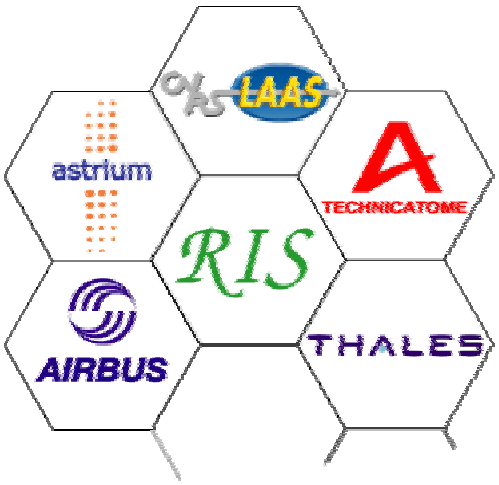

R éseau d'*I* ngénierie de la *S* ûreté de fonctionnement

COMPTE RENDU

de la première réunion du GT « Intergiciel et sûreté de fonctionnement »

Mardi 3 Juin 2003 — LAAS-CNRS

1. Introduction – Jean-Charles Fabre (LAAS-CNRS)

Faisant suite aux recommandations du Comité d'Orientation du *RIS* lors de la réunion qui s'est tenue le 19 Mars 2003, le Groupe de Travail (GT) portant sur « Intergiciel et Sûreté de Fonctionnement » a été mis en place. Son objectif est de préparer à terme une proposition de projet sur ce thème à l'horizon 2004. Les motivations sont nombreuses pour justifier cette stratégie.

Les groupes de travail précédents ont été motivés par le souci de se focaliser sur la synthèse d'une problématique.

En matière d'intergiciel, l'atelier que nous avions tenu sur le sujet en Juin 2002 nous a montré que les partenaires disposaient d'une maturité certaine sur ce thème, et d'une expérience pratique importante. En conséquence, il semble qu'un projet issu du *RIS* soit le bon cadre, à terme, pour factoriser les expériences, mettre en commun des moyens, des composants logiciels, des outils, mais aussi attaquer des problématiques communes. Un GT du *RIS* sur ce sujet semble être alors un environnement tout à fait opportun pour réfléchir ensemble à cet objectif et le concrétiser.

Notre proposition repose donc sur ce constat et veut se donner les moyens d'analyser dans un premier temps la pertinence de cette approche. L'idée est de procéder en deux temps:

1) Analyser les compétences des partenaires, leur complémentarité, les problématiques sur lesquelles des efforts de recherche et développement sont nécessaires,

2) Produire une proposition de projet sur la base des enseignements tirés de l'étape 1.

L'organisation pour procéder à cette analyse repose sur deux premières réunions de travail.

- Réunion 1: analyse des compétences, des complémentarités et des problématiques des partenaires
- Réunion 2: factorisation des objectifs et des motivations (contributions potentielles, problèmes ouverts) des partenaires pour un projet.

La synthèse de ces deux réunions décidera de la forme de poursuite des travaux. En bref, il nous semble que cette action, si elle se confirme, est une bonne façon de pérenniser le *RIS*, sous une autre forme, au delà de son terme, par exemple dans un cadre d'un projet RNTL.

L'objectif de cette première réunion du groupe de travail "Intergiciel et Sûreté de Fonctionnement" est donc de synthétiser les savoir-faire et d'identifier les thématiques de R&D dans la perspective de la préparation d'un tel projet commun.

Nous avons souhaité que les partenaires puissent exprimer leur savoir-faire sur ce thème ainsi que leur

prospectives de R&D dans ce contexte. A la suite des différentes interventions nous aurons un créneau de synthèse et de discussion.

Un certain nombre de partenaires intéressés n'ont pas pu être présents, notamment en raison des perturbations dans les transports. Le programme a dû être modifié en conséquence, mais tous les partenaires du *RIS* sont représentés et pourront donc s'exprimer lors de cette première réunion. Les intervention des partenaires du *RIS* seront donc les suivantes (voir liste des participants, section 11):

- Olivier Laurent-Rousseau AIRBUS
- Marie-Line Valentin AIRBUS
- Patrick Pleczon ASTRIUM
- Christophe Honvault ASTRIUM
- Jean-Denis Jouffrit TA
- Marc-Olivier Killijian LAAS-CNRS

Brigitte Bauer nous apporte le point de vue de THALES AVIONICS.

Philippe Chevalley de l'ESA (*European Space Agency*) s'est joint à nous lors de cette journée et présente le point de vue de l'ESA sur ce thème.

Jean-Charles Fabre donne donc la parole aux différents intervenants en les remerciant encore pour leur participation.

Les planches présentées lors des exposés sont données en annexe.

2. Intervention d'Olivier-Laurent Rousseau (AIRBUS)

L'objectif des travaux sous-jacents à cette présentation est la simulation dans le contexte aéronautique. Nous utilisons une plate-forme CORBA comme support pour effectuer de la simulation distribuée. Clairement, nos besoins dans les années à venir portent sur le temps réel et la sûreté de fonctionnement.

Nous avons souhaité développer un simulateur sur plate-forme à bas coût : Linux / Windows. Des applications distantes (dites "satellites") communiquent avec le cœur de la simulation au travers de modules de communication *ad-hoc*.

Notre objectif dans ces travaux a été de factoriser la partie communication (donc au niveau "intergiciel"), pour pouvoir éclater le cœur de la simulation sur plusieurs machines. Quelques données chiffrées sont données ici pour fixer les idées :

- Nombre de modèles: plus de 70
- Nombre d'acteurs: 5
- Nombre total des données: + 50000
- Nombre données échangées: + 20000
- Intervalle entre les échanges : < 50 ms

En pratique, notre premier objectif a été de synchroniser deux contrôleurs et rendre transparent la communication. Les solutions techniques envisagées a priori ont considéré l'utilisation de mécanismes de mémoire partagée et du service d'événements CORBA.

- Mémoire partagée: Sans matériel spécifique, cette technique a induit beaucoup de développements dans le monde universitaire, mais il existe peu de produits aboutis bâtis sur ce type de mécanisme. En outre, beaucoup de problèmes de performance ont été relevés par rapport à nos besoins. En conséquence, cette solution n'a pas été retenue.
- Service d'événements CORBA : Cette technique s'appuie sur le principe du Producteur/Consommateurs, un seul producteur pour chaque donnée. Bien que transparent au

niveau localisation, nos tests de performances (sur TAO, en particulier), ont été peu convaincants. Parmi les problèmes, notons la notion de données indépendantes, c'est-à-dire un canal par donnée, ce qui imposerait la mise en place d'un système de production/consommation sur 20000 canaux différents. Compte tenu du nombre important d'événements et de données de petite taille échangées, cette approche est mal adaptée à l'arrivée d'un nouvel acteur**.** Le Service d'événements CORBA n'a pas non plus été retenu.

Nous avons alors étudié deux solutions maisons, reposant sur:

- Un contrôleur central effectuant une ré-émission en fonction de ses connaissances. Il représente un point dur (*single point of failure*, goulot d'étranglement) et donc nécessite une attention particulière en ce qui concerne les performances et la sûreté de fonctionnement.
- Un contrôleur autonome, jouant un rôle symétrique vis-à-vis des acteurs, qui effectue une phase de négociation pour la découverte des schémas d'échange, le transit d'information s'effectuant de manière directe du producteur au consommateur

La solution retenue est mixte entre les deux précédentes : il existe un contrôleur central pour la synchronisation entre acteurs mais les échanges se font directement entre les acteurs, ces-derniers étant mis en œuvre sur CORBA.

En guise de bilan, les services CORBA n'ont pas répondu au problème de façon satisfaisante et l'apport se situe au niveau de la facilité de mise en œuvre, ainsi que l'interopérabilité niveau langage et plates-formes

J.-C. Fabre : Question sur la robustesse par rapport à la charge (*scalability*)?

O.-L. Rousseau: Principe de publication: chaque producteur produit toutes ses données à chaque cycle, indépendamment des modifications effectives. Le volume est constant.

M.-O. Killijian: Le problème de classement des données est difficile. Travail trop coûteux ? Vraiment trop difficile ?

O.-L. Rousseau: Beaucoup de données sont transmises en point-à-point dans l'avion. Le réseau de distribution est en fait assez complexe. Personne n'a pu nous donner de notion de famille (critères), mais plutôt des critères techniques (dynamicité très importante, difficulté de la communication avec les personnes métier).

F. Taiani: Quelles sont les contraintes de sûreté de fonctionnement?

O.-L. Rousseau: il faut survivre à la disparition de l'un des acteurs. Il n'y a pas de contraintes énormes. Il faut garantir du temps réel "mou" (performance).

J.-C. Fabre : Pourquoi CORBA ? Pourquoi un intergiciel ? Tout en java, même facilité, et beaucoup moins lourd. CORBA n'est-il pas trop lourd ?

O.-L. Rousseau: Ce que vous venez de dire est en partie la conclusion de notre travail. L'architecture globale évoluera peut-être vers une disparition de CORBA.

B. Bauer: Rien n'assure que TAO soit sur toutes les plates-formes que vous rencontrerez ?

O.-L. Rousseau: On a effectivement rencontré ce cas. Portage sur Windows avec extensions temps réel.

E. Marsden: Dans le cas particulier de TAO vous pouvez faire le portage.

O.-L. Rousseau: Effectivement, dans le cas d'un ORB commercial cela ne serait peut-être même pas possible.

3. Intervention de Marie-Line Valentin (AIRBUS)

Nous nous intéressons ici au monde de l'embarqué, c'est-à-dire que nous nous posons la problématique de l'utilisation d'intergiciel dans l'embarqué.

Il s'agit d'un monde dit « ouvert », c'est-à-dire que les applications embarquées dont on parle ici sont en dehors du monde avionique. Elles concernent principalement le domaine de la maintenance, de la documentation. Elles communiquent avec le monde avionique au travers de passerelles sécurisées

(notion de *firewall*) et il n'y a pas de certification requise pour ce type d'application (niveau E du DO-178B)

Dans ce contexte, nous recherchons des technologies logicielles à bas coût avec des intergiciels COTS.

Nous avons identifié différentes zones de confiance :

- applications de l'avionneur (maintenance) ;
- applications en cabine ;
- applications propres à la compagnie aérienne.

J.-C. Fabre : Quel type de *firewalls* utilisez vous? Des *firewalls* classiques ?.

M.-L. Valentin : Ce sont des *firewalls* classiques, mais critiques, donc de degré de certification élevé. En effet, la connexion monde-ouvert / systèmes avioniques pose de nouveaux problèmes de sécurité (intrusion, virus) et donc nécessite une architecture beaucoup plus rigide.

- Ségrégation des machines services d'audit / services rendus.
- Partitionnement du réseau en fonction de la confiance.

Chaque serveur est en fait partagé entre un serveur opérationnel et un serveur applicatif. Seul `https` est autorisé pour la communication entre les réseaux.

- Serveur AP (applicatif): il ne communique pas directement avec le monde avionique (Niveau d'étanchéité supplémentaire).
- Serveur OP (opérationnel): il s'agit d'héberger tous les modules des applications qui communiquent avec le monde avionique via les passerelles sécurisés. A tout moment, les ressources nécessaires doivent être disponibles ("Déterminisme" nécessaire).

Nous appliquons, autant que faire se peut, une réduction des services de base sur le serveur OP. Il existe en effet des règles strictes de communication avec le monde ouvert. Il faut être très prudent vis-à-vis de l'utilisation généralisée de COTS / intergiciels pour des raisons de sûreté / sécurité. Mais l'ouverture est souhaitée fortement par les compagnies pour qu'elles puissent intégrer leurs applications. En même temps, nous devons impérativement imposer des contraintes sur l'étanchéité.

J.-C. Fabre : Pour répondre à ces deux aspects, est-il nécessaire de définir des API strictes et acceptées par les parties en présence ?

B. Bauer : Nous sommes au début de la réflexion. Aujourd'hui nous n'avons pas assez de recul, ce n'est pas complètement mûr, nous n'avons pas assez d'expérience, ces approches ne sont pas encore suffisamment répandues. De plus, c'est difficile à normaliser et chacun des avionneurs est un peu libre.

J.-C. Fabre : Quelle est la solution d'AIRBUS ?

M.-L. Valentin : Pour nous, ce sont des mondes relativement étanches.

J.-C. Fabre : Plus cohabitation que coopération, donc ?

B. Bauer: Le niveaux de confiance est fonction du fournisseur. A chaque fois que l'on peut sortir des fonctions non critiques, on le fait.

P. Chevalley : Nous avons les mêmes besoins dans le spatial, comme la définition d'interfaces normalisées, de techniques d'étanchéité / confinement et d'interopérabilité.

J.-D. Jouffrit : Même problème pour TA.

M.-O. Killijian : Les serveurs AP sont-ils complètement libres ?

M.-L. Valentin : Non. En terme de plate-forme, il s'agit d'un OS (sans doute Linux) et de services de base.

F. Taiani: Quelles procédures pour s'assurer des différents degrés de confiance des applications ?

M.-L. Valentin : Utilisation d'audit des applications. On leur demande de donner des informations sur ce qu'elles font.

M.-O. Killijian: Dans le monde de la sécurité, on constate beaucoup mise à jour, de « *patches* ».

M.-L. Valentin: Oui, c'est aussi pris en compte.

B. Bauer : Quid des problèmes d'obsolescence, d'évolution de versions.

M.-L. Valentin: C'est moins le cas sur les serveurs OP, mais à prendre en compte absolument pour serveurs AP.

4. Intervention de Patrick Pleczon (ASTRIUM)

Le contexte des travaux est ici celui des applications sols et de l'utilisation d'intergiciel. Nous n'avons pas de contraintes temps-réel dur, seulement des contraintes de performance.

Par le passé, nous avons développé nos intergiciels en interne, avec des langages de type *IDL (Interface Definition Language)* « maison », pour des applications précises. CORBA n'existait pas encore, ou dans les cartons.

Aujourd'hui CORBA existe sous deux formes:

- ORB Commerciaux
- ORB Open Source (TAO)

Dans le futur, nous pouvons aussi considérer l'utilisation de *Fault-Tolerant CORBA* (FT CORBA), ce qui n'est pas le cas aujourd'hui.

En ce qui concerne les applications, nous pouvons citer *Open Center*, qui est un *framework* logiciel ainsi que des composants à la fois techniques et très métier. Il permet la réalisation de différents systèmes:

- centres de contrôle
- bancs de test
- simulateur de satellites

En termes de problématique, nous nous focalisons sur la gestion de la redondance, la synchronisation, la redondance des services CORBA qui, bien que jusqu'à récemment peu utilisés le deviennent de plus en plus (par exemple, le *Naming Service*). Il s'agit en effet d'un point unique de défaillance.

A titre d'exemple, dans le centre de contrôle Intelsat FDC il y a de l'ordre de 60 processus sur 5 machines et c'est *Orbix* qui est utilisé. Dans la pratique, nous avons dû résoudre des problèmes posés par *Orbix*. Les solutions ont été trouvées, mais elle restent aujourd'hui peu réutilisables. FT CORBA apporte des réponses en partie, mais il reste cependant des problèmes sur la synchronisation des états. Jusqu'à *Orbix-3*, les mécanismes sont propriétaires dans *Orbix*.

J.-C. Fabre : Commentaire : FT-CORBA c'est un standard mais qui ne fait que décrire des boites vides. Rien dans FT-CORBA n'est mis à disposition pour gérer / maîtriser l'état à l'exécution d'un processus. Beaucoup de sujets délicats sont laissés ouverts et donc à résoudre par l'intégrateur: détection des défaillances, gestion de l'état, etc.

E. Marsden : Quel est le contrat de service avec TAO ?

P. Pleczon: Pour l'instant il n'y en a pas. On s'est débrouillé sans (grâce aux forums dédiés en particulier).

J.-C. Fabre : Comment les gens vivent-ils ça ?

P. Pleczon: Les plus grosses préoccupations aujourd'hui sont en fait d'ordre légal, ou des réticences à passer vers ces solutions OpenSources (mais les choses ont beaucoup évoluées sur ce point). D'un point de vue technique, ça marche plutôt mieux qu'avec les solutions commerciales.

5. Intervention de Christophe Honvault (ASTRIUM)

Nous nous intéressons ici aux logiciels embarqués dans le spatial avec des architectures multi-

couches. En bref, une architecture typique s'organise comme suit :

- Exécutifs temps-réel : Historiquement faits maisons, maintenant ils sont souvent sur étagère comme VxWorks ou RTEMS ;
- Gestion des équipements ;
- Applications systèmes (télécommandes / télémesures) et interpréteurs de commandes ;
- Applications missions: très haut niveau (SCAO, etc.)

Beaucoup d'architectures sont centralisées (par exemple, E3000 similaire à Rosetta), mais il existe quelques expériences distribuées (par exemple COF, dans le laboratoire européen de la station spatiale internationale ISS). Pour ce qui concerne les noyaux temps-réels de base, nous utilisons le noyau VxWorks qui a été légèrement modifié (les sources ont été achetés pour permettre ces modifications).

Le projet A3M est un projet de R&D emblématique de cette nouvelle tendance aux architectures distribuée temps-réel tolérant les fautes. Ce projet a été mené pour le compte de l'ESA par Astrium et la société Axlog, avec une collaboration du LAAS et de l'INRIA.

Dans COF les temps de latence étaient très importants (RPC). Dans A3M, les temps de latence beaucoup plus faibles et les systèmes sont hétérogènes.

Dans A3M, les composants COTS sont utilisés au niveau exécutif (A3M est basé sur VxWorks). Au départ, l'idée était d'utiliser CORBA, mais nous avons rapidement constaté qu'un tel intergiciel était assez lourd et pas adapté pour du temps réel dur. CORBA à été abandonné, dans un premier temps pour les couches basses de la plate-forme. Le choix que nous avons fait s'est porté sur le développement d'un intergiciel minimal contenant les seuls services nécessaires au temps-réel et à la tolérance aux fautes. Nous avons procédé en deux phases :

- Phase 1: Analyse et modélisation globale de la problématique ;
- Phase 2: Implémentation de services de bases sur noyau temps-réel.

Astrium s'est tout particulièrement chargé des communications bas niveau (matériel et logiciel) et des applications de test. L'architecture repose donc sur un RTOS auquel ont été adjoint des modules spécifiques de communication. Au-dessus nous avons développé une couche de composants fondamentaux: détection de *crash* de processeur (par messages périodiques), consensus, coordination, en s'appuyant sur des algorithmes génériques.

J.-C. Fabre : Commentaire : la détection est effectuée à un niveau très bas. Les algorithmes sont asynchrones, mais pour la détection il est nécessaire de connaître les latences de communication.

C. Honvault : Des services fondamentaux permettent d'assurer que tout le monde reçoit les mêmes données dans le même ordre. Ils permettent aussi de décider de l'ordonnancement des tâches dans le système avec partage de ressources.

B. Bauer : Notion de tâche = tâche de VxWorks ?

C. Honvault : oui.

J.-C. Fabre : Une application est un ensemble de tâches TR distribuées. Il s'agit d'un saut technologique : « le temps réel distribué » dans ce contexte. Il existe, à partir d'une capture du problème, un pré-ordonnancement pour assurer que 2 tâches concurrentes sur une donnée partagée seront sérialisées. Les mécanismes sont utilisés pour permettre aux processeurs distribués de décider de la prochaine tâche à ordonnancer.

Sur cette architecture, nous avons effectué le développement d'applications d'évaluation: processus dupliqués avec vote majoritaire, mise en cohérence de données partagées, sur une architecture matérielle composée de 3 sites. La plate-forme matérielle se compose 3 cartes BLADE, équipées de processeur LEON sur FPGA. Le LEON est un Sparc V8 pour le spatial, dont les performances sont cependant de beaucoup inférieures aux processeurs commerciaux.

Nos perspectives à partir de l'intergiciel de base sont, de développer un intergiciel plus évolué, en tenant compte des problèmes de puissance de calcul limitée dans notre domaine. De fait, une sonde

dispose d'une très faible bande passante avec la terre. Notre souhait est de pouvoir déporter le maximum de calcul sur la sonde. Une idée est donc de pouvoir utiliser plusieurs processeurs. Un autre exemple est celui des vols en formation, l'idée étant de mener des expériences avec plusieurs engins qui tolèrent la défaillance d'un élément, même en mode dégradé. Bien entendu nous espérons trouver des solutions pour faciliter la gestion cohérente de telles expériences. Il existe aussi des domaines tels que la robotique spatiale, mettant en jeu des systèmes collaboratifs avec partition/répartition de la charge.

Pour la puissance de calcul, une idée serait d'utiliser un processeur commercial très puissant mais qui défaille très souvent. Les simulations actuelles sont défavorables à cette solution. Les défaillances peuvent être permanentes ou temporaires. Dans le cas de défaillances temporaires, comment reprendre les calculs interrompus dans une boucle temps-réel de manière fiable ? Des problèmes non triviaux sont à résoudre.

Nos axes d'études portent sur les systèmes présentant un besoin de collaboration, avec des communications à haut débit. Dans le cas de calculateurs physiquement distribués, la communication doit être fiable avec une faible latence.

Au niveau de l'architecture des logiciels, il faut constater que l'on ne fait pas un système distribué comme on fait un logiciel centralisé. Quelles fonctions dans les couches intergiciels ? *Codesign hardware / software* ?

Les résultats d'A3M sont très encourageants et correspondent aux attentes de l'ESA dans ce type d'étude. Il existe d'autres applications pour l'ESA, telles qu'un système embarqué de haut niveau qui supervise et vérifie (notion de *safety bag*) toutes les commandes envoyées aux actuateurs. Il détermine si la télécommande met en danger l'engin.

Le projet A3M se termine en Septembre 2003.

J.-C. Fabre : Commentaire : dans l'architecture ELEKTRA d'Alcatel pour l'aiguillage ferroviaire informatisé, il y a 6 calculateurs, dont 3 pour l'aspect fonctionnel de l'application, et 3 pour le *safety-bag*, à savoir la vérification de propriétés de sécurité-innocuité.

E. Marsden: Les performances du LEON sur FPGA sont faibles. Y-a-t il un consortium pour faire évoluer le LEON ?

C. Honvault : Oui. Aujourd'hui cadencé à 25MHz, le but d'Astrium est d'influencer l'évolution pour faire intégrer nos besoins. En particulier, nous souhaitons disposer d'une MMU. Aujourd'hui, il n'y a pas de MMU dans le LEON. Côté performance, ce qui pose problème ce sont les communications.

6. Intervention de Jean-Denis Jouffrit (TECHNICATOME)

Dans notre domaine qui est celui du Contrôle-Commande de chaufferies nucléaires, un intergiciel correspond à une couche intermédiaire qui sert à coordonner des applications réparties. Dans le contrôle commande de chaufferie nucléaire, il y a 3 niveaux de sécurité: A (cœur), B, C. Le niveau B est de nature cyclique (acquisition, calcul, actions), 24h/24 sans interventions. La pérennité est extrêmement importante. Le type d'intergiciel considéré est une couche de logiciel de sécurisation et d'encapsulation de COTS.

Le premier intergiciel réalisé est COMIT dont le rôle est de fournir une abstraction de l'exécutif COTS utilisé. Cet intergiciel spécifique repose sur les considérations suivantes :

- Modèle générique de COTS logiciel et matériel : OS (QNX6), carte unité centrale (CPU VME), E/S locales, E/S sortantes
- Modèle d'application : contrôle d'échéance, détection de défaillance, communication interprocessus, communication inter-CPU sur le même rack.

J.-C. Fabre: Quelle est la granularité du modèle. Classe ordonnancement ? Y-a-t il des fonctionnalités visibles à l'interface, ou bien s'agit-il de la fourniture d'un comportement interne transparent.

J.-D. Jouffrit: Non. Le modèle est concentré sur les messages. L'ordonnancement interne n'est pas apparent.

Un deuxième intergiciel, SECU assure la gestion des calculateurs répliqués. Il effectue la lecture et l'échange des données entre calculateurs, le traitement et la comparaison des résultats, l'éjection des calculateurs défaillants, et enfin les écritures en sortie. SECU effectue la synchronisation des logiciels à 500us près, l'échange de données entre voies, la consolidation par vote, le scellement / descellement des données échangées, etc.

Parmi les thèmes intéressant pour TA, nous avons identifié les suivants :

- Alternatives aux développements propriétaires : état de l'art, recensement de l'existant.
- Intergiciels COTS ou intergiciels en source libre
- Passerelles et interopérabilité en systèmes classés et non-classés.

Pour ce qui est de l'utilisation actuelle des intergiciels que nous avons développés, COMIT est utilisé sur le sous-marin nucléaire, LE VIGILANT, quant à SECU, il est actuellement en démonstration.

C. Honvault: Softworks ? Softkernel ? Exigence de la part de la DGA ? La DGA a imposé softworks sur les drones. Softworks est un noyau temps-réel objet.

B. Bauer: Ça sert à quoi d'avoir une architecture objet ?

C. Honvault: Configurabilité.

J.-C. Fabre: Grande finesse de configuration a priori.

C. Honvault: Dans la pratique, c'est un discours un peu marketing. Un composant peut en cacher 50.

J.-D. Jouffrit: QNX n'est pas orienté-objet mais il est configurable. On réalise l'intégration des composants dont on a besoin.

J.-C. Fabre: Ceci a en effet beaucoup d'intérêt pour le portage du noyau sur un nouveau processeur, peut-être plus que pour la spécialisation. Les classes à porter sont bien identifiées.

F. Taiani: Avez vous fait un effort vis-à-vis de la standardisation à la POSIX ?

J.-D. Jouffrit: On a effectivement regardé POSIX. C'est beaucoup trop riche ; beaucoup trop d'alternatives, de possibilités.

J. Arlat: Chaque application essaie d'identifier ses moyens de communication privilégiées avec un OS. Est-il possible d'exhiber un ensemble de services génériques ?

J.-D. Jouffrit: La recherche de la généricité peut entraîner le risque d'avoir trop de fonctionnalités inutiles.

B. Bauer: Pour nous c'est plutôt une approche générique vers laquelle on tend.

J.-C. Fabre: Lien avec la présentation sur A3M: dans une architecture en couches, il faut définir des services élémentaires dont on a vraiment besoin.

J.-D. Jouffrit: Il faut effectivement commencer par des services minimaux, de base, sinon on se retrouve avec une API trop large.

B. Bauer: Ce n'est pas évident. Ce que l'on a fait pour les militaires dépend de l'architecture choisie. Ce que l'on a défini dans un contexte pourra peut-être réutilisé dans un système similaire.

E. Marsden: Lorsque l'on effectue une analyse finale de l'architecture, le fait d'avoir le même intergiciel partout n'est-il pas un risque ?

J.-D. Jouffrit: Il y a effectivement le risque de défaillance de mode commun.

J.-C. Fabre: Est-ce une problématique qui vous semble majeure ?

J.-D. Jouffrit: Ceci fait partie des gros problèmes que l'on a, essentiellement sur les COTS. Certains COTS n'ont jamais été fait pour travailler ensemble. Comment maîtriser les comportements erratiques. On les découvre durant le développement / intégration.

B. Bauer: Le problème des défaillances de mode commun ont été prises en compte il y

a longtemps sur un Airbus, vis-à-vis de COTS faits par nous-même. Pour des COTS faits pour nous, il faut tester de façon approfondie et disposer des dossiers (conception, développement, tests) qui vont bien, c'est comme cela que l'on pourra aller plus loin.

C. Honvault: Quid d'une approche Boeing ? 3 compilateurs différents.

B. Bauer: On a ce genre de choses. Application écrite 4 fois.

J.-C. Fabre: TA applique une approche par encapsulation. Quand on utilise un COTS, finalement peu importe ce qu'il fait, pourvu qu'il réalise le minimum nécessaire, mais pas plus. Ce que l'on lui demande doit se faire sans perturber le reste du système.

J.-D. Jouffrit: Nos applications ne sont pas événementielles. Elles sont cycliques, bâties sur une boucle primaire unique. Tout est alors plus simple.

B. Bauer: Un OS n'est-il pas un peu riche pour une boucle ?

J.-D. Jouffrit: Si, bien sûr, mais on ne veut pas tout redévelopper: IP, la gestion des périphériques. Il s'agit en fait d'une boucle primaire plutôt applicative.

B. Bauer: Ce qui est important est hors de la partie ordonnanceur / synchronisation.

F. Taiani: Est-ce que ça sert d'avoir les sources ?

J.-D. Jouffrit: Nous sommes 12 personnes. Nous n'avons pas les moyens d'analyser des sources. Mais pour des cas de portage très spécifique, ça nous est arrivé d'acheter les sources.

O.-L. Rousseau: Même sans tout comprendre, il est utile de pouvoir ponctuellement regarder les sources.

C. Honvault: Nous avons acheté les sources de VxWorks. Cela nous a permis de comprendre le fonctionnement interne du noyau, de résoudre des bugs. Parfois, nous avons pu analyser un comportement incompréhensible extérieurement.

B. Bauer: Conclusion: Si on utilise des COTS, il faut avoir les sources.

J.-D. Jouffrit: Ca dépend de ce que l'on veut faire.

J.-C. Fabre: La vérité est au milieu. Elle n'est sûrement pas dans la boite noire, elle n'est pas non plus dans le détail extrême.

O.-L. Rousseau: Pour avoir le "au-milieu" il faut avoir les sources. Sans les sources on n'a rien.

J. Arlat: Le milieu qualifie le niveau d'investissement intellectuel. La façon de procéder dépend aussi de combien de systèmes sont développés par an.

O.-L. Rousseau: RTEMS est un noyau open source, mais il n'est pas extrêmement diffusé. Y-a-t'il des risques ?

B. Bauer: Tous les COTS ne sont pas forcément faits pour être mis ensemble. Des problèmes apparaissent du fait de l'abstraction.

7. Intervention de Philippe Chevalley (ESA)

Nos besoins couvrent les aspects plate-forme et charge utile. Nous avons aujourd'hui des modules spécialisés (mémoire, processeurs), communication (bus).

L'évolution tend vers des satellites de plus en plus intégrés, de plus en plus compacts, par exemple des mini-satellites avec beaucoup moins de mémoire, induisant un problème de stockage de données. Dans les solutions envisagées, par exemple dans des vols en formation, avec beaucoup plus de communication avec le sol, il y a un fort besoin d'architectures et de solutions distribuées.

A l'extrême, on pourrait imaginer des micro-satelllites en formation. L'intérêt est une facilité d'envoi, de meilleures performances par rapport à un seul gros satellite.

Nos besoins essentiels sont les suivants:

- Réutilisation de composants ;

- Maîtrise du déterminisme ;
- Confinement des erreurs (en particulier pour des applications de différentes criticités) ;
- Réduction des coûts : le développement se fait en deux phases, phase système et phase développement du logiciel. La phase système déborde souvent et c'est la phase logiciel qui doit rattraper le retard.
- Favoriser le transfert de technologies émergentes

Nos besoins à terme peuvent se résumer selon deux grand thèmes

- communication / coopération ;
- migration / reconfiguration.

En termes de solutions, nos travaux portent sur des couches intergicielle, telles que A3M ou RT-CORBA (*Real-Time CORBA*).

En termes de méthodologie, nous nous intéressons aux moyens d'exprimer facilement une architecture distribuée (*Avionics Architecture Description Language*, AADL), à la gestion de configuration, au prototypage et à la validation des systèmes.

Notre proposition actuelle s'articule autour d'une base de donnée de composants vérifiés auparavant. Nous les modélisons par AADL, tant en termes de propriétés que de performances.

J.-C. Fabre: Quelle est la granularité de la description ?

P. Chevalley: C'est d'assez haut niveau. Mais on peut exprimer des contraintes temporelles. Il y a un manque d'outils pour l'instant : Il n'existe que MetaH. C'est insuffisant.

J.-C. Fabre: Quel sorte de standard utilisez-vous?

P. Chevalley: Soit ISO, soit spatial.

Notre modèle de *framework* est orienté-objet. Dans une phase suivante, nous nous intéresserons à la génération automatique de code, dans ce *framework*, au-dessus d'un intergiciel.

F. Taiani: A propos de génération de code, ce qui est à l'origine c'est le modèle qui est un peu un programme puisque il permet de générer un exécutable. Quel en est le niveau d'abstraction ?

P. Chevalley: C'est tout le problème. Si le niveau d'abstraction est trop haut, il n'est pas possible de générer du code. S'il est trop bas, cela n'a plus d'intérêt.

Nous avons besoin d'un modèle de satellite pour valider l'architecture. L'activité de l'ESA porte donc sur les architectures. Les activités prévues porteront sur les logiciels distribués, la tolérance aux fautes, l'interopérabilité, le *plug-and-play* de composants, la re-programmation en vol, l'intelligence à bord, l'autonomie.

C. Honvault: Commentaire : Difficile de convaincre les donneurs d'ordre de mettre de l'autonomie dans les satellites. On travaille sur l'autonomie, mais peu de réalisations pratiques existent actuellement.

8. Intervention de Marc-Olivier Killijian (LAAS-CNRS)

Il existe une grande variété d'intergiciels, de COTS, avec un fort besoin en matière de sûreté de fonctionnement. Compte tenu des besoins d'adaptabilité actuels de la tolérance aux fautes, trois axes de recherche paraissent essentiels.

- Favoriser l'évolution des systèmes → composants
- Favoriser l'adaptabilité de la tolérance aux fautes → réflexivité
- Durcissement en renforçant le confinement d'erreur → empaquetage

La réflexivité est une approche conceptuelle qui distingue deux niveaux : le « Méta-niveau » contrôle le « niveau de base » par des opérations d'interception, d'introspection, d'intercession. L'intérêt

séparation possible des aspects fonctionnels et non-fonctionnels (*separation of concerns*).

Quels sont les besoins pour la tolérance aux fautes? Interception des communications, introspection de l'état des entités actives, intercession pour la restauration de l'état.

B. Bauer: Pouvez-vous donner des exemples de la réflexivité ?

J.-C. Fabre: Dans les années 60, la réflexivité a été inventée par les gens qui travaillaient sur les langages interprétés. Comment changer le comportement de mon programme sans modifier mon programme: en changeant l'interpréteur. Aujourd'hui il existe des compilateurs ouverts, c'est-à-dire adaptables.

M.-O. Killijian: Par exemple, on peut gérer la réplication à l'aide d'une architecture réflexive. La coordination entre répliques est gérée au méta-niveau. Dans une implémentation classique, les parties applicatives et la réplication sont intimement liées, ce qui ne facilité pas l'adaptation. Dans une réalisation réflexive, l'idée est de factoriser la gestion de la réplication au méta-niveau qui communique avec le niveau applicatif par une interface standard.

J.-C. Fabre: C'est une certaine forme d'encapsulation. La modélisation du fonctionnement du niveau de base se fait avec la granularité qui va bien.

Nous avons défini une architecture, baptisée FRIENDS, et nous avons réalisé un prototype sur *Orbacus*. Le méta-protocole (protocole d'interaction entre le niveau de base et le méta-niveau) a été réalisé avec un compilateur ouvert. Cependant, il existe un certain nombre de limitations: des informations qui sont nécessaires pour réaliser la réplication des objets ne sont pas disponibles au niveau langage. Il faut obtenir et agréger des informations à partir de plusieurs niveaux.

Aujourd'hui, des capacités réflexives sont introduites progressivement dans des COTS standards : CORBA / Java. Nous avons récemment réalisé un protoype avec des mécanismes réflexifs élémentaire fournis par Java (sérialisation de l'état – *JVM Reflective API*) et CORBA (*Portable Interceptors*). La réplication (passive) se fait de manière transparente du point de vue de la partie applicative.

Nous pouvons nous servir de cette même approche pour mettre en œuvre l'empaquetage, par l'ajout de *wrappers* au niveau système dans le but de se prémunir contre des comportements erratiques.

B. Bauer: Mais que ce se passe-t-il si quelque chose que l'on a pas pensé à vérifier se produit?

M.-O. Killijian: C'est un problème. Plus l'interface est réduite, plus il est facile de l'empaqueter.

B. Bauer: Nous n'avons pas de wrappers, mais notre technologie de développement est certifiée. Nous avons de l'outillage pour s'assurer que le code qu'on a développé a été correctement validé. Pourquoi pas des COTS certifiés DO 178 B ?

Réponse collective : Oui mais est-il raisonnable de penser avoir des COTS certifiés et pas chers ?

Dans nos expériences nous avons utilisé *Linux* qui n'est pas particulièrement réflexif. Il a donc été nécessaire de sortir de la boite *Linux* les informations dont on avait besoin pour programmer nos *wrappers*.

A l'avenir, nous prévoyons de passer à des architectures à composants avec modules beaucoup plus petits. Notre technologie de base sera celle des composants réflexifs. Le but est de permettre une adaptation générique qui passe par le méta-modèle de chaque composant. Par exemple, dans une architecture multi-couches, il y a un besoin d'adaptation dans chaque couche. Ce n'est pas gérable sur le long terme dans les grands projets. Il y a donc un impérieux besoin de séparation / découplage entre plate-forme et mécanisme. La couche de découplage, c'est le méta-modèle qui intègre les différents niveaux d'abstraction.

En guise de conclusion, j'identifierais trois axes de recherche qui portent sur:

- les *framework* architecturaux
- la validation des COTS
- l'ingénierie de tolérance aux fautes

Il y aura bien entendu des interactions entre ces 3 axes et des choix technologiques à faire.

B. Bauer: On ne savait pas que ça s'appelait comme ça, mais dès que l'on programmait un mini OS, on intégrait des capacités de traçage. C'est proche de la réflexivité.

J.-C. Fabre: Oui mais dans votre approche il n'y a pas de modifications possibles.

O.-L. Rousseau: Mais est-ce que ça vaut le coup de mettre une "verrue" réflexive plutôt que de corriger le bug ?

J.-C. Fabre: Dans nos expériences nous avons eu le cas. A partir des informations dans les *logs* nous avons pu effectivement corriger des fautes du logiciel. Quand le comportement erroné est dû à des fautes accidentelles physiques, ou des conditions environnementale particulières, la notion d'empaquetage prend tout son sens. Pour ce qui est de la gestion de la redondance, c'est encore une autre histoire, mais nous avons montré au LAAS le grand intérêt de cette technologie pour l'adaptabilité.

9. Discussion générale

La discussion générale a porté essentiellement sur les problèmes ouverts qui demanderaient un effort commun de recherche et développement, ce qui nous a amené à un tour de table pour recueillir le sentiment de chacun.

B. Bauer: Ce qui nous intéresse, c'est le côté systèmes embarqués par des extensions du logiciel de base. Il faut noter une grande révolution dans l'avionique: la séparation entre les développeurs d'API et les développeurs de plates-formes. Nos objectifs visent à pouvoir porter les applications à moindre coût. Le problème est celui de la stabilité des applications dans le temps. Notre gros problème est celui des COTS à cause d'une obsolescence très rapide. Nous avons aussi des besoins en matière de reconfiguration, par exemple la migration de module d'un processeur à l'autre dans des architectures multi-modules / multi-processeurs. Il s'agit de gérer un ensemble d'applications sur un ensemble de processeurs. En termes de norme pour le logiciel de base (OS), nous reposons sur ARINC 653. Cette API facilite les problèmes de portabilité. Dans un contexte plus large qu'un seul processeur, les besoin de services sont plus larges, à savoir les services classiques dans un intergiciel. Les logiciels de base doivent pouvoir communiquer entre eux, et ce de manière la plus transparente possible pour l'application. De plus, la notion de *firewall* dans un système critique, nécessite un haut niveau de certification, qui sera de facto à un niveau plus faible que l'OS qui est le plus critique. Ceci induit le problème de l'insertion de niveaux de certification hétérogène dans un système.

O.-L. Rousseau: Nos problèmes tournent autour de *l'Integrated Modular Avionics.* De plus, les performances de l'application dépendent du processeur. Si on change de processeur, il va y avoir un impact. Le problème est donc de savoir comment avoir une synchronisation forte dans un système multi-module / multi-processeurs. Il y a aussi un besoin de gestion du cycle de vie des applications: chargement, synchronisation, etc. La gestion des fautes au niveau d'un module ou d'un ensemble de modules est aussi un point crucial. Ces aspects ne sont pas présents dans l'ARINC 653, où l'on adresse uniquement le point de vue applicatif. Nous avons aussi des besoins qui sont de niveau système.

J.-C. Fabre: Mon impression est que les aspects isolation sont très importants. Pour Airbus la connexion avec le monde ouvert, pour TA les comportements erratiques des OS, pour Thalès la cohabitation de niveaux d'intégrité hétérogènes.

B. Bauer: Malléabilité du lien application / module matériel est aussi un souci. Par rapport à A3M, nous n'avons pas le problème de multiplier les processeurs pour des problèmes de puissance. Nous avons en général suffisamment de puissance.

J. Arlat: Deux applications sur un même processeur ne pose-t-il problème pour le confinement d'erreurs. De même, n'y a-t-il pas de nouveaux problèmes lorsque les modules sont grandement répartis (cf. satellite : sol / bord). Les modèles de fautes sont différents quand il n'y a pas de co-localisation.

O.-L. Rousseau: La co-localisation n'est pas utilisée pour faire de la communication plus sûre.

J.-C. Fabre: Dans le projet A3M, étant donné la banalisation des processeurs COTS, le but est de rendre possible l'utilisation de processeurs COTS sans mettre en péril les propriétés de sûreté de fonctionnement. Il faut permettre l'évolution, le déplacement d'applications d'un module à un autre.

O.-L. Rousseau: Dans IMA c'est aussi le but. A partir du moment où on normalise l'OS, on peut déplacer les applications. Les seules contraintes sont toujours de garantir les contraintes temporelles. Au fur et à mesure du développement, on voit apparaître les nécessités de reconfiguration des applications dans les modules IMA, ne serait-ce que pour mieux prendre en compte les besoins de chaque module.

J.-C. Fabre: Comme nous l'avons vu dans la présentation d'Airbus, la notion de *firewall* peut être très intéressante pour introduire du confinement sans rendre les choses complètement étanches, comme elles tendent à l'être dans l'ARINC 653.

Compte tenu de l'objectif du groupe de travail, je pense qu'un certain nombre de pistes ont été identifiées. Nous avons à faire un travail d'intégration et de factorisation. Le résultat de ce travail devrait être soumis au groupe d'ici à la prochaine réunion pour avancer sur la formalisation de ces problématiques, sous la forme d'un ensemble de sujets énoncés au cours de la journée et qui sont de niveau intergiciel. Il faut les organiser, les structurer.

J. Arlat: J'ai surtout entendu parler d'intergiciel *ad-h*oc pour traiter le problème des COTS générique.

J.-C. Fabre: Oui, mais on ne peut pas ignorer en parallèle le retour d'expérience sur l'utilisation d'intergiciel génériques (COTS, open source) et des limites rencontrés.

J.-D. Jouffrit: Mais est-ce que l'on ne risque pas de diverger entre les gens de l'embarqué (plus intéressés par les intergiciels *ad-hoc*) et les gens des systèmes sol (plus intéressé par des intergiciels génériques, tels que CORBA).

J.-C. Fabre: L'exercice est difficile, mais il existe des problématiques communes. Sur le plan algorithmique de base, par exemple, il y a des problématiques identiques (réinsertion de processus, par exemple) mais ensuite de toute évidence leur instanciation doit se faire dans deux contextes : embarqué / grands systèmes. Pour certaines problématiques, comme l'encapsulation, le confinement, la technologie est la même, les moyens techniques sont identiques, même si, encore une fois, l'instantiation est différente selon le contexte.

Il faudra aussi trouver une approche marketing à notre affaire. Décrocher un projet, c'est le vendre. Il faut architecturer notre proposition au-delà des problématiques techniques. Il est très difficile aujourd'hui de trouver des supports de financement pour de la recherche un peu fondamentale. Pas besoin d'y répondre aujourd'hui, mais nous aurons à y répondre très bientôt.

10. Clôture de la réunion.

Notre prochaine réunion devra statuer sur le devenir de notre action avec un objectif « projet ». Tous les aspects des partenaires n'ont pas été présentés, Thalès et grands systèmes, par exemple, mais l'avaient été lors de notre atelier sur ce sujet en juin 2002. Ce sont certainement des sujets à ne pas oublier dans la préparation de notre prochaine réunion.

Les commentaires ainsi que des avis sur l'organisation de notre travail futur, les thématiques communes, sont évidemment les bienvenus pour préparer notre prochaine réunion qui sera certainement déterminante pour la poursuite de cette réflexion et des travaux dans un projet commun.

Jean-Charles Fabre remercie l'ensemble des participants à cette première réunion du groupe de travail « Intergiciel et Sûreté de Fonctionnement ».

La réunion se termine à 16h45.

11. Participants

AIRBUS

Olivier-Laurent Rousseau

Marie-Line Valentin

ASTRIUM

Christophe Honvault

Patrick Pleczon

LAAS-CNRS

JeanArlat

Jean-Charles Fabre

Marc-Olivier Killijian

Eric Marsden

François Taiani

TA

Jean-Denis Jouffrit

THALES

Brigitte Bauer

Invité : European Space Agency (ESA)

Philippe Chevalley