Réseau d'Ingénierie de la Sûreté de fonctionnement

COMPTE RENDU

de l'atelier thématique du 6/06/2002

« Intergiciel et sûreté de fonctionnement »

1 Objectifs

L'atelier thématique du *RIS* portant sur le thème « Intergiciel et Sûreté de Fonctionnement a eu lieu au LAAS-CNRS le jeudi 6 Juin 2002. Cette journée a réunit les partenaires du *RIS* (Airbus, Astrium, LAAS-CNRS, Thales, Technicatome) ainsi que des participants extérieurs au *RIS* (INRIA, ESA, France Telecom, SNCF) qui ont contribué au succès de cet atelier en apportant des éclairages complémentaires sur différents plans, tant théoriques que relatifs à leur expérience dans d'autres domaines d'application.

La journée était organisée autour de présentations qui couvraient un large spectre de techniques et d'expériences sur l'utilisation, le développement d'intergiciel dans des systèmes pour lesquels la sûreté de fonctionnement est une caractéristiques non-fonctionnelle majeure. Trois sessions ont illustré les aspects suivants (voir programme §3):

- Utilisation et développement d'intergiciels spécifiques pour applications embarquées critiques ;
- Intergiciels dans les systèmes télécom et développement en source libre d'intergiciels ;
- Caractérisation et utilisation d'intergiciel CORBA dans des grands systèmes

La dernière session de la journée a permis à d'autres participants de donner leur point de vue, de relater leur expérience mais aussi d'identifier leurs problématiques particulières dans leur domaine, notamment le ferroviaire ou pour les grands programmes spatiaux. Les suites envisageables et la réponse que le *RIS* peut proposer pour analyser plus en détails cette problématique ont ensuite fait l'objet d'une discussion animée.

Le présent rapport à pour objectif de faire la synthèse des présentations qui ont eu lieu lors de cette journée ainsi que des discussions entre les participants.

La copie des transparents présentés est donnée en annexe.

2 Participants

Membres du *RIS* : Marie-Line Valentin (*Airbus France*) ; Jean-Paul Blanquart, Philippe David, Gilbert Griseri, Luc Planche, Philippe Rouzet, Eric Totel (*Astrium*) ; Jean Arlat, Jean-Charles Fabre, Marc-Olivier Killijian, Eric Marsden, David Powell, François Taiani (*LAAS-CNRS*) ; Jean Denis Jouffrit (*Technicatome*) ; Brigitte Bauer, Pascal Foix, Virginie Watine (*Thales*)

Représentants d'organismes invités : Eric Bornschlegl et Jean-Loup Terraillon (*Agence Spatiale Européenne*), Pierre Crégut (*France Telecom R&D*), Jean-Bernard Stefani (*INRIA Rhone-Alpes*), Gérard Le Lann (*INRIA Rocquencourt*) et Elias Martins (*SNCF*).

Les coordonnées des participants sont données en annexe.

3 Programme

Introduction	**Jean Arlat (LAAS-CNRS)**
Présentation de l'atelier	**Jean-Charles Fabre (LAAS-CNRS)**
Middleware embarqué pour la gestion des données de Columbus (ISS)	**Eric Totel (Astrium)**
Intergiciels et coordination distribuée robuste dans les systèmes temps-réel	**Gérard Le Lann (INRIA Rocquencourt)**
COMIT : une bibliothèquede composants logiciels sécurisée pour applications relatives à la sécurité	**Jean-Denis Jouffrit (Technicatome)**
Intergiciel : Abstraction et/ou adaptation	**Pierre Crégut (France Telecom R & D)**
ObjectWeb: Une initiative -source libre- pour le développement d'intergiciels : une introduction	**Jean-Bernard Stéfani (INRIA Rhone-Alpes)**
Sûreté de fonctionnement d'intergiciel CORBA: caractérisation de services par injection de fautes	**Eric Marsden (LAAS-CNRS)**
Point sur l'utilisation des technologies middleware dans les centres de contrôle et de mission des systèmes spatiaux	**Philippe Rouzet (Astrium)**
Middleware et tolérance aux fautes dans les futurs systèmes de supervision et de contrôle de Thales	**Virginie Watine (THALES)**
Autres interventions – Discussion – Débat – Actions futures	

4 Synthèse des présentations

4.1 Introduction (Jean Arlat, LAAS-CNRS)

Le thème « Intergiciel et Sûreté de Fonctionnement » de ce troisième atelier du $\mathcal{RIS}$ a été identifié par le comité directeur comme un sujet majeur suite aux suggestions des membres du réseau. Il fait suite à deux ateliers portant sur les thèmes « Logiciel libre et la Sûreté de Fonctionnement » et sur « Usage et perspectives pour la production de logiciels sûrs ». Le premier de ces ateliers a donné lieu à un groupe de travail dont les travaux doivent se terminer au dernier trimestre 2002. Jean Arlat rappelle aux participants à cette journée les origines du $\mathcal{RIS}$, qui fait suite au $\mathcal{LIS}$[1], ses motivations, son organisation et ses objectifs. Tout particulièrement, un atelier a pour objectif de défricher un thème, d'évaluer la pertinence d'un travail plus approfondi. Un groupe de Travail, aussi ouvert à d'autres partenaires, permet de mener ce travail de fond et peut ouvrir d'autres perspectives de partenariat. La décision sur l'opportunité de créer un groupe de travail est du ressort du comité directeur après avis des participants au premier atelier. En pratique, le $\mathcal{RIS}$ dispose d'un site web qui offre un résumé des activités menées au sein du réseau.

4.2 Présentation de l'atelier (Jean-Charles Fabre, LAAS-CNRS)

Après une présentation des participants à l'atelier, Jean-Charles Fabre aborde la problématique « Intergiciel et Sûreté de fonctionnement » ainsi que l'organisation de la journée. La notion d'intergiciel est large. Sa définition la plus largement accepté est celle d'une collection de logiciels qui résident entre le niveau exécutif (systèmes opératoire, micronoyau temps-réel) et le niveau applicatif propre à un domaine particulier. Cette couche de logiciel peut avoir différents objectifs et prendre des formes très diverses. Très globalement, on peut distinguer les intergiciels spécifiques des intergiciels génériques. Les intergiciels spécifiques ont pour objectifs de fournir des services tant fonctionnels, propres à un domaine applicatif donnée (cf. §4.3, §4.5), que non-fonctionnels, dans le domaine de la tolérance aux fautes et du temps-réel, par exemple (cf §4.4). Ce type d'intergiciel s'oppose à la notion plus classique d'intergiciel générique, couche de logiciel dont le but est plutôt d'offrir en particulier des services de répartition transparente des applications. CORBA ou de Java/RMI sont les exemples les plus typiques de cette seconde famille d'intergiciels. L'utilisation des intergiciels génériques voit plutôt son terrain de prédilection au sein des grands systèmes de supervision ou de contrôle (cf. §4.9, §4.10).

Qu'ils soient spécifiques ou génériques, la caractérisation de leurs modes de défaillance est certainement un aspect majeur de leur sélection et de leur utilisation dans des domaines d'application critiques (cf. §4.8). La perception et les besoins en terme d'intergiciel sont bien entendu variables d'un domaine d'application à l'autre, le domaine des telecom couvrant certainement une grande variété de types de systèmes (cf. §4.6). Il va de soi aussi que la réalisation

[1] *Laboratoire d'Ingéniérie de la Sûreté de fonctionnement*, crée en 1992 par le LAAS, Matra Marconi Space (aujourd'hui Astrium) et Technicatome. En 1996 le LIS renouvelé a accueilli trois nouveaux partenaires, Aérospatiale Airbus (actuellement Airbus France), Electricité de France et Thomson CSF (actuellement Thales).

d'intergiciels est un problème en soit, dans la mesure où les différents domaines applicatifs nécessitent souvent pour ne pas dire toujours des intergiciels taillés à façon, même s'ils sont conformes à des standards, ou que leur source est libre (cf. 4.7).

Différentes dimensions sont donc à considérer sur un sujet aussi vaste. Les présentations de la journée donnent une bonne illustration de différents aspects de cette problématique. La richesse et la variété des contributions en termes conceptuels et théoriques, en terme expérimental et de retour d'expérience, du point de vue des domaines applicatifs représentés (avionique, spatial, telecom, ferroviaire, grand public) laissent espérer une première analyse fructueuse. Place aux présentations.

4.3 Middleware embarqué pour la gestion des données de Columbus (ISS)

Intervention (Eric Totel, Astrium) – Cf. Transparents en annexe B1

Columbus Orbital Facility (COF) est un intergiciel pour applications embarqués dans l'ISS (*International Space Station*). Il s'agit d'une implémentation réelle dont les objectifs sont l'exécution des applications spécifiques aux expériences à bord ainsi que la gestion et le stockage des résultats. L'architecture est volontairement distribuée pour accroître la puissance de calcul, ce qui présente aussi un intérêt du point de vue de la tolérance aux fautes. En pratique le système comprend trois nœuds actifs et un nœud de réserve. Il s'appuie sur un serveur de fichier type NFS implanté de façon redondante (réplication semi-active). Des portables sur lesquels sont réalisés certaines expériences peuvent se connecter sur le système pour profiter de ses fonctionnalités. Le système comprend une partie "nominale" et une partie "vitale" qui "surveille" et reconfigure la partie nominale, et ses différentes fonctionnalités interagissent au travers de connections Internet et type bus 1553.

Du point de vue applicatif, l'architecture revêt un caractère banalisé au sens où les applications sont exécutables sur tous les nœuds et communiquent entre elles. Cette caractéristique à une influence sur le support d'exécution. L'ensemble des applications est supporté par le *Data Management System Software* (DMS SW). Le DMS SW, au cœur de la plateforme logicielle, s'instancie selon les fonctionnalités requises par le nœud sur lequel il repose. Le DMS (intergiciel + pilote d'interfaces), fourni un support sûr d'exécution pour des applications Ada et FLAP sur machine virtuelle. Il assure la gestion du cycle de vie des applications à distance ainsi que la distribution des données et des traitements. Sur chaque nœud, il offre une interface de contrôle et de gestion de messages (RPC, messages asynchrones, CCSDS -- protocole propre au spatial). Il permet aussi d'effectuer la reconfiguration du système. L'intergiciel en lui-même se compose de deux parties, l'une active, l'autre passive :

- Partie active : acquisition cyclique, surveillance, réalisées par échange de messages, gestion des applications et les redondances (notamment le système de fichiers).
- Partie passive : bibliothèques offrant différents types de services.

On peut distinguer trois grandes familles de composants, soit dédiés aux données (acquisition et distribution à bord), soit dédiés à la sûreté de fonctionnement (surveillance, reconfiguration …), soit, enfin, relatives à la gestion des applicatifs et des messages associés (machine virtuelle). Notons au passage que cet intergiciel est très spécifique au spatial et inclut toutes les fonctions nécessaires entre le système temps-réel de base et les tâches d'application.

En ce qui concerne les communications, il existe deux types de liens : Internet pour l'échange de messages RPC et autres messages liés à la distribution des applications, et le lien 1553 pour l'acquisition de données sur les différents équipements. Il existe donc, plusieurs types de messages, dont les messages type CCSDS empaquetés dans des messages logiciels et véhiculés par le protocole UDP .

En ce qui concerne la gestion des données, il faut distinguer les données locales de la distribution de celles-ci dans l'architecture. L'acquisition de données locales est effectuée de manière cyclique sur les nœuds, mais aussi des informations sur le système font aussi partie des données locales. Ces données sont soit utilisées uniquement en local, soit distribuées. La distribution est réalisée cycliquement par *multicast* selon un protocole périodique, une fenêtre temporelle étant allouée pour la transmission évitant de ce fait les possibles collisions. La détection d'erreur est aussi assurée au niveau de l'intergiciel de surveillance du *Data Pool*, par exemple). Si une anomalie est détectée, la reconfiguration du système est déclenchée. Des fonctionnalités de tolérance aux fautes sont aussi présentes pour tolérer des erreurs intermittentes.

Après détection d'erreur (nœud défaillant), la reconfiguration consiste essentiellement à un basculement sur un des nœuds redondants. Très schématiquement, deux nœuds sont en permanence surveillés par le 3ème, le 3ème surveillé quant à lui par la couche vitale. Le recouvrement s'effectue à partir points de reprise (*checkpointing*) obtenus au niveau de la couche intergiciel. Les applications ne sont pas reprises dans un état précédemment sauvegardé, mais simplement redémarrées.

Le serveur de fichier s'exécute en réplication semi-active, à savoir que lors d'une écriture sur le disque par le primaire (*leader*), le suiveur (*follower*) traite les données en parallèle. De plus, une surveillance du maître par l'esclave est effectuée en permanence ; lors d'une détection de défaillance (*timeout*) le principe de basculement conduit au remplacement du maître sur le bus.

Conclusion : Ces travaux constituent une étape importante et novatrice pour la mise en place d'infrastructures distribuées tolérant les fautes dans le spatial embarqué et ce, selon un schéma générique.

Questions/Commentaires :

- Gérard Le Lann : Quel est le degré de couverture de l'hypothèse de silence sur défaillance ?
- Eric Totel : Il est très difficile de répondre car nous n'avons pas mené de campagne d'injection de fautes. La synchronisation est faite au plus bas (après le cache mémoire). Si le maître ne répond plus, il y a peu de chance qu'il agisse encore.
- JP Blanquart : Le système fonctionne si l'hypothèse de silence sur défaillance est observée, et continue à marcher à peu près si ne n'est pas complètement le cas. Observons que il y a une possibilité de perdre des données lors de la distribution de la *Data Pool*. Mais ceci reste temporaire puisque l'acquisition de données est cyclique.
- J.-P. Auclair : Pourquoi l'option intergiciel est-elle intéressante. Quels sont les inconvénients et les avantages par rapport aux solutions du passé ?
- Eric Totel : Derrière le mot intergiciel se cachent beaucoup de choses. Ces choses existaient déjà, mais étaient appelées différemment.
- Philippe David : Idéalement, les notions de spécifications, de techniques de reprise, de distribution et d'asynchronisme, devraient coller aux résultats de la recherche. Ce n'était absolument pas ce que l'on faisait dans le spatial.
- Jean Charles Fabre : Les fonctionnalités offertes sont indépendantes des applications ; par exemple, le typage de messages est du ressort de l'intergiciel, ou encore la personnalisation des styles d'interaction. L'intergiciel fait la transition entre le système d'exploitation générique qui est en dessous et les attentes des couches supérieures.
- Pascal Foix : Quelles sont vos perspectives dans les 5 ans qui viennent pour continuer le développement ?
- Luc Planche : Colombus est très spécifique, et normalement développé pour des plate-formes satellites. Une station spatiale est un monde différent par rapport aux satellites. Une étude est en cours avec l'ESA (A3M) (*Advanced Avionics Architectures and Modules*). Nous en sommes à la 2ème phase qui implique l'INRIA (Gérard Le Lann) et le LAAS (Jean-Charles Fabre). Elle s'appuie sur un transfert de technologie algorithmique de l'INRIA dans le domaine spatial.
- Eric Bornschlegl : L'ESA à une volonté de standardisation des plate-formes : interopérabilité entre différents contractants et l'utilisation de composants COTS sont des soucis majeurs. Un des objectifs de l'étude est le passage d'un système très spécifique à un système standard. Il faut aussi favoriser l'extensibilité (*scalability*). Astrium a déjà de l'expérience avec Columbus. Le LAAS et l'INRIA sont déjà impliqués dans le projet. Notre désir est de voir une convergence entre des résultats de recherche et notre soucis de standardisation.

4.4 Intergiciels et coordination distribuée robuste dans les systèmes temps-réel

Intervention (Gérard Le Lann, INRIA) – Cf. Transparents en annexe B3

Les services pour la Sûreté de Fonctionnement dans un environnement distribué sont génériques et orthogonaux aux applications. Ils sont factorisés dans des composants et libèrent les programmeurs métier de ces préoccupations. A titre d'exemple, citons des services génériques comme: consensus, diffusion atomique, élection d'un *leader*, reconfiguration, … Les problèmes sous-jacents sont redoutables : il existe certains résultats d'impossibilité.

En dépit de l'accumulation de résultats depuis de nombreuses années, la rencontre entre les résultats de la recherche et les pratiques industrielles n'a pas été faite. Certains produits qui ne tenaient pas leurs promesses ont donné lieu à des échecs : Trafic Aérien aux USA, Bourse de Zürich.

Il y a certainement un fort besoin de coopération recherche-industrie face à des problèmes récurrents. A titre d'exemple, connaître le temps de détection de vraies défaillances borné calculable (pire cas ou moyenne) ou le temps d'obtention

d'un service pose des problèmes difficiles restés ouverts jusqu'à récemment. L'exemple classique est celui de la détection de défaillance par temporisations (*time-out*) : avec une temporisation courte, la détection est plus rapide, mais il y a beaucoup de fausses détections pouvant conduire à un suicide collectif. Si l'on accroît la temporisation, mais on aura moins de chances de tuer les "bons" processeurs, mais plutôt ceux qui sont très lents, et donc la latence de détection des vraies défaillances s'accroît d'autant. Les problèmes sous-jacents sont en fait bien connus et relatifs à l'ordonnancement.

A l'INRIA, deux études conduites par le projet REFLECS se sont focalisées sur cette problématique: ORECA (DGA), ATR (DGA, MENRT, CNRS). Elles ont apporté notamment des solutions nouvelles pour les problèmes de consensus, coordination, réplication (CDR).

Parmi les propriétés requises, telles que sûreté (au sens logique, c'est-à-dire ne pas rentrer dans des états dangereux), vivacité, confiance, ponctualité, c'est la sûreté qu'il importe de garantir à tout prix. Quand il y a défaillance, puis basculement sur une configuration de " survie " en attente d'une intervention extérieure, il est alors indispensable de démontrer que cette configuration est sûre.

Dans le principe, moins on fait d'hypothèses plus le taux de couverture de cette propriété sera élevé. Il existe de nombreux modèles de calcul ; les deux plus extrêmes sont le modèle synchrone (temps de calculs, de communication connus) et le modèle asynchrone (où la connaissance de borne temporelle ne peut être retenue comme hypothèse de conception). Le modèle asynchrone a donc le taux de couverture le plus élevé qui soit, mais, pour certains problèmes de type CDR, il implique des résultats d'impossibilité. Il est donc nécessaire de s'éloigner du modèle asynchrone, mais pas trop pour maximiser la couverture. Le choix des hypothèses n'est alors pas innocent.

Le question classique est celle-ci: Cela a-t-il un sens de considérer le modèle asynchrone dans un contexte temps-réel ?

Clairement, dire « asynchrone et temps-réel, cela ne marche pas ou est inefficace » est un préjugé. En fait, une solution asynchrone (moins demandeuse en hypothèses fortes) peut être implémentée dans système conforme au modèle synchrone, dans lequel il existe la possibilité de calculer / dimensionner les temps de réponse. Dans le modèle synchrone, la notion de temps est une entrée initiale du processus de conception des mécanismes de tolérance aux fautes. Dans une solution asynchrone, cette entrée n'est pas considérée lors de la conception des mécanismes mais uniquement lors de leur instanciation dans un système concret (concept de late *binding*).

Concernant l'inefficacité des solutions asynchrones, nous pouvons affirmer que, pour les mêmes hypothèses temporelles des éléments d'un système, et pour les problèmes de type CDR, les solutions synchrones sont plus mauvaises au pire cas que les solutions asynchrones. Différents algorithmes ont été développés à l'INRIA : Consensus Uniforme Rapide (FastUCS), Coordination Uniforme Rapide (FastUCN). Un prototype a été réalisé et évalué par des agences spatiales et des industriels pour des utilisations réelles.

Si l'on prend le problème du *Consensus Uniforme*, la question qui se pose est la suivante : *quand les processeurs proposent en même temps des décisions différentes, est-il possible de converger vers une décision unique, la même pour tous (processeurs défaillants et non-défaillants) ?*

C'est effectivement compliqué à cause des temps de communication variables et des défaillances. La présentation du *Consensus Uniforme* montre que des solutions prouvées correctes et efficaces peuvent être obtenues (cf. transparents en annexe B3). Les performances comparatives selon des deux modèles accréditent le fait que la solution FastUCS, bien que reposant sur un modèle asynchrone, est efficace.

Si l'on prend le problème de la *Coordination Uniforme*, la question qui se pose est la suivante : *quand les processeurs proposent en même temps des contributions à un objectif global, est-il possible de converger vers une agrégation unique de ces contributions, la même pour tous (processeurs défaillants et non-défaillants) ?*

Un exemple typique est celui d'une proposition initiale correspondant à des résultats de calcul partiel. L'agrégation des résultats locaux permet d'obtenir un résultat global. Comme des défaillances sont possibles, et que les délais sont variables, les processeurs peuvent « voir » des agrégations différentes. L'une des applications les plus classiques est l'ordonnancement distribué en présence de files d'attente. L'INRIA a développé l'algorithme FastUCN (*Fast Uniform Coordination*) construit à partir de FastUCS (*Fast Uniform Consensus*).

En conclusion, il est à noter que, très souvent, les solutions synchrones excluent les phénomènes de files d'attente, ce qui les met hors jeu pour la plupart des problèmes réels. Beaucoup de travail reste à faire à l'intersection des disciplines des algorithmes distribués, de la tolérance aux fautes et de l'ordonnancement en-ligne.

Questions/Commentaires :

- JL. Terraillon : Pourquoi a-t-on besoin du modèle synchrone pour la phase d'instantiation des mécanismes CDR?

- G. Le Lann : On a besoin de « passer par » un modèle synchrone seulement si l'on doit prouver le « temps réel » (la ponctualité). Pour ce faire, on doit établir un système de contraintes (les conditions de faisabilité), ce qui impose d'exprimer la stratégie de l'adversaire "la pire'' du point de vue des temps d'exécution. L'adversaire génère des pics de « charge » et les défaillances à des instants « bien choisis », tout en respectant les hypothèses de temps de communication et d'exécution. Pour pouvoir raisonner sur ces instants « bien choisis », on a besoin d'une base de temps.

- JL. Terraillon : Le pire cas ne correspond-il pas à la situation dans laquelle tous les événements arrivent en même temps ?

- G. Le Lann : C'est souvent le cas (surtout dans les systèmes centralisés), mais pas toujours (il existe des contre-exemples avec les systèmes distribués, cf. l'étude ORECA)..

- E. Bornschlegel: N'y a-t-il pas une difficulté à formaliser les hypothèses de départ : Qu'est-ce qui est synchrone, qu'est-ce qui ne l'est pas ?

- G. Le Lann : Il existe une réelle difficulté, celle de prédire l'avenir. Le choix d'un modèle peut être du ressort du donneur d'ordre, ou bien de l'industriel concepteur. Les responsabilités (vis-à-vis de ces choix) sont clairement établies dès le début d'un projet, ce qui a le mérite de la clarification (on sait que les non-dits et le flou dans les documentations manipulées par les donneurs d'ordre et les industriels sont les principales causes d'échecs). Quand on veut se « couvrir », en présence d'incertitude, il est évidemment recommandé de choisir des modèles proches du modèle asynchrone (cf. les taux de couverture). On a vu que résoudre les problèmes de type CDR n'est pas plus difficile que l'on soit en synchrone ou en asynchrone**.**

4.5 COMIT : une bibliothèquede composants logiciels sécurisée pour applications relatives à la sécurité

Intervention (Jean-Denis Jouffrit, Technicatome) – Cf. Transparents en annexe B4

Jusqu'aux années 90, le contrôle-commande des chaufferies nucléaires pour les sous-marins était réalisé avec des techniques analogiques. Depuis cette époque, les premières solutions consistaient dans l'utilisation de matériels et logiciels spécifiques. Comme les programmes s'étalent sur une longue durée, les problèmes d'obsolescence, notamment du matériel, sont vite devenus très sensibles. L'approche suivie par la suite a consisté à développer des solutions à base de composants COTS, en particulier au niveau des exécutifs de base. La question est alors de savoir comment garantir la pérennité des applications et leur portage. L'idée est alors de découpler l'application de son support d'exécution, c'est-à-dire du système d'exploitation. Cette couche d'abstraction des fonctionnalités de l'exécutif correspond à notre intergiciel.

En termes d'applications visées, il s'agit du contrôle-commande d'une chaufferie nucléaire : supervision, régulation, commande des actionneurs. Ce type d'application embarquée est de niveau B (*safety related*). Ce sont des applications cycliques (non événementielles) qui doivent fonctionner 24h sur 24 sans intervention. La pérennité est en général supérieure à 10 ans.

Les objectifs de la bibliothèque sont de fournir une indépendance totale par rapport aux COTS matériels et logiciels et une sécurisation des interactions avec l'application sous la forme d'un contrôle des paramètres d'appel au COTS, mais aussi de restriction à l'utilisation de ces services de base.

Pour des raisons de réutilisation et de factorisation, nous avons défini un modèle uniforme de développement pour éviter des développements redondants. Cette bibliothèque se compose de deux parties, une générique, l'autre spécifique au COTS. Vis-à-vis des applications, l'interface se compose de classes de base génériques, dont l'implémentation concrète s'effectue pour une cible.

Cette bibliothèque s'appuie sur une modélisation sous forme de classes ; les fonctionnalités disponibles sont : gestion CPU, I/O locales tout ou rien ou analogiques, I/O distantes, système graphique, réseau de terrain déterministe. Le but est d'avoir des composants réutilisables et instanciables pour des besoins tels que la gestion des ressources matérielles (découplage entre noms physiques et noms logiques), le contrôle d'échéance de processus (notion de "*heart beats*" pour la détection d'erreurs), la gestion des défauts, la mise dans un état sûr. Notre approche s'appuie sur une programmation défensive : si un paramètre n'est pas correct lors d'appels système, il y a alors mise en « arrêt du système ». Enfin, la communication inter-processus est synchrone au moyen de messages sécurisés.

La construction de cette bibliothèque s'appuie sur un modèle d'application (toute nouvelle application s'y conforme) et sur la notion d'instance. Une configuration est exactement définie à partir des besoins (pas de code mort). Il y a ensuite un intégration qui est faite avec une application de test représentative de l'application définitive.

En terme d'OS, il s'agit du µnoyau QNX6. Le système graphique associé est PHOTON. Le réseau de terrain utilisé FIP.

La démarche développement suit une approche niveau B. Le codage est quant à lui effectué en C++ avec de nombreuses restrictions. L'allocation dynamique est uniquement acceptée lors du démarrage du système et ensuite elle est interdite. Les tests unitaires ont une couverture de 100 % du code atteignable.

Pour ce qui concerne les temps de calcul, le peu de code à dérouler et une grosse puissance de calcul nous offre beaucoup de marge, ce qui facilite l'assurance des hypothèses synchrones. Il y a réplication sur 3 calculateurs avec comparaison des résultats à chaque étape. Dans ce type d'application nucléaire le processus est relativement lent et donc l'aspect performance n'est pas un facteur essentiel pour l'application. Enfin la bibliothèque comprend de l'ordre de 10000 lignes de code.

Questions/Commentaires :

- J.-B. Stéfani : Quelles fonctions de l'OS sont-elle utilisées ?
- J.-D. Jouffrit : Il s'agit des messages, des niveaux de priorité, de la gestion de taches concurrentes (*threading*), de la gestion et du contrôle d'accès à la mémoire. Le but est d'utiliser le moins possible l'OS.
- J.-B. Stéfani : Comment la redondance est-elle gérée ?
- J.-D. Jouffrit : La bibliothèque est en charge de la Sécurité. En fait, elle assure des fonctions de consensus, les traitements sont découpés en tranches de temps (notion de *slot* temporel) en s'appuyant sur une horloge matérielle dédiée. La bibliothèque gère aussi la coordination entre les répliques.
- Luc Planche : 10 ans est une longue durée de vie et peut-être que C++ / UML ne seront plus là ?
- J.-D. Jouffrit : Disons que C++ / UML offrent une certaine pérennité. En revanche, le prototypage des nouveaux modèles s'appuient sur des COTS qui ne seront peut-être pas ceux du produit final. Il existe certainement réellement un problème de pérennité, en particulier pour ce qui concerne l'OS.

4.6 Intergiciel : Abstraction et/ou adaptation

Intervention (Pierre Crégut, France Telecom R&D) – Cf. Transparents en annexe *Erreur ! Source du renvoi introuvable.*

Cette présentation s'inscrit dans le programme de recherche long terme *Vision E-mediate*. Une grande partie des travaux s'effectue à Grenoble. Ce programme *E-Mediate* a été conçu il y a deux ans par Jean-Bernard Stéfani (alors en poste à FT R&D). Les travaux s'inscrivent aussi dans le cadre de *ObjectWeb* (cf.4.7).

La définition d'intergiciel correspond pour nous à une abstraction standardisée pour utilisation de composants sur étagère. La standardisation est tout à fait essentielle. Les besoins ressentis à FT R&D sont de faire « éclater » les intergiciels monolithiques actuels comme CORBA et de favoriser une approche à composant.

Le rôle d'un opérateur de communication est d'aller le plus loin possible (avec des liens de communication) qui sont très divers (Fibre optique, réseaux mobiles, satellites, …) et les protocoles très variés. L'objectif est de voir tout cet ensemble comme un réseau unique, même s'il est *de facto* hétérogène. Le réseau global doit fonctionner en permanence et tout changement doit pouvoir se faire "de la façon la plus souple" possible. C'est une exigence récurrente dans le domaine des Télécoms : il faut qu'il y ait toujours « un service » même dégradé. Qu'est que ce service dégradé ? Ce n'est pas toujours simple à définir.

D'autre part, la sécurité joue un rôle crucial : Utilisateurs malveillants peuvent poser des problèmes très importants de disponibilité (par exemple, *deny of service attacks*).

Compte tenu du nombre N de services et des diverses infrastructures M, notre objectif est d'éviter d'avoir NxM implémentations. Les intergiciels actuels proposent une abstraction uniforme. Ils remplissent une fonction (distribution d'objets), mais aussi différents services supplémentaires. Du fait de cette richesse, le problème essentiel est principalement de rejeter les solutions monolithiques, pas facilement composables entre elles.

Par exemple, à DTL/ASL nous avons constaté que pour les plate-formes « mondes virtuelles répartis » (projet interne à FT R&D), qui sont relativement génériques au niveau fonctionnalité, la difficulté a été l'adaptation de l'intergiciel au problème, ce qui a demandé de modifier le coeur de l'intergiciel. Les travaux de FT R&D cités sont des exemples : le but est de montrer l'approche composant pour le diagnostic ou les problèmes de composition de formalismes (temps).

Notre orientation actuelle est donc de s'appuyer sur des ORB minimaux et d'imposer une séparation des composants pour les besoins non-fonctionnels. Cette notion de survie aux évolutions a déjà été vécue dans les OS (des OS monolithique aux µnoyaux et aux exo-noyaux).

Notre but est donc de fournir un cadre architectural pour l'abstraction des ressources (bien modélisées dans ObjectWeb) et pour les exigences non-fonctionnelles.

Pour que ce cadre soit vraiment réutilisable, il est nécessaire d'intégrer la description des exigences au modèle de composants. Une description synthétique doit favoriser la « composabilité » des exigences.

Du fait des liens qui existent entre description des exigences et leur implémentation, la réalisation doit s'appuyer sur des composants dans un cadre à composants qui favorise leur composition.

A ce titre, les travaux inspirateurs que nous pouvons citer sont ceux d'Anis Arora (Ohio State University, USA) et les travaux effectués au LAAS sur le durcissement de µnoyaux.

A FT R&D, les travaux actuels portent sur des diagrammes de séquence UML pour analyser la propagation les erreurs dans les systèmes. En outre, nous cherchons à faire la jonction entre le cadre synchrone (utilisé en local) et asynchrone (utilisé pour la répartition).

En conclusion, nous sommes encore loin du but. Nous ressentons le besoin d'un cadre commun (langage – description logique) pour décrire des exigences à différents niveaux et pour différents types de besoins. La réalisation de la composition de ces exigences (interférence, couplage, articulation) reste un objectif. De plus, le problème de la validation des spécifications et des implémentations est un problème encore ouvert. Les intergiciels actuels sont inadaptés.

Questions/Commentaires :

- G. Le Lann : Une couche intergicielle est un assemblage de composants avec des capacités non-fonctionnelles. La composition est bien sûr un objectif. Cependant, il y a des compositions mortelles, par exemple l'interférence entre modules non-fonctionnels qui débouchent sur des situations d'interblocage et sur le silence global du système (exemple : arbitrage de ressource + Ordonnanceurs qui agissent indépendamment). *Quelles sont les possibilités d'identifier ces impossibilités ?*

- P. Crégut : Le problème des interactions non voulues est un problème très vieux dans la téléphonie. Nous n'avons pas encore trouvé de solutions miracles. Il est probable qu'une approche conservative (lister explicitement les types d'assemblages autorisés) soit la seul e solution réaliste dans l'immédiat.

- G. Le Lann: Il semble qu'il soit nécessaire de faire reposer l'obligation de fournir une description des boîtes sur les épaules des fournisseurs de boîtes.

- P. Crégut : Oui.

- J.-C. Fabre : En effet, il apparaît nécessaire d'avoir un niveau de description des boîtes suffisamment détaillé (ni trop concret, ni trop abstrait), pour identifier des conflits et surtout homogène pour pouvoir les confronter.

4.7 ObjectWeb: Une initiative -source libre- pour le développement d'intergiciels : une introduction

Intervention (Jean-Bernard Stéfani, INRIA) – Cf. Transparents en annexe B6

Cette présentation de l'initiative ObjectWeb couvre à la fois sa nature, son organisation et ses objectifs. ObjectWeb est une initiative pour le développement d'intergiciels en source libre. Le but est de construire des intergiciels disponibles en *copyleft* (licence GPL ou dérivé). Le code source est disponible et modifiable à condition qu'il y ait redistribution des modifications sous le même modèle de licence.

Au départ, il s'agissait d'une initiative de l'INRIA, FT R&D, Bull/Evidian. Depuis Janvier 2002 il s'agit d'un Consortium International hébergé par l'INRIA. Il y a donc de nombreux autres partenaires. Plusieurs projets d'accompagnement sont prévus ou en place, nationaux (RNTL, RNRT, …) industriel, européen (FP6, IP)

Dans ObjectWeb, nous souhaitons fournir une aide à la programmation répartie sous la forme d'intergiciels à composants. Il y a certes un fort tropisme vers Java, mais ce n'est pas exclusif.

Pourquoi cette initiative ? Nombreuses sont les opportunités pour construire des intergiciels. L'intergiciel à terme devient une commodité (*utility* en anglais, cf. théorie économique) au sens où il doit être utilisé par tous, à un coût marginal nul. Son but est, de façon très générique, de fournir une interface standard pour la programmation sur réseaux. Sa vocation essentielle est d'être diffusé le plus largement possible. Un corrolaire est donc une baisse des coûts de production inversement proportionnelle à sa diffusion.

Clairement, il n'existe pas aujourd'hui d'intergiciel à la fois flexible et avec haute couverture fonctionnelle qui soit en source libre. Nos objectifs à atteindre sont de deux natures :

- Objectif technique = Construire un logiciel exploitable industriellement
- Objectif de Communauté = Construire et faire vivre une communauté

Comme c'est déjà le cas pour d'autres initiatives de développement de logiciel en source libre, le succès de notre initiative repose sur une effet d'échelle, dans le sens où des contributions extérieures s'adjoignent aux développements des fondateurs initiaux, pour construire une base de code.

En ce qui concerne les objectifs techniques, quatre pistes sont suivies :

- Développement totalement décentralisé, ce qui est un défi en soi, sous forme de projets séparés mais partageant un modèle d'architecture homogène.
- Respect, autant que faire se peut, des standards du moment, comme CORBA (de l'OMG, avec les extensions composants), Java (J2E, serveurs d'EJB, embarqué), W3C (web services).
- Obtention d'une haute couverture fonctionnelle. La liste des fonctions peut être très longue, et actuellement il n'existe pas de produit sur le marché avec l'ensemble des fonctionnalités attendues. Certains éléments sont présents dans ObjectWeb, mais en terme de sûreté de fonctionnement et de sécurité, ou encore partage de charge, les composants font défaut et donc les contributions sont les bienvenues.
- Qualité suffisante pour être déployé dans des applications réelles. Cet objectif repose sur la mise en place d'une méthode de développement, avec des normes internes au projet pour la systématisation des tests, et autres *benchmarks*, etc.

En ce qui concerne les objectifs de communauté, nos ambitions sont de :

- Développer une communauté de développeurs / utilisateurs, au travers d'un site Web http://www.objectweb.org, visant à évoluer sous forme de site Source Forge, et par l'organisation d'une conférence annuelle pour rassembler les contributeurs.
- Utiliser des financements publics pour aider au développement, y compris de façon indépendante à l'initiative des développeurs.
- Constituer une base amont avec un ensemble d'universités pour aider au développement de supports d'enseignement et au développement d'applications réparties.
- Favoriser les relations avec les acteurs du monde industriel, ce qui constitue une condition *sine qua non* pour le succès. C'est déjà la cas, actuellement ; à ce titre, certains éléments de la base de code ObjectWeb sont utilisés pour des activités commerciales.

L'organisation d'ObjectWeb repose sur une totale autonomie des projets partenaires. ObjectWeb fournit un cadre commun de support à ses membres, personnes physiques, organismes (personnalités morales). Ces derniers élisent le conseil d'administration (*Board*) du Consortium. Il s'appuie sur un collège d'architectes pour superviser les développements techniques au sein d'ObjectWeb. Ce collège n'est pas élu mais composé de personnalités scientifiques et techniques. Sa mission est de maintenir l'intégrité architecturale au sein d'ObjectWeb. Il effectue des choix techniques sur les priorités de développement, le modèle d'architecture. Un comité exécutif gère, quant à lui, la vie du consortium au quotidien.

En pratique, actuellement nous recevons plusieurs milliers de connexions par jour sur notre site, le composant le plus attractif étant Jonas qui a plusieurs centaines d'utilisateurs actuellement.

La liste des projets en cours est la suivante :

- Jonathan = Canevas de construction d'ORB conformes à des spécifications de type CORBA / RMI.
- Jonas = Logiciel, logiquement au dessus de Jonathan, qui fournit des fonctions conformes à la spécification EJB de Sun (transaction par composants).
- Joram = Middleware à messages (communications asynchrones persistantes).
- Jorm = Gestion de persistance générique, c'est-à-dire rendant un objet Java persistant avec tout type de stockage (DB, LFS, …)
- OpenCCM = Conforme à la spécification CCM de l'OMG (~= EJB)
- Proactive = Calcul sur grille, c'est-à-dire infrastructure d'objets répartis pour calculs parallèles et distribués.

D'un point de vue architectural, notre vision conceptuelle est clairement non monolithique. Nous nous positionnons *a contrario* du principe d'un seul produit qui fasse tout. Notre philosophie est celle d'une infrastructure basée sur des canevas qui permettent de construire / d'instancier l'intergiciel idéal pour l'application visée.

Pour remplir cet objectif, nous souhaitons élaborer une base de composants logiciels pour construire des intergiciels. Un composant est une unité d'exécution similaire à la notion objet, mais aussi une unité de configuration dynamique (reconfiguration des liens, par exemple).

Actuellement les projets dans ObjectWeb ont des origines diverses, et, pour l'instant, la base de logiciel n'est pas conçue de manière concertée. Il sera nécessaire d'effectuer une réingénierie complète de la base actuelle pour imposer un modèle architectural homogène.

Le canevas envisagé concerne les aspects : nommage, liaisons, composants, ressources et protocoles. Ce canevas sera partagé par l'ensemble des composants, par exemple un canevas de construction de pile de protocoles de communication. Il est clair que l'aspect changement dynamique en fonction des besoins est primordial.

Du point de vue de la couverture fonctionnelle, nous donnons une grande importance au caractère administrable de l'ensemble de ces composants, en particulier pour ce qui concerne les fonctions de supervision et de gestion de configuration.

Bien entendu, la disponibilité est un aspect crucial, par exemple les serveurs d'application Internet. C'est d'ailleurs un des domaines du succès de Jonas, tout autant que les très grands systèmes d'informations basés sur cette technologie. C'est certainement l'un des points sur lesquels nous espérons de nombreuses contributions, compte tenu des besoins en la matière.

Questions/Commentaires :

- J. Arlat : Y-a-t'il des activités d'évaluation de la robustesse et de la Sûreté de Fonctionnement ?
- J.-B. Stéfani : Non. Pas encore, mais c'est souhaité. Il y a déjà eu une grande campagne d'évaluation de performance sur Jonas.
- L. Planche : Du point de vue utilisateur pur, quel est le support fourni, un support technique non garanti pouvant faire préférer une solution propriétaire?
- J.-B. Stéfani: Notons que la réactivité du support Linux est bien meilleure que la *hotline* Microsoft. C'est un fait. Sur Jonas a priori ça fonctionne : beaucoup de questions sont résolues par d'autres utilisateurs. Notons aussi qu'il n'y a pas de support garanti de la part même d'ObjectWeb. ObjectWeb suscite l'interaction. En outre, la condition du succès est bien entendu l'appropriation de composants système par une large base d'utilisateurs.
- ObjectWeb est une communauté d'intérêt et donc il n'y aura jamais de garanties institutionnelles de la part d'ObjectWeb. Cependant, il y aura des garanties tangibles de la part de partenaires présents dans ObjectWeb. Il est aussi certain qu'exploiter un logiciel libre requiert une expertise interne. Pour les modifications qui s'imposent, nous souhaitons faciliter ce type d'intervention par l'architecture du code, les modifications étant inévitables.
- J.-P. Auclair : Pourquoi les solutions logiciels libres émergent-elles ? Les gens qui développent ne sont pas des philanthropes. Comment ça marche ?
- J.-B. Stéfani: Oui, la philanthropie existe… j'en suis d'ailleurs un exemple vivant ! (rires). Il existe plusieurs papiers très sérieux dont un d'un économiste Toulousain et du MIT qui affirment et démontrent la viabilité économique de cette stratégie[2].
- J.-P. Auclair : N'y-a-t-il pas un risque comme celui de la bulle Internet ?
- J.-B. Stéfani: ObjectWeb est supporté par des partenaires solides : INRIA, EPTS bien connu, Bull et France Telecom, grandes sociétés. L'infrastructure repose sur la notion d'économie de commodité. Une infrastructure logicielle doit etre vu comme un bien commun.

[2] J. Lerner (Harvard), J. Tirole (CERAS, Toulouse & MIT) : The simple economics of open source -- National Bureau of Economic Research (NBER) Working Paper No.w7600, March 2000.

4.8 Sûreté de fonctionnement d'intergiciel CORBA: caractérisation de services par injection de fautes

Intervention (Eric Marsden, LAAS-CNRS) – Cf. Transparents en annexe B7

Notre objectif est de fournir des éléments d'information aux fournisseurs de COTS, notamment d'ORB, et aussi aux intégrateurs de système, en particulier en identifiant les problèmes de robustesse, puis par empaquetage, de développer des *wrappers* pour augmenter la robustesse.

La technique de base est l'injection de fautes par logiciel (SWIFI) pour effectuer l'analyse des modes de défaillance d'un composant. Pour effectuer ce type d'expérience certains choix préliminaires sont à faire : Quelles types de fautes ? Quelle type d'injection ? Quelle charge applicative ? Quelles observations ?

Dans le cas d'un ORB , une requête traverse plusieurs couches : ORB, OS, matérielle. Des fautes peuvent intervenir dans chacune de ces couches. Dans les travaux actuels, les types de fautes correspondent à des fautes physiques transitoires. Elles sont appliquées aux requêtes IIOP client-serveur et correspondent à des corruptions de messages. D'autres techniques sont actuellement utilisées pour couvrir différents composants de l'intergiciel ; plus classiquement des techniques de corruption de la mémoire s'effectuent au niveau du courtier, mais aussi des fautes de conception peuvent être simulées par des techniques de mutation, et enfin la modification des paramètres d'appels de l'interface de l'ORB. Nous simulons aussi d'autres types de fautes de l'environnement (manque de mémoire, surcharge réseau).

Aujourd'hui ce sont les services qui ont été ciblés : ce sont des services standardisés au sens CORBA. Le service de nom ou le service d'événement sont des services qui possèdent en effet une interface standard et, donc, il est possible de comparer différentes implémentations. Ils sont en général très importants pour le fonctionnement du système et des applications. C'est un pas vers la caractérisation globale du courtier. Les modes de défaillance relevés sont les suivants : 1) Défaillance du nœud (*site crash*); 2) Défaillance du service (*service crash*); 3) Blocage du service (*service hang*); 4) Défaillance au niveau applicatif (*application failure*). Ce dernier cas signifie qu'une erreur s'est propagée jusqu'à la couche applicative. C'est pour nous le cas le plus grave. Il signifie qu'aucun des mécanismes de détection n'a permis de détecter l'erreur. De plus, la norme CORBA stipule des exceptions pour lesquelles les implémentations se doivent de mettre en œuvre des mécanismes de détection à même de les lever lors de conditions incorrectes particulières. Ce type d'observation est aussi effectué lors de nos expériences.

Nous avons donc réalisé un banc expérimental sur lequel un contrôleur lance une charge applicative et le service cible. Pour le service de noms, l'application de charge effectue une construction déterministe d'un arbre de nom, puis demande la résolution d'un nom. La corruption des messages de requête a lieu au bout d'un intervalle temporel fixé mais paramétrable.

Les fautes physiques modélisées correspondent au bit flip ou à la mise à zéro de 2 octets successifs. Ce modèle de faute est basé sur un article de Stone et Partridge[3] (à SIGCOMM 2000) où sont rapportés les fautes relevées sur des systèmes distribués locaux et à grande échelle. Il s'avère que ces types de fautes sont les plus fréquemment observés (une faute de cette nature toutes les 80h sur un LAN 100Mb fortement chargé). Ces erreurs peuvent aussi modéliser l'effet des fautes physiques dans les matériels clients/serveurs ainsi qu'au niveau du réseau. Ce modèle de faute permet aussi d'étudier la propagation des erreurs au sein d'un objet CORBA vers un autre objet CORBA distant. Dans une certaine mesure, il permet d'avoir une idée de l'impact d'attaques par un agent malicieu qui consisterait à corrompre volontairement des bits ou des octets dans des messages IIOP. En effet, l'injection est effectuée sur les messages IIOP avant empaquetage et transmission par une pile TCP.

Nous présentons ici des résultats sur 4 implémentations courantes d'ORB (JavaORB, Orbacus, OmniORB et ORBit). Nous constatons des résultats assez disparates et des profils de comportement en présence de fautes bien différenciés selon les implémentations. En fonction d'un profil d'utilisation donné, on peut donc classer les différentes implémentations : JavaORB et ORBit ne sont pas très bon. La répartition des exceptions n'est pas la même selon les cibles. Cela signifie que le code dévolu à la détection au sein de l'implémentation ne sont pas aux mêmes niveaux selon les cibles. Des résultats similaires ont été observés sur le Service d'Evénements.

Nous avons effectué une analyse plus fine selon le bit injecté, à savoir les 96 bits de l'entête (*header*) d'un message IIOP. Un bit particulier, le bit 48 détermine l'ordre des octets (*byte order)*. La corruption de ce bit conduit essentiellement à un service crash.

En conclusion, nous avons défini une méthode de caractérisation de la robustesse d'intergiciel CORBA. Actuellement nous disposons de résultats sur les modes de défaillance d'un service CORBA par corruption des messages IIOP, résultats qui peuvent être très différents selon les implémentations. L'environnement expérimental est facilement portable et la technique d'injection facile à appliquer à d'autres cibles. De plus elle est faiblement intrusive.

[3] J. Stone and C. Partridge, "When the CRC and TCP checksum disagree", in *Proceedings of the 2000 ACM SIGCOMM Conference* , pp. 309-319, 2000.

Enfin, actuellement le modèle de fautes est limité : robustesse par rapport à une certaine classe de fautes uniquement. Nos travaux en cours visent à étendre le modèle des fautes injectées au cas de la propagation d'erreurs depuis l'OS sous-jacent. C'est une forme de test de robustesse vis-à-vis de problèmes d'acquisition de ressources permettant de faire une analyse d'un intergiciel COTS du point de vue du traitement des codes d'erreurs et des exceptions de bas niveau relevées par l'OS.

Questions/Commentaires :

- P . Foix : Comment étendre ce travail pour l'utiliser dans une méthode de développement : choix de produit, etc. ?

- J.-C. Fabre : Il existe tout un travail en amont pour connaître le contexte d'utilisation du produit cible, pour adapter l'évaluation et l'interprétation des résultats. La première phase du processus d'évaluation consiste à identifier le contexte (profil d'utilisation du système). Les résultats présentés ont été faits avec les mêmes charges spécifiques. Pour obtenir des réponses concernant un système particulier, il est aussi nécessaire d'affiner aussi le modèle de faute et la forme de l'injection. Par ailleurs, il faut noter que ce type de résultat, s'il est intéressant pour la sélection d'un intergiciel COTS, il apporte aussi un éclairage sur l'évaluation de l'évolution d'un intergiciel au travers des ses différentes versions successives. Nous avons aussi des résultats montrant des modifications importantes de comportement sur deux versions successives d'un intergiciel COTS.

4.9 Point sur l'utilisation des technologies middleware dans les centres de contrôle et de mission des systèmes spatiaux

Intervention (P. Rouzet, Astrium)) – Cf. Transparents en annexe B8

Cette présentation porte sur l'utilisation d'intergiciels dans les centres de contrôle et de missions pour les systèmes spatiaux. Elle aborde un retour d'expérience sur ce point, les aspects positifs et négatifs de l'expérience, et aussi les tendances actuelles.

Essentiellement, nos solutions s'appuient sur CORBA pour réaliser des applications distribuées pour de nombreuses raisons : plus grande capacité de calcul, exigences plus complexes, flexibilité. Plusieurs possibilités avaient été envisagées avant le choix de CORBA: couches basses (sockets), solutions maisons, intergiciel propriétaire (ILOG Broker).

Les intérêts de CORBA pour nous sont multiples, principalement l'existence d'une norme, mais aussi la fourniture d'un modèle adapté aux langages utilisés. D'autres part, CORBA représente une reconnaissance certaine du point de vue des clients d'Astrium.

Nous avons 2 projets de composants:

- *Open Center* : composants techniques et métiers utilisés dans les centres de contrôle, bancs de test, simulateurs satellites, avec un fort objectif de réutilisation qui a été atteint.

- Produit *OpsWare* : logiciels pour l'automatisation des opérations satellite. Mêmes utilisations que précédemment.

En termes d'application, nous pouvons donner l'exemple d'Intelsat FDC. Il s'appuie sur des *clusters* HP avec des contraintes de tolérance aux fautes en environnement hétérogène. Pour les couches de base, nous utilisons un courtier (*broker*) maison d'Intelsat.

Le point positif est l'interopérabilité. Ce système doit cependant obéir à des contraintes de performance et de disponibilité, dans une architecture machine assez complexe. Il s'agit d'un gros projet (de l'ordre de 40 personnes, 12 M€).

Nous énonçons ici quelques uns des problèmes rencontrés. Nous avons eu des difficultés concernant la relocalisation de processus, car le broker gérait très mal ces aspects là. Il a donc été nécessaire de redévelopper une surcouche pour gérer ces problèmes. L'utilisation d'Orbix fut à la fois une chance car il a permis de résoudre ces problèmes, mais il a aussi été source de nouvelles difficultés : Orbix ne masque pas totalement les couches de communication. Plus généralement, certains aspects ne sont pas bien maîtrisés dans l'implémentation des composants sur étagère, ce qui pose des problèmes au moment de l'adaptation.

Pour des raisons contractuelles et de partenariat, nous avons été amenés à effectuer d'autres cas d'étude avec un environnement hétérogène, en particulier le développement d'un centre de contrôle, pas entièrement développé par Astrium, mais en coopération avec des partenaires qui apportent leur logiciel. L'interopérabilité est alors un facteur

essentiel. L'IDL (*Interface Definition Language*) est un moyen très puissant de modéliser et de communiquer sur des interfaces. De plus en plus, les systèmes sont hétérogènes au niveau des langages, des IHM et autres capacités graphiques, des noyaux. L'intergiciel permet de faire abstraction de ces aspects-là. Nous énonçons ci-dessous les leçons que nous avons pu tirer de ces expériences.

Points positifs :

- L'intergiciel facilite l'interaction et simplifie l'implémentation des communications inter-processus. Il permet une uniformisation de la mise en œuvre des interfaces (description, modèle sous-jacent). Il permet le déploiement transparent de processus sur les *clusters* (avec néanmoins certaines limitations pour la tolérance aux fautes).

- Nous avons observé une relative indépendance des implémentations des ORB.

- Les différents modèles de concurrence permettent de gérer la dynamique et la performance.

- Enfin, d'autres fonctionnalités intéressantes sont fournies par des services associés.

Points négatifs :

- Persistance de la culture des interfaces fonctionnelles : les développeurs sont parfois perturbés par la philosophie de l'orienté objet.

- Maîtrise de l'implémentation : Un certain nombre de règles sont à respecter, règles qui ne sont pas toujours explicitement décrites par le fournisseur.

- Tolérance aux fautes : c'est un aspect qui pose le plus de problèmes et qui nécessite souvent de comprendre comment fonctionne l'ORB dans un contexte de stress. Avec nos contraintes de coût et de temps, il n'est pas toujours possible de faire des campagnes similaires à celles qui ont été décrite par E. Marsden. C'est pourtant une information nécessaire pour développer la couche supplémentaire qui ajoute la tolérance aux fautes.

- Tentation d'utilisation des aspects non standards fournis par les ORB: bien qu'ils soient souvent très utiles, dans le cadre d'application hétérogènes, si un partenaire les utilise, ces aspects non standard entraînent une perte d'interopérabilité.

- Aspects commerciaux : Les coûts de licences sont prohibitifs aussi bien du point de vue développement que du déploiement. Un coût de licences de 300K€ pour un centre de contrôle est indéfendable. De plus, le support technique est loin d'être efficace, il en est de même pour obtenir certaines clarifications des fonctionnalités. En outre, les ORB ont des bugs. Enfin, la disponibilité des solutions CORBA sur les nombreuses plates-formes (types, versions) dépend du bon vouloir du fournisseur.

- Evolution : Revenir à la norme. De plus en plus, nous nous intéressons aux ORB en source libre qui offrent (souvent) un meilleur support que les ORB commerciaux. Il est en effet nécessaire de comprendre et d'anticiper le comportement du coutier vis-à-vis des pannes, pour ajouter la couche qui va bien. En ce qui concerne l'utilisation du *Naming Service*, élément permettant l'évolution du système (configuration dynamique, changement de serveur), jusqu'à maintenant nous avons choisi des solutions plus statiques au travers d'un annuaire de références propriétaire, car le NS n'est pas interopérable.

Questions/Commentaires :

- J.-C. Fabre : les fournisseurs cherchent à biaiser le mécanisme de la norme pour verrouiller leur clientèle. Il y a souvent un problème de clarification de ce qui est supporté par une version donnée de l'ORB.

- J.-P. Auclair : A quelle époque avez-vous pris la décision de vous lancer dans CORBA ?

- P. Rouzet : Depuis 95 / 96. Jusqu'à présent nous avons été fidèle à Orbix. Nous n'allons cependant pas rester sur Orbix encore très longtemps pour les raisons de coût évoquées.

4.10 Middleware et tolérance aux fautes dans les futurs systèmes de supervision et de contrôle de Thales

Intervention (Virginie Watine, Thales Group) – Cf. Transparents en annexe B9

Je fais partie d'une entité qui étudie des solutions à base d'intergiciel pour toutes les entités du groupe Thales. Notre problème essentiel est la durée des cycles de vie de nos systèmes, à savoir de 10 à 20 ans. Nos systèmes évoluent beaucoup durant une telle période et doivent s'appuyer sur des technologies du commerce. C'est vrai pour les CPU, les

OS, les intergiciels. Si la frontière est mal gérée entre les systèmes à longue durée de vie et les techniques qui évoluent, il y a en fait une perte énorme d'énergies. Ceci permet de souligner l'importance de la standardisation.

Au sein de Thalès, nous avons 2 programmes pilotes :

- MIRROR : Développement à base de modèles et gestion de lignes de produits (outillage de la démarche MDA – *Model Friven Architecture*).

- ALICE : Architecture basée composant et services associés, avec une préoccupation forte du point de vue de la séparation entre propriétés fonctionnelles et propriétés non-fonctionnelles. La notion de « conteneur » de composant permet séparer les métiers, les compétences, les points de vue. C'est dans le cadre de ce programme pilote que sont étudiés les futurs intergiciels de Thales.

Les buts recherchés sont la réduction des délais et des coûts, la séparation plate-forme métier / plate-forme architecture, et la minimisation des risques.

Le second volet de la problématique concerne la segmentation des systèmes selon leurs caractéristiques non-fonctionnelles. Il n'y a pas les mêmes exigences pour des systèmes de gestion de données techniques et pour les systèmes de contrôle proche du temps-réel (système de surveillance, supervision, …). A exigences différentes, intergiciels sensiblement différents. Nous ne croyons pas à un intergiciel universel idéal "*one does fit all*". Sur le marché les offres disponibles varient selon les segments. Le problème est de qualifier les différentes offres quand les offres sont disponibles. Pour les segments proches du temps-réel, CORBA est la bonne solution. Cependant, nous avons observé une grande hétérogénéité de produits, en fonctionnalité et en qualité.

Les raisons de notre choix de CORBA est le fait qu'il est aujourd'hui le seul standard non propriétaire. Il existe donc plusieurs implémentations, certes de différente qualité, mais qui laissent cependant la liberté de choix. De plus, CORBA supporte l'interopérabilité vis-à-vis de différentes plate-formes et de différents langages. Il n'y a donc pas d'impérieuse nécessité d'un ORB unique qui supporte toutes les plates-formes et langages requis. CORBA permet d'autre part d'effectuer une intégration douce des logiciels existants. Il n'est pas nécessaire de faire table rase du passé. CORBA supporte l'asynchronisme, le synchronisme différé, et bientôt la *data distribution*.

Nous observons une évolution de CORBA vers des caractéristiques de plus en plus spécifiques : CORBA-RT / FT-CORBA / *Data Parallel*. Le monde initial de CORBA (informatique de gestion) est aujourd'hui occupé par les EJB et le canevas transactionnel.

Notre avis sur le modèle à composant de CORBA CCM est mitigé. Notre souhait à Thales est de garder l'idée composant / conteneur mais d'ajouter d'autres services techniques dans le conteneur : équilibrage de charge (*load balancing*) temps-réel, etc. Se pose alors le problème de la composition interne du container qui doit être la plus flexible possible.

Le produit PERCO (*PowERed COrba*) que nous développons est une plate-forme multi-domaines permettant de configurer et executer des systèmes du segment 'proche du temps-réel' à exigence de tolérance aux fautes. Elle est donc basée sur CORBA et CCM avec des extensions propres à cette classe de systèmes.. L'ère du gros intergiciel est certainement révolue et en tout cas pas satisfaisante de notre point de vue. La modularité par rapport aux fonctionnalités et aux qualités d'implémentation est impérative.

En outre, Thales participe activement à l'OMG pour faire évoluer les standards. Enfin, PERCO est destiné à être une plate-forme privilégiée cible de notre processus de développement orienté MDA (*Model Driven Architecture*) étudié dans la cadre du programme pilote complémentaire MIRROR.

Le support de la tolérance aux fautes de PERCO sera basé sur FT-CORBA. Notre approche est pragmatique et nous allons meneré des études préliminaires pour répondre à certaines questions : Quels sont les besoins chez Thales ? Qu'est qui existe aujourd'hui sur le marché ? Quel est le niveau de séparation maximale des problématiques fonctionnelles et non-fonctionnelles ? Quel support est nécessaire du point de vue des outils de modélisation pour utiliser MDA ? et en déduire les compléments à développer en interne. Ces travaux commenceront en 2003.

Questions/Commentaires :

- G. Le Lann : Comment avez-vous pu mener un projet interne tel que PERCO aussi ambitieux ?

- V. Watine: Nous avons reçu un support européen. Nous espérons rendre les résultats de PERCO disponibles auprès des utilisateurs et intégrés à de futurs standards.

- G. Le Lann : UBSS était le projet précédent. Comment s'est effectuée la liaison entre les deux projets/systèmes ?

- V. Watine: Nous devrons appliquer un processus continu pour une migration progressive de l'un vers l'autre. A terme, nous prévoyons une bascule totale sur PERCO. UBSS était un logiciel beaucoup trop coûteux. PERCO a été aussi financé par Thales Group et par les unités qui y trouvent un intérêt. Nous avons défini des solutions d'interopérabilité entre les différents segments, même si on ne croit pas à une solution unique qui supporte tous les segments. Il a été important d'identifier les préoccupations communes à tous les segments.

- G. Le Lann : Comment faire coller les propriétés propres ajoutées et celles fournies par le standard ?

- V. Watine: On a pas besoin de tout ce qu'offre le standard, et certains services ne sont pas encore standardisés (par exemple, la distribution de données). Nous devons donc mener une étude besoin/existant avant de lancer un développement. Notre politique est d'être conforme aux standard, sauf si c'est infaisable (standard inexistant) et de pousser à la normalisation nos extensions.

- G. Le Lann : Comment modéliser les contraintes applicatives par rapport à ce que les COTS peuvent fournir ? Problème des responsabilités ?

- V. Watine: CORBA n'est pas utilisé sur les systèmes les plus critiques. Les niveaux les plus critiques où l'on trouve CORBA sont les systèmes de supervision, tels les systèmes de contrôle de trafic aérien.

- G.Le Lann : Si on trouve qu'un avion s'est écrasé à cause de VxWorks, pourquoi WindRiver ne serait-il pas responsable ?

- Réponse (de l'audience) : Ils indiquent dans la licence qu'il ne sont pas responsables pour une utilisation dans des applications critiques.

- G. Le Lann : Finalement tout est redéveloppé en interne. Le besoin de solution fiable et robuste n'est pas uniquement corrélé à des marchés de niche : par exemple , l'automobile.

- V. Watine: Le jour où la responsabilité des fournisseurs de COTS pourra être engagée, on quittera le monde un peu fou du logiciel, où aujourd'hui il est malheureusement acceptable qu'un logiciel se plante !

- G. Le Lann : Depuis toujours il existe un mythe que l'informatique est tellement complexe, que c'est normal que ca ne marche pas.

- P. Rouzet : Le problème dans le spatial est que nous n'avons aucun poids sur les fournisseurs et donc sur l'évolution du marché. Même en y allant ensemble, nous ne représentons que « trois fois rien » ! Notre seul atout est de jouer sur l'image de marque.

- G. Le Lann : Le problème est grave, confer le Bug de l'an 2000 (ou Pseudo Bug.) Transposition à l'industrie automobile, un problème technique sur un véhicule qui serait connu de tous n'engagerait pas la responsabilité du constructeur. Aujourd'hui les constructeurs rappellent toute une série de véhicules et effectuent les modifications gratuitement. On ne vous demande pas 1000€ pour résoudre un problème inhérent au véhicule. Ce type de pratique dans l'industrie du logiciel est tout à fait inacceptable.

- P. Foix : Nous avons mis en place un plate-forme de partage d'informations sur les COTS. Plus y aura d'industriels dans ce programme européen, plus il sera possible de faire pression sur les fournisseurs de COTS.

5 Synthèse des discussions et conclusions de l'atelier

Jean-Charles Fabre ouvre le débat en donnant tout d'abord la parole aux participants qui ne se sont pas exprimés au cours des exposés.

Elias Martins (SNCF)

Le premier Integiciel utilisé dans les postes d'aiguillages a été installé sur le poste de Marseille et gère une partie du TGV méditerranée et toutes les circulations de Marseille (systèmes MISTRAL). La décision à été prise à peu près au même moment qu'Astrium et pour les même raisons. Nous nous sommes orientés vers un choix propriétaire : ILOG Broker + ILOG Views + ILOG Server. Pour répondre à nos contraintes d'architecture et FDMS, nous avons eu besoin de développer le code de gestion de la redondance ainsi que des mécanismes de dynamisation des données. Aujourd'hui, si nous envisagions un portage vers CORBA ou vers une version plus récente de l'intergiciel Ilog, il serait nécessaire de redévelopper une grande partie de ce code. Le développement et la validation du système MISTRAL s'est effectué sur 5 ans, sans les COTS il aurait duré probablement 10 ans. Ceci est déjà un grand avantage, mais nous sentons bien que nous sommes fortement liés à l'éditeur Ilog.

Eric Bornschegl (ESA)

Il existe deux mondes différents : le monde du sol et celui de l'embarqué. Au niveau des systèmes Sol, l'ESA encourage les plate-formes communes " SCOS 2000 ". Le maillage de stations sols est déjà implémenté par différentes entreprises européennes. Il est supervisée par l'ESOC (Darmstadt). Côté embarqué, une petite entreprise britannique " Science Systems " offre un MicroCorba pour satellite au plan britannique. Il est extrêmement réduit. Pour le bord, c'est un monde un peu nouveau pour l'ESA. Bien entendu, il existe une tendance à l'homogénéisation entre ce qui est fait au sol et ce qui fait au bord, par exemple dans une perspective de constellation de satellites. A long terme, nous envisageons un certaine transparence entre un maillage sol et un maillage spatial. Dans le spatial, les gens n'aiment pas trop le logiciel qui est considéré comme coûteux, difficilement testable. Au niveau des agences, nous soutenons l'effort de standardisation, de modularisation. Notre souhait est de voir apparaître un outillage industriellement viable. Par exemple, UML devient une pratique qui se développe. Il est, en outre, important de noter que les besoins ne sont pas les mêmes selon la classes des satellites, par exemple, entre l'ISS et les satellites de télécom. Il faut respecter une séparation entre l'aspect interfaçage communication & logiciel et la charge utile (*payload*) fonctionnelle du satellite et autant que faire se peut, réduire la partie spécifique à la mission, pour faciliter l'intégration et réduire ses coûts. Une homogénéisation entre charge utile et avionique est souhaitable.

D'autre part, une étude est en cours avec Astrium pour évaluer ce que l'on peut gagner à utiliser des intergiciel dans le spatial. Cette approche est extrêmement nouvelle pour le spatial et est considéré comme instable encore aujourd'hui. Nous ne cherchons pas à précipiter le mouvement, mais à suivre une approche inverse basée sur une étude des besoins par rapport aux contraintes de l'embarqué spatial selon une approche proposée par l'INRIA (modélisation de la problématique). Ensuite dans une deuxième phase, nous nous posons les questions suivantes : Qu'offrent les produits commerciaux ? Faut-il développer des modules algorithmique pour leur intégration ensuite au sein du COTS commercial choisi ? Cette phase induit un prototypage et va démarrer. Notre approche est très prudente et recherche un bon équilibre entre en l'aspect théorique et l'aspect expérimental.

Marie-Line Valentin (Airbus)

Historiquement, l'utilisation de CORBA dans notre contexte avionique s'est faite en premier lieu dans les Systèmes d'Information, c'est-à-dire au sol, au niveau des ateliers système et logiciels (mise en œuvre de Orbix pour ne pas le citer.) Bien sûr, nous avons des problèmes de support et de coût. Nous nous tournons maintenant vers des solutions open-source (ex : mise en place des plate-formes CORBA TAO dans les ateliers de simulation). Ces systèmes présentent certaines contraintes temporelles (mais pas de temps-réel "dur"). Dans le cadre du Projet A-380, CORBA est une solution envisageable pour l'architecture des applications de maintenance (OMS : On-board Maintenance System) et l'accès à la documentation embarquée. Clairement, ce type de solution n'est pas envisagé côté avionique, le projet A-380 introduisant déjà beaucoup d'innovations technologiques (avionique modulaire, réseau Ethernet, …).

Discussion et bilan de la journée

J.-C. Fabre (LAAS):

Lors de l'organisation de la journée, nous avons ressenti un intérêt fort pour le sujet que nous avons abordé. Cette journée a été, de mon point de vue, riche d'enseignements. Sans nul doute, d'autres retours d'expérience sont à présenter et à discuter pour nourrir notre réflexion.

Comment poursuivre sur ce thème ? Dans le cadre du *RIS* c'est la notion de groupe de travail qui permet de mener une étude approfondie sur un thème. Nous aimerions avoir l'avis des participants sur l'opportunité de ce mode de travail. Au bénéfice de tous, il est toujours intéressant de confonter les expériences et d'en faire une synthèse. Typiquement un groupe de travail s'étale sur 12 à 18 mois et est ouvert à d'autres participants apportant leur expérience spécifique.

Quelques thèmes peuvent être proposés pour aborder la problématique qui a été illustrée aujourd'hui par les exposés, par exemple :

- Evaluation / Caractérisation des solutions existantes ; cet aspect est très important pour guider les choix des composants à utiliser.
- Spécialisation d'intergiciel, et instantiation d'intergiciels taillés spécifiquement à partir de composants réutilisables.

Il est bien évident que chacun peut suggérer des thèmes ou des participants supplémentaires pour des thèmes spécifiques.

J. Arlat (LAAS):

La journée d'aujourd'hui a été centrée sur une problématique au cœur des préoccupations des partenaires du *RIS*. Cette journée confirme que la notion d'intergiciel est un thème moteur pour les membres du *RIS* et au-delà.

Il existe effectivement de nombreux sous-thèmes : Distribution–Communication, Pérennité et Modélisation des Exigences, des fonctionnalités fournies, Caractérisation des solutions disponibles, Persistance.

L'Intergiciel est au cœur de ces préoccupations et peut être multiforme : standard générique, sur mesure, paramétré, etc. Aujourd'hui nous souhaiterions synthétiser les avis sur l'action à suivre. Nous fournirons ces informations au comité directeur qui choisira en conséquence de lancer ou non un groupe de travail sur ce thème. Il sera en effet possible d'étendre la palette des domaines industriels représentés, par exemple l'automobile. Si, comme les débats d'aujourd'hui le laissent à penser, cette problématique est largement paratgée, notre proposition au comité directeur sera de mener un groupe de travail sur ce thème là. Le débat est ouvert.

P. Foix (THALES):

Beaucoup de matière et de travaux en cours. Nous sommes favorables au montage d'un groupe de travail. Cependant, je pense que nous devons rester dans le cadre de la thématique du $\mathcal{RIS}$: la Sûreté de fonctionnement, la Certification, le comportement en présence de fautes. Je pense qu'il ne faut pas chercher à se disperser sur les thématiques connexes qui sont très vastes.

J.-P. Blanquart (Astrium):

Il y a effectivement beaucoup à partager. Les problématiques sont communes et nous devons partager l'information et la compléter. Faut-il se limiter à la Sûreté de Fonctionnement ? Oui bien sûr, mais la Sûreté de Fonctionnement est un vaste domaine. Beaucoup de choses qui débordaient de ce thème durant la journée me paraissent pourtant importantes. Avant de faire de la Sûreté de Fonctionnement, il faut réaliser un système qui fonctionne. Il faut certes établir un cadre, mais le laisser ouvert.

J.-P. Auclair (Consultant):

La Sûreté de Fonctionnement peut tirer tout le reste.

E.Bornschegl (ESA):

Il existe certains reproches de certains esprits conservateurs à l'agence, par exemple concernant la multiplication des couches, la complexification des systèmes pour résoudre des problèmes qui ne se poseraient pas (ou moins) si on en restait à des solutions plus simples, plus basiques. Attention à ne pas empiler des couches ! Ceci peut induire des difficultés à contrôler ce qu'il se passe en matière de Sûreté de Fonctionnement.

J.-C. Fabre (LAAS):

Le problème existe effectivement. Le mieux est de ne pas trop avoir de couches entre le matériel et l'application, mais la journée à montré que cela est parfois nécessaire quand les offres du marché sont mal adaptées.

J.-B. Stefani (INRIA):

Sur l'utilisation de couches, deux problèmes peuvent être identifiés : Pas de caractérisation précise des interactions et des fonctionnalités. Le but est certainement de conjuguer fonctionnalité et simplicité. Une architecture n'échappe pas à la définition d'une infrastructure, incluant des couches d'abstractions. Les recherches de FTR&D et de l'INRIA ont montré qu'il est possible de descendre la flexibilité dans les couches les plus proches du matériel. Une implémentation des fonctionnalités usuellement trouvées dans les intergiciels peut être effectuée à des niveaux très bas avec des empreintes mémoire très petites. Il est important de lutter contre l'inflation des fonctionnalités au détriment de l'abstraction et de la simplicité.

J.-D. Jouffrit (TA):

Nous sommes plus intéressés par la problématique des COTS que celle des intergiciels. Si le sujet des COTS est dedans, nous sommes intéressés.

M.-L. Valentin (Airbus):

Airbus serait également intéressé par la mise en place d'un groupe de travail.

J. Arlat (LAAS):

J'en conclu donc que l'ensemble des partenaires est favorable à une poursuite.

G. Le Lann (INRIA):

Je suis frappé par la récurrence de la préoccupation de " wrappers " de composition à cause des manques observés dans les solutions du marché. Quel que soit l'instant que l'on considère le niveau de maturité des COTS disponibles (en fonctionnalité et en qualité), il est bien clair qu'un candidat ne correspondra jamais totalement aux besoins du moment

des clients qui sont toujours en avance de phase. Il faut accepter de toujours avoir à faire quelque chose en plus pour adapter la solution COTS aux besoins réels.

E. Martins (SNCF):

Je suis tout à fait d'accord sur ce point. Le système que nous avons obtenu par adaptation successives satisfait totalement la SNCF. Le problème est : Comment le maintenir et le faire évoluer ?

G. Le Lann (INRIA):

C'est un problème que je voulais aborder. Il faut effectivement dépenser beaucoup d'efforts en termes d'expérience amont pour prototyper les services émergents dont la nécessité est ressentie. Ma proposition est de mettre en commun ces efforts pour faciliter quand le besoin se présente l'adaptation des COTS du marché aux besoins du client. Aujourd'hui, il n'y pas de solution qui tienne devant les défaillances byzantines. Si ce besoin est partagé par d'autres, alors la mise en commun est très avantageuse.

J. Arlat (LAAS):

Un groupe de travail peut permettre d'identifier et d'ébaucher ces points d'effort commun.

P. Crégut (FTR&D):

France Télécom est intéressé par un groupe travail. Pour FT R&D l'aspect caractérisation est très important. Il y a aussi le problème des fonctionnalités supplémentaires. Le découpage en objets plus simples est une stratégie que nous soutenons.

J.-B. Stéfani (INRIA):

Oui, la caractérisation est un aspect important. Le dimensionnement de l'architecture aussi, notamment par la définition d'un bon cadre architectural pour maîtriser l'adaptation. Vouloir faire du temps-réel en prenant un intergiciel dit temps-réel est une « vue de l'esprit ». Nécessairement, l'implication de l'exécutif temps-réel sous-jacent est prépondérante. RT-CORBA ne résoudra rien et ne permettra jamais d'obtenir une interopérabilité quelconque.

V. Watine (THALES):

Le but raisonnable de RT-CORBA n'est pas, pour l'instant, d'offrir des propriétés temps-réel dures, mais de permettre d'intégrer un exécutif temps-réel dans un monde qui n'est pas temps-réel à travers une interface CORBA.

E. Bornschegl (ESA):

Je pense qu'il y a un intérêt certain pour un support basé outils à la capture et à l'analyse des exigences. Un petit outil qui permettrait de tracer les propagations de contraintes de réalisation suite au choix de conception, par exemple temps-réel, qui sont faits aiderait probablement à sensibiliser les utilisateurs à l'interdépendance des contraintes.

G. Le Lann (INRIA):

Tout à fait d'accord. Encore une fois, ceci nécessite une mise en commun des efforts. Il existe déjà des outils pour la capture et l'analyse des exigences.

J.-C. Fabre / J. Arlat (LAAS):

Le *RIS* peut être un catalyseur pour lancer ce type d'action.

E. Bornschegl (ESA):

Ceci faciliterait l'interface entre différents corps de métier.

J.-C. Fabre (LAAS):

Fin de la réunion. Remerciements aux orateurs pour la diversité des présentations et leur qualité. Le mot de la fin à J.Arlat.

J. Arlat (LAAS):

Merci à Jean-Charles aussi qui a déjà beaucoup investi sur ce sujet. Très bientôt, un compte-rendu sera diffusé avec copie des transparents et nous vous demanderons de le commenter, et d'apporter les précisions que vous jugerez nécessaires pour diffusion au comité d'orientation. Pour ce qui concerne les planches présentées aujourd'hui, il est possible de restreindre la diffusion, en enlevant des transparents sur des informations sensibles. Ensuite, les transparents seront rendus disponibles sur le site Web. D'ici un mois, il y aura donc diffusion du compte-rendu et nous vous donnerons des nouvelles d'ici la rentrée en ce qui concerne la suite à donner à cet atelier.

Annexes

Annexe A – Liste des participants

Société-Organisme / Nom	Email
Membres du *RIS*	
AIRBUS:	
Marie-Line Valentin	<marie-line.valentin@airbus.com>
ASTRIUM:	
Jean-Paul Blanquart	<Jean-Paul.BLANQUART@astrium-space.com>
Philippe David	<Philippe.DAVID@astrium-space.com>
Gilbert Griseri	<Gilbert.GRISERI@astrium-space.com>
Luc Planche	<Luc.PLANCHE@astrium-space.com>
Philippe Rouzet	<Philippe.ROUZET@astrium-space.com>
Eric Totel	<Eric.TOTEL@astrium-space.com>
LAAS-CNRS:	
Jean Arlat	<Jean.Arlat@laas.fr>
Jean-Charles Fabre	<Jean-Charles.Fabre@laas.fr>
Marc-Olivier Killijian	<Marco.Killijian@laas.fr>
Eric Marsden	<Eric.marsden@laas.fr>
David Powell	<David.Powell@laas.fr>
François Taiani	<Francois.Taiani@laas.fr>
TECHNICATOME:	
Jean Denis Jouffrit	<jouffrij@tecatom.fr>
THALES Group:	
Brigitte Bauer (THALES Avionique)	<Brigitte.Bauer@thales-avionics.com>
Pascal Foix (THALES Group)	<pascal.foix@thalesgroup.com>
Virginie Watine (THALES Group)	<virginie.WATINE@fr.thalesgroup.com>
Représentants d'organismes invités	
EUROPEAN SPACE AGENCY:	
Eric Bornschlegl	<Eric.Bornschlegl@esa.int>
Jean-Loup Terraillon	<Jean-Loup.Terraillon@esa.int>
FRANCE TELECOM R & D Lannion:	
Pierre Cregut	<pierre.cregut@rd.francetelecom.com>
INRIA Rhones Alpes:	
Jean-Bernard Stefani	<Jean-Bernard.Stefani@inrialpes.fr>
INRIA Rocquencourt:	
Gerard Le Lann	<Gerard.Le_Lann@inria.fr>
SNCF	
Elias Martins	<elias.martins@sncf.fr>

Annexe B – Copie des transparents