurs de services sont techniquement capables de combiner les informations personnelles grâce au mapping effectué entre les différentes identités de l'utilisateur dans la fédération. Mais d'un autre côté, cela peut également permettre d'utiliser un service anonymement, car seul le fournisseur ayant émis les identifiants utilisés connait la réelle identité de l'utilisateur [21].

#### 1.5.2.1 Confiance entre les différents fournisseurs de services de la fédération

Il y a une exigence de confiance entre tous les fournisseurs de services de la fédération.

**T4 :** L'accès au service par assertions entre fournisseurs de services à la place de l'utilisateur, ne peut avoir lieu que lorsque cela est légitimement exigé par le client.

#### 1.5.2.2 Confiance dans le mapping d'identité

Il y a une exigence de confiance à avoir vis-à-vis du mapping qui est fait entre les différents identifiants.

**T5 :** Le mapping des identités entre les fournisseurs de services doit être correct.

#### 1.5.2.3 Confiance de l'utilisateur vis-à-vis du fournisseur de service

L'exigence de confiance de l'utilisateur vis-à-vis des fournisseurs de services en plus de **T1** et **T2** est :

**T6 :** Le fournisseur de service adhère à la politique acceptant de réguler la corrélation d'information d'identité à propos d'un même utilisateur et provenant d'autres fournisseurs de services.

### 1.5.3 Gestion de l'identité centralisée

Dans les modèles de gestion de l'identité centralisée, il n'existe qu'un unique identifiant et justificatif utilisé par tous les fournisseurs de services. Dans l'article « Trust Requirements in Identity Management » [19], les auteurs proposent trois implémentations possibles qui sont l'identité commune, la méta-identité et le Single Sign-On (SSO).

#### 1.5.3.1 L'identité commune

Dans le modèle à identité commune, il n'y a qu'une entité unique et séparée qui joue le rôle de fournisseur d'identité. Tous les fournisseurs de services utilisent alors les identités fournies par ce seul fournisseur d'identités. Ce modèle est illustré à la figure 1.10 ci-dessous.

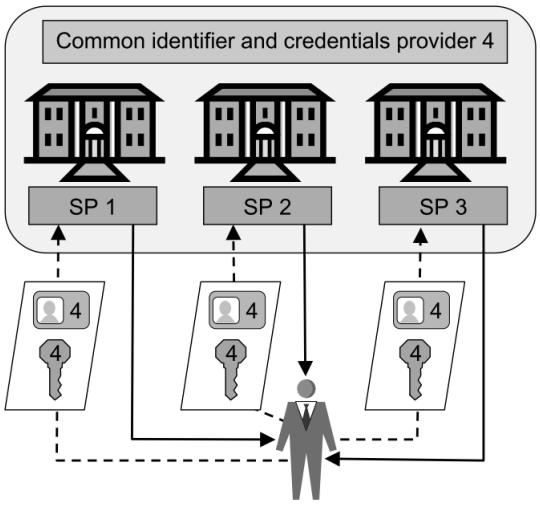


FIGURE 1.10 – Gestion de l'identité centralisée : identité commune [20]

Ce modèle peut être implémenté en utilisant une infrastructure à clés publiques (PKI). L'utilisateur peut alors utiliser les différents services à l'aide d'un seul identifiant, mais n'est pas directement authentifié sur tous les services contrairement au modèle de gestion de l'identité fédérée présenté en 1.5.2.

L'avantage de ce modèle est qu'il facile à gérer pour les utilisateurs et les fournisseurs de services. L'utilisateur n'a besoin que d'un seul identifiant. Mais il a aussi des désavantages, le principal est qu'il est pratiquement et politiquement impossible de définir et de gérer un espace de noms stable et unique pour tous les utilisateurs à travers le monde. Bien que les utilisateurs ne doivent pas exposer des informations personnelles à d'autres parties que les fournisseurs de services, l'identifiant unique commun permet aux différents fournisseurs de services de faire correspondre des informations personnelles appartenant au même utilisateur, ce qui représente une menace pour la protection de la vie privée [21]. Ce modèle est donc compliqué à mettre en place pour une utilisation à grande échelle, mais à plus petite échelle ce modèle peut fonctionner [20].

#### 1.5.3.2 La méta-identité

Les fournisseurs de services peuvent échanger de l'information à un niveau commun ou méta. On peut implémenter cela en effectuant un mapping entre les différentes identités de l'utilisateur (qu'il possède chez différents fournisseurs de services) à un méta-identifiant auquel on vient lier un justificatif d'identité. Ce modèle est illustré à la figure 1.11 ci-dessous.

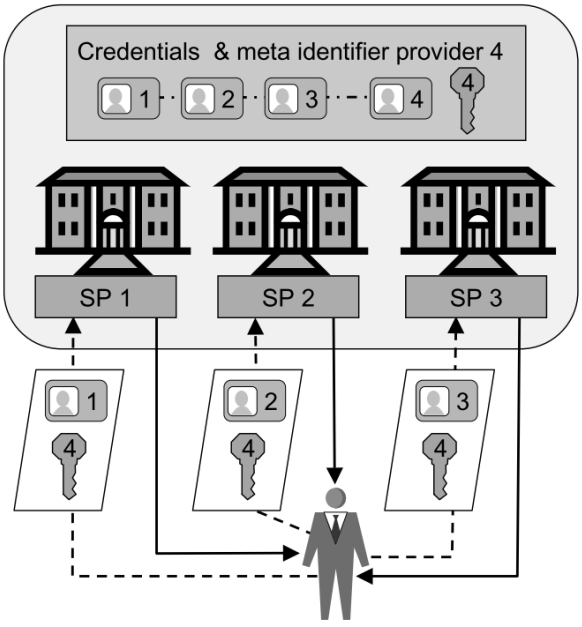


FIGURE 1.11 – Gestion de l'identité centralisée : méta-identité [20]

Cette approche est un moyen de faciliter l'intégration de différents domaines d'identité, comme lors de la fusion de plusieurs organisations. Ainsi, le modèle de méta-identité permet

à des domaines d'identité isolés de mettre en commun des informations et éviter la réplication des informations.

#### 1.5.3.3 Single Sign-On

Une extension des modèles précédents (identité commune et méta-identité) pourrait être d'autoriser les utilisateurs authentifiés par un fournisseur de services, d'être authentifiés directement par les autres fournisseurs de services. Cette solution est appelée Single Sign-On (SSO) parce que l'utilisateur ne doit s'authentifier qu'une seule fois pour accéder à tous les services. Une entité est normalement responsable pour l'allocation des identifiants et pour effectuer les authentifications. Ce modèle est illustré à la figure 1.12 ci-dessous.

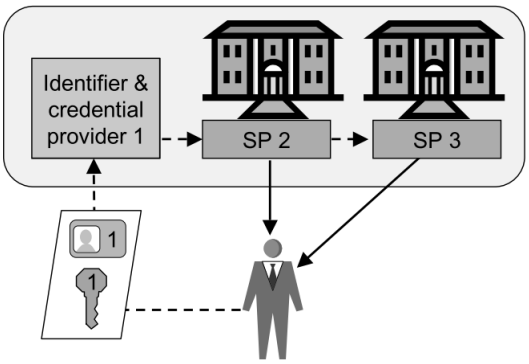


FIGURE 1.12 – Gestion de l'identité centralisée : SSO [20]

Le scénario de SSO est similaire au modèle de gestion de l'identité présenté en 1.5.2 sauf qu'il n'existe pas de mapping entre les différentes identités de l'utilisateur, car le même identifiant est utilisé par tous les fournisseurs de services [20].

L'avantage de ce modèle est de fournir une facilité d'utilisation grâce au SSO. Il convient très bien dans des réseaux fermés où différents services sont gérés par la même organisation, comme dans les universités par exemple, et utilisent la même politique. Le désavantage est que ce modèle n'est pas approprié dans les réseaux ouverts où les différents fournisseurs de services ne sont pas gouvernés par une seule politique commune. Il est en effet difficile propable que les fournisseurs de services acceptent qu'un seul fournisseur d'identité soit utilisé pour gérer l'identité et l'authentification des utilisateurs en leurs noms. Cela violerait le principe de respect de la vie privée visant à limiter l'information personnelle exposée [21].

#### 1.5.3.4 Confiance de l'utilisateur vis-à-vis du fournisseur de services

L'utilisateur a les exigences de confiance vis-à-vis des fournisseurs de services suivantes : **T1**,**T2** et **T6**.

#### 1.5.3.5 Confiance du fournisseur de services vis-à-vis de l'utilisateur

Le fournisseur de service émet une seule exigence de confiance vis-à-vis de l'utilisateur : **T3**.

#### 1.5.3.6 Confiance du fournisseur de services vis-à-vis du fournisseur de justificatifs

Il y a une exigence de confiance entre les fournisseurs de services et les fournisseurs de justificatifs.

**T7 :** Le fournisseur de justificatifs a implémenté des procédures adéquates pour l'inscription de l'utilisateur et pour l'émission des justificatifs.

## 1.6 Exigences requises des systèmes de gestion de l'identité

### 1.6.1 Exigences de confiance des modèles de gestion de l'identité

Les trois modèles de gestion de l'identité présentés précédemment en section 1.5, ont impliqué sept exigences de confiance numérotées de T1 à T7. Le tableau 1.1 présente les différents modèles vis-à-vis des exigences rencontrées. Pour rappel les sept exigences sont :

**T1 :** Le fournisseur de services protège la vie privée du client.
**T2 :** Le fournisseur de services a implémenté des procédures et mécanismes satisfaisants d'inscription et d'authentification (du point de vue du client).
**T3 :** L'utilisateur gère ses justificatifs d'identité avec le soin adéquat.
**T4 :** L'accès au service par assertions entre fournisseurs de services à la place de l'utilisateur, ne peut avoir lieu que lorsque cela est légitimement exigé par le client.
**T5 :** Le mapping des identités entre les fournisseurs de services doit être correct.
**T6 :** Le fournisseur de services adhère à la politique acceptant de réguler la corrélation d'information d'identité à propos d'un même utilisateur et provenant d'autres fournisseurs de services.
**T7 :** Le fournisseur de justificatifs a implémenté des procédures adéquates pour l'inscription de l'utilisateur et pour l'émission des justificatifs.

| | **T1** | **T2** | **T3** | **T4** | **T5** | **T6** | **T7** |
|---|---|---|---|---|---|---|---|
| Isolé | • | • | • | | | | |
| Fédéré | • | • | • | • | • | • | |
| Centralisé | • | • | • | | | • | • |

TABLE 1.1 – Comparaison des modèles de gestion de l'identité [19]

Comme nous pouvons le voir dans le tableau 1.1, le modèle de gestion de l'identité isolé requiert le moins d'exigence de confiance alors que le modèle de gestion de l'identité fédéré en demande le plus. Satisfaire les exigences de confiance est coûteux, moins il y a d'exigences requises le mieux c'est. Le coût alloué à la satisfaction des exigences de confiance doit être équilibré avec les coûts de facilité d'utilisation de la solution implémentée.

Dans le contexte de cette comparaison, il est aussi utile de faire remarquer que les modèles de gestion de l'identité fédérée et centralisée, possèdent de clairs avantages leur permettant de contre-balancer le coût induit par l'augmentation des exigences de confiance [19].

L'un des buts des modèles de gestion de l'identité fédérée et centralisée est de simplifier la gestion des identités des clients en minimisant le nombre d'identifiants et justificatifs qu'ils

doivent gérer. Ce n'est pas pour autant qu'il n'y aura plus à l'avenir qu'un seul domaine fédéré ou centralisé, il restera toujours des domaines isolés. Les utilisateurs devront donc toujours gérer plusieurs identités même si elles seront tout de même moins nombreuses.

### 1.6.2 Les sept lois de l'identité

Dans son article paru en 2005, Cameron [9] propose un ensemble de sept lois sous forme de principes auxquels doit se conformer une architecture d'un système de gestion d'identité universel et durable. Ces lois ont étés proposées, débattues et raffinées au travers d'un long et toujours ouvert dialogue sur internet [9]. Voici les sept lois proposées par Cameron.

1. **User Control and Consent :** Le système de gestion de l'identité ne doit jamais révéler l'information identifiant un utilisateur sans son consentement.

   Rien n'est aussi vital à un système de gestion d'identité que ses utilisateurs. Il doit être attirant en termes de convenance et de simplicité, mais pour résister, il doit gagner la confiance de ses utilisateurs. Cette confiance sera atteinte par une conception qui accorde à l'utilisateur un certain contrôle. Il doit protéger l'utilisateur contre les déceptions, vérifier l'identité de toutes les parties demandant de l'information et avertir l'utilisateur du but de la collecte de ses informations. C'est toujours l'utilisateur qui décidera de fournir ses informations d'identité ou non.

2. **Minimal Disclosure for a Constrained Use :** Le système de gestion de l'identité doit divulguer le moins d'informations identifiantes possibles, car c'est la solution la plus stable à long terme.

   Les systèmes de gestion d'identité doivent être conçus sur la base qu'une brèche est toujours possible, ce qui représente un risque. Pour diminuer ce risque, il faut acquérir des informations seulement si l'on en a le besoin de savoir et ne les conserver que s'il y a en le besoin. Ces deux pratiques permettent d'assurer le moins de dégâts en cas d'infraction. Cela permet également de diminuer la valeur des informations détenues et le système devient une cible moins attractive pour le vol d'identité, réduisant également le risque. L'accumulation d'informations agrège aussi le risque. Pour minimiser le risque, minimisons l'accumulation.

3. **Justifiable Parties :** Le système de gestion d'identité doit être conçu de telle façon que la divulgation d'informations identifiantes ne soit limitée qu'aux parties qui ont une place nécessaire et justifiable dans une certaine relation d'identité.

   Le système d'identité doit mettre son utilisateur au courant des parties avec qui il interagit lors du partage de ses informations. Cette exigence s'applique autant à l'utilisateur divulguant ses informations qu'au fournisseur de services qui dépend de ces informations. Chaque intervenant doit être en mesure de fournir une politique concernant l'utilisation des informations divulguées.

4. **Directed Identity :** Un système de gestion d'identité universel doit soutenir les identifiants « omnidirectionnels » pour l'utilisation par des entités publiques et des identifiants « unidirectionnels » pour l'utilisation par des entités privées, facilitant

ainsi la découverte en empêchant la sortie inutile de possibilités de corrélation.

Les entités publiques possèdent des identifiants fixes et bien connus. Ces identifiants publics sont vus comme des signaux lumineux, ils émettent leur identité à toutes les personnes qui se présente, ces signaux sont omnidirectionnels. Les individus eux décident s'ils veulent s'identifier ou non. Le système peut alors mettre une relation d'identité unidirectionnelle. Une relation d'identité unidirectionnelle avec un service différent impliquerait l'utilisation d'un identifiant sans corrélation avec le précédent. Aucune corrélation n'est donc émise qui peut être partagée entre des sites pour assembler des profils d'activités et des préférences dans des super-dossiers.

5. **Pluralism of Operators and Technologies :** Une solution d'identité universelle doit utiliser et permettre l'interopérabilité de technologies d'identité multiples exécutées par des fournisseurs d'identité multiples.

   L'idéal serait une représentation unique de l'identité, mais les nombreux contextes dans lesquels une identité est requise ne le permettraient pas. Un système unique ne peut exister, car ce qui rendrait un système idéal dans un contexte serait disqualifié dans un autre contexte. Ce n'est pas seulement le fait de disposer de plusieurs fournisseurs d'identité, mais d'avoir plusieurs systèmes de gestion d'identité qui proposent différentes fonctions potentiellement contradictoires. Un système universel doit adopter la différenciation, en reconnaissant que chacun d'entre nous est simultanément, dans des contextes différents, un citoyen, un salarié, un client, un personnage virtuel.

6. **Human Integration :** Les systèmes de gestion d'identité doivent définir l'utilisateur humain comme un composant du système distribué, intégré par des mécanismes de communication homme-machine sans équivoque offrant la protection contre des attaques d'identité.

   Nous avons réussi à sécuriser les canaux de communication entre les serveurs et les navigateurs des utilisateurs. Mais nous n'avons pas encore trouvé le moyen de sécuriser la connexion entre l'écran du navigateur et le cerveau de l'individu. C'est cette courte distance qui est soumise aux attaques. Le système d'identité doit s'étendre et intégrer l'utilisateur humain. L'échange d'informations d'identité doit être vu comme une cérémonie où l'échange est limité et où tout est déterminé.

7. **Consistent Experience across Contexts :** Le système de gestion d'identité doit garantir à ses utilisateurs une expérience simple, cohérente en permettant la séparation de contextes par de multiples opérateurs et technologies.

   Projetons-nous dans un futur dans lequel nous avons à disposition un certain nombre d'identités contextuelles. Nous pouvons nous attendre à ce que différents individus aient des usages et effectuent des combinaisons différentes de ces identités. Pour rendre cela possible, il faut transformer ces différentes identités en élément que l'utilisateur pourra voir sur son bureau, en ajouter, en supprimer, en modifier, en sélectionner et en partager. Ces différentes identités seront alors à utiliser dans un contexte bien déterminé et pourront ainsi permettre à l'utilisateur de contrôler les informations qu'il est sur le point de divulguer.

Le respect de ces lois selon Cameron peut expliquer les succès et les échecs des systèmes de gestion d'identité.

### 1.6.3 Quatre challenges

La plupart des organisations ont des difficultés à garder des traces et à contrôler tous les identifiants et mots de passe des utilisateurs dans leur système. En particulier avec les mouvements de personnel entrant et sortant ainsi que les changements de rôle constant, il est de plus en plus difficile d'être sûr qu'un utilisateur possède les droits d'accès appropriés au système.

Les systèmes de gestion de l'identité sont conçus pour aider à gérer les identifiants des utilisateurs à travers des systèmes multiples et aussi bien à fournir une façon de gérer l'accès des utilisateurs au cours de leur cycle de vie dans l'organisation. Ils sont fermement intégrés avec d'autres systèmes comme ceux qui fournissent un mécanisme de Single Sign-On. En tant que tels, ces systèmes de gestion de l'identité sont devenus extrêmement communs dans les infrastructures des technologies de l'information.

Bien que l'architecture des systèmes de gestion de l'identité soit assez simple, il y d'autres facteurs qui rendent l'installation et la gestion de tels systèmes plus difficiles. Quatre de ces facteurs sont la définition des rôles, la propagation des comptes utilisateurs, l'initialisation du système et l'intégration avec les mécanismes de Single Sign-On (SSO) [29].

#### 1.6.3.1 Définition des rôles

Les rôles sont utilisés pour agréger des autorisations en groupe faisant sens, dans le but qu'un individu puisse se voir assigner un rôle, héritant ainsi d'un ensemble d'autorisations spécifiques aux systèmes et ressources. Prenons l'exemple d'une université qui pourrait avoir défini trois rôles : les étudiants, les professeurs et les employés. Cela permet au système de créer directement de nouveaux comptes ayant accès à un ensemble spécifique de ressources. Un des problèmes qui se posent, est comment peut-on définir ces rôles tels qu'ils soient compatibles avec les rôles des systèmes sous-jacents ?

Il y a aussi un problème concernant les rôles multiples et la définition d'un rôle primaire. Par exemple, un individu qui serait à la fois étudiant et employé. Afin de décider à quelles ressources il a accès, il faut lui assigner un rôle primaire. Le changement de rôles s'avère aussi parfois être un défi, il peut signifier que l'utilisateur a des allocations de ressources complètement différentes.

#### 1.6.3.2 Propagations des comptes utilisateurs

Les fournisseurs de services dépendants d'un système de gestion de l'identité ont généralement besoin d'utiliser des comptes locaux afin que les personnes puissent s'y connecter. Cependant, nous avons besoin d'un moyen automatisé pour charger ces comptes dans ces systèmes quand le système de gestion de l'identité les crée, les désactive et supprime leurs accès ou la synchronisation devient totalement déréglée. Idéalement, le système de gestion de l'identité peuplerait directement et mettrait à jour ces comptes sur base du rôle qui leur

est assigné. Cependant, cela n'est pas toujours le cas et exige alors qu'un fichier soit envoyé périodiquement entre le système de gestion de l'identité et les systèmes qui exigent des informations locales.

#### 1.6.3.3 Initialisation du système

Lorsqu'un nouvel utilisateur est créé, il doit initialiser son compte à l'aide d'un mot de passe sécurisé. C'est une étape critique, car le nouveau justificatif permet alors d'obtenir l'accès intégral à tous les systèmes de l'organisation dont il possède les autorisations. Une méthode doit être conçue pour qu'un nouvel utilisateur puisse s'identifier au système en sécurité et ensuite créer un mot de passe difficile à déchiffrer. Une méthode souvent utilisée pour valider l'authentification d'un nouvel utilisateur est de lui demander de répondre à une question à laquelle seul l'individu est susceptible de connaitre la réponse. Une fois que cela est fait, il lui est demandé de créer un mot de passe respectant certaines règles définies par l'organisation. Plus il y aura de contraintes sur le mot de passe, plus les utilisateurs auront difficile pour inventer un mot de passe qui respecte les contraintes, cependant, il sera plus difficile de le déchiffrer.

#### 1.6.3.4 Intégration avec Single Sign-On

Le mécanisme de Single Sign-On (SSO) a déjà été explicité précédemment dans ce chapitre notamment en 1.3.2.2 et 1.5.3. Un système qui utilise une ressource externe pour authentifier l'identité d'un individu est un mécanisme de SSO. Idéalement, une fois l'authentification réalisée, l'individu n'aura à se connecter qu'une seule fois pour utiliser tous les services donc l'accès lui sont autorisés. L'utilisation d'un système de gestion de l'identité et une solution de gestion d'accès rend cela plus facile pour déployer une telle solution, car l'ensemble des mots de passe et des comptes devraient tous être synchronisés.

# Chapitre 2

# Problématiques liées à un serveur d'identité

**Sommaire**



## 2.1 Introduction

Ce chapitre est consacré aux problématiques liées à la gestion d'un serveur d'identité. Les problématiques présentées dans ce chapitre ont été identifiées durant le stage effectué à l'Université du Luxembourg. Un prototype a été développé afin d'apporter une solution à ces problèmes. Le serveur d'identité sur lequel les problématiques ont été identifiées était un Active Directory, outil fourni par la société Microsoft. Les détails concernant les solutions implémentées sont disponibles au chapitre 4.

Au cours du stage réalisé, trois problématiques ont été identifiées avec l'aide de l'administrateur de l'Active Directory. Ces trois problématiques sont les suivantes : la gestion des rôles, la gestion des mots de passe et la gestion des nouveaux utilisateurs. Elles sont présentées ci-dessous dans le reste de ce chapitre.

Ces problématiques ne couvrent évidemment pas l'ensemble de toutes les questions pouvant que l'on est en droit de se poser quant à la gestion d'un serveur d'identité, mais elles reflètent les besoins réels d'un administrateur d'un tel serveur dans une grande organisation, en l'occurrence l'Université du Luxembourg.

## 2.2 Gestion des rôles

L'augmentation croissante du nombre d'utilisateurs dont l'identité est stockée sur un serveur d'identité impose aux grandes organisations de remettre en question l'attribution des permissions aux utilisateurs. En effet, maintenir des permissions pour chaque utilisateur du système est impossible et peut même devenir la cause de problèmes de sécurité. C'est pourquoi l'existence des rôles semble logique, les rôles regroupent un ensemble de permissions et sont ensuite attribués aux utilisateurs. Les différentes permissions attribuées à l'utilisateur via les rôles, lui permettent d'accéder ou non à certaines parties du système et d'effectuer dès lors des opérations spécifiques à sa fonction. C'est ce qu'on appelle le contrôle d'accès sur base du rôle ou encore Role-Based Access Control (RBAC) en anglais. La figure 2.1 illustre ce principe.

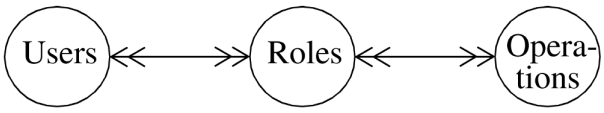


FIGURE 2.1 – Utilisateurs, rôles et opérations [11]

Les rôles correspondent généralement à des fonctions déjà présentes dans l'organisation par exemple un chef de projet ou encore à des activités de l'organisation [30]. Cette correspondance des rôles par rapport à l'entreprise les rendent plus stables du fait que les fonctions et activités d'une entreprise sont des choses qui sont amenées à se renouveler moins fréquemment que les informations liées à un employé par exemple [24].

Bien que les rôles facilitent la gestion des accès au système, la gestion des rôles ne reste cependant pas une chose aisée et la sous-section 2.2.1 va lister quelques-uns des problèmes découlant de la gestion des rôles. Ensuite, la sous-section 2.2.2 présentera des solutions déjà existantes permettant de faire face aux problèmes énoncés.

Les solutions que nous avons imaginées afin de faire face aux problèmes de la gestion des groupes sont présentées aux points 4.2 et 4.3.

### 2.2.1 Problèmes

#### 2.2.1.1 Création des rôles

Afin que le système de rôles soit optimal, les rôles doivent être créés de manière réfléchie afin qu'ils répondent de manière précise aux fonctions et aux activités de l'entreprise. La création de ces rôles n'est pas une tâche aisée. En effet, les rôles ainsi créés doivent couvrir tous les besoins et être en nombre minimal. Le nombre de rôles doit être minimal afin de faciliter la maintenance et la connaissance des rôles. Si le nombre de rôles est trop élevé, les accès fournis par ceux-ci doivent certainement se recouper. Ce recoupement n'est pas une bonne chose, car cela implique une complexité inutile, complique la maintenance et peut causer des problèmes de sécurité. L'un des défis majeurs dans l'implémentation d'un système RBAC, est la définition d'un ensemble complet et correct de rôles [30]. Ce processus connu sous le nom de role engineering a été identifié comme l'un des composant les plus coûteux dans la mise en place d'un système RBAC [27].

Plusieurs approches sont disponibles afin de découvrir les rôles qu'il faut créer [1]. Trois approches ont été identifiées et sont présentées ci-dessous.

- **Approche Top-down :** cette approche est principalement guidée par le business. Les rôles sont définis sur base des responsabilités d'une fonction donnée ou en passant en revue la structure organisationnelle de l'entreprise. Les permissions sont ainsi liées avec chaque fonction de travail. Cette approche fournit différents avantages comme le fait que les rôles soient supervisés par l'entreprise, l'alignement de rôles avec les fonctions de l'entreprise et permet la réutilisation des fonctions de l'entreprise.

- **Approche Bottom-up :** cette approche est basée sur l'exécution d'un algorithme de role-mining, concept présenté en 2.2.2.1. Les rôles sont découverts en fonction des accès décernés aux utilisateurs dans le système actuel. Cela permet aux rôles d'être normalisés et rationalisés. Les rôles sont définis pour respecter une demande spécifique ou des exigences du système d'accès.

- **Approche Hybride :** cette approche combine les deux approches précédentes. Elle exploite les rôles normalisés et rationalisés fournis par le role-mining tout en les alignant avec les différents postes de l'entreprise et donc avec l'implication de l'entreprise.

#### 2.2.1.2 Suppression des rôles

Au fil du temps et de l'évolution de l'entreprise, certains rôles deviennent obsolètes et sont donc inutilisés. Ces rôles inutilisés risquent fort d'être oubliés s'ils ne sont pas rapidement supprimés. Les problèmes liés à la suppression des rôles sont quasiment équivalents aux problèmes soulevés par la création de rôles. En effet supprimer un rôle doit être une action réfléchie. La suppression ne peut se faire qu'après avoir effectué des vérifications permettant de s'assurer de l'inutilité du rôle. Ce sont surtout ces vérifications qui posent problème. En effet, il est difficile de pouvoir assurer que le rôle que l'on souhaite supprimer n'est pas utilisé lors d'évènements spécifiques ne se produisant que très rarement. Ce rôle pourrait alors être détecté comme inutile, vu son inutilisation apparente, mais être en réalité encore actif. La suppression de rôle de telle sorte pourrait provoquer le chaos total pour le système et peut-être même nuire à l'entreprise. C'est pourquoi la suppression des rôles doit être une

tâche prise très au sérieux. Afin de maintenir le système de rôles à son niveau optimal, la suppression des rôles est une tâche inéluctable.

Une solution personnalisée a été implémentée pour résoudre ce problème de suppression de rôle, dont le détail est présenté au point 4.2. La solution proposée ne s'attaque pas directement aux rôles, mais plutôt au concept de groupe que l'on peut retrouver dans un Active Directory. Les groupes représentent un ensemble d'utilisateurs. Le groupe se voit également attribuer des droits d'accès qui sont ensuite par transitivité transmis aux individus composant ce groupe. Les groupes peuvent donc être vus comme une généralisation des rôles. C'est sur ce concept que se base la solution proposée. Les vérifications concernant l'inutilisation du groupe se font principalement en observant les attributs constituant le groupe, notamment certaines dates clés comme la date de création ou de dernière modification du groupe.

### 2.2.2 Solutions existantes

Parmi les solutions existantes, nous avons choisi d'en étudier une en particulière, il s'agit du role-mining. Cette solution est présentée ci-dessous au point 2.2.2.1.

#### 2.2.2.1 Role-mining

Cette solution est particulièrement adaptée au problème de la création des rôles. Mais peut également être utilisée pour contrer le problème de la suppression des rôles.

Le terme role-mining est utilisé dans un sens plus précis pour définir un processus de role engineering automatisé [12]. Le processus de role engineering est généralement interprété comme étant la tâche de configuration d'un système RBAC, typiquement la création et l'assignation des rôles aux utilisateurs. Le role-mining est en fait une dérivation du data-mining appliqué aux processus de role engineering. Comme présenté précédemment en 2.2.1.1, il existe différentes approches permettant de réaliser les processus de role engineering. Les deux approches top-down et bottom-up ont toutes deux leurs avantages et inconvénients. Néanmoins l'approche bottom up excelle dans le fait qu'une grande partie des processus de role engineering peuvent être automatisés [30]. Le role-mining peut dès lors être utilisé comme un outil, combiné avec une approche top-down, permettant d'identifier de potentiels rôles qui seront par la suite examinés afin de déterminer s'ils sont appropriés étant donné les fonctions et les processus de l'entreprise.

Les exigences vis-à-vis du role-mining, sont assez proches de celles relatives aux systèmes RBAC. Dans l'article « On the definition of Role Mining » [12], les auteurs identifient trois exigences concernant le role-mining.

- **Exigence n° 1 : Approvisionnement.** Une configuration RBAC doit permettre aux utilisateurs d'effectuer leurs tâches.

  D'un point de vue technique, cela signifie qu'une configuration RBAC doit fournir aux utilisateurs les privilèges dont ils ont besoin pour effectuer des actions sur les ressources qu'ils demandent.

- **Exigence n° 2 : Sécurité.** Une configuration RBAC doit fournir la sécurité.

Cela signifie que la configuration doit se conformer aux politiques de sécurité de l'entreprise pour qu'aucun utilisateur ne puisse accéder aux ressources auxquelles il n'a pas accès. En conséquence, s'il y a des erreurs dans l'assignation des accès aux utilisateurs, le processus de role-mining devrait pouvoir les détecter afin d'empêcher que ces erreurs soient migrées vers la configuration RBAC. D'une perspective plus pratique, les configurations RBAC devraient se soumettre aux exigences d'audit et simplifier les audits de sécurité. La configuration devrait, par exemple, permettre de répondre facilement aux questions comme : pourquoi x a l'accès à y ? Ceci est étroitement lié à l'exigence n° 3.

- **Exigence n° 3 : Facilité de maintenance.** Une configuration RBAC doit rester aussi facile que possible à maintenir.

  La facilité de maintenance diminue les coûts d'administration et aide les administrateurs à travailler avec le système. Ici, trois aspects principaux entrent en jeu : minimalisme (un petit système est toujours plus facile à maintenir), interprétation (un système compréhensible est plus facile à maintenir) et capacité de généralisation (qui facilite l'ajout de nouveaux utilisateurs).

Pour résumer, on peut définir grossièrement le processus de role-mining comme étant le processus qui pour un ensemble d'utilisateurs et un ensemble de permissions produira comme résultat un ensemble de rôles, une assignation utilisateur-rôle, une assignation rôle-permission tout cela cohérent avec l'assignation utilisateur-permission déjà existante et tout en minimisant le nombre de rôle.

Le problème majeur du role-mining reste l'absence de consensus entre les différentes approches. Il n'existe pas de métrique universelle permettant de comparer les différents algorithmes sur la qualité des résultats produits ni de consensus sur l'utilisation d'un algorithme plutôt que d'un autre [12].

## 2.3 Gestion des mots de passe

L'identité seule n'est pas suffisante pour de fournir un accès sécurisé aux multiples services du système d'information de l'entreprise. De nos jours, l'accès et l'authentification à la plupart des services se font via l'utilisation du couple nom d'utilisateur et mot de passe. Si bien que ce dernier est devenu l'unique barrière entre un agresseur potentiel et les informations de la victime ou de l'organisation [25]. Le problème est que de nombreuses études ont montré qu'un pourcentage élevé de mots de passe peut être deviné facilement [8]. Sachant cela, il est dès lors important pour une organisation désireuse d'éviter une attaque par vol de mot de passe d'imposer à ses utilisateurs d'utiliser des mots de passe dits forts, c'est-à-dire plus difficiles à deviner et rendant par la même occasion les attaques plus difficiles.

La plupart des utilisateurs utilisent des services divers et variés pour lesquels ils possèdent à chaque fois une combinaison potentiellement différente nom d'utilisateur et mot de passe. Dans les meilleurs cas, qui sont assez rares, les mots de passe sont tous différents. Le problème est qu'il difficile pour l'utilisateur de retenir des dizaines de mots de passe différents et donc l'utilisateur choisit généralement un mot de passe identique ou similaire pour l'ensemble des services qu'il utilise [13].

La sous-section 2.3.1 va lister quelques-uns des problèmes découlant de la gestion des mots de passe. Ensuite, la sous-section 2.3.2 présentera des solutions déjà existantes aux problèmes énoncés.

La solution que nous avons imaginée afin de faire face aux problèmes de la gestion des mots de passe est présentée au point 4.5.

### 2.3.1 Problèmes

#### 2.3.1.1 Initialisation du mot de passe

Lorsque de nouveaux utilisateurs sont ajoutés au système, ils doivent initialiser leurs comptes avec un mot de passe sécurisé. Le mot de passe initial doit être de préférence fourni sur un canal sûr. Lorsque ce mot de passe initial est fourni par l'administrateur du système ou lorsqu'il est communiqué sur un canal non confidentiel, il doit être changé dès la première connexion de l'utilisateur. L'administrateur qui a fourni un mot de passe sur un canal non sûr doit avoir une vigilance plus soutenue afin de s'assurer que le mot de passe n'est pas utilisé par un tiers [5]. Cette étape est une étape critique dans le processus de création du mot de passe, sachant que la nouvelle combinaison nom d'utilisateur et mot de passe fournira un accès intégral à tous les systèmes de l'organisation auxquels l'utilisateur a accès [29]. Une méthode doit être conçue pour qu'un nouvel utilisateur puisse s'identifier au système de manière sécurisée, pour ensuite être amené à créer un mot de passe beaucoup plus compliqué à trouver.

Une méthode souvent utilisée pour valider une identité est de demander à l'utilisateur de répondre à une question dont la réponse est une donnée personnelle et dont seul l'utilisateur peut connaitre la réponse. Cette donnée provient généralement du système d'information de l'entreprise et peut inclure d'autres identifiants uniques et mots-clés que l'utilisateur est censé connaitre. Une fois qu'ils ont réalisé cette étape, les utilisateurs sont invités à définir un mot de passe fort.

Un mot de passe fort est un mot de passe qui doit être difficile à décoder et qui possède généralement au moins huit caractères incluant des majuscules et minuscules, des caractères spéciaux ainsi que des numéros. D'autres exigences peuvent également être ajoutées pour renforcer le mot de passe, comme ne pas contenir de mots de plus de trois caractères qui se trouvent dans le dictionnaire. Plus il y a de contraintes sur le mot de passe, au plus il sera compliqué pour les utilisateurs de construire un mot de passe répondant à toutes les contraintes, mais en contrepartie, le mot de passe sera difficile à casser.

#### 2.3.1.2 Renouvellement du mot de passe

Les mots de passe doivent avoir une date de validité maximale. À partir de cette date, l'utilisateur ne doit plus pouvoir s'authentifier sur le système si le mot de passe n'a pas été changé. Ceci permet de s'assurer qu'un mot de passe découvert par un utilisateur mal intentionné, ne sera pas utilisable indéfiniment dans le temps [5].

Il est important de garder à l'esprit qu'un mot de passe n'est pas inviolable dans le temps. C'est pour cette raison qu'il est nécessaire de changer régulièrement son mot de passe et qu'il est important de ne pas utiliser le même mot de passe pour tous les services vers lesquels on

se connecte. L'agence nationale française de la sécurité des systèmes d'informations préconise même une fréquence de renouvellement de nonante jours pour les systèmes contenant des données sensibles [5].

Une solution personnalisée concernant le renouvellement du mot de passe a été implémentée, et est présentée au point 4.5. La solution que nous avons imaginée consiste à effectuer des rappels de plus en plus réguliers aux utilisateurs dont le mot de passe est proche de la date d'expiration, les invitant ainsi à en changer.

### 2.3.2 Solutions existantes

Parmi les solutions existantes, nous avons choisi d'en étudier deux en particulier à savoir le Single Sign-On et la définition d'une politique de gestion des mots de passe. Ces solutions sont présentées ci-dessous.

#### 2.3.2.1 Single Sign-On

Pour faire simple, la solution de Single Sign-On (SSO) permet à l'utilisateur de ne devoir s'authentifier qu'une et une seule fois pour obtenir l'accès à l'entièreté des services et applications de l'organisation. Cette solution a été présentée en détail précédemment au point 1.5.3.3.

Un des gros avantages du SSO, est qu'il permet de réduire le nombre de mots de passe utilisés par l'utilisateur. Par la même occasion, puisque l'utilisateur ne doit plus retenir qu'un seul mot de passe, celui-ci peut être plus complexe, être donc plus résistant aux attaques et donc augmenter la sécurité du système d'information de l'organisation. L'utilisation d'un mot de passe unique permet également aux utilisateurs de ne plus perdre de temps à essayer de retrouver les différents mots de passe pour accéder à diverses applications puisqu'ils sont automatiquement authentifiés auprès de tous les services et applications disponibles.

En revanche, un des revers de cette solution facilitant grandement l'authentification des utilisateurs est le risque potentiel d'intrusion en cas d'attaque réussie. En effet, si un individu mal intentionné parvient à casser les informations d'identification d'un utilisateur, le malfrat se verra offrir un accès à tous les différents services et applications de l'organisation. Pour recourir à ce problème, des méthodes d'authentification fortes devraient être idéalement combinées avec d'autres systèmes comme l'utilisation de cartes à puce ou autres systèmes.

#### 2.3.2.2 Politique de gestion des mots de passe

Les mots de passe sont souvent la seule protection d'une station de travail. Il est donc indispensable de mettre en œuvre une politique de gestion des mots de passe intégrée à la politique de sécurité des systèmes d'information. Cette politique de gestion de mots de passe devra être à la fois technique et organisationnelle [5].

C'est impératif pour les organisations de mettre en place et d'appliquer une politique de gestion des mots de passe. Les politiques de gestion expliquent quelles sont les règles à mettre en place et ce qu'il est attendu des utilisateurs. Mettre en place une politique de gestion fournit également un cadre de travail pour faire respecter les règles imposées sur le mot de passe d'une façon consistante [26].

Plusieurs critères peuvent être définis et mis en œuvre dans de nombreux systèmes pour s'assurer de la qualité des mots de passe. Ces critères sont, par exemple : une longueur minimale obligatoire prédéfinie, l'impossibilité de réutiliser les n derniers mots de passe, le nombre de tentatives possibles avant verrouillage de compte, la manière de déverrouiller un compte qui a été bloqué précédemment. Pour éviter les dénis de service liés au blocage de tous les comptes sur un système d'information, il peut être intéressant que le déblocage des comptes se fasse de manière automatique après un certain délai, la mise en place d'une veille automatique avec un déblocage par saisie du mot de passe.

Les politiques de gestion des mots de passe doivent être aussi fortes que nécessaire, sans plus. Forcer les utilisateurs à respecter une politique de gestion qui veut faire face au risque théorique maximal est un coût énorme, non seulement en termes monétaires, mais aussi sur le plan de la ressource la plus précieuse pour une organisation, à savoir la bonne volonté de ses membres [15].

## 2.4 Gestion des nouveaux utilisateurs

L'ajout de nouveaux utilisateurs est une tâche quasiment quotidienne pour une grande organisation. Bien qu'en apparence cette tâche semble être relativement aisée, elle peut se révéler bien plus complexe qu'elle n'y paraît. Ce n'est pas le fait de créer de nouveaux utilisateurs qui pose problème, cette manipulation est elle tout à fait aisée et est une tâche connue par l'administrateur. Ce qui pose problème se trouve plus du côté de la configuration des attributs des nouveaux utilisateurs. En effet, l'administrateur ne connait rien des nouveaux utilisateurs qu'il est en train de créer et il lui est donc difficile de lui attribuer les bons attributs dont notamment les droits d'accès spécifiques à leurs fonctions et compétences dans l'entreprise. Ces connaissances sont plutôt en possession du service du personnel ayant embauché les nouveaux utilisateurs, mais ceux-ci ne sont évidemment pas autorisés à effectuer les manipulations permettant la création des nouveaux comptes d'utilisateurs.

Un problème majeur que nous avons relevé concernant les nouveaux utilisateurs est présenté dans la sous-section 2.4.1 suivante. Ensuite la sous-section 2.4.2 présentera des solutions déjà existantes permettant de faire face au problème énoncé.

### 2.4.1 Problèmes

#### 2.4.1.1 Attribution des droits d'accès

L'attribution des droits d'accès aux nouveaux utilisateurs peut se révéler être un véritable dilemme pour l'administrateur du système. En effet, ce dernier n'étant pas un membre des ressources humaines, il est incapable de connaitre les besoins d'accès de ces nouveaux utilisateurs. Généralement, le nouvel utilisateur reçoit les accès pour le strict minimum et doit introduire une demande auprès de l'administrateur à chaque fois qu'il désire pouvoir avoir accès à des données ou services auxquels il ne possède pas encore l'accès. Les milliers voire millions d'utilisateurs que peuvent compter certains environnements rendent cette tâche terrifiante [4].

Cette démarche de demande d'accès au fil du temps dans l'entreprise pour le nouvel utilisateur représente une perte de temps pour lui et l'administrateur ainsi qu'une perte économique pour l'entreprise. L'utilisateur n'ayant pas accès aux données nécessaires pour

effectuer correctement ses tâches ne peut qu'attendre qu'on lui autorise l'accès aux données utiles. C'est pourquoi nous avons désiré construire une solution qui permet de suggérer automatiquement à l'administrateur les potentiels droits d'accès dont le nouvel utilisateur pourrait avoir besoin dans ses fonctions quotidiennes. L'administrateur n'aurait donc plus qu'à demander une vérification des suggestions proposées auprès du responsable du nouvel utilisateur. Le nouvel utilisateur aurait lui accès à potentiellement toutes les données et services dont il a besoin pour effectuer ses tâches quotidiennes, sans en avoir fait expressément la demande.

Une solution personnalisée concernant le renouvellement du mot de passe a été implémentée, et présentée au point 4.4. Cette solution fonctionne sur base des résultats produits par une autre solution implémentée qui se réfère à la gestion des groupes et qui est présentée au point 4.3. Comme pour la solution présentée concernant la suppression des rôles, la solution d'aide à l'attribution des droits d'accès ne propose pas directement d'attribuer des rôles aux nouveaux utilisateurs, mais présente à plutôt à l'administrateur des suggestions d'appartenance à des groupes.

### 2.4.2 Solutions existantes

Il n'existe pas de nombreuses solutions concernant le problème d'assignation des droits d'accès aux nouveaux utilisateurs. Néanmoins, une solution existante consistant à modifier le modèle RBAC est présentée ci-dessous au point 2.4.2.1.

#### 2.4.2.1 Rule-Based RBAC

Dans l'article « A Model for Attribute-Based User-Role Assignment » [4], les auteurs proposent une solution permettant d'automatiser l'attribution des rôles. Cette solution n'est pas exclusivement réservée pour de nouveaux utilisateurs, mais peut également être utilisée pour des utilisateurs déjà existants. Il s'agit en réalité d'une modification du modèle RBAC, présenté en 2.2, qui n'est plus exécuté sur base de rôle, mais bien sur base de règle. Les auteurs ont donc logiquement dénommé leur nouveau modèle Rule-Based RBAC.

Dans ce modèle, les entreprises définissent un ensemble de règles qui sont déclenchées pour assigner automatiquement des rôles aux utilisateurs. Ces règles prennent en compte les attributs des clients qui sont exprimés dans le langage fourni par le modèle et toute sorte de contraintes sur l'utilisation des rôles.

La figure 2.2 ci-dessous, représente une illustration du modèle Rule-Based RBAC. La figure nous montre que les utilisateurs ont des relations plusieurs à plusieurs avec les valeurs d'attributs. De plus, ils possèdent aussi des relations plusieurs à plusieurs implicites cette fois avec des expressions d'attributs. Un utilisateur peut dès lors avoir une ou plusieurs expressions d'attributs dépendantes des informations qu'il fournit. De même, deux utilisateurs ou plus peuvent fournir des expressions d'attributs identiques. Une expression d'attributs spécifique correspond à un rôle ou plusieurs rôles. La figure 2.2 montre aussi qu'un rôle peut être lié hiérarchiquement à un rôle au plus. Cette figure nous illustre aussi qu'un rôle peut correspondre à une expression d'attribut ou plus.

Dans le but d'assigner le rôle ou les rôles spécifiés par la règle à un utilisateur, les vérifications suivantes doivent être respectées : l'utilisateur doit fournir des attributs qui satisfont

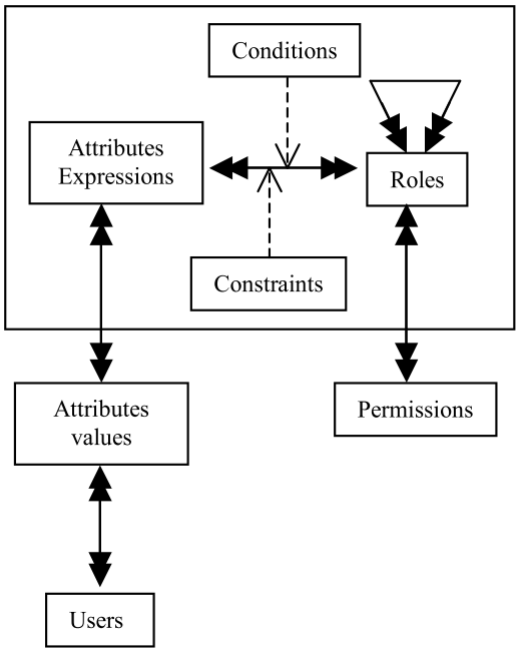


FIGURE 2.2 – Le modèle Rule-Based RBAC [4]

les expressions d'attributs et toutes les contraintes doivent être également satisfaites. De plus, les conditions permettent une révocation dynamique de rôles si une condition requise par la règle ne peut plus être satisfaite.

Le lecteur est invité à se rediriger vers l'article [4] s'il désire davantage de détails et d'informations concernant cette solution.

# Chapitre 3

# Proactive Computing

## Sommaire



## 3.1 Concepts

Nous allons commencer ce chapitre en proposant une définition du terme proactif. Lequel vient de pro et actif, et qui est en opposition du terme réactif. La définition officielle nous apprend que le terme proactif ce dit de quelque chose ou de quelqu'un qui est orienté vers le futur, qui agit en prévoyant les éventuels problèmes et leur solution et non en se contentant de réagir aux situations qui se présentent [33].

Appliqué au domaine informatique, cela veut donc dire que contrairement aux systèmes réactifs, les systèmes proactifs n'exigent plus d'interactions de la part de l'utilisateur pour prendre une décision et effectuer une action. Les systèmes dits proactifs sont donc capables de prendre des décisions automatiquement en fonction des événements qu'ils détectent. C'est une façon de rendre un système partiellement ou totalement autonome. Le concept de proactive computing a été invoqué pour la première fois par David Tennenhouse dans son article paru en l'an 2000 [28].

Pendant les quarante années précédentes, les recherches dans le monde de la technologie de l'information se sont concentrées sur une informatique interactive, guidée par la vision de Joseph Carl Robnett Licklider, vision centrée sur l'humain. En parallèle, l'industrie informatique avançait tête baissée vers le point de rupture homme/machine/réseau c'est-à-dire le point où le nombre d'ordinateurs interactifs en réseau surpasserait le nombre d'hommes dans le monde.

Il est maintenant temps pour l'informatique de définir ses nouveaux objectifs qui propulseront la société au-delà de l'informatique interactive et du point de rupture homme/machine/réseau évoqué ci-dessus. Il est essentiel de réfléchir à un monde où le nombre d'ordinateurs surpassera le nombre de personnes vivant sur la planète. Nous devons considérer ce que ces ordinateurs «d'excès» feront et proposer un agenda de recherche pouvant mener vers une amélioration de la productivité et la qualité de vie de l'homme.

Bien que manquant de clarté sur ce que ces quarante prochaines années de calcul peuvent apporter, les premières étapes dans l'élaboration d'un nouvel agenda sont de réexaminer fondamentalement la frontière entre le monde physique et virtuel, d'effectuer des changements dans les constantes de temps auxquelles les calculs sont appliqués et un mouvement de la vision de l'informatique centrée sur l'homme vers une informatique supervisée par l'homme, ou voire même non supervisée. Se concentrer sur ces trois champs de recherche permettra l'activation d'un nouveau mode de calcul que Tennenhouse appelle le *Proactive Computing*. Tennenhouse décrit trois nouvelles branches dans lesquelles il est nécessaire de mener des activités de recherche :

- **Getting physical :** Les systèmes proactifs seront connectés au monde qui les entoure, utilisant des capteurs et des déclencheurs capables de surveiller et de conceptualiser leur environnement physique. La recherche dans ce domaine explorera le couplage des systèmes connectés avec leurs environnements.

- **Getting real :** Les ordinateurs proactifs devront répondre de manière routinière à des stimulus externes à une vitesse plus élevée que celle de l'homme. La recherche dans ce domaine doit combler le vide entre la théorie du contrôle et l'informatique.

- **Getting out :** L'informatique interactive place délibérément l'être humain dans la boucle de contrôle. Cependant, rétrécir les constantes de temps et les nombres élevés d'opérations, demande de la recherche dans les modes d'opération proactifs dans lesquels les hommes sont au-dessus de la boucle de contrôle et deviennent des superviseurs.

Il y a deux raisons pour lesquelles nous devons réaffecter certaines de nos ressources intellectuelles à l'informatique proactive. La vaste majorité des nouveaux ordinateurs seront

proactifs et seront la principale source d'information dans un avenir proche.

Bien que la consommation d'ordinateurs interactifs reste élevée, la proportion d'ordinateurs embarqués ne fait qu'augmenter et excède celle de l'ordinateur traditionnel poussant donc la communauté de recherche à investir une large fraction de ses ressources intellectuelles dans cette nouvelle ère qu'est le proactif computing. En faisant cela, nous suivrons également les données informatiques, qui bougent d'un environnement dans lequel nos sources d'information sont largement fournies par l'intermédiaire de l'homme vers un environnement où l'ordinateur découvre directement les informations provenant du monde qui l'entoure. En fait, la principale raison d'avoir plus d'appareils que d'hommes, et de les avoir de manière distribuée est leur intime connectivité avec le monde physique et la source d'information sans fin qu'ils sont capables de produire.

Bien que le déploiement de l'Internet se soit concentré sur l'extension de la portée géographique du réseau, le nombre de nœuds sur le réseau peut être augmenté considérablement en descendant jusqu'au moindre dispositif embarqué de chaque emplacement géographique.

Les recherches qui ont fondé le proactif computing, se basent sur la notion d'ubiquitous computing présenté dans un article de 1991 [32], dans lequel Mark Weiser prévoit que l'informatique deviendrait tellement omniprésente que les individus ne seraient plus conscients de chacune de ses applications. Comme indiqué dans la figure 3.1, l'espace de l'informatique ubiquitaire peut être divisé en deux dimensions, une ayant un rapport avec le degré auquel les applications sont centrées sur la bureautique et l'autre ayant un rapport avec le degré auquel le nœud informatique et ses interfaces sont en interaction avec les êtres humains contre l'interaction avec le reste de son environnement.

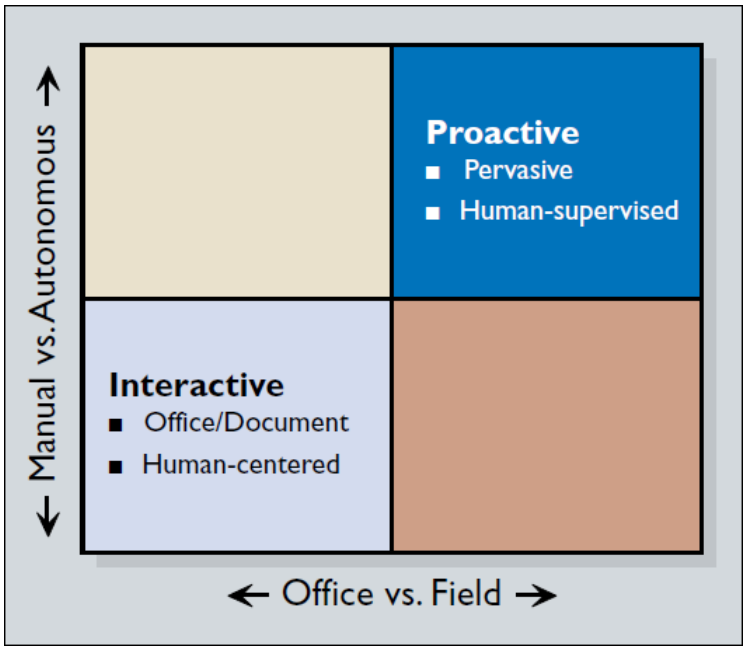


FIGURE 3.1 – Les quatre quadrants de l'informatique ubiquitaire [28]

Bien que la vision de Weiser était assez radicale, peu de chercheurs ont rompu avec l'accent traditionnellement mis sur l'interaction humaine. Au lieu de cela, la plupart des recherches ont été limitées au quadrant inférieur gauche de la figure 3.1. Le proactive computing propose un agenda pour l'exploration du quadrant supérieur droit de l'informatique ubiquitaire et l'expansion de nos horizons intellectuels au-delà du domaine interactif. En bref, il est temps pour les chercheurs de déclarer la victoire sur l'automatisation de bureau.

## 3.2 Deux visions

Nous allons maintenant nous intéresser à la relation entre le proactive computing et l'autonomic computing, qui considèrent tous deux la conception des systèmes qui sont au-delà de la portée de notre infrastructure informatique existante. L'autonomic computing, comme décrit par le manifeste d'IBM, est une déclaration claire des difficultés et des défis faisant face à l'industrie informatique d'aujourd'hui. Particulièrement, l'autonomic computing aborde le problème de la gestion de la complexité et se concentre plûtot sur la maintenance des systèmes. La recherche d'Intel explore l'avenir de l'informatique qui chevauche l'autonomic computing, mais explore aussi les nouveaux domaines d'application qui exigent des principes que nous appelons le proactive computing, demandant la transition des systèmes interactifs d'aujourd'hui aux environnements proactifs qui prévoient nos besoins et actent en notre nom. Deux grandes entreprises que sont Intel et IBM avec respectivement le Proactive Computing [28] et l'Autonomic Computing [18] ont exploré ces nouveaux domaines de recherche. Ces deux approches ont chacune leur spécificité, mais elles tentent toutes les deux d'apporter une solution aux problèmes qui limitent la croissance des systèmes informatiques d'aujourd'hui [31].

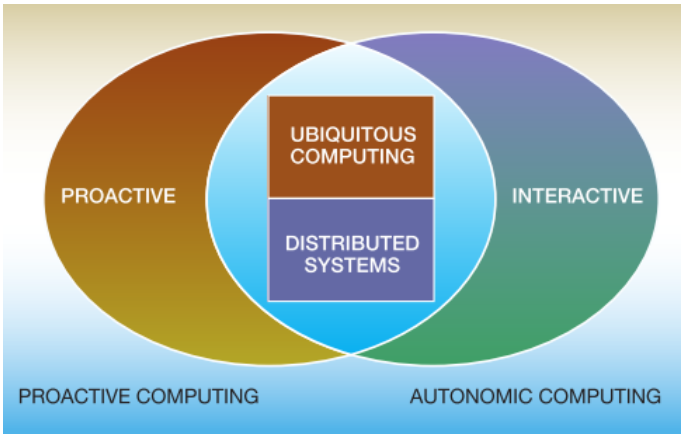


FIGURE 3.2 – La relation entre les différents paradigmes [31]

Comme illustré à la figure 3.2, le recouvrement intellectuel entre l'autonomic et le proactive computing est bien présent. Ces deux paradigmes que sont l'autonomic et le proactive computing, nous fournissent des outils pour faire avancer la conception de systèmes qui touchent à une large gamme de nouveaux domaines.

### 3.2.1 IBM : Autonomic Computing

La maintenance et les opérations des systèmes informatiques actuels deviennent de plus en plus complexes, la gestion de tels systèmes devient un véritable casse-tête pour les administrateurs. Pour prévenir ce problème grandissant et auquel le monde actuel sera de plus en plus confronté, certains ont commencé à rêver à des systèmes plus automatisés autant dans leur déploiement que dans leur maintenance. Une innovation majeure dans cette idée d'automatiser la maintenance des systèmes informatiques provient de l'entreprise IBM. Avec IBM, vient le terme Autonomic Computing caractérisant la notion de systèmes informatiques capables de s'adapter à des changements internes et externes avec une intervention minimale de l'être humain [22]. Le terme autonomic est choisi délibérément avec une connotation biologique. En effet l'autonomic computing a comme source d'inspiration le système nerveux autonome humain, capable de s'adapter rapidement à toutes sortes de stimulus [14].

L'essence de l'autonomic computing est le self-management, le but est de libérer les administrateurs système du détail des opérations et maintenance du système permettant ainsi de fournir aux utilisateurs des performances maximales vingt-quatre heures sur vingt-quatre et sept jours sur sept [18]. Le manifesto d'IBM à propos de l'autonomic computing, énonce fréquemment quatre propriétés de self management qui sont le self-configuration, self-optimization, self-healing et self-protection et qui sont présentées ci-dessous du point 3.2.1.1 au point 3.2.1.4. La figure 3.3 présente une comparaison entre l'informatique actuelle et l'informatique autonome (autonomic computing) par rapport aux quatre propriétés de self-management.

| Concept | Current computing | Autonomic computing |
|---|---|---|
| Self-configuration | Corporate data centers have multiple vendors and platforms. Installing, configuring, and integrating systems is time consuming and error prone. | Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly. |
| Self-optimization | Systems have hundreds of manually set, nonlinear tuning parameters, and their number increases with each release. | Components and systems continually seek opportunities to improve their own performance and efficiency. |
| Self-healing | Problem determination in large, complex systems can take a team of programmers weeks. | System automatically detects, diagnoses, and repairs localized software and hardware problems. |
| Self-protection | Detection of and recovery from attacks and cascading failures is manual. | System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent systemwide failures. |

FIGURE 3.3 – Comparaison entre l'informatique actuelle et autonome [18]

#### 3.2.1.1 Self-configuration

Les systèmes doivent être capables de s'adapter automatiquement aux changements dynamiques d'environnement [3]. Quand un nouveau composant est introduit dans le système, il s'incorpore homogènement et le reste du système s'adapte à sa présence. Comme une nouvelle cellule dans le corps ou une nouvelle personne dans une population [18].

Les systèmes doivent être conçus pour fournir cette propriété en tant que fonctionnalité du système. Cette fonctionnalité permettrait ainsi d'ajouter des fonctions dynamiquement à l'entreprise avec un minimum d'interactions humaines.

L'idée fondamentale de la propriété de self-configuration des systèmes est que pendant la conception on parle plutôt de ce que l'on désire du système plutôt que la manière avec laquelle on va le réaliser. La propriété de configuration est l'activateur pour les propriétés de self-optimization, self-healing et self-protection présentées ci-dessous.

#### 3.2.1.2 Self-optimization

Les systèmes autonomes chercheront continuellement des façons d'améliorer leur opération, identifiant et saisissant des occasions de se rendre plus efficaces dans leur performance ou dans leur coût. En analogie avec le corps humain, les muscles deviennent plus forts par l'exercice et le cerveau modifie son circuit pendant l'apprentissage. Les systèmes autonomes contrôleront également l'expérience, accorderont leurs propres paramètres et apprendront à faire des choix appropriés concernant le fait de conserver des fonctions ou de les externaliser. Ils chercheront proactivement à améliorer leurs fonctions en trouvant, vérifiant et appliquant les dernières mises à jour disponibles [18].

#### 3.2.1.3 Self-healing

Les systèmes autonomes doivent être conscients des problèmes potentiels et doivent avoir la capacité de se reconfigurer afin de continuer à fonctionner sans problèmes [14].

Pour qu'un système possède la propriété de self-healing, il doit être capable de se guérir d'un composant défectueux premièrement en détectant et en isolant le composant défectueux, en le déconnectant, le réparant et en réintroduisant le composant réparé ou remplacé sans aucune perturbation apparente. Les systèmes devront prévoir les problèmes et prendre des mesures pour empêcher que les erreurs aient un impact sur les applications. La propriété de self-healing a pour but de minimiser les interruptions pour permettre la disponibilité maximale des services offerts par le système [3].

#### 3.2.1.4 Self-protection

Malgré l'existence de pare-feu et d'outils de détection d'intrusion, les individus doivent à présent décider comment protéger les systèmes des attaques malicieuses et des échecs en cascade [18].

Les systèmes autonomes doivent anticiper, détecter, identifier et se protéger d'attaques provenant de toutes parts. Les systèmes avec la propriété de self-protecting doivent avoir la capacité de définir et de gérer les accès utilisateurs aux ressources de l'entreprise, de détecter des intrusions et de prévenir quand elles se produisent, fournir des backups et rétablir les capacités des ressources comme à l'original [3].

En analogie avec le corps humain, la propriété de self-protection peut être comparée au système immunitaire. Une implémentation parfaite de cette propriété ne requiert aucune intervention humaine ou connaissance [22].

### 3.2.2 Intel : Proactive Computing

Le Proactive Computing comme le défini Tennenhouse dans son article [28], supporte les mêmes buts que l'Autonomic Computing [18], mais en plus essaye de définir comment les différents systèmes informatiques pourront être utilisés dans le futur. Cette vision du Proactive Computing veut permettre l'activation de systèmes avec des réseaux de milliers d'ordinateurs par personne. La conception des systèmes proactifs est guidée par sept principes [31] qui sont :

- Se connecter avec le monde réel
- Interconnexion profonde des réseaux
- Macro traitement
- Accepter l'incertitude
- Anticipation
- Fermer la boucle de contrôle
- Rendre les systèmes personnels

Ces sept principes sont regroupés dans trois grands thèmes exposés précédemment qui sont Getting physical, real and out. Une attention particulière est placée sur les systèmes supervisés par l'être humain plutôt que des systèmes contrôlés par l'être humain ou complètement automatiques. De nos jours, de plus en plus de personnes possèdent plus d'un ordinateur. Ce ratio ne cesse d'augmenter au fil du temps. Cette augmentation incessante d'ordinateurs exige des solutions pour permettre à l'humain de sortir de la boucle de contrôle et de s'élever ainsi au rôle de superviseur. Le Proactive Computing se concentre donc sur des opérations où l'humain est le superviseur, des systèmes dans lesquels il reste en dehors de la boucle de contrôle autant que possible sauf en cas de décisions critiques.

## 3.3 Description du système proactif utilisé

Le proactive engine développé par l'équipe de recherche et de développement du professeur Zampunieris de l'Université du Luxembourg a été initialement conçu pour superviser une plateforme d'e-learning dans le but d'enrichir l'expérience de l'apprentissage en ligne [34]. Le proactive engine a été développé pour ajouter de la proactivité aux systèmes existants. L'idée de ce moteur est d'avoir l'accès à l'état des systèmes via leur base de données utilisant un adaptateur (wrapper) de base de données. De ce fait, il peut facilement être ajouté à différentes sortes différentes de systèmes qui utilisent des bases de données afin de stocker leurs états.

En interne, le proactive engine est un Rule-running system (RRS). En général, chaque tâche à exécuter est modélisée dans une règle (Rule). Les règles peuvent être répétitives, dans ce cas elles ont une fonction de monitoring et sont alors dans l'attente de certains types d'événement. Après l'occurrence d'un tel événement, d'autres règles peuvent alors être générées, qui elles sont alors responsables de prendre des actions ou autres. Les règles ainsi générées sont alors gérées par un composant, appelé QueueManager, qui exécute les règles dans la file d'attente une par une.

Une propriété très importante des systèmes proactifs est que le moteur proactif peut réagir à l'occurrence ou à l'absence d'événements sans connaître l'ordre exact des événements.

### 3.3.1 Composants

Le proactive engine développé par l'Université du Luxembourg est donc un Rule-running system (RRS). Ses différents composants sont listés ci-dessous :

- Deux files First In First Out : currentQueue et nextQueue.
  currentQueue contient les règles qui vont être exécutées au cours de l'itération courante et nextQueue va contenir les règles qui ont été générées au cours de l'itération courante. Â la fin de l'itération nextQueue devient currentQueue et nextQueue devient vide.

- Trois paramètres :
  - F : la fréquence des itérations
  - D : le temps minimum entre deux itérations (latence)
  - N : nombre maximum de règles exécutées lors d'une itération

- Une base de données propre au proactive engine où sont stockés les deux files, les trois paramètres et la définition des règles.

Les différentes définitions présentées ci-dessous vont décrire les différents composants du moteur.

#### 3.3.1.1 Les règles

Les règles sont des classes Java, respectant une certaine forme et constituées d'un nombre quelconque de paramètres et dont l'exécution se déroule en cinq étapes. Ces cinq étapes ont chacune un rôle différent [34].

1. **Data Acquisition :** Étape d'acquisition des données importantes à l'exécution de la règle et qui serviront dans les étapes suivantes. On récupère l'état du système supervisé via l'adaptateur de base de données mis en place mais aussi via des paramètres de la règle ou encore d'autres sources de données comme des capteurs par exemple. Autrement dit, on récupère des données propres au contexte d'utilisation.

2. **Activation Guards :** C'est un ensemble de tests et de conditions que les données récoltées dans l'étape précédente doivent satisfaire et qui vont déterminer si les étapes n° 3 et n° 4 s'exécuteront. Si les activation guards ne sont pas satisfaites, ces étapes seront tout simplement ignorées. Il faut noter que l'étape n° 5 s'exécutera dans tous les cas. Il y a une variable booléenne locale nommée «activated» et automatiquement définie avec le résultat des activation guards.

3. **Conditions :** L'objectif de la partie Conditions est d'évaluer le contexte dans un plus grand détail que les activation guards. Si toutes les conditions sont satisfaites, l'étape n° 4 Actions de la règle sera débloquée et s'exécutera.

4. **Actions :** Liste d'instructions à exécuter si les activation guards et conditions sont vérifiées. Une action peut être l'envoi d'un message ou la modification de l'état du système.

5. **Rule Generation :** Cette étape est exécutée indépendamment du fait que les parties Activation guards et Conditions soient vérifiées ou pas. Dans cette étape, la règle peut créer d'autres règles qui seront exécutées par la suite. Une règle peut dans certains cas également se cloner afin de ne pas mourir.

Les règles sont déclenchées par des évènements, typiquement des changements d'état. La non-détection d'un évènement est aussi considérée comme un évènement et peut donc être détectée par le moteur. Durant une itération du RRS, les règles sont exécutées une par une. L'algorithme d'exécution d'une règle est présenté ci-après à la figure 3.4.

1. **repeat for** each data acquisition DA
   (a) **perform** DA
   (b) **if** error **then**
   **raise** exception on system manager console and **go** to step 7
   **else**
   **create** new local variable and initialize it with the result of DA
2. **create** new local Boolean variable 'activated' initialized to false
3. **repeat for** each activation guard test AG
   (a) evaluate AG
   (b) **if** result == false **then go** to step 6
   **else if** AG == last activation guard test
   **then** activated = true
4. **repeat for** each conditions test C
   (a) evaluate C
   (b) **if** result == false **then go** to step 6
5. **repeat for** each action instruction A
   (a) **perform** A
   (b) **if** error **then raise** exception on system manager console and **go** to step 7
6. **repeat for** each rule generation R
   (a) **perform** R
   (b) **insert** newly generated rule as the last rule of the system
7. **delete** all local variables
8. **discard** rule from the system

FIGURE 3.4 – L'algorithme d'exécution d'une règle [34]

#### 3.3.1.2 Les scénarios

Un scénario est un ensemble de règles qui visent à effectuer une tâche spécifique. Une règle est une partie du scénario qui exécute des actions spécifiques lorsque certaines conditions sont rencontrées. Un scénario peut être vu comme une fonctionnalité et les règles qui le composent comme les différents modules que l'on assemble afin de réaliser les tâches induites par la fonctionnalité espérée.

#### 3.3.1.3 Les files

Deux files First In First Out, *currentQueue* et *nextQueue* sont indispensables au bon fonctionnement du proactive engine. La première file *currentQueue* contient les règles qui seront exécutées lors de l'itération en cours et la deuxième file contient les règles générées par les règles de l'itération courante. À la fin de l'itération *nextQueue* devient *currentQueue* tandis que *nextQueue* est quant à elle réinitialisée à nul.

#### 3.3.1.4 L'itération

Une itération est un cycle du workflow du moteur. Selon le paramètre N choisit, un nombre fixe de règles est exécuté par itération. Lors d'une itération, les règles sont exécutées une par une selon leur ordre d'apparition dans la file *currentQueue*.

#### 3.3.1.5 Le QueueManager

Le *QueueManager* est l'unité de contrôle du moteur qui gère les deux files. Chaque itération consiste en trois étapes. Pour commencer, la file *currentQueue* est enregistrée dans la base de données pour des raisons de backup. Ensuite, les règles de cette file sont dépilées et exécutées une par une. En exécutant les règles, de nouvelles règles peuvent être générées et sont donc ajoutées dans la file *nextQueue*. À la fin de l'exécution, le temps d'exécution et le nombre de règles sont enregistrés dans la base de données pour raisons statistiques. Les règles de la file *nextQueue* sont ajoutées à la file *currentQueue* tandis que la file *nextQueue* est vidée.

Il y a deux terminaisons possibles pour le moteur, soit il ne reste plus aucune règle dans les files, soit un signal de fin lui est envoyé, ce signal lui parvient alors via la base de données. Dans ce deuxième cas, les règles non encore exécutées sont sauvées dans la base de données afin de pouvoir être exécutées ultérieurement si besoin est.

Le moteur proactif n'accède pas seulement à la base de données du système monitoré mais possède également sa propre base de données. Le moteur est codé en Java et utilise les technologies XML et Hibernate. Cette deuxième base de données, qui elle est réservée au moteur est multifonctionnelle. Toutes les informations nécessaires à la bonne exécution du moteur y sont stockées comme les paramètres ou les signaux. Mais elle peut également stocker des données qui sont utilisées par des règles, des messages ou encore servir de backup en cas d'erreur.

## 3.4 Conception de système autonome avec l'aide du Proactive Computing

Cette section a pour but de présenter comment un système proactif peut être utilisé comme base pour ajouter une certaine autonomie à un système déjà existant. Le système proactif utilisé ici est le Proactive Engine (PE) présenté en section 3.3. Il a été démontré que le PE peut prendre le contrôle de tâches permettant ainsi aux systèmes déjà existants de recevoir les propriétés manquantes, présentées du point 3.2.1.1 au point 3.2.1.4, leur permettant ainsi de devenir des systèmes autonomes. Le but ici n'étant pas l'amélioration du PE, mais de l'utiliser en tant qu'une unité centrale de contrôle d'un système autonome.

Construire ou rendre un système autonome implique qu'une unité de contrôle va prendre le rôle de l'administrateur du système ou d'un membre de l'équipe informatique. Ce travail inclut de la maintenance comme effectuer des mises à jour ou des sauvegardes. Mais ce travail peut également jouer sur la sécurité et la protection du système contre de potentielles attaques. Le proactive engine qui constitue cette unité centrale devient donc le responsable afin de contrôler, gérer et protéger l'entièreté du système.

Dans ce contexte, il est très important que le système soit self-aware, c'est-à-dire que le proactive engine qui sera ajouté au système est capable de savoir exactement dans quel état le système se trouve et est capable de surveiller tous les processus prenant place dans le système.

De plus, l'unité de contrôle est responsable du fait que le système reste dans un état correct lui permettant d'être capable d'effectuer les tâches qui lui sont demandées. Cela implique qu'un but global a été défini et vers lequel le système devra être mené.

### 3.4.1 Avantages

Dans notre approche, nous avons choisi d'utiliser le proactive engine comme base d'unité de contrôle centrale. On peut utiliser le PE sans que cela implique de différences soit dans une méthode top-down c'est-à-dire que le système a été conçu pour être autonome, soit dans une méthode bottom-up c'est-à-dire un système auquel on vient greffer le PE dans le but de le rendre autonome. Cela rend le proactive engine très flexible et peut donc être adapté sans trop de difficultés à de nombreux systèmes.

Le proactive engine est indépendant du système existant et peut donc être ajouté et modifié sans causer l'interruption du système existant. L'interaction avec le système existant ne se fait que par sa base de données. Cette connexion avec la base de données est essentielle afin que le moteur proactive puisse connaitre l'état du système sur lequel il va être ajouté. Le proactive engine a été développé en Java et est donc disponible sur toutes les plateformes sur lesquelles il est installé. Il est donc possible d'utiliser les milliers de librairies que propose ce langage.

Les règles et les scénarios exécutés par le proactive engine peuvent être développés pour les besoins spécifiques d'un système existant. Pour rendre un système autonome, cela implique que le système implémente au moins une des propriétés exposées du point 3.2.1.1 au point 3.2.1.4. Ces propriétés peuvent être implémentées et gérées complètement par le moteur si le système existant n'implémentait aucune de ces propriétés. Si le système existant offrait déjà des fonctionnalités autonomes, le système autonome et le proactive engine peuvent travailler de pair afin de rendre le système existant totalement autonome.

La découpe en règles et en scénarios permet d'offrir des fonctionnalités adaptées et personnalisées aux besoins des systèmes existants. Cette découpe permet de rendre l'utilisation du proactive engine très flexible. Elle permet également d'ajouter des fonctionnalités par la suite suivant l'évolution des besoins. Ces nouvelles fonctionnalités peuvent être facilement codées sous forme de règles et intégrer par la suite au proactive engine sans avoir d'impacts sur le système existant.

### 3.4.2 Inconvénients

L'approche explicitée ci-dessus ne conviendra pas forcément au mieux pour certains systèmes ayant des besoins bien spécifiques, notamment les systèmes existants ne disposant pas de source d'information dans laquelle le système peut stocker son état.

Le proactive engine ayant été conçu initialement pour être ajouté à une plateforme d'e-learning, ce dernier n'offre pas forcément les fonctionnalités dont on aurait besoin directement, mais nécessaires. Dans la situation actuelle et dans certains cas il faut donc trouver certaines parades afin d'obtenir le résultat escompté. Cela peut entrainer une certaine complexification du proactive engine s'opposante avec sa facilité d'utilisation dans sa version d'origine.

### 3.4.3 Exemples de systèmes réalisés à l'aide du Proactive Engine

La liste d'exemples de systèmes ne se veut évidemment pas exhaustive et présente deux projets développés avec l'aide du proactive engine présenté précédemment en 3.3.

#### 3.4.3.1 Proactive Learning Management System

Ce premier exemple est à l'origine de la création du proactive engine à savoir le fait de rendre une plateforme d'e-learning existante en partie autonome [34]. Cette initiative avait donc pour but d'améliorer l'expérience de l'utilisateur de la plateforme en essayant de réintroduire un certain côté humain grâce à l'ajout de certaines fonctionnalités. Il a été conçu pour aider les utilisateurs à mieux interagir dans un environnement éducationnel, en fournissant des analyses programmables, automatiques et continues des interactions de l'utilisateur tout en ajoutant des actions appropriées choisies par le système proactive engine lui-même.

#### 3.4.3.2 Silentmeet

Dans ce deuxième exemple, le proactive engine n'a pas été ajouté par après, mais le projet a été construit directement à l'aide du proactive engine. Il s'agit en fait d'une application mobile permettant d'organiser des réunions de manière proactive [23]. Chaque participant potentiel doit avoir préalablement avoir installé l'application sur son téléphone. Ensuite l'organisateur de la réunion invite d'autres utilisateurs et choisit une plage horaire. L'application se charge ensuite de trouver le moment idéal pour tout le monde, en se basant sur les agendas de chacun et sans qu'aucune intervention ne soit nécessaire à part la création de la réunion. De plus, l'application est capable de mettre en silencieux la sonnerie du téléphone des personnes des participants à la réunion.

# Chapitre 4

# Implémentation des solutions

## Sommaire

## 4.1 Introduction

Les solutions qui vont être présentées ci-dessous, font partie intégrantes des réalisations entreprises lors du stage effectué à l'Université du Luxembourg lors du premier quadrimestre de l'année académique 2015-2016. Le but de ce stage était de développer un prototype d'aide à la gestion d'un serveur d'identité en utilisant le Proactive Engine (PE), développé par l'équipe du professeur Zampunieris.

### 4.1.1 Composant du système

Différents composants ont été utilisés pour réaliser le prototype. En voici la liste exacte.

- Le Proactive Engine présenté en détail au point 3.3. C'est le composant sur lequel repose toute la solution.
- Un serveur d'identité de type Active Directory déployé dans l'environnement de l'Université du Luxembourg et vendu par la firme Microsoft.
- Deux bases de données MySQL. Une est utile au stockage des données nécessaires au bon fonctionnement du Proactive Engine et l'autre sert à stocker les résultats produits.

### 4.1.2 Définitions des objectifs

Trois objectifs ont été définis en début de stage d'un commun accord avec le gestionnaire de l'Active Directory (AD) afin de subvenir à des besoins réels. Ces trois objectifs sont explicités ci-dessous.

- **Aide à la gestion de groupes :** Les groupes d'un AD, permettent de regrouper des utilisateurs afin de leur accorder certains droits. En effet, les groupes se voient attribuer des droits. De ce fait, les utilisateurs faisant partie d'un groupe héritent des droits du groupe. C'est donc un moyen permettant au gestionnaire de distribuer à l'utilisateur ses droits. La gestion des groupes est une tâche complexe, car elle requiert la connaissance de l'entièreté des groupes ce qui devient de plus en plus complexe

en fonction de l'évolution du nombre de groupes. C'est pourquoi nous avons décidé de fournir au gestionnaire, une fonctionnalité qui lui permettrant de faciliter cette gestion. Cet objectif étant assez conséquent, il a été séparé en deux sous-scénarios.

  - **Détection des groupes inutiles :** Le nombre croissant de groupes ne fait qu'augmenter la complexité de la maintenance liée à la gestion des groupes. C'est pourquoi contrôler cette expansion nous a semblé être évident. Pour cela, l'objectif est de fournir au gestionnaire une liste des groupes potentiellement inutiles. L'implémentation de cet objectif est présentée en 4.2.

  - **Création de profils :** La création de profils de groupe va de pair avec l'aide à l'attribution des groupes. C'est sur base des résultats que cette fonctionnalité va produire, les profils de groupe et que les suggestions vont se faire. Ces profils de groupe seront construits tant sur base des attributs du groupe lui-même que sur les attributs des membres qui composent ce groupe. Le résultat de ce scénario sera donc un ensemble de profils. Un profil n'est pas spécifique à un seul groupe, mais correspond à un ensemble de groupes. Les détails de l'implémentation de cet objectif sont présentés en 4.3.

- **Aide à l'attribution de groupes :** Lorsqu'un nouvel utilisateur est créé, le gestionnaire a la lourde tâche d'affecter l'utilisateur à un ou plusieurs groupes. Cela permettra à l'utilisateur d'avoir accès aux ressources nécessaires afin de pouvoir effectuer son travail correctement. L'attribution de groupe n'est pas une chose aisée, c'est pourquoi nous avons décidé d'intégrer à notre prototype une fonctionnalité de suggestion. Cette fonctionnalité qui se base sur les profils de groupes et les caractéristiques du nouvel utilisateur fournit au gestionnaire une liste de groupes auxquels il faut potentiellement ajouter l'utilisateur. Les détails de l'implémentation de cet objectif sont présentés en 4.4.

- **Aide à la gestion des mots de passe :** Les mots de passe représentent avec l'identifiant de l'utilisateur le moyen d'accès au système. De ce fait l'identifiant étant fixé, le mot de passe est la seule partie variable de ce couple. Son importance est donc capitale à la sécurité du système. Une politique de renouvellement des mots de passe peut donc être mise en place afin de minimiser les risques en changeant les mots de passe régulièrement. L'objectif de cette fonctionnalité va donc être de détecter les mots de passe en fin de vie et de prévenir l'utilisateur afin qu'il le mette. Les détails de ce scénario sont présentés en 4.5.

Au total, ce sont quatre scénarios proactifs qui ont été implémentés et déployés pour parvenir à réaliser les objectifs fixés. Nous avons imaginé notre solution pour que l'administrateur reste maître de son système et que les décisions lui reviennent. Toutefois, modifier ce prototype pour rendre le système totalement autonome est bien évidemment possible sans que cela ne pose de grandes difficultés.

## 4.2 Aide à la gestion des groupes : détection des groupes inutiles

### 4.2.1 Description

Ce scénario apporte une solution à la problématique liée à la gestion des rôles dans un système de gestion d'identité et plus précisément la question de la suppression de rôle. Ce qu'on appelle les rôles, peut être matérialisé par le concept de groupe dans un Active Directory.

L'idée derrière ce scénario est assez simple. On récupère la liste des groupes présents sur l'Active Directory, on parcourt cette liste et pour chaque groupe, on effectue des vérifications par rapport à quatre critères. Pour chaque groupe, une évaluation est disponible et est stockée dans une table temporaire. Une fois l'entièreté de la liste parcourue, il ne reste qu'à extraire les groupes ayant une évaluation négative, à compiler cela et en faire part au gestionnaire. Nous avons décidé des critères suivants afin de détecter les groupes potentiellement inutiles. Ils sont au nombre de quatre.

- Le groupe ne contient plus aucun membre
- 70% ou plus des membres du groupe possèdent un compte désactivé ou expiré
- Aucune modification n'a été apportée au groupe depuis au moins cinq ans
- La date du dernier mail envoyé via la mailing liste du groupe remonte à plus d'un an

Nous avons également mis en place une liste d'exclusion qui permet d'éviter la détection des groupes utilisés notamment pour des raisons techniques. Cette liste d'exclusion est conceptualisée par une table présente dans la base de données utilisée par le moteur proactif. La figure 4.1 présente l'ensemble des règles utilisées ainsi que l'enchainement de celles-ci.

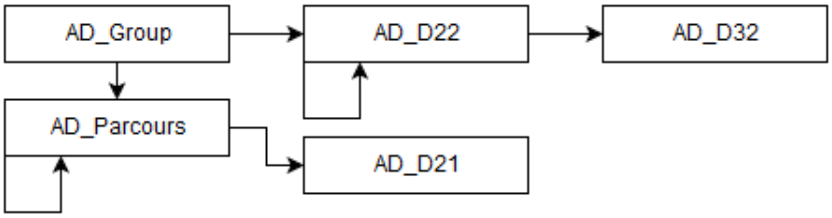


FIGURE 4.1 – Scénario de détection des groupes inutiles

### 4.2.2 Implémentation

#### 4.2.2.1 Règle AD_Group

- **Paramètres :**
  listGroup : la liste des groupes présents dans l'Active Directory

- **Description :**
  Cette règle est la règle d'initialisation du scénario. Elle récupère la liste des groupes

de l'Active Directory, crée la table nécessaire au stockage des résultats et pour terminer, elle crée la règle nécessaire au parcours de la liste.

- **Pseudo code :**

```
dataAcquisition() :
 this.listGroup = ((AbstractActiveDirWrapper) dataNativeSystem).search("OU=
     Groups,DC=uni,DC=lux","(objectClass=group)");

activationGuards() :
 return true;

conditions() :
 return true;

actions() :
 ((AbstractActiveDirWrapper) dataNativeSystem).createTableGroup();

rulesGeneration() :
 createRule(new AD_Parcours(this.listGroup.iterator(),0));
 createRule(new AD_D22(this.listGroup.size()));
 return true;
```

#### 4.2.2.2 Règle AD_Parcours

- **Paramètres :**
  listGroup : un itérateur sur la liste des groupes

- **Description :**
  Tant que la liste des groupes n'est pas vide, cette règle itère cinq groupes par cinq groupes sur les éléments de la liste de groupes et crée une règle AD_D21 pour chaque groupe.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return this.listGroup.hasNext();

conditions() :
 return true;

actions(): /

rulesGeneration() {
 if(super.getActivated()){
  for(int j = 0; j<5; j++){
   if(this.listGroup.hasNext()){
    SearchResult x = this.listGroup.next();
    createRule(new AD_D21(x));
   }
  }
  createRule(new AD_Parcours(this.listGroup,i+5));
 }
 return true;
```

#### 4.2.2.3 Règle AD_D22

- **Paramètres :**
  size : la taille de la liste des groupes
  count : le nombre de lignes déjà présentes dans la table créée par la règle AD_Group.

- **Description :**
  Cette règle est la règle qui fournit les résultats de l'analyse et qui envoie le mail récapitulatif au gestionnaire. Tant que la table qui contient les résultats n'atteint pas le nombre de groupes présents dans l'Active Directory, la règle se clone. Sinon, elle créée une règle AD_32.

- **Pseudo code :**

```
1  dataAcquisition() :
2   this.count = ((AbstractActiveDirWrapper) dataNativeSystem).countLines("
       Group_delete");
3
4  activationGuards() :
5   return (this.count == this.size);
6
7  conditions() :
8   return true;
9
10 actions() :
11  new Mail("siu.ProactiveGroup@uni.lu","Suppression",((AbstractActiveDirWrapper
       ) dataNativeSystem).getGroupDelete()).send();
12
13 rulesGeneration() :
14  if(getActivated()){
15     createRule(new AD_D32("Group_delete"));
16    }
17    else
18     createRule(new AD_D22(this.size));
19    return true;
```

#### 4.2.2.4 Règle AD_D32

- **Paramètres :**
  tableName : le nom donné à la table que l'on souhaite supprimer

- **Description :**
  Cette règle supprime la table dont on lui fournit le nom, en l'occurrence la table temporaire qui a été créée par la règle AD_Group.

- **Pseudo code :**

```
1  dataAcquisition() : /
2
3  activationGuards() :
4   return true;
5
6  conditions() :
7   return true;
8
9  actions() :
```

```
10  ((AbstractActiveDirWrapper) dataNativeSystem).dropTable(this.tableName);
11
12 rulesGeneration() :
13  return false;
```

#### 4.2.2.5 Règle AD_D21

- **Paramètres :**
  group : le groupe dont les informations vont être récoltées.
  groupName : le Distinguish Name du groupe
  listMember : la liste des membres dont le compte est actif
  listMemberDisabled : la liste des membres dont le compte est désactivé
  listMemberExpired : la liste des membres dont le compte est expiré
  totalMembers : le nombre total de membres y compris ceux dont les comptes sont désactivés ou expirés
  whenChanged : la date de dernière modification du groupe
  mail : la liste de distribution du groupe

- **Description :**
  Cette règle récupère les informations utiles afin de déterminer si le groupe est susceptible de pouvoir être supprimé et en réalise l'évaluation. Elle insère les résultats dans la table créée par la règle AD_Group.

- **Pseudo code :**

```
1  dataAcquisition() :
2   SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmSS");
3   sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
4   try {
5    this.groupName = this.group.getAttributes().get("distinguishedName").get().
         toString();
6    this.whenChanged = sdf.parse(this.group.getAttributes().get("whenChanged").
         get().toString()).getTime();
7    this.mail = this.group.getAttributes().get("mail").get().toString();
8   } catch (Exception e) {
9    this.mail = "";
10  }
11  this.listMember = ((AbstractActiveDirWrapper) dataNativeSystem).search("OU=
        UNI-Users,DC=uni,DC=lux","(&(objectClass=person)(memberOf="+this.
        groupName+")(!(userAccountControl:1.2.840.113556.1.4.803:=2)))");
12  this.listMemberDisabled = ((AbstractActiveDirWrapper) dataNativeSystem).
        search("OU=UNI-Users,DC=uni,DC=lux","(&(objectClass=person)(memberOf="+
        this.groupName+")(userAccountControl:1.2.840.113556.1.4.803:=2))");
13  this.listMemberDisabled.addAll(((AbstractActiveDirWrapper) dataNativeSystem).
        search("OU=UNI-DisabledUsers,DC=uni,DC=lux","(&(objectClass=person)(
        memberOf="+this.groupName+"))"));
14  this.listMemberAccountExpires = ((AbstractActiveDirWrapper) dataNativeSystem)
        .search("OU=UNI-Users,DC=uni,DC=lux","(&(objectClass=person)(memberOf="+
        this.groupName+")(accountExpires<="+Long.toString(new Date().getTime())+"
        )(!(accountExpires=0)))");
15  this.totalMembers = listMember.size() + listMemberDisabled.size() +
        listMemberAccountExpires.size();
16 }
17
18 activationGuards() :
19  return true;
```

```
20
21 conditions() :
22  return true;
23
24 actions() :
25  GregorianCalendar cal = new GregorianCalendar();
26  cal.add(Calendar.YEAR, -5);
27  if(this.listGroup.size()!= 0 ){
28   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        false, "");
29  }
30  else if(this.totalMembers == 0) {
31   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        true, "No Members");
32  }
33  else if((((double) this.listMemberDisabled.size()+(double) this.
      listMemberAccountExpires.size())/ (double) this.totalMembers) >= 0.7) {
34   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        true, (double) this.listMemberDisabled.size()/(double) this.totalMembers
        *100+"% of members disabled or have an expired account");
35  }
36  else if(this.whenChanged < cal.getTimeInMillis()){
37   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        true, "No modification since "+new Date(this.whenChanged));
38  }
39  else if(!this.mail.equals("") && false){
40   //DO SOMETHING
41   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        true, "No mail sent since "+new Date());
42  }
43  else {
44   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.groupName,
        false, "");
45  }
46
47 rulesGeneration() :
48  return false;
```

### 4.2.3 Résultat

Le résultat de l'exécution de ce scénario est un mail récapitulatif envoyé au gestionnaire du système de gestion d'identité et regroupant les informations suivantes :

- Le nombre de groupes détectés comme potentiellement inutiles
- Le Distinguish Name des groupes ayant été détectés comme potentiellement inutiles
- La raison pour laquelle le groupe a été détecté comme potentiellement inutile. Cette raison se base sur les quatre critères que nous avons établis précédemment.
- Le script powershell permettant au gestionnaire du système de supprimer le groupe inutile.

Un exemple de mail envoyé est illustré à la figure 4.2.

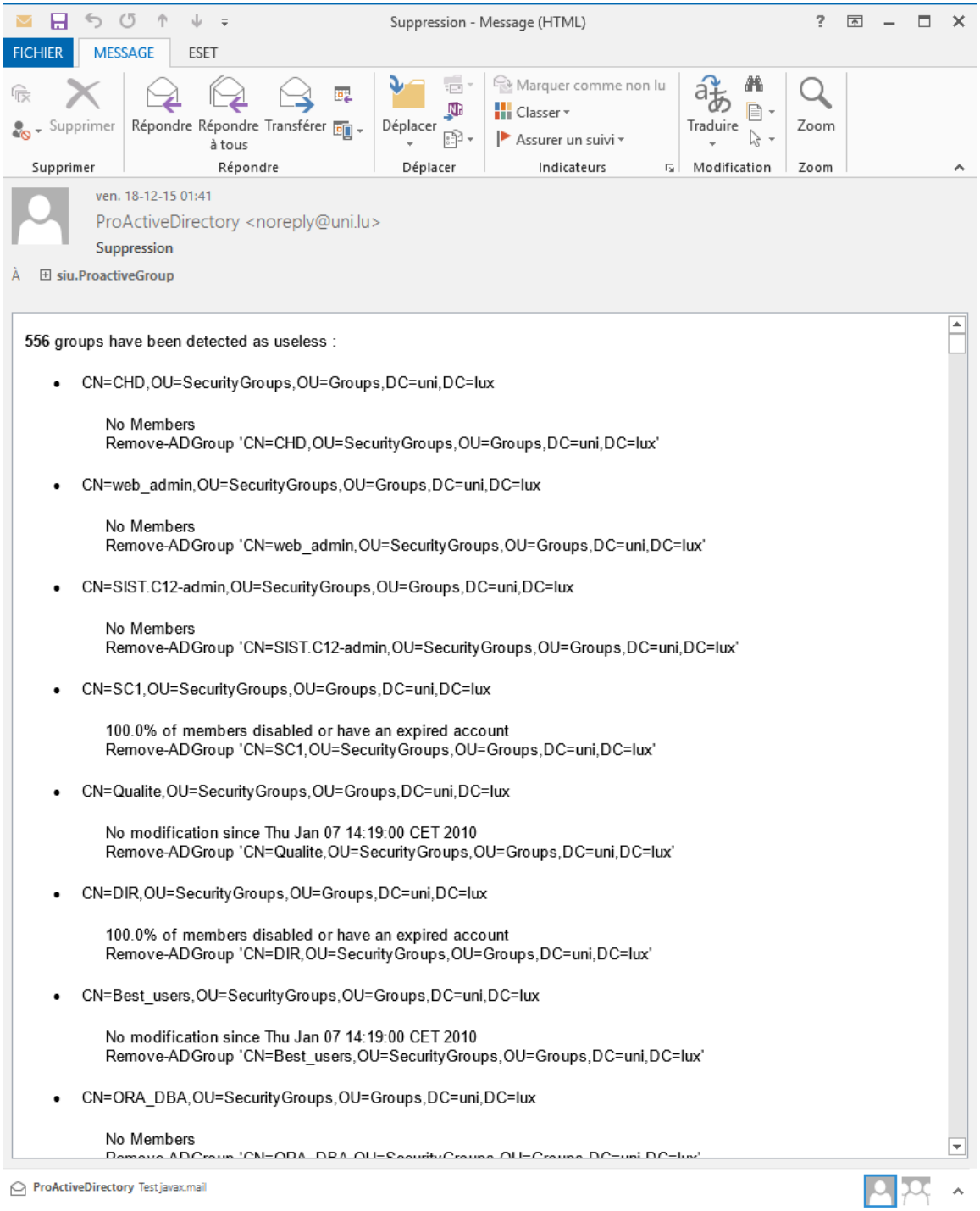


FIGURE 4.2 – Résultat du scénario de détection de groupes inutiles

## 4.3 Aide à la gestion des groupes : création de profils

### 4.3.1 Description

Ce scénario est complémentaire avec le scénario d'aide à l'attribution des groupes qui sera présenté en 4.4, car il génère les profils qui seront utiles pour produire les suggestions. Cette solution apporte en partie une solution au problème de la gestion des groupes, car les profils créés permettent de catégoriser les groupes et peuvent éventuellement permettre de faire un tri ou de regrouper des groupes similaires. La vocation première de ce scénario est bien évidemment d'apporter également une solution au problème d'aide à l'attribution des rôles en fournissant les profils sur lesquels ce scénario complémentaire viendra se baser.

Ce que nous avons appelé un profil est en réalité un ensemble d'attributs que nous avons extrait à partir du groupe ainsi que des membres qui composent ce groupe. Voici les données que nous utilisons.

- Données du groupe :
  - La date de création du groupe
  - La date de la dernière modification du groupe
  - Le nombre de membres qui compose le groupe
  - Une mesure d'importance de la taille du groupe
  - Une mesure d'activité du groupe

- Données des membres :
  - Âge de création moyen des membres
  - La date de création la plus récente parmi les membres. En d'autres mots, le plus "jeune" membre du groupe en terme de date de création.
  - Le manager majoritaire parmi tous les membres du groupe.
  - Le rôle majoritaire parmi tous les membres du groupe.
  - Liste des tops groupes c'est-à-dire les groupes auxquels une grande majorité des membres appartiennent également

Le but de ce scénario n'étant pas de créer un profil unique par groupe, mais plutôt d'arriver à ce que le nombre de profils soit inférieur au nombre de groupes, les profils doivent être comparables entre eux. Pour ce faire, un ordre de comparaison a été imaginé. Cette comparaison se basera sur le manager, le rôle, la mesure de taille et les tops groupes. Si ces valeurs sont égales alors, on conclura que les deux profils sont identiques.

Le fonctionnement du scénario peut être résumé comme suit : après avoir obtenu la liste des groupes présents sur l'Active Directory, on parcourt cette liste et pour chaque groupe, on récupère les données nécessaires à la construction d'un profil temporaire. Ce profil temporaire est ensuite comparé aux profils déjà existants. Si la comparaison avec les profils existants est positive, le groupe est rattaché au profil. Sinon, un nouveau profil est créé et stocké dans la base de données.

La figure 4.3 présente l'ensemble des règles utilisées ainsi que l'enchaînement de celles-ci.

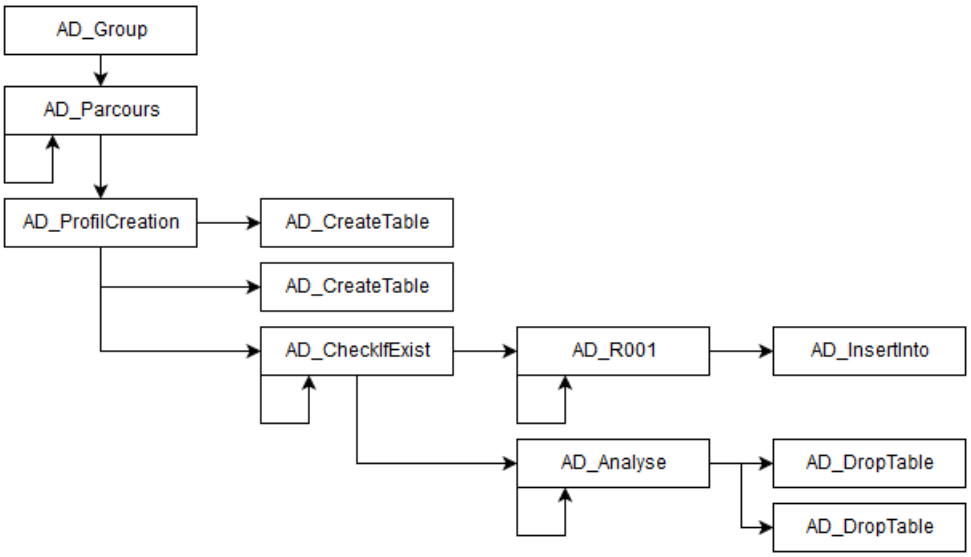


FIGURE 4.3 – Scénario de création de profils

### 4.3.2 Implémentation

#### 4.3.2.1 Règle AD_Group

- **Paramètres :**
  listGroup : la liste des groupes présents dans l'Active Directory

- **Description :**
  Cette règle est la règle d'initialisation du scénario. Elle récupère la liste des groupes de l'Active Directory, elle crée les tables nécessaires au stockage des profils si ces dernières n'existent pas encore, sinon elle efface leurs contenus. Et pour terminer, elle crée la règle nécessaire au parcours de la liste.

- **Pseudo code :**

```
1  dataAcquisition() :
2   this.listGroup = ((AbstractActiveDirWrapper) dataNativeSystem).search("OU=
       Groups,DC=uni,DC=lux","(objectClass=group)");
3
4  activationGuards() :
5   return true;
6
7  conditions() :
8   return true;
9
10 actions() :
11  ((AbstractActiveDirWrapper) dataNativeSystem).cleanDb();
12
13 rulesGeneration() :
14  createRule(new AD_Parcours(this.listGroup.iterator(),0));
15  return true;
```

#### 4.3.2.2 Règle AD_Parcours

- **Paramètres :**
  listGroup : un itérateur sur la liste des groupes

- **Description :**
  Tant que la liste des groupes n'est pas vide, cette règle itère cinq groupes par cinq groupes sur les éléments de la liste de groupes et créée une règle AD_ProfilCreation pour chaque groupe.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return this.listGroup.hasNext();

conditions() :
 return true;

actions(): /

rulesGeneration() {
 if(super.getActivated()){
  for(int j = 0; j<5; j++){
   if(this.listGroup.hasNext()){
    SearchResult x = this.listGroup.next();
    createRule(new AD_ProfilCreation(x,i+j));
   }
  }
  createRule(new AD_Parcours(this.listGroup,i+5));
 }
 return true;
```

#### 4.3.2.3 Règle AD_ProfilCreation

- **Paramètres :**
  group : le groupe qui est en cours d'analyse
  profil : le profil du groupe analysé
  i : le numéro du groupe

- **Description :**
  Cette règle va créer un profil correspondant au groupe analysé. Elle crée également deux tables temporaires, une qui va contenir l'ensemble des données relatives aux membres du groupe ainsi qu'une seconde qui va contenir les tops groupes potentiels, matérialisés par la création des deux règles AD_CreateTable. Elle crée également une règle AD_CheckIfExists.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return true;

conditions() :
```

```
 return true;

actions() :
 this.profil = new Profil(this.group,i);
 this.profil.setMembers(((AbstractActiveDirWrapper) dataNativeSystem).search("
     OU=UNI-Users,DC=uni,DC=lux","(&(objectClass=person)(memberOf="+profil.
     getDistinguishedName()+"))"));

rulesGeneration() :
 if (getActivated()){
  createRule(new AD_CreateTable(this.profil.getNumber()));
  createRule(new AD_CreateTable(this.profil.getNumber()+"G"));
  createRule(new AD_CheckIfExists(this.profil));
 }
 return true;
```

#### 4.3.2.4 Règle AD_CreateTable

- **Paramètres :**
  tableName : le nom de la table à créer

- **Description :**
  Cette règle est utilisée pour créer une table dont on lui spécifie le nom dans la base de données.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return true;

conditions() :
 return true;

actions() :
 ((AbstractActiveDirWrapper) dataNativeSystem).createTable(this.tableName);

rulesGeneration() :
 return false;
```

#### 4.3.2.5 Règle AD_CheckIfExists

- **Paramètres :**
  tableName : le nom de la table dont il faut vérifier l'existence
  tableExist : un booléen indiquant l'existence de la première table
  tableGExist : un booléen indiquant l'existence de la deuxième table
  group : le profil du groupe

- **Description :**
  Cette règle s'assure que les tables créées par les règles AD_CreateTable lancées par la règle AD_ProfilCreation sont présentes dans la base de données. Si elles sont présentes, cette règle crée deux autres règles sinon elle se clone jusqu'à ce que les tables soient bien créées.

- **Pseudo code :**

```
dataAcquisition() :
 this.tableExist = ((AbstractActiveDirWrapper) dataNativeSystem).tableExist(
     this.group.getNumber());
 this.tableGExist = ((AbstractActiveDirWrapper) dataNativeSystem).tableExist(
     this.group.getNumber()+"G");

activationGuards() :
 return tableExist && tableGExist;

conditions() :
 return true;

actions() : /

rulesGeneration() :
 if(getActivated()){
  createRule(new AD_R001(this.group));
  createRule(new AD_Analyse(this.group));
 }
 else
  createRule(new AD_CheckIfExists(this.group));
 return true;
```

#### 4.3.2.6 Règle AD_R001

- **Paramètres :**
  listUser : un itérateur sur la liste des membres d'un groupe
  group : le profil du groupe

- **Description :**
  Cette règle itère sur la liste des membres d'un groupe et crée des règles AD_InsertInto pour chaque membre tant que la liste n'est pas vide. Si c'est le cas, la règle insère un mot clé dans une table de la base de donnée qui sera utile pour la suite du scénario.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return this.listUser.hasNext();

conditions() : return true;

actions() : /

rulesGeneration() :
 if(super.getActivated()){
  for(int i = 0; i<100; i++){
   if(this.listUser.hasNext())
   createRule(new AD_InsertInto(this.listUser.next(),this.group));
  }
  createRule(new AD_R001(this.listUser, this.group));
 }
 else
  createRule(new AD_InsertInto("Finish", this.group));
 return true;
```

#### 4.3.2.7 Règle AD_InsertInto

- **Paramètres :**
  user : un utilisateur membre du groupe
  manager : le manager de l'utilisateur
  tableName : le nom de la table dans laquelle il faut insérer des données
  group : le profil du groupe
  memberOf : les groupes auxquels l'utilisateur appartient
  endMessage : un flag permettant de savoir s'il faut insérer le mot clé de fin de la règle AD_R001
  whenCreated : la date de création de l'utilisateur dans l'Active Directory
  role : le rôle attribué à l'utilisateur

- **Description :**
  Cette règle insère des données dans les deux tables temporaires créées par les règles AD_CreateTable.

- **Pseudo code :**

```
dataAcquisition() :
 try {
  if(endMessage.equals("")){
   this.whenCreated = user.getAttributes().get("whenCreated").get().toString()
       ;
   this.memberOf = Collections.list(user.getAttributes().get("memberOf").
       getAll());
   this.manager = user.getAttributes().get("manager").get().toString();
            this.role = user.getAttributes().get("extensionAttribute4").get().
                toString();
  }
 } catch (Exception e) {
  this.manager="noManager";
 }

activationGuards() :
 return true;

conditions() :
 return true;

actions() :
 SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmSS");
 sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
 if(endMessage.equals("")){
  try {
   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.tableName,
       this.manager,Long.toString(sdf.parse(this.whenCreated).getTime()),this.
       role);
  } catch (Exception e) {
   e.printStackTrace();
  }
  if(this.memberOf != null){
   ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.tableName+"G"
       ,this.memberOf,this.group.getDistinguishedName());
  }
 }
 else {
  ((AbstractActiveDirWrapper) dataNativeSystem).insertInto(this.tableName+"G"
```

```
        ,(String) this.endMessage,null,null);
 }

rulesGeneration() :
 return false;
```

#### 4.3.2.8 Règle AD_Analyse

- **Paramètres :**
  count : le nombre de lignes présentes dans la première table
  finish : un booléen qui permet d'indiquer la présence ou non du mot clé dans la deuxième table
  group : le groupe qui est en train d'être analysé
  tableName : le nom de la table dans laquelle il faut stocker les résultats

- **Description :**
  Cette règle attend que les deux tables temporaires soient peuplées de données avant de les analyser afin de construire le profil complet d'un groupe. Pour détecter quand elle peut se lancer, elle attend que le nombre de lignes dans la première table soit égal au nombre de membres d'un groupe. Et pour la deuxième table, elle attend que le mot clé de fin soit inséré. Elle crée ensuite deux règles AD_DropTable.

- **Pseudo code :**

```
dataAcquisition() :
 this.count = ((AbstractActiveDirWrapper) dataNativeSystem).countLines(this.
     tableName);
 this.finish = ((AbstractActiveDirWrapper) dataNativeSystem).getFinish(this.
     tableName + "G");

activationGuards() :
 return (count == this.group.getMembersSize() && this.finish);

conditions() :
 return true;

actions() :
 if (this.group.getMembersSize() != 0) {
  this.group.setManager(((AbstractActiveDirWrapper) dataNativeSystem).
      getManager(this.tableName));
  System.out.println(this.group.getManager());
  this.group.setTopGroup(((AbstractActiveDirWrapper) dataNativeSystem).
      getTopGroup(this.tableName + "G"));
  for (int i = 0; i < 3; i++)
   System.out.println(this.group.getTopGroup()[i]);
  this.group.setMostRecentCreationDate(((AbstractActiveDirWrapper)
      dataNativeSystem).getMax(this.tableName));
  System.out.println(this.group.getMostRecentCreationDate());
  this.group.setMeanCreationDate(((AbstractActiveDirWrapper) dataNativeSystem)
      .getMean(this.tableName));
  System.out.println(this.group.getMeanCreationDate());
  this.group.setRole(((AbstractActiveDirWrapper) dataNativeSystem).getRole(
      this.tableName));
  System.out.println(this.group.getRole());
  ((AbstractActiveDirWrapper) dataNativeSystem).checkProfil(this.group);
 }

```

```
rulesGeneration() :
 if (super.getActivated()) {
  createRule(new AD_DropTable(this.tableName));
  createRule(new AD_DropTable(this.tableName + "G"));
 } else {
  createRule(new AD_Analyse(this.group));
 }
 return true;
```

#### 4.3.2.9 Règle AD_DropTable

- **Paramètres :**
  tableName : le nom donné à la table que l'on souhaite supprimer

- **Description :**
  Cette règle supprime la table dont on lui fournit le nom.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return true;

conditions() :
 return true;

actions() :
 ((AbstractActiveDirWrapper) dataNativeSystem).dropTable(this.tableName);

rulesGeneration() :
 return false;
```

### 4.3.3 Résultats

Les profils déjà calculés seront stockés dans une base de données et seront modélisés de la sorte, présenté en figure 4.5. En figure 4.4 est présenté un exemple de profil créé pour un groupe.

```
Date de creation : Tue Feb 04 16:03:00 CET 2014
Derniere modification : Thu Oct 29 09:32:00 CET 2015
Manager : CN=Rieneke.Cruchten,OU=GestionRessourcesHumaines,OU=Rectorat,OU=UNI-
    Users,DC=uni,DC=lux 10
top :     CN=ReportingGroup {7d1687f8-2337-4a16-8160-1a337ddc9c1f},OU=CRM,OU=
    SecurityGroups,OU=Groups,DC=uni,DC=lux 181
    CN=Acme-ContratEtu-Demande,OU=ACME,OU=SecurityGroups,OU=Groups,DC=uni,DC=
        lux 180
    CN=ReportingGroup {02979a8d-bb7c-4ab2-a2be-60cd9557e6dd},OU=CRM,OU=
        SecurityGroups,OU=Groups,DC=uni,DC=lux 178
    CN=Staff,OU=SecurityGroups,OU=Groups,DC=uni,DC=lux 148
    CN=Atlas_FSTC_All_Members,OU=ServerSharesManagedByMRE,OU=Groups,DC=uni,DC=
        lux 20
Mean :    Thu Feb 17 10:11:38 CET 2005
Total members: 185
```

FIGURE 4.4 – Exemple de profil

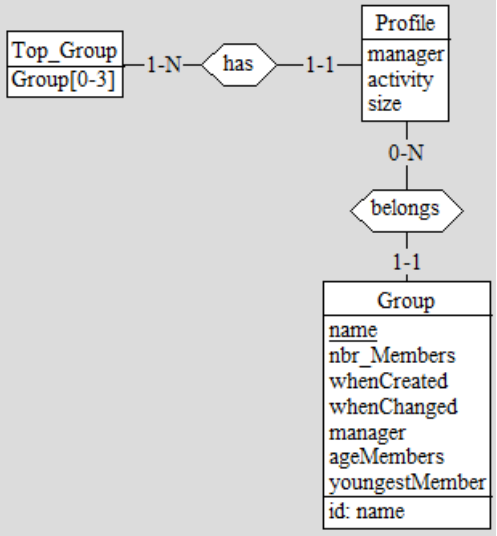


FIGURE 4.5 – Schéma relationnel concernant la création de profils

## 4.4 Aide à l'attribution des groupes

### 4.4.1 Description

Le scénario d'aide à l'attribution de groupes repose sur les profils créés par le scénario présenté en 4.3. La combinaison de ces deux scénarios constitue une solution quant au problème de l'attribution de groupes pour de nouveaux utilisateurs. Le but de ce scénario est de fournir une liste de suggestions de groupes auxquels le nouvel utilisateur a de fortes chances de devoir appartenir.

Afin de fournir des suggestions correctes, nous devons recueillir des informations à propos du nouvel utilisateur afin de pouvoir le comparer aux profils à notre disposition. Pour cela, nous utilisons donc les mêmes informations que lors de la construction des profils de groupes.

Les informations que nous utilisons à propos des nouveaux utilisateurs sont les suivantes : nous nous servons des données concernant le manager et le rôle du nouvel utilisateur. Sur base de ces deux données, nous filtrons les profils existants afin d'effectuer un second tri avec les priorités suivantes.

1. Manager
2. Rôle
3. Mesure d'activité du groupe
4. Mesure de la taille du groupe
5. Liste des tops groupes

De façon plus détaillée, l'algorithme fonctionnera comme suit, un premier filtrage des profils sera effectué au niveau du manager et du rôle, nous en retirons une liste de profils ayant tous le même manager et rôle, lesquels sont égaux au manager et rôle du nouvel utilisateur. Si cette liste ne renvoie aucun résultat, seule une comparaison au niveau du manager est effectuée. Ensuite, nous trions cette liste avec les critères suivants : la mesure d'activité du profil ainsi que la mesure de la taille du profil. De cette liste triée, nous ne gardons que les X premiers profils. Nous procédons ensuite à l'intersection des tops groupes afin d'obtenir la suggestion finale.

L'implémentation de ce scénario, ainsi que les différentes règles qui le composent, est présentée en figure 4.6.

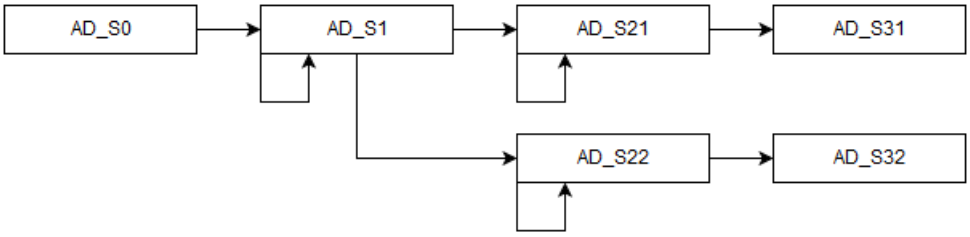


FIGURE 4.6 – Scénario de suggestion

## 4.4.2 Implémentation

### 4.4.2.1 Règle AD_S0

- **Paramètres :**
  listUser : la liste des nouveaux utilisateurs détectés
  day : la date du jour

- **Description :**
  Cette règle est la règle d'initialisation du scénario. Si elle détecte que de nouveaux utilisateurs ont été créés, elle déclenche alors le scénario de suggestions. Elle crée une table temporaire où seront stockés les profils qui correspondent aux nouveaux utilisateurs.

- **Pseudo code :**

```
dataAcquisition() :
 this.listUser = ((AbstractActiveDirWrapper) dataNativeSystem).search("OU=UNI-
     Users,DC=uni,DC=lux","(&(objectClass=person)(whenCreated>="+this.day+".OZ
     )(manager=*))");

activationGuards() :
 return (this.listUser.size()>0);

conditions() :
 return true;

actions() : ((AbstractActiveDirWrapper) dataNativeSystem).
    createTableSuggestion();

rulesGeneration() :
 if(getActivated())
  createRule(new AD_S1("Suggestion",this.listUser));
 return true;
```

#### 4.4.2.2 Règle AD_S1

- **Paramètres :**
  tableName : le nom de la table dont on veut s'assurer l'existence
  tableExist : un booléen qui indique l'existence de la table
  listUser : la liste des nouveaux utilisateurs

- **Description :**
  Cette règle s'assure que la table créée par la règle AD_S0 est bien présente dans la base de données. Elle se clone tant que cela n'est pas le cas et crée les règles AD_S21 et AD_S22 dans le cas contraire.

- **Pseudo code :**

```
dataAcquisition() :
 this.tableExist = ((AbstractActiveDirWrapper) dataNativeSystem).tableExist(
     this.tableName);

activationGuards() :
 return true;

conditions() :
 return tableExist;

actions() : /

rulesGeneration() :
 if(getActivated()){
  createRule(new AD_S21(this.listUser.iterator()));
  createRule(new AD_S22(this.listUser.size()));
 }
 else
  createRule(new AD_S1(this.tableName, this.listUser));
 return true;
```

#### 4.4.2.3 Règle AD_S21

- **Paramètres :**
  listUser : un itérateur sur la liste des nouveaux utilisateurs

- **Description :**
  Cette liste itère sur la liste des nouveaux utilisateurs. Pour chaque utilisateur, elle crée une règle AD_31. Elle se clone tant qu'elle n'a pas parcouru toute la liste.

- **Pseudo code :**

```
dataAcquisition() : /

activationGuards() :
 return this.listUser.hasNext();

conditions() :
 return true;

actions() : /

rulesGeneration() :
 if(getActivated()){
  createRule(new AD_S31(this.listUser.next()));
  createRule(new AD_S21(this.listUser));
 }
 return true;
```

#### 4.4.2.4 Règle AD_S31

- **Paramètres :**
  user : un nouvel utilisateur
  userManager : le manager du nouvel utilisateur
  userName : le Distinguish Name de l'utilisateur
  role : le rôle du nouvel utilisateur

- **Description :**
  Cette règle recherche les profils correspondant à un nouvel utilisateur et les insère dans la table créée lors de la règle AD_S0.

- **Pseudo code :**

```
dataAcquisition() :
 try {
  this.userManager = (String) user.getAttributes().get("manager").get();
  this.userName = (String) user.getAttributes().get("distinguishedName").get()
      ;
  this.role = (String) user.getAttributes().get("extensionAttribute4").get();
 } catch (Exception e) {
  // TODO Auto-generated catch block
  this.role ="noRole";
 }

activationGuards() :
 return true;

```

```
conditions() :
 return true;

actions() :
 ((AbstractActiveDirWrapper) dataNativeSystem).findSuggestion(this.userManager
     ,this.userName,this.role);

rulesGeneration() :
 return false;
```

#### 4.4.2.5 Règle AD_S22

- **Paramètres :**
  size : la taille de la liste des nouveaux utilisateurs
  count : le nombre d'utilisateurs dont l'analyse est déjà faite
- **Description :**
  Cette règle attend que l'ensemble des nouveaux utilisateurs aient été analysés et envoie un mail automatique au gestionnaire afin de l'avertir des correspondances trouvées.

- **Pseudo code :**

```
dataAcquisition() :
 this.count = ((AbstractActiveDirWrapper) dataNativeSystem).countSuggestion();

activationGuards() :
 return (this.count == this.size);;

conditions() :
 return true;

actions() :
 new Mail("siu.ProactiveGroup@uni.lu","Suggestion","<b>"+this.count+"</b> user
     (s) have been created : <br>"+((AbstractActiveDirWrapper)
     dataNativeSystem).getSuggestion()).send();

rulesGeneration() :
 if(getActivated()){
  createRule(new AD_S32("Suggestion"));
 }
 else
  createRule(new AD_S22(this.size));
 return true;
```

#### 4.4.2.6 Règle AD_S32

- **Paramètres :**
  tableName : le nom donné à la table que l'on souhaite supprimer

- **Description :**
  Cette règle supprime la table dont on lui fournit le nom.

- **Pseudo code :**

```
dataAcquisition() : /

```

```
activationGuards() :
 return true;

conditions() :
 return true;

actions() :
 ((AbstractActiveDirWrapper) dataNativeSystem).dropTable(this.tableName);

rulesGeneration() :
 return false;
```

### 4.4.3 Résultat

Le résultat de ce scénario est un mail envoyé quotidiennement au responsable de l'Active Directory, et qui a la structure suivante.

- Nombre de nouveaux utilisateurs détectés
- Pour chaque nouvel utilisateur
  - Le Distinguish Name de l'utilisateur
  - Raison de la correspondance avec le/les profils correspondant
  - Le Distinguish Name de l'ensemble des groupes auxquelles l'utilisateur peut potentiellement être ajouté.
  - Liste de commandes powershell permettant au gestionnaire d'ajouter l'utilisateur si la suggestion est bonne

Un exemple de mail envoyé est illustré à la figure 4.7.

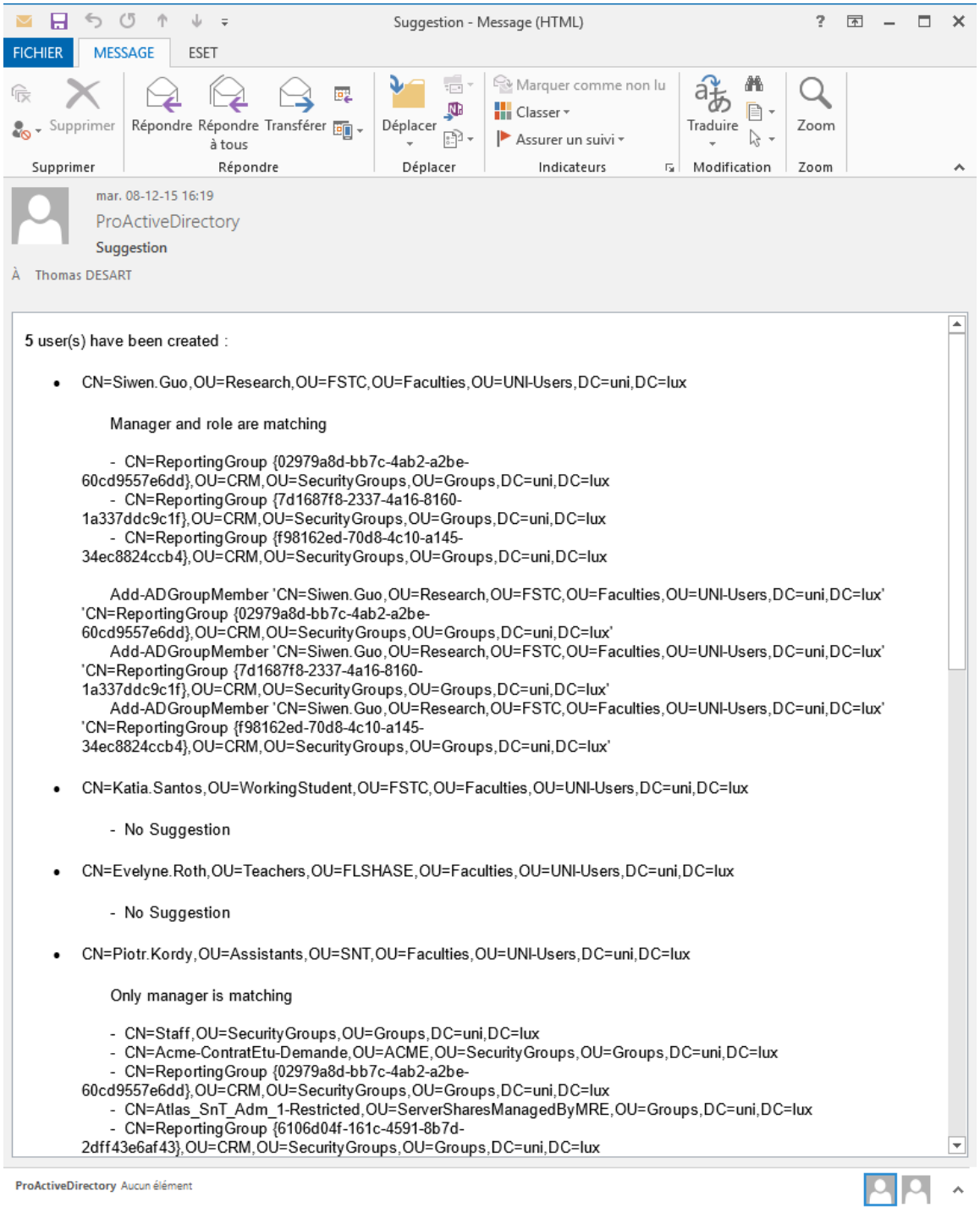


FIGURE 4.7 – Résultat du scénario de suggestion

## 4.5 Aide à la gestion des mots de passe

### 4.5.1 Description

Ce scénario propose une solution quant au problème de la gestion des renouvellements des mots de passe.

Le fonctionnement du scénario est le suivant : premièrement, on récupère la liste des utilisateurs de l'Active Directory. Ensuite, on parcourt cette liste et pour chaque utilisateur, on fait les vérifications d'usage. La politique de renouvellement du mot passe, dans le cas de la solution implémentée, oblige l'utilisateur à modifier son mot de passe tous les deux cent dix jours. Ce scénario va prévenir les utilisateurs de l'échéance proche de leur mot de passe. Les rappels se font par mail. Il y a quatre rappels.

- 1$^{er}$rappel : Trente jours avant la date d'expiration du mot de passe
- 2$^{e}$rappel : Quinze jours avant la date d'expiration du mot de passe
- 3$^{e}$rappel : Trois jours avant la date d'expiration du mot de passe
- 4$^{e}$rappel : Un jour avant la date d'expiration du mot de passe avec en plus l'envoi d'une notification est également envoyée au manager de l'utilisateur ainsi qu'à l'équipe du service informatique

Les données des vérifications sont ensuite stockées dans une base de données. Ce scénario utilise au total trois tables.

- Une table qui va stocker le nom des utilisateurs dont le mot de passe a expiré depuis au moins trente jours.
- Une table qui va servir à connaître combien de rappels l'utilisateur a déjà reçu. Par exemple, si l'utilisateur a déjà reçu le premier rappel, on peut directement lui envoyer le second. Une fois qu'il aura modifié son mot de passe, son nom est supprimé de cette table.
- Une table de log qui garde un historique complet des rappels envoyés à l'utilisateur.

Le résultat de ce scénario se traduit par l'envoi de plusieurs mails. Les mails envoyés aux utilisateurs, mais également un mail envoyé au gestionnaire lui indiquant tous les comptes dont le mot de passe est expiré depuis plus de trente jours. La figure 4.8 illustre les règles utilisées et leur organisation afin d'obtenir le résultat escompté.

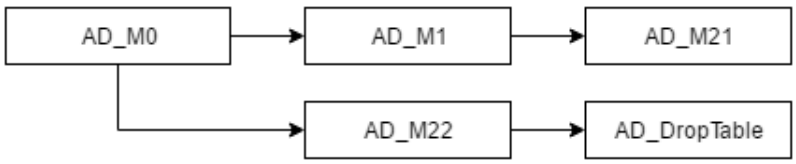


Figure 4.8 – Scénario de Gestion des mots de passe

### 4.5.2 Implémentation

#### 4.5.2.1 Règle AD_M0

- **Paramètres :**
  listUser : la liste de tous les utilisateurs présents dans l'Active Directory

- **Description :**
  Cette règle est la règle d'initialisation du scénario. Elle se charge de récupérer la liste des utilisateurs, de créer les trois tables afin d'assurer le bon fonctionnement du scénario et de créer les règles AD_M1 et AD_M22.

- **Pseudo code :**

```
1  dataAcquisition() :
2   this.listUser = ((AbstractActiveDirWrapper) dataNativeSystem).search("OU=UNI-
       Users,DC=uni,DC=lux","(&(objectClass=person)(mail=*)(pwdLastSet=*)(!(
       pwdLastSet=0)))");
3
4  activationGuards() :
5   return true;
6
7  conditions() :
8   return true;
9
10 actions() :
11  ((AbstractActiveDirWrapper) dataNativeSystem).createTable("Password30");
12  ((AbstractActiveDirWrapper) dataNativeSystem).createTable("Password");
13  ((AbstractActiveDirWrapper) dataNativeSystem).createTable("password_historic"
       );
14
15 rulesGeneration() :
16  createRule(new AD_M1(this.listUser.iterator()));
17  createRule(new AD_M22());
18  return true;
```

#### 4.5.2.2 Règle AD_M1

- **Paramètres :**
  listUser : un itérateur sur la liste des utilisateurs présents dans l'Active Directory

- **Description :**
  Cette règle effectue le parcours de la liste d'utilisateurs. Afin d'améliorer les performances du scénario, elle crée cinq cents règles AD_M21 à la fois. Une fois le parcours de la liste terminé, elle insère le mot qui servira de déclencheur pour une autre règle.

- **Pseudo code :**

```
1  dataAcquisition() : /
2
3  activationGuards() :
4   return this.listUser.hasNext();
5
6  conditions() :
7   return true;
8
9  actions() : /
```

```
10
11 rulesGeneration() :
12  if(super.getActivated()){
13  for(int i = 0; i<500; i++){
14   if(this.listUser.hasNext())
15    createRule(new AD_M21(this.listUser.next()));
16  }
17  createRule(new AD_M1(this.listUser));
18   }
19   else
20  ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password30", "
        Finish", null, null);
21  return true;
```

#### 4.5.2.3 Règle AD_M21

- **Paramètres :**
  user : l'utilisateur dont les données vont être analysées
  userAccountControl : le type de compte que l'utilisateur s'est vu attribué
  pwdLastSet : la date de dernière modification du mot de passe
  mail : l'adresse e-mail de l'utilisateur en cours d'analyse
  accountExpires : la date à laquelle le compte de l'utilisateur est supposé expirer
  displayName : le nom et le prénom de l'utilisateur
  location : le site géographique sur lequel l'utilisateur travaille
  dn : le Distinguish Name de l'utilisateur
  manager : le manager de l'utilisateur
  m : le nombre d'avertissements déjà reçus par l'utilisateur

- **Description :**
  Cette règle effectue les vérifications d'usage concernant un utilisateur. En fonction de la date d'expiration du mot de passe de l'utilisateur, elle lui envoie une notification par mail.

- **Pseudo code :**

```
1  dataAcquisition() :
2    try {
3  this.userAccountControl = Integer.parseInt(this.user.getAttributes().get("
       userAccountControl").get().toString());
4  this.pwdLastSet = new Date((Long.parseLong(this.user.getAttributes().get("
       pwdLastSet").get().toString())/FILETIME_ONE_MILLISECOND) -
       FILETIME_EPOCH_DIFF);
5  this.mail = this.user.getAttributes().get("mail").get().toString();
6  this.dn = this.user.getAttributes().get("distinguishedName").get().toString()
       ;
7  this.accountExpires = Long.parseLong(this.user.getAttributes().get("
       accountExpires").get().toString());
8  this.displayName = this.user.getAttributes().get("displayName").get().
       toString();
9    } catch (Exception e) {
10   e.printStackTrace();
11   }
12   this.m = ((AbstractActiveDirWrapper) dataNativeSystem).getFrom("Password",
         this.dn);
13
14 activationGuards() :
```

```
15  this.calendar.setTime(this.pwdLastSet);
16  calendar.add(Calendar.DATE, 180);
17  if(calendar.getTimeInMillis()<(System.currentTimeMillis()) && (this.
        accountExpires > System.currentTimeMillis() || this.accountExpires == 0))
18   return true;
19  else{
20   ((AbstractActiveDirWrapper) dataNativeSystem).deleteFrom("Password", this.dn
        );
21   return false;
22  }
23
24 conditions() :
25  return (!((this.userAccountControl & 0x00010000) == 0x00010000) && !((this.
        userAccountControl & 2) == 2));
26
27 actions() :
28  Long today = System.currentTimeMillis();
29  Long pwd = calendar.getTimeInMillis();
30  if((pwd + 1000*60*60*24*14L) > today ){
31   String s = makeText(this.displayName ,30);
32   if(this.m !=1){
33    System.out.println("\t\t"+this.mail+"␣moins␣de␣30␣jours");
34     ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password", this.
           dn, "1");
35    //new Mail(this.mail, "Password Expiry Notification - First Reminder", s).
          send();
36    new Mail("siu.ProactiveGroup@uni.lu", "Password␣Expiry␣Notification␣-␣First
          ␣Reminder", s).send();
37   }
38  }
39  else if ((pwd + 1000*60*60*24*26L) > today){
40   String s = makeText(this.displayName ,15);
41   if(this.m != 2){
42    System.out.println("\t\t"+this.mail+"␣moins␣de␣15␣jours");
43    ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password", this.
          dn, "2");
44    //new Mail(this.mail, "Password Expiry Notification - Second Reminder", s).
          send();
45    new Mail("siu.ProactiveGroup@uni.lu", "Password␣Expiry␣Notification␣-␣
          Second␣Reminder", s).send();
46  }
47  }
48  else if ((pwd + 1000*60*60*24*29L) > today){
49   String s = makeText(this.displayName ,3);
50   if(this.m != 3){
51    System.out.println("\t\t"+this.mail+"␣moins␣de␣3␣jours");
52    ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password", this.
          dn, "3");
53    //new Mail(this.mail, "Password Expiry Notification - Second Reminder", s).
          send();
54    new Mail("siu.ProactiveGroup@uni.lu", "Password␣Expiry␣Notification␣-␣Third
          ␣Reminder", s).send();
55   }
56  }
57  else if ((pwd + 1000*60*60*24*30L) > today){
58   String s = makeText(this.displayName ,1);
59   try {
60    this.manager = this.user.getAttributes().get("manager").get().toString();
61    this.manager = ((AbstractActiveDirWrapper) dataNativeSystem).search(this.
          manager,"(&(objectClass=person))").get(0).getAttributes().get("mail").
          get().toString();
62    System.out.println(this.manager);
```

```
63    } catch (Exception e) {
64     this.manager = "";
65    }
66    try {
67     this.location = this.user.getAttributes().get("l").get().toString();
68    } catch (Exception e) {
69     this.location="";
70    }
71    this.location= this.getSIULoc(this.location);
72    if(this.m != 4){
73     System.out.println("\t\t"+this.mail+" moins de 1 jours");
74     ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password", this.
           dn, "4");
75     new Mail("siu.ProactiveGroup@uni.lu", "Password Expiry Notification - 
           Fourth Reminder", s).send();
76     //new Mail(this.mail, "Password Expiry Notification - Fourth Reminder", s).
           send(this.manager,this.location);
77    }
78   }
79   else if (pwd + 1000*60*60*24*60L < today){
80    ((AbstractActiveDirWrapper) dataNativeSystem).insertInto("Password30", this.
          dn, null, "expired since "+((today -(pwd+1000*60*60*24*30L))
          /(1000*60*60*24L))+" days");
81   }
82
83  rulesGeneration() :
84   return false;
```

#### 4.5.2.4 Règle AD_M22

- **Paramètres :**
  finish : un booléen indiquant la présence ou non du mot clé de fin dans la table

- **Description :**
  Cette règle attend l'insertion du mot clé par la règle AD_M1 dans la table. Une fois ce mot clé détecté, elle analyse la table où sont stockés les utilisateurs dont le mot de passe a expiré depuis plus de trente jours. Elle en notifie le gestionnaire en lui présentant un récapitulatif.

- **Pseudo code :**

```
1   dataAcquisition() :
2    this.finish = ((AbstractActiveDirWrapper) dataNativeSystem).getFinish("
         Password30");
3
4   activationGuards() :
5    return this.finish;
6
7   conditions() :
8    return true;
9
10  actions() :
11   new Mail("siu.ProactiveGroup@uni.lu", "Password Expiry Notification", ((
         AbstractActiveDirWrapper) dataNativeSystem).getExpiredPassword()).send();
12
13  rulesGeneration() :
14   if(getActivated())
15    createRule(new AD_DropTable("Password30"));
```

```
16  else
17   createRule(new AD_M22());
18  return true;
```

### 4.5.3 Résultat

Le résultat de ce scénario consiste en l'envoi de mails soit à l'utilisateur lorsqu'il atteint une des quatre étapes, comme illustré en 4.9, soit au gestionnaire illustré en 4.10 en lui fournissant la liste de tous les utilisateurs dont le mot de passe est expiré depuis au moins trente jours.

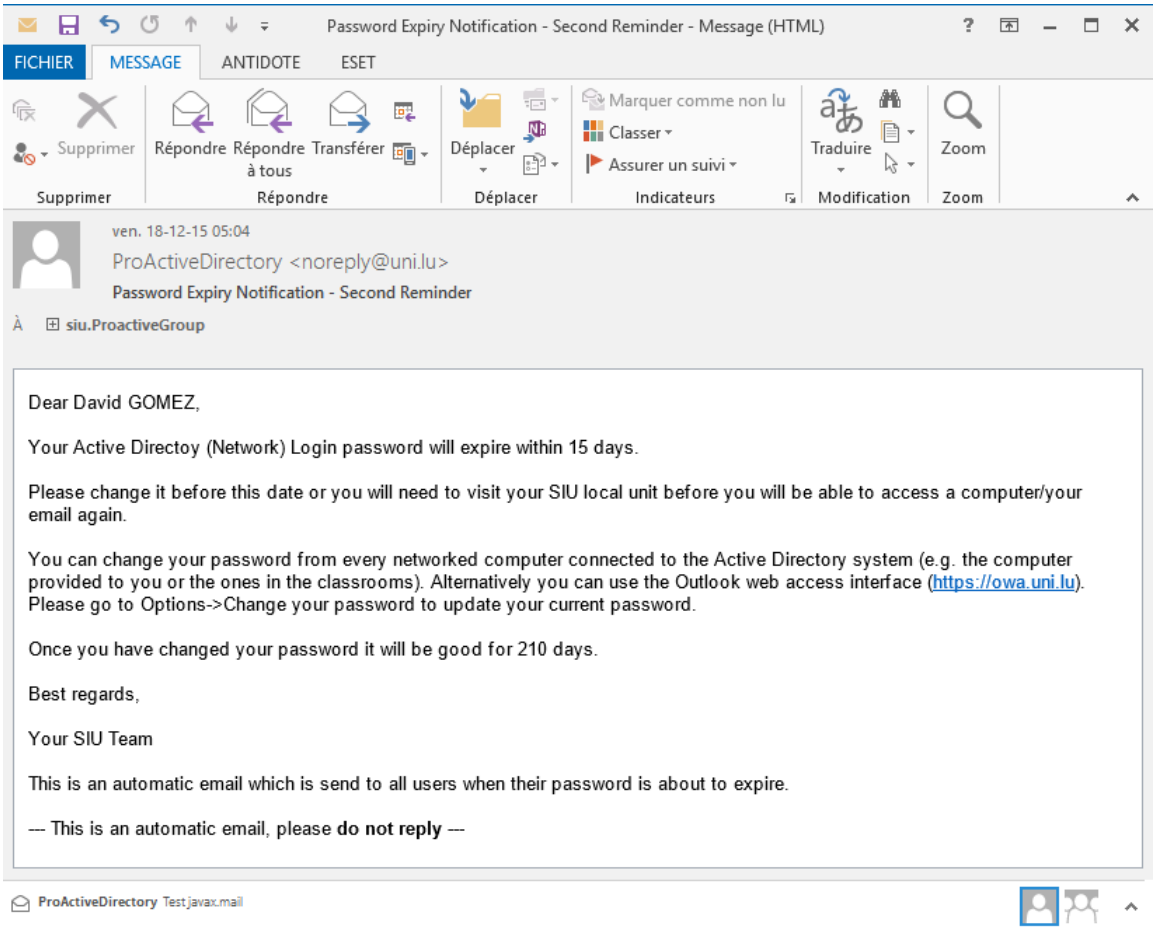


FIGURE 4.9 – Aide à la gestion des mots de passe : rappel

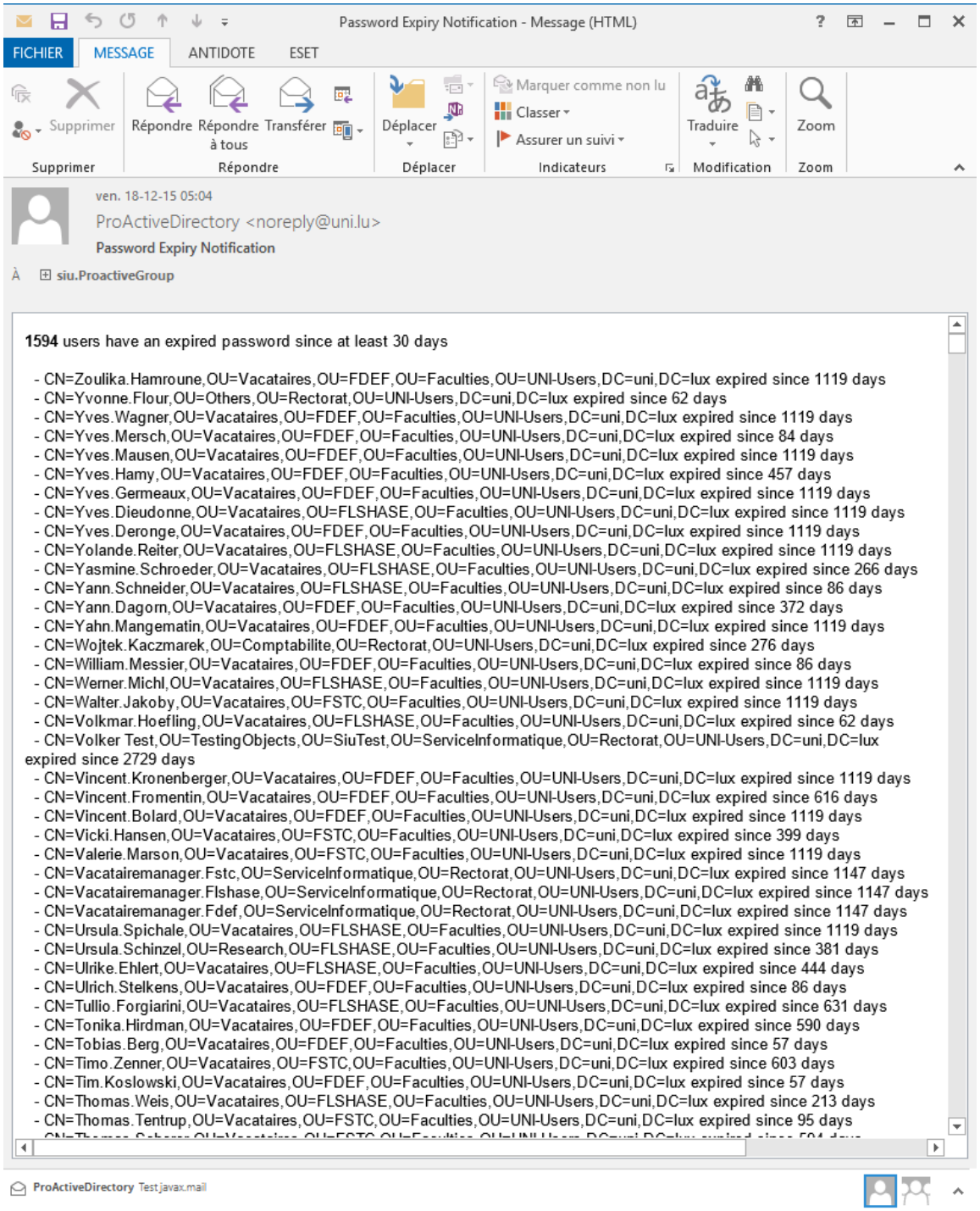


FIGURE 4.10 – Aide à la gestion des mots de passe : utilisateurs expirés

## 4.6 Évaluation

Ce chapitre présente un prototype qui reste assez simple. Le but de ce prototype n'était pas de créer un système le plus complet possible. Par contre, le but recherché était de prouver qu'il est possible d'ajouter une certaine autonomie à un serveur d'identité grâce à la technologie qu'est le proactive computing et de créer ainsi un système d'aide à la gestion d'un serveur d'identité. Ce prototype atteint ce but et présente donc une façon d'automatiser un Active Directory à l'aide du Proactive Engine.

Les fonctions développées découlent de besoins réels de l'administrateur du système. Bien que ces fonctions soient limitées, les résultats fournis étaient assez concluants et pouvaient même s'avérer être utiles pour l'administrateur. N'ayant pas pu réaliser de tests sur le long terme quant à la réelle aide apportée par le prototype, nous ne pouvons tirer de conclusions à ce sujet. Nous pouvons seulement affirmer que durant les quelques semaines où le système a été testé, une réelle utilité et aide pouvait être amenée par certaines des fonctions implémentées. De plus, le prototype développé doit être vu comme un outil d'aide et pas une automatisation complète de la gestion d'un serveur d'identité. Le prototype pourrait facilement être adapté pour une automatisation totale. En conclusion, nous avons répondu aux besoins de l'administrateur, désireux d'obtenir un outil l'aidant dans son travail plutôt qu'une automatisation totale.

Plus particulièrement deux fonctions, que sont l'aide à la détection des groupes inutiles et l'aide à la gestion des mots de passe, ont montrés des résultats positifs. La première exécution de l'aide à la détection des groupes inutiles nous a confirmé qu'il est impossible pour un seul homme de maintenir tout cela. En effet plus de cinq cents groupes ont été détectés comme inutiles. Il s'est avéré que pratiquement l'ensemble des groupes proposés étaient réellement inutiles. L'administrateur n'ayant plus le temps de vérifier l'ensemble des groupes, cette fonctionnalité de détection de groupes s'est montrée très utile en fournissant même des résultats plus que concluants.

Le scénario de gestion des mots de passe quant à lui bien qu'étant assez simple et ne présentant pas de réel challenge intellectuel, s'est avéré être fournisseur de résultats plus que probants et présentant une vraie aide pour l'administrateur. En effet, ce scénario a permis de détecter que de nombreux utilisateurs n'avaient pas modifié leur mot de passe depuis bien longtemps et qu'il était donc vraisemblable que ces personnes ne fassent potentiellement plus partie de l'université. Cette fonctionnalité destinée à la base pour la gestion des mots de passe permet donc également de nettoyer l'Active Directory en détectant des comptes de personnes ayant quitté l'organisation ou devenues inactives. Cette fonctionnalité a donc présenté une véritable aide pour l'administrateur littéralement submergé par le nombre de comptes présents dans le système.

Le dernier scénario, qui est la suggestion d'appartenance pour un nouvel utilisateur, est lui un peu plus difficile à évaluer du fait que des tests à long terme n'ont pu être effectués. Ces tests auraient permis de réellement se rendre compte de la précision des résultats produits. Cette fonctionnalité repose sur un autre scénario qu'est la création de profils de groupes. Ces deux scénarios ne se basent que sur les données présentes dans l'Active Directory pour proposer des suggestions. Le problème principal de ce scnéario est que les données manipulées sont majoritairement statiques. Dès lors, pour que ce scénario soit totalement efficace, il faudrait interagir avec d'autres systèmes ou applications pouvant fournir des don-

nées dynamiques, par exemple les accès effectués aux différents services. La combinaison de ces données statiques et dynamiques permettrait sans doute d'augmenter la précision des résultats et dès lors de fournir des suggestions fiables à l'administrateur.

Globalement, l'évaluation du prototype est largement positive. Il a permis de répondre positivement à la question initiale de savoir s'il était possible d'ajouter une certaine autonomie à un serveur d'identité. Cette réponse positive permet d'envisager une amélioration du prototype afin de fournir un système autonome ou semi-autonome effectuant des tâches de maintenance sur un serveur d'identité.

Le choix du proactive computing par le biais du proactive engine a été fait car c'est un moyen simple de rendre un système existant totalement ou partiellement autonome. Les avantages et inconvénients de l'utilisation du proactive computing dans le but de construire un système autonome sont présentés au point 3.4. Nous avons donc choisi l'approche bottom-up, consistant ajouter de l'autonomie à un système via un module externe plutôt que l'approche top-down qui aurait consisté à concevoir un système de gestion de l'identité autonome from scratch. De plus, cette approche bottom-up permet ainsi de potentiellement pouvoir utiliser l'outil développé sur l'ensemble des serveurs d'identité fournit par le marché, avec un coût très limité. En outre, l'utilisation du proactive engine pour l'implémentation de la solution, nous a permis de tester l'utilisation du proactive engine dans un domaine pour lequel il n'a pas été spécialement conçu et ainsi démontrer que cela était tout à fait possible.

L'approche top-down n'aurait pas été réalisable en tenant compte des ressources dont nous disposions. Cette approche bien qu'elle ait ses avantages, comme le fait que l'on aurait pu créer un système totalement personnalisé, ne nous aurait pas fourni des résultats aussi rapidement dans le laps de temps accordé par le stage. De plus, cette solution n'aurait pas été adaptable aux systèmes déjà existants et aurait donc été un des seuls systèmes de gestion d'identité autonome.

# Conclusion

Nous avons commencé ce travail par l'exploration des différents concepts qui composent la gestion d'identité. Nous avons étudié les différentes parties prenantes ainsi que les exigences de chacune des parties présentes. Nous nous sommes ainsi rendu compte que la gestion des identités n'était pas une chose simple et que cela pouvait devenir un véritable casse-tête pour les grandes organisations disposant de serveurs bien chargés en identités. Cela nous a également permis de découvrir qu'il existe différents modèles de gestion de l'identité proposés aux organisations. Chacun des modèles présentant des avantages et des inconvénients tant du côté des organisations que des utilisateurs.

Après avoir étudié l'ensemble des composants de la gestion d'identité, nous nous sommes concentrés sur les problèmes de maintenance que l'on pouvait rencontrer sur un serveur d'identité. Nous avons identifié au total trois problèmes se répartissant sur l'ensemble des tâches de maintenance pouvant être effectuées. Les administrateurs système se trouvent notamment confronté quotidiennement à ces trois problèmes. Ces différents problèmes génèrent des coûts de maintenance importants pour les organisations et peuvent engendrer de graves problèmes de sécurité informatique au sein de l'organisation allant à l'encontre de la réglementation en matière du respect de la vie privée.

Afin d'apporter une solution à ces problèmes, nous avons voulu utiliser un nouveau mode de calcul appelé le proactive computing. Le proactive computing propose des solutions quant aux problèmes liés à l'explosion du nombre d'ordinateurs et il pose la question du comment l'homme va-t-il faire pour contrôler ces milliards d'ordinateurs. L'idée qui se cache derrière ces concepts est donc d'automatiser au maximum la gestion des ordinateurs pour ne devoir interagir avec eux qu'en cas de décisions critiques. Nous disposions d'un moteur implémentant les concepts dictés par le proactive computing en la présence du proactive engine développé par l'équipe du Professeur Zampunieris de l'Université de Luxembourg. C'est donc à l'aide de ce proactive engine que nous avons développé et implémenté notre application. Notre prototype tente d'apporter une solution aux problèmes évoqués en proposant un outil d'aide permettant de faciliter l'exécution des tâches de maintenance quotidienne d'un serveur d'identité (Active Directory) en utilisant les principes prescrits par le proactive computing.

La prototype développé prouve donc bien qu'il est possible de construire un outil d'aide à la gestion d'un serveur d'identité basé sur le proactive computing. Cet outil peut devenir totalement autonome dans le futur. Grâce a ce prototype, nous ouvrons le chemin à d'autres expériences, car nous n'avons fait qu'explorer et introduire les prémisses d'une solution aux problèmes liés à la maintenance des serveurs d'identité. Bien que la solution présentée au chapitre quatre reste simple, elle n'en est pas moins efficace. En effet, les premiers résultats

produits étaient plus qu'encourageants et ne font que nous convaincre que le proactive computing peut fournir une solution adaptée aux problèmes évoqués au cours du chapitre deux et même devenir une proposition alternative d'automatisation des systèmes d'informations. Dès lors que nous avons la confirmation que la solution proposée peut être viable, un futur travail pourrait être d'améliorer ce prototype en y ajoutant de nouvelles fonctions ou l'intégrer à d'autres systèmes.

Le travail réalisé est une première tentative concluante quant à la gestion d'un Active Directory par un moteur proactif. Ce travail portant sur un ensemble restreint de fonctionnalités, la porte est donc ouverte à l'implémentation de bien d'autres fonctionnalités additionnelles. De plus, lors de ce travail l'optique était de produire un outil capable d'aider le gestionnaire dans son travail et non pas de le remplacer. Toutefois, il serait facilement envisageable de poursuivre l'automatisation partielle ou totale de la gestion de l'annuaire de l'université avec un tel moteur. Ce qui a été effectué peut-être vu comme la première étape d'un projet plus ambitieux qui permettrait de rendre progressivement toute cette gestion quasiment autonome. Bien évidemment cela ne peut être fait sans concertation et sans l'accord des administrateurs système qui verraient leur métier actuel transformé vers un rôle de superviseur se focalisant sur les tâches les plus complexes qui requièrent une décision humaine.

Notons aussi que les données d'identités sont très sensibles et que de ce fait, laisser à l'heure actuelle la gestion complète à une machine ne nous parait pas raisonnable vu les enjeux sécuritaires importants liés à la gestion d'un serveur d'identité.

# Bibliographie


[1] Access control role engineering. http://fr.slideshare.net/ifourajit/access-control-role-engineering.

[2] Définitions : identité - dictionnaire de français larousse. http://www.larousse.fr/dictionnaires/francais/identit%C3%A9/41420.

[3] a.G. Ganek and T. a. Corbi. The dawning of the autonomic era. *IBM Systems Journal*, 42(1) :5–18, 2003.

[4] Mohammad A Al-Kahtani and Ravi Sandhu. A model for attribute-based user-role assignment. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 353–362. IEEE, 2002.

[5] ANSSI-Agence Nationale de la Sécurité des Systèmes d'Information. Recommandations de sécurité relatives aux mots de passe Informations. 2012.

[6] Identity Assurance and Expert Group. Liberty Identity Assurance Framework. *Business*, pages 1–128.

[7] Elisa Bertino and Kenji Takahashi. *Identity Management : Concepts, Technologies, and Systems*. Artech House, 2011.

[8] Matt Bishop. Password management. In *Proceedings of COMPCON*, volume 91, page 167, 1991.

[9] Kim Cameron. The laws of identity. *Microsoft Corp*, 2005.

[10] Kim Cameron and Michael B. Jones. Design rationale behind the identity metasystem architecture. *ISSE/SECURE 2007 - Securing Electronic Business Processes : Highlights of the Information Security Solutions Europe/SECURE 2007 Conference*, pages 117–129, 2007.

[11] David F Ferraiolo, Janet a Cugini, and D Richard Kuhn. Role-Based Access Control : Features and Motivations. *Proceedings of the 11th Annual Computer Security Applications Conference*, (JANUARY 1995) :241–248, 1995.

[12] Mario Frank, Joachim M Buhmann, and David Basin. On the Definition of Role Mining. *Proceeding of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT '10)*, pages 35–44, 2010.

[13] Shirley Gaw and Edward W Felten. Password management strategies for online accounts. In *Proceedings of the second symposium on Usable privacy and security*, pages 44–55. ACM, 2006.

[14] Salim Hariri, Bithika Khargharia, Houping Chen, Jingmei Yang, Yeliang Zhang, Manish Parashar, and Hua Liu. The Autonomic Computing Paradigm. *Cluster Computing*, 9(1) :5–17, 2006.

[15] Philip G Inglesant and M Angela Sasse. The true cost of unusable password policies : password use in the wild. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 383–392. ACM, 2010.

[16] ISO/IEC. ISO/IEC 24760-1 :2011 Information technology – Security techniques – A framework for identity – Part 1 : Terminology and concepts. 2011, 2011.

[17] ITU-T. NGN identity management framework, Recommendation Y.2720. *Y.2720*, pages 1–34, 2009.

[18] O Jeffrey and M David. The Vision of Autonomic Computing. *Journal of Ecology*, 90(January) :936–946, 2002.

[19] Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *Proceedings of the 2005 Australasian workshop on Grid computing and e-research-Volume 44*, pages 99–108. Australian Computer Society, Inc., 2005.

[20] Audun Jøsang and Simon Pope. User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference*, page 77. Citeseer, 2005.

[21] Audun Jøsang, Muhammed Al Zomai, and Suriadi Suriadi. Usability and privacy in identity management architectures. *Conferences in Research and Practice in Information Technology Series*, 68 :143–152, 2007.

[22] Philippe Lalanda, Julie A. McCann, and Ada Diaconescu. *Autonomic Computing - Principles, Design and Implementation.* Undergraduate Topics in Computer Science. Springer, 2013.

[23] Gilles I F Neyens, Remus-alexandru Dobrican, and Denis Zampunieris. Enhancing Mobile Devices with Cooperative Proactive Computing.

[24] Ravl S Sandhu. *Role-based Access Control '*, volume 46. 1998.

[25] Richard Shay, Abhilasha Bhargav-Spantzel, and Elisa Bertino. Password policy simulation and analysis. In *Proceedings of the 2007 ACM workshop on Digital identity management*, pages 1–10. ACM, 2007.

[26] Wayne C Summers and Edward Bosworth. Password policy : the good, the bad, and the ugly. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin, 2004.

[27] Gregory Tassey, Michael P Gallaher, Alan C O'Connor, and Brian Kropp. The economic impact of role-based access control. *Economic Analysis*, 2002.

[28] David Tennenhouse. Proactive computing. *Communications of the ACM*, 43(5) :43–50, 2000.

[29] K Tracy. Identity management systems. *Potentials, IEEE*, 27(6) :34–37, 2008.

[30] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem : finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 175–184. ACM, 2007.

[31] R. Want, T. Pering, and D. Tennenhouse. Comparing autonomic and proactive computing. *IBM Systems Journal*, 42(1) :129–135, 2003.

[32] Mark Weiser. The Computer for the 21st Century, 1991.

[33] Wikipedia. Proactif — Wikipedia, the free encyclopedia. `https://fr.wikipedia.org/wiki/Proactif`, 2015.

[34] Denis Zampunieris. Implementation of a Proactive Learning Management System. *E-Learn World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education*, pages 3145–3151, 2006.