UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

KARIM BOUDAOUD

Combinaison de SysML et Model Checker pour la Simulation et la Vérification Formelle

Sous-titre

Thèse présentée et soutenue à Besançon, le 14 Mars 2018

Composition du Jury :

HULK INCROYABLE	Professeur à l'Université de Gotham City	Président

N° | X | X | X |

école doctorale **sciences pour l'ingénieur et microtechniques**

Titre : Combinaison de SysML et Model Checker pour la Simulation et la Vérification Formelle

Mots-clés : SysML, Diagramme de Séquence , Vérification Formelle , Automate Temporisé ,Modèle de transformation, UPPAAL

Résumé :

SysML est un langage de modélisation des systèmes rapidement émergeant comme une norme de facto utilisée pour les spécifications logicielles, les diagrammes de séquence UML fournissent une technique visuelle pour modéliser et dépeindre les comportements logiciels. Cependant, les diagrammes de séquence ne peuvent pas automatiquement analyser et vérifier les comportements logiciels dû au manque de sémantique stricte. Pour assurer la fiabilité du logiciel systèmes, une description du comportement et une approche de vérification formelle est proposé dans ce projet, qui utilise le diagramme de séquence SYSML et un modèle d'automate.

Premièrement, une relation complète est établie entre le diagramme de séquence et le réseau d'automates temporisés à partir de certaines règles de transformation.

Ensuite, suivant la base des règles prédéfinies, la transformation du modèle sera établie.

La vérification formelle peut être ensuite effectué pour vérifier les propriétés du domaine basées sur le langage TCTL comme étant une logique expressive non ambiguë avec un vérificateur de modèles automatisés (UPPAAL).

Notre proposition comble le fossé entre la modélisation visuelle et la modélisation formelle logiciel, ainsi qu'avancer l'étape de vérification le plus tôt possible dans le cycle de développement des systèmes hétérogène.

Notre approche a été évaluée sur un système de simulation d'ATM. L'étude de cas montre que cette approche proposée est efficace et efficiente dans le comportement, description et vérification formelle du logiciel.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besanon, France

REMERCIEMENTS

Nous ne pouvons pas laisser passer l'occasion de la présentation de ce rapport sans exprimer nos remerciements et notre gratitude à tous ceux qui ont bien voulu nous apporter l'assistance nécessaire au bon déroulement de ce projet. Nous adressons nos sincères remerciements à notre encadrant, Messabihi Mohammed, pour l'attention et l'intérêt qu'il a porté à notre travail et pour ses multiples conseils et orientations tout au long de l'élaboration de ce projet. Nous tenons aussi à témoigner notre reconnaissance envers tous nos collègues de travail pour toutes les discussions concernant ce sujet.

SOMMAIRE

I

CONTEXTE ET PROBLÉMATIQUES

1

INTRODUCTION

Les systèmes critiques temps réel deviennent de plus en plus complexes et exigeants un niveau de sûreté et de fiabilité très élevé. Cela nécessite l'utilisation des méthodes de spécification et de vérification dans la phase de conception pendant le cycle de développement de ces systèmes afin de réduire la notion d'erreur avant même de commencer l'implémentation ce qui permet le développement d'un système fiable, de réduire les coûts et de gagner du temps.

1.1/ CONTEXTE

Ce sujet est situé dans un contexte de recherche et développement pour la modélisation et la vérification de systèmes hétérogènes (par exemple des systèmes répartis sur des plateformes diverses ou des logiciels embarqués sur divers matériels). La composition souple de divers composants est une voie pour modéliser et construire de tels systèmes. Cependant afin de les analyser et garantir leur correction vis à vis des exigences, il faut formaliser les propriétés locales ou globales et les vérifier.

1.2/ PROBLÉMATIQUE

1.3/ OBJECTIFS

Les objectifs de cette thèses sont :

- Proposer un profile SysML permettant de mieux décrire les exigences afin de les intégrer dans le processus de vérification. Ce profile inclura essentiellement la description formelle des exigences.
- Exploiter les exigences étendues de ce profile pour vérifier localement chaque composant s'il satisfait bien ses propriétés (souvent comportementales). On peut se baser sur l'un des diagramme comportementaux de SysML (par exemple, diagramme de séquence ou état transition, etc.). Des outils de *model checking* peuvent être utilisés pour vérifier de telles propriétés.

II

CHAPITRE 1

2

ÉTAT DE L'ART

2.1/ LANGAGES DE MODÉLISATION

1.5 Systems Engineering Modeling Languages

Les langages de modélisation sont couramment utilisés pour spécifier, visualiser, stocker, documenter, et échanger des modèles de conception. Ils sont spécifiques au domaine et contiennent toutes les informations syntaxiques, sémantiques et de présentation concernant une application donnée dans un domaine. Différents langages de modélisation ont été définis par des organisations et des entreprises afin de cibler différents domaines tels que le développement web (WebML), télécommunications (TeD), matériel (HDL), logiciels, et plus récemment, les systèmes (UML). D'autres langages tels que IDEF ont été conçus pour un large éventail d'utilisations, y compris la modélisation fonctionnelle, la modélisation des données et la modélisation de résaux. Bien que l'ingénierie des systèmes existe depuis plus de cinq décennies, jusqu'à récemment, il n'y avait pas de langage de modélisation dédié à cette discipline. Traditionnellement, les ingénieurs systèmes ont beaucoup travaillé sur la documentation pour exprimer les exigences système et, en l'absence d'une langue standard spécifique, ont utilisé différents langages de modélisation pour exprimer une solution de conception complète. Cette diversité de techniques et d'approches a limité le travail coopératif et l'échange d'information. Parmi les langages de modélisation existants qui ont été utilisés par les systèmes ingénieurs, on peut citer HDL, IDEF et EFFBD. Afin de fournir une solution à ce problème, OMG et INCOSE, avec un certain nombre d'experts dans le domaine de l'ingénierie système, ont collaboré pour la construction d'un langage de modélisation standard pour IE. UML, étant le langage de modélisation par excellence pour l'ingénierie logicielle, était le langage de choix destiné à la personnalisation à l'égard des besoins des ingénieurs systèmes. Cependant, UML 1.x s'est révélée inadéquate pour une telle utilisation et donc la révision en évolution de UML (c'est-à-dire, UML 2.0) a été publié, avec des fonctionnalités spéciales pour les ingénieurs systèmes. En avril 2006, une proposition pour un langage de modélisation standard pour la modélisation des systèmes, à savoir SysML, a été soumis à l'OMG, avec l'objectif d'aboutir à un processus de normalisation final.

2.1.1/ UML

1.5.1 UML 2.x : Unified Modeling Language

Le langage de modélisation unifié (UML) est une modélisation visuelle à usage général

langue dont la maintenance a été assumée par OMG depuis 1997. C'est le résultat de la fusion de trois notations majeures : la méthodologie de Grady Booch pour décrire un ensemble d'objets et leurs relations, la technique de modélisation des objets de James Rumbaugh (OMT), et la méthodologie de cas d'utilisation d'Ivar Jacobson. Bien que UML ait été conçu à l'origine pour spécifier, visualiser et documenter systèmes logiciels, il peut également être appliqué à divers autres domaines tels que l'organisation des sociétés et processus d'affaires. UML a de nombreux avantages, il est largement accepté par de nombreux leaders de l'industrie, est non exclusif et extensible et est également commercialement soutenu par de nombreux outils et manuels. La norme UML a été révisé plusieurs fois et de nombreuses versions ont été publiées jusqu'à la version 2.5.1 publiée en 2018.

2.1.1.1/ METAMODEL UML

1.5.1 UML 2.x : Unified Modeling Language

L'infrastructure définit les concepts de bas niveau dans UML et représente un métamodèle. La superstructure UML concerne la définition des éléments UML. Le document de la superstructure contient la description de la syntaxe UML, y compris les spécifications des diagrammes. Il définit 13 diagrammes qui peuvent être classés en deux catégories principales : diagrammes structurels et diagrammes comportementaux. Enfin, la spécification OCL définit une langue pour écrire des contraintes et des expressions dans les modèles UML.

2.1.1.2/ ARCHITECTURE UML

2.1.2/ SYSML

1.5.2 SysML : Systems Modeling Language

2.1.2.1/ ARCHITECTURE SYSML

2.1.2.2/ AVANTAGES DE SYSML

2.1.3/ AUTOMATES

2.2/ INGÉNIERIE DIRIGÉE PAR LES MODÈLES - IDM

2.2.1/ DÉFINITIONS

2.2.2/ TRANSFORMATION DES MODÈLES

2.2.2.1/ MODEL TO MODEL

2.2.2.2/ MODEL TO TEXT

2.3/ MODÉLISATION DES SYSTÈMES COMPLEXES

2.3.1/ LANGAGE SEMI-FORMEL

2.3.2/ LANGAGE FORMEL

2.4/ VÉRIFICATION ET VALIDATION

2.4.1/ TEST ET SIMULATION (VALIDATION)

La vérification par simulation est la méthode traditionnelle de vérification. Comme son nom l'indique, elle essaye de tester le bon fonctionnement d'un composant en le soumettant à un système réel d'évaluation. Clairement le succès de cette technique compte beaucoup sur la détermination et l'attachement de l'individu qui fait le contrôle. Aussi chaque fois qu'un changement est fait au code HDL (Hardware Description Language) on a besoin du même effort pour vérifier la simulation résultante. Mais, bien que la technique standard de vérification soit le test direct, cette méthode permet d'afficher la présence d'erreurs mais n'est pas capable de prouver avec certitude leur absence. Cette conclusion découle de plusieurs arguments dont figure principalement l'impossibilité d'énumérer toutes les entrées possibles.

2.4.2/ MÉTHODES FORMELLES (VÉRIFICATION)

La vérification formelle est un processus [3] qui permet de prouver qu'un système se Comporterait en parfait accord avec sa spécification. Cela revient à utiliser des approches mathématiques qui permettent de montrer que l'implémentation satisfait la Spécification. Dans ce cas une considération de tous les cas possibles est implicite. Ces dernières années, les méthodes formelles ne sont plus uniquement des thèmes de recherche, mais plutôt un concurrent potentiel, si ce n'est pas une solution de remplacement, de

la vérification par simulation. Ces points de vue partent d'une correcte énumération des caractéristiques des méthodes formelles. Les avantages des méthodes de vérification formelle comparées à la vérification par simulation sont nombreux. En effet, ces méthodes considèrent toutes les entrées possibles au système, vérifient la validité des propriétés du système mathématiquement, ne nécessitent pas une spécification des sorties du système prévue de plus elles permettent, pour certains outils, d'identifier les traces des erreurs s'il y a lieu. La vérification formelle possède, elle Aussi, certains inconvénients [3]. En effet, elle demande un effort supplémentaire pour parvenir à une description complète et simple du système à vérifier. C'est qu'il est nécessaire de définir une spécification, d'une part, tenant en considération tous les détails du système, et d'autre part, assez simple à manipuler dans la phase de vérification. D'autre part c'est difficile de faire un raffinement sans perte des propriétés du système.

2.4.2.1/ MODEL CHEKING

2.4.2.2/ THEOREM PROVING

III

CHAPITRE 2

3

APPROCHE

Vers une Vérification Formelle des Modèles SYSML

IV

Chapitre 3

4

Séquence Formel : Génération d'automates de vérification

4.0.1/ De SYSML vers UPPAAL

4.0.2/ Algorithmes développer

4.0.3/ Étude de cas

5

Conclusion et Futur Travaux

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/