INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet SOLIDOR

Construction de systèmes et d'applications distribués

Rennes, Rocquencourt

THÈME 1B

Table des matières

1 Composition de l'équipe

Responsable scientifique

Michel Banâtre [DR Inria, UR Rennes]

Assistantes de projet

Fabienne Cuyollaa [Adjoint administratif, UR Rennes]

Sylvie Loubressac [Technicien de la recherche, UR Rocquencourt]

Personnel Inria

Gilbert Cabillic [CR, UR Rennes]

Valérie Issarny [DR, UR Rocquencourt]

Personnel CNRS

Jean-Paul Routeau [IR]

Personnel Université

Françoise André [Professeur, université de Rennes 1]

Maria-Teresa Segarra [ATER, université de Rennes 1, jusqu'au 31/08/2001]

Frédéric Weis [MC, IUT de Saint-Malo]

Personnel INSA

Isabelle Puaut [MC, INSA de Rennes, en délégation CNRS]

Ingénieurs experts Inria

Anis Ben Arbia [UR Rocquencourt, depuis 1/11/2001]

Pascal Chevochot [UR Rennes, jusqu'au 31/07/2001]

Paul Couderc [UR Rennes]

Jean-Philippe Lesot [UR Rennes]

Salam Majoul [UR Rennes, depuis le 1/10/2001]

David Mentré [UR Rocquencourt, jusqu'au 30/11/2001]

Pierre Tiako [UR Rennes, jusqu'au 15/09/2001]

Stéphane Tudoret [UR Rennes, depuis le 1/10/2001]

Grégory Watts [UR Rennes, depuis le 1/10/2001]

Apostolos Zarras [UR Rocquencourt]

Chercheurs doctorants

Nazim Boudeffa [bourse Inria, UR Rennes]

Malika Boulkenafed [bourse Inria/région IdF, UR Rocquencourt]

Djalel Chefrour [bourse Inria, UR Rennes, depuis le 1/10/2001]

Gregory Cobena [X-Télécom, co-encadrement projet Verso, UR Rocquencourt, depuis le 1/09/2000]

Antoine Colin [bourse DGA, jusqu'au 30/09/2001]

David Decotigny [bourse DGA]

Teresa Higuera [bourse Inria, UR Rocquencourt]

Christos Kloukinas [bourse Inria, UR Rocquencourt]

Frédéric Le Mouël [bourse Inria, UR Rennes, jusqu'au 30/09/2001]

Frédéric Parain [bourse Inria, UR Rennes]

Françoise Sailhan [bourse Inria/région IdF, UR Rocquencourt, depuis 1/10/2001]

Pushpendra Singh [bourse Inria, UR Rennes, depuis le 1/10/2001]

Ferda Tartanoglu [bourse Inria, UR Rocquencourt, depuis 1/10/2001]

David Touzet [bourse Inria, UR Rennes]
Arnaud Troël [bourse Inria, UR Rennes]

Autres personnels

Nicole Lévy [Collaborateur extérieur, UR Rocquencourt, depuis le 1/10/2001]
Viet Khoi Nguyen [Ingénieur associé, UR Rocquencourt]
Stéphane Tudoret [poste d'accueil jeune, UR Rennes, jusqu'au 30/09/2001]

2 Présentation et objectifs généraux

Un système distribué est un élément logiciel qui gère des ressources matérielles (calculateurs reliés par un réseau de communications), et qui offre une infrastructure au-dessus de laquelle différentes applications peuvent être bâties. La gestion des ressources doit être non seulement correcte, afin que le comportement des applications l'utilisant soit bien défini et en adéquation avec leurs spécifications, mais également performante, afin que les applications puissent par exemple offrir aux utilisateurs des temps de réponse raisonnables.

Ce sont les multiples aspects de la construction de systèmes distribués qu'étudie le projet Solidor. Les activités de recherche du projet Solidor s'articulent autour de deux axes :

- *La construction de systèmes distribués par spécification abstraite.* Cet axe est abordé au travers de la notion d'*architecture logicielle*, qui décrit de façon abstraite l'organisation d'un système. Cette description se fait en énonçant les propriétés des composants formant le système, et en exprimant les propriétés des interconnexions entre les composants qui interagissent. L'abstraction permet d'ignorer les détails de mise en œuvre du système, permettant ainsi de mieux en appréhender le comportement. De plus, l'abstraction permet d'utiliser des méthodes de spécification formelles, et ainsi de bénéficier des outils de vérification automatique de propriétés. Cet axe de recherche, dont les résultats sont exposés dans le paragraphe 6.1 est essentiellement exploré à Rocquencourt.
- *La construction de systèmes distribués embarqués et/ou mobiles.* Les applications utilisant ces systèmes ont une large gamme d'exigences selon leur domaine d'utilisation (par exemple contraintes de temps-réel, de consommation mémoire, de consommation électrique, de mode de traitement des déconnexions en environnement sans-fil). Il nous est donc apparu nécessaire d'étudier plusieurs types de systèmes, chacun capable de satisfaire les exigences d'une classe d'applications bien particulière. Cet axe de recherche, essentiellement exploré à Rennes, peut par conséquent être divisé en différents sous-axes, chacun d'eux étant dédié à un environnement matériel et applicatif différent :
  - *Support pour applications critiques (§ 6.2).* L'objectif est ici la construction d'environnements d'exécution pour applications distribuées ayant à la fois des contraintes de *temps-réel strict* (impératif de respect des échéances) et de *tolérance aux fautes*. Par ailleurs, de manière à limiter le coût d'achat et de maintenance de tels environnements, nous nous intéressons à l'utilisation d'éléments *sur étagères* (composants à vocation générale, non dédiés à un domaine d'applications particulier).
  - *Support pour applications multimédia sur ordinateurs de poche (§ 6.3).* L'objectif est ici de supporter, via un environnement Java, l'exécution d'applications ayant des contraintes de *temps-réel souples* (vidéo-conférence, vidéo à la demande). Le maté-

riel visé est l'ordinateur de poche, qui possède des *ressources limitées*, tant en ce qui concerne la capacité mémoire que l'autonomie en énergie, et est le plus souvent composé de ressources *hétérogènes*.

- *Systèmes d'information spontanés (SIS) (§ 6.4).* Cet axe traite de la conception de systèmes d'information dits "spontanés" qui se créent lorsque deux ou plusieurs ordinateurs de poche, disposant d'un système de communication "courte portée", peuvent coopérer lorsqu'ils se trouvent dans un même voisinage physique ; cette coopération s'évanouit lorsqu'ils ne se trouvent plus à portée de communication. Les principaux problèmes que nous abordons ici concernent la prise en compte de la mobilité physique des utilisateurs pour la gestion des différentes ressources. Ils traitent aussi de la découverte de l'ensemble des informations accessibles sur les calculateurs appartenant au SIS.
- *Développement d'applications en environnement mobile (§ 6.5).* Ce dernier axe concerne le développement d'applications intégrant la mobilité au dessus de systèmes de communication sans fil. Nous nous concentrons principalement sur la proposition de schémas adaptatifs qui prennent en compte la grande variation des ressources mises en jeu pour l'exécution des applications.

3 Fondements scientifiques

Mots clés : Architectures logicielles, temps-réel, tolérance aux fautes, informatique mobile, caches.

Résumé : *Le projet Solidor étudie les différentes facettes de la construction de systèmes répartis suivant les axes des architectures logicielles et des systèmes embarqués et/ou sans fil. Les thématiques de recherche qui en découlent recoupent les recherches actuelles en systèmes temps-réel, en tolérance aux fautes, en noyaux de systèmes d'exploitation et en informatique mobile. En complément, le point de vue des architectures logicielles fait appel à une méthodologie de construction de systèmes permettant de bâtir des systèmes corrects garantissant la satisfaction de contraintes applicatives.*

Nous présentons dans les paragraphes suivants un rapide panorama des recherches actuelles dans les domaines abordés par le projet.

3.1 Architectures logicielles de systèmes

Le domaine des architectures logicielles est apparu au début des années 90 et consiste à promouvoir le développement de systèmes logiciels à partir de la définition de leur architecture [SG96]. Une architecture logicielle décrit abstraitement l'organisation d'un système en terme d'interconnexion de composants caractérisant des unités de calcul ou de stockage, *via* des connecteurs caractérisant les protocoles d'interaction entre composants. Un avantage de

[SG96] M. Shaw, D. Garlan, *Software Architecture: Perspectives on an Emerging Disciplines*, Prentice Hall, 1996.

cette approche est qu'en faisant abstraction des détails de mise en œuvre du système, elle permet d'appréhender plus simplement le comportement du système. En particulier, elle rend viable l'utilisation de méthodes formelles pour la spécification du comportement de systèmes complexes, et des outils associés pour la vérification automatique de propriétés sur les architectures.

De nombreux travaux complémentaires sont actuellement en cours sur la spécification formelle du comportement d'une architecture. Nous pouvons les distinguer suivant les propriétés de l'architecture qui sont privilégiées : (*i*) les *propriétés invariantes de l'architecture* qui décrivent l'organisation du système, (*ii*) les *propriétés fonctionnelles* qui décrivent le comportement des composants, (*iii*) les *propriétés d'interaction* qui décrivent le comportement des protocoles de communication (ou connecteurs), et (*iv*) les *propriétés non-fonctionnelles* qui décrivent la gestion de ressources. Chaque type de propriété est détaillé ensuite.

Propriétés invariantes de l'architecture Ces propriétés permettent de décrire l'organisation du système. Spécifier les propriétés invariantes d'une architecture facilite l'implantation correcte d'une architecture par raffinement. La nature des propriétés spécifiées permet d'identifier la famille d'architectures à laquelle appartient celle du système que l'on construit. Cela permet la réutilisation de conceptions existantes lorsque l'on rencontre des architecture partageant certaines caractéristiques. Différentes méthodes de spécification sont utilisées, comme la logique du premier ordre, le formalisme Z, le modèle Cham ou encore les grammaires de graphe.

Propriétés fonctionnelles La spécification des propriétés fonctionnelles d'une architecture permet de vérifier la correction d'une interaction entre deux composants au regard du comportement de l'action découlant de cette interaction. Elle permet en outre de spécifier le comportement global (du point de vue des propriétés fonctionnelles) de l'architecture en fonction de celui de chacun de ses composants et de leurs interconnexions. La spécification de propriétés fonctionnelles peut aussi être exploitée pour la recherche automatique de composants à partir de leur spécification, et permet ainsi la réutilisation de logiciels. La méthode de spécification formelle utilisée dans ce cadre s'appuie sur la logique de Hoare. Remarquons ici que les propriétés spécifiées sont utilisées pour vérifier la validité de l'interconnexion de composants et non la correction d'une mise en œuvre pour une spécification donnée. Ceci rend ainsi praticable l'emploi d'outils pour une automatisation des processus de vérification ou de recherche.

Propriétés d'interaction La spécification des propriétés d'interaction d'une architecture vise les vérifications de correction de l'architecture compte tenu des protocoles de communication attendus par les composants et de ceux effectivement implantés par le système de communication utilisé. Elle permet ainsi de vérifier la validité des interconnexions de composants via les connecteurs ou encore la satisfaction de propriétés globales comme l'absence d'interblocage. L'algèbre de processus fondée sur CSP est une méthode de spécification possible. Elle permet d'utiliser l'outil FDR pour automatiser les vérifications.

Spécification de propriétés non-fonctionnelles La spécification de propriétés non-fonctionnelles a fait l'objet de très peu d'attention dans la communauté des architectures logicielles. Les quelques travaux qui abordent ce type de propriétés traitent soit les performances à l'exécution, soit la sécurité du système logiciel. Toutefois, la prise en compte des propriétés non-fonctionnelles est cruciale lorsque l'on considère la construction de systèmes (d'exploitation) distribués puisqu'il s'agit de propriétés inhérentes à ces systèmes. Aussi, les travaux de recherche du projet Solidor dans ce cadre se concentrent sur la spécification de propriétés non-fonctionnelles de différente nature. Une des méthodes de spécification proposées s'appuie sur la logique temporelle linéaire. Cette solution permet d'appréhender le raffinement d'architectures logicielles du point de vue de la réalisation de propriétés non-fonctionnelles. Par ailleurs, du fait de la forme particulière des propriétés non-fonctionnelles, nous sommes à même de fournir des outils d'aide au raffinement d'architectures et de recherche de composants implantant des fonctions de gestion de ressources pour effectivement réaliser une propriété non-fonctionnelle.

3.2 Temps-réel

De façon générale, un système temps-réel se distingue d'un système traditionnel par sa capacité à garantir que l'exécution de tâches respecte certaines échéances [Sta96]. Il existe deux grandes classes de systèmes temps-réels : les systèmes dits *temps-réel strict* et ceux dits *temps-réel souple*.

Systèmes temps-réel strict. Les systèmes dits temps-réel strict sont ceux pour lesquels le système doit *impérativement* garantir le respect des échéances fixées pour l'exécution des tâches [But97]. Respecter impérativement des échéances est typiquement une contrainte inhérente aux applications à sûreté critique, comme le sont les applications de contrôle de centrales nucléaires ou de pilotage de fusées. Une telle contrainte signifie par exemple réagir *dans un délai maximum* à la variation d'une température au sein du cœur nucléaire.

Offrir *la garantie* que les échéances de toutes les tâches d'une application seront respectées est complexe et nécessite de respecter un ensemble de contraintes lors du développement des logiciels.

Il faut d'abord procéder à l'analyse complète des logiciels, pour déterminer les lois d'arrivée des tâches (sont-elles périodiques, apériodiques ou sporadiques), leur temps d'exécution au pire-cas (WCET ou *Worst-Case Execution Time*), l'échéance de chaque tâche et les ressources à mobiliser (mémoire, accès aux disques, etc.). Des outils (e.g. outils d'analyse statique de code pour l'obtention des WCET[PB00]) peuvent être utilisés pour assister le concepteur d'applications dans cette tâche d'analyse. Notons que ces contraintes doivent s'appliquer à tous les logiciels impliqués par l'exécution de l'application, c'est à dire non seulement l'application

[Sta96] J. Stankovic, « Strategic directions in real-time and embedded systems », *ACM Computing Surveys 28*, 4, décembre 1996, p. 751–763.

[But97] G. Buttazzo, *Hard real-time computing systems: Predictable scheduling algorithms and applications*, Kluwer Academic Publishers, 1997.

[PB00] P. Puschner, A. Burns, « A Review of Worst-Case Execution-Time Analysis », *The Journal of Real-Time Systems 18*, 2, mai 2000, p. 115–128, Special issue on WCET analysis.

elle-même, mais également le système d'exploitation la supportant. Le but est de pouvoir prédire dans tous les cas de figure le comportement de toutes les parties du système (il faut donc disposer d'un système d'exploitation qui soit lui aussi *prévisible*).

Une fois l'analyse des logiciels effectuée, il est nécessaire de sélectionner un algorithme d'ordonnancement, chargé de décider de l'ordre d'exécution des tâches (par exemple EDF – *Earliest-Deadline First* – , RM[LL73] – *Rate Monotonic*). Il est alors possible d'appliquer un *test de faisabilité*. Un test de faisabilité est un test effectué hors-ligne, qui vérifie que toutes les échéances spécifiées seront respectées compte tenu de l'algorithme d'ordonnancement choisi et des propriétés sur les tâches ayant été capturées lors de la phase d'analyse. Ce test passé avec succès assure que l'application est faisable. Sinon, il faut modifier les caractéristiques de l'application (modifier les échéances par exemple), puis retester la faisabilité de l'application.

Systèmes temps-réel souple. Les contraintes temps-réel souple ont pour particularité de tolérer le non respect de certaines échéances lors de l'exécution des tâches. Cette tolérance est acceptée car les applications concernées ne relèvent pas du domaine des applications à sûreté critique. Le comportement erroné du système du point de vue des contraintes temps-réel n'a par conséquent pas d'incidence grave sur l'application. Les exemples typiques d'applications ayant des contraintes temps-réel souple sont les applications multimédias à flux de données continus. Le système doit assurer le respect de contraintes temporelles dans la délivrance des flux de données afin de garantir la qualité des images et du son. Toutefois, les contraintes imposées peuvent être adaptées puisque la qualité des données tolère une dégradation qui ne sera que faiblement perçue par l'utilisateur. Deux solutions existent pour satisfaire des contraintes temps-réel souple.

- La *réservation de ressources* où une application ayant des contraintes temps-réel effectue préalablement au lancement de l'exécution d'une tâche une réservation des ressources nécessaires à l'exécution de celle-ci pour en garantir la correction temporelle. Dans le cas où le système peut garantir la disponibilité des ressources demandées au regard de sa charge, il réserve les ressources afin que celles-ci ne soient plus utilisées pour le compte d'autres tâches et il notifie l'acceptation d'exécution à l'application. Dans le cas contraire, le système examine s'il peut toutefois exécuter la tâche en garantissant des échéances moins contraignantes. Il propose éventuellement ces échéances à l'application qui choisit ou non de lancer l'exécution de la tâche, laquelle exhibera un comportement dégradé. On parle alors de *négociation de qualité de service*.
- La *dégradation du comportement du système* compte tenu des contraintes temps-réel (plus simplement, *dégradation de qualité de service*) où le lancement de l'exécution de toute tâche ayant des contraintes temps-réel est systématique. Le système adapte ici la répartition des ressources entre les différentes tâches en fonction de sa charge globale. Dans le cas où les ressources disponibles ne permettent pas de satisfaire les contraintes temps-réel de certaines tâches, le système adapte ces contraintes de manière à pouvoir les satisfaire, ce qui conduit à un fonctionnement dégradé de l'application.

[LL73] C. Liu, J. Layland, « Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment », *Journal of the ACM 20*, 1, janvier 1973, p. 46–61.

Le choix pour l'une ou l'autre solution dépend des applications et de l'environnement d'exécution. Par exemple, une application s'appuyant sur Internet reposera sur une dégradation de la qualité de service. En revanche, un service commercial de vidéo à la demande par câble devra s'appuyer sur une réservation de ressources.

Notons enfin que pour les deux solutions préconisées et à l'instar des systèmes garantissant le respect de contraintes temps-réel strict, le système implante dans tous les cas une politique d'ordonnancement des tâches qui privilégie l'exécution des tâches ayant des contraintes temps-réel, par rapport aux exécutions des autres tâches. De même, les temps d'exécution associés aux différentes fonctions du système doivent pouvoir être bornés dans le cas d'une politique de réservation de ressources.

3.3 Tolérance aux fautes

Afin de traiter les erreurs présentes dans un système informatique, trois fonctions sont distinguées[ABC+95] : (i) la *détection d'erreurs*, dont le rôle est d'identifier les états erronés comme tels ; (ii) le *diagnostic d'erreurs*, qui permet d'estimer les dommages créés par l'erreur qui a été détectée et par les erreurs éventuellement propagées avant la détection ; (iii) le *recouvrement d'erreurs*, qui permet de substituer un état exempt d'erreur à l'état erroné.

Les formes les plus couramment utilisées de mécanismes de détection d'erreur sont les codes détecteurs d'erreur, les méthodes de réplication et comparaison, les contrôles temporels et d'exécution, les contrôles de vraisemblance et les contrôles sur les données structurées.

Les méthodes de recouvrement d'erreur reposent sur une *redondance* des données ou des calculs, par exemple par réplication de processus sur des machines différentes. Deux stratégies de réplication de processus sont utilisables :

- La *réplication active*, où chaque calcul est répliqué et exécuté simultanément sur n machines distinctes. Les copies doivent synchroniser leurs exécutions à chaque fois qu'un message est reçu ou envoyé afin d'assurer que toutes les copies réalisant un même calcul reçoivent les messages provenant d'autres processus dans le même ordre. Plusieurs modes de défaillance d'une copie existent, allant du silence sur défaillance (défaillance la plus simple) à la défaillance byzantine (défaillance la plus complexe). Dans le cas du silence sur défaillance, une copie défaillante ne produit plus aucun résultat. Disposer de n copies sur des machines distinctes permet donc d'absorber la défaillance simultanée de $n-1$ machines. Dans le cas de défaillances byzantines, une copie défaillante continue de produire des résultats mais ceux-ci sont erronés. Là, il est nécessaire de procéder à un vote sur les résultats pour élire le résultat correct produit par $\lceil (2n-1)/3 \rceil$ composants non défectueux.
- La *réplication passive*, où un composant logiciel est répliqué en n exemplaires, mais une seule des n copies effectue le calcul. Les $n-1$ autres copies sont passives et ne prennent la relève qui si la copie active est défaillante. Pour que cette stratégie fonctionne, il est nécessaire que la copie active transmette aux copies passives son état d'exécution. Cet état d'exécution est stocké par chacune des copies passives et constitue un *point de reprise*. Si

[ABC+95] J. Arlat, J. P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J. C. Fabre, H. Guillermain, M. Kaâniche, K. Kanoun, J. C. Laprie, C. Mazet, D. Powell, C. Rabéjac, P. Thévenod, *Guide de la sûreté de fonctionnement*, Cépaduès, 1995.

la copie active est défaillante, l'une des copies passives est activée et reprend l'exécution du calcul à partir du dernier point de reprise enregistré. On dit que le processus effectue un retour arrière. Recréer un état d'exécution simule une remontée dans le temps en ramenant le calcul dans l'état qu'il occupait avant la manifestation de la défaillance.

Enfin, ces deux stratégies de réplication s'accompagnent obligatoirement d'un mécanisme d'échange *atomique* d'information entre les processus, propriété indispensable à la construction d'un système tolérant aux fautes.

3.4 Informatique mobile

L'informatique mobile pose aux concepteurs de systèmes des problèmes essentiellement liés à la prise en compte de la mobilité géographique, à la variabilité des conditions de communication et à l'énergie électrique limitée dont disposent les calculateurs portables [IK96]. Résoudre ces problèmes demande d'intervenir à de nombreux niveaux (système d'exploitation, couches chargées des communications, logiciels applicatifs).

De façon générale, les travaux en informatique mobile ont pour objet de rendre compatibles les applications et systèmes conçus dans des environnements traditionnels avec les caractéristiques particulières des environnements mobiles. Une approche vise à donner aux applications l'illusion d'un environnement stable, et tente de masquer la variabilité des performances physiques [NSN+97]. Une autre approche vise à adapter le système aux variations du contexte de l'utilisateur (position géographique par exemple). Celle-ci prône le développement d'environnement d'ubiquité informatique [Wei93]. Cependant, les prototypes actuellement développés ont une échelle souvent restreinte (couvrant un bâtiment ou un campus).

Ces approches adoptent des points de vue très différents : l'une étend les fonctionnalités des systèmes statiques à des environnements dynamiques ; l'autre exploite des informations contextuelles signifiantes pour les utilisateurs. La première cache la mobilité, la seconde l'exploite.

4 Domaines d'applications

Mots clés : Systèmes embarqués, applications mobiles, avionique modulaire embarquée, ordinateurs de poche, robotique embarquée, systèmes d'information spontanés, assistance à la conduite automobile.

Résumé : *Les activités du projet Solidor s'appliquent à des domaines différents, dans lesquels les applications ont pour principales propriétés d'être distribuées et pour la plupart embarquées. Les applications relevant de chaque domaine imposent chacune des contraintes très différentes sur le système distribué (contraintes de*

[IK96] T. Imielinski, H. Korth (éditeurs), *Mobile Computing*, Kluwer Academic Press, Boston, 1996.

[NSN+97] B. Noble, M. Satyanarayanan, D. Narayanan, J. Tilton, J. Flinn, K. Walker, « Agile Application-Aware Adaptation for Mobility », *in : Sixteen ACM Symposium on Operating Systems Principles*, p. 276–287, Saint Malo, France, octobre 1997.

[Wei93] M. Weiser, « Some Computer Science Issues in Ubiquitous Computing », *Communication of the ACM 36*, 7, juillet 1993, p. 75–83.

temps de réponse strict et de tolérances aux fautes pour les applications embarquées à sûreté critique, contraintes de mobilité pour les applications utilisant des communications sans fil, contraintes de consommation électrique pour les ordinateurs de poche).

À chaque domaine présenté ci-après, et en fonction du degré d'avancement des recherches associées, correspond soit une action industrielle (stade avancé de recherches), soit une action de validation expérimentale.

4.1 Avionique modulaire embarquée

Mots clés : Avionique embarquée, sûreté critique, temps-réel strict, tolérance aux fautes..

Les systèmes actuels d'avionique embarquée doivent assurer des missions complexes et critiques dans des environnements agressifs. Les contraintes que doivent respecter ces systèmes et les objectifs qu'ils doivent atteindre imposent à leurs concepteurs de résoudre des problèmes combinés de tolérance aux fautes, de temps-réel strict, et de traitements distribués. La combinaison de ces problèmes entraîne souvent la mise au point de solutions spécifiques, ne fonctionnant que pour une application particulière. Chaque solution est généralement dotée de mécanismes de tolérance aux fautes spécifiques, et s'applique à une architecture matérielle donnée. Étant entièrement dédiée à une application, chaque solution est lourde et coûteuse.

Afin de diminuer ces coûts, l'avionique modulaire embarquée tend à faire disparaître la notion d'équipement propriétaire, développé par des équipementiers spécialisés, pour lui substituer celle de composant standard intéropérable, tant matériel que logiciel, nommé communément composant *"sur étagères"* ou en anglais COTS, pour Commercial-Off-The-Shelf. Elle tend à favoriser la réutilisabilité, l'intéropérabilité et la réduction des coûts de développement.

Les recherches dont les résultats sont présentés dans le paragraphe 6.2 s'intègrent dans ce contexte d'avionique modulaire embarquée. Elles s'intéressent à construction d'exécutifs pour des applications ayant des contraintes de temps-réel strict, de tolérance aux fautes, et de réutilisation de composants COTS.

4.2 Ordinateurs de poche

Mots clés : Java, multiprocesseurs hétérogènes, temps-réel souple, équilibrage de charge, applications multimédias.

Aujourd'hui, l'utilisation des ordinateurs de poche *(appliances)* est cantonnée à la gestion des agendas, à la prise de notes ou au traitement du courrier électronique. Demain, l'intégration du traitement de données multimédias au sein de ces ordinateurs permettra la mise en place de nouvelles applications (vidéoconférence, *vidéo à la demande*), chargées dynamiquement à partir de fournisseurs de services. Il est clair que permettre l'exécution de telles applications nécessite de disposer, en local, d'une importante puissance de traitement. Celle-ci doit cependant rester compatible avec les contraintes liées à l'embarquabilité, en particulier pour tout ce qui à trait à la consommation électrique. C'est dans ce contexte que se place notre collaboration avec *Texas Instruments* (voir les paragraphes 6.3 et 7.1.2).

L'architecture actuellement retenue repose sur un multiprocesseur hétérogène (processeurs dédiés au traitement du signal et processeurs standard). Pour tenir compte des aspects liés au chargement dynamique des applications, ainsi qu'à leur mobilité potentielle, Java a été retenu comme langage de programmation. Dans ce cadre, nos travaux seront dédiés à l'étude de solutions pour l'exécution d'applications Java au dessus des architectures multiprocesseurs hétérogènes embarquées pour lesquelles la gestion de la consommation électrique est primordiale. Nous devrons tenir compte de l'exécution concurrente d'applications Java, dont certaines gèrent des données multimédias. Ceci suppose l'étude de solutions pour la gestion du temps-réel souple au niveau de la machine d'exécution (la *Java Virtual Machine* – JVM), ainsi que la proposition d'algorithmes de placement pour une utilisation optimale des ressources processeurs en tenant compte de la nature des applications (applications de traitement de signal, applications de calcul standard).

4.3 Systèmes d'information pour ordinateurs de poche en environnement mobile

Mots clés : systèmes de communication sans fil, voisinage physique, protocole de communication, systèmes embarqués, robotique embarquée, assistance à la conduite automobile.

On peut envisager deux approches pour l'utilisation des ordinateurs de poche dotés d'interface de communication sans fil (*wireless appliances*). La première consiste à s'appuyer sur des infrastructures globales pour mener à bien toute communication. Cela nécessite l'utilisation de bornes réceptrices et émettrices, fixes, géographiquement dispersées. La présence de ces bornes, et l'existence de protocoles de communication adaptés, dissimulent aux applications les problèmes liés à la mobilité, et leur donne l'illusion que le réseau délivre un service « uniforme », quelle que soit la position géographique de l'utilisateur.

Nous étudions actuellement une autre approche pour l'utilisation de ces ordinateurs de poche, complémentaire de la précédente, en nous appuyant sur des interactions dites « de proximité ». Notre démarche est la suivante : il est possible d'établir des communications directes entre entités mobiles physiquement proches, en l'absence de toute infrastructure de communication fixe (comme le sont les bornes). En s'appuyant sur la notion de proximité physique, deux calculateurs mobiles, dotés chacun d'un module de communication, et se trouvant à portée de communication, peuvent échanger directement des messages. Dès que l'échange est possible, les deux calculateurs impliqués forment un système d'information, qui présente la particularité d'être spontané. Cette spontanéité signifie que le système d'information n'existe que par la rencontre fortuite de deux calculateurs, qu'il n'existe que tant que ces deux calculateurs sont suffisamment proches l'un de l'autre, et qu'il disparaîtra une fois les calculateurs éloignés (voir la section 6.4). Des technologies de communication sans fil telles que Bluetooth doivent permettre la mise en œuvre de tels systèmes dans un futur proche.

Afin d'illustrer le principe des Systèmes d'Information Spontanés (SIS), considérons l'exemple suivant : un groupe de personnes assiste à une conférence. Plusieurs exposés se déroulent simultanément, et les participants peuvent se déplacer librement d'une session à l'autre, en fonction des sujets abordés et de leurs centres d'intérêts. Il est envisageable que dans un futur proche,

chacune de ces personnes soit équipée avec un ordinateur de poche capable d'interactions de proximité. Nous appelons de tels ordinateurs des W-PDAs (Wireless Personal Digital Assistant, assistant personnel numérique sans fil). Un W-PDA est en mesure de stocker toutes sortes d'informations : l'identité de son possesseur, ces centres d'intérêt, une copie de sa présentation, des références bibliographiques, etc. Dans le contexte de la conférence, chaque rencontre physique entre deux personnes peut être perçue comme une opportunité d'échanger (de glaner) spontanément des données. Bien entendu, dans la mesure où les utilisateurs sont mobiles, ces informations ne pourront être transmises que pendant une période limitée, dont la durée n'est pas connue a priori. De plus, pour construire un tel système, il faut prendre en compte le fait que les utilisateurs ne se connaissent pas forcément au moment où la rencontre se produit. C'est à partir de ce type d'interactions que nous étudions les mécanismes nécessaires à la mise en œuvre des Systèmes d'Informations Spontanés.

En plus des systèmes de communication de personne à personne, les systèmes d'information spontanés sont applicables aux « systèmes à commandes » dans lesquels nous plaçons la robotique mobile et l'assistance à la conduite automobile. Dans ces deux cas, chaque robot ou chaque voiture dispose d'un calculateur doté d'un émetteur-récepteur, et chacun évolue dans un environnement où n'existe aucune infrastructure de communication. En robotique mobile, chaque robot est autonome et coopère avec d'autres robots physiquement proches pour accomplir ensemble une tâche commune. La formation spontanée d'un système d'information par la proximité de plusieurs robots élimine tout besoin de contrôle centralisé (comme le demandent les approches actuelles), et tend à favoriser la résistance aux défaillances et améliore la flexibilité du système. En automobile, le calculateur de chaque véhicule coopère avec ceux des véhicules voisins dans le but d'assurer la sécurité des personnes transportées : le système d'information spontané, créé par un petit groupe de voitures proches, s'échange des informations de vitesse, de freinage ou de direction, et substitue ainsi aux approches actuelles de détection passive et de reconnaissance de mouvements une approche active où les véhicules échangent des informations spontanément. De plus, dans le cas de l'automobile, aucune infrastructure de voirie n'est nécessaire.

Les bases de données peuvent également exploiter le principe des systèmes d'information spontanés. L'idée est la suivante : l'entrée de la base et la ressource qu'elle représente (par exemple une pièce en cours de fabrication) sont confondues. C'est le rapprochement des pièces, et donc la création d'un SIS, qui forme de manière temporaire, une base de données. L'éloignement d'une pièce entraîne bien entendu la sortie de la base de données. Nous estimons qu'une telle approche présente un réel intérêt dans les domaines de l'aéronautique ou de l'automobile, où les processus de fabrication impliquent une multitude de pièces, et où se posent d'importants problèmes de traçabilité. L'évolution future des étiquettes électroniques (en terme de capacité mémoire et capacité de communications) actuellement utilisées dans l'industrie pour marquer et suivre les pièces, nous permet de penser qu'une telle classe d'applications pourrait voir le jour.

5 Logiciels

Résumé : *Les travaux de recherche conduits dans le projet Solidor donnent lieu au développement de nombreux logiciels. Ces logiciels sont réalisés dans le cadre de partenariats avec des industriels, et par conséquent attachés aux domaines d'application couverts par le projet (voir paragraphe 4). Ces partenariats sont détaillés dans le paragraphe 7.1.*

5.1 Environnement pour l'adaptation systématique de middleware – Aster

Participants : Valérie Issarny [correspondante], Christos Kloukinas, Apostolos Zarras.

Dans le cadre de notre activité de recherche sur l'exploitation de la spécification d'architectures logicielles pour la conception, l'analyse et la construction de systèmes distribués, nous avons entrepris la réalisation d'un prototype d'environnement de développement d'applications distribuées. Précisément, cet environnement vise le développement de systèmes client-serveur à base de composants, et prend en charge la construction de *middlewares* adaptés aux besoins des applications, en termes des propriétés non-fonctionnelles fournies, à partir de la description architecturale des applications. L'approche mise en œuvre consiste à identifier une plate-forme *middleware* de base, ainsi que les services de gestion de la distribution complémentaires, nécessaires pour garantir les propriétés non-fonctionnelles requises par l'application. Par exemple, si l'on considère une application distribuée exigeant la mise en œuvre de propriétés transactionnelles, un *middleware* adapté est une implantation de l'ORB (*Object Request Broker*) de CORBA combinée avec les services de transaction et de verrouillage (*i.e.*, les services OTS –*Object Transaction Service*– et CCS –*Concurrency Control Service*). L'environnement comprend les éléments suivants :

- Un langage de description d'architectures (ou ADL pour *Architecture Description Language*) qui permet notamment de spécifier les propriétés non-fonctionnelles, requises ou fournies par les composants architecturaux.
- Un système de stockage, mémorisant l'historique des conceptions d'architectures *middleware*, incluant les architectures concrètes, mises en œuvre à partir de plate-formes *middleware* disponibles. La description des architectures ainsi mémorisées intègre en particulier la spécification des propriétés non-fonctionnelles fournies. Cette spécification est exprimée au moyen de la logique temporelle linéaire, ce qui permet d'organiser le stockage des architectures suivant une relation de raffinement entre les propriétés qu'elles fournissent, mais aussi d'offrir des fonctions pour la recherche et l'ajout systématiques d'architectures dans le système. Nous utilisons le démonstrateur de théorèmes STeP de Stanford, pour la mise en œuvre de ces différentes fonctionnalités.
- Un outil pour l'intégration automatique d'une architecture *middleware* sélectionnée, au sein d'une application, à partir de la description architecturale de cette dernière.

Notre environnement est indépendant de toute plate-forme *middleware* et peut par conséquent être exploité pour des infrastructures comme CORBA, *Enterprise Java Beans* ou encore DCOM. Toutefois, nous avons principalement centré nos expérimentations autour de l'infrastructure CORBA, ce qui nous a conduits à développer des services *middleware* particuliers mettant en œuvre des propriétés de tolérance aux fautes. L'environnement Aster fait l'objet

de divers enrichissements liés à l'expérimentation de nos résultats de recherche dans le domaine de la conception, analyse et construction de systèmes logiciels distribués à partir de leur description architecturale. Nous nous intéressons notamment à l'introduction de méthodes et outils pour la combinaison systématique d'architectures *middleware* offrant différentes propriétés non-fonctionnelles. Nous examinons par ailleurs l'extension de l'environnement Aster afin de permettre l'analyse quantitative de propriétés de fiabilité et de performance d'architectures de systèmes distribués, en nous appuyant sur des outils de modélisation existants.

5.2 Environnement d'exécution pour l'avionique embarquée – Hades

Participants : Michel Banâtre, Pascal Chevochot, Antoine Colin, David Decotigny, Isabelle Puaut [correspondante].

Les aspects contractuels liés à ce logiciel sont détaillés dans le paragraphe 7.1.1.

Les travaux décrits dans le paragraphe 6.2, visant à développer un environnement d'exécution pour applications critiques (à temps de réponse strict et en fort besoin en tolérance aux fautes) font l'objet d'une expérimentation *via* la réalisation d'un prototype. Ce prototype est nommé HADES, acronyme de *Highly Available Distributed Embedded System*.

Ce prototype comporte d'une part un environnement dédié au développement et l'analyse des applications, et d'autre part un environnement permettant l'exécution de ces applications. Les applications visées par le prototype, de par le contexte de partenariat avec la DGA et la société Dassault-Aviation (cf. § 7.1.1), sont des applications d'avionique embarquée, ayant les propriétés d'être distribuées et de comporter des contraintes de temps-réel strict et de haute disponibilité.

L'environnement de développement d'applications comprend un ensemble d'outils exécutés hors-ligne (i.e. avant exécution des applications). Ces outils incluent :

- Un environnement graphique de développement d'applications, permettant de saisir leurs tâches sous la forme de graphes orientés acycliques.
- Des tests de faisabilité permettant de tester le respect des échéances temporelles des applications en considérant le pire scénario de charge possible.
- Un outil de réplication automatique d'applications permettant de rendre les applications tolérantes aux fautes avec le minimum d'intervention de la part du concepteur d'applications [11]. Cet outil transforme les graphes des applications en répliquant (totalement ou partiellement) leurs traitements, et ceci selon différentes méthodes de réplication (réplication active, passive, semi-active et temporelle).
- Un outil d'analyse des temps d'exécution au pire-cas de programmes [1], permettant, par analyse statique de code C, de donner leur pire temps d'exécution en prenant en compte les propriétés de l'architecture pour éviter le pessimisme de l'analyse.

L'environnement d'exécution (ou support d'exécution) est une couche logicielle *middleware* qui s'appuie sur un micro-noyau temps réel sur étagères (à ce jour, le support d'exécution a été porté sur les noyaux Chorus et RTEMS). Ce support d'exécution [5] offre :

- Un ensemble de services pour l'exécution d'applications distribuées temps-réel dans un environnement sujet aux défaillances. Ces services incluent notamment un service de communication fiable temps-réel, un service d'ordonnancement des processus, un service

de gestion de groupes et un service de synchronisation d'horloges. Par ailleurs, le support d'exécution définit un ensemble de mécanismes de détection d'erreurs permettant d'assurer que chaque calculateur soit fonctionne correctement, soit s'arrête de manière silencieuse (silence sur défaillances).
- Un outillage pour l'examen du comportement du support d'exécution. En particulier, un logiciel d'injection de fautes [12] a été développé pour évaluer la couverture de l'hypothèse de silence sur défaillances. Les performances du support d'exécution ont également été évaluées à l'aide d'une application du domaine de l'avionique (conduite de tir de missiles), qui a été portée sur le système.

L'architecture logicielle du support d'exécution est conçue de manière à pouvoir remplacer tout service par un service ayant la même interface mais ayant des propriétés ou une implémentation différente, ce qui permet notamment de faciliter le portage du support d'exécution sur de nouveaux noyaux temps-réel.

Le support d'exécution a été transféré à la société Dassault-Aviation au printemps 2001, et est actuellement en cours de portage sur leur architecture.

5.3 Environnement d'exécution Java pour architecture embarquée

Participants : Michel Banâtre, Gilbert Cabillic [correspondant], Valérie Issarny, Frédéric Parain, Teresa Higuera, Jean-philippe Lesot, Salam Majoul, Jean-Paul Routeau, Pierre Tiako.

Le contrat correspondant est détaillé dans le paragraphe 7.1.2.

L'objectif de ces développements est de concevoir et réaliser un environnement d'exécution Java qui puisse prendre en compte les contraintes d'embarquabilité liées à un *appliance* (i.e. ordinateur de poche disposant d'un moyen de communication sans fil). Ce type de matériel est plus particulièrement caractérisé par le fait d'une capacité mémoire limitée et d'une autonomie d'énergie limitée. Vu de l'utilisateur, un appliance permet l'exécution d'applications différentes telles qu'un agenda, un accès Internet, de la téléphonie, mais aussi des applications multimédias (reconnaissance vocale, lecture de sons ou de séquences vidéos diffusées, vidéoconférence, etc).

Cet environnement permet l'exécution de différents types d'applications, en offrant un ensemble d'APIs Java tout en tenant compte des contraintes d'embarquabilité d'un appliance. En outre, il est conçu pour s'exécuter sur différents types de systèmes en définissant une couche d'abstraction du matériel qui permette de le porter d'un système à l'autre. Ainsi, le système temps-réel VxWorks de la société WindRiver est supporté, mais également des systèmes d'exploitation plus généralistes comme Windows ou Linux.

Enfin, les aspects liés à la performance de cet environnement d'exécution sont traités. Notre approche est d'avoir décomposé en modules l'ensemble de l'environnement d'exécution Java. Cette décomposition modulaire (voir [4] pour les détails) permet tout d'abord de comprendre les problèmes réels de performance d'un module et ensuite de pouvoir se concentrer sur les améliorations à apporter sur ce module. Afin de réaliser cette modularité, nous avons été amenés à écrire la totalité de notre environnement d'exécution Java.

5.4 Environnement pour la mise en oeuvre des systèmes d'information spontanés

Participants : Michel Banâtre, Gilbert Cabillic, Paul Couderc, David Touzet, Arnaud Troël, Stéphane Tudoret, Grégory Watts, Frédéric Weis [correspondant].

Le contrat correspondant est détaillé dans le paragraphe 7.1.3.

L'objectif de ces développements est de mettre en œuvre une plate-forme matérielle et logicielle permettant d'expérimenter les mécanismes propres aux SIS, notamment la gestion efficace des interactions de proximité, la construction dynamique de l'espace visible au sein du SIS, les interactions spontanés au sein de cette espace visible. L'ensemble de ces mécanismes est décrit dans [30].

La plate-forme matérielle est constituée de PDAs fonctionnant sous Windows CE, et disposant de cartes de communication sans fil 802.11b. Cette configuration s'enrichira à terme de cartes Bluetooth, apte à mettre en œuvre le principe de communication par voisinage spécifique aux SIS (elles offrent une portée de communication de l'ordre d'une dizaine de mètres). Les modules prenant en charge les mécanismes de base des SIS (décrit dans le paragraphe 6.4) ont été réalisés et expérimentés : détection de présence des nœuds voisins dans le voisinage physique, maintien dynamique de la liste des nœuds (en fonction des entrées et des sorties) présents dans le SIS, construction et présentation de l'espace visible à l'utilisateur.

Enfin, nous expérimentons sur cette plate-forme les interactions spontanés entre nœuds voisins présentés dans le paragraphe 6.4.3 [10].

6 Résultats nouveaux

Résumé : *Ce paragraphe présente les résultats des recherches menées par le projet Solidor au cours de l'année 2001. Ces recherches correspondent aux activités ayant été brièvement introduites dans le paragraphe 2.*

6.1 Architectures logicielles de systèmes distribués

Participants : Anis Ben Arbia, Malika Boulkenafed, Valérie Issarny, Christos Kloukinas, Viet Khoi Nguyen, David Mentré, Nicole Lévy, Françoise Sailhan, Ferda Tartanoglu, Apostolos Zarras.

Comme indiqué dans la section 3.1, la description de l'architecture d'un système logiciel permet de s'abstraire des détails d'implémentation de bas niveau et ainsi de mieux appréhender la conception et la construction de systèmes logiciels complexes. En particulier, une telle approche à la mise en œuvre de systèmes distribués rend possible l'utilisation de méthodes formelles pour l'analyse et la conception de ces systèmes.

Dans le cadre des activités de recherche du projet SOLIDOR s'effectuant à l'UR de Rocquencourt, nous étudions l'exploitation de la description d'architectures logicielles (*i.e.*, définition de langages, et de méthodes et outils associés) pour mécaniser l'analyse, la conception et la construction de systèmes distribués, où nous sommes en particulier intéressés par la réalisation des propriétés non fonctionnelles.

Nous étudions également la définition de nouvelles architectures logicielles de systèmes distribués intégrant des sites mobiles. Les architectures des systèmes distribués visés doivent en particulier permettre d'offrir une qualité de service acceptable aux utilisateurs tout en tenant compte des ressources relativement limitées de certains des terminaux utilisateurs, de la variation de la bande passante disponible, et de la charge globale du système.

Nous donnons ci-après un récapitulatif de nos résultats de recherche dans le domaine des architectures logicielles de systèmes distribués. Indiquons par ailleurs que les activités du projet Solidor à l'UR de Rocquencourt ont fait l'objet de la proposition de création du projet ARLES (*ARchitectures Logicielles Et Systèmes*). Le projet ARLES vise l'étude de solutions au développement orienté architecture de systèmes distribués, qui permettent une composition de systèmes offrant des garanties de qualité (ou propriétés extra-fonctionnelles) aux utilisateurs, et s'intéresse plus particulièrement aux systèmes contribuant à la réalisation de la notion d'*ubiquité informatique* (ou *intelligence ambiante*).

6.1.1 Présentation

La description d'architectures logicielles de systèmes repose sur trois éléments principaux : (*i*) les composants caractérisant abstraitement les unités de calcul ou de stockage du système, (*ii*) les connecteurs caractérisant abstraitement les protocoles d'interaction entre composants et (*iii*) les configurations décrivant les structures de (sous-)systèmes en termes de compositions d'instances de composants *via* des instances de connecteurs. Les notations utilisées pour les spécifications de ces différents éléments dépendent alors de l'exploitation visée de l'architecture (*e.g.*, analyse, construction). L'essentiel est de maintenir un niveau d'abstraction élevé afin de faciliter l'appréhension du comportement du système à partir de la description de son architecture. En général, la description d'architectures est abordée d'un point de vue fonctionnel, mettant en avant les fonctionnalités des composants du système et les protocoles d'interaction exploités pour leur composition. Toutefois, la mise en œuvre de propriétés non-fonctionnelles est tout aussi fondamentale. Une caractéristique intéressante de ces dernières propriétés est qu'elles peuvent souvent être implantées en réutilisant des "services systèmes" existants. Il est donc possible de considérer une spécification abstraite des propriétés non-fonctionnelles, pouvant être exploitées pour mécaniser l'intégration des services nécessaires à leur satisfaction, au sein des architectures d'applications. Cette problématique motive un certain nombre de nos travaux. Nos travaux au cours de l'année 2001 ont plus spécifiquement porté sur les points suivants :

- Conception et mise en œuvre d'un environnement de développement orienté architecture, pour assister la conception, l'analyse et la construction de systèmes distribués à partir de leur description architecturale.
- Etude de solutions à la composition de systèmes au niveau architectural.
- Conception d'architectures de systèmes distribués pour utilisateurs mobiles et mise en œuvre de prototypes associés.

Nous précisons ci-après les résultats que nous avons obtenus.

6.1.2 Environnement de développement orienté architecture

Les résultats du domaine des architectures logicielles sont pour la plupart complémentaires. Ils se traduisent par des ADL (*Architecture Description Language*) qui reposent majoritairement sur la définition de composants, de connecteurs et de configurations. Ces définitions diffèrent en fonction de l'exploitation visée de l'ADL, laquelle peut relever de l'aide à l'analyse du comportement du système ou à la construction de ce dernier. Il est par conséquent intéressant de fournir un environnement de développement s'appuyant non pas sur un unique ADL mais sur un ADL spécialisable, où chaque spécialisation assiste une tâche du processus de développement. C'est dans ce contexte que s'inscrit notre activité de recherche sur la conception et réalisation d'un environnement de développement orienté architecture. La définition d'un ADL spécialisable, visant à intégrer dans un même environnement les différentes solutions apportées dans le domaine des architectures logicielles, a déjà été abordée dans la littérature. Pour la définition de l'ADL de base de notre environnement de développement, nous nous appuyons sur les exigences suivantes :

- La spécialisation de l'ADL, ce qui conduit à n'introduire que les notions élémentaires à la base de la définition d'une architecture, à savoir la définition de la structure du système en termes de composants, de connecteurs et de configurations, sans contraindre le style architectural.
- La définition d'un ADL fondé sur un langage standard, de manière à garantir une utilisation effective de notre environnement mais aussi la disponibilité d'une spécification adéquate des services disponibles, sujets à composition.

Au regard de ce qui précède, nous avons défini un ADL minimal reposant sur la définition des éléments structurels du système, et fondé sur le langage UML (*Unified Modeling Language*)[1]. [25]. Nous avons retenu UML car son utilisation se généralise, il permet d'exploiter des outils d'aide au développement existants et une traduction en XML est disponible. L'inconvénient d'UML est l'imprécision de sa sémantique, ce qui devrait être en partie résolu à terme au regard des travaux en cours. Ceci n'a par ailleurs qu'une incidence mineure sur la définition précise de notre ADL puisque ce dernier fait l'objet d'une définition propre et n'utilise que peu d'éléments d'UML. Le problème se pose toutefois dans le cas du raffinement du modèle architectural en termes de modèles UML traditionnels dans un souci d'implémentation, ce qui n'est pas une utilisation première visée pour notre environnement puisque nous sommes principalement intéressés par le développement de systèmes distribués, construits par composition de services existants. Nous avons intégré le profil UML correspondant à notre ADL dans un environnement existant[2], que nous avons en outre enrichi d'un analyseur de contraintes OCL (*Object Constraint Language*), implémenté dans le langage OCAML. L'analyseur OCL permet de vérifier la correction syntaxique des descriptions d'architectures, au regard des contraintes OCL associées aux éléments architecturaux.

[1] OMG, UML Semantics, `http://www.omg.org`

[2] L'environnement que nous enrichissons est Rational Rose que nous avons principalement utilisé pour des raisons pragmatiques, la principale étant la disponibilité d'une licence site ; les enrichissement que nous apportons pourraient être intégrés dans tout autre environnement puisque nos outils traitent des fichiers XMI (définition standardisée de la traduction de modèles UML en XML).

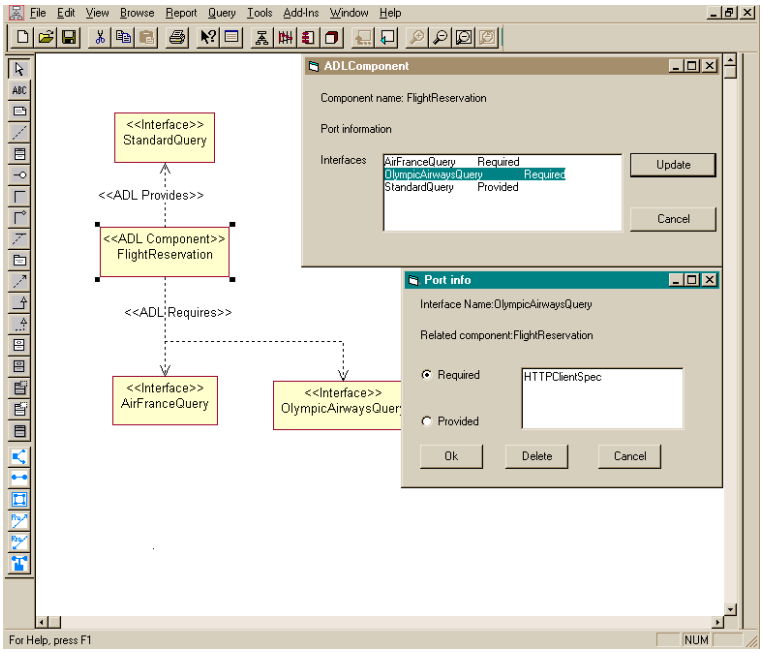
A titre d'illustration, la figure 1 donne une définition de composant et de connecteur au moyen de notre ADL, dans le cadre d'un système offrant les fonctionnalités d'une agence de voyages, à partir de la composition de services Web existants. Le composant *FlightReservation* offre des fonctions de réservation de vols à partir de services Web de compagnies aériennes. Ce composant requiert ainsi les interfaces de ces derniers services (voir la définition associée à *Interfaces*), avec lesquels il interagit *via* le protocole HTTP, en tant que client (voir définition de *Port* donnée informellement par *HTTPClientSpec*). Le connecteur *HTTP* abstrait un connecteur multi-partie basé sur le protocole HTTP. Ce connecteur comprend notamment le rôle (défini ici de manière informelle par *HTTPServerSpec*) implantant la partie serveur du protocole pour ce qui relève des interactions avec les services des compagnies aériennes. Le comportement du protocole est par ailleurs défini informellement par *HTTPProtocolSpec*. Cette définition de l'ADL s'apparente naturellement à celle de l'ADL Wright du CMU si l'on considère une substitution des définitions informelles par des processus CSP. Nous avons proposé une spécialisation dans ce sens de notre ADL, en nous appuyant sur l'outil SPIN.

Etant donnée la définition de l'ADL de base de notre environnement, notre objectif est d'en offrir différentes spécialisations pour supporter la mécanisation de la conception, de l'analyse et de la construction des systèmes distribués, en nous concentrant sur la qualité des systèmes produits. Suite à l'expérience des membres du projet dans ce domaine, au travers de l'activité ASTER, nous avons déjà proposé des solutions à l'analyse qualitative (au moyen de la vérification de modèles - *model checking*) et quantitative (du point de vue de la fiabilité et de la performance) [26]. Celles-ci reposent sur la spécialisation de la définition des éléments architecturaux par l'intégration d'attributs de qualité ne nécessitant pas de connaissance de modèles formels par les développeurs[3]. Les descriptions d'architectures obtenues à partir de ces spécialisations de l'ADL sont ensuite automatiquement traduites en des modèles formels, traités par des outils d'analyse existants. Nous implémentons actuellement ces solutions au sein de notre environnement.

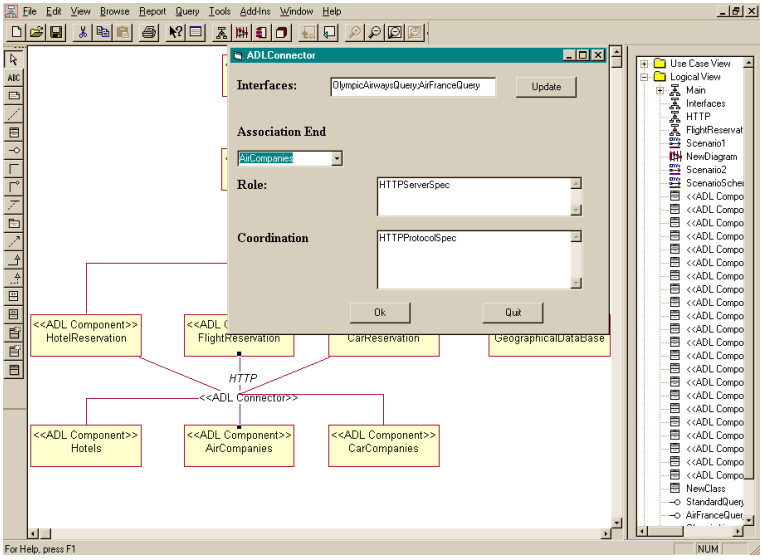
6.1.3 Composition de systèmes

La composition de systèmes a trait à la mise en œuvre de systèmes distribués à partir de services (applicatifs et exécutifs) existants. Cette problématique devient prépondérante dans le développement des systèmes distribués actuels de par le souci croissant et parfois la nécessité de réutiliser des composants du commmerce (COTS) et des systèmes legs. La mécanisation de la composition va en outre devenir un élément clé dans le développement des futurs systèmes distribués lorsque l'on considère le domaine d'application de l'intelligence ambiante qui requiert une composition dynamique des systèmes au regard de l'environnement spécifique de l'utilisateur (profil, localisation, environnement d'exécution, *etc.*). Dans ce cadre, nous nous intéressons à l'étude de solutions à la mécanisation de la composition de systèmes à partir de leur description architecturale. Au cours de l'année 2001, nous avons plus spécifiquement

[3]Cette caractéristique qui découle de la nécessité d'une utilisation effective de notre approche pour la description d'architectures de systèmes et la spécification du comportement des services n'est qu'en partie satisfaite dans le cas de l'analyse de modèles. Dans ce dernier cas, le développeur doit fournir une spécification de processus caractérisant le comportement des composants et des connecteurs. Nous avons cependant choisi un outil (SPIN) dont le langage (PROMELA) s'apparente à un langage de programmation traditionnel et qui est ainsi plus facile d'utilisation.



a) Une définition de composant



b) Une définition de connecteur

FIG. 1 – Définition de composant et connecteur

examiné la composition d'architectures *middleware* et la composition de systèmes du point de vue de la sûreté de fonctionnement en résultant.

Composition d'architectures middleware. Les systèmes informatiques deviennent de plus en plus plus complexes et doivent fournir un nombre toujours croissant de propriétés non fonctionnelles. La réalisation de ces propriétés s'appuie en général sur une infrastructure *middleware*. Cependant, le problème de concevoir une architecture *middleware* spécialisée pour un système donné reste posé, et devient vite complexe lorsque différentes propriétés fonctionnelles doivent être offertes. En général, l'architecte s'appuie sur son expérience pour composer l'architecture *middleware* adaptée à son système, étant donnés les différents composants offerts par l'infrastructure. Le problème avec cette approche est que les compositions originales, qui répondent à un besoin spécifique mais sont peu communes, ne sont jamais utilisées, puisque les architectes tendent à produire des solutions fondées sur les systèmes qu'ils connaissent. De plus, différentes architectures *middleware* sont éligibles pour offrir un ensemble de propriétés non fonctionnelles données. Aussi, il devient vite difficile de considérer ces différentes options sans outils d'aide à la conception. Au regard de ce qui précède, nous avons étudié une solution à la mécanisation de la composition d'architectures *middleware*, étant données des architectures existantes connues. Nous avons ainsi formulé ce problème de composition en terme d'analyse de modèles, ce qui nous permet d'exploiter un outil existant (SPIN) pour dériver les compositions valides d'architectures *middleware* [19]. Cette solution est en outre complétée par une solution à la composition au niveau structurel qui est immédiate de par la composition au niveau architectural. Cette dernière solution permet de réduire l'espace d'états à analyser. En conséquence, nous fournissons à l'architecte le moyen d'identifier simplement, et ce dès le début du processus de développement, les différentes compositions d'architectures *middleware* répondant à son problème. Cette solution offre également une aide aux architectes d'infrastructures *middleware*, qui peuvent évaluer la pertinence des évolutions envisagées.

Composition de systèmes et sûreté de fonctionnement. La sûreté de fonctionnement d'un système est une propriété clé de tout système et doit être prise en compte dans son développement. Dans ce contexte, nous nous intéressons à la sûreté de fonctionnement des systèmes composés, en nous concentrant dans un premier temps sur l'exploitation de techniques de tolérance aux fautes. Lors de la composition de systèmes, l'utilisation de techniques de tolérance aux fautes peut être soit masquée au niveau des éléments de l'architecture, soit implantée au niveau architectural dans le cas où la configuration du système se voit affectée. Le premier cas relève en grande partie du développement de l'architecture *middleware* et rejoint la problématique du paragraphe précédent. En collaboration avec Jean-Pierre Banâtre, nous avons débuté une étude sur le premier point. Nous avons proposé une solution au traitement d'exceptions au niveau architectural. Cette proposition complémente le traitement d'exceptions interne aux éléments de l'architecture, et facilite la réutilisation de composants logiciel [18]. Plus généralement, nous nous intéressons au problème de la mise en œuvre de solutions de tolérance aux fautes, spécifiques aux applications, au niveau architectural. Dans ce cadre, nous examinons la définition d'un style architectural pour le développement de systèmes distribués par composition de systèmes qui soient tolérants aux fautes. Cette étude, menée en collaboration avec l'Université de Newcastle, vise l'aide à la composition du système du point de vue de la

réalisation de la tolérance aux fautes, en offrant notamment un langage de description d'architectures dédié, et des méthodes et outils associés, pour automatiser une partie du processus de composition/intégration (*e.g.*, génération de *wrappers*).

6.1.4 Architectures de systèmes distribués pour utilisateurs mobiles

Les évolutions en matière de réseaux sans fil et d'ordinateurs de petite taille, permettent *a priori* à tout utilisateur d'accéder à des services numériques en tout lieu. La figure 2 donne un aperçu des interactions possibles, à terme, depuis un même site mobile où les ronds représentent des sites mobiles, et le cercle la portée du WLAN par rapport au site mobile qui se trouve en son centre. Le choix de l'une ou l'autre de ces interactions dépend de plusieurs facteurs, qui outre la fonctionnalité visée, relèvent de la consommation de ressources (en particulier énergétique), du temps de réponse, et de la sûreté de fonctionnement induits par telle ou telle interaction. L'objectif de nos travaux est de mieux comprendre ces critères par l'étude de services exécutifs supportant la réalisation de systèmes distribués dans un tel environnement. Au cours de cette année, nous nous sommes plus spécifiquement concentrés sur la gestion de l'accès aux données.

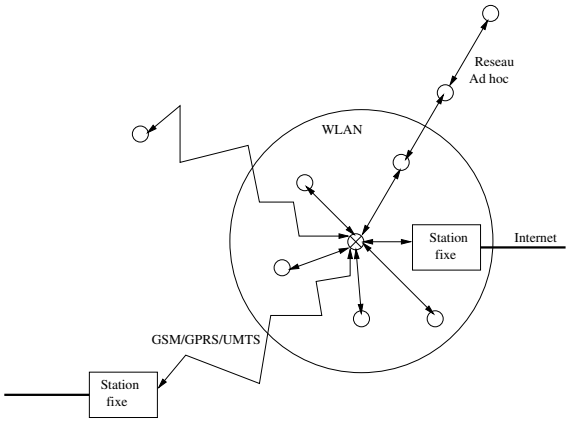


FIG. 2 – Interactions en environnement mobile

Dans l'étude de solutions à l'accès aux donnée depuis des terminaux mobiles, notre objectif est de permettre à un utilisateur d'accéder à ses données depuis tout terminal mobile, en tout lieu, sans requérir une connexion systématique à un site distant spécifique. Notre approche pour offrir une garantie de sûreté de fonctionnement et de performance lors de l'accès aux données repose sur l'emploi du *caching*. Les données deviennent accessibles, non seulement depuis les sites les hébergeant, mais également depuis les sites dont les utilisateurs ont des centres d'intérêt commun, ou encore des sites au sein desquels les données peuvent être stockées de manière sûre. Nous considérons ainsi que le sites (mobiles ou non) implantent des caches de données des serveurs. Nous distinguons deux cas suivant que les données sont privées ou publiques.

Accès aux données privées. Pour un accès simplifié aux données privées tout en offrant des garanties de sûreté de fonctionnement et de performance, nous examinons la conception

et l'implantation d'un système de gestion de fichiers distribués, sans serveur, où l'accès à un fichier se traduit de manière transparente par un accès au site *le plus proche* stockant le fichier [21]. Le critère de proximité, qui reste à définir précisément, est notamment évalué en termes de temps de réponse et de consommation énergétique. La conception de ce système débouchera sur un ensemble de services exécutifs, réutilisables pour le développement de divers systèmes distribués intégrant des sites mobiles. Nous étudions plus spécifiquement les aspects suivants :

- *Caching et partage de données dans un WLAN :* Il s'agit ici de considérer le partage de fichiers entre des sites mobiles pouvant interagir directement *via* le WLAN. Nous sommes typiquement dans le cas de la coopération entre utilisateurs ayant accès aux mêmes fichiers. L'intérêt de cette solution est qu'elle conduit *a priori* à une optimisation du temps de réponse et, de manière générale, à une optimisation de la consommation des ressources. Toutefois, cette approche requiert des solutions relatives à la gestion de groupes de sites mobiles, de la cohérence des données et de la sûreté de fonctionnement, comme précisé ci-après, ainsi qu'un travail d'expérimentation conséquent pour sa validation.
- *Groupes de sites mobiles :* Différents protocoles ont été proposés pour permettre à des sites mobiles d'interagir directement lorsqu'ils se trouvent à portée l'un de l'autre. Nous nous intéressons à la création dynamique de groupes de sites mobiles, à partir de ces protocoles, les groupes étant définis en fonction de l'application. Dans un premier temps, il s'agit de définir des groupes de sites au regard des fichiers qu'ils sont susceptibles de partager, identifiés par l'appartenance à des *domaines de sécurité* communs.
- *Cohérence des données :* La gestion de la cohérence des données réparties sur des sites mobiles a fait l'objet de différents travaux depuis le début des années 90. Ceux-ci sont principalement centrés sur le traitement des déconnexions, abordant le préchargement de données et leur réconciliation. Notre intérêt se porte plus spécifiquement sur la gestion de la cohérence de données au sein d'un groupe de sites mobiles, l'objectif étant d'examiner une solution optimale en terme de consommation de ressources induite.
- *Sécurité :* La sécurité est un problème incontournable dans le contexte de la mobilité. Nous nous intéressons au problème du partage sécurisé des données, et plus spécifiquement à des solutions minimisant la consommation de ressources sur les sites mobiles.
- *Disponibilité et fiabilité :* Les sources de défaillances en environnement mobile sont nombreuses, du fait des déconnexions volontaires ou non, des sites. Nous examinons la définition de mécanismes de tolérance aux fautes pour ce contexte particulier, de manière à accroître la disponibilité et la fiabilité, tout en optimisant la consommation de ressources induite.
- *Interaction via les différents réseaux :* Comme schématisé dans la figure 2, un site mobile peut interagir avec un autre site *via* différents réseaux de communication. Nous nous intéressons à une gestion optimisée de l'accès aux données *via* ces différents réseaux.

La conception du système de gestion de fichiers distribués aux caractéristiques énoncés ci-avant, a débuté au cours de l'année 2001, et un premier prototype centrée sur l'échange de données sécurisé entre sites mobiles interagissant *via* un WLAN, devrait être disponible début 2002 (voir partie inférieure de la figure 3). Ce prototype, implémenté en OCAML, gère le partage de données entre sites mobiles pouvant interagir *via* IEEE 802.11, la cohérence des données partagées par de tels sites, ainsi que la sécurisation des interactions [21]. Ce prototype sera ensuite enrichi par l'intégration de nos solutions aux problèmes énoncés précédemment.

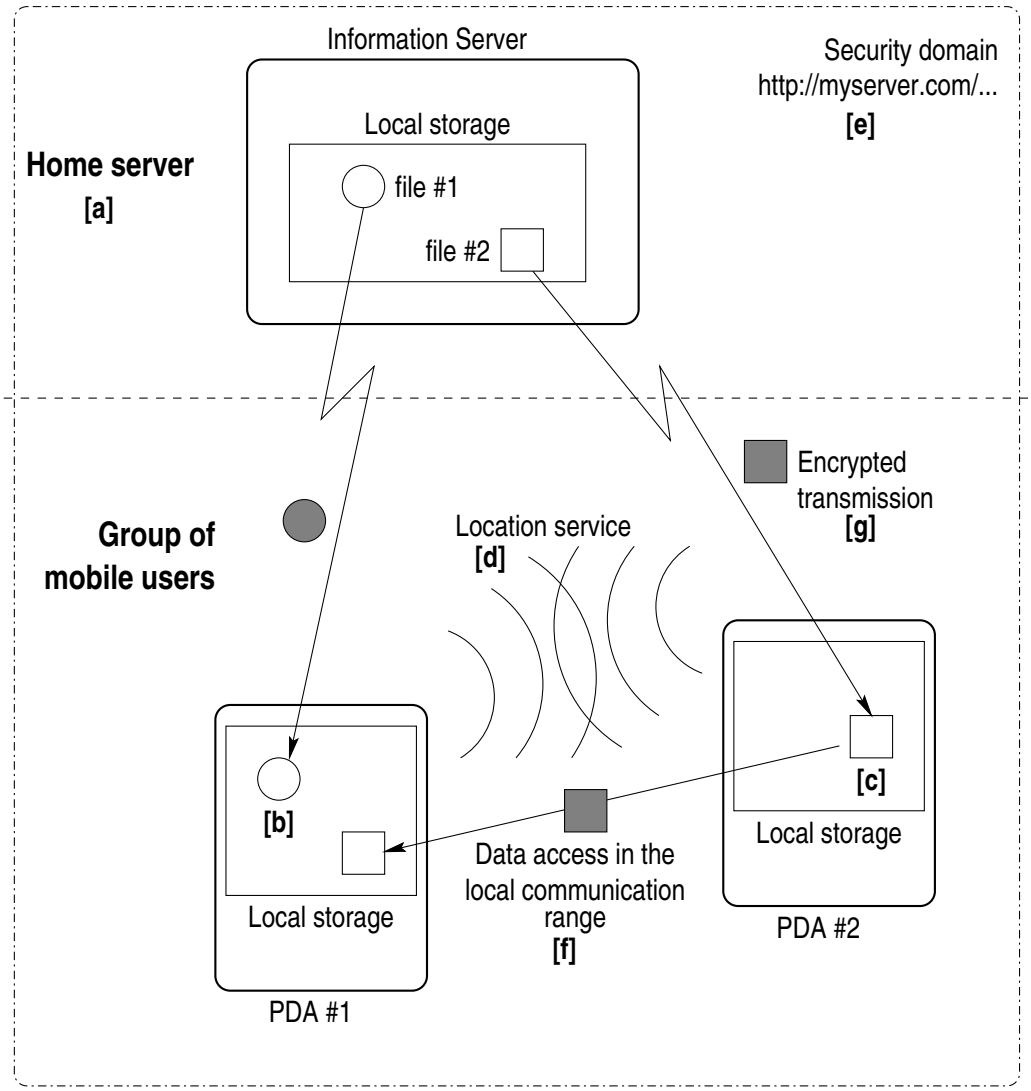


FIG. 3 – Partage de fichiers dans un WLAN

Accès aux données publiques. Pour la gestion de l'accès à des données publiques depuis un site mobile, nous nous concentrons sur un type de données particulier, à savoir les données du Web qui sont parmi les plus populaires. Le *caching* de données dans le Web a fait l'objet de nombreux travaux, du fait de son intérêt majeur pour diminuer les temps de réponse pour les utilisateurs. Le *caching* de données est généralement couplé à un préchargement, qui s'il conduit à un accroissement de la consommation de la bande passante, peut s'avérer bénéfique pour diminuer les temps de réponse. Nous nous intéressons ici à étudier le bénéfice potentiel du *caching* de données réparti sur des sites mobiles. Nous avons débuté une étude sur le sujet dans le courant de l'année 2001, en considérant la recherche de pages Web depuis des sites mobiles accessibles *via* un réseau *ad hoc* (nous avons retenu le protocole ZRP dans un premier temps), plutôt que suivant une approche classique s'appuyant sur un accès depuis un serveur *proxy* fixe. Il s'agit ici d'examiner le gain potentiel en termes de temps de réponse, de disponibilité et de consommation de ressources. A terme, nous comptons coupler le résultat de ces travaux, avec ceux relatifs à la gestion de l'accès aux données privées.

6.2 Systèmes temps-réel hautement disponibles

Participants : Nazim Boudeffa, Michel Banâtre, Pascal Chevochot, Antoine Colin, David Decotigny, Isabelle Puaut.

Dans le cadre de ces travaux nous nous intéressons ici à la construction de systèmes ayant à la fois des contraintes de *temps-réel* (strict ou souple) et de *tolérance aux fautes*. Nos travaux sur ce thème s'articulent autour de deux axes, qui sont la construction de *supports d'exécution* adaptés aux applications temps-réel à haute disponibilité, et la conception de méthodes et outils pour *valider leur comportement temporel*. Les résultats obtenus pendant l'année 2001 selon ces deux axes sont donnés ci-dessous.

6.2.1 Support d'exécution hautement disponible pour applications temps-réel strict

Les supports d'exécution adaptés aux applications temps-réel strict hautement disponibles, doivent non seulement réagir en temps borné et connu (temps-réel strict) mais également intégrer des mécanismes de tolérance aux fautes (haute disponiblité). Par ailleurs, de manière à limiter le coût d'achat et de maintenance de tels environnements, de nombreux indstriels (dans l'avionique, l'automobile) s'intéressent à l'utilisation d'éléments, tant matériels que logiciels, sur étagères (en anglais COTS, pour Commercial-Off-The-Shelf), c'est à dire des composants à vocation générale, non dédiés à un domaine d'applications particulier.

C'est dans cet esprit de cohabitation entre temps-réel strict, tolérance aux fautes et utilisation d'éléments COTS qu'a été conçu le support d'exécution HADES (acronyme de *Highly Available Distributed Embedded System*). Sa conception, débutée en 1997, a été soutenue par la DGA et la société Dassault-Aviation (cf.§ 7.1.1). Deux voies ont été explorées pour mettre en place le support d'exécution Hades :

- Définition des fonctionnalités à intégrer dans le support d'exécution, de manière à ce qu'il ait un comportement bien défini, et ce même en présence de fautes. Ainsi, nous avons conçu un ensemble de protocoles (synchronisation d'horloges, gestion de groupes,

diffusion multipoint), et de mécanismes (détection d'erreurs). Ces protocoles et mécanismes [5] permettent au support d'exécution de vérifier un ensemble de propriétés : silence sur défaillance des calculateurs, diffusion fiable en temps borné, détection des arrêts de calculateurs en temps borné. Ils ont été intégrés dans un intergiciel (middleware) construit au dessus d'un noyau de système d'exploitation temps-réel. Nous ne détaillons pas plus avant ce support d'exécution, ce dernier étant présenté dans le paragraphe 5.2.
- Analyse d'ordonnançabilité [11], permettant de vérifier que toutes les échéances des tâches seront respectées, et ce même en présence de fautes. Cette analyse s'appuie sur les propriétés assurées par le support d'exécution.

Un prototype du support d'exécution a été finalisé au printemps 2001 et a été transféré chez Dassault-Aviation, où il est actuellement en cours de portage. Nos travaux sur le support d'exécution en 2001 ont essentiellement porté sur une l'évaluation des performances du support d'exécution, et ce de deux points de vue :
- Comportement du support d'exécution en présence de fautes [12]. Nous avons à cet effet mis au point un outil d'injection de fautes qui nous a notamment permis d'évaluer la couverture de l'hypothèse de silence sur défaillance des calculateurs.
- Comportement temporel du support d'exécution [5], en terme de performances au pire-cas des protocoles qu'il intègre et de consommation de temps CPU.

Cette évaluation de performances nous a permis de quantifier le bien fondé d'intégration d'éléments COTS dans des supports d'exécution temps-réel hautements disponibles. Elle est présentée dans l'article [5], qui effectue la synthèse des résultats acquis lors de la conception du support d'exécution Hades.

6.2.2 Validation du comportement temporel de systèmes temps-réel

Dans les systèmes temps-réel *strict*, le non respect d'une échéance peut avoir des conséquences humaines, financières ou écologiques graves. Il est donc fortement recommandé pour de tels systèmes d'utiliser des méthodes d'analyse d'ordonnançabilité, qui permettent de vérifier hors-ligne que les tâches du système respecteront leurs échéances, et ce même dans les pires conditions d'exécution du système. Une entrée clé des méthodes d'analyse d'ordonnançabilité est le *pire temps d'exécution* d'un programme considéré de manière isolée, nommé communément WCET, pour *Worst-Case Execution Time*. Les principaux résultats obtenus dans l'année dans l'objectif de vérifier l'ordonnançabilité de systèmes temps-réel strict, portent sur la définition de méthodes d'obtention de temps d'exécution au pire-cas de logiciels, que ces méthodes opèrent par analyse statique du code source des programmes, ou par leur exécution sur l'architecture cible. Nous avons également travaillé sur l'approche alternative aux méthodes d'analyse d'ordonnançabilité, par simulation du système temps-réel.

Obtention de temps d'exécution au pire-cas par analyse statique de programmes. Nous nous intéressons ici à l'obtention de temps d'exécution de WCET par analyse statique de leur code source. Une des difficultés si l'on utilise ce type de méthode est d'éviter l'obtention de WCET qui soient une estimation trop pessimiste des temps d'exécution effectifs, ce qui entraînerait alors une sur-estimation des ressources matérielles nécessaires à l'exécution des programmes, et donc une sous-utilisation de ces ressources lors de l'exécution.

Nos contributions dans le domaine de l'obtention de WCET par analyse statique résident à la fois dans la définition d'une *méthode* d'obtention de WCET sûrs et précis, et dans la développement d'un *outil* mettant en œuvre cette méthode.

La méthode proposée [1] est une méthode à base d'arbre (*tree-based*). Elle consiste à calculer le WCET d'un programme en effectuant un parcours hiérarchique de bas en haut de son arbre syntaxique, le WCET d'un nœud de l'arbre étant utilisé pour calculer le WCET de son père. Les apports de la méthode sont de réduire les deux principales sources de pessimisme de l'analyse :

- La première source de pessimisme est la difficulté d'identifier précisément le pire chemin d'exécution dans le programme, en particulier le nombre maximum d'itérations des boucles. Nous proposons à cet effet une méthode d'annotation des boucles, qui permet de décrire de manière précise le comportement des boucles imbriquées dites "non-rectangulaires" [1].
- La deuxième cause du pessimisme est due aux éléments d'architectures, qui améliorent les performances moyennes du processeur, mais posent des difficultés pour identifier de manière sûre mais pas trop pessimiste son comportement temporel au pire cas. Nous proposons des méthodes de prise en compte pour trois de ces éléments (le cache d'instructions, la prédiction de branchement et le pipeline), et adaptons la méthode de calcul des WCET par parcours de l'arbre syntaxique du programme de manière à prendre en compte de manière conjointe ces trois éléments d'architecture [15].

La méthode proposée a été intégrée dans un outil nommé HEPTANE (voir paragraphe 5.2), analysant des programmes C et calculant leur WCET sur une architecture Intel Pentium. La structure modulaire de l'analyseur est conçue pour pouvoir aisément l'adapter à de nouvelles architectures cibles, nouvelles méthodes de prise en compte de la micro-architecture ou nouveaux langages de programmation.

Cet outil a été utilisé pour analyser des logiciels temps-réel, allant de petits programmes de tests [15] à un noyau de système temps-réel [16], permettant ainsi d'évaluer les performances de la méthode d'analyse. Ces expérimentations ont montré : (i) que les restrictions imposées sur le langages source afin de le rendre analysable statiquement sont réalistes, même sur du code complexe ; (ii) que le pessimisme de l'analyse est raisonnable, même s'il pourrait encore être réduit par identification de chemins d'exécution infaisables ou prise en compte d'autres éléments architecturaux ; (iii) que la prise en compte de la micro-architecture est incontournable pour l'obtention de bonnes performances de l'estimation.

Obtention de temps d'exécution au pire-cas par exécution des programmes. Obtenir les WCET des programmes par analyse statique des programmes présente deux inconvénients qui peuvent être jugés inacceptables selon le logiciel à évaluer et l'architecture cible. D'une part, l'utilisation d'une méthode d'analyse statique nécessite d'avoir accès au code source du logiciel à analyser. D'autre part, il est indispensable d'avoir des informations précises sur la structure interne du processeur, ce qui est de plus en plus difficile (sans contact privilégié avec un constructeur) étant donnée la complexité grandissante des processeurs. Ces deux raisons nous amènent actuellement à étudier les outils de caractérisation de temps de réponse de type *boite noire*, nécessitant moins d'informations sur le matériel et le logiciel, au détriment de la sûreté des estimations. L'idée est ici d'obtenir les temps d'exécution par exécution du logiciel, en orientant la génération des données d'entrée du logiciel de manière à se placer autant que

faire se peut dans le scénario aboutissant au pire temps de réponse du logiciel. La méthode triviale consistant à générer toutes les configurations possibles de paramètres d'entrée posant des problèmes d'explosion combinatoire, il est nécessaire de réduire la taille de l'espace de recherche. Soulignons aussi qu'un des problèmes est de générer des données d'entrée pour le logiciel qui soient valides, afin d'évaluer le pire temps de réponse du logiciel et non pas le temps de réponse en présence de fautes d'origine logicielle.

Validation de systèmes temps-réel par simulation. La simulation est une approche alternative à celle de l'analyse d'ordonnançabilité pour valider le comportement temporel de systèmes temps-réel. Contrairement aux méthodes d'analyse d'ordonnançabilité, les méthodes de simulation ne requièrent pas une connaissance statique des caractéristiques temporelles du système et de son environnement. Ainsi, elles peuvent être utilisées pour valider une large classe de systèmes temps-réel, dont les systèmes temps-réel souples, pour lesquels bien souvent on ne connait pas statiquement de manière sûre toutes les caractéristiques temporelles. Bien que moins précise qu'une expérimentation sur machine cible, la simulation d'un système temps-réel est plus légère à mettre en place que le système final, et ainsi permet, à un coût de mise en œuvre raisonnable, d'évaluer les performances d'une politique d'ordonnancement ou de l'architecture d'un système.

De nombreuses methodes et outils de simulation généralistes existent, mais aucun d'entre eux n'est réellement adapté à la simulation de systèmes temps-réel réaliste. De ce fait, nous avons défini une infrastructure de simulation à événements discrets *modulaire* adaptée à de tels systèmes. Cette infrastructure, nommée Artisst, pour *Artisst is a Real-Time System Simulation Tool* est composée de modules communiquant par messages, et pouvant être remplacés selon les besoins de la simulation. Les modules sont de trois types : (i) simulation de l'environnement externe ; (ii) simulation du système d'exploitation temps-réel ; (iii) évaluation de métriques. Le module de simulation du système d'exploitation est lui même modulaire, et permet en particulier de remplacer ou modifier l'ordonnançeur, qui choisit l'ordre d'exécution des tâches.

L'infrastructure Artisst, dans son état d'avancement actuel [29], fournit un ensemble de modules d'entrée (génération de flots d'événements d'entrée à partir de fichiers de traces ou de lois de distribution), de sortie (représentation graphiques ou textuelle) et plusieurs modules de simulation de systèmes temps-réel (ordonnancement Earliest Deadline First [LL73] et Task Pair Scheduling [Str95]).

6.3 Environnement Java embarqué pour ordinateur de poche

Participants : Michel Banâtre, Gilbert Cabillic, Teresa Higuera, Valérie Issarny, Jean-philippe Lesot, Salam Majoul, Frédéric Parain, Jean-Paul Routeau, Pierre Tiako.

Ces travaux s'inscrivent dans le cadre de la conception et la réalisation d'un environnement d'exécution Java embarqué adapté à un ordinateur de poche disposant d'un moyen de commu-

[LL73] C. Liu, J. Layland, « Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment », *Journal of the ACM 20*, 1, janvier 1973, p. 46–61.

[Str95] H. Streich, « TaskPair-Scheduling: An Approach for Dynamic Real-Time Systems », *International Journal of Mini and Microcomputers 17*, 2, 1995, p. 77–83.

nication sans fil. Cet environnement doit tout d'abord prendre en compte les contraintes liées à l'architecture matérielle qui est dans notre cas un multiprocesseur hétérogène à mémoire partagée. L'architecture possède également d'autres contraintes : une capacité mémoire limitée, une capacité de traitement limitée par processeur, et une autonomie d'énergie limitée.

L'utilisation de ces ordinateurs de poche sans fil, aujourd'hui cantonnée à la gestion des agendas, à la prise de notes ou au traitement du courrier électronique est en pleine évolution. Demain, l'intégration du traitement de données multimédias au sein de ces ordinateurs permettra la mise en place de nouvelles applications, telles que la vidéo-conférence ou la vidéo à la demande, chargées dynamiquement à partir de fournisseurs de services.

Nos travaux actuels se focalisent sur trois axes de recherches :

- la conception et le développement de la machine d'exécution Java modulaire distribuée Scratchy,
- la proposition d'une stratégie d'ordonnancement multi-critère d'applications multimédias écrites en langage Java conciliant la gestion du temps-réel souple et la minimisation de la consommation énergétique,
- la définition d'une technique de ramasse-miette adaptée aux applications multimédias.

Nous décrivons dans la suite de ce paragraphe les résultats obtenus autour de ces différents axes.

6.3.1 Machine d'exécution Java modulaire Scratchy

Les travaux menés autour de la machine d'exécution Scratchy ont pour objectif d'apporter une solution à plusieurs problèmes décrits ci-après. Tout d'abord, ils doivent permettre de comprendre les problèmes de performances de la machine d'exécution Java liés à l'interprétation de bytecode. Puis, il faut pouvoir supporter l'exécution d'une application Java au-dessus d'une architecture multiprocesseur et ce, de manière transparente pour le concepteur de l'application Java. Enfin, cette machine d'exécution doit pouvoir s'exécuter sur différents types de processeurs et différents systèmes d'exploitation temps-réel.

Comme nous l'avons indiqué dans le paragraphe 5.3, notre approche est d'avoir décomposé de manière modulaire l'ensemble de l'environnement d'exécution Java afin de pouvoir travailler indépendamment sur chaque module. Afin de réaliser cette modularité nous procéder en deux étapes. Tout d'abord nous avons conçu un outil de développement qui permet depuis un ensemble de modules indépendants de générer le code source final de notre environnement Java. Cet outil génère automatiquement les appels entre les modules et permet de plus de réaliser des optimisations sur l'allocation mémoire des structures utilisées par les différents modules. Ensuite, nous avons écrit en partant de rien la totalité de notre environnement d'exécution Java en le décomposant en un ensemble de modules. Ces différents aspects sont détaillés dans [4]. Enfin, nous avons conçu et réalisé des modules afin de permettre l'exécution de la Jvm au-dessus d'un multiprocesseur hétérogène. Ces modules gèrent tant le partage que la conversion à la volée des objets accédés par les différents processeurs.

Notre environnement d'exécution Java est compatible avec la spécification CLDC [mic00] de Sun. Néanmoins, dû au support de la gestion de l'arithmétique flottante et des threads, notre

[mic00] S. MICROSYSTEMS, « CLDC and the K Virtual Machine (KVM) », *rapport de recherche*, Sun microsystems, 2000, http://java.sun.com/products/cldc/.

environnement est très proche de la spécification CDC (voir également [mic00]). Concernant les processeurs, le Pentium en mono-processeur et bi-processeur est complètement supporté. Les processeurs de type DSP C55X et ARM9 le sont partiellement. Concernant les systèmes d'exploitation cibles, notre environnement est compatible avec les systèmes Linux, Windows ainsi que VxWorks qui est un système temps-réel commercial.

6.3.2 Ordonnancement multi-critère d'applications multimédias

Afin de permettre l'ordonnancement d'applications multimédias qui puisse concilier la gestion du temps-réel souple et la minimisation de la consommation énergétique, nous avons été amenés à introduire tout d'abord un modèle d'application multimédia permettant d'exprimer les contraintes temporelles souples des applications écrites dans le langage Java. Ensuite, après avoir étudié les techniques de réduction de la consommation pour les systèmes embarqués (voir [9]), nous avons défini une politique d'ordonnancement multi-critère compatible avec ce modèle. Enfin, afin d'obtenir les informations propres à chaque application nécessaires à la réalisation de la politique d'ordonnancement (estimation du temps d'exécution et estimation de la consommation énergétique des traitements temps-réel d'une application), nous examinons la conception d'une technique de caractérisation (*"profiling"*) d'applications multimédia.

Nous décrivons successivement ces trois points dans la suite de ce paragraphe.

Écriture d'applications multimédias dans le langage Java. Pour autoriser l'écriture d'applications multimédias dans le langage Java, nous avons défini un modèle d'application multimédia. Ce modèle a pour objectif de permettre la caractérisation des contraintes temporelles des applications multimédias à flux continus. Nous avons ainsi introduit la notion d'application décomposée en un ensemble de traitements périodiques, chaque traitement étant décomposé en une ou plusieurs unités d'exécution (correspondant à l'exécution d'une partie d'un traitement). Nous avons également introduit dans ce modèle la notion de mode d'exécution qui permet de prendre en compte dynamiquement les différents comportements des applications en fonction de leurs données en entrées. Enfin, afin de permettre à un concepteur d'une application multimédia d'exprimer ce modèle en Java, nous avons défini et réalisé une Api qui permet de représenter ce modèle.

Ordonnancement multi-critère. Le travail de l'ordonnanceur est réalisé via l'exécution d'une règle de placement exécuté par chaque processeur qui permet de choisir la tâche à exécuter parmi l'ensemble des tâches Java prêtes à s'exécuter. Cette règle a été définie afin d'atteindre plusieurs objectifs :

- prise en compte dynamique d'une nouvelle application temps-réel,
- introduction de parallélisme de manière transparente pour l'utilisateur au niveau de l'exécution en utilisant l'ensemble des processeurs,
- Exécution *au meilleur effort* des applications multimédias ;
- prise en compte de l'hétérogénéité des processeurs au niveau temporel, en réalisant le placement en fonction du temps d'exécution des traitements qui varient d'un processeur à l'autre,

– prise en compte de l'hétérogénéité des processeurs au niveau énergétique, en réalisant le placement en fonction de la consommation énergétique des traitements afin de minimiser la consommation énergétique globale du multiprocesseur.

Cet ordonnancement a été mis en oeuvre et validé au-dessus du système VxWorks s'exécutant sur une architecture bi-processeur Pentium. De plus, afin d'évaluer (travail en cours) le comportement de la règle de placement, un simulateur a été réalisé afin de simuler une architecture multiprocesseur hétérogène.

Caractérisation (profiling) d'application multimédia. L'ordonnanceur se basant sur les estimations temporelles et énergétiques propre à chaque application, disposer d'un outil de caractérisation (profiling) permettant d'obtenir ces estimations était nécessaire. Notre approche consiste à se focaliser sur le profiling au niveau des bytecodes des traitements temps-réel d'une application Java en définissant une technique de profiling d'une application Java. Cet outil a été conçu et réalisé en s'appuyant sur une extension de l'interpréteur de notre machine d'exécution Java. Nous avons de plus expérimenté cette technique pour plusieurs applications afin d'obtenir des données nécessaire à l'évaluation de la règle de placement multi-critère. Ces évaluations ont étés menées dans le cadre de processeurs DSPC55X et ARM9.

6.3.3 Ramasse-miettes adapté aux applications multimédia

Au niveau d'un environnement d'exécution Java, il est nécessaire de disposer d'un ramasse-miettes qui libère l'espace mémoire alloué aux objets Java devenus inutiles. Dans notre cas, ce ramasse-miettes a la particularité de ne pas devoir perturber l'exécution des applications multimédias afin de permettre de respecter leurs contraintes temporelles.

Afin de proposer une solution au ramasse-miettes dans ce contexte, deux approches ont étés suivies, nous les décrivons ci-après.

– (i) L'émergence d'une interface permettant de construire des applications temps-réel dans le monde Java (Real-Time Specification for Java (noté RTSJ) [fJ]) a amené à proposer une solution au ramasse-miettes dans ce contexte. RTSJ permet la définition de zone mémoires appelées régions. Afin de les supporter pleinement, la conception d'une stratégie de ramasse-miettes a été proposée [17]. De plus, une étude détaillée a été menée afin de diminuer le surcoût temporel introduit par le ramasse-miettes en utilisant un support matériel. Suite à cette étude une extension du jeux d'instruction d'un processeur Java existant a également été proposé. L'article [6] présente ces différents aspects.
– (ii) Notre seconde approche a été de proposer une stratégie qui prenne en compte les propriétés de notre modèle d'exécution d'application multimédia. Cette stratégie de ramasse-miettes permet notamment de s'exécuter de manière parallèle dans un environnement multiprocesseur tout en gérant l'hétérogénéité des processeurs. Une réalisation de ce ramasse-miette a été faite au sein de notre Jvm modulaire Scratchy et une étude est en cours afin de réaliser une évaluation tant qualitative que quantitative de cette stratégie. De plus, nous proposons une adaptation de cette stratégie pour les applications ne sup-

[fJ] T. R.-T. S. for Java, *G. Bollella and J. Gosling and B. Brosgol and P. Dibble and S. Furr and M. Turnbull*, Addison-Wesley.

portant pas notre modèle d'exécution par le biais d'une Api Java que nous avons définie et réalisée.

6.4 Systèmes d'information spontanés (SIS)

Participants : Michel Banâtre, Gilbert Cabillic, Paul Couderc, David Touzet, Arnaud Troël, Stéphane Tudoret, Grégory Watts, Frédéric Weis.

L'informatique mobile connaît actuellement un essor considérable dû au développement conjugué des communications sans fil et des technologies embarquées. Ces évolutions récentes peuvent par exemple être illustrées par le succès de la téléphonie mobile. Ces architectures cherchent pour la plupart à déployer un ensemble de services homogènes sur la totalité de leur zone de couverture. Le système WAP, par exemple, propose à ses utilisateurs l'accès au même ensemble de serveurs Web, quelque soit leur position géographique.

Un certain nombre d'études se distinguent aujourd'hui de cette approche classique. Pour ce faire, elles proposent d'exploiter la mobilité des utilisateurs, et plus généralement leur contexte, afin d'adapter les services rendus par un système à la situation des utilisateurs. L'ensemble de ces travaux est aujourd'hui regroupé sous l'étiquette *ubiquité numérique*. L'exploitation de ce principe permet d'envisager le déploiement de nouveaux types d'applications.

Notre approche à ces travaux est ici de montrer que dans un contexte particulier, celui du voisinage physique des entités communicantes, il est possible de tirer partie des facultés de mobilité et de la communication sans fil, pour créer de façon spontanée des systèmes d'information, que nous appelons SIS (Système d'Information Spontanés). Dans ce cadre, le fonctionnement des nœuds mobiles s'affranchit de toute infrastructure, qu'elle soit de communication ou d'acquisition du contexte.

Notre objectif est de définir, dans le cadre d'un environnement mobile et embarqué, une architecture système permettant la mise en œuvre d'un SIS. Dans ce but, nos travaux actuels se focalisent sur trois axes de recherches : (i) la définition d'interactions sans fil de proximité efficaces entre des nœuds mobiles, (ii) la représentation des informations échangées entre les nœuds mobiles, (ii) et l'accès aux informations entre nœuds voisins, au sein d'un SIS. Nous décrivons dans la suite de ce paragraphe les résultats obtenus autour de ces trois axes.

6.4.1 Gestion des interactions de proximité au sein d'un SIS

Un SIS est un système instable : les nœuds entrent et sortent dynamiquement du système, en fonction de leur mobilité et de leur position relative. On ne peut donc *a priori* pas faire d'hypothèse sur le temps de communication entre les nœuds. De plus, les échanges au sein d'un SIS s'appuient sur une communication sans fil, soumise à de perpétuelles perturbations (grandes variations de performances en fonction de l'environnement, variation de la qualité du signal reçu, taux d'erreurs variables ...). Dans ce cadre, les déconnexions lors de transferts d'informations entre nœuds voisins doivent être considérées comme des événements normaux, et les paramètres caractérisant la mobilité des nœuds voisins ne doivent pas être masqués au système embarqué. La prise en compte de ces paramètres dans le but d'améliorer les communications sans fil est un problème activement étudié dans de nombreux domaines d'applications, notamment pour améliorer les protocoles de routage dans les réseaux ad hoc.

Dans le cadre des SIS, l'originalité de notre approche est de prendre en compte les paramètres caractérisant la mobilité des nœuds voisins, non pas au niveau des protocoles de communication (comme c'est le cas dans les réseaux ad hoc), mais dans le cadre du système d'exploitation. Notre démarche est la suivante : nous proposons d'estimer le temps de communication « restant » ou *link expiration time* entre deux entités mobiles au sein d'un SIS, ce temps étant considéré comme une ressource par le système d'exploitation du nœud et devant nous permettre de construire un schéma prédictif de communication. Afin de le calculer, il est nécessaire de s'appuyer sur des paramètres caractérisant la mobilité des nœuds. Ces paramètres peuvent être obtenus directement au niveau de la carte de communication sans fil du nœud : distance, qualité et puissance du signal reçu, taux d'erreurs. Il est également possible de faire appel à des systèmes externes de localisation, comme le GPS par exemple.

Afin de déterminer le temps restant de communication, deux solutions, reposant sur des estimations successives de la distance séparant les deux nœuds communicants, ont été proposées [24]. La première s'appuie sur la méthode des moindres carrés, et la seconde sur une méthode d'estimation non linéaire. Notons que ces approches s'appuient essentiellement sur la distance euclidienne séparant les deux entités protagonistes. Cette mesure est difficile à obtenir au niveau d'une carte de communication sans fil. Toutefois, dans le cadre de nos simulations, la distance est équivalente à la puissance du signal reçu. Cette puissance est une mesure plus facile à obtenir dans la réalité. Dans le cadre de notre étude, dont un des objectifs est de montrer l'impact de la prédiction sur les échanges d'informations, cela s'avère efficace, et ces méthodes peuvent s'appliquer sans problème à partir d'une mesure reposant sur la puissance.

Le schéma prédictif permet de déterminer si une information qu'une application souhaite transmettre dans le cadre d'un SIS, peut être envoyée, compte tenu du temps de communication restant. Nous avons fait l'hypothèse qu'une application construite au sein d'un SIS, transmet ces informations sous la forme de blocs caractérisés pour chacun d'entre eux par une taille et une priorité. Ce choix nous a amené à étudier différentes politiques d'ordonnancement visant à sélectionner le ou les blocs d'informations pouvant être transmis, en fonction des tailles et des priorités des blocs proposés par l'application d'une part, et de la "viabilité" de ces blocs d'autre part (estimée par rapport au taux d'erreurs du canal de communication et au temps de communication restant).

L'ensemble des schémas prédictifs ont été évalués grâce à l'environnement de simulation NS (Network Simulator) développé par l'université de Berkeley. Cet environnement de simulation, a été adapté au contexte de notre étude afin de prendre en compte la mobilité des entités, et de pouvoir embarquer au sein de chaque entité simulée l'architecture de communication, propre aux systèmes d'informations spontanés. Cette étude a permis de démontrer que l'approche prédictive dans les communications au sein d'un SIS permet d'éviter d'entamer des émissions qui ne peuvent pas mener à leur terme, et donc d'améliorer l'efficacité dans l'utilisation de la bande passante [24].

6.4.2 Représentation des informations au sein d'un SIS

Toujours dans le cadre des communications de proximité, nous étudions le problème de la représentation des données. En effet, compte tenu de l'instabilité d'un SIS et des temps de communication limités, il n'est pas possible d'assurer le transfert atomique d'informations si

elles sont trop volumineuses. De plus, le schéma prédictif peut être mis en défaut, à cause par exemple de l'imprécision relative des paramètres retenus pour caractériser la mobilité des nœuds. Il faut donc se doter d'un mode de représentation de l'information propre aux SIS. Notre idée initiale était de découper l'information en blocs élémentaires, chacun des blocs étant caractérisé par une taille et une priorité. Le transfert de l'information, assuré par le protocole de communication sousjacent, porte sur chacun de ces blocs. Une des pistes suivies est d'appliquer des politiques de d'adaptation (i.e. de dégradation) pour générer ces blocs. Par exemple, une image (l'information à transmettre) peut donner lieu à différents types de représentations, avec une précision plus ou moins grande. Ces différents niveaux de précision constituent les blocs à transmettre, le choix s'effectuant en fonction du temps de communication estimé entre deux nœuds.

Nous étudions également une seconde approche, complémentaire de la précédente. Il s'agit de proposer un format de données, échangé entre les nœuds d'un SIS, qui permettent d'exploiter une information même si elle n'a pas été totalement reçue par le nœud distant. Pour cela, un mode de représentation s'appuyant sur le langage HTML, appelé *progressive HTML* a été défini. Ce dernier permet de structurer un document sous une forme arborescente, l'importance de chaque partie de ce document étant fonction de sa profondeur dans l'arbre (plus l'information est loin de la racine, moins elle est importante). Nous avons ensuite associé ce mode de représentation des données à une fonction de transmission entre les nœuds mobiles d'un SIS, afin que la ou les parties les plus importantes d'un document soient systématiquement transmises au début de la communication.

6.4.3 Techniques d'accès à l'information au sein d'un SIS

Un SIS est constitué d'un ensemble de nœuds dont le nombre varie perpétuellement. Chacun de ces nœud est porteur d'un ensemble d'informations, qu'il est susceptible de rendre accessible dans le cadre du SIS auquel il participe. L'ensemble de ces informations "publics", distribuées sur l'ensemble des nœuds participant à un instant t à un SIS donné, est appelé l'espace visible du SIS. Cette espace doit refléter dynamiquement l'ensemble des informations potentiellement accessibles dans le SIS. Nous étudions actuellement les mécanismes systèmes nécessaires à la construction de l'espace visible. La difficulté réside dans la nature totalement décentralisée d'un SIS. En effet, chaque nœud est autonome. Il est donc impossible de faire jouer à l'un d'entre eux le rôle d'un annuaire centralisé (à la façon par exemple dont le *lookup service* centralise les services au sein du système JINI). Dans un premier temps, un protocole "minimal" (ou *bootstrap*) a été proposé, afin de permettre à deux nœuds qui viennent de se détecter mutuellement (dans un même voisinage physique), de s'identifier et de démarrer une connexion. Ce protocole nous permet d'"exporter" vers chaque nœud participant au SIS une liste d'identifiants représentant les informations publics disponibles au sein de l'espace visible. Nous cherchons maintenant à étendre cette solution en prenant en compte l'aspect dynamique du SIS, notamment en collaborant étroitement avec le niveau prédictif.

Le deuxième axe que nous étudions concerne les modes d'interactions au sein de l'espace visible d'un SIS. Il s'agit de définir de quelle manière les nœuds d'un SIS peuvent accéder aux informations de l'espace visible associé. Ce problème nous renvoie aux techniques d'accès à l'information utilisées par l'ubiquité numérique . Le principal objectif de l'ubiquité numérique est

de réaliser une intégration transparente des environnements physique et numérique. Et cette intégration doit permettre aux utilisateurs d'interagir le plus naturellement possible avec les calculateurs qui les environnent. Dans le cadre spécifique des SIS, si on pousse ce raisonnement à terme, les interactions doivent avoir lieu sans que l'utilisateur n'ait conscience d'accéder aux informations des nœuds se trouvant dans le même voisinage physique. Nous avons proposé une solution appelé le *Web de proximité*, qui permet à des utilisateurs d'échanger automatiquement des informations en fonction de leur centres d'intérêts respectifs. Trois problèmes ont été considérés dans le cadre de cette étude [23] :

- la définition de profils utilisateur : afin d'émettre automatiquement des requêtes "pertinentes" vers un nœud distant, les centres d'intérêt d'un utilisateur doivent être découverts. Dans ce but, le système doit construire localement et automatiquement un profil de l'utilisateur, c'est à dire les sujets sur lesquels ce dernier souhaite récupérer de l'information au sein d'un SIS.
- l'indexation des informations publics : chaque nœud doit automatiquement organiser ces documents publics, afin de permettre des échanges efficaces durant les périodes (souvent brèves) de communication au sein d'un SIS. Une solution d'indexation thématique a été retenue. Ainsi, à chaque objet, il est proposé d'adjoindre une enveloppe dans laquelle sont stockées toutes les informations utiles le concernant (en particulier une liste de mots clefs décrivant le contenu de l'objet). Cette séparation entre l'objet et sa représentation a pour but de faciliter ensuite la recherche d'informations sur les nœuds distants au sein d'un SIS.
- la découverte spontanée des informations pertinentes : pendant la rencontre de deux nœuds au sein d'un SIS, il est nécessaire de sélectionner les documents en fonction des profils utilisateurs. C'est sur ce plan que la politique d'indexation de l'information prend tout son sens. A partir d'une définition, les informations sont regroupées par domaine, en utilisant une technique dérivée du *data mining*. Cette dernière permet, par l'association des mots clefs apparaissant fréquemment ensemble, de distinguer les principaux centres d'intérêts d'un utilisateur.

Le protocole de découverte proposé permet à un nœud de découvrir, parmi les autres entités appartenant au SIS, toutes celles qui correspondent "le mieux" à son profil. Afin de proposer une implémentation complète de ce protocole, une architecture composée de quatre modules a été définie :

- *Storage* : ce module se charge d'une part du stockage des documents publics (accessible via l'espace visible) au niveau de chaque nœud, ainsi que de la sauvegarde des informations obtenues sur une entité distante du SIS. Dans la mesure où les ressources d'une entité participant à un SIS (par exemple un PDA) sont limitées, ce module a été conçu comme un cache. Ainsi, seules les informations les plus récemment accédées au sein d'un SIS sont conservées localement.
- *Database manager* : ce module implante le mécanisme de *data mining* qui réalise le mécanisme d'indexation des informations au niveau de chaque nœud du SIS.
- *Information Discovery* : ce module réalise la découverte des éléments communs entre les profils utilisateurs du nœud local et du nœud distant.
- *Communication* : ce dernier module réalise l'interface entre les mécanismes de découverte automatique, et l'interface de communication sans fil. Dans nos travaux futurs, il devra

collaborer avec les mécanismes prédictifs et d'adaptation des données décrits dans la partie précédente.

6.5 Informatique nomade et réseaux sans fil

Participants : Françoise André, Frédéric Le Mouël, Maria-Teresa Segarra.

Les propriétés de l'informatique mobile, où sont impliqués des ordinateurs portables utilisant des réseaux sans fil, diffèrent de celles définies pour des contextes plus traditionnels, comme celui des réseaux locaux filaires par exemple. Ces différences ont des répercussions sur les applications, notamment parce que celles-ci doivent être mises en œuvre avec des ressources limitées (faibles débits réseau par exemple). De plus, les propriétés du nomadisme informatique combinées à celles des réseaux sans fil rendent envisageable la mise au point de nouveaux modes de fonctionnement pour les applications, comme par exemple la possibilité (parfois la nécessité) de travailler en mode déconnecté.

Pour un utilisateur d'applications, le nomadisme présente de nombreux avantages. Pour les concepteurs d'applications, ou encore pour les concepteurs de systèmes d'exploitation répartis, le nomadisme introduit une complexité supplémentaire. Les applications existantes, non pensées en vue d'être utilisées dans un contexte nomade, risquent fort de ne pas fonctionner. Les problèmes qu'elles peuvent rencontrer sont par exemple dus aux mécanismes internes qu'elles utilisent, et qui peuvent être incompatibles avec la mobilité. Les techniques de verrouillage traditionnelles aux bases de données et utilisées pour protéger l'accès concurrent aux données partagées sont un exemple d'incompatibilité. Si l'application ayant posé des verrous se déconnecte, la base peut rester partiellement inaccessible, et cela sans limite dans le temps. A l'identique, un système d'exploitation réparti traditionnel peut ne pas être capable d'offrir un support d'exécution adéquat à des applications mobiles, si le nomadisme n'a pas fait partie de sa conception dès l'origine.

L'activité Molène a pour but de fournir un ensemble de services système génériques permettant à des concepteurs de rendre des applications ou des systèmes d'exploitation répartis compatibles avec les propriétés de l'informatique nomade. Molène est une couche logicielle fondée sur le modèle objet qui se place entre un système d'exploitation réparti traditionnel et une application elle aussi traditionnelle (c'est-à-dire initialement non prévues pour fonctionner dans un contexte nomade). Molène permet au système d'exploitation réparti et à l'application d'ignorer certains problèmes liés à la mobilité, et leur donne l'illusion que chacun fonctionne dans un cadre traditionnel. La modélisation objet utilisée dans Molène permet de spécialiser aisément un comportement particulier (gestion de la cohérence de données en présence de déconnexions par exemple), et d'introduire des stratégies d'adaptation dynamiques afin d'obtenir un comportement mieux adapté aux conditions d'exécution courantes.

Un prototype de Molène a été réalisé et des expériences d'utilisation de celui-ci ont été conduites dans des domaines d'application divers. Une de ces expériences concerne la distribution des applications en fonction d'un environnement d'exécution mobile et hautement variable. La distribution des applications "mobiles" permet de pallier la pauvreté des ressources du terminal mobile par une utilisation au mieux des ressources de l'environnement. Selon la disponibilité des ressources, la distribution améliore les performances ou en évite la dégrada-

tion. Pour s'adapter aux fluctuations de l'environnement mobile, la distribution doit prendre en compte les différents critères du cadre mobile : batterie, bande passante, latence, probabilité de déconnexion, probabilité de reconnexion, pour assurer le dynamisme aussi bien au niveau du placement que des politiques de distribution elles-mêmes. Ces différentes fonctionnalités sont mises en œuvre dans le système AeDEn (*Adaptive Distribution Environment*) [8].

Ces expériences nous ont permis de valider l'intérêt des mécanismes proposés et notamment de l'adaptation dynamique à l'environnement. Ce travail se poursuit en étudiant notamment la coordination lors de l'adaptation simultanée de plusieurs composants. Plusieurs perspectives sont ouvertes en particulier pour régler les problèmes d'adaptation dans le cadre de l'utilisation conjointe de divers types de réseaux mobiles.

6.6 Systèmes d'information contextuels et navigation spatiale

Participants : Michel Banâtre, Paul Couderc.

L'informatique mobile connaît depuis le début des années 1990 un essor important. Deux aspects principaux contribuent à cet essor. D'une part le développement des calculateurs mobiles : ordinateurs portables, assistants numériques, téléphones mobiles, outils de navigation GPS sont autant de formes des calculateurs qui peuvent être embarqués par un utilisateur aujourd'hui. D'autre part, les infrastructures de communication pour terminaux mobiles sont également en plein essor, en particulier les réseaux téléphoniques cellulaires.

Parallèlement à ces développements, le Web s'est imposé comme le fédérateur des systèmes d'informations grande échelle, en offrant une interface universelle. Aujourd'hui, de plus en plus d'entités réelles (personnes, objets, administrations, entreprises, produits, lieux ...) ont une *existence* sur le Web, c'est-à-dire une représentation sous la forme d'une page ou d'un site.

Technologiquement, l'informatique mobile offre aujourd'hui le potentiel pour permettre un accès omniprésent à des systèmes d'informations grande échelle comme le Web. Cependant, offrir cet accès n'est pas suffisant.

A cause de sa structure anarchique et dynamique, la recherche d'informations sur le Web est complexe, et peut être longue et fastidieuse. Des moteurs de recherche indexant une partie du Web facilitent ces recherches, mais dans le cadre d'une utilisation mobile, ils sont inadaptés : l'utilisateur doit pouvoir accéder très rapidement aux informations du Web en rapport avec la situation où il se trouve.

Ces travaux visent à proposer une intégration du système d'information (Web) à l'environnement physique, permettant ainsi à l'utilisateur d'accéder très rapidement aux informations du Web en rapport avec sa situation (réelle). Notre système repose sur la notion de contexte , qui représente un ensemble de noeuds du système d'information proches d'un point de référence selon une *dimension* donnée. En particulier, la dimension spatiale permet de déterminer les noeuds du système d'information qui sont connexes dans l'espace physique à un point donné. Dans notre modèle, un système Web classique ne permet à l'utilisateur que de naviguer selon la dimension «thématique» (correspondant aux hyperliens de la page courante), et selon la dimension «historique» (correspondant à la suite des pages visitées). Grâce à ce système, l'utilisateur peut naviguer dans le système d'information implicitement, en se déplaçant physiquement.

Mais un tel concept n'a d'intérêt que s'il est possible de construire simplement des sys-

tèmes d'informations reflétant cette dimension spatiale. En effet, l'espace physique réel étant extrêmement dynamique (à cause des nombreuses entités mobiles qu'il comporte), le problème de la représentation du contexte spatial des noeuds du système d'information est difficile. En particulier, les solutions centralisées répercutant dans une base de données géographiques les positions des entités mobiles du système à chaque déplacement, trouvent leurs limites.

Notre approche à ce problème, décrit dans [2], utilise la notion de contexte spatial : il s'agit d'un ensemble d'informations *accessibles dans une zone limitée de l'espace*, diffusées directement par les entités physiques (ou un représentant «électronique» embarqué sur l'entité) auxquelles ces informations correspondent. Nous avons défini une architecture permettant une implémentation du contexte spatial, qui reste efficace même en présence d'une grande mobilité des entités. Cette architecture repose sur des technologies de communication sans fil courte portée qui permettent d'implémenter simplement la notion de contexte spatial.

Un prototype à base d'ordinateurs de poches Ipaq et de wave lan est actuellement opérationnel.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Contrats nationaux

7.1.1 Hades (partenariat DGA/Dassault-Aviation)

Participants : Michel Banâtre, Pascal Chevochot, Isabelle Puaut.

- Numéro de la convention : 3198C4980031303011.
- Intitulé : Conception et réalisation d'un support d'exécution réparti hautement disponible pour applications temps-réel.
- Activité de recherche concernée : § 6.2.
- Partenaires : DGA, Dassault-Aviation.
- Période : février 1999 à août 2001.
- Financement : DGA.

L'évolution des systèmes d'avionique fait que, dans le contexte économique actuel, l'utilisation de composants matériels et logiciels *sur étagère* devient une nécessité, principalement pour des raisons de complexité grandissante (avionique modulaire embarquée). Cette action a pour objectif de concevoir et mettre en œuvre un support d'exécution distribué tolérant aux fautes pour les applications du domaine de l'avionique, et ce à partir de composants (processeurs, réseau, système d'exploitation) sur étagères. Plus précisément, les objectifs visés sont : (i) fournir des mécanismes de tolérance aux fautes qui soient compatibles avec des échéances temps-réel strictes et qui soient transparentes pour le concepteur d'application ; (ii) fournir des mécanismes permettant de récupérer en-ligne les ressources inutilisées ; (iii) s'adapter à des demandes de qualité de service variées (échéances temporelles critiques, besoins en terme de tolérance aux fautes). Cette convention, financée par la DGA, est effectuée en étroite collaboration avec la société Dassault-Aviation, cible du transfert industriel des résultats de recherche produits.

Un contrat de collaboration, de durée identique à la convention DGA, nous lie par ailleurs à Dassault-Aviation pendant la durée de la convention DGA, et définit les termes des transferts

de connaissances et de logiciels Inria / Dassault-Aviation : fourniture à l'Inria d'une application avionique – conduite de tir de missiles – pour évaluer les fonctionnalités et les performances du support d'exécution ; transfert du support d'exécution à Dassault-Aviation.

Le code du support d'exécution a été transféré à Dassault-Aviation au printemps 2001 et est en cours de portage sur la plate-forme LynxOS/PowerPC existant chez Dassault-Aviation.

7.1.2 Texas Instruments

Participants : Michel Banâtre, Gilbert Cabillic, Teresa Higuera, Valérie Issarny, Jean-philippe Lesot, Salam Majoul, Frédéric Parain, Jean-Paul Routeau, Pierre Tiako.

- Numéro de la convention : 198C2730031303202
- Intitulé : Real time Java Distributed Processing Environment
- Activité de recherche concernée : § 6.3.
- Partenaire : *Texas Instruments.*
- Financement : *Texas Instruments.*
- Date de début : 01/10/1998, date de fin : 30/09/2001.
- Avenant Date de début : 01/10/2001, date de fin : 30/09/2003.

L'objectif de ce partenariat est la conception et la mise en œuvre d'un environnement Java temps-réel au-dessus d'architectures multiprocesseurs hétérogènes embarquables (comme par exemple les ordinateurs de poche), supportant l'exécution concurrente d'applications multimédias.

7.1.3 Alcatel

Participants : Michel Banâtre, Gilbert Cabillic, Paul Couderc, Stéphane Tudoret, Arnaud Troël, David Touzet, Gregory Watts, Frédéric Weis.

- Numéro de la convention : 101C078,
- Intitulé : logiciels et applications pour ordinateurs de poche sans fil utilisant des communications par voisinage physique.
- Activité de recherche concernée : § 6.4.
- Partenaire : *Alcatel.*
- Financement : *Alcatel.*
- Date de début : 01/10/2000, date de fin : 30/09/2003.

L'objectif de cette action est (i) de fournir une *architecture logicielle* pour les ordinateurs de poches sans fil permettant à ces derniers de coopérer dynamiquement et spontanément quand ils sont physiquement proches ; (ii) de fournir des *applications* qui exploitent cette architecture logicielle.

8 Actions régionales, nationales et internationales

8.1 Actions européennes

8.1.1 Projet LTR C3DS

Participants : Valérie Issarny, Christos Kloukinas.

- Numéro de la convention : 197A93200000MC005.
- Intitulé : *Convention Esprit Ltr – C3DS*, Environnement pour le développement de services distribués complexes.
- Activité de recherche concernée : §6.1.
- Partenaires : Université de Newcastle (UK – coordinateur), Imperial College (UK), Inria-Rhône-Alpes, Bull SA (Grenoble).
- Période : 01/98–02/2001.
- Financement : CEE–ESPRIT.

L'action Esprit Ltr C3DS (*Control and Coordination of Complex Distributed Services*) porte sur la conception et la réalisation d'un environnement de développement de services distribués. Cet environnement s'appuie sur les notions suivantes : (*i*) la description des architectures logicielles des services, (*ii*) l'emploi d'agents, éventuellement mobiles, pour la réalisation des actions au sein des composants distribués constituant un service, et (*iii*) l'emploi de la technique de *workflow* pour coordonner les actions relatives à un même service.

8.1.2 Projet IST CALIM

Participants : Valérie Issarny, Apostolos Zarras.

- Numéro de convention : 100D0052,
- Intitulé : IST Calim – *Corba Architecture for Legacy Integration and Migration*
- Activité concernée : §6.1
- Période : [Janvier 2000 - Février 2001]
- Partenaires : EADS CCR (Suresne) – Coordinateur, Fiat-CRF (Italie), Informate (Belgique), Steria (Toulouse).

L'action CALIM porte sur la définition d'un processus de développement d'architectures de systèmes et des méthodes et outils associés pour faciliter l'évolution des systèmes d'information s'appuyant sur des systèmes logiciels complexes existants. L'approche retenue repose sur une intégration puis une évolution des systèmes logiciels existants, au sein du système d'information, en s'appuyant sur une plate-forme Corba.

8.1.3 Projet IST DSoS

Participants : Valérie Issarny, Christos Kloukinas, Viet Khoi Nguyen, Ferda Tartanoglu.

- Numéro de convention : 100D0150,
- Intitulé : IST DSoS – *Dependable Systems of Systems*
- Activité concernée : §6.1
- Période : [Avril 2000 - Mars 2003]

– Partenaires : Université de Newcastle (RU) – Coordinateur, CNRS-LAAS (Toulouse), DERA/QinetiQ (RU), LRI (Orsay), Université d'Ulm (Allemagne), Université technique de Vienne (Autriche).

L'action DSoS vise à fournir des solutions facilitant la composition de systèmes de systèmes sûrs de fonctionnement, à partir de systèmes informatiques autonomes. Cette action se concentre notamment sur la conception, le placement et les propriétés des interfaces des systèmes à composer. Elle examine également la définition de méthodes et outils pour la construction effective des systèmes à partir de la spécification des interfaces des sous-systèmes à composer, ainsi que pour l'évaluation et la validation de la sûreté de fonctionnement des systèmes composés.

8.1.4 Projet IST OZONE

Participants : Valérie Issarny, Françoise Sailhan.

– Numéro de convention : En cours d'attribution
– Intitulé : IST OZONE – *New technologies and services for emerging nomadic societies*
– Activité concernée : §6.1
– Période : [Novembre 2001 - Avril 2004]
– Partenaires : Philips (Pays Bas) – *Coordinateur*, EPICTOID (Pays Bas), INRIA (Projets Cosi, Imara, Langue & Dialogue, Maia, Moscova, Parole, Sirac/Sardes, Solidor-Rocquencourt/Arles, Temics), IMEC (Belgique), LEP (Paris), TMM (Rennes), TU/e (Pays Bas)

L'action OZONE vise à développer de nouveaux concepts, techniques et outils pour fournir un environnement informatique invisible, afin de permettre une utilisation nomade, grand public, des technologies de l'information. Une base importante du projet est l'utilisation des nouvelles technologies pour une recherche et une utilisation de l'information, centrées sur l'utilisateur, comparée à l'approche actuelle qui est centrée sur l'infrastructure informatique. Ceci requiert des solutions adaptées au niveau des interfaces utilisateurs, de l'environnement logiciel et de l'infrastructure matérielle.

8.1.5 Projet ITEA Vivian

Participants : Anis Ben Arbia, Malika Boulkenafed, Valérie Issarny, David Mentré, Apostolos Zarras.

– Numéro de convention : 201D0060
– Intitulé : ITEA VIVIAN – *Opening mobile platforms for the development of component-based applications*
– Activité concernée : §6.1
– Période : [Juin 2000 - Aout 2002]
– Partenaires : Nokia NRC (Finlande) – Coordinateur, ADISOFT (D), CAS (D), HUT (FIN), INRIA (F), INT (F), MEMODATA (F), PALMWARE (F), PARAVANT (FIN), PHILIPS (NL), UNICOM (FIN)

L'action Vivian porte sur la définition d'une plate-forme pour ordinateurs de poche personnels, sans fil (*e.g.* Psion serie 5, *Communicator* de Nokia), afin de faciliter la construction d'applications logicielles distribuées pour ces systèmes. La solution retenue s'appuie sur une plate-forme *middleware* à base de composants, de type CORBA. Dans ce cadre, les problèmes abordés portent sur la définition et répartition des services composant l'architecture *middleware* sachant que les services devant s'exécuter sur les ordinateurs personnels doivent s'accommoder des ressources limitées disponibles.

8.1.6 Projet IST BRAIN

Participants : Michel Banâtre, Frédéric Weis.

- Numéro de convention : 100D0053
- Intitulé : on BRAIN (Broadband Radio Access for IP Based Networks)
- Période : [Janvier 2000-Mars 2001]
- Partenaires : Siemens (Allemagne) – Coordinateur, Btelecom (Angleterre), Dtelecom T-NOVA (Allemagne), Nokia (Finlande), Ericsson (Suède), NTT DoCoMo (Japon), Sony Europe (Allemagne), Agora (Espagne).

L'action BRAIN (Broadband Radio Access for IP Based Networks) porte sur la définition d'une architecture système pour terminaux mobiles, permettant de se connecter aussi bien à des réseaux respectant le standard HiperLan 2, qu'à des réseaux plus globaux comme l'UMTS.

8.2 Actions nationales

8.2.1 Groupes de travail nationaux

P. Chevochot et I. Puaut participent au groupe de travail GDR-ARP "Spécification Temps-réel et Stochastique" (STS).

I. Puaut participe à l'atelier thématique *Logiciel libre et sûreté de fonctionnement* du RIS (Réseau d'Ingénierie de la Sûreté de fonctionnement). Le but du RIS est de promouvoir et de faciliter la création, le pilotage et l'animation d'instances de réflexion sur des thèmes appartenant au domaine de l'ingénierie de la sûreté de fonctionnement des systèmes "à logiciel prépondérant".

I. Puaut participe à l'action spécifique CNRS intitulée *SOC embarqués, systèmes d'exploitation et architectures multiprocesseur* animée par M. Auguin (Université de Nice Sophia Antipolis) et P. Sainrat (IRIT).

8.3 Relations bilatérales internationales

Valérie Issarny et Apostolos Zarras participent à une coopération bilatérale Franco-Portugaise avec l'INESC-Porto, soutenue par INRIA/ICCTI. Le thème de cette coopération est l'adaptation dynamique d'architectures logicielles de systèmes distribués.

8.4 Réseaux et groupes de travail internationaux

8.4.1 CaberNet

- Numéro de la convention : 1 01 D0236
- Intitulé : IST NoE CaberNet (Network of Excellence in Distributed and Dependable Systems)
- Période : [Janvier 2001 - Décembre 2003]
- Partenaires :
  - Université de Newcastle (coordinateur),
  - Alcatel Austria, Technische Universitaet Wien (Autriche),
  - Université Catholique de Louvain (Belgique),
  - INRIA, LAAS-CNRS, Sun Microsystems International, Université Joseph Fourier (France)
  - GMD, Universités d'Aachen, de Dortmund, d'Essen, de Hamburg, de Kaiserslautern, de Karlsruhe, de Stuttgart, Technische Universitaet Hamburg-Harburg, Friedrich Alexander University (Allemagne),
  - Foundation for research and technology d'Hellas (Grêce),
  - Trinity College Dublin (Irlande),
  - CNUCE-CNR, Instituto di Elaborazionre dell'informazione - NCR, Politecnico di Milano, Scuola superiore S. Anna, Tecnopolis CSATA Novus Ortus, TNO Bari, Universités de Bologne et Pise (Italie),
  - Universités de Twente et de Vrije (Hollande),
  - Critical software SA, Universités de Lisbonne et de Porto (Portugal),
  - Universités de Catalogne, de Madrid, de Valence (Espagne),
  - Universités de Chalmers, Maelardalen Hoegskola (Suède),
  - École polytechnique de Lausanne (Suisse),
  - Universités de Cambridge, Lancaster, British Telecom, City University, Hewlett-Packard, Imperial College of science technology and medecine, Microsoft research (Royaume-uni).
- Durée : 3 ans

Cabernet est un réseau d'excellence dans le domaine des systèmes distribués et des systèmes sûrs de fonctionnement. Sa mission est de coordonner la recherche européenne de haut niveau dans ces domaines, dans le but de la rendre visible aux gouvernements et aux acteurs industriels d'une part, et d'améliorer la qualité de l'enseignement d'autre part.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

9.1.1 Comités de programme

- F. André est membre du comité de programme de la conférence CARI'2002 - 6ième Colloque Africain sur la Recherche en Informatique, Yaoundé, Cameroun, octobre 2002.
- V. Issarny est membre du comité de programme de WPDRTS'2001 – 9th workshop on parallel and distributed real-time systems. Avril 2001, San Francisco, USA.

- V. Issarny est membre du comité de programme de CFSE'2 – seconde conférence francaise sur les systèmes d'exploitation. Avril 2001, Paris, France.
- V. Issarny est membre du comité de programme de RENPAR'13 – Treizièmes Rencontres Francophones du Parallélisme. Avril 2001, Paris, France.
- V. Issarny est membre du comité de programme du Workshop on Concurrency in Dependable Computing. Juin 2001, Newcastle upon Tyne, RU.
- V. Issarny est responsable du comité de programme du Fourth CaberNet Plenary Workshop. Octobre 2001, Pise, Italie.
- V. Issarny est membre du comité de programme de SAINT'02 – Symposium on Applications and the Internet. Janvier-Février 2002, Nara, Japon.
- V. Issarny est membre du comité de programme d'AlgoTel'02. Avril 2002, Montpellier, France.
- V. Issarny est membre du comité de programme de WPDRTS'2002 – 10th workshop on parallel and distributed real-time systems. Avril 2002, Fort Lauderdale, USA.
- V. Issarny est membre du comité de programme de ISORC'2002 – The 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. Avril 2002, Washington DC, USA.
- V. Issarny est membre du comité de programme de ICSE 2002 – the 24th International Conference on Software Engineering, May 2002, Buenos Aires, Argentine.
- V. Issarny est membre du comité de pilotage de WIESS, Workshop on industral experiences with system software, co-sponsorisé par USENIX, IEETCOS et ACM-SIGOPS.
- I. Puaut est membre du comité de programme de ISORC 2002 - Object-Real-Time Distributed Computing, qui aura lieu à Washington, DC en avril 2002.
- I. Puaut est membre du comité de programme de l'école d'été sur les systèmes d'exploitation organisée par l'ASF (Association ACM Sigops de France) à Hammamet, Tunisie, en avril 2002.

9.1.2 Autres responsabilités sur un plan international

- V. Issarny est vice-présidente de l'ACM SIGOPS.
- V. Issarny est reponsable du *Research Coordination and Training Committee* de CaberNet.
- V. Issarny est membre du *core expert group* du groupe de travail de l'ISTAG, sur *Software technologies, services and distributed systems* (WG9).

9.1.3 Autres responsabilités sur un plan national

- F. André est membre de la commission de spécialistes de la 27^e section de l'Université de Rennes 1.
- V. Issarny est vice-présidente du prix ASF de la recherche en système.
- I. Puaut est membre élue de la commission de spécialistes de la 27^e section de l'Insa (membre du bureau) et membre nommée de la commission de spécialistes de l'Ifsic.
- I. Puaut est membre du conseil de laboratoire et du comité des projets de l'IRISA.
- I. Puaut est membre du jury du titre d'ingénieur DPE (Diplômés Par l'État), spécialité

informatique, de l'Insa de Rennes.

9.2 Enseignement universitaire

Les chercheurs et enseignants/chercheurs du projet Solidor coordonnent et participent à des enseignements de premier, second et troisième cycle à l'Ifsic, l'Insa de Rennes et l'IUT de St-Malo. Nous détaillons ci-dessous uniquement les interventions des membres du projet en *troisième cycle*.

- F. André assure une formation de système d'exploitation en DESS Compétences Complémentaires en Informatique (CCI) à l'Ifsic.
- G. Cabillic assure une formation de système d'exploitation et Java embarqué en DESS Compétences Complémentaires en Informatique (CCI) à l'Ifsic.
- G. Cabillic assure une formation de Java embarqué à l'École des Mines de Nantes (filière FI4).
- M. Banâtre est responsable du cours de systèmes d'exploitation répartis ($5^{ème}$ année Insa de Rennes option informatique).
- M. Banâtre assure une formation de systèmes répartis à l'École des Mines de Nantes (filière FI4).
- M. Banâtre, I. Puaut et F. Weis interviennent dans le module SYCR (SYstèmes d'exploitation et de Calculs Répartis) du DEA d'informatique.
- M. Banâtre et F. Weis assurent une formation de systèmes répartis en Diic 3 ARC à l'Ifsic.
- F. Weis assure une formation sur les réseaux en DESS Domotique et Réseaux Intérieurs l'université de Rennes I (UFR Structure et Propriétés de la Matière).

9.3 Accueil de stagiaires

Pendant l'année 2001, les membres du projet ont accueilli et encadré des stagiaires venant de différents établissements :

- M. Adly, ITI Egypte (stage de fin d'étude de 9 mois)
- M. Bellengé, INSA de Rennes (stage de $4^{ème}$ année)
- R. Loue, INSA de Rennes (stage de fin d'études)
- G. Lormeau, IUP Vannes (stage de $4^{ème}$ année)
- J-B. Pelletier, Institut d'Informatique d'Entreprise (IIE), (stage de fin d'étude)
- F. Sailhan, Université Pierre et Marie Curie, Strasbourg (stage de DEA)
- F. Tartanoglu, Université de Paris 6 (stage de DEA)
- M. Mac Weng, IFSIC (stage de DEA)

9.4 Participation à des colloques, séminaires, invitations

Un membre du projet a été invité à l'étranger pour un séjour de longue durée :

- G. Cabillic. Séjour de 3 mois (Mars, Avril et Mai) dans le département de Recherche et Développement "Wireless" à Texas Instruments Dallas, Texas.

Des membres du projet ont participé à des conférences et « workshops » ; on se reportera à la bibliographie pour en avoir la liste. Certains ont également donné des séminaires :

- M. Banâtre *Wireless Appliances et interactions de proximité*, Séminaire Labri, Bordeaux. Février 2001.
- M. Banâtre *Système d'exploitation et Ubiquité numérique*, Journées RNRT. Mars 2001.
- V. Issarny. *Architectures logicielles de systèmes et mobilité*, LIP6, Paris, France. Avril 2001.
- V. Issarny. *Software Architectures and Mobility*, INESC, Porto, Portugal. Juin 2001.
- C. Kloukinas. *Composing Software Architectures*, Université de Newcastle, RU. Mai 2001.
- C. Kloukinas. *SPIN-ning Software Architectures*, Verimag, Grenoble, France. Octobre 2001.
- I. Puaut. *Estimation de temps d'exécution au pire-cas par analyse statique de programmes*, réunion de groupe de travail Spécification Temporelle et Stochastique (STS) du pôle Architecture Réseaux et Systèmes (ARP) du GDR, février 2001.
- I. Puaut. *Caractérisation de robustesse et de temps de réponse de systèmes d'exploitation temps-réel*, atelier thématique *Logiciel libre et sûreté de fonctionnement* du RIS (Réseau d'Ingénierie de la Sûreté de fonctionnement), Toulouse, septembre 2001.
- I. Puaut. *Systèmes dexploitation pour applications temps-réel embarquées tolérantes aux fautes*, journée de veille scientifique et technologique Iliatech, organisée par l'Inria-Rocquencourt, Bougival, octobre 2001.
- F. Weis. *Interface de communication courte portée pour les systèmes contextuels*, Journée de veille technologique, "Communications sans fil de proximité et domaine d'applications", IRISA, Rennes, septembre 2001.
- A. Zarras. *A UML Profile and Methodology for Modeling Dependable Systems*, INESC, Porto, Portugal. Juin 2001.
- A. Zarras. *Quality Analysis of Enterprise Information Systems*, Journée "Architecture Logicielle", Club SEE "Systemes Informatiques de Confiance", Cercle "Objectif Zero-Defaut", ENST, Paris, France. Octobre 2001.

9.5 Dépôts de brevets

- G. Cabillic, J.P. Lesot, M. Banatre, F. Parain, J.P. Routeau, V. Issarny, T. Higuera, G. Chauvel (Texas Instruments), S. Lasserre (Texas Instruments), D. Dinverno (Texas Instruments) « Data Processing Apparatus, System and Method. ». Demande d'extension de brevet Européen, numéro 0040344.5, Octobre 2001.
- M. Banâtre, P. Couderc. « Procédé et dispositif de téléphonie mobile permettant l'accès à un service contextuel exploitant la position et/ou l'identification de l'usager ». Demande d'extension pour la protection à l'étranger du brevet Français PCT, Mai 2001.

10 Bibliographie

Thèses et habilitations à diriger des recherches

[1] A. Colin, *Estimation de temps d'exécution au pire cas par analyse statique et application aux systèmes d'exploitation temps-réel*, Thèse de doctorat, Université de Rennes I, Octobre 2001.

[2] P. Couderc, *Une approche pour le déploiement incrémental de composants logiciels sur Internet*, thèse de doctorat, Université de Rennes I, décembre 2001.

[3] I. PUAUT, *Supports d'exécution à temps de réponse contraint tolérants aux fautes*, Habilitation à diriger des recherches, Université de Rennes I, Novembre 2001.

Articles et chapitres de livre

[4] G. CABILLIC, J. LESOT, M. BANÂTRE, F. PARAIN, T. HIGUERA, V. ISSARNY, *The Application of Programmable DSPs in mobile Communications*, WILEY press, 2001, ch. A Flexible Java Environment for Wireless PDA Architectures based on DSP Technology, À Paraître.

[5] P. CHEVOCHOT, I. PUAUT, G. CABILLIC, A. COLIN, D. DECOTIGNY, M. BANÂTRE, « Hades : a distributed system for dependable hard real-time applications built from COTS components », *IEEE Transactions on Computers*, 2001, À Paraître.

[6] T. HIGUERA, V. ISSARNY, M. BANÂTRE, G. CABILLIC, F. PARAIN, J. LESOT, « Memory Management for Real-time Java :an Efficient Solution using Hardware Support », *Real-Time Systems journal (Kluwer Pub.)*, 2001, À Paraître.

[7] V. ISSARNY, LNCS 2022, ch. Concurrent Exception Handling.

[8] F. L. MOUËL, F. ANDRÉ, « AeDEn : un cadre général pour une distribution adaptative et dynamique des applications en environnements mobiles », *Revue électronique sur les Réseaux et l'Informatique Répartie (RERIR) 11*, mars 2001.

[9] F. PARAIN, G. CABILLIC, M. BANÂTRE, V. ISSARNY, T. HIGUERA, J. LESOT, « Techniques de réduction de la consommation dans un système embarqué temps-réel », *Techniques et Sciences informatiques (TSI) 20*, 10, 2001, p. 1247–1278.

[10] D. TOUZET, M. BANÂTRE, P. COUDERC, J.-M. MENAUD, F. WEIS, « Side Surfer : Enriching Casual Meeting with Spontaneous Information Gathering », *ACM Computer Architecture Newsletter*, 2001, À Paraître.

Communications à des congrès, colloques, etc.

[11] P. CHEVOCHOT, I. PUAUT, « Conception de systèmes distribués temps-réel strict tolérants aux fautes avec du matériel "sur étagère" », *in : Proc. of the 9th International Conference on Real-Time Systems (RTS'01)*, p. 209–226, Mars 2001.

[12] P. CHEVOCHOT, I. PUAUT, « Experimental evaluation of the fail-silent behavior of a distributed real-time run-time support built from COTS components », *in : Proc. of the International Conference on Dependable Systems and Networks (DSN'01)*, p. 304–313, Göteborg, Sweden, Juillet 2001.

[13] P. CHEVOCHOT, I. PUAUT, « valuation du silence sur défaillance d'un exécutif temps-réel construit à base de composants COTS », *in : Seconde Conférence Française sur les Systèmes d'Exploitation*, p. 61–72, Paris, Avril 2001.

[14] A. COLIN, I. PUAUT, « Analyse de temps d'exécution au pire cas du système d'exploitation temps-réel RTEMS », *in : Seconde Conférence Française sur les Systèmes d'Exploitation (CFSE2)*, p. 73–84, Paris, Avril 2001.

[15] A. COLIN, I. PUAUT, « A modular and retargetable framework for tree-based WCET analysis », *in : Proc. of the 13th Euromicro Conference on Real-Time Systems*, p. 37–44, Delft, The Netherlands, Juin 2001.

[16] A. COLIN, I. PUAUT, « Worst-Case Execution Time Analysis of the RTEMS Real-Time Operating System », *in : Proc. of the 13th Euromicro Conference on Real-Time Systems*, p. 191–198, Delft, The Netherlands, Juin 2001.

[17] T. Higuera, V. Issarny, M. Banâtre, G. Cabillic, F. Parain, J. Lesot, « Region-based Memory Management for Real-time Java », *in : Proc. of the 4th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2001)*, Magdeburg, Germany, mai 2001.

[18] V. Issarny, J.-P. Banâtre, « Architecture-based Exception Handling », *in : Proc. of the 34th Annual Hawaii International Conference on System Sciences (HICSS'34)*, Hawaii, USA, jan 2001.

[19] C. Kloukinas, V. Issarny, « Spin-ning Software Architectures : A Method for Exploring Complex Systems », *in : Proc. of the Working IEEE/IFIP Conference on Software Architecture (WICSA2001)*, p. 67–76, Amsterdam, The Netherlands, Aug 2001.

[20] J. Malenfant, M. Segarra, F. André, « Dynamic Adaptability : the MolèNE Experiment », *in : Proc. of the 3rd International Conference on Metalevel Architectures (REFLECTION 2001)*, Kyoto, Japan, septembre 2001.

[21] D. Mentré, M. Boulkenafeda, V. Issarny, « Towards a Network File System for Roaming Users », *in : Proc. of the 4th CaberNet Plenary Workshop*, p. 67–76, Pisa, Italy, Oct 2001.

[22] B. Randell, R. Stroud, C. Jones, H. Kopetz, D. Powell, V. Issarny, N. Moffat, M.-C. Gaudel, F. von Henke, « DSoS (Dependable Systems of Systems) », *in : Supplement of the 2001 Proc. of the Conference on Dependable Systems and Networks*, p. D28–D31, Göteborg, Sweden, 2001.

[23] D. Touzet, M. Banâtre, P. Couderc, J.-M. Menaud, F. Weis, « Side Surfer : a Spontaneous Information Discovery and Exchanges System », *in : Proceedings of the International Workshop on Ubiquitous Computing and Communications (UCC'01)*, Barcelona, Spain, septembre 2001.

[24] A. Troël, M. Banâtre, P. Couderc, F. Weis, « Predictive scheme for proximate interactions », *in : Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC'2001)*, Mesa, AZ, United States, p. 235–239, avril 2001.

[25] A. Zarras, V. Issarny, C. Kloukinas, K. Nguyen, « Towards a Base UML Profile for Architecture Description », *in : Proc. of the 1st ICSE Workshop on Describing Software Architecture with UML*, p. 22–26, May 2001.

[26] A. Zarras, V. Issarny, « Automating the Performance and Reliability Analysis of Enterprise Information Systems », *in : Proc. of the 16th International Conference on Automated Software Engineering (ASE'01)*, Nov 2001. À Paraître.

[27] A. Zarras, V. Issarny, « UML-Based Modeling of Software Reliability », *in : Proc. of the 1st ICSE Workshop on Describing Software Architecture with UML*, p. 36–40, May 2001.

Rapports de recherche et publications internes

[28] A. Colin, I. Puaut, « A modular and retargetable framework for tree-based WCET analysis », *rapport de recherche n°PI1386*, IRISA, Mars 2001, `http://www.irisa.fr/bibli/publi/pi/2001/1386/1386.html`.

[29] D. Decotigny, I. Puaut, « Artisst : An Extensible Framework for the Simulation of Real-Time Systems », *rapport de recherche n°PI 1423*, IRISA, Novembre 2001, `http://www.irisa.fr/bibli/publi/pi/2001/1423/1423.html`.

Divers

[30] « appliance to appliance, application needs and their modeling using SIS », Deliverable D2.1, collaboration INRIA/ALCATEL, july 2001.

[31] « Conception et réalisation d'un support d'exécution réparti hautement disponible pour applications temps-réel », Fourniture 2.4 du marché 98.34.375.00.470.75.65 INRIA/DGA - Description du prototype, Février 2001.

[32] « Conception et réalisation d'un support d'exécution réparti hautement disponible pour applications temps-réel », Fourniture 2.5 du marché 98.34.375.00.470.75.65 INRIA/DGA - Portage d'une application avionique, Mai 2001.

[33] « Conception et réalisation d'un support d'exécution réparti hautement disponible pour applications temps-réel », Fourniture 2.6 du marché 98.34.375.00.470.75.65 INRIA/DGA - Rapport de synthèse finale, Août 2001.

[34] « État d'avancement à 18 mois », Fourniture du contrat, collaboration INRIA-Texas Instruments, March 2001.

[35] « État d'avancement à 24 mois », Fourniture du contrat, collaboration INRIA-Texas Instruments, October 2001.