# Expérimentation de composants de preuve pour le développement de composants logiciels embarqués

Arnaud Monégier du Sorbier*, Stéphane de Belloy*,
Florian Turpin*, Philippe Dhaussy**

* Thales Air Systems, BP 20351 F-94628 RUNGIS Cedex
{arnaud.monegier, stephane.debelloy, florian.turpin}@thalesgroup.com

** Laboratoire DTN-LISYC, ENSIETA, BREST, F-29806 cedex 9
dhaussy @ensieta.fr
http://www.ensieta.fr/dtn

**Résumé.** Pour améliorer les pratiques industrielles dans le domaine de la validation formelle de modèles de composants embarqués, Thales Air Systems et l'ENSIETA ont engagé une réflexion commune et des expérimentations pour la mise en œuvre de composants MDA nommés *Unités de Modélisation* et *Unités de Preuve*. Le but recherché est de capitaliser au sein de composants de modélisation les données et les activités mises en œuvre dans les processus de conduite des analyses formelles sur les modèles de conception de composants. Ces unités doivent être manipulées à différents niveaux d'abstraction. En particulier, les Unités de Preuve sont exploitées actuellement dans nos expérimentations par une technique de vérification de type *model-checking* avec la mise en œuvre d'observateurs. Dans une approche Ingénierie Dirigée par les Modèles (IDM), les modèles d'Unités de Modélisation et de preuve sont transformés en modèles d'automates temporisés puis en codes exploitables par l'outil OBP (*Observer-Based Prover*) développé à l'ENSIETA. Dans cet article, nous décrivons une application de cette approche pour la validation d'un modèle de gestion des états d'un système de contrôle et d'engagement de Thales Air Systems. L'article rend compte du retour d'expérience sur la mise en œuvre de cette technique et la mise en œuvre des Unités de Preuve.

**Mots clés** : Ingénierie Dirigée par les Modèles (IDM), Validation Formelle, Composants Model Driven Architecture (MDA) : Unités de Modélisation (UdM) & Unités de Preuve (UdP), Model-Checking, Observer-Based Prover (OBP), Context Description Language (CDL), Modèles d'automates temporisés.

## 1 Introduction

### Contexte des expérimentations de validation formelle à Thales Air Systems

La part des activités de modélisation dans le processus de développement industriel prend de plus en plus d'importance. De la capture des exigences jusqu'à la génération de code, les modèles et leur transformations servent de fil conducteur plus ou moins automatique à la réalisation des différentes activités système ou logiciel. Dans le cadre de la validation formelle de modèles, notre objectif est d'intégrer au mieux ces activités dans le processus de développement. Afin de pouvoir exploiter correctement ces modèles pour la suite du processus de développement et la génération de l'application finale, il est primordial de pouvoir s'assurer de la bonne adéquation de ceux-ci par rapport à la formulation de leurs exigences amont. Cette vérification de l'adéquation du modèle avec les exigences de l'application constitue la validation formelle de ce modèle. Pour que cette validation soit efficace, il est important qu'elle soit appliquée dans un cadre de développement formalisé et rigoureux. Pour les techniques de preuve formelles, l'approche IDM [5] est une opportunité permettant d'embarquer dans des modèles productifs, c'est-à-dire interprétables par des programmes (ou d'autres modèles), les notions ou méta-entités définissant les objets manipulés durant les preuves. Au regard des propositions autour du standard EIA-632 et sa déclinaison sous forme de méta-modèle SPEM de l'OMG [11], il est pertinent de

formaliser les processus de validation formelle par la construction de modèles de processus spécifiques aux activités de vérification. Ceux-ci doivent incorporer les données nécessaires aux preuves. Dans cet objectif, Thales Air Systems et l'ENSIETA étudie des composants de modélisation nommés *Unité de Modélisation (UdM)* et *Unité de Preuve (UdP)* [7], ainsi que de leurs relations mutuelles, pour structurer et référencer les activités et les données nécessaires à la conduite des preuves à mettre en œuvre lors des analyses formelles de modèles Des travaux et expérimentations de ces concepts ont été menés sur un exemple concret et industriel chez Thales Air Systems.

**Les concepts d'Unités de Modélisation (UdM) et d'Unités de Preuve (UdP)**

L'Unité de Modélisation constitue le cadre dans lequel s'effectue le développement de l'application, de sa phase de capture et de formulation de ses exigences jusqu'à celles conduisant à sa génération et tests finaux. Elle est structurée en 5 vues : la vue de *Contexte (*ou *CIM : Computation Independant Model)*, la vue *Logique* ou *PIM (Platform Independant Model)*, la vue *Physique* ou *PSM (Platform Specific Model)*, obtenue à partir des modèles PIM et PDM *(Platform Dependant Model)*, la vue *Implémentation ou EPBS (End Product Breakdown Structure)* et la vue *Tests et Intégration*. Chacune de ces vues met en œuvre des modèles définis par leurs méta-modèles correspondants. A chaque niveau N de décomposition du Système ou du Logiciel, l'Unité de Modélisation se raffine en plusieurs Unités de Modélisation plus détaillées de niveau N-1.

Nous définissons la notion *d'Unité de Preuve* comme un concept, de niveau utilisateur, englobant les données nécessaires à l'activité de preuve en vue de la validation d'un modèle. Ces unités sont générées, à partir des Unités de Modélisation, comme un composant structurant et référençant les données nécessaires aux preuves. Un composant de ce type doit être considéré comme un objet élémentaire, exploité par le modèle de processus de validation. Celui-ci manipule, pour un modèle donné soumis à validation et un contexte donné, un ensemble d'Unités de Preuve référençant l'ensemble des exigences et des scénarios à considérer pour ce modèle. En tant que composant MDA, il embarque non seulement les références aux modèles (contexte, propriétés, modèles à valider), et aux méta-modèles associés, mais aussi l'ensemble des règles de transformation permettant d'interpréter ce composant pour la génération des programmes d'analyse formelle. Ceux-ci déclenchent les analyses spécifiques, en fonction de l'outil de preuve choisi, et la construction des données de retour d'analyse.

Cet article présente, au paragraphe 2, les concepts d'Unités de Modélisation et de Preuve et les liens entre elles. Le paragraphe 3 décrit brièvement une expérimentation de ces concepts menée pour la validation d'un modèle conçu pour la génération d'une fonctionnalité opérationnelle d'un composant logiciel. Enfin, en guise de conclusion, nous tirons un premier bilan et des perspectives pour poursuivre l'étude de ces concepts.

## 2. Mise en œuvre des Unités de Modélisation et de Preuve

**Structure des Unités de Modélisation**

La vue de *Contexte* d'une unité de modélisation (figure 1.a) sert à modéliser le besoin par une approche systémique. Elle s'intéresse d'abord à l'aspect externe au système ou logiciel comme une boîte noire dont est défini l'ensemble des missions des interactions avec l'extérieur. L'aspect interne est ensuite analysé en termes de capacités. Le système ou logiciel est alors vu comme une boîte blanche. La vue *Logique* permet de modéliser les capacités du système ou du logiciel par une approche fonctionnelle, objet ou composant logique indépendamment de toute implantation. La vue *Physique* transforme le modèle logique du système ou du logiciel en ses différentes entités physiques (modules ou composants) et interactions mutuelles par sa projection sur le modèle de la plateforme retenue. La vue *Implémentation* réalise et génère le code des différentes entités physiques identifiées précédemment. La vue *Tests* et

*intégration* permet les tests et l'intégration progressive de ses différentes entités codées sur la plateforme choisie.

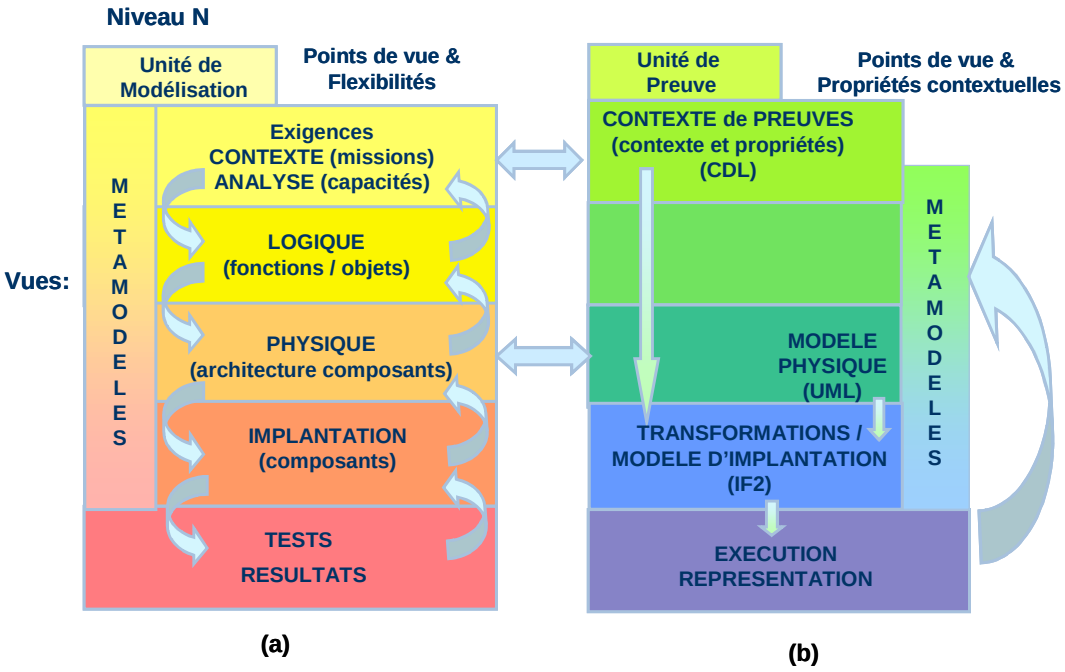


**Figure 1. Expérimentation des Unités de Modélisation et de Preuve pour un niveau d'abstraction N**

### Structure des Unités de Preuves

Une Unité de Preuve fait référence (figure 1.b) à plusieurs entités que sont les contextes de preuve, les méta-modèles et modèles impliqués dans la construction des preuves, les transformations de modèles en vue de générer les codes assimilables par les prouveurs et, enfin, les entités de représentation des résultats des preuves. En d'autres termes, les modèles d'Unités de Preuve sont des modèles, conformes à un méta-modèle. Ces modèles sont ensuite traduits par transformation en modèles concrets pris en charge par des outils d'analyse formelle. Le méta-modèle définissant les Unités de Preuve est composé des méta-entités qui définissent les éléments les constituant et que nous détaillons ici.

Les *contextes de preuve* regroupent les données caractérisant les exigences ou propriétés devant être vérifiées, le comportement de l'environnement dans lequel est plongé le modèle et le type d'analyse souhaitée par l'utilisateur. Le comportement de l'environnement est décrit sous la forme d'un modèle CDL [6] pour décrire le comportement de l'environnement et les propriétés qui sont à vérifier. Dans un contexte d'IDM, une Unité de Preuve fait également référence aux *méta-modèles* et les *modèles* mis en œuvre dans l'activité de validation. Ceux-ci sont impliqués dans le processus de construction des preuves, ou plus précisément dans la construction des prouveurs ou des simulateurs qui permettront les analyses sur le modèle. Les modèles à valider référencés sont manipulés par l'utilisateur dans des formats, par exemple, du type UML, SDL, AADL, etc. Les modèles issus automatiquement des transformations seront

produits dans des formats de bas niveau comme ceux reposant, par exemple, sur les automates temporisés, les réseaux de Petri, les algèbres de processus, etc.

Les méta-modèles de ces langages doivent donc être référencés par l'Unité de Preuve. Les modèles formels n'ont pas pour objectif d'être manipulés explicitement par l'utilisateur mais doivent rester « cachés ». Pour pouvoir générer les prouveurs (les simulateurs d'analyse), des *transformations* de modèles sont nécessaires pour traduire, d'une part, les modèles des contextes de preuve et, d'autre part, les modèles de conception en des programmes formels exploitables par ces outils de preuve ou de simulation. L'ensemble des règles de transformations est référencé dans une unité. Enfin, en vue de pouvoir présenter à l'utilisateur les résultats des analyses réalisées, un modèle de *représentation* ou de diagnostic est également référencé. L'interprétation de ce modèle doit permettre à l'utilisateur non seulement de détecter mais aussi de reconnaître et comprendre le type de défaillance d'une propriété et le composant du modèle à incriminer. Ce modèle doit être structuré de manière à être le réceptacle des données générées par les analyseurs lors de l'exécution des preuves. Des transformations doivent permettre de construire des modèles de données interprétables dans le modèle de l'utilisateur contenant les informations nécessaires au diagnostic. Ces données exploitables par l'utilisateur donneront souvent lieu à une visualisation. Dans notre approche, la mise en œuvre des modèles de représentation doit encore faire l'objet de développements spécifiques.

**Liens entre les Unités de Modélisation et de Preuve**

La validation formelle des modèles élaborés dans le cadre d'une Unité de Modélisation s'effectue au sein d'une Unité de Preuve structurée similairement sous forme de vues servant de cadre aux modèles et transformations nécessaires à cette validation et obtention des preuves associées. Comme pour l'UdM, chacune des vues des UdP met en œuvre des modèles définis par leurs méta-modèles correspondants. La vue Contexte de l'UdM alimente la vue Contexte de l'UdP en permettant l'écriture des scénarios et des propriétés à vérifier selon les points de vue ou cas d'utilisation considérés. La vue Logique de l'UdM n'est pas utilisée dans l'UdP. La vue Physique de l'UdM fournit, dans la vue Physique de l'UdP, les éléments de modèle (en totalité ou en partie) du Système ou du Logiciel à vérifier. La vue Implantation de l'UdP transforme le modèle à valider en code formel assimilable par un outil d'analyse. Dans nos premières expérimentations, le langage formel choisi pour l'instant est le langage IF2 [4], basé sur les automates temporisés [1]. Les propriétés à vérifier ainsi que les scénarios du contexte sont également traduits en langage IF2. Des développements en cours permettront bientôt de pouvoir générer des programmes FIACRE [3], langage pivot défini dans le cadre du projet TopCased[1]. Ceci permettra une connexion aux outils de vérification (TINA [2], CADP [9]) prévus dans la plateforme TopCased. La vue *Exécution-Représentation* de l'UdP permet d'exécuter l'analyse formelle des propriétés sur le modèle et de récupérer et d'analyser les résultats de retour de la validation.

## 3. Expérimentation des Unités sur un cas d'étude

Les concepts d'Unités de Modélisation et de Preuve ont été expérimentés sur un cas d'étude de Thales Air Systems : un système de contrôle anti-aérien. L'Unité de Modélisation a permis de structurer et formaliser les différentes étapes de modélisation du cas d'étude, de l'expression de ses besoins jusqu'à son codage et ses tests. Les données et modèles issus de ces différentes étapes ont servi à alimenter directement un ensemble d'Unités de Preuve dans le but de permettre la validation formelle du modèle à partir duquel le code de ce cas d'étude a été généré.

Pour la mise en œuvre des Unités de Preuve, nous nous sommes appuyés sur l'outil prototype OBP (*Observer-Based Prover*) [7], développé à l'ENSIETA, et le langage CDL, permettant de décrire les

[1] http://www.topcased.org

contextes de preuve. Le cas d'étude porte sur la partie logicielle d'un système anti-aérien, gérant et contrôlant l'ensemble de ses états, ses modes et ses actions en réponse aux informations ou observations reçues de l'environnement. Il s'agit d'un système critique soumis à des exigences de fiabilité et de sûreté.

La première étape du travail [8] a été la traduction manuelle du modèle de conception décrit en UML. Ce modèle a conduit à la génération d'un programme Ada d'environ 38000 lignes. Le modèle comporte un automate de 365 états et 560 transitions. A des fins de validation, ce modèle a été traduit sous la forme d'un automate formel IF2. Le nombre d'exigences exprimées dans le cahier des charges du système est de 170. Ensuite, il a fallu formaliser, à partir des données de contexte fournies par l'UdM, les scénarios d'interaction entre le système et son environnement. Ce contexte est décrit par plusieurs modèles CDL qui décrivent des ensembles de scénarios. Dans ces modèles CDL, nous avons intégré la description des propriétés correspondantes aux exigences devant être vérifiées sur le modèle de conception. Les propriétés ont été spécifiées à l'aide de patrons de description [6]. Les propriétés ont été traduites ensuite en automates observateurs [10] comportant des états de rejet, permettant de détecter les cas de falsification des propriétés. Enfin, les automates de contextes, observateurs et le modèle de conception ont été traduits automatiquement en des automates au format IF2 dans l'outil OBP. Ceux-ci peuvent ensuite composés par le simulateur IFx [4] qui permet de générer un système de transitions. Le diagnostic consiste alors en une analyse d'accessibilité, au sein d'OBP, des états de rejet sur le graphe d'exécution qui résulte de la composition du modèle à valider, d'un contexte et des observateurs.

Par cette méthode, nous avons pu vérifier une trentaine de propriétés de type accessibilité, sûreté et de vivacité bornée. Ces types de propriétés correspondent à une large majorité des propriétés à traiter dans le cahier des charges.

## 4 Bilan et perspectives

Cette première expérimentation a permis de mettre en évidence l'intérêt de la validation formelle pour les applications industrielles. L'apport constaté concerne, notamment, l'appropriation bénéfique par les ingénieurs de Thales Air Systems du langage CDL pour la description des contextes et des exigences.

**Appropriation du langage CDL**

Dans le cas d'étude traité, les modèles CDL ont été élaborés à partir, d'une part, des scénarios décrits dans les dossiers de conception et formalisés par des diagrammes de séquence et, d'autre part, à partir des documents répertoriant les exigences systèmes et dérivées. Lors de l'élaboration des modèles CDL, nous avons été confrontés à deux difficultés majeures. Une première est liée à un manque de formalisation complète et cohérente de la description du comportement de l'environnement du système étudié. Les scénarios d'utilisation du système, et donc les interactions entre le modèle étudié et son environnement sont décrits dans plusieurs documents parfois de manière incomplète.

Certaines informations concernant des modes d'interactions étaient implicites. L'élaboration des modèles CDL a donc nécessité des discussions avec les experts maîtrisant le système. La deuxième difficulté a concerné la compréhension des exigences. Celles-ci sont regroupées dans des documents de niveaux différents correspondants aux exigences systèmes ou dérivées. Ces dernières sont des exigences rédigées suite à l'interprétation des exigences systèmes au regard des choix de conception. L'ensemble des exigences analysées était rédigé sous forme textuelle et certaines d'entre elles donnaient lieu à plusieurs interprétations différentes. D'autres faisaient appel à une connaissance implicite du contexte dans lequel elles doivent être prises en compte. En effet, la plupart des exigences sont à prendre en compte dans une configuration donnée, lorsque le système a atteint une phase opérationnelle. Or les informations, concernant l'historique ayant mené à l'état dans lequel doit se trouver le système, ne sont en général pas explicités dans l'exigence. Elles font partie d'un ensemble d'informations qui doivent se trouver décrites explicitement dans les documents d'analyse du système. Mais parfois, elles doivent être

déduites d'une interprétation sur le comportement du système, pouvant être faite par les experts le connaissant parfaitement. Suite à notre proposition d'une mise à disposition, au travers du langage CDL, d'un cadre de formalisation des contextes et des exigences, les ingénieurs se le sont approprié et y ont trouvé immédiatement un intérêt pour mieux rédiger leurs spécifications. En ce sens, le couplage entre un modèle de type CDL et des ensembles de propriétés est une voie d'étude à poursuivre.

**Preuves de propriétés établies**

Durant l'expérimentation, nous avons établi la preuve d'environ une trentaine d'exigences significatives de type vivacité bornée, pour les contextes que nous avons élaborés. Celles-ci ont été rédigées à l'aide des patrons de propriétés proposés dans [6]. Chaque propriété a été convertie, soit automatiquement quand l'algorithme de traduction était opérationnel, soit manuellement, en un automate observateur au format IF2. La conception des algorithmes de traduction se poursuit actuellement. Le pourcentage de propriétés, non traduisibles en un automate observateur, car ne rentrant pas dans la catégorie de sûreté ou d'accessibilité, est de moins de 20% de l'ensemble des exigences contenues dans les documents que nous avons étudiés. L'élaboration des contextes a permis de restreindre le comportement du modèle à valider par la génération des chemins, représentant chacun un comportement de l'environnement. Chaque chemin, généré sous la forme d'un automate linéaire dont la taille en nombre d'états est comprise entre 100 et 300, a été composé avec l'observateur et le modèle. L'occurrence de la falsification d'une propriété dans chacun des modèles des deux d'études nous a permis de détecter une anomalie résiduelle dans le modèle concerné. Le but de l'expérimentation était de démontrer la faisabilité de la technique pour un ensemble de contextes identifiés et d'exigences significatives. La poursuite de notre travail s'oriente aujourd'hui vers la méthodologie de construction des ensembles complets de contextes pour pouvoir assurer le bien fondé des preuves. La manipulation des contextes, dans notre approche, doit se faire avec précaution pour que les preuves des propriétés soient significatives dans la mesure où elles sont chacune réalisées pour des contextes restreints.

**Perspectives pour l'intégration dans le processus de développement**

Lors de l'étude des cas, une réflexion s'est engagée entre les ingénieurs de Thales Air Systems et les enseignants-chercheurs de l'ENSIETA sur la capitalisation des activités de preuves menées. Nous constatons que la structuration et la mémorisation des données nécessaires à chaque preuve dans les Unités de Preuve permet une meilleure structuration de l'activité. L'étude de l'intégration de ces composants IDM, UdM et UdP, doit se poursuivre. Afin d'optimiser cette démarche dans le processus de développement Système ou Logiciel, des améliorations et automatisations sont recherchées.

Le premier axe concerne l'utilisation du langage CDL pour exprimer les propriétés et les scénarios pertinents de la vue Contexte de l'UdP en s'appuyant sur les exigences, les diagrammes de Use-Cases et de Séquences élaborés au niveau de la vue Contexte de l'UdM. Il facilite ainsi la formalisation de l'environnement externe pour vérifier le modèle du Système ou du Logiciel en cours de réalisation. Les autres axes et perspectives portent sur la cohérence et continuité les vérifications formelles à chaque niveau d'abstraction : exigences, modèles, code, au sein des Unités de Modélisation et de Preuves, renforçant ainsi leurs relations et synergies mutuelles au profit de la qualité finale de l'application Système/Logicielle.

Au niveau vue *Contexte*, le langage CDL permet de formaliser l'environnement du Système/Logiciel. En étendant son application à la formalisation des exigences du Système/Logiciel, nous pourrons vérifier, par exécution, la cohérence des deux formalisations interne et externe et par conséquent la qualité du jeu d'expression des exigences. Au niveau vue *Logique/Physique*, le niveau sur lequel porte actuellement la validation formelle, les transformations du langage CDL et de Modèle Système/Logiciel vers le langage formel sont à automatiser. Au niveau *Tests/Intégration*, les scénarios d'environnement exprimés par le langage CDL et transposés dans le langage cible de l'application générée à partir de son modèle pourraient être utilisés dans les tests de l'application finale. Cette approche permet ainsi de mieux maîtriser les jeux de tests optimaux à mettre en œuvre ainsi que l'étendue et la profondeur de la

couverture des tests de l'application suivant ses principaux niveaux d'abstraction concourrant à sa réalisation industrielle.

## 5 Références

[1] Alur R, Dill D. A Theory of Timed Automata. *In Theoretical computer Science*, 126(2) : 183-235, 2004.

[2] Berthomieu B., Vernadat F., "Time Petri nets analysis with TINA", *3rd Int. Conf. on the Quantitative Evaluation of Systems (QEST'2006)*, Riverside (USA), 11-14 Septembre 2006, p. 123-124.

[3] Berthomieu B, Bodeveix JP., Filali M., Garavel H., Lang F., Peres F., Saad R, Stoecker J., Vernadat F., "The Syntax and Semantics of FIACRE, Version 1.0 alpha", Technical report projet ANR05RNTL03101 OpenEmbeDD.

[4] Bozga M., Graf S., and Mounier L. IF2.0: A validation environment for component-based real-time systems. *In Proceedings of Conference on Computer Aided Verification, CAV'02, Copenhagen, LNCS. Springer Verlag*, 2002.

[5] Clarke T., Evans A., Sammut P., Willians J. Applied Meamodeling : A foundation for Language Driven Development. *Technical report*, version 0.1, Xactium, 2004.

[6] De Belloy S., Dhaussy Ph. CDL : Un langage de description de contextes, syntaxe et sémantique. *Rapport technique, Ensieta*, 2007.

[7] Dhaussy Ph, Boniol F. Mise en œuvre de composant MDA pour la validation formelle de modèles de système d'information embarqués. *Revue RSTI, numéro spécial Ingénierie des Systèmes d'Information, N°5*, dec. 2007.

[8] Dhaussy Ph, J.Auvray, S.De Belloy, F.Boniol, E. Landel, *Un langage de contexte de preuve pour la validation formelle de modèles logiciels*, Actes des conférences LMO'08 et CAL'08, Montréal, 3 au 7 mars 2008.

[9] Fernandez J-C et al., « CADP: A Protocol Validation and Verification Toolbox », *in Alur R. and Henzinger T.A, editors, Proceedings of CAV'96* (new Brunswick, USA), Vol. 1102 LNCS, August 1996.

[10] Halbwachs N., Lagnier F. and Raymond P. Synchronous observers and the verification of reactive systems. *In 3rd int. Conf. on Algebraic Methodology and Software Technology (AMAST'93)*, 1993.

[11] Leblanc H., Millan T., Ober I. Démarche de développement orienté modèles : de la vérification de modèles à l'outillage de la démarche », *in 1ères journées IDM*, 2005, page 125-139, Paris.