

Projet CORSS

Composition et raffinement de systèmes sûrs

Rapport intermédiaire

SVF (FéRIA Toulouse), ARLES (INRIA Rocquencourt), MOSEL (LORIA Nancy),
OBASCO (EMN Nantes), PHOENIX (INRIA Bordeaux)
ACI Sécurité Informatique

avril 2005

1 Introduction

Le projet CORSS a pour but de faire collaborer des équipes issues de la communauté système et des équipes issues de la communauté méthodes formelles. Durant cette première phase, le travail a consisté principalement à établir la coopération entre les différentes équipes. Nous avons considéré trois études de cas concernant :

- Les services de téléphonie,
- Le langage de description d’ordonnanceurs Bossa,
- Les algorithmes de groupes et cohérence de données dans les réseaux Ad Hoc.

Chaque étude a donné lieu à un développement dont nous donnons l’état d’avancement dans les parties suivantes.

2 Services de téléphonie

Le protocole Session Initiation Protocol (SIP) pour la Voix sur IP (VoIP) et la téléphonie mobile de troisième génération fournit un support à la mobilité des utilisateurs et à la notification de présence. Il permet également de gérer divers types de dialogues multimédia incluant les conversations audio/vidéo, les jeux et les messageries instantanées¹

L’équipe Phoenix conçoit un langage dédié au développement de services de téléphonie. Ce langage, appelé Session Processing Language, offre des constructions et des extensions dédiées au domaine qui abstraient les difficultés des technologies sous-jacentes. De par sa conception, SPL garantit des propriétés critiques, bien au delà de celles que peuvent garantir des langages généralistes. SPL a été spécifié de manière formelle. L’équipe Phoenix a également conçu une machine virtuelle SIP qui fournit une interface haut niveau dédiée au développement de services.

2.1 Modélisation de SIP

Le groupe Phoenix avait précédemment décrit le protocole SIP selon les point de vue architectural [Lat04] et scénario d’échange [Lat04]. L’une des principales propriétés garanties par la spécification de services en SPL est le respect du protocole SIP. Afin de vérifier cette propriété, l’équipe Féria a proposé une démarche reposant sur la coopération de deux techniques de modélisation. Le protocole SIP et une abstraction de l’interprète SPL sont modélisés à l’aide de diagrammes états-transitions d’UML. L’interprète SPL est ensuite spécifié en logique d’ordre supérieur afin d’en vérifier la conformité avec son abstraction. Pour simuler le protocole

¹Bien que le protocole le permette nous ne prendrons pas en compte la gestion de conférence. On ne traitera que des communications à deux participants.

SIP et sa compatibilité avec le modèle abstrait de services, les diagrammes états-transitions sont traduits en Uppaal afin de montrer l'absence de blocage du système.

Ce premier travail sera approfondi et automatisé en utilisant l'outil RT/Hugo développé par les partenaires de l'équipe Mosel.

2.2 Modélisation du filtre SPL

L'équipe Phoenix a défini la sémantique du langage SPL [BCL⁺is] et les règles de vérification statique. La modélisation de l'infrastructure SPL concerne l'interaction entre l'interprète SPL et le niveau SIP. Il s'agit d'assurer la transparence du filtre SPL vis à vis du niveau SIP. Ce filtre SPL est décomposé de la manière suivante :

- un module plate-forme réalisant l'interface entre l'interprète SPL et SIP.
- une partie applicative comportant l'interprète SPL et le code du service (écrit en SPL).

2.2.1 Plateforme

La figure 1 montre l'organisation générale du filtre, elle doit être interprétée de manière symétrique : le filtre peut recevoir et modifier des messages provenant du serveur.

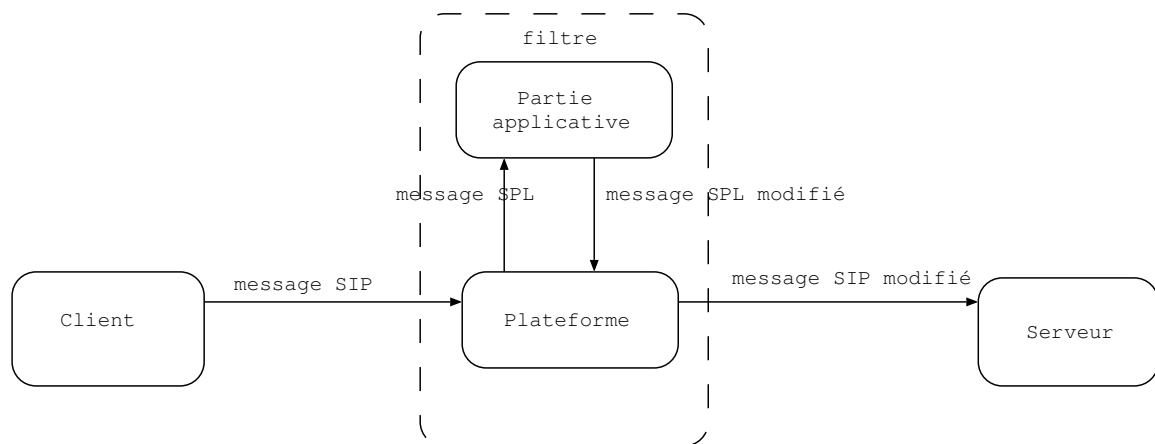


FIG. 1 – Architecture du filtre

La plateforme permet de recevoir les messages SIP, de les transmettre à la couche applicative et de transmettre à la couche SIP les messages modifiés par la couche applicative. Mais elle ne transmet pas tous les messages SIP qu'elle reçoit. Si le même message SIP est reçu plusieurs fois elle ne le transmettra qu'une seule fois. Elle permet donc à la partie SPL de ne traiter que des scénarios ne comprenant pas de répétitions inutiles de messages.

2.2.2 La partie applicative

La partie applicative est écrite en SPL. Elle reçoit les messages de la plate-forme. Elle les transforme éventuellement, en fonction de son état interne, puis les renvoie vers la plate-forme afin qu'ils soient adressés à leurs destinataires. Le langage SPL ne permet aucune création de message, seule une modification de certains champs de messages est possible.

2.3 Méthodologie de vérification

Les deux parties du filtre vont être modélisées dans deux formalismes différents. La partie plate-forme comporte essentiellement du contrôle et sera modélisée par des automates temporisés. La partie applicative

est concernée par la correction des services utilisateur exprimés dans le langage SPL, elle sera formalisée en logique d'ordre supérieur. Pour assurer l'indépendance de ces vérifications, il est nécessaire d'introduire une abstraction de la partie applicative exprimable en termes d'automates temporisés.

Le langage SPL permet, via une mémoire globale, d'introduire des interdépendances entre les sessions, ce qui rend la vérification en termes d'automates finis impossible. Afin d'éliminer cette interdépendance, nous avons introduit une abstraction du filtre SPL décorrélant les sessions en supprimant cette mémoire globale. Le déterminisme du langage est alors perdu. L'abstraction nécessite encore une mémoire par session conservant le dernier message envoyé. L'abstraction et la plateforme SIP peuvent alors être modélisés par un produit d'automates indépendants, indexés par les sessions. Cette information peut donc être ignorée pour vérifier la compatibilité de l'abstraction avec la plateforme SIP. Une modélisation en logique d'ordre supérieur (en Isabelle) est ensuite nécessaire pour vérifier la conformité de l'interprète SIP avec le modèle abstrait non déterministe.

En résumé, la validation se fait en deux étapes :

- Validation de la correction de l'interprète SPL par rapport à l'abstraction (en Isabelle),
- Validation de l'interaction entre l'abstraction de la partie applicative, la plate-forme et l'environnement SIP.

2.3.1 Modélisation de la partie applicative

La partie applicative est définie par un programme SPL. Afin de valider la compatibilité entre l'interprète SPL et le modèle abstrait, nous allons définir en Isabelle la sémantique de SPL et montrer que cette sémantique est bien un raffinement du modèle abstrait de la partie applicative.

2.3.2 Modélisation de la plateforme

Le comportement de l'ensemble du système client/serveur SIP et plateforme est modélisé par un ensemble d'automates temporisés en UPPAAL. Un automate représente une abstraction non déterministe de la partie applicative. UPPAAL nous permet de simuler le fonctionnement du système avec des pertes éventuelles de messages. On peut alors vérifier qu'un couple d'automates représentant un client et un serveur SIP ne change pas de comportement si on ajoute un automate modélisant le filtre.

2.3.3 Extensions envisagées

Après avoir validé la plate-forme d'exécution, nous envisageons de définir un langage de propriétés de niveau applicatif et d'étudier les mécanismes de preuve associés.

3 Bossa : un langage de description d'ordonnanceurs

Le langage Bossa [LLMM04] est un DSL (Domain Specific Language) dédié pour l'implantation d'ordonnanceurs de systèmes. Bossa facilite l'intégration de nouveaux ordonnanceurs dans des systèmes généralistes (Windows, Linux) et dans des noyaux de systèmes, e.g., embarqués. Bossa propose une approche à deux niveaux. Le premier niveau décrit les états, les événements et transitions d'états autorisés par le système sous-jacent et nécessite une connaissance approfondie du fonctionnement du noyau. Le deuxième niveau permet de décrire dans un langage spécialisé (le DSL) divers algorithmes d'ordonnement. Le compilateur Bossa vérifie que l'ordonneur respecte bien la politique décrite dans la première phase et génère le code C d'un module pouvant être directement chargé dans le noyau.

Lors de cette première année, nous avons étudié la formalisation de l'approche Bossa à l'aide de la notion de raffinement. Ainsi, dans le contexte de la méthode B, la méthodologie Bossa de développement d'un ordonnanceur peut être illustrée par la figure 2.

Le principe de base de cette traduction est le suivant :

- un scheduler est spécifié à l'aide d'une machine de spécification (.mch).
- une politique est exprimée à l'aide de machines de raffinements (.ref).

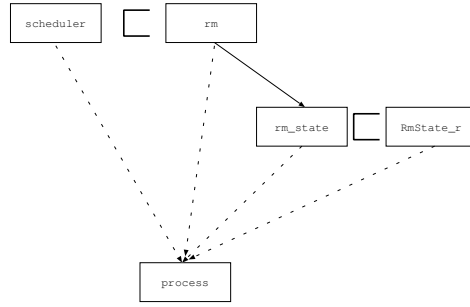


FIG. 2 – Architecture de la modélisation B

A chaque événement Bossa est associée une opération B.

L'étude que nous avons menée peut aussi être vue comme la définition d'une sémantique dénotationnelle pour Bossa. A chacune des règles Bossa, nous avons associé une opération B. A titre d'exemple, une telle opération B a le type suivant :

$$GState \times (* src *)Process \times (* dst *)Process \rightarrow GState \rightarrow bool$$

où les types `GState` et `State` sont spécifiés ainsi :

```

type GState = (running : Process  $\times$  state : Process  $\rightarrow$  State)
        State  = {NoWhere, Running, Blocked, Ready, Terminated}

```

Valider un développement Bossa consiste donc à valider un raffinement entre des opérations du type précédent. Toutefois, il est à noter que, contrairement à l'approche actuelle Bossa où toutes les preuves sont automatiques grâce à des techniques issues de l'interprétation abstraite, e.g., l'analyse de flux, ici nous sommes dans un contexte de preuve assistée et le taux de preuve automatiquement «déchargées» par le(s) prouveur(s) de l'atelier B est encore faible. Plus précisément, si en ce qui concerne les aspects méthodologiques, nous avons développé une architecture de machines B dans laquelle toutes les obligations de preuves sont «déchargées» automatiquement, il nous reste à travailler les aspects qui concernent les implantations effectives de politiques. Il nous semble tout à fait raisonnable d'envisager l'écriture de règles de preuve, de procédures de décision, voire de machine prédéfinies (de bibliothèques) qui permettraient alors de prouver automatiquement un bon nombre d'obligations. Enfin, on notera que le travail que nous avons fait n'est pas spécifique pour la méthode B. Moyennant le développement de la mécanisation d'un noyau de développement par raffinement, il est tout à fait possible d'envisager la même approche dans d'autres cadres logiques tels que ceux offerts par les assistants de preuve : Coq, Isabelle et PVS.

Enfin, il nous semble aussi intéressant pour la suite du projet de :

- finaliser les travaux sur les liens entre Bossa et les méthodes formelles classiques telles que B. Les travaux actuels ont principalement considéré l'approche B *classique*, nous étudierons les apports du B événementiel.
- Valider la correction de la description en Bossa de politiques d'ordonnancement.
- Développer un environnement de preuve permettant de raisonner sur des programmes parallèles qui seront ordonnancés à l'aide de schedulers développés à l'aide de Bossa.
Cet environnement de preuve pourrait se situer soit dans un contexte assisté (qui s'appuiera sur la mécanisation vue précédemment), soit dans un contexte automatique (utilisation de vérificateurs de modèles : Tina, ...).

4 Groupes et cohérence de données dans les réseaux Ad Hoc

Les travaux du projet ARLES portent sur la notion de groupe d'une part et d'autre part, sur la cohérence de données répliquées dans le contexte des réseaux ad hoc (MANET). En effet, ce type de réseau aux

connexions très évolutives nécessite de revisiter des algorithmes étudiés sous les hypothèses très statiques dans le cadre des systèmes distribués.

Dans ce cadre les travaux suivants ont été réalisés :

- Etude et modélisation d’un algorithme de gestion de groupe conçu dans le cadre du projet ARLES. La modélisation du protocole de constitution de groupes de processus a été réalisée en TLA+. Plusieurs versions différentes représentant différents niveaux d’abstraction ont été programmées par les équipes de Toulouse et Nancy. Celles-ci ont permis de mettre en évidence les propriétés de base de l’algorithme proposé.
- Etude d’un protocole de cohérence de données répliquées dans le cadre du projet ARLES. Cette étude a fait l’objet d’un DEA en 2004 [Per04].

Les principaux enseignements tirés de cette étude sont les suivants :

- La difficulté de vérifier les propriétés de l’algorithme par parcours exhaustif des états (model checking). L’explosion combinatoire des états a limité les vérifications à quelques sites (< 5);
- La vérification des propriétés de l’algorithme nécessite de faire des hypothèses minimales de stabilité momentanée du système global. Cette approche particulière mérite d’être approfondie car elle soulève des problèmes de validité des modélisations compte tenu des hypothèses faites au niveau de la modélisation du réseau par exemple;
- Ces modélisations ont conduit à formaliser les propriétés assurées par l’algorithme et à proposer une nouvelle approche dans la constitution des groupes de processus. Ceci a fait l’objet d’une soumission [FIM⁺05] au symposium IPDCS 2005 (4th International Symposium on Parallel and Distributed Computing) à Lille. Un travail de DEA est en cours pour parvenir à la preuve formelle de ce nouvel algorithme.

5 Conclusion

Comme nous l’avons dit en introduction, lors de la première phase, le travail a consisté principalement à établir la coopération entre les différentes équipes. Chacune des études considérées a permis :

- soit de confronter les approches généralement utilisées par les équipes issues de communautés différentes (SPL, Bossa).
- soit de revisiter des protocoles pour un nouveau contexte applicatif (MANET).

Dans le cadre de la suite des travaux, nous envisageons de consolider les études entamées et d’approfondir des processus de validation dédiés aux domaines considérés.

Références

- [BCL⁺is] Laurent Burgy, Charles Consel, Julia Lawall, Nicolas Palix, and Laurent Reveillère. Telephony software engineering : A domain specific language approach. Technical report, LABRI, 2005 (soumis).
- [FIM⁺05] Mamoun Filali, Valérie Issarny, Philippe Mauran, Gérard Padiou, and Philippe Quéinnec. Maximal membership in ad hoc networks. Technical Report 2005-07-R (soumis), IRIT, Université Paul Sabatier, 118 Route de Narbonne, F-31062 Toulouse cedex, avril 2005.
- [Lat04] Fabien Latty. SIP architecture éléments de message. Technical report, ENSEIRB, 2004.
- [LLMM04] Julia Lawall, A.-F. Le Meur, and Gilles Muller. On designing a target-independent DSL for safe OS process-scheduling components. In *Third International Conference on Generative Programming and Component Engineering (GPCE’04)*, volume 3286 of *Lecture Notes in Computer Science*, pages 436–455, Vancouver, october 2004. Springer-Verlag.
- [Per04] Florent Peres. Modélisation et vérification d’algorithmes répartis. Rapport de DEA programmation et systèmes (EDIT). Technical report, IRIT, Université Paul Sabatier, 118 Route de Narbonne, F-31062 Toulouse cedex, juillet 2004.