Formalisation des besoins à l'aide de schémas LSCs

Jeanine Souquières, Maritta Heisel

Formalisation des besoins à l'aide de schémas LSCs

Jeanine Souquières
LORIA—Université Nancy2
B.P. 239 Bâtiment LORIA
F-54506 Vandœuvre-les-Nancy
Jeanine.Souquieres@loria.fr

Maritta Heisel
FG Softwaretechnik
TU Ilmenau
D-98693 Ilmenau
maritta.heisel@tu-ilmenau.de

Résumé : Dans notre approche pour l'expression des besoins [HS99b], nous proposons d'intégrer une étape de formalisation très tôt dans le développement afin d'analyser de manière détaillée les besoins des utilisateurs et de découvrir les inconsistances et les problèmes à partir des difficultés rencontrées lors de la formalisation. Afin d'améliorer la lisibilité et l'écriture des besoins formalisés, nous proposons d'utiliser les LSCs, Life Sequence Charts, au lieu des formules sur les traces d'un système (c'est-à-dire des suites d'événements se produisant sur un état du système à un moment donné) pour la formalisation des besoins décomposés sous forme de fragments. Nous proposons, en particulier, des schémas graphiques définis à l'aide des LSCs pour exprimer différents types de besoins. Ces schémas constituent un guide à l'étape de formalisation.

1 Introduction

L'expression des besoins pour un système informatique est reconnue depuis de nombreuses années comme un réel problème. Elle comporte plusieurs processus consistant à identifier les besoins du client, les analyser et les documenter. L'analyse concerne le domaine d'application, le logiciel à réaliser s'inscrivant dans un environnement et/ou un système existant. Elle conduit à une meilleure compréhension du problème et inclut des étapes de négociation avec le client, pour clarifier les besoins, détecter d'éventuelles incohérences, évaluer les propositions et choisir parmi différentes alternatives.

Dans notre approche pour l'expression des besoins [HS99b], nous proposons d'intégrer une étape de formalisation très tôt dans le développement afin d'analyser de manière détaillé les besoins des utilisateurs et de découvrir les inconsistances et les problèmes à partir des difficultés rencontrées lors de la formalisation. En particulier, une analyse des interactions possibles entre les différents besoins est effectuée durant ce processus de formalisation [HS00]. Classiquement, on appelle interaction, des incohérences logiques ou bien des comportements non souhaités, comportements issus d'une combinaison de différents besoins exprimés séparemment, chaque besoin isolé étant considéré comme raisonnable.

L'approche propose de décomposer les besoins en fragments. Cette décomposition est justifiée par la difficulté de prendre en compte en même temps les différents aspects d'un système et d'en avoir d'emblée une vue globale. Les fragments correspondent à des scénarios indépendants exprimant le comportement attendu du système. Pour une meilleure traçabilité, chaque fragment est nommé.

Dans l'approche publiée dans [HS99b], nous avons proposé de formaliser les besoins à l'aide de formules sur les traces d'un système, c'est-à-dire des suites d'événements se produisant sur un état du système à un moment donné. Cette formalisation n'est pas facile à manipuler et une formalisation à l'aide de graphiques semble plus adaptée pour une première étape de formalisation. C'est pourquoi, nous proposons d'utiliser les Live Sequence Charts, LSCs, pour décrire les

besoins. Nous avons identifié un ensemble de schémas, décrits en termes de LSCs, permettant d'exprimer différents types de besoins.

Les LSCs, Live Sequence Charts [DH99], proposent une notation avec une sémantique formelle, notation plus expressive que les MSCs, Messages Sequence Charts [IT96], cette notation étant elle-même une variante des diagrammes de séquence d'UML [RJB98] pour décrire les interactions entre objets dans la description de scénarios. Les LSCs font la distinction entre les scénarios existentiels qui *peuvent* apparaître dans un système et les scénarios universels qui *doivent* apparaître dans un système. De plus, les LSCs permettent d'exprimer des comportements interdits. Ils peuvent aussi spécifier des messages qui *peuvent* être reçus (*cold*) et des messages qui *doivent* être reçus (*hot*). Une condition peut être *cold*, signifiant qu'elle *peut* être vraie (sinon le contrôle est transmis à l'extérieur du bloc courant) ou *hot* signifiant qu'elle *doit* être vraie (sinon le système s'interrompt). De la même façon, la progression suivant les axes temporels peut être *hot*, exprimé par un axe continu renforçant la progression ou *cold*, exprimé par un axe en pointillé, c'est-à-dire dans lequel la progression n'est pas obligatoire.

Les scénarios sont d'une grande aide pour représenter les aspects comportementaux d'un système. Ils restent partiels en ce sens qu'ils donnent un aperçu du comportement de l'ensemble des acteurs du système dans une situation donnée. Concernant la formalisation des besoins,

- les LSCs sont adaptés à l'expression des besoins sous forme de fragments;
- ils proposent une description graphique synthétique et lisible;
- ils ont une plus grande expressivité que les Messages Sequence Charts ou les diagrammes de séquence d'UML, grâce en particuleir à la prise en compte des comportements optionnels;
- ils offrent une sémantique formelle;
- cette formalisation reste compatible avec la recherche d'interactions car les LSCs ont une sématique formelle et nous permettent d'exprimer les besoins sous une forme schématique comme requis par l'algorithme de recherche d'interactions [HS00].

Un état de l'art sur l'expression des besoins est proposé dans le chapitre 2. Le chapitre 3 présente une description de schémas graphiques définis à l'aide des LSCs permettant la formalisation de différents types de besoins. Le chapitre 4 propose une généralisation des différents schémas proposés. Une conclusion est proposée dans le paragraphe 5.

Parmi les exemples, nous utiliserons l'ascenseur avec comme cahier des charges celui formalisé dans [HS00], l'automate bancaire formalisé dans [HS99a] et le contrôle d'accès à un bâtiment étudié dans [SH00].

2 Etat de l'art

La distinction entre expression des besoins et spécifications n'est pas facile à effectuer et une certaine confusion est présente dans la littérature du domaine. Des travaux de recherche récents ont tenté de clarifier la nature des étapes constituant l'expression des besoins [JZ95, PM95, ZJ97]. De manière indépendante, Jackson, Zave et Parnas ont effectué une distinction entre les propriétés du domaine (appelé "indicative" par Jackson [Jac01] et NAT par Parnas [PM95]) et les besoins, distinction essentielle non prise en compte dans les différents langages de spécification existants. Une autre distinction importante effectuée par Jackson, Zave et Parnas est celle entre les "requirements" et les spécifications. Les requirements sont exprimés en termes de phénomènes du monde réel ou objets partagés par le logiciel et son environnement, avec un vocabulaire accessible par les utilisateurs [Jac01]. Ils capturent les relations requises entre les phénomènes de l'environnement, phénomènes surveillés et contrôlés par le logiciel [PM95]. Les spécifications sont formulées en termes de phénomènes partagés par le logiciel et son environnement et capturent des relations

entre les entrées et les sorties du logiciel. Les spécifications sont considérées comme des besoins particuliers. De notre point de vue, les besoins formalisés sous la forme de scénarios représentent des fragments. Ils ne constituent donc pas un modèle et ne déterminent pas entièrement le comportement du logiciel. Ils servent à en exprimer des propriétés. C'est pourquoi, par opposition à Jackson et Parnas, nous considérons qu'une spécification n'est pas un ensemble de besoins mais un modèle du logiciel à développer satisfaisant les besoins.

Alors que les hypothèses ne sont pas utilisées par Parnas, ni par Jackson et Zave, elles sont utilisées dans l'approche KAOS [DL96] : les besoins doivent être satisfaits alors que les hypothèses sont sous le contrôle de l'environnement. La notion de buts introduite dans KAOS est intéressante pour traiter les conflits et les interactions entre différents besoins, leur non détection pouvant conduire à des échecs importants [Lam00]. Les conflits et les interactions doivent être détectés et résolus même s'ils peuvent être temporairement utiles pour progresser dans la phase d'élicitation des besoins. La notion d'obstacles ou de condition d'obstruction a été utilisée dans [LL98] pour produire un arbre de raffinement dont la racine est la négation d'un but. Des techniques heuristiques et formelles sont disponibles pour générer systématiquement les obstacles à partir de la spécification des buts et des propriétés du domaine [LL00]. Cette approche est complémentaire à celle que nous proposons.

Le système et son environnement peuvent être décrits à l'aide de scénarios, ou séquences spécifiques d'actions qui illustrent un comportement. Différentes approches ont été étudiées et proposées dans la littérature pour aider à définir des scénarios [Je98]. Une étude [WPJH98] a montré que les scénarios jouent un rôle important dans les phases d'expression et de validation des besoins et sont utilisés pour une grande variété de systèmes. Des efforts de recherche importants ont vu le jour notamment avec les approches orientées objets et l'utilisation de notations telles que celles proposées par UML [RJB98]. Le problème soulevé par les scénarios est qu'ils sont partiels et certaines propriétés requises sont souvent laissées implicites. Leur couverture ne permet pas de vérifier l'absence d'erreurs. Notre approche complémentaire avec la formalisation des différents fragments à l'aide de LSCs et l'analyse des interactions entre ces fragments apporte une solution à ces différents problèmes. L'utilisation de schémas graphiques pour exprimer différents types de besoins constitue une aide à la formalisation.

3 Description des schémas et exemples d'application

Dans nos traveaux antérieurs sur l'ingenierie des besoins [HS99a, HS99b, SH00], nous avons remarqué que certaines formes de besoins étaient souvent utilisées. Par exemple, il est nécessaire d'exprimer des besoins comme "Après que *X* a eu lieu, *Y* doit se produire". C'est pourquoi nous prososons un ensemble de schémas qui permettent d'exprimer les différents types de besoins les plus utilisés d'une manière simple et ré-utilisable.

Dans la suite, nous donnons un ensemble de schémas exprimés à l'aide des LSCs. Ces schémas ne sont pas une extension de la notation LSC, mais des LSCs aves des *variables*. Par exemple, une variable *c* qui est utilisée dans un schéma peut être instanciée par une condition. Après l'instantiation de toutes les variables d'un schéma LSC, on obtient un LSC concret.

Afin de réutiliser notre algorithme de détection des interactions entre différents besoins [HS00], nous considérons qu'un besoin est constitué d'une précondition et d'une postcondition. Dans les LSCs, la précondition d'un besoin s'exprime à l'aide de l'activation d'un chart. Un chart peut être activé par un événement *e* ou par une condition *c*, appelée condiction d'activation (AC) comme exprimé dans la partie gauche de la figure 1.

Une simple condition d'activation ou un événement ne suffisent pas toujours pour activer un LSC. Dans ce cas, un chart d'activation appelé $Pre - chart$ est nécessaire, comme précisé dans la partie droite de la figure 1.

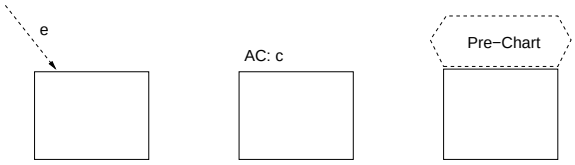


FIG. 1 – *Activation de LSCs*

Nous utiliserons la notation proposée figure 2 pour faire référence à l'une des trois précédentes possibilités d'activation d'un chart.

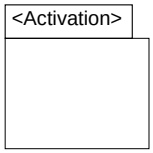


FIG. 2 – *Forme générale d'un LSCs*

Dans la suite, nous donnons des schémas de charts correspondant à des types de postconditions permettant de formaliser différents types de besoins.

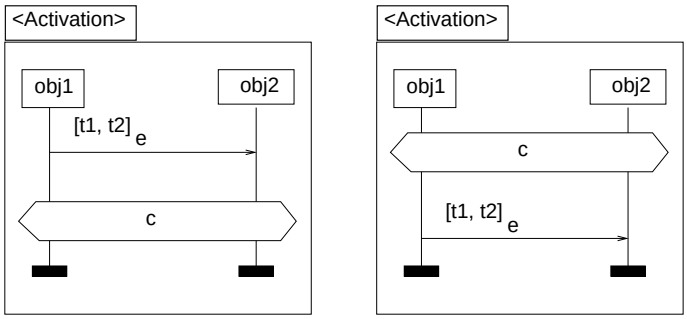
3.1 Nécessairement suivi de

Description On souhaite exprimer le fait qu'un événement, une condition d'activation ou un chart est nécessairement suivi par un événement *e*, celui-ci se produisant dans un intervalle de temps $t1 \,..\, t2$. Cet événement établit la condition *c*.

Une variante de ce schéma correspond au cas où l'activation établit la condition *c*. On peut vouloir exprimer qu'après l'établissement de la condition *c*, l'événement *e* doit se produire.

La ligne continue (hot) sur les axes temporels signifie que la progression doit avoir lieu, les événements et les conditions doivent se produire.

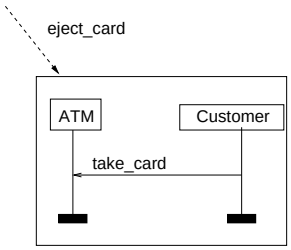
Schéma LSC



Variables $Activation, obj1, obj2, t1, t2, c, e$

Exemple1 ATM - ass_2: "When the machine ejects the card, the customer will take it."

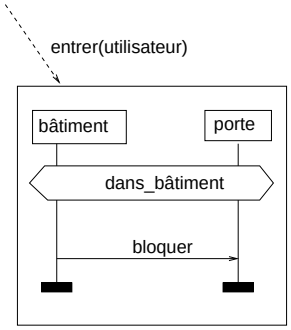
LSC instancié:



Dans cette instance de schéma “nécessairement suivi de “, la condition est vraie, elle peut être supprimée.

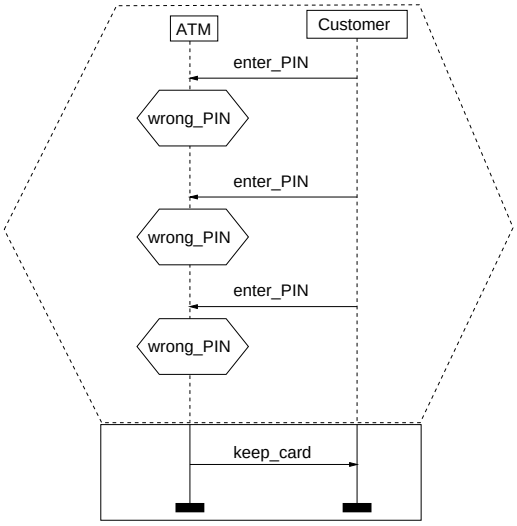
Exemple2 Contrôle d’accès - Besoin 3.2.4: “Si l’utilisateur entre, son passage est détecté et le journal est mis à jour; la porte est bloqueé.”

LSC instancié:



Exemple3 ATM - req_4: “A customer has only three trials to type the right PIN. If three times a wrong PIN is entered, the card is kept by the teller machine.”

LSC instancié:



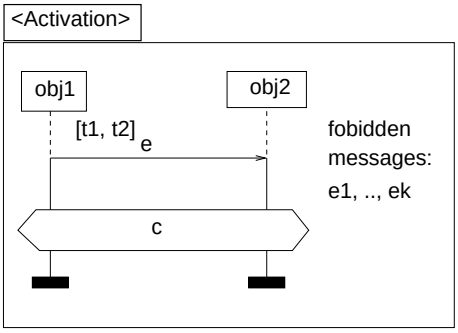
Ici, le pré-chart indique que trois essais infructueux peuvent être consécutifs et conduire à l’absorbtion de la carte par l’automate.

3.2 Peut être suivi de

Description On souhaite exprimer le fait qu'un événement, une condition d'activation ou un chart peut être suivi d'un événement *e*, si les événements $e1, \ldots,$ *ek* ne se produisent pas. L'événement *e* se produira dans l'intervalle de temps $t1$ .. $t2$. Il établira la condition *c*.

Les axes en pointillés (cold) signifient que la progression n'est pas obligatoire; mais si l'événement *e* se produit, alors *c* devient nécessairement vraie. Il est possible de donner dans un chart des messages interdits (forbidden messages), les messages non mentionnés pouvant se produire librement.

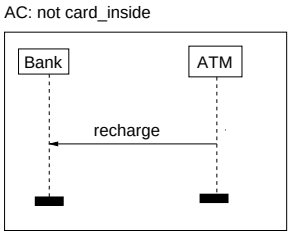
Schéma LSC



Variables *Activation*, $obj1$, $obj2$, $t1$, $t2$, c, e, $e1$, $\ldots$, *ek*

Exemple ATM - req_7: "The bank can recharge the machine with money between two withdrawal transactions."

LSC instancié:

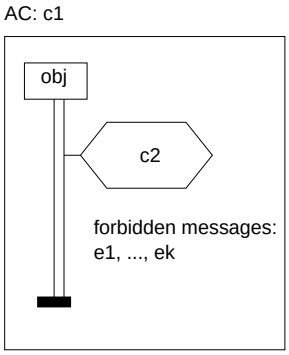


Ici, nous n'avons pas de conditions ni de messages interdits.

3.3 Invariant

Description On souhaite exprimer le fait que lorsque la condition $c1$ est vraie, la condition $c2$ reste égale à vrai tant que les événements $e1, \ldots,$ *ek* ne se produisent pas.

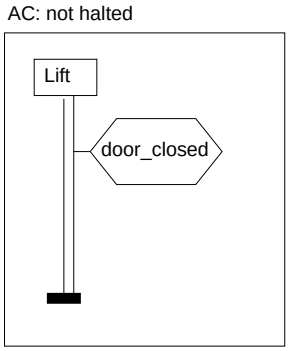
Schéma LSC



Variables $c1, obj, c2, e1, \ldots, ek$

Exemple Lift - req_{11}: “Whenever the lift moves, its doors must be closed.”

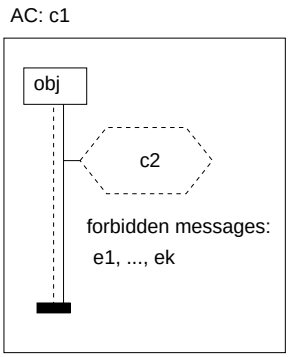
LSC instancié:



3.4 Evénements impossibles

Description On veut exprimer le fait que si la condition $c1$ est vraie, les événements $e1, \ldots, ek$ ne peuvent pas se produire tant que la condition $c2$ reste vraie.

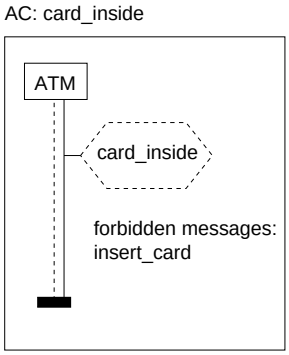
Schéma LSC



Variables $c1, obj, c2, e1, \ldots, ek$

Exemple ATM - $fact_1$: The reader can hold only one card at a time.

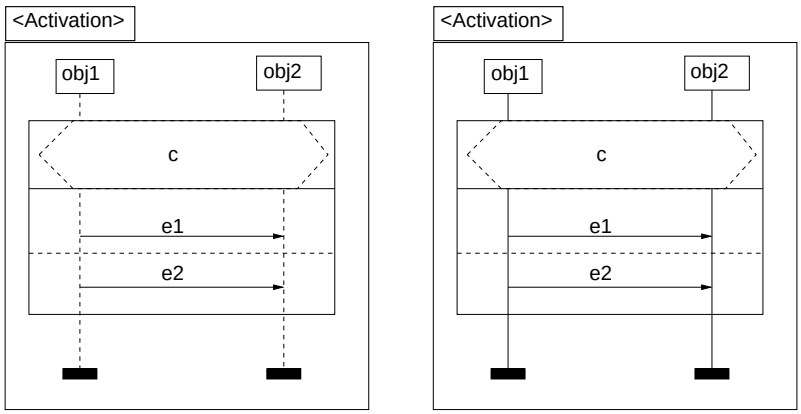
LSC instancié:



3.5 Conditionnelle

Description Lorsque le chart est activé, soit l'événement $e1$, soit l'événement $e2$ se produira en fonction de la condition c.

Schéma LSC

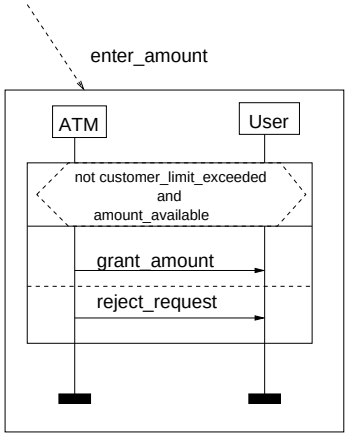


Nous avons considéré deux cas; dans le premier cas, l'un des événements $e1$ ou $e2$ *doit* se produire, dépendant de la condition c, ceci est exprimé par la ligne pointillée sur les axes temporels. Dans le second cas, l'un des événements $e1$ ou $e2$ *peut* se produire, ceci est exprimé par la ligne continue sur les axes temporels.

Variables $Activation, obj1, obj2, c, e1, e2$

Exemple ATM - req_5: "A user can only withdraw money if a weekly limit is not exceeded and if the demanded amount does not exceed the amount currently available in the teller machine."

LSC instancié:



Remarque. Les schémas présentés ci-dessus couvrent une grande partie des besoins usuels. Tous les besoins étudiés dans les différentes études de cas citées dans le papier ont été exprimés à l'aide des schémas proposés.

4 Généralisation des schémas proposés

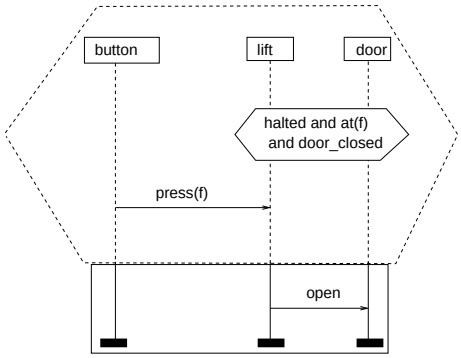
Les schémas présentés dans le chapitre 3 peuvent être généralisés de deux manières différentes.

4.1 En fonction du nombres d'objects interagissant

Jusqu'ici, nous avons montré seulement un ou deux objets partageant des événements. Il est possible d'introduire plus d'objets si nécessaire.

Exemple Lift - req_{10}: "When the lift is halted at floor f with the door closed and receives a call for the floor f, it opens its door."

LSC instancié:

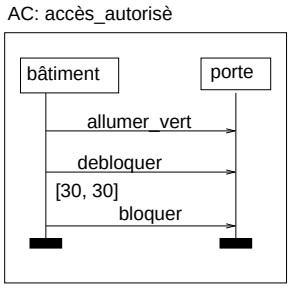


4.2 En fonction du nombre d'événements

Plutôt que d'avoir un seul événement comme indiqué dans les schémas du chapitre 3, plusieurs événements doivent ou peuvent se produire.

Exemple Contrôle d'accès - Besoin 3.2.3: "Après lecture de la carte, si l'accès est autorisé, le voyant vert s'allume, la porte est débloquée et la personne dispose de 30 secondes pour entrer avant reblocage de la porte."

LSC instancié:



5 Conclusion

Les approches basées sur les scénarios connaissent un succès important surtout dans le domaine des télécommunications. Les scénarios sont d'une grande aide lorsque l'on souhaite représenter les aspects comportementaux d'un système. Ils présentent un ensemble de vues différentes

du comportement de tous les acteurs. Ils sont partiels, concrets et intuitifs, mais ils peuvent être incohérents. Les LSCs apportent explicitement un statut au scénario avec trois possibilités:

– universel, dans lequel le comportement est toujours valable,
– existentiel, dans lequel le comportement doit pouvoir être exhibé par le système,
– interdit dans lequel le comportement ne peut pas être exhibé par le système.

La sémantique des LSCs a été définie en dérivant un automate de Büchi [KW01]. Cet automate peut être utilisé pour résoudre les problèmes de consistence, de synthèse et de vérification de modèles [BH02]. Elle permet de détecter les incohérences et les ambiguités. La syntaxe graphique des LSCs facilite la compréhension des besoins par les différents protagonistes d'un projet, apportant une compréhension intuitive. De nombreux travaux visent à outiller ce langage. Notons en particulier l'algorithme de génération de spécifications à partir de LSCs proposé par Harel et Kugler [HH00]. Un outil d'exécution du comportement de systèmes réactifs décrits à partir de scénarios a été implanté [HR01, HKP02], utilisant des techniques de vérification formelle. Des vérifications de protocoles sur base d'une spécification exprimée en LSC, grâce au model checker FormalCheck ont été effectuées [BG01].

Notre approche apporte une aide à l'utilisation des LSCs via les schémas proposés. Ces schémas ne sont pas une extension des LSCs mais des LSCs particuliers avec variables. L'approche apporte une vérification de la cohérence des différents scénarios décrits via l'agorithme de recherche des interactions entre les différents scénarios directement réutilisable grâce aux expressions schématiques que l'on peut associer aux schémas proposés.

Références

[BG01] A. Bunker and G. Gopalakrishnan. Using Live Sequence Charts for Hardware Protocol Specification and Compliance Verification. In *IEEE International High Level Design Validation and Test Workshop*. IEEE Computer Press, 2001.

[BH02] Y. Bontemps and P. Heymans. Turning High-Level Live Sequence Charts into Automata. In *Workshop Scenarios and State-Machines, ICSE'02, Orlando, USA*. ACM Press, 2002.

[DH99] W. Damm and D. Harel. LSCs: Breathing life into message sequence charts. In *Proc. Third Conference on Formal Methods for Open Object-Based Distributed Systems*, 1999.

[DL96] R. Darimont and A.van Lamsweerde. Formal Refinement for Patterns for Goal-Driven Requirements Elaboration. In *Proc FSE-4, ACM Symposium on the Foundation of Software Engineering*, pages 179–190. ACM Press, 1996.

[HH00] D. Harel and H.Kugler. Synthesizing State-Based Objects Systems from LSC Specifications. In *Proc. Fifth Int. Conf. on Implementation and Application (CIAA 2000)*. LNCS, Springer Verlag, 2000.

[HKP02] D. Harel, H. Kugler, and A. Pnueli. Smart Play-Out Behavioural Requirements. Technical Report Report MCS02-08, Weizmann Institute of Science, 2002.

[HR01] D. Harel and R. Ramelly. Specifying and Analysing Behavioural Requirements: The Play-In/Play-Out Approach. Technical Report Report MCS01-15, Weizmann Institute of Science, 2001.

[HS99a] M. Heisel and J. Souquières. A Method for Requirements Elicitation and Formal Specification. In LNCS Springer Verlag, editor, *Proceedings of the 18th International Conference on Conceptual Modeling*, number 1728, pages 309–324, November 1999.

[HS99b] M. Heisel and J. Souquières. De l'élicitation des besoins à la spécification formelle. *TSI*, 18(7), 1999.

[HS00] M. Heisel and J. Souquières. A heuristic algorithm to detect feature interactions in requirements. In Stephen Gilmore and Mark Ryan, editors, *Language Constructs for Describing Features*, pages 143–162. Springer-Verlag, 2000.

[IT96] Z.120 ITU-TS. Recommendation Z.120: Message Sequence Chart (MSC). Technical report, ITU-TS, Genova, 1996.

[Jac01] M. Jackson. *Problem Frames. Analyzing and structuring software development problems*. Addison-Wesley, 2001.

[Je98] M. Jarke and R. Kurki-Suanio (eds). *Special issue on Scenario Management*. Transactions on Software Enginnering. IEEE, December 1998.

[JZ95] M. Jackson and P. Zave. Deriving Specifications from Requirements : an Example. In *Proceedings 17th Int. Conf. on Software Engineering, Seattle, USA*, pages 15–24. ACM Press, 1995.

[KW01] J. Klose and H. Wittke. An Automata Based Representation of Live Sequence Charts. In Tiziana Margaria and Wang Yi, editors, *Proceedings of TACAS 2001*, page 512. Springer Verlag, LNCS 2031, 2001.

[Lam00] A.van Lamsweerde. Formal SPecification: a Roadmap. *Future of Software Engineering, ACM Press*, 2000.

[LL98] A. Van Lamsweerde and E. Letier. Integrating Obstacles in Goal-directed Requirements. In *Proc. of the 20 th International Conference on Software Engineering, ICSE'98*, Kyoto, Japan, 1998. I.E.E.E.

[LL00] A.van Lamsweerde and E. Letier. Handling Obstacles in Goal-directed Requirements Engineering. In Transactions on Software Engineering, editor, *Special Issue on Exception Handling8*. IEEE, 2000.

[PM95] D.L. Parnas and Madey. Functional Documents for Computer Systems. *Science of Computer Programming*, 25:41–51, 1995.

[RJB98] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Guide*. Addison-Wesley, 1998.

[SH00] J. Souquières and M. Heisel. Une méthode pour l'élicitation des besoins : application au système de contrôle d'accès. In Yves Ledru, editor, *Proceedings Approches Formelles dans l'Assistance au Développement de Logiciels - AFADL'2000*, pages 36–50. LSR-IMAG, Grenoble, 2000. http://www-lsr.imag.fr/afadl/Programme/ProgrammeAFADL2000.html.

[WPJH98] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. Scenario Usage in System Development: A Report on Current Practice. *IEEE Software*, March 1998.

[ZJ97] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1):1–30, January 1997.