Proposition de recherche doctorale

Raffinement et Vérification d'Architectures pour la Conception de Systèmes Temps-Réels Critiques

Mots clés :

- **Directeur de thèse** : Fabrice Kordon
- **Co-encadrant(s)** :
- **Unité de recherche** : Laboratoire Traitement et Communication de l'Information
- **Ecole doctorale** : École Doctorale Informatique, Télécommunications, Électronique de Paris
- **Domaine scientifique principal**: Divers

Résumé du projet de recherche (Langue 1)

{{{Contexte}}} Lors de la conception de systèmes temps réel embarqués (STRE) les exigences non-fonctionnelles (contraintes énergétiques, temporelles, spatiales, etc.) auxquelles le système doit répondre ont une importance particulière. Cette importance vient à la fois du nombre de propriétés non-fonctionnelles, et de la nécessité de garantir leur respect pour assurer la fiabilité d'un STRE. Ceci est d'autant plus vrai dans le cas des STRE critiques, tels les systèmes autonomes (robots mobiles) ou de transport (avionique, automobile, ferroviaire), dont les défaillances peuvent avoir des conséquences catastrophiques en termes financiers ou humains. La fiabilisation de ce type de systèmes est un problème difficile, qui ne peut être résolu par le seul usage des approches basées sur les tests: -* Au niveau de la conception, il est de plus en plus communément accepté dans les systèmes critiques d'utiliser des méthodes formelles comme le {theorem proving} ou le {model checking}. L'avantage du {model checking} est que son usage peut être automatisé et requiert moins de compétences de la part de ses utilisateurs. En revanche, ils soufrent d'un problème bien connu d'explosion combinatoire en particulier dans le cas de systèmes complexes. -* Au niveau de la mise en oeuvre du système, il est de plus en plus communément accepté dans les systèmes critiques d'utiliser des techniques de génération de code pour automatiser la production du logiciel. En particulier, la génération de code prend en charge l'intégration du code métier au sein d'infrastructures d'exécution. Ainsi, les ingénieurs se focalisent sur la partie métier du système, alors que l'environnement de développement se chargeraient de l'intégration et de la validation des systèmes. De tels objectifs s'expriment aujourd'hui dans des normes internationales, comme dans l'avionique [4, 5]. De façon synthétique l'utilisation de modèles formels et de techniques de génération de code pose un problème difficile: comment garantir que les modèles formels prennent en compte les caractéristiques du système après génération de code. En effet, un processus de conception classique dans les approches MDE pour les STRE consiste à analyser un modèle vis-à-vis d'exigences puis à générer le code correspondant à ce modèle. Cependant, cette génération de code peut avoir un impact sur la validité des résultats d'analyse précédents, ce qui constitue la problématique principale de cette thèse: comment garantir que le modèle analysé prend en compte les informations produites lors de la génération de code? Dans la suite de ce sujet de thèse, nous appellerons modèle rigoureux le modèle fourni par l'utilisateur (rigoureux dans la mesure ou il dispose d'une sémantique claire et bien définie), et modèle formel celui servant à la vérification.

Résumé du projet de recherche (Langue 2)

{{{Objectifs du travail de thèse}}} Dans ce contexte, les travaux basés sur l'ingénierie dirigée par les modèles tente de répondre à ce problème en définissant différents niveaux d'abstraction pour la description d'un STRE. Cependant, ce type d'approche aboutit au paradoxe suivant: (i) si la modélisation est trop abstraite, elle ne représente pas fidèlement l'ensemble des mécanismes et interactions existantes entre les composants logiciels d'un STRE; (ii) inversement, si la modélisation est trop proche des détails d'implémentation du STRE, les outils de validation automatique se heurtent rapidement au problème de l'explosion combinatoire. Il existe diverses manières de répondre au problème de validation et par conséquent il faut adapter celui-ci en fonction (i) des objectifs en termes de précision du résultat, (ii) de la complexité inhérente à l'évaluation de la propriété non-fonctionnelle à valider (il existe des classes de problèmes plus difficiles à résoudre que d'autres), et (iii) de la complexité du système en cours de conception. Ce problème peut être décomposé dans les objectifs indiqués ci-après: -# OBJ1: Vérifier les exigences fonctionnelles et non-fonctionnelles à partir d'un niveau d'expression qui n'est pas initialement celui du langage de spécification formel. Nous utiliserons le langage AADL (({Architecture Analysis and Design Language})) pour la spécification d'architectures, et RDAL ({Requirements Definition and Analysis Language}) pour la spécification des exigences. Cela implique une traduction vers la spécification formelle, et pose des problèmes méthodologiques (voir OBJ4). Cela implique également de maintenir la cohérence entre génération de code et vérification formelle (voir OBJ3). -# OBJ2: Utiliser efficacement les techniques de vérification (combattre l'explosion combinatoire) et les approches méthodologiques (choix des techniques, choix du niveau d'abstraction, approches CEGAR - {Counter Example Guided Abstraction Refinement} - et/ou compositionnelle) pour la vérification. -# OBJ3: Définir les règles de transformation entre modèles permettant de maintenir la cohérence entre le modèle rigoureux, le modèle formel et le code généré. -#OBJ4: Prendre en compte les aspects méthodologiques, en particulier les problèmes liés au diagnostic, à l'agrégation de propriétés (abstractions en fonction des contraintes) et à la traçabilité entre la spécification de haut niveau et les différents modèles intermédiaires (modèle rigoureux, modèle formel et code généré). Le Laboratoire Traitement et Communication de l'Information (LTCI) et le Laboratoire d'Informatique de Paris 6 (LIP6) ont récemment contribué sur des approches complémentaires, qui répondent partiellement à ces objectifs. Nous présentons ces travaux dans la section suivante, avant de décrire comment nous prévoyons de les étendre au cours de la thèse. {{{Travaux en Cours}}} Les problèmes que nous avons évoqués précédemment ont fait l'objet de nombreux travaux. Nous présentons dans cette section les résultats récents obtenus par les équipes de recherche qui encadreront le travail de thèse. Le LTCI a récemment développé un environnement de raffinement de modèles dont le but est d'automatiser les étapes de conception qui visent à augmenter le niveau de détail et de précision d'un modèle de STRE. Cet environement open-source, appelé RAMSES [2] (Reffinement of AADL Models for Synthesis of Embedded Systems), vise à générer le code des STRE. Les étapes de raffinement sont mises en oeuvre par transformation de modèles et implémentent une étape classique de l'IDM: la transition PIM/PSM ({Platform Independent Model/Platform Specific Model}). En quelques mots, cette transformation de modèle interprète le comportement d'un STRE tel que spécifié dans un modèle d'architecture AADL (Architecture Analysis and Design Language) afin de représenter l'implémentation de ces comportements en s'appuyant sur des composants de la plate-forme d'exécution qui accueillera le logiciel produit. Les apports de cette transformation sont doubles: (i) la génération de code, depuis le modèle PSM est rendu simple et générique, (ii) le modèle PSM est plus précis et détaillé que le modèle PIM. Le LIP6 a récemment développé un langage de modélisation formel qui permet de représenter et d'analyser le comportement d'applications logicielles avec un formalisme proche de ceux des langages de programmation. Ce langage, appelé GAL ({Guarded Action Language}), dispose d'un langage d'action dont la syntaxe est très proche de celle du langage de programmation C. GAL propose également un modèle de composants hiérarchiques, permettant d'identifier facilement les symétries et comportements locaux d'un modèle, ce qui facilite par la suite l'analyse de propriétés. Un moteur de vérification de spécifications GAL est en cours d'évaluation et fournit déjà des résultats prometteurs. Des résultats intermédiaires relatifs à ces travaux ont déjà été publiés [6,3]. Les travaux du LIP6 et du LTCI sont complémentaires vis-à-vis de la problématique présentée précédemment: le LTCI dispose d'une approche outillée pour le raffinement de modèles et la génération de code, alors que le LIP6 propose des techniques de {model checking} avec un langage de spécification proche de celui des langages de programmation. Les modèles produits par le LTCI, couplés aux techniques de vérification du LIP6, devraient apporter une réponse intéressante au problème présenté au début de ce sujet de thèse. En plus de l'intérêt expérimental du travail de couplage, l'enjeux principal de cet axe de travail est de définir le ou les niveaux d'abstraction pertinents pour mener à bien la vérification formelle. Nous nous intéresserons également, dans ce sujet, à l'intégration de la formalisation d'exigences et à leur utilisation dans un processus de fabrication que nous définirons. En particulier, nous prévoyons d'utiliser le langage RDAL qui pourrait s'appuyer sur l'utilisation de méthodes formelles pour la validation des exigences d'un STRE. {{{Approche}}} Nous présentons dans cette section l'approche que nous souhaitons mettre en oeuvre dans cette thèse. L'objectif est, représentant dans un modèle architecturale les détails d'implémentation de ce système (incluant l'architecture du code généré), de réduire l'écart entre le modèle analysé et le modèle produit comme système final. Cependant, nous devront enrichir considérablement les travaux existants pour: -* spécifier, à partir de l'expression des exigences en RDAL, et d'un modèle architectural AADL, les transformations de modèle permettant de vérifier et d'instrumenter le code généré. Cette étape vise à répondre aux objectifs OBJ1 et OBJ2. -* formaliser et enrichir le langage pivot utilisé pour la génération de code, afin de permettre l'analyse de modèles à différents niveaux d'abstraction. Cette étape vise à répondre à l'objectif OBJ3. -* définir un processus outillé de génération de code qui alterne les transformations de modèles (AADL vers AADL) et les vérifications formelles. Cette étape vise à répondre à l'objectif OBJ4. Pour guider et évaluer les travaux, nous évaluerons l'approche au moyen de systèmes réels, comme le pacemaker[1].

Informations complémentaires (Langue 1)

Le travail proposé dans ce sujet de thèse fera l'objet d'actions de dissémination, sous la forme de collaborations internationales et de publications en conférences internationales. Compte tenu de l'utilisation du langage AADL, le sujet proposé renforcera les relations existantes avec différents acteurs du domaines: Le SEI (Software Engineering Institut), qui est fortement impliqué dans la définition du langage AADL; l'IRIT (Institut de Recherche en Informatique de Toulouse), qui travaille également sur l'analyse formelle d'architectures AADL; et l'UBS (Université de Bretagne Sud), qui a défini le langage de spécification d'exigences RDAL. L'étudiant sélectionné pour réaliser les travaux de thèse aura probablement l'opportunité de collaborer avec l'un ou l'autre de ces organismes, par exemple sous la forme d'un séjour d'étude. Les résultats seront publiés régulièrement dans les conférences internationales qui comptent dans le domaine.

Informations complémentaires (Langue 2)

[1] Boston Scientific. PACEMAKER System Specification, January 2007. [2] Fabien Cadoret, Etienne Borde, Sebastien Gardoll, and Laurent Pautet. Design patterns for rule-based refinement of safety critical embedded systems models. In Proceedings of the 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, ICECCS '12, pages 67–76, Washington, DC, USA, 2012. IEEE Computer Society. [3] M. Colange, S. Baarir, F. Kordon, and Y. Thierry-Mieg. Towards Distributed Software Model-Checking using Decision Diagrams. In 25th International Conference on Computer Aided Verification (CAV), volume to be published of Lecture Notes in Computer Science, page to be published, Rome, Italy, March 2013. Springer. [4] RTCA. DO-178C: SOFTWARE CONSIDERATIONS IN AIRBORNE SYSTEMS AND EQUIPMENT CERTIFICATION, 2011. [5] RTCA. DO-330: SOFTWARE TOOL QUALIFICATION, 2011. [6] Yann Thierry-Mieg, Denis Poitrenaud, Alexandre Hamez, and Fabrice Kordon. Hierarchical set decision diagrams and regular models. In Proceedings of the 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, TACAS '09, pages 1–15, Berlin, Heidelberg, 2009. Springer-Verlag.