RenPar'2000
Besançon, 19 juin - 22 juin 2000

PLERS : Plateforme Logiciel Embarqué Reconfigurable pour Satellites – Application au satellite Corot †

Bertil Folliot[1,2], Damien Cailliau[3], Ian Piumarta[1,2], Remy Bellenger[3]

[1]Laboratoire d'Informatique de Paris 6 / CNRS
Université Pierre et Marie Curie
4 Place Jussieu
F-75252 Paris Cedex 05

[2]Institut National de la Recherche en Informatique et Automatique
Projet Système à Objets Répartis
Domaine de Voluceau, Rocquencourt
78153 Le Chenay Cedex

[3]Observatoire de Meudon
Département d'Etudes Spatiales / CNRS
5 Place Jules Janssen
92195 Meudon Cedex

Couriel : Bertil.Folliot@lip6.fr, Damien.Cailliau@obspm.fr, Ian.Piumarta@inria.fr, Remy.Bellenger@obspm.fr

---

Résumé

Beaucoup de logiciels embarqués souffrent d'une trop grande rigidité qui limite les possibilités de maintenance, d'évolution ou de reconfiguration. Ceci est particulièrement vrai des logiciels embarqués dans un environnement aussi contraignant que l'espace. Ce papier présente les besoins et contraintes en reconfigurabilité des logiciels spatiaux puis décrit la spécification d'un système de contrôle-commande permettant de piloter des modifications fonctionelles et d'implantation du logiciel de vol de l'instrument COROT. Le principe en cours de réalisation s'articule autour d'un langage de configuration et d'une machine virtuelle, pilotant les requêtes de reconfiguration entre le sol et l'instrument. Une telle plate-forme apporte l'indépendance désirée entre le pilotage des reconfigurations et leur implémentation, permettant de vérifier au sol les contraintes temps-réél du système.

Mots-clés : Systèmes reconfigurables, logiciels spatiaux, commandabilité, évolutivité

---

1. Introduction

Il est très difficile de pouvoir surveiller et maintenir un logiciel embarqué lorsque son lieu d'opération est éloigné, soit physiquement, soit parce que les possibilités de contact sont restreintes. Ceci est particulièrement vrai des logiciels embarqués sur satellites, sondes planétaires ou robots opérant sur d'autres corps du système solaire où seules des interventions sporadiques et non interactives à distance sont possibles.

Les modifications depuis le segment sol de ces logiciels embarqués sont donc une opération critique, utilisée à l'heure actuelle en cas d'urgence pour corriger des erreurs matérielles ou logicielles dans les systèmes de contrôle. Elles se basent sur une technique consistant à "brutalement" remplacer le code binaire, et restent sévèrement limitées (coût en communication, pas de retour arrière possible, etc.). Ces méthodes interdisent toute réelle optimisation de l'usage du matériel en modifiant les fonctionnalités réalisées. Ceci fige les traitements réalisés à bord lors de l'étape de spécification de l'expérience, bien avant son exploitation – plusieurs années séparent l'étude de faisabilité d'une mission de sa mise en service – et peut priver la communauté scientifique astrophysique d'avancées majeures. Enfin, la durée de vie des

†Ce travail est partiellement financé par le projet LIP6 PLERS

satellites dépasse souvent celle de la mission scientifique. La encore, la rigidité du logiciel embarqué ne permet pas de charger, à un coût dérisoire, de nouvelles missions scientifiques.

Le satellite d'astrophysique français Corot - COnvection, ROtation et Transits planétaires - [1], dont le lancement est prévu fin 2004, pour une durée d'opérations de deux ans et demi, est le premier à notre connaissance à instaurer comme contrainte utilisateur la possibilité de modifier en cours de mission les traitements réalisés par l'instrument. Nous appelons de telles modifications de haut niveau fonctionnel *reconfigurations*.

Le projet PLERS (Plateforme Logicielle Embarquée Reconfigurable pour Satellites), regroupant l'Observatoire de Meudon, le LIP6 et l'INRIA s'attache à apporter une solution à cette problèmatique. Dans [3] nous présentions les contraintes et principes mis en oeuvre pour rendre le logiciel de Corot reconfigurable. Dans ce papier nous proposons un système de contrôle et de commande de ces reconfigurations. Après une analyse des besoins en terme de reconfigurabilité des logiciels du domaine spatial, nous proposons section 3 une solution répondant au cahier des charges Corot. Un résumé des techniques de modification et de reconfiguration de logiciel actuellement utilisées dans divers domaines de l'informatique est présenté dans la partie 4. Enfin, la section 5 conclue sur les perspectives de tels systèmes.

2. Reconfigurabilité des logiciels spatiaux - Besoins et Contraintes

[11] présente quelques raisons justifiant la nécessité d'étudier de nouvelles architectures logicielles qui soient dynamiquement reconfigurables. La majeure partie de ces raisons reste valable, mais est complétée, dans un contexte spatial, par un certain nombre de contraintes plus spécifiques.

2.1. Besoins

Fautes matérielles et humaines

Les satellites et sondes spatiales souffrent souvent de malfonctions et de pannes dues à l'environnement sévère dans lequel ils évoluent : vibrations au lancement pouvant mener au blocage d'un des panneaux solaire, impacts de protons pouvant détruire des cases mémoire ou des capteurs. Certaines de ces pannes peuvent être prises en compte en adaptant le logiciel de pilotage et de gestion bord à la nouvelle situation, soit par correction, soit pour poursuivre la mission en mode dégradé. Or la difficulté d'interaction et l'éloignement du système en poste rendent très délicate toute opération de maintenance du logiciel – 100% de fiabilité et de tolérance aux fautes est un but évident mais rarement atteint dans le développement de systèmes embarqués, a fortiori pour les logiciels. Que la possibilité de telles modifications de logiciel soit prévue par conception permettrait d'éviter de perdre entièrement des systèmes très onéreux.

Evolution et réutilisation

Une autre classe de justification pour l'évolutivité des logiciels spatiaux réside plus au niveau fonctionnel. Les durées de transfert des sondes planétaires (un survol de Pluton impliquerait un voyage de 10 ans), ainsi que la durée de vie des satellites (le matériel reste souvent utilisable au-delà de la durée d'opération prévue, lorsque l'ensemble des objectifs scientifiques ont été remplis) permettrait de faire évoluer les logiciels embarqués de manière à les faire bénéficier des évolutions scientifiques ou de leur apporter de nouvelles fonctionnalités (résultant de réorientations des objectifs, de prise en compte de nouveaux résultats ou d'ajout de capacités). Par ailleurs, la réutilisation de satellites permettrait de continuer des missions à un coût raisonnable. Tout ceci n'est possible que si ces logiciels sont évolutifs.

Cas particulier de COROT

Corot se trouve dans la situation peu classique où une modification totale par traitement des données brutes est effectuée à bord pour extraire et formater l'information scientifiquement significative, ce de manière à rendre le volume de données compatible avec le lien de transmission vers le sol. Habituellement, la majeure partie des missions ont soit une bande passante de transmission de données au sol plus élevée, soit un volume moyen d'information acquise beaucoup plus réduit. Or les traitements bord sont basés sur des hypothèses sur la magnitude des perturbateurs du signal utile, hypothèses elles-mêmes basées sur des modèles théoriques. Cependant, ces modèles n'ont pas été confrontés aux conditions réelles de la mission. Il est donc nécessaire de pouvoir tester, modifier ou supprimer ces algorithmes, voire d'en ajouter au cas où la réalité en orbite s'écarterait de ces hypothèses de manière à ne pas perdre l'ensemble de la mission.

2.2. Contraintes

La situation particulière des logiciels embarqués en milieu spatial impose un certain nombre de contraintes. Tout d'abord l'absence de réelle interaction entre le segment sol et l'instrument : COROT offre 2 à 4 brefs contacts par jours, permettant uniquement l'envoi vers le satellite d'un volume de 140 kbits/jour. La capacité du lien descendant est de 500 Mbits par jours, essentiellement monopolisée par les données scientifiques. Par ailleurs, le lien ascendant ne permet pas d'établir un dialogue à chaque contact : la réponse à une requête du sol ne pourra être reçue que lors du passage suivant du satellite en vue de la station de communication. Ce manque d'interactivité impose que le flot d'information "non utile" (autre que les données) soit le plus faible possible, tout en étant exhaustif.

En outre, de part les contraintes usuelles de sécurité et de tolérance aux fautes, il est nécessaire que toute reconfiguration puisse être exactement réversible de manière à éviter les modifications définitives et à aisément pouvoir retrouver une configuration du logiciel stable.

De plus, la technique adoptée devra être simple à mettre en oeuvre, avoir peu d'impact sur l'architecture et la conception du logiciel (la reconfiguration ne doit pas être la contrainte de conception dominante), être peu intrusive et ne pas remettre en cause les modes de développement et de programmation usuels dans le domaine satellitaire, de manière à être facilement utilisable. Il est également interressant que les reconfigurations aient peu d'impact sur le déroulement du logiciel, donc de permettre un certain dynamisme.

Dans le contexte de satellites scientifiques tels Corot, il y a une séparation nette au sein du segment sol entre Segment Sol Opérationnel (SSO) et Segment Sol de Traitement Scientifique (STS). Le premier est chargé des opérations de contrôle-commande du satellite et de l'instrument, généralement pris en charge par une Agence Spatiale, alors que le deuxième s'occupe du traitement sol des données et de la planification des observations, pris en charge par les laboratoires d'astrophysique intervenant. Les concepteurs des modifications à apporter au logiciel (spécifiant le code à développer et l'aspect final du logiciel) font donc nécessairement partie du STS, alors que leur mise en oeuvre est prise en charge par le SSO. Ceci impose que des outils simples, cohérents, utilisables par des non spécialistes du logiciel embarqué soient conçus, permettant de planifier, vérifier et suivre les opérations de reconfiguration.

3. Solution technique

3.1. Configuration

Le logiciel de vol de Corot est un logiciel qui s'articule autour d'un OS s'appuyant sur un noyau multitâche. Ainsi une *configuration* consiste en une collection de tâches appelant des fonctions et connectées entre elles par des canaux de communication pour former des chaînes de traitement concurrentes, pilotées par les données. La partie haute de la Figure 1 présente une configuration comportant 2 tâches communiquant par l'intermédiaire d'une boite-aux-lettres (MB) et appelant des fonctions.

La reconfigurabilité est assurée par l'usage d'indirections, ou proxies (non représentés dans la Figure 1), entre les différents éléments utilisés par chaque tâche. L'indépendance ainsi fournie permet à des services spécifiques de l'OS, les services de gestion de configuration, de modifier les éléments d'une configuration et leur organisation les uns par rapport aux autres.

3.2. Reconfiguration

On appelle donc *reconfiguration* le passage d'une configuration à une autre. Etant donnée l'architecture du logiciel de vol de Corot, ces modifications reviennent à la création, destruction ou modification de tâches, de canaux de communications entre tâches, mais aussi des fonctions utilisées par ces tâches.

La Figure 1 donne un exemple de ces niveaux de reconfigurabilité. A la configuration initiale sont ajoutés une tâche (Task3) et l'ensemble des éléments dont elle a besoin, ainsi qu'un canal de communication de manière à réaliser l'insertion de cette nouvelle tâche dans le système (le flot de données). Une telle modification suggère la capacité de modifier l'agencement des tâches les unes par rapport aux autres, et donc le séquencement des opération. De manière sous-jacente, l'ajout, la modification ou le retrait de code du logiciel implique l'ajout, la modification ou le retrait de paramètres utilisés par ce code.

3.3. Commande-Contrôle des reconfigurations

La chaîne de mise en oeuvre des reconfigurations s'articule autour de l'ensemble des moyens de développement et de test du logiciel de vol tels que définis lors de la conception de la version initiale (localisés

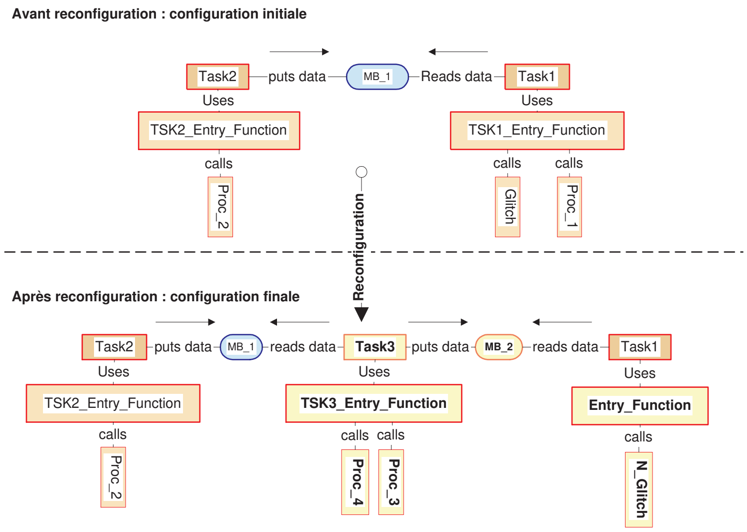

FIG. 1 – Configuration et reconfiguration

dans le STS) d'une part, et, d'autre part de trois applicatifs :

- Un simulateur d'ordonnancement dont le rôle est de vérifier que les modifications escomptées sur le logiciel respectent les contraintes temporelles du système. Cet applicatif s'intègre dans la suite de développement du logiciel, prise en charge par le STS et n'est pas détaillé ici.
- Un extracteur de code dont le rôle est d'extraire et de mettre en forme l'ensemble du code des nouvelles fonctionnalités que doit intégrer le logiciel.
- Un planificateur de reconfiguration dont le rôle est de regrouper les buts de reconfiguration souhaités et de générer l'ensemble des commandes (les scripts) permettant d'atteindre ces buts de manière optimisée (optimisation sur le nombre d'opérations, sur l'allocation mémoire des nouveaux éléments, etc.).

Ce dernier applicatif, présenté Figure 2, réalise la chaîne de contrôle-commande des reconfigurations. Il s'articule autours d'un langage de configuration de haut niveau, en cours de spécification, le CCL (pour COROT Configuration Language) et d'une machine virtuelle dédiée à CCL. Le langage CCL possède un niveau de description de configuration et un niveau de description d'opérations de reconfiguration. Le CCL est un langage compilé en bytecode éxécutable par la machine virtuelle. Dès que définie et validée par le STS, une description en CCL de la configuration finale souhaitée est fournie au planificateur de reconfiguration qui en vérifie la validité. Par comparaison avec le descripteur de la configuration active à bord, il traduit un script CCL d'opérations de configuration, compilé et exécuté par une machine virtuelle s'interfaçant avec l'"extracteur de code". La machine virtuelle permet de retourner à l'opérateur, par l'intermédiaire d'une interface fonctionnelle, l'effet attendu de ces commandes sur la configuration, appelé f(x). Le script est alors traduit en octets, eux-même empaquetés dans le format de télécommandes spécifique à COROT, en vue de transmission.

A bord du satellite, les scripts sont reconstruits à partir des paquets de télécommande reçus. La machine virtuelle identique à celle du segment sol exécute le script de manière à reproduire exactement la même séquence d'opérations de reconfiguration f(x). Ces opérations sont fournies aux services de gestion de configuration de l'OS de manière à être mises en oeuvre. Ainsi la machine virtuelle s'interface directement avec les primitives des services de gestion de configuration chargés de la création, de la destruction et de la modification des tâches, fonctionnalités et canaux de communication utilisés par l'application. Une fois la nouvelle configuration installée et vérifiée à bord, un diagnostique est émis vers le planificateur de reconfiguration de manière à ce qu'il mette à jour son descripteur de configuration active. Il est à noter que l'extraction de code n'est pas une étape nécessaire. En effet, un satellite tel Corot possède à bord de la mémoire protégée lui permettant de stocker du code non actif. Ainsi, pourvu que l'ensemble du code nécessaire soit présent à bord, une reconfiguration peut être très simplement mise en oeuvre par émission

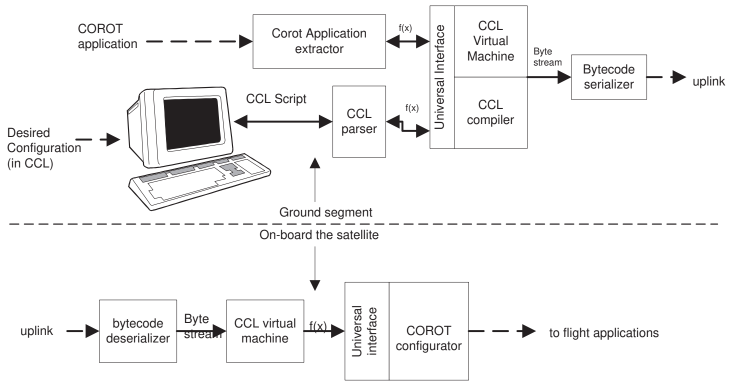

FIG. 2 – Système de commande-contrôle de reconfiguration du logiciel de vol Corot

seule d'un script de reconfiguration.

La caractéristique majeure d'une telle conception est que l'interface fonctionnelle vue par l'opérateur est la même que celle vue par le gestionnaire de configuration. Ceci facilite grandement la validation et permet un développement indépendant des étapes intermédiaires. La machine virtuelle CCL implémente ainsi une fonction identité entre les commandes déterminées et simulées au sol et les opérations de configuration implémentées par les services de gestion de configuration à bord du satellite. Un autre avantage d'un tel mode de fonctionnement est qu'il isole parfaitement la conception du logiciel de la commande de reconfiguration. Les interfaces entre STS et SSO sont suffisamment simples et indépendantes de l'implémentation pour être comprises et utilisées par des non spécialistes du logiciel. L'usage d'un langage de haut niveau permet non seulement des vérifications formelles sur les reconfigurations (par exemple sur l'ordonnancement temps réél), mais ouvre également une perspective de ré-utilisabilité des outils développés et du code : une configuration est un ensemble de briques modulaires dont l'agencement est piloté par un système indépendant de leur implémentation.

4. Autres approches

Jusqu'à présent, les interventions effectuées sur les satellites scientifiques ont été des opérations de maintenance, comme dans le cas du satellite ESA-NASA Soho [10], ou plus récemment la sonde NASA Galilleo en orbite autour de Jupiter. La technique du patch utilisée permet très peu de souplesse dans les modifications. De même, à part dans le cas du satellite scientifique italien BeppoSAX [8], peu d'études systèmatiques ont été menées en vue de concevoir un segment sol de maintenance du logiciel bord qui soit souple, allège les procédures et assure le maximum de fiabilité. Dans le cas de BeppoSax, le concepteur du logiciel de vol contrôle l'ensemble de la chaîne de commande et de maintenance. Le système mis en place s'appuie intensivement sur des modèles simulés et physiques du satellite de manière à pouvoir mettre au point et valider les patchs à envoyer avec un maximum de sécurité. Les procédures et les équipements nécessaires en deviennent très lourds, ce d'autant plus que le logiciel n'a pas été conçu dans l'optique d'être intrinsèquement reconfigurable, mais "patchable".

Un autre axe suivi dans le secteur spatial est l'autonomie : le logiciel de vol est équipé d'agents intelligents capables de gérer les erreurs et d'assurer une mission par la simple connaissance de grands buts pré-programmé. L'exemple le plus marquant est l'expérience Remote Agent [9] embarquée sur la sonde de la NASA Deep Space 1 qui se déroula avec succès.

[12] propose une méthode permettant de concevoir des logiciels de robotique reconfigurables dynamiquement en se basant sur un noyau multitâche dédié. Il propose dans [7] un environnement graphique de commande et déploiement de configuration, mais qui n'offre pas à l'heure actuelle de réel dynamisme, au sens où les séquences de reconfiguration doivent être programmées à l'avance. En outre l'ensemble du système proposé s'appuie fortement sur un lien de communication interactif.

D'autres secteurs de l'informatique s'interessent également à aux systèmes dynamiquement reconfigurables. C'est le cas des études sur le parallèlisme (citons par exemple [4]) ou sur les applications réparties.

Suivant un axe similaire au notre, [2] analyse l'interêt de l'usage de langages de description d'architecture (ADL) dans la définition d'application réparties reconfigurables basées sur des agents.

Enfin, [6] propose de définir une plateforme d'exécution dynamiquement reconfigurable, indépendante de l'implémentation, capable d'intégrer dynamiquement n'importe quel type de langage. La notion de Machine Virtuelle Virtuelle (MVV) étudiée permet de porter un pas plus loin l'intégration entre langage et système [5].

5. Conclusion

Il existe un besoin fort d'un système permettant de réaliser des logiciels spatiaux intrinsèquement reconfigurables, d'alléger les procédures sol, de réduire le flot de données échangées, de fiabiliser l'ensemble des opérations et de modifier le logiciel à un haut niveau fonctionnel. Cet article analyse les besoins et contraintes sous-jacents à la modification de logiciels embarqués sur des satellites scientifiques, notamment dans le cadre de la mission Corot. Les principes présentés, le langage de script CCL, la machine virtuelle dédiée ainsi que les chaînes de traitement, sont en cours de réalisation. Cette étude est une première étape vers la définition d'une solution générique de plateforme logiciel embarqué reconfigurable pour satellites.

Bibliographie

1. Auvergne M., Baglin A., et al. – Du coeur des étoiles aux planètes habitables, les enjeux de Corot – Journal des Astronomes Français, N. 60, p. 27–34, 1999.
2. Bellissard L., de Palma N., Riveill M. – Dynamic Reconfiguration of Agent-based Applications – Proceedings of ACM European SIGOPS Workshop, Sintra, 7-10 Sept. 1996.
3. Cailliau D., Bellenger R. – The Corot instrument's software : towards intrinsically reconfigurable real-time embedded processing software in space-borne instruments – Proceedings of the 4th IEEE International Symposium on High Assurance System Engineering (HASE'99), Washington, DC., p. 75–80, Nov. 17-19, 1999.
4. Derk M. D., debrunner L. S. – Reconfiguration for fault tolerance using graph grammars – ACM Trans. on Computer Systems, Vol. 16, No. 1, p. 41–54, February 1998.
5. Baillarquet C. – MVV : langage & système, plus qu'un mariage de raison – Proceedings of the First ACM/SIGOPS French Conference on Operating Systems (CFS'1), Journée des Jeunes Chercheurs (JCS99), June 8-11, Rennes, France, 1999.
6. Folliot B., Piumarta I., Riccardi F. – A dynamically configurable, multi-language execution platform – Proceedings of SIGOPS'98 Workshop, Sintra, Portugal, Sept. 1998.
7. Gertz M. W., Stewart D. B., Kholsa P. K. – A human-machine interface to support reconfigurable software assembly for virtual laboratories – IEEE Robotics and Automation Magazine, Vol. 1, No. 4, p. 5–13, December 1994.
8. Martinelli A., Torchia F. – The software maintenance in the BeppoSax scientific mission – Proceedings of the Data Systems in Aerospace Conference (DASIA'97), Sevilla, Spain, p.369–374, May 26-29, 1997.
9. Muscettola N., Pandurang Nayak P. et al. - Remote agent : to boldly go where no AI system has gone before – Artificial Intelligence, Vol. 103, No. 1–2, p. 5–48, Aug. 1998.
10. Olive J. P. – Soho Recovery – Proceedings of the Flight Mechanics Conference, 1999.
11. Stewart D. B., Arora G. – Dynamically reconfigurable software – Does it make sense ? – Proceedings of IEEE International Conference on Engineering of Complex Computeur Systems (ICECCS) and Real-Time Application Workshop (RTAW), Monteréal, Canada, p. 217–220, Oct. 21-25, 1996.
12. Stewart D. B., Volpe R. A., Khosla P. K. – Design of diynamically reconfigurable real-time software using port-based objects – IEEE Trans. on Software Engineering, Vol. 23, N. 12, p. 759–776, December 1997.