Une approche visant à l'assistance de l'ingénierie des exigences avec des cas d'utilisations

Stéphane S. Somé

Ecole d'ingénierie et de technologie de l'information (EITI)
Université d'Ottawa
800 King Edward, P.O. Box 450, Stn. A
Ottawa, Ontario, K1N 6N5, Canada
ssome@site.uottawa.ca

RÉSUMÉ. Les cas d'utilisations qui décrivent des interactions possibles entre un système et son environnement sont un moyen efficace pour la capture des besoins des usagers dans le cadre de l'ingénierie des exigences de systèmes informatiques. Dans la pratique courante, des définitions informelles de cas d'utilisations ainsi qu'un processus d'analyse manuel sont utilisés. Cet article propose une approche automatisée visant à l'assistance de l'ingénierie des besoins avec des cas d'utilisations. Notre approche inclus la formalisation des cas d'utilisations, une forme restreinte du langage naturel pour la description de cas d'utilisations, la dérivation d'une spécification exécutable à partir de cas d'utilisations et la génération d'un environnement permettant la simulation de cette spécification.

ABSTRACT. Uses cases which describe possible interactions between a system and its environment are an effective means for the capture of users needs during requirement engineering. In the current practice, informal definitions of use cases as well as a manual analysis process are used. This article proposes an automated approach aiming at assisting use cases based requirements engineering. Our approach includes use cases formalization, a restricted form of natural language for use cases description, the derivation of an executable specification from case uses, and the generation of an environment allowing the simulation of this specification.

MOTS-CLÉS : Cas d'utilisations, Ingénierie des exigences, Simulation, Machines à états finis

KEYWORDS : Use cases, Requirement engineering, Simulation, State machines

1. Introduction

Un cas d'utilisation est la «spécification d'une séquence d'actions, incluant des variantes, qu'un système (ou un sous-système) peut exécuter en interaction avec des acteurs du système»[8]. Un cas d'utilisation décrit une partie du comportement d'un système sans en exposer la structure interne. Les cas d'utilisations sont utiles pour capturer, documenter et valider les exigences. Plusieurs approches de développement de logiciels incluant le processus unifié de développement de logiciel [7] recommandent les cas d'utilisations pour la description des besoins des usagers.

La nature partielle des cas d'utilisations permet à plusieurs usagers ayant diverses vues d'un même système de fournir différents cas d'utilisations décrivant ce système. Ces cas d'utilisations peuvent inclure des parties communes. La nature partielle aide également au développement de systèmes par l'addition incrémentielle de services. Il est cependant souvent difficile de visualiser le comportement global résultant de la combinaison de plusieurs cas d'utilisations. Les cas d'utilisations définis séparément peuvent être contradictoires et l'ensemble de cas d'utilisations décrivant un système peut être incomplet. Une solution consiste à dériver un modèle exécutable intégrant tous les cas d'utilisations. Un tel modèle permet la simulation du système et partant de là, la validation ainsi que la vérification du comportement global intégrant l'ensemble des cas d'utilisations.

Dans cet article, nous présentons une approche visant à appuyer l'ingénierie des exigences avec les cas d'utilisations. Cette approche fournit un cadre pour l'élicitation, la clarification, la composition ainsi que la simulation de cas d'utilisations. Elle est soutenue par l'outil UCEd (Use Case Editor)[2]. UCEd permet la capture de cas d'utilisations dans une forme restreinte du langage naturel et produit une spécification exécutable intégrant les comportements partiels des cas d'utilisations. UCEd se base sur de l'information contenue dans un modèle de domaine d'application pour l'analyse syntactique des cas d'utilisations ainsi que la génération de spécification.

Cet article est organisé comme suit. La section suivante présente les cas d'utilisations ainsi que la notation permettant leur description. La section 3 traite de la modélisation du domaine. Nous présentons une approche de génération de machines à états finis à la section 4. La section 5 présente l'outil UCEd et finalement la section 6 présente quelques travaux reliés et conclut cet article.

2. Cas d'utilisations

Les cas d'utilisations décrivent des interactions entre systèmes informatiques et acteurs. Un *modèle de cas d'utilisations* comprend des cas d'utilisations, des acteurs, des *relations* entre acteurs et cas d'utilisations, ainsi que des *relations* entre cas d'utilisations. Une relation entre un acteur et un cas d'utilisation représente la participation de cet acteur au cas d'utilisation. Les relations entre cas d'utilisations comprennent la relation d'inclu-

sion (*include*) et la relation d'extension (*extend*) [8].Dans le langage UML, un *diagramme de cas d'utilisations* est la description graphique d'un modèle de cas d'utilisation. La Figure 1 décrit un diagramme de cas d'utilisations d'un *Système de Surveillance de Patients* (PM System) destiné à être utilisé par des infirmières et médecins pour la surveillance des signes vitaux de patients dans un environnement hospitalier. Les diagrammes de cas

Figure 1. *Exemple de diagramme de cas d'utilisation du PM System.*

d'utilisations ne décrivent pas les interactions de cas d'utilisations. Nous avons défini pour ce faire, une forme restreinte du langage naturel. Nous utilisons un format de cas d'utilisations inspiré de [12].

La Figure 2 décrit les détails du cas d'utilisation *Log in* décrivant la procédure d'ouverture de session du PMSystem. Un cas d'utilisation consiste en une séquence d'étapes (*steps*). Chacune des étapes comprend une *opération* du système ou d'un acteur. Chaque étape peut également comprendre un ensemble d'*extensions* décrivant des possibilités de continuation du cas d'utilisation selon certaines *conditions*. Notre notation de cas d'utilisations utilise une forme *restreinte* du langage naturel. Par exemple, les conditions apparaissant dans les cas d'utilisations sont des *phrases prédicatives* décrivant des *situations* dans le système et son environnement, tandis que les opérations sont des *phrases actives* où une composante effectue une action donnée comme verbe. Chaque cas d'utilisation est interprété comme un *scénario primaire* et 0 ou plusieurs *scénarios secondaires* qui sont des séquences alternatives d'événements. Le scénario primaire est décrit à la section ayant pour titre *Steps*, tandis que les scénarios secondaires consistent en interactions du scénario primaire suivies d'interactions décrites à la section de titre *Extensions*.

Title : Log in

Precondition : PMSystem is ON AND NO user is logged in AND NO card is inserted

Steps : 1 : User inserts a Card
2 : PMSystem asks for Personal Identification Number (PIN)
3 : User types PIN
4 : PMSystem validates USER identification
5 : PMSystem displays a welcome message to User
6 : PMSystem ejects Card

Extensions : 1a : Card is not regular
1a1 : PMSystem emits alarm
1a2 : PMSystem ejects Card
4a : User identification is invalid AND User number of attempts is less than 4
4a1 Go back to Step 2
4b : User identification is invalid AND User number of attempts is equal to 4
4b1 : PMSystem emits alarm
4b2 : PMSystem ejects Card

Postcondition : User is logged in

Figure 2. *Cas d'utilisation décrivant le procédé d'ouverture de session du PM System.*

3. Modèle du domaine

Un modèle du domaine est un modèle de classe de haut-niveau capturant des concepts du domaine et leurs relations. Un modèle du domaine est essentiel pour l'analyse des cas d'utilisations. En effet, les éléments du domaine sont utilisés comme *dictionnaire* pour le traitement du langage naturel de la description des cas d'utilisations. Le modèle du domaine d'application fournit également une formalisation des opérations nécessaires à la génération de spécification (voir section 4).

Nous utilisons les diagrammes de classe d'UML [10] pour la description de modèles du domaine. Nous avons étendu la notation des diagrammes de classe d'UML par des *stéréotypes* [8] de façon tel qu'il soit possible de spécifier les *effets* des opérations par la définition d'un ensemble de conditions ajoutées (*added-conditions*) et d'un ensemble de conditions retirées (*withdrawn-conditions*). Ces deux ensembles sont utiles pour la génération de spécification à partir de cas d'utilisations. La Figure 3 décrit une représentation graphique du modèle de domaine du PM System selon la notation UML, ainsi qu'une partie des conditions ajoutées et retirées d'opérations.

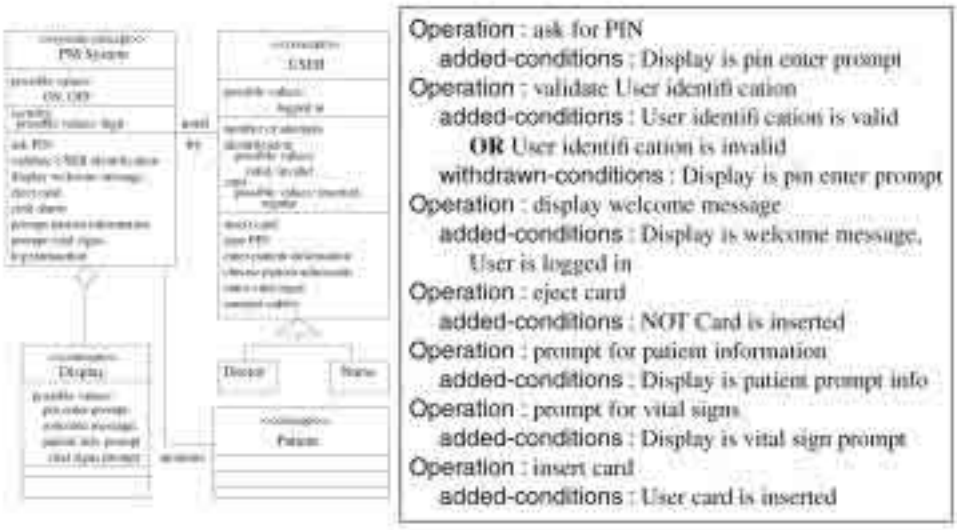

Figure 3. *Représentation partielle du modèle de domaine du PM system.*

4. Génération de machine à états finis

La génération de machines à états [11] est basée sur (1) la spécification des effets des opérations par des ensembles de conditions retirées et ajoutées, et (2) la relation entre états et conditions. Chaque état est défini par un ensemble de *conditions caractéristiques* vérifiées dans celui-ci. Ces conditions sont formulées comme prédicats $<E,V>$ avec E une entité du domaine et V une valeur. L'algorithme consiste pour chaque cas d'utilisation UC, à augmenter une machine à états finis initialement vide M avec des états et des transitions tel que la machine obtenue comprend tous les scénarios du cas d'utilisations UC comme séquences de transitions. Nous utilisons les effets des opérations pour la détermination des états et transitions comme suit. Supposez "-" un opérateur tel que C_1 et C_2 étant 2 ensembles de conditions, $C_1 - C_2$ est obtenu en supprimant toutes les conditions incluses dans C_2 de C_1, et $C_1 + C_2$ est obtenu par l'ajout de toutes les conditions de C_2 à C_1. Etant donné un état s avec *pred(s)* comme conditions caractéristiques, l'exécution de l'opération *op* ayant comme conditions ajoutées *add_conds(op)* et comme conditions retirées *withdr_conds(op)* produit un état s' tel que $pred(s') = (pred(s) - withdr\_conds(op)) + add\_conds(op)$.

La Figure 4 décrit une machine à états finis générée à partir du cas d'utilisation *Log in*. La génération débute par la détermination d'un état correspondant à la précondition du cas d'utilisation à composer. Dans l'exemple du cas d'utilisation *Log in*, il s'agit de l'état S1 caractérisé par les prédicats {<PMSystem,ON>, <User,not logged in>, <Card,not inserted>}. Nous générons ensuite un ensemble de transitions correspondant aux étapes du cas d'utilisation. Par exemple, l'étape *1* du cas d'utilisation *Log in* comprend l'opération *insert Card* par l'acteur *User*, ainsi que l'extension *1a* ayant pour condition *Card is*

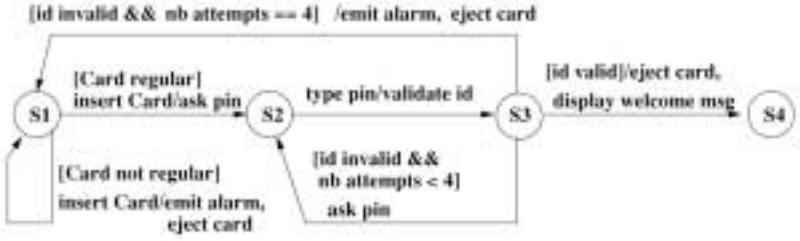

Figure 4. *Machine à états finis générée à partir du cas d'utilisation Log in. Le format UML «[garde] action/réactions» est utilisé pour les transitions.*

not regular. Deux transitions correspondent à cette étape. Une transition de l'état S1 à S1 ayant pour garde *Card is not regular*, action *insert Card* et réactions *emit alarm* ainsi que *eject card*. Cette transition est créée pour l'extension *1a*. La garde corresponds à la condition de l'extension, l'action à l'opération de l'étape et les réactions aux réactions stipulées dans l'extension. La seconde transition correspondant à l'étape *1* du cas d'utilisation est la transition de l'état S1 à S2. Elle est créée en considérant toutes les réactions pouvant suivre l'action *insert Card* dans le scénario primaire. L'état S2 est obtenu en considérant l'ensemble des conditions ajoutées et retirées par les opérations *insert Card* et *ask pin* à partir des conditions caractéristiques de l'état S1.

L'algorithme de génération de machine à états finis permet la composition de cas d'utilisations liés ou ayant des parties communes. Les conditions caractéristiques des états servent à l'identification de correspondances entre cas d'utilisations séparés.

5. UCEd

L'outil Use Case Editor (UCEd) a été développé pour démontrer notre approche. UCEd inclut un éditeur de modèle du domaine, un module de capture de cas d'utilisations, un module de composition de cas d'utilisations et un simulateur de cas d'utilisations. L'éditeur de modèle du domaine fournit une interface de définition de modèle du domaine tels que décrit à la section 3. UCEd utilise le format XMI [8] permettant l'importation de modèles créés par des éditeurs UML tels que ArgoUML[1]. Le module de capture de cas d'utilisations permet l'édition de modèles de cas d'utilisations conformes au langage UML. De façon similaire au modèle du domaine, UCEd permet d'importer des modèles de cas d'utilisation en format XMI. UCEd fournit une interface facilitant l'écriture de la description des cas d'utilisations (champs, numérotage automatique, etc). Les cas d'utilisations sont vérifiés par rapport au modèle du domaine pour la détection d'inconsistances et omissions. Le module de composition de cas d'utilisations implémente notre algorithme de génération de machines à états finis. Le simulateur de cas d'utilisations de UCEd génère une interface à partir de la machine générée permettant

de *simuler* les cas d'utilisations. La Figure 5 présente une vue du simulateur. UCEd gé-

Figure 5. *Vue du Simulateur de cas d'utilisations de UCEd.*

nère un "bouton" pour chaque opération d'acteur tel que cliquer sur ce bouton simule un événement correspondant à l'opération. Si l'état de simulation courant inclut une transition sur une opération sélectionnée, le simulateur affiche toutes les réactions possibles du système et modifie l'état de simulation courant. Le simulateur rapporte certaines contradictions telles que la présence de transitions non déterministes à l'utilisateur d'outil pour la correction (si nécessaire).

6. Conclusion

Nous avons présenté une approche visant à supporter l'ingénierie des exigences. Cette approche permet aussi bien la capture et la clarification de cas d'utilisations que celle des concepts appropriés du domaine. Les préconditions et postconditions des opérations peuvent servir de *contrat* des opérations en vue de la conception du système.

Notre travail est relié à un ensemble de travaux visant à l'analyse de cas d'utilisations, tels que [9], [6], [3] et [5]. Ces travaux mettent toutefois plus l'accent sur la fourniture de directives et langages restreints pour la production de cas d'utilisations en évitant les ambiguïtés et erreurs. Par exemple, Rolland et Ben Achour [9] proposent des structures et modèles linguistiques pour la spécification de cas d'utilisation. Ils proposent également un processus itératif pour l'écriture de cas d'utilisations sous forme de texte non ambiguë du langage naturel. Notre démarche présente des similarités en ce qui a trait à la définition d'une notation pour les cas d'utilisations. Toutefois, nous proposons un cadre plus général d'acquisition de cas d'utilisations en conjonction avec un modèle du domaine. Nous

nous intéressons également à la génération automatique de spécifications à partir de cas d'utilisations et à leur simulation.

Etant donné qu'il n'existe pas de méthode uniforme de description du contenu des cas d'utilisations, plusieurs différentes notations sont utilisées dans la pratique. Nous avons défini une syntaxe abstraite des cas d'utilisations telle que toute syntaxe concrète pourrait être utilisée en autant que les cas d'utilisations puissent être interprétés dans notre syntaxe abstraite. Nous planifions d'étendre notre notation en permettant entre autres l'intégration d'exigences non fonctionnelles telles que des contraintes de performance.

7. Bibliographie

[1] ArgoUML project. http ://argouml.tigris.org.

[2] Use Case Editor (UCEd) toolset. http ://www.site.uottawa.ca/ßsome/Use_Case_Editor_UCEd.html.

[3] K. Böttger, R. Schwitter, D. Richards, O. Aguilera, and Diego Mollá. Reconciling use cases via controlled languages and graphical models. In *INAP 2001, Proceedings of the 14th International Conference on Applications of Prolog*, pages 186–195, 2001.

[4] A. Cockburn. *Writing Effective Use Cases*. Addison Wesley, 2001.

[5] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari. Application of linguistic techniques for use case analysis. In *RE'02, Proceedings of the 10th Requirements Engineering Conference*, 2002.

[6] Glinz. Improving the quality of requirements with scenarios. In *Proceedings of the Second World Congress on Software Quality*, pages 55–60, 2000.

[7] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1998.

[8] OMG. OMG Unifi ed Modeling Language Specifi cation version 1.4, 2001.

[9] C. Rolland and C. Ben Achour. Guiding the construction of textual use case specifi cations. *Data & Knowledge Engineering Journal*, 25(1-2) :125–160, 1998.

[10] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1998.

[11] S. Somé. An approach for the synthesis of state transition graphs from use cases. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'03)*, volume I, pages 456–462, 2003.

[12] A. Cockburn. *Writing Effective Use Cases*. Addison Wesley, 2001.