Ingénierie des exigences de la sécurité informatique :

Revue de travaux de recherche de l'élicitation à la spécification

Meryem Kassou

ENSIAS, Université Mohammed V-Souissi, Madinat Al Irfane, Rabat, Maroc
meryem.kassou@gmail.com

Laila Kjiri

ENSIAS, Université Mohammed V-Souissi, Madinat Al Irfane, Rabat, Maroc
kjiri@ensias.ma

Résumé

L'expansion des réseaux d'un côté et l'importance de l'information de l'autre côté ont fortement contribué à ce que la sécurité joue un rôle important dans le développement d'applications. Cependant, les exigences de sécurité ont pendant longtemps été reléguées à la deuxième place comparativement aux exigences fonctionnelles, ce qui peut endommager les applications développées.

L'un des défis de la recherche actuelle est donc de développer des méthodes d'expression et de modélisation des exigences de sécurité qui sont expressives, simples, fiables et non coûteuses.

Cet article a pour but de passer en revue des travaux de recherche relatifs aux approches d'élicitation et de spécification des exigences de sécurité dans le contexte d'un processus de développement logiciel. Il y est question également de leur analyse, de leur classification en catégories et de la mise en évidence de leur contribution à l'ingénierie des exigences de sécurité ainsi que de leurs limites.

Abstract

Nowadays Security plays an important role in the development of applications because of network expansion on one side and of the importance of information on the other side. However, security requirements have long been relegated to second place compared to the functional requirements which can cause some damage to the applications.

One of the challenges of current research in the area of requirement engineering is to develop simple, expressive, reliable and costless methods for expressing and modelling security requirements.

The purpose of this article is to review research studies related to approaches for eliciting and specifying security requirements in the context of software development process. It aims then to analyse and classify them into categories and to highlight their contribution to security requirements engineering field and their limits.

Mots-clés

Ingénierie des exigences, Sécurité, Elicitation, Spécification formelle, Framework de modélisation

Keywords

Requirement Engineering, Security, Elicitation, Formal Specification, Modelling framework

1. Introduction

L'Ingénierie des exigences est la première étape fondamentale dans tout projet de développement d'application. Les méthodes de développement logiciel divisent les exigences logicielles en deux catégories : les exigences fonctionnelles et les exigences non fonctionnelles (Glinz, 2005). Si les exigences fonctionnelles décrivent les fonctions que

l'application doit accomplir, les exigences non fonctionnelles se concentrent sur la manière dont ces fonctions sont exécutées. La sécurité est considérée comme une exigence non fonctionnelle.

Ces dernières années, l'expansion des réseaux d'une part et l'importance de l'information d'autre part ont fortement contribué à ce que les exigences de sécurité jouent un rôle clé dans le développement d'applications (Firesmith, 2003b). Cependant, les exigences de sécurité ont pendant longtemps été reléguées à la deuxième place comparativement aux exigences fonctionnelles (Matoussi et Laleau, 2008).

En effet, intégrer la sécurité au sein d'un système a toujours commencé par l'identification des besoins de sécurité après la conception de l'application. Cela pose un problème de cohérence et de cohabitation entre les exigences fonctionnelles et les exigences de sécurité. En outre, l'implémentation de l'application peut inclure des mécanismes de sécurité qui ne sont tout simplement pas nécessaires. Par ailleurs, la mise en œuvre des mécanismes de sécurité peut empêcher le fonctionnement de l'application (Giorgini, Massacci, *et al.*, 2005).

L'un des défis de la recherche actuelle est donc d'intégrer les exigences de sécurité avec le processus standard des exigences. Ainsi, les exigences de sécurité peuvent-elle être formulées et intégrées dans la conception des systèmes à un haut niveau d'abstraction en même temps que les exigences fonctionnelles. De cette manière, il devient possible de développer des applications sécurisées qui soient conçues dans le but de prévenir les violations d'une politique de sécurité (Giorgini, Massacci, *et al.*, 2005).

Plusieurs travaux sont consacrés aux méthodes d'expression des exigences de sécurité à un haut niveau d'abstraction. Nous nous intéressons dans cet article aux travaux de recherche sur les approches d'élicitation et de spécification des exigences de sécurité, dans le contexte d'un processus de développement logiciel.

Nous définirons donc, dans la première section de cet article, le processus d'ingénierie d'exigences ainsi que les particularités des exigences de sécurité. Ensuite, nous nous intéresserons à deux phases de ce processus appliqué au contexte de la sécurité, à savoir l'élicitation et la spécification des exigences de sécurité. A cet effet, nous passerons d'abord en revue, dans la deuxième section, des travaux de recherche sur les techniques d'élicitation des exigences de sécurité et expliquer pourquoi elles sont indissociables des méthodes de spécification. Puis, en section 3, nous focaliserons sur les approches de spécification des exigences de sécurité qui seront classées par catégorie. Notre travail se termine par une synthèse des approches de spécification des exigences de sécurité qui fait ressortir leur contribution à l'ingénierie des exigences de sécurité ainsi que leurs faiblesses.

2. Définitions et contexte

2.1 Exigence et processus d'ingénierie des exigences

Selon IEEE Std. 1233-1998 (1998), une exigence est une « condition ou capacité qui doit être remplie ou possédée par un système ou un de ses composants pour satisfaire à un contrat, une norme, une spécification ou tout document imposé de façon formelle ».

Une exigence formalisée doit être abstraite (elle est indépendante de la méthode de mise en œuvre), non ambiguë (elle est énoncée de manière à n'être interprétable que d'une seule manière), traçable (il est possible d'établir une relation entre la déclaration précise des besoins du client et les énoncés spécifiques de la définition du système) et vérifiable (elle doit offrir un moyen de prouver que le système satisfait à son énoncé).

Une exigence formalisée est en fait le résultat du processus d'Ingénierie des Exigences qui est la première étape importante dans le développement d'une application. Ce processus inclut plusieurs phases dont le découpage diffère selon plusieurs définitions (Nuseibeh et Easterbrook, 2000), (Kotonya et Sommerville, 1998). Cependant, typiquement un processus d'ingénierie d'exigences est un processus itératif qui est constitué des phases d'Elicitation, d'Analyse, de Spécification, de Validation et de Gestion.

L'élicitation des exigences consiste en la collecte et le développement d'exigences à partir d'une variété de sources et de parties prenantes. L'analyse se focalise sur l'examen, la compréhension des exigences élicitées et leur vérification pour la qualité en termes d'exactitude, de complétude, de clarté et de consistance. La spécification n'est autre que l'enregistrement et la documentation des exigences de sorte que celles-ci soient utilisables par les parties prenantes, et en particulier, par les développeurs qui doivent concevoir et construire le système. Enfin, la validation est la confirmation de la qualité des exigences et de leur conformité aux besoins des parties prenantes. La gestion, quant à elle, est exécutée tout au long du processus d'ingénierie des exigences. Elle inclut les activités de contrôle de l'évolution et de changement des exigences, de contrôle de version, de traçabilité des exigences et les statuts des exigences.

2.2. Particularités des exigences de sécurité

L'expression d'une exigence de sécurité se base sur l'analyse des biens et des services qui doivent être protégés et sur l'analyse des menaces de sécurité desquelles ces biens et services doivent être protégés (Firesmith, 2003*a*).

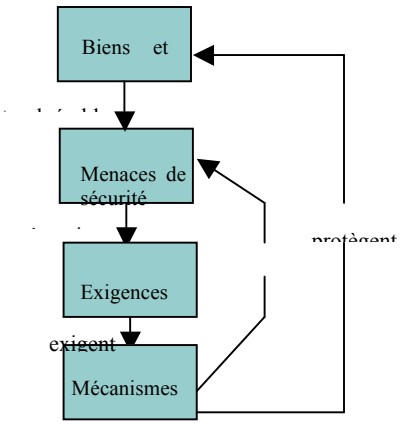


Figure 1. Menaces, exigences et mécanismes de sécurité. Source: (Firesmith, 2003*a*)

La Figure 1 illustre bien la relation qui existe entre biens et services, menaces de sécurité, exigences de sécurité et mécanismes de sécurité et souligne l'importance des exigences de sécurité.

L'ingénierie des exigences de sécurité a suscité chez la communauté de recherche un intérêt particulier en raison des pertes causées par des applications mal sécurisées, du fait que la sécurité n'était pas intégrée dès le début du cycle de développement ou qu'elle était mal spécifiée.

Selon Van Lamsweerde (2004), il y a trois raisons qui expliquent ce fait :

- tout d'abord, les premières phases du développement des logiciels donnent la priorité à l'élaboration des exigences fonctionnelles au détriment des

exigences non-fonctionnelles telles que la sécurité en vue d'obtenir un produit qui fonctionne en un court laps de temps.

- La deuxième raison est l'absence d'un mécanisme constructif et efficace pour l'élaboration d'exigences de sécurité d'une manière complète, cohérente et claire.
- La troisième raison est l'absence d'une approche précise et bien définie qui permet de produire la conception et l'implémentation des exigences de sécurité tout en assurant une prise en charge correcte de toutes les exigences et en permettant une traçabilité des exigences lors des différentes phases de développement.

Par ailleurs, selon la définition d'une exigence formalisée fournie par (IEEE Std. 1233-1998, 1998) et dans le contexte de la sécurité, certaines difficultés paraissent évidentes :

- afin d'être indépendante de la méthode de mise en œuvre, l'exigence de sécurité doit être spécifiée durant les premières étapes de développement.
- Une exigence de sécurité, rédigée dans un langage informel, peut s'avérer incohérente et ambiguë et donc difficilement vérifiable.
- Le langage de spécification de l'exigence de sécurité doit permettre une traçabilité très forte entre les phases de spécification, de conception puis d'implémentation.

L'un des défis de la recherche consacrée aux exigences de sécurité a été de développer des méthodes et des langages qui permettent de spécifier les exigences de sécurité de manière claire, expressive et non ambiguë.

Par ailleurs, l'exigence de sécurité est souvent liée à un environnement technique qui restreint le recueil des besoins à un jargon de spécialistes de sécurité et limite l'apport des autres parties prenantes en l'absence de méthodes pour guider l'opération d'identification des exigences de sécurité.

Dans la suite de cet article, nous avons jugé important de passer en revue aussi bien les approches d'élicitation des exigences de sécurité que les approches de spécification de ces exigences.

3. Elicitation des exigences de sécurité

Plusieurs techniques permettent de guider l'opération d'identification des exigences de sécurité. Nous les avons passées en revue et avons sélectionné celles qui, selon nous, sont les mieux structurées et les plus directives (nous n'avons pas mentionné par exemple les techniques basées sur les séances de brainstorming ni celles basées sur les interviews). Nous proposons de les classer selon les catégories suivantes : approches basées sur les modèles (Hadavi, Hamishagi, *et al.*, 2008), approches basées sur les scénarios (Giorgini, Massacci, *et al.*, 2005) et approches basées sur les meilleures pratiques.

3.1. Approches fondées sur les modèles

Ces approches utilisent des modèles et des patrons comme outil de connaissance ou comme support pour produire des exigences de sécurité.

Les modèles d'attaque (*attack pattern*) (Barnum et Sethi, 2006) constituent un outil de connaissance qui aide à la conception, au développement et au déploiement des logiciels sécurisés. Ils découlent de la notion de « *design pattern* » appliquée dans un contexte « destructif » plutôt que « constructif ». En effet, les design patterns aident à résoudre les problèmes récurrents qui peuvent être rencontrés lors du développement d'applications. Les modèles d'attaque permettent aux développeurs d'acquérir une meilleure connaissance des attaques possibles (en se mettant à la place d'un attaquant) et des meilleures méthodes pour contrer ces attaques.

Le *System Security Engineering - Capability Maturity Model* (SSE-CMM) est un modèle qui permet d'évaluer la maturité des processus de sécurité d'un système d'information à travers un barème de cinq niveaux (Phillips, 2003). Ce modèle permet de produire quelques lignes directrices pour la production d'exigences de sécurité.

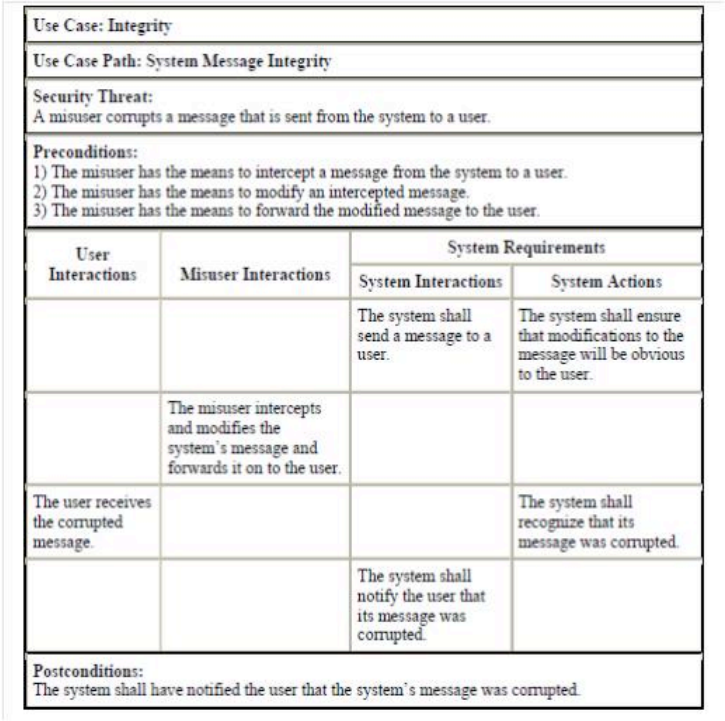
3.2. Approches fondées sur les scénarios

Ces approches sont dérivées des *use case* et permettent, à travers des cas d'utilisation dans le cadre de la sécurité, de déduire des exigences de sécurité.

Les *Misuse case* et les *Abuse Case* sont une extension conceptuelle des *use case* décrivant les actions qui ne doivent pas être exécutées car elles pourraient engendrer des dommages aux ressources du système et aux parties prenantes. Ils présentent une solution pour l'obtention des exigences de sécurité (Sindre et Opdahl, 2001), (McDermott et Fox, 1999).

Les *Mitigation case* sont également utilisés pour exprimer des exigences de sécurité. Les *Mitigation case* permettent de déterminer les exigences et les contre-mesures qui doivent être prises contre les scénarios offensifs des *Misuse case*. Dans ce processus, les *use case* et *Misuse case* principaux sont d'abord produits. Ensuite, chacun de ces cas est divisé en *Small case*. Pour chaque nouveau cas de *Misuse case*, un cas *Mitigation case* est présenté comme une exigence (Hadavi, Hamishagi, *et al.*, 2008).

Firesmith (2003*a*) considère que l'utilisation des *Misuse case* est plus efficace pour analyser les menaces que pour définir les exigences de sécurité. Il définit les *Security case* qui permettent de déterminer les exigences de sécurité en se basant sur les *Misuse case*.



Use Case: Integrity			
Use Case Path: System Message Integrity			
Security Threat: A misuser corrupts a message that is sent from the system to a user.			
Preconditions: 1) The misuser has the means to intercept a message from the system to a user. 2) The misuser has the means to modify an intercepted message. 3) The misuser has the means to forward the modified message to the user.			
User Interactions	Misuser Interactions	System Requirements	
		System Interactions	System Actions
		The system shall send a message to a user.	The system shall ensure that modifications to the message will be obvious to the user.
	The misuser intercepts and modifies the system's message and forwards it on to the user.		
The user receives the corrupted message.			The system shall recognize that its message was corrupted.
		The system shall notify the user that its message was corrupted.	
Postconditions: The system shall have notified the user that the system's message was corrupted.			

Figure 2. Exemple de *Security Case.* Source : (Firesmith, 2003*a*)

Les Anti-exigences (*anti-requirements*) permettent d'exprimer les exigences relatives aux utilisateurs malveillants ou pirates. Une Anti-exigence est satisfaite lorsque la menace imposée par un attaquant met le système en une ou plusieurs catégories de risques (Hadavi, Hamishagi, *et al.*, 2008).

Pour illustrer l'approche basée sur les scénarios pour l'élicitation d'exigences de sécurité, nous avons extrait de (Firesmith, 2003*a*) un exemple concret de S*ecurity use case* présenté en figure 2.

Ce scénario décrit les exigences de sécurité déduites du cas d'un utilisateur malveillant qui corrompt un message envoyé du système à l'utilisateur.

3.3. Approches fondées sur les meilleures pratiques

Ces approches se fondent sur l'expérience de plusieurs équipes pour fournir une liste d'exigences de sécurité.

CLASP, *Comprehensive, Lightweight Application Security Process,* (Viega, 2005) est une méthode structurée pour l'extraction des exigences de sécurité dès les premières étapes de développement. Elle s'est fondée sur l'expérience de plusieurs équipes de développement et sur la compilation de *best practices* pour créer un ensemble complet d'exigences de sécurité.

Les normes de sécurité, fondées sur les meilleures pratiques comme la norme ISO 17799/27001, sont d'autres ressources appropriées pour le développement des exigences de sécurité. Elles peuvent être utilisées pour capturer les exigences de sécurité en utilisant des politiques d'organisation (Su, Bolzoni, *et al.*, 2006).

Tableau 1. Caractéristiques des approches d'élicitation des exigences de sécurité

Approches d'élicitation	**Description**	**Points forts**	**Points faibles**
Approches fondées sur les modèles	Elles utilisent des modèles et des patrons comme outil de connaissance ou comme support pour produire des exigences de sécurité	- Base de connaissance (réutilisation, richesse) - Exigences de sécurité détaillées	Trop spécialisées
Approches fondées sur les scénarios	Elles sont dérivées des *use case* et permettent, à travers des cas d'utilisation dans le cadre de la sécurité, de déduire des exigences de sécurité	- Faciles à utiliser par les parties prenantes - Complets (élicitation, analyse, spécification)	Exigences de sécurité de haut niveau
Approches fondées sur les meilleures pratiques	Elles se fondent sur l'expérience de plusieurs équipes pour fournir une liste d'exigences de sécurité	- Standards - Support facile à l'élicitation (*checklis*t)	Exigences de sécurité de haut niveau

Nous reprenons dans le tableau 1 les approches d'élicitation des exigences de sécurité présentées dans cette section et précisons les points forts et les points faibles que nous avons relevés pour chacune d'elles.

Comme la phase d'élicitation est étroitement liée aux autres phases du processus d'ingénierie des exigences de sécurité et particulièrement à la phase de spécification, les approches utilisées pour l'élicitation des exigences de sécurité et celles utilisées pour leur spécification sont dans une grande mesure interdépendantes (Nuseibeh et Easterbrook, 2000).

4. Spécification des exigences de sécurité

4.1 Critères de classification et d'analyse

La spécification d'une exigence de sécurité utilise des langages et des outils d'expression et de modélisation des exigences de sorte que celles-ci soient utilisables par les parties prenantes, et en particulier, par les développeurs qui doivent concevoir et construire le système. Plusieurs travaux ont proposé des approches qui diffèrent les unes des autres.

Certains travaux de recherche ont utilisé les *frameworks* de modélisation des exigences (fonctionnelles et non fonctionnelles) pour le cas particulier des exigences de sécurité. D'autres travaux ont étendu ces *frameworks* avec des attributs de sécurité. Par ailleurs, certains résultats de recherche présentent des *frameworks* construits à partir de l'utilisation combinée de méthodes formelles et semi-formelles.

Dans ce qui suit, nous présentons et analysons ces différentes contributions.

4.2 Utilisation des *frameworks* de modélisation existants

Dans cette approche, la méthode de spécification utilise les *frameworks* disponibles comme i*/tropos pour modéliser certaines exigences de sécurité. Les fonctionnalités d'analyse du *Framework* sont ensuite utilisées pour la conception et la mise en œuvre.

Certains travaux ont été réalisés selon cette approche. Liu, Yu, *et al.* (2003) utilise i*/tropos pour modéliser les exigences de sécurité et de confidentialité et pour analyser les dépendances et ainsi vérifier si le système est sécurisé. Tropos est une méthodologie de développement pour construire des systèmes logiciels orientés agents. Elle utilise les concepts d'acteur, d'objectif, de plan, de ressources et de dépendance intentionnelle pour définir les obligations des acteurs (*dependees*) à d'autres acteurs (*dependers*). Tropos accorde beaucoup d'attention aux premières exigences en soulignant la nécessité de comprendre pourquoi et comment le système cible répondrait aux objectifs organisationnels.

He et Anton (2003) présentent un *framework* orienté objectif pour la modélisation des exigences de confidentialité dans le processus d'ingénierie des rôles. L'objectif de ce *framework* est de combler le fossé entre les pratiques de confidentialité des entreprises et les préférences des utilisateurs qui peuvent être en conflit. Les exigences de confidentialité sont modélisées comme des contraintes relatives aux permissions et aux rôles.

Le NFR Framework (*Non Functional Requirement Framework*) est un *framework* de modélisation d'objectifs (Mylopoulos, Chung *et al.,*1999). Il analyse les *Soft goals* qui sont des exigences non fonctionnelles (dont la sécurité) difficiles à exprimer et sur lesquelles les parties prenantes se sont mises d'accord. Le *NFR Framework* explicite la relation entre les Exigences Non Fonctionnelles (ENF) et les décisions de conception. Cela permet de mieux comprendre l'impact de chaque décision de conception. L'intérêt principal de ce *framework* est qu'il peut être réutilisé par d'autres modèles pour manipuler les ENF (Matoussi et Laleau, 2008).

Analyse :

L'approche d'utilisation des *frameworks* disponibles a deux avantages (Giorgini, Massacci, *et al.*, 2005). D'une part, elle est peu onéreuse car ni un nouveau langage, ni un nouveau Framework pour la modélisation et l'analyse ne sont nécessaires, et d'autre part, toutes les caractéristiques du Framework de modélisation sont immédiatement utilisables. Si le Framework est équipé d'une sémantique formelle et de procédures de raisonnement formelles, ces dernières seront également héritées.

L'inconvénient majeur de cette approche est qu'elle ne répond pas à un large éventail d'exigences de sécurité. En effet, ne disposant pas de structures conceptuelles adéquates, elle ne permet pas, par exemple, de spécifier des contraintes d'autorisation ou de confidentialité. Par ailleurs, le lien entre les exigences de sécurité et les exigences fonctionnelles doit être introduit par le concepteur par des prédicats ou par des relations.

4.3 Amélioration (extension) des *frameworks* existants

Selon cette approche, les *frameworks* de modélisation sont améliorés avec une nouvelle structure conceptuelle (syntaxe et sémantique) afin de répondre à un plus large éventail d'exigences de sécurité et afin de transmettre des exigences de sécurité plus complètes et plus exactes pour l'analyse du système (Hadavi, Hamishagi *et al.,* 2008) (Giorgini, Massacci *et al.,* 2005). Par conséquent, les fonctionnalités d'analyse, de conception et de mise en œuvre du *framework* sont révisées ou améliorées pour prendre en charge des exigences de sécurité plus précises. Parmi les travaux utilisant cette approche, nous citons ci-dessous ceux qui ont étendu les *framework* UML, Tropos et KAOS, afin de modéliser des propriétés de sécurité.

UMLsec (Jurjens, 2001) est une extension d'UML pour modéliser des propriétés de sécurité comme la confidentialité et le contrôle d'accès. UMLsec inclut toutes les analyses UML et artéfacts de conception tels que les diagrammes d'activités, de déploiement, de séquences et d'états. La conception du système peut ainsi être représentée avec UMLsec. Les exigences de sécurité sur les modèles UMLsec sont ensuite vérifiées par *model-checking*.

Secure Tropos (Mouratidis, Manson *et al.*, 2003) est une extension de la méthode Tropos dont le but est de permettre de capturer de manière adéquate les exigences de sécurité. Elle a introduit l'utilisation de nouveaux concepts comme les diagrammes de sécurité, les contraintes de sécurité et les capacités de sécurité comme concepts de base.

KAOS, *Knowledge Acquisition in autOmated Specification*, (Van Lamsweerde, 2001) est une approche semi formelle pour l'élicitation, l'analyse et la modélisation des exigences. Le modèle d'exigences de cette approche est fondé sur la logique du premier ordre temporelle. KAOS a ensuite été étendu de sorte qu'il puisse manipuler des exigences non fonctionnelles telles que les exigences de sécurité. La méthode générale pour l'élaboration d'exigences de sécurité KAOS est fondée sur la construction progressive et sur la spécification de deux modèles concurrents (Mansour, 2009) :

- un modèle intentionnel du système à construire qui couvre aussi bien le logiciel que son environnement,
- un anti-modèle dérivé du modèle qui montre comment les spécifications des éléments du modèle peuvent être menacées, pourquoi elles peuvent l'être et par qui. Les contre-mesures qui sont dérivées pour répondre aux menaces permettent de définir de nouvelles exigences de sécurité qui seront introduites dans le modèle initial.

Un autre travail (Lodderstedt, Basin *et al.*, 2002) présente un langage de modélisation fondé sur UML, SecureUML, visant à intégrer l'information de contrôle d'accès aux modèles d'applications définis avec UML. L'approche est axée sur les politiques de modélisation de contrôle d'accès et leur intégration dans un processus de développement de logiciels fondé sur les modèles. Le langage s'appuie sur une extension du modèle de contrôle d'accès RBAC (*Role Based Access Control*). RBAC est un modèle de contrôle d'accès où les utilisateurs et leurs privilèges sont découplés par rôles (Sandhu, Coyne *et al.*, 1996). SecureUML a ajouté la notion de contraintes d'autorisation pour définir les pré-conditions pour régir l'accès à une opération. (Lodderstedt, Basin *et al.*, 2002) présente l'exemple concret d'un *scheduler* sécurisé spécifié avec SecureUML que nous avons repris en figure 3. Il s'agit d'une application constituée de deux composantes : Calendrier (*calendar*) et entrée (*entry*). Un

calendrier peut contenir plusieurs entrées, chacune représentant un rendez-vous avec une date de début, une date de fin et un emplacement. Chaque entrée est détenue par un utilisateur dont le nom est stocké dans l'attribut *owner*. Les expressions supplémentaires indiquées dans le schéma sont utilisés pour exprimer des informations de contrôle d'accès.

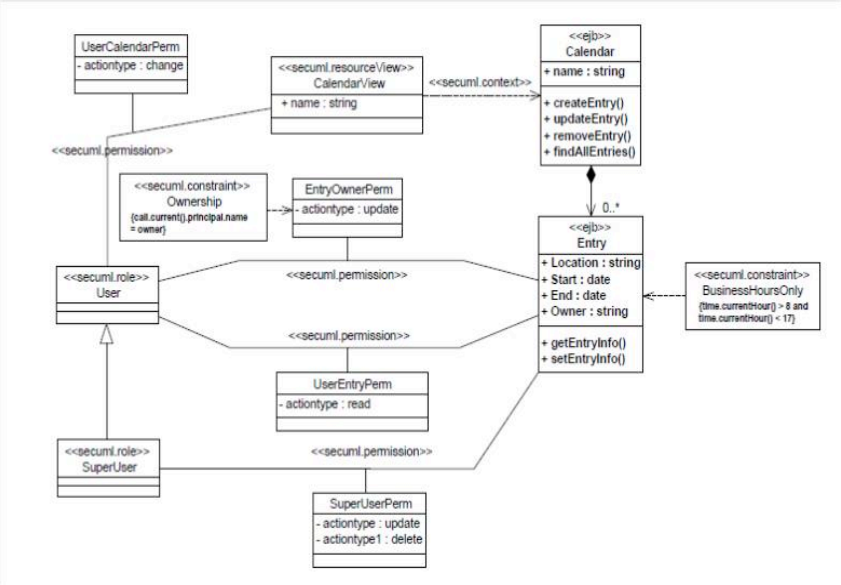


Figure 3. Exemple de *scheduler* sécurisé. Source : (Lodderstedt, Basin, *et al.*, 2002)

Analyse :

L'avantage de l'approche d'extension des *frameworks* de modélisation est qu'elle permet de répondre à un plus large éventail d'exigences de sécurité et de transmettre des exigences de sécurité plus complètes et plus exactes pour l'analyse du système. L'inconvénient est que les fonctionnalités d'analyse, de conception et de mise en œuvre du *framework* sont révisées ou améliorées pour prendre en charge des exigences de sécurité plus précises.

4.1 Construction de Framework de modélisation à partir de méthodes semi-formelles et formelles

Les méthodes formelles de spécification des exigences permettent de spécifier de manière rigoureuse, non ambiguë et vérifiable les exigences de sécurité. Mais un problème récurrent est leur coût élevé et leur difficulté d'utilisation.

Les méthodes semi formelles de spécification des exigences sont des méthodes fondées sur des langages en général graphiques qui disposent d'une syntaxe claire et d'une sémantique ambiguë. Elles sont donc faciles à utiliser mais leurs faiblesses sont dues au caractère ambigu des notations semi formelles, à leur forme limitée de spécification et d'analyse et au fait qu'elles ne disposent pas d'outils de preuve (Mansour, 2009).

Les méthodes semi formelles et formelles peuvent être combinées de manière à se compléter. Dans ce qui suit, nous décrivons des approches qui utilisent les méthodes formelles en combinaison avec des méthodes semi formelles et ce dans le cadre de *frameworks* bien définis.

Le Formal Design Analysis Framework (FDAF) (Dai et Cooper, 2005) est un *framework* d'architecture orienté aspects qui permet la conception et l'analyse automatique des propriétés non fonctionnelles des architectures logicielles, en utilisant UML et un ensemble de méthodes formelles existantes. Dans le FDAF, les propriétés non fonctionnelles, y compris la sécurité,

sont représentées comme des aspects. FDAF traduit une conception UML étendue semi formelle en une notation formelle.

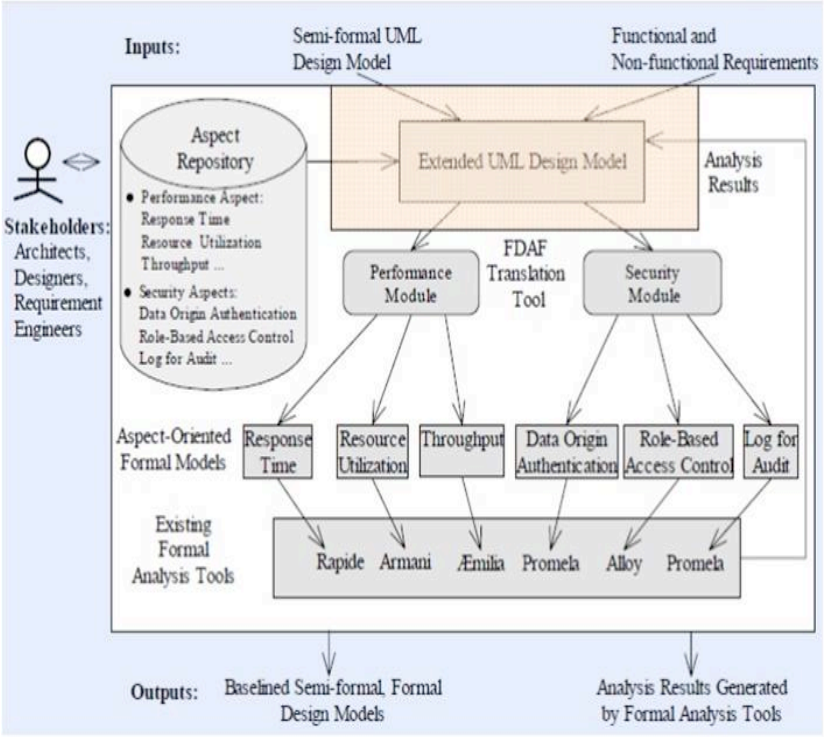


Figure 4. Formal Design Analysis Framework. Source : (Dai et Cooper, 2005)

La limite majeure de FDAF est que la variété des notations formelles utilisées pour formaliser les ENF constitue un obstacle pour étudier les différentes interactions entre les ENF. En conséquence, l'utilisateur ne peut pas détecter les conflits entre les différents types d'ENF (Matoussi et Laleau, 2008).

Une autre proposition de formalisation des Exigences Non Fonctionnelles (Botella, Burgués *et al.*, 2001) est le langage Nofun (acronyme pour Not Functional) qui décrit de manière formelle et évalue la qualité des logiciels à base de composants, en se fondant sur le *framework* des standards de qualité ISO/IEC. Un modèle de qualité est défini par des caractéristiques générales de logiciels qui sont affinées en sous-caractéristiques puis en attributs, dans une hiérarchie à plusieurs niveaux. La sécurité fait partie des sous-caractéristiques définies par ce modèle de qualité. Les exigences de qualité en général et celles de la sécurité en particulier sont définies comme des restrictions dans le modèle de qualité.

Le langage NoFun se compose de trois parties différentes. Dans la première partie, les caractéristiques et les attributs de qualité des composants sont définis. Dans la deuxième partie, des valeurs sont affectées aux attributs de base de la qualité des composants. Troisièmement, les exigences de qualité dont font partie les exigences de la sécurité, peuvent être spécifiées pour les composants, à la fois indépendamment et dépendamment du contexte (propriétés de qualité générales ou dépendantes d'un domaine informatique particulier). Ce langage contient des mécanismes de structuration et de définition de types d'éléments qui permettent de définir des modèles de qualité de manière détaillée. Cependant, il n'offre pas de possibilité de détecter les conflits possibles entre les différentes exigences de qualité.

Mansour (2009) utilise la méthode B pour spécifier les exigences de sécurité. Ce travail propose une approche formelle qui tire les spécifications de conception d'un ensemble d'exigences de sécurité modélisées en utilisant une extension de KAOS pour la sécurité.

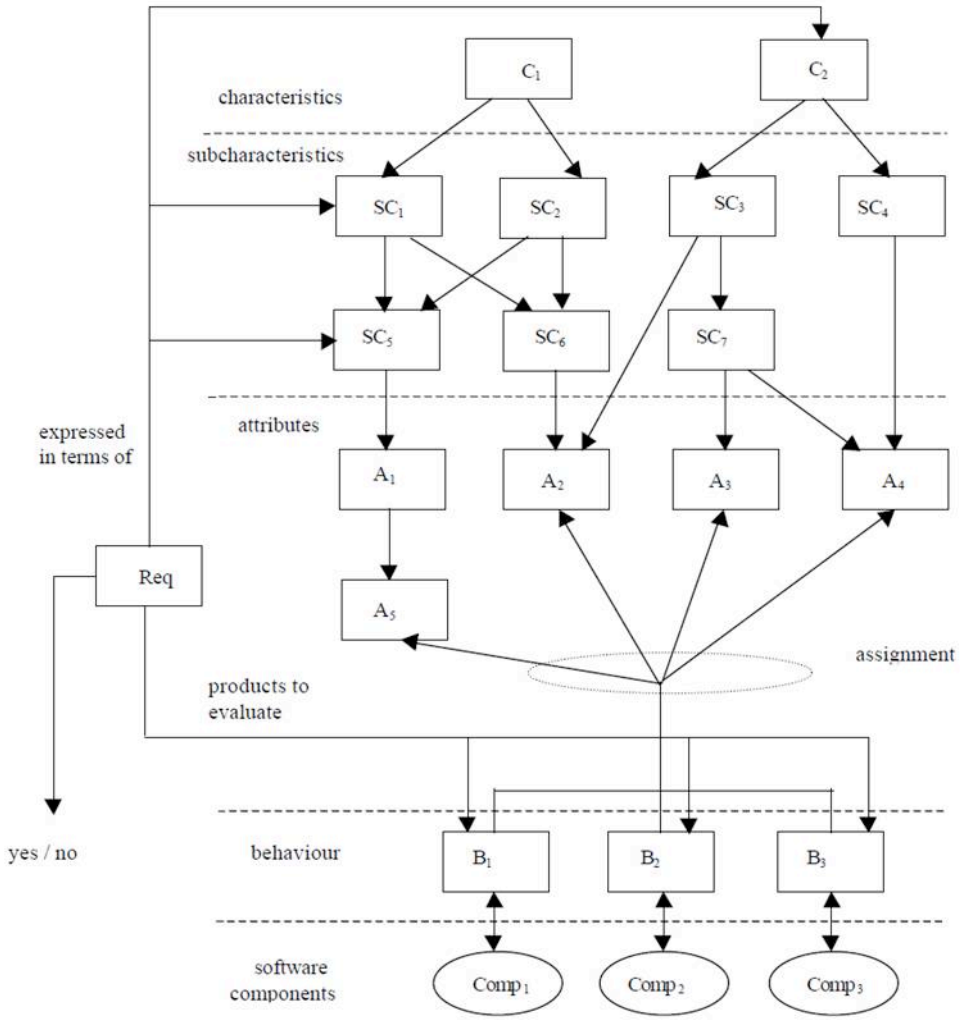


Figure 5. Modèle de qualité ISO/IEC représenté avec NoFun. Source : (Botella *et al.*, 2001)

La première étape de l'approche consiste à transformer le modèle d'exigences de sécurité KAOS en un modèle abstrait B. La deuxième étape consiste à affiner le modèle en utilisant les mécanismes de raffinement B afin de générer des spécifications de conception et d'exécution. Ce processus de transformation et de raffinement est détaillé en Figure 6.

L'intérêt de cette approche est qu'elle permet d'utiliser le langage formel et les mécanismes de raffinement de la méthode B. Ceci permet de préserver les propriétés du système déjà prouvées dans les modèles de niveau supérieur. Par ailleurs, la méthode formelle intervient à un stade dans le processus de spécification où la frontière entre le logiciel et son environnement a déjà été établie : les exigences de sécurité sont d'abord spécifiées par KAOS qui tient compte de cette interaction (Darimont et Van Lamsweerde, 1996).

Ledang et Souqières (2002) suggèrent une démarche de formalisation des spécifications UML comme suit:

- Traduction des *use case* en B. Chaque cas d'utilisation est modélisé comme une opération B.
- Modélisation des opérations de classe. Chaque opération de classe est modélisée comme une opération B dans une machine abstraite.
- Modélisation des diagrammes d'états qui se fait en deux étapes, la création d'une opération abstraite B pour chaque événement puis la mise en œuvre (ou raffinement) de l'opération B.

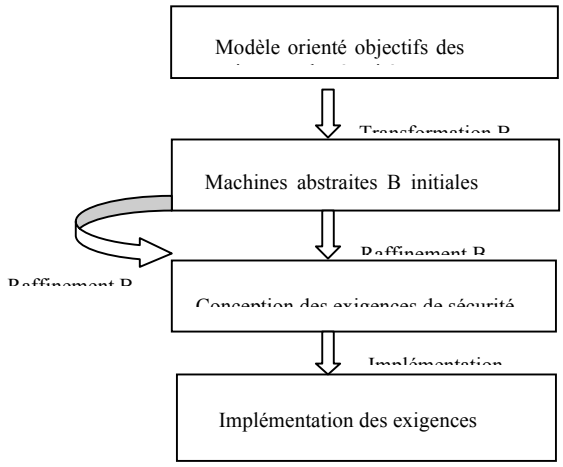


Figure 6. Dérivation formelle des spécifications de la sécurité. Source : (Mansour, 2009)

Le principal objectif pour traduire UML en B est de permettre une analyse formelle des spécifications UML à travers leurs spécifications correspondantes en B. L'autre objectif est d'utiliser les spécifications UML comme un outil pour faciliter la réalisation de spécifications en B.

Analyse :

L'utilisation combinée de méthodes semi-formelles et formelles permet de spécifier des exigences de sécurité en bénéficiant des avantages des deux méthodes : la facilité d'utilisation de la première et la rigueur de la deuxième.

Cependant, selon Van Lamsweerde (2004) et Darimont *et al.* (1996), les techniques de spécification formelles souffrent de lacunes qui expliquent pourquoi elles ne sont pas entièrement adéquates pour la phase critique d'élaboration et d'analyse des exigences. Ces lacunes sont décrites comme suit:

- La grande majorité des techniques se concentre sur la modélisation et la spécification du logiciel seul sans tenir compte de son environnement. En effet, les méthodes d'ingénierie des exigences ont besoin d'ontologies qui offrent des abstractions de haut niveau tels que *goal*, *goal refinements*, *agent, responsibility assignments* et ne peuvent se limiter à des abstractions de programmation comme *operation* ou *state.*
- Il n'existe pas de méthode constructive pour modéliser ou spécifier des systèmes complexes. Le problème n'est pas simplement de traduire les instructions en langage naturel dans des modèles semi formels et/ou des spécifications formelles. L'Ingénierie des exigences, en général, prévoit que les exigences complexes soient élicitées, élaborées, structurées, interdépendantes et négociées.

Si la première lacune peut être comblée par une utilisation combinée de méthodes semi formelles qui seront utiles lors de la phase d'élaboration des exigences de sécurité, la deuxième lacune est beaucoup plus liée à la non disponibilité d'outils d'analyse et de gestion des exigences de sécurité.

5. Synthèse

Nous avons passé en revue plusieurs approches de spécification des exigences de sécurité. Certains travaux de recherche ont utilisé les *frameworks* de modélisation des exigences

(fonctionnelles et non fonctionnelles) pour le cas particulier des exigences de sécurité. Cette approche s'avère intéressante dans la mesure où elle permet d'utiliser des outils qui ont déjà fait leurs preuves et qui sont donc faciles à utiliser car maîtrisés par les participants au processus d'élicitation, d'expression et de spécification des exigences. Cependant, ces outils présentent des limites qui sont liées à :

- leur expressivité car ils ne possèdent pas suffisamment d'attributs pour spécifier correctement des exigences de sécurité,
- leur degré de formalisme si ces outils ne se fondent pas sur un langage formel.

D'autres recherches se sont donc consacrées à la manière de pallier ces lacunes.

- en étendant ces *frameworks* avec une expressivité adéquate liée aux exigences de sécurité,
- en utilisant des langages formels qui permettent de spécifier de manière rigoureuse, non ambiguë et vérifiable les exigences de sécurité.

Le tableau 2 résume les principales caractéristiques des approches de spécification des exigences de sécurité analysées.

Tableau 2. Avantages et limites des approches de spécification des exigences de sécurité

Approches de spécification	**Avantages**	**Limites**
Frameworks de modélisation existants	Coût et facilité d'utilisation.	Expressivité limitée du langage de spécification des exigences de sécurité.
Extension des *frameworks* de modélisation	Expressivité du langage de spécification des exigences de sécurité.	Révision des outils du *Framework* pour prendre en charge des exigences de sécurité plus précises.
Combinaison de méthodes semi-formelles et formelles	Facilité d'utilisation du semi formel et rigueur du formel.	Coût des méthodes formelles et indisponibilité d'outils d'analyse et de gestion des exigences

Cette analyse a permis de relever que toutes les approches de spécification des exigences de sécurité ont des avantages et des limites. La justification du choix d'une approche de spécification est donc d'abord liée à son contexte d'utilisation et à ses contraintes. Par ailleurs, elle est également liée à la méthode d'élicitation utilisée.

6. Conclusion

Les exigences de sécurité jouent un rôle important dans le développement des logiciels. L'Ingénierie des exigences de sécurité implique le développement de méthodes et d'outils qui permettent la construction de spécifications d'exigences de sécurité complètes, cohérentes et claires.

Cet article présente une revue d'approches de spécification des exigences de sécurité et une analyse de leurs apports et faiblesses. Nous avons également cité quelques approches d'élicitation des exigences de sécurité car nous avons estimé que dans le contexte de la sécurité, elles sont parfois indissociables des approches de spécification des exigences de sécurité.

Idéalement, une approche de spécification des exigences de sécurité devrait tenir compte de tous les critères suivants : l'expressivité, la rigueur, la simplicité d'utilisation, le coût et la

complétude de l'approche. En réalité, le choix de l'approche de spécification des exigences de sécurité est un compromis entre ces critères et les besoins réels de l'application.

Une des perspectives de recherche dans ce domaine serait de définir des outils intégrés qui permettraient de couvrir toutes les phases de l'Ingénierie des exigences de sécurité (élicitation, analyse, spécification, vérification et gestion) et de tenir compte des aspects organisationnels dans la définition des exigences.

Une autre perspective serait d'améliorer les méthodes d'analyse des conflits entre les exigences non fonctionnelles entre elles puis entre les exigences fonctionnelles et les exigences de sécurité.

Références

Barnum, S., Sethi, A. (2006). *Introduction to Attack patterns*. cigital. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/attack/585-BSI.html

Botella, P., Burgués, X., Franch, X., Huerta, M. (2001). Modeling Non-Functional Requirements, *Proceedings of Jornadas de Ingenieria de Requisitos Aplicada (JIRA)*, Sevilla, Spain.

Dai, L., Cooper, K. (2005). *Formal design analysis framework : an aspect-oriented architectural framework*. PhD Dissertation, Dallas University of Texas.

Darimont, R., Van Lamsweerde, A. (1996). Formal Refinement Patterns for Goal-Driven Requirements. *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*, San Francisco, 179-190.

Firesmith, D.G. (2003a). Security Use Cases. *Journal of Object*, *Technology. Vol. 2, No. 3*, 53-64.

Firesmith, D.G. (2003b). Engineering Security Requirements. *Journal of Object, Technology. Vol 2, No 1*, 53-64.

Giorgini, P., Massacci, F., Zannone, N. (2005). Security and Trust Requirements Engineering. *In Foundations of Security Analysis and Design III - Tutorial Lectures, volume 3655 of Lecture Notes in Computer Science, Springer*, 237–272.

Glinz, M. (2005). Rethinking the Notion of Non-Functional Requirements. *Proceedings of the Third World Congress for Software Quality*, Munich, Germany, Vol. II. 55-64.

Hadavi, M.A., Hamishagi, V.S., Sangchi, H.M. (2008). Security Requirements Engineering; State of the Art and Research Challenges. *In Proceedings of the International MultiConference of Engineers and Computer Scientists*, Hong Kong: 19-21 March.

He, Q., Anton, A.I. (2003). A Framework for Modeling Privacy Requirements in Role Engineering. *In Proceedings of the 9th International Workshop on Requirements Engineering- Foundation for software quality*, Klagenfurt/Velden, Austria: 16-17 Juin, 137–146.

Software Engineering Standards Committee of IEEE Computer Society (1998), *IEEE Guide for Developing System Requirements Specifications.* Norme IEEE 1233, édition 1998.

Jurjens, J. (2001). Towards Secure Systems Development with UMLsec, *Fundamental Approaches to Software Engineering (FASE/ETAPS), International Conference*, Genoa.

Kotonya, G., Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. New York: John Wiley & Sons.

Ledang, H., Souqières, J. (2002). Integration of UML Views using B Notations. *Proceedings of workshop on Integration and Transformation of UML Models*, Malaga, Spain.

Liu, L., Yu, E.S.K., Mylopoulos, J. (2003). Security and Privacy Requirements Analysis within a Social Setting. *In Proceedings IEEE international conference on requirements engineering (RE'03), Monterey, California*, 151–161.

Lodderstedt, T. Basin, D. Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security. *In Proceedings of the 5th International Conference on the Unified Modeling Language*, Dresden, Germany: October 2002, 426–441.

Mansour, R. (2009), *Formal Analysis and Design for Engineering Security (FADES)*, PhD in Computer Science and Applications, Blacksburg, Virginia : Faculty of the Virginia Polytechnic Institute and State University.

Matoussi, A., Laleau, R. (2008). *A Survey of Non-Functional Requirements in Software Development Process*. Rapport Technique TR–LACL–2008–7. Paris : Laboratoire d'Algorithmique, Complexité et Logique (LACL), Université Paris 12.

McDermott, J., Fox, C. (1999). Using Abuse Case Models for Security Requirements Analysis. *In Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC)*, Phoenix, Arizona.

Mouratidis, H., Manson, G., Giorgini, P. (2003). Modelling Secure Multiagent Systems. *In Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne-Australia,* 859-866.

Mylopoulos, J., Chung, L., Yu, E. (1999). From Object-oriented to Goal oriented Requirements Analysis. *Communications of the ACM, Vol. 42 No. 1*, 31-37.

Nuseibeh, B., Easterbrook, S. (2000). Requirements Engineering: A Roadmap. *In Proceedings of International Conference on Software Engineering*, *ACM Press*. Limerick, Ireland.

Phillips, M. (2003), *Using a Capability Maturity Model to Derive Security Requirements*, SANS InfoSec Reading Room, GSEC Practical v1. http://www.sans.org/rr/whitepapers/bestprac/1005.php

Sandhu, R.S., Coyne, E.J, Feinstein, H.L, Youman, C.E (1996). *Role-based access control models*. IEEE Computer, 29(2), 38–47.

Sindre, G., Opdahl, A. L. (2001). Capturing Security Requirements through Misuse Cases. *In Proceedings the 14th Norwegian Informatics Conference (NIK'2001)*, Tromsø, Norway.

Su, X., Bolzoni, D., van Eck, P.A.T. (2006). A Business Goal Driven Approach for Understanding and Specifying Information Security Requirements. *In 11th International Workshop on Exploring Modelling Methods in Systems Analysis and Design (EMMSAD2006)*, Luxembourg, 465–472.

Van Lamsweerde, A. (2001). *Building Formal Requirements Models for Reliable Software.* LNCS, Vol. 2043, Springer.

Van Lamsweerde, A. (2004), Elaborating Security Requirements by Construction of Intentional Anti-Model. *Proceedings of the 26th International Conference on Software Engineering (ICSE)*, 148-157

Viega, J. (2005). *The CLASP Application Security Process.* Secure Software.