
# Réseau d'Ingénierie de la Sûreté de fonctionnement

# COMPTE RENDU

## de l'atelier thématique du 19/10/2001

### « Usages et perspectives pour la production de logiciel sûr »

## 1. Déroulement de l'atelier

Le deuxième atelier thématique du *RIS* a eu lieu le 19 octobre 2001 au LAAS-CNRS sur le thème « Usages et perspectives pour la production de logiciel sûr ». Etaient représentés : AIRBUS France, ASTRIUM, LAAS-CNRS, TECHNICATOME, RATP.

L'atelier était centré sur des présentations de la pratique et des perspectives en matière de production de logiciel sûr. Le sujet a été introduit par une présentation de la problématique générale et des objectifs de l'atelier tels qu'ils étaient fixés lors de la réunion du Comité d'Orientation du *RIS* du 19 avril 2001. L'atelier a été clos par des discussions sur les suites envisageables. Le présent compte rendu a pour objectif de faire une synthèse des présentations et discussions de cet atelier.

## 2. Participants

Jean Arlat (LAAS), Jean-Pierre Auclair (Consultant), Jean-Paul Blanquart (ASTRIUM), Philippe Chevalley (LAAS), Yves Crouzet (LAAS), Frédéric Cuppens (ONERA), Philippe David (ASTRIUM), Jean-Charles Fabre (LAAS), Olivier Fernandez (AIRBUS France), Mohamed Kaâniche (LAAS, animateur de l'atelier), Olfa Kaddour (LAAS), Gérard Ladier (AIRBUS France), Jean-Claude Laprie (LAAS), Raymond Marie (IRISA), Loïc Pelhate (RATP), David Powell (LAAS), Famantanantsoa Randimbivololona (AIRBUS France, rapporteur de l'atelier), Juan-Carlos Ruiz Garcia (LAAS), Jean-Michel Tabart (TECHNICATOME), Pascale Thévenod (LAAS), Hélène Waeselynck (LAAS), Patrick Welby (ASTRIUM). Les coordonnées des participants sont données en annexe.

## 3. Programme

| | |
|---|---|
| *Introduction et présentation de l'atelier* | Mohamed Kaâniche (LAAS-CNRS) |
| *Vérification de logiciels avioniques par analyse statique* | Famantanantsoa Randimbivololona (AIRBUS France) |
| *État des lieux et perspectives pour le développement des logiciels embarqués sur satellite* | Patrick Welby (Astrium) |
| *Processus de maîtrise du développement d'un logiciel de sécurité à Technicatome* | Jean-Michel Tabart (Technicatome) |
| *B : une méthode de développement de logiciels sûrs* | Loïc Pelhate (RATP) |
| *Définition d'une stratégie de test d'un protocole à méta-objets* | Juan-Carlos Ruiz Garcia (LAAS-CNRS) |
| *Discussion générale et synthèse* | |

# 4. Synthèse des présentations

**Introduction et présentation de l'atelier (LAAS-CNRS)**

La production de logiciel sûr nécessite aujourd'hui des efforts, en matière de coûts, élevés. Une part de plus en plus importante est due aux phases de vérification. Afin de fixer les idées et de caractériser l'état de l'art actuel, l'effort de développement de tels logiciels se situe en moyenne dans une fourchette de 5 à 10 hommes.années par mille lignes de code. Le pourcentage de cet effort consacré aux vérifications est en moyenne de l'ordre de 50 à 75 %. L'une des principales causes la plus communément admise est la complexité croissante des logiciels et les difficultés, de plus en plus grandes, pour maîtriser leur développement, surtout dans un contexte où les contraintes économiques imposent des délais de plus en plus courts. De plus, l'introduction de nouvelles technologies de conception pose aussi des problèmes de vérification nouveaux.

L'analyse de l'état de l'art montre, de façon générale, une séparation très forte des activités de création d'un logiciel sûr des activités liées à sa vérification. Cette séparation est tout à fait dommageable sur le plan économique dans la mesure où fort peu d'informations relatives aux processus de développement sont effectivement utilisées pour faciliter et éventuellement alléger les phases de vérification. Le problème qui se pose est alors le suivant : Étant donné que l'on développe du logiciel sûr en s'appuyant sur des méthodes rigoureuses, comment s'y prendre pour le vérifier de façon efficace, et dans quelle mesure certaines activités traditionnelles de vérification (par exemple, les tests unitaires ou les tests d'intégration) peuvent elles être supprimées sans dégrader le niveau de confiance du logiciel produit ?

Une utilisation plus poussée des méthodes formelles durant le développement peut contribuer à aboutir à une meilleure coopération entre les activités de création et de vérification du logiciel par la rigueur de la démarche et par la possibilité de faire des vérifications (à base de modèle ou par preuve de propriété) au plus tôt dans le cycle de vie. Ces méthodes ont été longtemps l'objet d'un fort scepticisme dans l'industrie, néanmoins le succès de l'expérience METEOR a montré que l'emploi de telles méthodes dans un processus industriel est possible et présente en même temps des avantages multiples. Par conséquent, il est important d'analyser dans quelle mesure une telle démarche est applicable dans d'autres domaines d'application. Outre la formalisation du processus de développement, une solution au problème posé peut être envisagée à travers un outillage et une automatisation plus poussés de certaines étapes du processus (par exemple, génération automatique du code, des séquences de test, etc.) et par la sélection de formalismes permettant d'aborder le développement et la vérification du logiciel de façon intégrée de bout en bout (ces approches sont étudiées par exemple dans le cadre des projets Safeair ou Crysis).

**Vérification de logiciels avioniques par analyse statique (AIRBUS France)**

Le développement des logiciels avioniques embarqués est guidé par le cadre défini dans la norme DO-178B. En particulier, les activités de vérification sont essentiellement basées sur des tests et des revues manuelles. Néanmoins, avec l'augmentation de la taille et la complexité des logiciels et l'évolution des technologies matérielles, la satisfaction des exigences de vérification définies dans la norme DO-178B en s'appuyant sur ces techniques de vérification s'avère de plus en plus difficile et coûteuse.

Afin de résoudre ce problème AIRBUS a mené depuis plusieurs années des travaux en recherche et développement concernant l'introduction de techniques de preuves de propriétés et de vérification par analyse statique (logique de premier ordre, interprétation abstraite), en remplacement au moins partiel des techniques de vérification par exécution. Il est envisagé d'appliquer ces techniques au niveau du code source (ou bien du binaire dans certains cas), et non pas au niveau des phases amont de spécification ou de conception.

L'objectif visé est de réduire l'effort de vérification tout en respectant les critères de la DO-178B en utilisant une stratégie de vérification mixte combinant des preuves, des tests et des revues manuelles. Ces travaux de recherche et développement (menés en particulier en collaboration avec le CEA ou dans le cadre du projet IST DAEDALUS) ont confirmé l'intérêt de ces approches nouvelles par rapport à l'état actuel de la pratique dans le domaine du logiciel avionique civil. Il est envisagé de commencer à utiliser ces techniques sur les logiciels de l'A380.

**États des lieux et perspectives pour le développement des logiciels embarqués sur satellite (Astrium)**

La sûreté de fonctionnement du logiciel dans les systèmes spatiaux repose principalement sur les processus utilisés pour produire et valider les logiciels en mettant l'accent sur la prévention et sur l'élimination des fautes. En général, le développement de logiciel sûr n'est pas contraint par un cadre normatif similaire à celui de l'avionique par exemple. La tendance vers l'augmentation de complexité et de criticité des logiciels, vers la diminution des coûts et des délais, et l'évolution tant des technologies logicielles que des architectures matérielles, conduisent à faire évoluer également les processus et les outils utilisés pour le développement et la validation du logiciel. Cette évolution s'avère aussi nécessaire afin de mieux maîtriser les interactions entre les processus de développement système et les processus logiciel.

La présentation a fait le point sur les approches actuellement utilisées ou étudiées pour le logiciel spatial. L'accent a été mis sur les aspects liés au prototypage, au choix de formalismes de modélisation adaptés au contexte des logiciels embarqués dans les systèmes spatiaux, à la génération automatique de code et de tests, à la réutilisation et à l'intégration dans le processus industriel d'approches dites "d'ingénierie simultanée" pour le développement conjoint du matériel et du logiciel. En particulier, l'analyse des méthodes et outils de modélisation a porté sur les approches dites déclaratives (B, Z, VDM) qui s'avèrent cependant difficilement applicables, sur les approches synchrones (Lustre-SCADE, Esterel, MatrixX,…) qui sont envisagées pour prendre en compte les aspects temps réel, et enfin sur les approches asynchrones pour modéliser les communications, les algorithmes de traitement de données et les performances réseaux (ObjectGeode, Tau SDL, Rhapsody, Rose RT, Opnet).

Les problèmes engendrés par la réutilisation ont également été soulevés. En particulier, la réutilisation n'est pas nécessairement source de réduction de coûts surtout si le modèle de cycle de vie n'est pas adapté pour répondre aux besoins de la réutilisation et prendre en compte l'impact des évolutions des spécifications au cours du cycle de développement. En effet, une modification même mineure des spécifications système en cours de développement peut se traduire par le fait qu'une solution de réutilisation envisagée en début du cycle devienne impossible à mettre en œuvre.

**Processus de maîtrise du développement d'un logiciel de sécurité à Technicatome**

On distingue deux catégories de logiciels liés à la sûreté : les logiciels de niveau A, critiques vis-à-vis de la sûreté et associés à une architecture de calculateur de sécurité, et les logiciels de moindre criticité dits de niveau B.

Pour la première catégorie, le développement du logiciel est guidé par les référentiels définis dans la norme CEI 880 pour les applications du domaine nucléaire et la norme CENELEC EN 50128 pour les applications du domaine ferroviaire. Généralement, ces logiciels sont de faible complexité (taille inférieure à 15000 lignes de code source) et leur développement est actuellement bien maîtrisé, le processus évoluant par petites touches pour traiter les éventuels problèmes identifiés. Les spécifications et conceptions sont basées sur des méthodes semi-formelles (SA-RT, SD et outil Team Work). Les vérifications effectuées à ces étapes du cycle de vie sont basées sur des analyses manuelles sans réelle exécution dynamique. Les activités de tests sont effectuées selon le processus traditionnel du cycle en V en utilisant comme outil de support ATTOL pour les tests unitaires et l'environnement ATLG pour les tests d'intégration et de validation sur calculateur cible.

On identifie davantage d'interrogations concernant le développement de logiciels de niveau B. Les questions qui sont posées concernent l'introduction de technologies à base de COTS (matériels et logiciels), l'utilisation d'approches orientées objet (UML, C++) et la génération automatique de code.

**B : une méthode de développement de logiciels sûrs (RATP)**

Les systèmes de transport utilisent des automatismes de contrôle-commande qui sont majoritairement constitués de logiciels. La problématique de la validation de ces logiciels de sécurité a poussé les industriels et les exploitants à utiliser des méthodes formelles de développement afin de garantir le niveau de sécurité atteint. Ces méthodes lorsqu'elles intègrent un mécanisme de preuve, permettent de démontrer que le logiciel satisfait certaines propriétés dès les premières phases de conception. L'intérêt de ces méthodes est aussi d'éviter de faire certains tests traditionnels, par exemple les tests unitaires.

Cet exposé a porté sur un bilan concernant l'utilisation de la méthode B pour le développement de logiciels sûrs pour la RATP, en particulier dans le cadre de METEOR, et sur les utilisations futures et extensions envisagées de cette méthode. La RATP intervient en tant que maître d'ouvrage. L'utilisation de B dans le cadre de METEOR a permis une meilleure maîtrise des phases de spécification, de conception et de réalisation du logiciel par la rigueur de la démarche, l'incorporation de preuves aux étapes successives d'affinements, et l'utilisation de la génération automatique de code. Concernant les coûts, ils sont comparables à ceux d'un logiciel développé et validé selon des méthodes classiques. En particulier, l'effort de test a été considérablement réduit puisque les tests unitaires ont été complètement remplacés par la preuve. On peut noter aussi que 85% des preuves ont été traités de façon automatique, le reste étant traité en semi-automatique. Outre METEOR, la méthode B a été appliquée à d'autres systèmes ferroviaires, mais également à d'autres domaines d'application, par exemple l'automobile et les cartes à puces.

Le logiciel de METEOR est séquentiel et non interruptible. La méthode B est bien adaptée pour ce type de caractéristiques. Pour les systèmes présentant des contraintes temps réel, des travaux en cours étudient l'utilisation de la méthode B dans le développement de ce type de système à travers le couplage de l'atelier B avec l'outil SCADE. Les autres extensions envisagées concernent l'utilisation de B pour effectuer des preuves au niveau système (B-événementiel et B-système).

**Définition d'une stratégie de test d'un protocole à méta-objets (LAAS-CNRS)**

Les contraintes sur le coût de développement et de maintenance des logiciels sûrs de fonctionnement, associés à des besoins de flexibilité accrus, incitent à l'utilisation de composants logiciels hautement réutilisables et facilement adaptables. Les systèmes réflexifs, tels que le système *FRIENDS* développé au LAAS, possèdent la flexibilité nécessaire pour faire face à ces défis. Ces systèmes s'appuient sur une approche de développement orientée objets et en particulier sur les protocoles à méta-objet.

L'utilisation de la technologie objet est généralement destinée à des logiciels qui ne sont pas critiques. Le principal problème posé par l'utilisation de cette technologie dans le cadre de systèmes critiques est lié à la difficulté de vérifier ces systèmes avec un niveau d'assurance en conformité avec les exigences des normes.

Les travaux présentés dans cet exposé visent à apporter quelques solutions à cette problématique. En particulier, la présentation a porté sur la définition et l'expérimentation d'une stratégie globale de vérification des systèmes réflexifs et en particulier des protocoles à méta-objet.

# 5. Synthèse des discussions et conclusion de l'atelier

Les discussions ont couvert un large spectre et ont généralement focalisé sur des problèmes spécifiques liés aux différentes présentations. Par exemple :

- critères de qualification des outils utilisés durant le processus de production de logiciel sûr,
- problèmes liés au téléchargement et la reprogrammation de logiciel en vol,
- problèmes posés par la réutilisation,
- difficulté de maîtriser les évolutions des spécifications et les interactions entre les processus système et logiciel,
- intérêt pour l'intégration de plusieurs formalismes avec des interfaces permettant de passer d'un formalisme à l'autre,
- limites de la méthode B vis-à-vis de la prise en compte de contraintes temps réel,
- introduction de nouvelles technologies (matériel et logiciel) et impact sur la vérification.

Les discussions générales en fin de journée ont eu pour objectif de faire un tour de table pour recueillir les sentiments des différents participants vis-à-vis de l'intérêt de mettre en place un groupe de travail traitant du thème de l'atelier. Les présentations ont montré que, bien que tous les partenaires industriels s'accordent sur le besoin de trouver des solutions permettant d'optimiser l'effort de vérification en s'affranchissant par exemple des phases de test unitaires ou des tests d'intégration, les orientations envisagées par chacun d'eux divergent.

Cette situation soulève une question intéressante : les différents partenaires ont des problèmes similaires mais ont choisi des orientations différentes pour les appréhender avec plus au moins de succès. Il serait donc intéressant d'étudier pourquoi de telles différences existent et de les analyser en profondeur. L'explication peut venir de l'analyse des cultures de chaque entreprise et des processus mis en œuvre pour le développement de logiciel sûr. Néanmoins, on peut se demander si ce constat, à lui seul, est suffisant pour justifier la mise en place d'un groupe de travail.

## Annexe 1 - Quelques pointeurs sur sites INTERNET

Méthode B :
http://www.lsr.imag.fr/B

Projet DAEDALUS (preuves par interprétation abstraite)
http://www.di.ens.fr/~cousot/projects/DAEDALUS/index.html

Projet Safeair :
http://safeair.org/safeair.htm

Projet Crysis :
http://borneo.gmd.de/~ap/crisys