Approches et expérimentations autour des composants : applications aux composants logiciels, aux objets d'apprentissages et aux services distribués

Yvon Kermarrec

Année 2005

Mémoire d'Habilitation à Diriger les Recherches

Présentée devant

L'UNIVERSITÉ DE BRETAGNE OCCIDENTALE

par

Yvon Kermarrec

Travail réalisé au sein du département LUSSI
de l'École Nationale Supérieure des Télécommunications de Bretagne

Sujet

Approches et expérimentations autour des composants

applications aux composants logiciels, aux objets d'apprentissages et aux services distribués

Soutenue le 11 Mars 2005, à l'ENST Bretagne

devant le jury composé de

Jacqueline	BOURDEAU	Rapporteur	TeleUniversité - Montréal
Isabelle	DEMEURE	Rapporteur	ENST Paris
Ed	SCHONBERG	Rapporteur	Courant Institute - New York University
Michel	FARINE	Examinateur	ENSAM
Ioannis	KANELLOS	Examinateur	ENST Bretagne
Marcel	LE FLOCH	Examinateur	Université de Bretagne Occidentale
Lionel	MARCÉ	Examinateur	Université de Bretagne Occidentale

"Then I was standing on the highest mountain of them all,
and round about beneath me was the whole hoop of the world.
And while I stood there I saw more than I can tell and
I understood more than I saw ;
for I was seeing in a sacred manner the shapes of all things in the spirit,
and the shape of all shapes as they must live together like one being. "

Black Elk in *Black Elk speaks*

"Dans l'espace qui lie ciel et terre
se cache le plus grand des mystères
comme la brume voilant l'aurore
il y a tant de belles choses que tu ignores.

Dans le temps qui lie ciel et terre
se cache le plus beau des mystères
penses-y quand tu t'endors
l'amour est plus fort que la mort."

Françoise Hardy

Remerciements

Mes premiers hommages vont aux membres du jury de mon habilitation, pour leur confiance, et pour avoir accepté d'évaluer mon travail. Ces quelques mots ne traduisent que bien faiblement toute ma reconnaissance pour les interactions aussi bien scientifiques, qu'éducatives et surtout humaines, que j'entretiens avec eux, parfois depuis plusieurs années.

Je suis très reconnaissant à Jacqueline Bourdeau d'avoir accepté de rapporter ce travail et pour la richesse de nos interactions sur les thèmes d'environnements d'apprentissage. J'espère que nos propositions de recherche seront retenues par les évaluateurs européens et canadiens et nous permettront ainsi de concrétiser nos visions.

Isabelle Demeure a accepté de rapporter mes travaux et je la remercie pour la qualité des interactions et coopérations que nous avons développées depuis mon passage à l'ENST Paris tant en enseignement que dans des projets de recherche. Je la remercie bien vivement pour sa clairvoyance et son soutien à mes travaux.

Je remercie Michel Farine pour m'avoir encouragé à soutenir mon habilitation et surtout pour son amitié depuis maintenant presque 11 ans. Les mots expriment difficilement toute ma reconnaissance et surtout tout ce que j'ai appris avec lui sur le métier de chercheur et les valeurs humaines.

Je suis également très honoré de la participation de Marcel Le Floch, qui dirige l'Ecole Doctorale de l'Université de Bretagne Occidentale. A travers lui, je remercie aussi l'Université pour m'avoir donné l'opportunité d'enseigner et ceci dès ma première année de doctorat, ouvrant ainsi ma passion pour cette activité.

Je remercie Ioannis Kanellos pour son soutien et ses initiatives lors de mon intégration au département IASC de l'ENST Bretagne. Son amitié et nos discussions m'ont permis de découvrir une partie de la thématique de l'indexation, qui lui tient au coeur, mais aussi de me recentrer sur les aspects humains de l'informatique.

Lionel Marcé a ouvert plus d'une voie et m'a permis de mener à bien mon projet d'habilitation. Je lui suis naturellement très reconnaissant pour ses soutiens et encouragements, qu'il a manifestés depuis mon arrivée à Brest.

Je suis enfin très reconnaissant à Ed Schonberg à plus d'un titre. D'une part, il m'a donné une chance inouïe en m'intégrant dans son équipe de New York University et, surtout, il m'a témoigné son amitié et soutien à de très nombreuses occasions, me permettant ainsi d'intégrer mes idées et approches dans l'environnement GNAT, produit par son équipe. Ed est pour moi un mentor exceptionnel et je suis extrêmement fier et touché de sa présence.

Mes activités de recherche, d'enseignement et de coordination sont diverses et variées et le lecteur intéressé trouvera dans ce document un bref survol des activités réalisées et des résultats obtenus. Je tiens naturellement à souligner que ces résultats sont le fruit de mon travail et sont également fortement liés aux équipes que j'ai intégrées ces dernières années. Je reconnais les contributions de mes collègues ci-dessous sans lesquels je n'aurai pas pu atteindre les résultats actuels.

J'adresse un merci particulier à Jean-Pierre Barthélemy, pour m'avoir intégré dans son département et m'avoir donné toutes les chances pour réussir mon changement d'orientation en recherche et en enseignement. Sa clairvoyance et ses appuis ont été exceptionnels et constants pour débrouiller des situations complexes et pour soutenir mes engagements et paris.

J'adresse également mes remerciements les plus vifs à Gilles Coppin, qui exerce ses talents de chef de département. Je le remercie pour la confiance qu'il me témoigne et surtout pour la liberté dont je jouis pour le choix de mes activités, en recherche et en enseignement. Son engagement à mes côtés est précieux.

Enfin, la direction de l'ENST Bretagne m'a permis de développer mes activités internationales et pour cela je tiens à remercier tout particulièrement Gilbert Lainey et Ian Simpson. Je suis très reconnaissant à Gilbert et à Ian pour m'avoir confié le réseau EUNICE et donné un contexte favorable au montage et développement de coopérations de recherche et d'enseignements avec plusieurs équipes européennes et internationales. Je suis enfin très reconnaissant à Bernard Ayrault et Alain Hillion pour m'avoir permis de mener à bien mon projet de sabbatique à Vancouver et d'avoir accédé à ma demande de changement d'affectation au sein de l'ENST Bretagne.

Mes thésards ont également eu un rôle majeur et des impacts certains sur mes activités de ces quinze dernières années. Je leur suis reconnaissant pour la confiance qu'ils m'ont manifestée et pour le bout de chemin que nous avons mené ensemble avec succès malgré

les écueils. Ces doctorants sont devenus bien vite des collègues et des amis. Avec Laurent Pautet, Laurent Nana, Martin Heusse et Oussama Zein, j'ai vécu des moments exceptionnels et je les en remercie... et souhaite continuer avec eux les activités entreprises.

Je remercie enfin mes très nombreux collègues et amis que j'ai découverts au fil de mes intégrations dans mes différentes équipes à Brest, York, New York, Paris et Vancouver. Mes collègues du département informatique de l'ENST Paris (avec une mention particulière à Anne Germa, Olivier Hudry, Irène Charon, Philippe Dax), de New York University (with special thanks to Robert Dewar, Dennis Shasha, Kurt Benhke, Bernard Banner), du département informatique de l'ENST Bretagne (Paul Geffroy, JF Mercel, André Lasquellec, JD Laisné), de Raytheon (with special thanks to Thierry Schneider, Vincent Celier, Paul Doherty). Je les remercie pour leur amitié tant de fois manifestée et surtout pour la chance d'avoir accompli un bout de chemin en leur compagnie, que ce soit sur des activités scientifiques, pédagogiques ou sociales ... ou sportives.

J'ai également eu, au fil des années, la chance de collaborer sur des projets de recherche avec de nombreuses personnes et la liste sera partielle. Pierre Dissaux (de TNI), Scott et Laura Moody (de Boeing), Joel Sherril (de OAR Corp.), Anthony Gargaro (de Computer Science Corp.), Tucker Taft (de Intermetrics), Philippe Kruchten (de Rational), JP Rosen (d'AdaLog), Pierre Rollin et Annie Gravey (alors à France Télécom), Patrick Farail (d'Airbus Industries), John Scott et Paul Fostner (de British Telecom Labs), O. Mehl (de Karlsruhe), R. Braek (de NTNU), etc. Merci pour le bout de chemin fait ensemble ...

Merci aussi à mes collègues universitaires avec lesquels j'ai pu eu des échanges passionnants sur l'art d'enseigner nos domaines (parfois) ardus. Les échanges sont nombreux avec mes collègues du réseau EUNICE tout comme avec ceux des Universités avec qui j'ai coopéré (University of York, UPC Barcelone, Institute of Education, Texas AM University, UC Irvine, Poznan University of Technology, etc.).

Un grand merci aussi à Oussama Zein, mon dernier thésard en cours, et à Marcel Duault pour leur lecture attentive et vigilante des versions successives de ce mémoire.

Finalement, je pense avoir trouvé un cadre exceptionnel au sein du département IASC / LUSSI pour mes activités professionnelles. L'amitié et l'ambiance propices aux interactions et au développement personnel, sont certes le fruit de JP Barthélemy et G. Coppin, mais ces synergies sont surtout possibles par la contribution de chacun des membres du département. Ph. Tanguy, S. Garlatti, Ph. Picouet, I. Kanellos sont des amis et collègues que j'apprécie beaucoup. J'adresse également une mention spéciale à Ghislaine pour sa patience à m'expliquer les subtilités des processus administratifs ... qui souvent me dépassent.

Merci aussi à ma famille et à mes parents, en particulier, pour m'avoir donné leur amour et la possibilité de mener à bien mes études.

Enfin, si mes mots n'expriment pas toute l'intensité et la richesse de mes interactions avec mes collègues et amis, ils sont encore trop légers pour remercier mon épouse, Isabelle, et mes enfants, Guillaume et Margaux, ... sans qui je ne suis pas grand chose !

Table des matières

Table des figures

CHAPITRE 1

Introduction et parcours de recherche

Dans le cadre de mes travaux en vue de l'obtention du doctorat en informatique, je me suis intéressé à la notion de composants logiciels réutilisables pour la simulation des systèmes distribués avec le langage Ada. Cette thématique à base de composants réutilisables a été explorée et j'avais à l'époque deux objectifs principaux :

- **Le développement d'une base de composants logiciels pédagogiques**
  En effet, les recherches en algorithmiques distribuées avaient déjà été bien entamées et de nombreux algorithmes avaient été proposés afin de résoudre les problèmes liés au contrôle distribué (gestion de l'exclusion mutuelle et détection de la terminaison, par exemple). Ces travaux se limitaient à des études théoriques, très souvent. D'un autre côté, les architectures distribuées commençaient également à être disponibles dans les universités et centres de recherche et nous connaissions alors tous les bénéfices du passage d'une architecture centralisée et monolithique à une architecture distribuée. L'ENST de Bretagne intégrait ces thèmes d'algorithmiques et de machines distribuées dans le cursus car leur usage devait être largement diffusé dans les systèmes de télécommunications. Cependant, ce cursus se limitait à une étude théorique et il n'y avait à cette époque ni travaux pratiques, ni applications. Afin de mettre en pratique les aspects liés à la distribution, nous avions donc comme premier objectif de concevoir et réaliser des briques logicielles qui puissent être facilement assemblées pour former des programmes de simulation et d'expérimentation.

- **L'expérimentation de nouvelles approches de conception de logiciels et de simulation distribuée**
  Le langage Ada 83 venait d'être normalisé deux ans avant le démarrage de ma thèse et son approche, pour la réutilisation et le développement de grands logiciels, apparaissait comme des réponses aux attentes des industriels mais aussi des universitaires. Le langage met l'accent sur le génie logiciel et sur la notion de composants logiciels. À partir de l'étude de ce langage, nous avons voulu aller plus loin et voir comment des composants réutilisables pouvaient être définis et surtout comment nous pouvons les

assembler et les intégrer dans différents environnements.

Ces développements se sont traduit par le développement de nombreux composants logiciels qui ont été utilisés par plusieurs générations d'étudiants et qui leur ont permis d'avoir un cadre d'apprentissage des approches et solutions liées aux systèmes distribués.

1.1 Autour de la réutilisation

Depuis mes études de doctorat, j'ai donc entrepris la définition et la réalisation d'une boîte à outils de composants logiciels en vue de la simulation et de l'apprentissage des systèmes distribués. Ce domaine est vaste et nous pensions qu'une approche composants permettrait d'atteindre plus facilement nos objectifs pédagogiques et fonctionnels. En effet, une association de composants élémentaires permet de produire un objet plus complexe, qui combine les fonctionnalités de ses composants, et répond à des impératifs de coûts mais aussi et surtout de qualité.

Ces composants ont été utilisés dans un contexte d'enseignement et ont servi de support dans le cadre de travaux pratiques en systèmes distribués. La notion de composants de simulation paramétrés s'est révélée être une approche innovante pour les étudiants puisqu'ils pouvaient faire varier aisément les propriétés comportementales du réseau (introduction de perte ou de déséquencement de messages, par exemple) et surtout se servir des composants disponibles pour concevoir et développer des entités de plus haut niveau (développements de composants à partir de composants de contrôle distribué et de communication, en particulier).
Nous avons enfin souhaité que ces composants suivent les standards ou normes et de ce fait leurs interfaces standardisées en font des briques destinées à des assemblages dans de nombreux contextes : recherche mais aussi industriels. Du fait de l'existence de la communauté GNAT et de la licence GNU de la "Free Software Foundation", nous avons diffusé ces composants dans la communauté afin qu'ils puissent être réutilisés et intégrés dans des environnements divers et variés.

Dans le contexte du composant logiciel, je me suis intéressé principalement à l'assemblage de composants. La définition d'une architecture logicielle et d'un schéma d'assemblage apparaissent incontournables dès lors que le composant logiciel devient la brique de base.

Le nombre de composants grandissant, j'ai alors été sensibilisé à deux questions :

- ◇ comment décrire un composant (quelles propriétés ou caractéristiques retenir) et comment l'indexer pour le retrouver ?
- ◇ comment assembler des composants et assurer la cohérence de l'ensemble et la validité de l'assemblage ?

Mes activités de recherche se sont naturellement orientées vers ces deux questions avec une recherche plus importante sur la notion de composition et d'assemblage.

Mes travaux de recherche se sont ensuite orientés dans le domaine des nouvelles technologies pour l'enseignement et les services web, du fait de mon changement d'affectation au sein de l'ENST Bretagne. Les deux questions, liées aux composants logiciels, sont réapparues dans le contexte des composants de cours et des composants pour les services, et j'ai décidé d'y répondre, fort de mon expérience dans le domaine des composants logiciels. Ces travaux de recherche se sont déroulés dans le cadre d'une équipe élargie (celle de mon département à l'ENST Bretagne et d'autres au niveau national et européen) et de projets de R&D.

Ce document présente donc mes approches et démarches pour adresser le problème de la réutilisation de composants logiciels, de composants de cours et de services. Pour chacun de ces domaines d'applications, je présenterai :

- ◇ **Une description de la nature des différents composants identifiés.**
  Je me suis pour cela intéressé à la notion de granularité d'un composant et à l'indentification de critères pour qu'il soit réutilisable. Nous verrons en particulier que la conception d'un composant réutilisable nécessite une analyse approfondie puisqu'il faut en particulier anticiper les contextes futurs d'un composant, que l'on pourrait concevoir comme ad hoc, sinon.

- ◇ **Une approche de description et d'indexation adaptée à ces composants.**
  Ces deux notions sont impératives lorsqu'il s'agit de gérer un grand nombre de composants ou de mettre en pratique une approche de partage de composants au sein d'une communauté. Les approches classiques visent à fournir en plus du composant une description plus ou moins formelle du composant, des services offerts et des pré requis qu'il impose de son environnement. Cette description se révèle bien insuffisante et les limites de la recherche par seuls mots clés sont connues. Notre approche vise à proposer un schéma de méta données, qui permettra de décrire les composants de manière très précise. Ce schéma prend en compte des propriétés statiques (comme l'auteur, la date de création ou de modification du composant, etc.) ainsi qu'une description de ce que fait le composant et de son comportement. Nous aborderons en particulier ce dernier point dans la suite de ce document. La notion de partage nécessite enfin des consensus sur la terminologie utilisée ainsi que la définitions de pratiques communes.
  Un autre point relié à l'indexation est celui de la recherche de composants dans une base qui peut comprendre plusieurs milliers de composants. Nous souhaitons proposer des mécanismes flexibles que l'utilisateur peut contrôler afin d'affiner sa recherche ou au contraire l'élargir, si aucun composant ne répond à ses critères.

◇ **Un modèle de composition et d'assemblage de ces composants.**
Nous avons indiqué que la conception d'un composant réutilisable nécessite des actions supplémentaires afin justement de le rendre intégrable dans d'autres contextes et de le rendre utilisable par d'autres usagers. La nature même d'un composant réutilisable impose de l'assembler et de le combiner avec d'autres afin de proposer de nouveaux services. Nous présenterons les différents modèles de composition que nous avons retenus et montrerons comment les propriétés (ou index) de chaque composant peuvent être utilisées afin de favoriser l'assemblage cohérent.

1.2 Structure du document

Ce document d'Habilitation à Diriger des Recherches se présente sous la forme d'une synthèse de mes différents travaux de recherche depuis mon doctorat et une description de mes différentes activités en enseignement, recherche et autres tâches de coordination.

Il comprend en particulier :

◇ Une synthèse de mes activités dans le domaine des composants logiciels.
◇ Une synthèse de mes activités dans le domaine des technologies pour l'enseignement.
◇ Une synthèse de mes activités dans le thème des services des systèmes distribués.
◇ Une conclusion sous la forme de perspectives et de travaux possibles en R&D dans un futur proche.
◇ Un curriculum vitae étendu figurant en annexe : il met en lumière mon parcours depuis mon doctorat et est complété par la liste de mes publications.

CHAPITRE 2

Composants logiciels pour l'enseignement et les grands projets

La première activité que je vais développer est liée aux composants logiciels et a démarré lors de mes travaux en thèse. Dans un premier temps, je vais dégager les traits caractéristiques des composants logiciels et détailler en particulier la notion d'interface et de réutilisation. Je montrerai ensuite comment le langage Ada permet d'exprimer ces traits et de définir des composants logiciels réutilisables et quels sont les approches d'assemblage. Dans la section 3, je présenterai la bibliothèque de composants logiciels qui a été réalisée pour la simulation de systèmes distribués. En section 4, je montrerai que cette notion de composants présente aussi des intérêts pour la conception de systèmes de bas niveau. En conclusion, je présenterai mes travaux autour du projet RNTL COTRE et indiquerai les perspectives de recherche dans ce thème.

2.1 La notion de composant logiciel

2.1.1 Une définition du composant logiciel

De nombreuses définitions du terme composant logiciel existent et il serait vain d'espérer trouver un consensus et une définition unique. Cependant, à partir de quelques références, nous allons essayer de cerner quelques traits caractéristiques de ces composants logiciels.

Une définition du composant "idéal" est fournie dans la documentation technique de COM (Component Object Model de Microsoft) :

> "A component is a reusable piece of software in binary form that can be plugged into other components from other vendors with relatively little effort" [1]

Cette première définition insiste sur la propriété d'un composant qui apparaît comme une unité destinée à être assemblée et à fonctionner avec d'autres. Elle insiste sur un modèle de composition à l'exécution, puisque l'assemblage fait ici référence à du code binaire. B. Meyer

[2] resitue deux autres définitions qu'il qualifie de définition large ("wide definition") et de définition étroite ("narrow definition").

- ◇ La définition étroite du composant correspond à celle fournie par C. Szyperski [3] qui restreint le composant à des unités binaires, c'est-à-dire exécutables par une machine.
- ◇ La définition large [4] ne précise pas la nature du composant qui peut être une classe, une unité binaire, un paquetage, par exemple, mais est surtout orientée client (*"client oriented software"*). Les composants peuvent être utilisés par d'autres éléments de programmes (ses clients), et les auteurs et clients de ces éléments de programmes ne connaissent pas nécessairement les auteurs des composants.

Tout comme [5], nous retiendrons cette seconde approche du fait de sa globalité et généralité. Enfin B. Meyer resitue l'approche de développement à base de composants en tant qu'évolution naturelle de l'approche objet : il ne s'agit donc pas de remise en question des travaux de la communauté objet.

Dans [6], I. Crnkovic et M. Larsson indiquent que la description d'un composant doit comporter deux parties : un ensemble d'interfaces permettant de décrire les interactions possibles avec d'autres composants (voire de décrire aussi les conditions liées à ces interactions) et l'environnement, et du code exécutable qui peut être associé au code d'autres composants via les interfaces. Ils précisent enfin que, pour améliorer la qualité d'un composant, son auteur peut rajouter les éléments suivants :

- ◇ une spécification des caractéristiques non fonctionnelles.
- ◇ un code permettant de valider la connexion d'un composant
- ◇ des informations complémentaires comme les contextes d'utilisation.

Ces définitions soulignent toutes 2 points essentiels :

- ◇ la raison d'être du composant est la réutilisation
- ◇ le composant doit interagir avec d'autres composants pour former des entités plus élaborées et proposer des services étendus.

2.1.2 La réutilisabilité

Le développement à base de composants est basé sur un concept fondamental : la réutilisabilité. Ce concept repose sur des observations de domaines extérieurs au logiciel. L'industrie électronique, par exemple, utilise largement les composants pour produire des appareils aux fonctionnalités plus complexes que celles des composants d'origine ou pour produire d'autres composants. L'approche CBSD (*"component-based software development"*) [7] vise à construire des systèmes logiciels complexes à partir de composants déjà existants.

Pour justifier une telle approche à base de réutilisation, B. Meyer énonce six points forts que nous présentons avec quelques éléments d'explication :

- ◇ Respecter les échéances en ayant la capacité de produire des logiciels rapidement. Le volume de logiciel à produire sera limité, puisqu'une partie des composants est déjà réalisée.
- ◇ Diminuer l'effort de maintenance en déportant la maintenance des composants aux fournisseurs de ces composants. Cet argument renforce l'intérêt économique de la réutilisation puisque la maintenance nécessite un budget conséquent surtout pour les systèmes informatiques ayant une longue durée de vie.
- ◇ Augmenter la fiabilité en utilisant des composants en provenance de fournisseurs réputés et compétents dans leurs domaines de spécialisation. De plus, la fiabilité pourra être augmentée du fait de l'intégration et de la réutilisation de ces composants par un grand nombre de clients. Plus le composant est réutilisé et donc testé, plus il sera fiable et intégrable dans divers environnements.
- ◇ Accroître l'efficacité en laissant le concepteur du composant (un spécialiste du domaine) retenir la meilleure solution. Il s'agit ici par exemple de reconnaître les spécialités métier qui nécessitent expertises et compétences dans des domaines parfois pointus.
- ◇ Assurer la cohérence en développant d'une manière uniforme avec les composants.
- ◇ Assurer l'investissement en amortissant le développement d'un composant sur plusieurs projets.

2.1.3 Adaptabilité et extensibilité

La réutilisabilité est à elle seule une caractéristique importante mais pas suffisante pour le développement à base de composants. Les propriétés complémentaires sont l'adaptabilité et l'extensibilité. Celles-ci feront qu'un composant développé dans un certain environnement, en identifiant certaines particularités, puisse être adapté à d'autres besoins, à d'autres contextes et environnements.

Pour répondre à ces aspects complémentaires, B. Meyer donne cinq exigences sur les structures de modules. Parmi elles, nous trouvons par exemple la prise en compte de la variation de type : un composant qui met en œuvre le type abstrait "pile" devra s'adapter à différentes natures de types d'éléments à empiler.

2.1.4 Interactions entre composants

La construction de logiciels à partir de composants nécessite l'assemblage de ces composants et surtout leur couplage afin de permettre leurs interactions.

Nous n'allons pas détailler les différentes architectures, mais un des aspects essentiels est la notion d'interface de composants. L'interface d'un composant est la caractéristique du composant qui précise comment l'utiliser. En effet, un composant est censé agir comme une boîte noire, c'est-à-dire comme une entité rendant certains services sans pour autant

divulguer la manière dont ce service est rendu, cette information n'étant pas nécessaire aux utilisateurs du composant. De plus, cacher les détails de l'implantation permet de mettre à jour un composant sans que cela n'affecte les applications en place utilisant déjà ce composant.

La description de l'interface d'un composant comprend différents éléments qui peuvent inclure :

- le nom des opérations (des services) fournies,
- leurs paramètres et
- les types de ces paramètres.

À ces informations viennent s'en ajouter d'autres qui se retrouvent dans différents modèles de composants. Dans CCM (*CORBA Component Model*, par exemple, il existe la notion de facettes, des interfaces différentes fournies par le composant que le client utilisera pour interagir, ou de réceptacles, points de connexions avec un autre composant. Les autres informations qui peuvent également être pertinentes dans un composant sont les contrats.

2.1.5 Synthèse

Les avantages des développements à base de composants sont la réutilisabilité, l'adaptabilité et l'extensibilité qui favorisent la qualité et la fiabilité tout en diminuant les coûts de développement.

I. Crnkovic et M. Larsson [6] présentent néanmoins cinq risques potentiels liés aux développements à base de composants :

- Les temps initiaux de développement et les efforts nécessaires sont accrus. Penser réutilisation nécessite pour le concepteur de prendre en compte des contextes futurs et possibles du composant qu'il souhaite développer.
- Les exigences de réutilisabilité sont souvent peu claires et ambiguës. Cette difficulté est principalement liée à la prise en compte de contextes futurs et possibles d'utilisation. Il est difficile de les anticiper et donc de les décrire avec précision.
- L'utilisation et la réutilisation sont des notions antagonistes. La réutilisation nécessite une définition paramétrée du composant ce qui peut le rendre plus difficile à utiliser que le composant, jetable.
- Les coûts de maintenance des composants sont plus élevés. Ceci est relié à la notion de gestion des exigences : comme un composant intervient dans divers projets/produits, après modification d'un composant, il faut vérifier sa conformité avec toutes les exigences.
- La fiabilité des applications est sensible aux changements. Tout changement, au niveau de l'application ou d'autres composants, peut avoir un impact sur la fiabilité d'un composant par une méconnaissance de toutes les caractéristiques du composant. Ce domaine est particulièrement bien connu dans le domaine des logiciels et services télé-

coms sous le terme *feature interaction*.

Les avantages énoncés précédemment confirment l'intérêt d'une approche à base de composants. Néanmoins, cette démarche n'est pas toujours facile à suivre dans la pratique. En effet, l'assemblage des "pièces" logicielles pour constituer une application efficace et facile à utiliser nécessite un travail qui peut s'avérer fastidieux car il met en cause deux parties, dans ce contexte, dissociées : les producteurs et les consommateurs des composants logiciels [5].

2.2 Composants logiciels et Ada

2.2.1 Le langage Ada

Ada est avant tout un langage de génie logiciel conçu par le ministère américain de la défense (US DoD) pour réaliser des applications logicielles de grande ampleur. Le langage est généraliste (son domaine naturel est celui des systèmes embarqués et temps réel mais il s'adresse aussi aux applications scientifiques ou de gestion), unificateur (du fait de la prise en compte du temps réel et des approches objets, par exemple) et normalisé (la première norme du langage date de 1983 et la seconde version de 1995).

Ada est largement utilisé (voire incontournable) en avionique et en informatique embarquée ainsi que pour le contrôle de trafic (aérien, ferroviaire) où la fiabilité est cruciale. Il est aussi apprécié quand le code à développer est conséquent (donc très difficile à maintenir) et pour les projets de longue durée.

Le langage, en digne successeur de Pascal et de Modula, était naturellement destiné à être le support de l'enseignement en informatique, mais son utilisation en formation est limitée à ce jour. Dès 1992, le US DoD a souhaité que soit développé un environnement complet et de qualité pour la future révision du langage Ada et ceci afin d'encourager la diffusion du langage au sein des établissements universitaires. Il s'agissait en particulier d'éviter ce qui s'était passé avec Ada 83 [8] : des compilateurs et environnements trop chers pour un contexte d'apprentissage avaient fermé les portes académiques à Ada. Une autre intension du US DoD était aussi de favoriser l'émergence d'activités de recherche et développements autour d'Ada. Dans ce contexte, l'équipe de New York University des Professeurs R. Dewar et E. Schonberg a réalisé l'environnement GNAT [9] et un compilateur Ada 95 [10] de la famille des compilateurs GCC, qui ont fait l'objet depuis de nombreuses contributions et extensions par la communauté de ses utilisateurs.

2.2.2 Typage fort et cohérence des assemblages

Un type Ada définit un ensemble de valeurs muni d'un ensemble d'opérations portant sur ces valeurs. Il représente une entité du domaine de problème, et non une entité machine. Le programmeur exprime les exigences de son problème à l'aide d'une riche collection de constructeurs de types.

Un ensemble de règles de compatibilité de types permet de déterminer les constructions et assemblages possibles et ceci dès la compilation. Si de telles vérifications ne pouvaient avoir lieu à la compilation, du fait par exemple de valeurs et bornes connues uniquement lors de l'exécution, un code spécifique serait inséré afin de réaliser les tests à l'exécution.

Le modèle de typage fort est naturellement très contraignant pour le programmeur puisque toute conversion doit être explicite mais garantit la détection de très nombreuses erreurs et ceci dès les premières phases de développement.

2.2.3 Organisation de projet et bibliothèques hiérarchiques

Une notion centrale en Ada est celle de paquetage (*"package"*). Le paquetage permet de regrouper dans une seule entité des types de données, des objets, et des sous-programmes (procédures et fonctions) manipulant ces objets. Le paquetage comporte une partie visible (ou publique) comportant les informations utilisables à l'extérieur du paquetage, et une partie cachée (ou privée) qui regroupe les détails d'implémentation auxquels les utilisateurs du paquetage n'ont pas accès.

L'interface du paquetage est décrite par une spécification (*"package specification"*) et le code associé à une éventuelle initialisation est décrite dans une section séparée : le corps (*"package body"*).

Les paquetages peuvent être compilés séparément, et seule leur spécification est nécessaire pour permettre l'utilisation du paquetage. On réalise ainsi une séparation complète entre les spécifications d'une unité fonctionnelle et son implémentation. De plus, les règles très rigoureuses de recompilation garantissent que, d'une part, il n'est pas possible d'utiliser une information cachée d'un paquetage, et d'autre part qu'en cas de modification de la partie "spécification", la recompilation des unités qui utilisaient cette spécification est obligatoire : le langage garantit donc la cohérence globale des différentes unités constituant un programme.

Ada 95 a enrichi les fonctionnalités des paquetages avec la notion de paquetages hiérarchiques. Il est possible de créer des paquetages "enfant" qui rajoutent des fonctionnalités à des paquetages existants sans avoir à toucher à leurs "parents", donc sans recompilation des utilisateurs de ces parents. Ceci permet une meilleure séparation en fonctionnalités "princi-

pales" et fonctionnalités "annexes", facilitant l'utilisation aussi bien que la maintenance.

Par exemple, lorsque l'on fait un type de données abstrait, on ne souhaite généralement pas mélanger les propriétés de base avec des opérations plus spécifiques ou les opérations d'entrées-sorties. Nous indiquons ci-après la spécification d'un type abstrait permettant la définition et la manipulation de nombres complexes, puis nous l'étendons à l'aide d'un paquetage enfant.

```
package Nombres_Complexes is
      type Complexe is private;

      -- Opérations sur les nombres complexes

      function "+" (c1, c2 : Complexe) return Complexe;


private
      -- l'organisation interne du type n'est pas accessible de l'extérieur
      type Complexe is ...
end Nombres_Complexes;


-- Définition d'un paquetage enfant consacré aux E/S sur les complexes

package Nombres_Complexes.IO is

-- Opérations Entrées-sorties sur les complexes

end Nombres_Complexes.IO;
```

2.2.4 Généricité et réutilisation

Une autre propriété très importante du langage Ada est la notion d'unités génériques. Il s'agit d'unités paramétrables, qui permettent de définir un algorithme indépendamment des types d'objet manipulés. L'unité générique n'est pas utilisable par elle-même : elle constitue une sorte de patron dont on doit en faire une instanciation en précisant les valeurs des paramètres génériques et resituant ainsi dans un contexte précis. Par exemple, si on s'intéresse à une opération qui permute ses deux arguments, on s'aperçoit que le même traitement est nécessaire pour permuter des caractères ou des nombres complexes. La généricité permet justement de prendre en compte cette variation du type des paramètres. La traduction en Ada d'une telle opération repose sur une procédure générique, avec comme paramètre le type des données. Dans le code indiqué ci-dessous, la déclaration du type Elem comme "private" signifie que n'importe quel type pour lequel l'affectation et la comparaison d'égalité sont définis peut être utilisé.

```
generic
      type Elem is private;
```

```
procedure Swap (X,Y : in out Elem);

procedure Swap(X,Y : in out Elem) is
      Temp : Elem;
begin
      Temp := X;
      X := Y;
      Y := Temp;
end Swap;
```

Cette procédure générique peut ensuite être instanciée et adaptée à un contexte d'utilisation. Dans notre exemple, cette adaptation consiste à préciser le type des données. La procédure ainsi instanciée peut désormais être utilisée par le programmeur et s'appliquera aux entités de type Couleur :

```
procedure Swap_Couleur is new Swap(Elem => Couleur);
```

La généricité ne se limite pas en Ada à des variations sur les types. Le lecteur intéressé trouvera dans [11] une présentation précise sur le thème de la généricité. L'exemple suivant donne la spécification d'une unité générique de tri de tableau. Les paramètres génériques sont le type d'indice du tableau, le type de composant, le type du tableau lui-même, et une fonction de comparaison. Cette fonction de comparaison est munie d'une valeur par défaut (clause is <>), signifiant que si l'utilisateur ne fournit pas de valeur, la comparaison prédéfinie (si elle existe) doit être utilisée. Les avantages de cette démarche sont évidents : à partir d'un algorithme écrit une fois pour toutes, et que l'on pourra optimiser soigneusement, il est possible d'obtenir instantanément les fonctions de tri sur n'importe quel type de données, avec n'importe quel critère de comparaison.

```
generic
       type Index  is (<>);
       type Composant is private;
       type Tableau is array(Index) of Composant;
       with function "<"(X,Y : Composant) return Boolean is <>;
procedure Tri (Tab : in out Tableau);
```

À partir de ce modèle de composant, il est également possible de l'instancier en fournissant les différentes valeurs des paramètres génériques. Les instanciations ci-après fournissent ici deux exemples d'instanciations du même algorithme pour la définition d'une fonction de tri ascendant et de tri descendant. Les deux nouvelles procédures sont issues du même algorithme et toute modification de l'algorithme sera répercutée sur toutes les instanciations.

```
type Int is range 1..10;
type Arr is array(Int) of Float;
procedure Tri_Ascendant  is new
     Tri (Index => Int, Composant => Float, Tableau => Arr);
procedure Tri_Descendant  is new
    Tri (Index => Inf, Composant => Float, Tableau => Arr, "<"  => ">");
```

Les génériques constituent un outil très puissant et expressif pour la réutilisation car ils permettent de définir des algorithmes généraux applicables à des types identifiés (des entiers, des réels, etc.), des types dotés de certaines opération si (comme l'affectation ou la comparaison). De plus, la définition de sous-programmes en paramètres génériques explicite les opérations qu'un environnement doit offrir lors d'une instanciation.

2.2.5 Définition de composants logiciels avec Ada

La généricité en Ada a été introduite afin de permettre la définition d'unités de programmes qui soient paramétrables et adaptables à différents contextes. À ce titre, ces unités génériques constituent des composants logiciels compatibles avec les définitions fournies en début de ce chapitre. De plus, le typage fort et les règles associées à la compatibilité des types garantissent une cohérence des assemblages et de ce fait contribuent à la solidité de ces assemblages. Enfin, la séparation de la spécification et de la mise en œuvre du code (et de l'algorithme) sert d'interface de haut niveau pour le composant logiciel. La diversité des paramètres génériques (des constantes, des sous programmes, des variables et même d'autres paquetages) apporte des variations intéressantes pour la définition de connecteurs [12].

Je n'ai pas abordé dans cette section le modèle de tâches d'Ada, ni celui du modèle objet, qui sont deux points forts du langage. Naturellement, l'association de ces différents traits du langage fournit les moyens pour exprimer aussi bien des types abstraits que des machines abstraites et élargit ainsi la notion de composant. La machine abstraite, association d'un objet et d'une tâche imbriquée, permet d'exprimer des fonctions de contrôle sur l'appel des méthodes, par exemple.

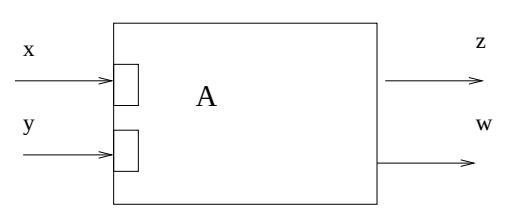
2.2.6 Un exemple de définition et d'assemblage de composants

Nous souhaitons illustrer par un exemple la facilité de composition des composants en utilisant le langage Ada. Nous allons en effet définir un modèle de composant A à partir de 3 autres modèles de composants appelés B, C et D.

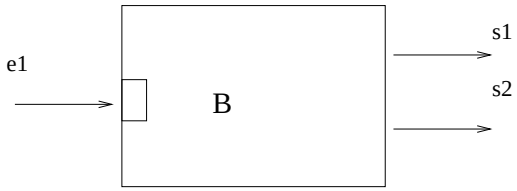
Chaque modèle de composant se présente sous une forme de boîte noire avec deux types d'interfaces comme dans la figure 2.1 :

- ◇ une interface entrante : elle fournit les services qui peuvent être appelés de l'extérieur. Par exemple, le composant C offre les services e1 et e2.
- ◇ une interface sortante : qui traduit les dépendances du composant vis-à-vis de son environnement. Ce dernier doit lui fournir les procédures Z et W.

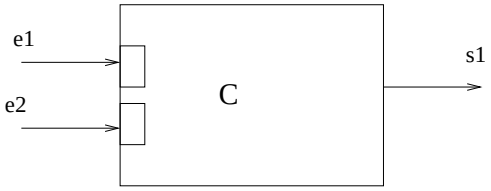
L'assemblage des composants consiste tout simplement à connecter les différentes interfaces entrantes / sortantes des composants et ceci est réalisé par l'intermédiaire de connecteurs. L'architecture globale à base de ces composants est présentée dans la figure 2.2. Nous



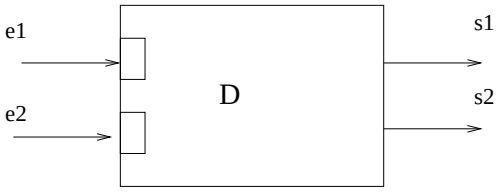

```
generic
  with procedure z ;
  with procedure w;
package a is
  procedure x;
  procedure y;
end a;
```




```
generic
  with procedure s1;
  with procedure s2;
package B is
  procedure e1;
end b;
```




```
generic
  with procedure s1;
package C is
  procedure e1;
  procedure e2;
end C;
```




```
generic
  with procedure s1;
  with procedure s2;
package D is
  procedure e1;
  procedure e2;
end D;
```

FIG. 2.1: Spécification de composants logiciels

détaillons ci-après la mise en oeuvre d'une connexion puis le code de l'assemblage global. Pour connecter les composants B et C, nous devons relier l'interface de sortie de B (soit B.S1) avec l'interface entrante de C (soit E1) : il faut donc que B sache que B.S1 correspond à un appel de E1 de C. Pour cela, il faut simplement rajouter un connecteur lors de l'instanciation du composant B : le connecteur apparaît comme une procédure limitée à une seule instruction (c'est-à-dire l'appel de la procédure E1 définie dans C).. À noter que A apparaît lui aussi comme un modèle de composant et qu'il peut donc être assemblé avec d'autres ou placé dans un environnement.

Le code correspondant à la mise en œuvre du modèle de composant A se présente alors de la manière suivante :

```
-- spécifications des connecteurs

  procedure Connecte_B_C;
  procedure Connecte_B_D;
```

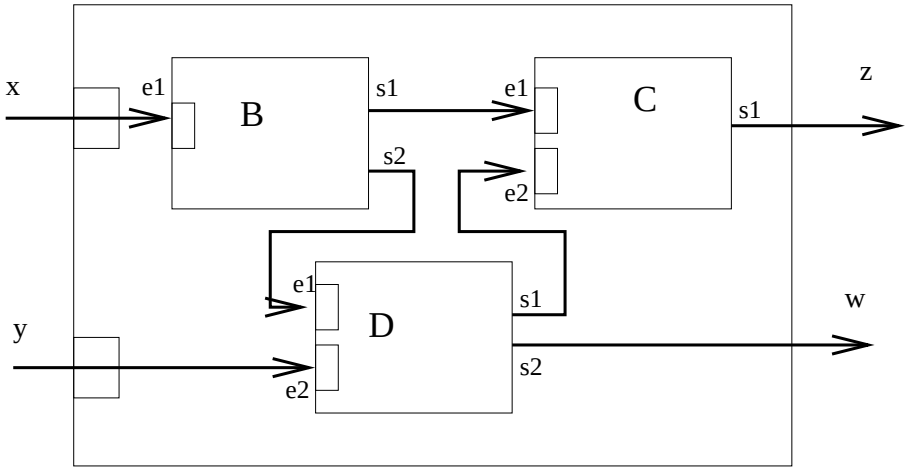


FIG. 2.2: Assemblage de composants logiciels

```
procedure Connecte_D_C;

-- instanciations des paquetages

package Un_B is new B (s1 => Connecte_B_C, s2 => Connecte_B_D);
package Un_C is new C (s1 => z);
package Un_D is new D (s1 => Connecte_D_C, s2 => w);

-- définition des connecteurs entre composants

procedure Connecte_B_C is
begin
  Un_C.e1;
end Connecte_B_C;

procedure Connecte_B_D is
begin
  Un_D.e2;
end Connecte_B_D;

procedure Connecte_D_C is
begin
 Un_C.e2;
end Connecte_D_C;


-- les interfaces entrantes sont reliées aux composants B et D

procedure X is
begin
  Un_B.e1;
end X;

procedure Y is
```

```
  begin
    Un_D.e1;
  end Y;
```

Le modèle de composant A peut enfin être instancié et intégré dans un environnement qui lui fournit au moins les procédures Z et W.

```
-- définition des connecteurs et services fournis par l'environnement
procedure Fournisseur_Z is ...
procedure Fournisseur_W is ...

-- instanciation du composant
Package Instance_De_A is  new A (Z => Fournisseur_Z, W => Fournisseur_W);

-- utilisation du composant par appel de la procédure X

Instance_De_A.X;
```

2.3 Composants logiciels avec Ada pour l'enseignement

2.3.1 Le contexte

L'enseignement de l'algorithmique distribuée est difficile pour l'enseignant et pour l'élève. En effet, l'absence d'un état global et l'existence d'un réseau de communication (avec sa topologie et ses propriétés ou défauts) se superposent à des notions de concurrence et de non déterminisme qui sont elles-mêmes difficiles à cerner.

Notre démarche consistait alors à définir un ensemble de composants logiciels en vue de la simulation et de l'expérimentation d'algorithmes distribués afin que les étudiants puissent visualiser des phénomènes et, in fine, mieux comprendre le domaine.

Ce projet s'inscrivait aussi dans le cadre plus global de la définition d'une bibliothèque de composants logiciels Ada pour l'enseignement de l'informatique à l'ENST Bretagne.

2.3.2 Composants "réseau"

Le premier composant que nous avons créé permet de simuler un réseau d'interconnexion de sites dans un système distribué. Pour cela nous avions retenu les critères suivants :

- ◇ Le composant doit offrir les deux primitives principales de communication que sont l'envoi d'un message vers un destinataire, et la diffusion d'un message vers tous les sites.

- ◇ Le composant doit aussi assurer la transmission de n'importe quel type de données : les données de contrôle sont souvent complexes afin de restaurer, par exemple, un état global.
- ◇ Le composant doit pouvoir relier un nombre quelconque de sites afin de simuler n'importe quel réseau.
- ◇ Il doit aussi pouvoir être configurable afin par exemple de simuler des pertes ou altération de messages. De nombreux algorithmes distribués sont en effet sensibles aux propriétés du réseau et il nous semblait donc important que l'étudiant puisse les faire varier simplement (introduction d'une probabilité de perte ou de duplication de message, par exemple).
- ◇ Son implémentation ne doit naturellement pas introduire d'interblocage dans le système.
- ◇ Pour simplifier, nous avons retenu le maillage complet comme topologie du réseau. La généralisation au maillage quelconque doit en effet s'accompagner de la définition d'une matrice d'interconnexion et d'une fonction de routage afin d'acheminer le message vers sa destination en l'absence de lien direct. La figure 2.3 présente le composant réseau que nous avons réalisé.

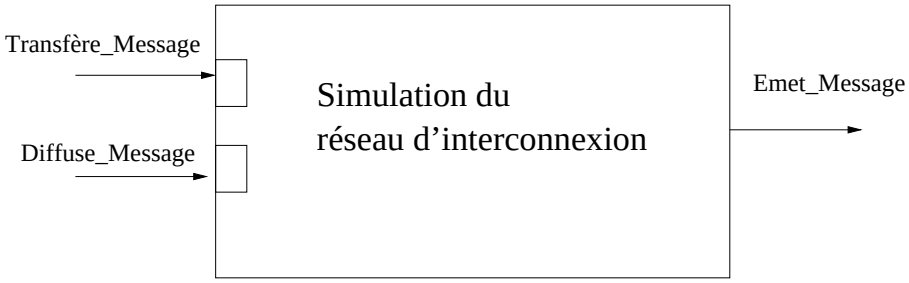


FIG. 2.3: Spécification du composant réseau

La spécification Ada de ce composant, qui prend en compte les différentes contraintes présentées ci-dessus, devient alors :

```
generic
  type Sites is range <>;
  type Message is private;
  with procedure Emet_Message (mess : message; a_qui : Sites);
  diffusion_à_soi : boolean := true;
package Medium_De_Communication is
  procedure Transfere_Message
      (message_à_transférer : Message; de_qui, a_Qui : Sites);
  procedure Diffuse_Message
      (message_à_diffuser: Message; Site_Diffuseur : Sites);
end Medium_De_Communication;
```

Avec un tel composant, son utilisation dans une situation réelle nécessite une configuration du même ordre que celle présentée en section précédente : il faut relier les entrées et les

sorties afin de les connecter. La réalisation d'un réseau composé de deux sites se représente comme en figure 2.4.

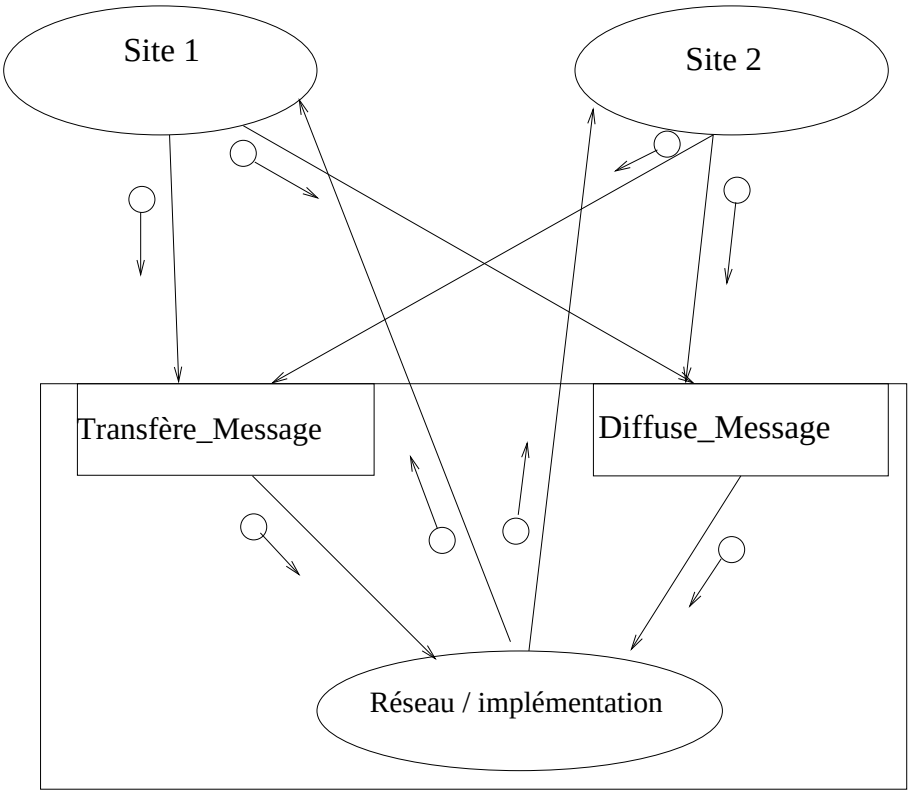


FIG. 2.4: Le composant réseau reliant deux sites

2.3.3 Composants pour le contrôle distribué

Nous avons en particulier développé des composants pour le contrôle distribué et proposé des mises en œuvre pour :

- ◇ des algorithmes d'exclusion mutuelle à partir de la synthèse proposée par M. Raynal dans [13] : algorithme de Lamport [14], de Ricart Agrawala [15], de Raynal.
- ◇ des algorithmes à base d'horloges vectorielles [16] ou de mémoire partagée [17].
- ◇ des algorithmes de détection de terminaison (algorithme de détection de propriétés stables de M. Raynal [18], de détection de terminaison de Dijkstra [19]) et de détection d'interblocage.

Enfin, pour permettre une simulation, nous avons développé des composants simulant le comportement d'utilisateurs, appelant les services de contrôle distribués fournis par les différents composants produits. Dans le cas d'un algorithme d'exclusion mutuelle, un ou plusieurs sites utilisateurs peuvent ainsi être simulés, chaque utilisateur simulé créé suit un comportement aléatoire ou non pour réaliser les opérations suivantes :

- ◇ demander l'exclusion mutuelle au bout d'une période d'attente déterminée.
- ◇ garder le privilège de l'exclusion mutuelle pendant un laps de temps simulant l'utilisation de la ressource.

◇ relâcher l'exclusion mutuelle et la rendre ainsi accessible à un autre.

2.3.4 Assemblage et composition

A partir des différents composants que nous avons présentés (composant réseau, composant de simulation d'utilisateurs, composant de mise en œuvre d'un contrôle distribué), nous pouvons les assembler grâce à la généricité comme dans 2.2.6.

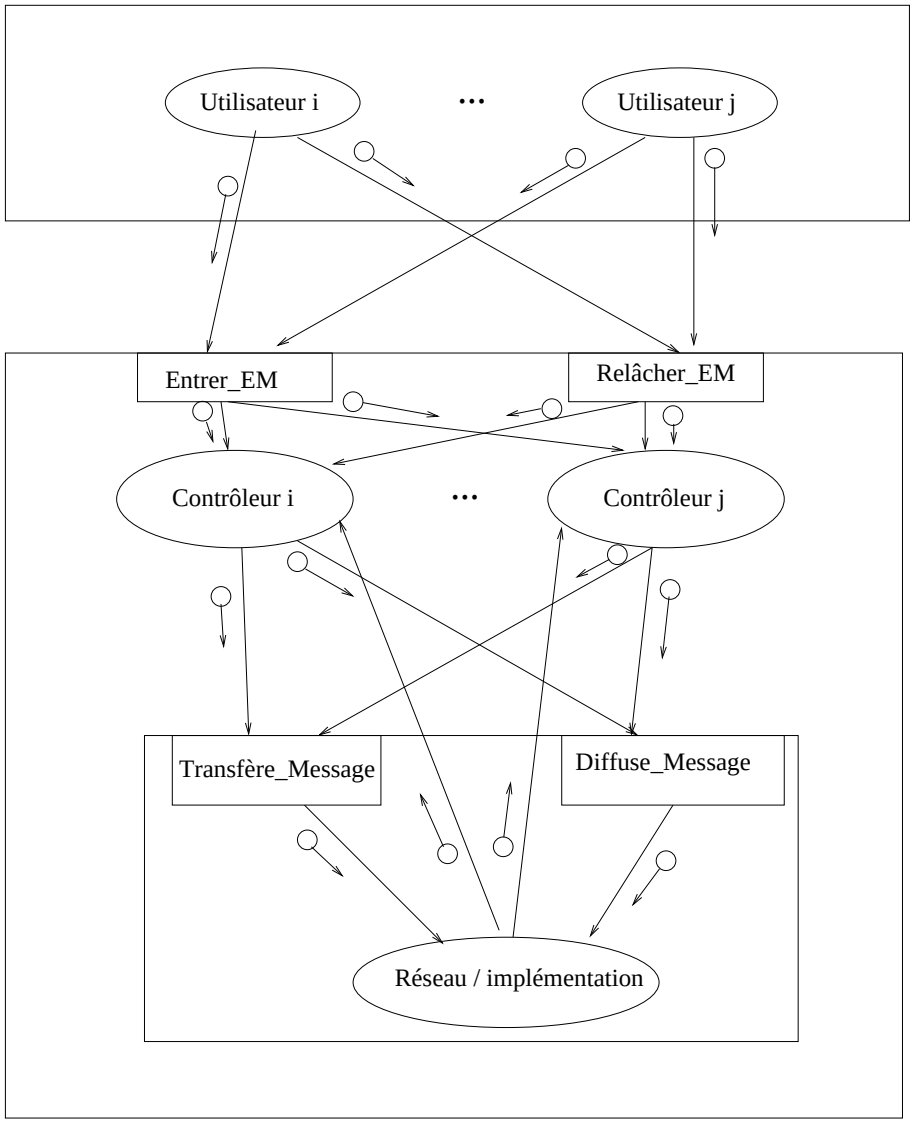


FIG. 2.5: Simulation de deux sites pour l'accès à l'exclusion mutuelle

2.3.5 Synthèse sur l'utilisation de composants Ada en enseignement

Nous avons réalisé plusieurs composants logiciels réutilisables en vue de la simulation d'algorithmes et de systèmes distribués. La définition de ces composants est facilement compréhensible surtout si elle est accompagnée d'un graphique identifiant les différents flots

de données et de contrôle ainsi que les différentes entités du système. Les diagrammes de R. Buhr [20] apportent ainsi une représentation visuelle intéressante pour l'assemblage des composants.

Ces composants ont été utilisés par plusieurs générations d'étudiants à l'ENST Paris et à l'ENST Bretagne et je pense que leur utilisation leur a permis de mieux comprendre l'algorithmique distribuée par ces simulations. L'impact des propriétés du réseau est rendu plus explicite et permet de prendre en compte le comportement de l'algorithme en cas de panne ou de défaillance d'un site ou d'un lien de communication.

2.4 Composants métiers avec Ada

2.4.1 Le contexte et la problématique

Depuis le début de ma thèse en 1985, je travaillais dans le domaine des composants logiciels Ada pour la simulation de systèmes distribués. Ces activités se sont ensuite renforcées, à partir de 1991, avec le démarrage de la thèse de L. Pautet en collaboration avec Dassault Electronique. De plus L. Pautet et moi-même connaissions l'équipe GNAT de NYU pour y avoir travaillé sur les compilateurs Ada 83. En 1993, le Professeur E. Schonberg a sollicité notre participation conjointe aux côtés de l'équipe de NYU afin de prendre en charge l'annexe systèmes distribués de Ada 95 (cette dernière est présentée en annexe E du manuel de référence d'Ada sous le terme DSA - *Distributed System Annex*).

Le projet a débuté par un partenariat entre l'équipe de A. Gargaro (concepteur principal du modèle de distribution d'Ada 95) de la société Computer Sciences Corporation (CSC), l'équipe de R. Volz de Texas A&M University (équipe déjà impliquée dans les travaux autour de la distribution pour Ada 83 [21], de l'équipe de NYU pour l'intégration dans l'environnement GNAT et des équipes de l'ENST Paris et de l'ENST Bretagne. Un premier prototype nommé ADEPT [22] a été réalisé entre les partenaires afin de montrer la faisabilité d'un système complet et riche pour la distribution : la société CSC était, en particulier, impliquée dans la réponse à un appel d'offres avec la Marine américaine, et il s'agissait de démontrer la faisabilité et l'intérêt d'un tel modèle de distribution. Cette validation de concepts a permis de mieux faire comprendre les potentialités de l'approche avec, en particulier, cette notion d'objets distribués (associant le modèle objet d'Ada 95 et la distribution) et de type de sous-programmes distants et surtout à la maintenir dans le document normalisé. Enfin, la réalisation des premiers prototypes a mis en évidence les lacunes du modèle qui ont pu être ainsi corrigées, avant la normalisation du langage. Le projet initial ADEPT [23] a ensuite migré vers une solution plus opérationnelle nommée GLADE, et ceci en partenariat plus étroit entre NYU et les ENST.

Ce projet de mise en œuvre de l'annexe distribuée d'Ada 95 constituait un véritable défi pour notre équipe. L'annexe DSA était en effet instable du fait du processus de définition et de normalisation du langage Ada 95, et nous devions la mettre en œuvre dans un environnement GNAT, également soumis à de nombreuses restructurations d'une équipe internationale. Notre contribution principale était d'arriver à une mise en œuvre opérationnelle de l'annexe et de valider les concepts retenus pour la distribution en Ada 95. Dans ce contexte, nous avons retenu une approche à base de composants, fournissant différents services aux applications distribuées, et assemblés pour former d'autres composants ou intégrés dans des applications.

Enfin, ces réalisations ont permis de mener des recherches et développements complémentaires liés aux thèmes des objets distribués et d'Ada 95 par les membres de notre équipe [24] et d'autres équipes internationales (Uppsala University, University of Madrid, EPFL Lausanne en particulier [25] [26] [27] [28]). Les résultats sont désormais commercialisés par la société Ada Core Technologies et la plate-forme proposée est intégrée dans des projets industriels (Boeing [29] en particulier) et sert pour l'enseignement et la formation.

2.4.2 Composants "système"

2.4.2.1 Première tentative avec ExTRA et Ada 83

A l'initiative de la DGA, le groupe de travail ExTRA (Extension Temps Réel pour Ada) composé d'industriels (Dassault Electronique, Sagem, Aérospaciale, etc.) a produit des spécifications d'extensions [30] en vue du développement des applications temps réel en Ada (il faut noter également les efforts similaires du groupe américain CIFO [31]). Ces deux groupes ont proposé des spécifications de haut niveau afin d'enrichir le modèle temps réel d'Ada 83 par l'ajout de différents mécanismes de coopération de tâches, une gestion des ressources partagées plus efficace et une communication asynchrone.

En liaison avec Dassault Electronique, nous avons retenu le service de communication asynchrone décrit par la spécification du composant *"Device"* comme composant central pour notre architecture :

```
package Devices is
   type Device is private;
   type Request is private;
   ...
   function Create (Name : in String) return Device;
   procedure Destroy (A_Device : in out Device);
   function Request_Create (Name : in String) return Request;
   procedure Request_Destroy (A_Request : in out Request);
   ...
   package Synchronous_Management is
      procedure Open (A_Device : in Device;
```

```
                                    A_Mode     : in     Device_Mode);
        ...
    end Synchronous_Management;
    package Asynchronous_Management is
        procedure   Open_Initiate  (A_Device   : in     Device;
                                    A_Mode     : in     Device_Mode := Device_Inout;
                                    A_Request  : in     Request);
        procedure   Wait_Open      (A_Request  : in     Request;
                                    A_Report   : out    Report;
                                    Time_Out   : in     Duration);
    ...
    end Asynchronous_Management;
    generic
        type Data is private;
    package Synchronous_Transfer is
        procedure   Read           (A_Device   : in     Device;
                                    Item       : out    Data);
        procedure   Read           (A_Device   : in     Device;
                                    Item       : out    Data;
                                    Passed     : out    Boolean;
                                    Time_Out   : in     Duration);
        ...
    end Synchronous_Transfer;
    generic
        type Data is private;
    package Asynchronous_Transfer is
        procedure   Read_Initiate  (A_Device   : in     Device;
                                    A_Request  : in     Request);
        procedure   Wait_Read      (Request    : in     Request;
                                    Item       : out    Data;
                                    A_Report   : out    Report;
                                    Passed     : out    Boolean;
                                    Time_Out   : in     Duration);
        ...
    end Asynchronous_Transfer;
    ...
    Usage_Error       : exception;
    Device_Error      : exception;
    Request_Error     : exception;
    ...
private
    – Implementation dependent
end Devices;
```

La définition de périphérique selon ExTRA correspond à celle d'un composant de communication au sens large : une entité qui permet de lire et d'écrire une donnée. L'implantation a eu lieu en Ada 83 standard tout en ayant comme soucis majeurs et constants la portabilité et l'efficacité [32]. Nous avons retenu, dans un premier temps, les protocoles de communication offerts par le monde Unix [33] : les fichiers, les tubes et le service de transport standard TLI (*"Transport Library Interface"*) assimilables aux sockets de Berkeley. Nous avons conçu l'in-

terface avec le composant de communication de haut niveau. La structure et les interactions entre les composants logiciels sont indiquées par la figure 2.6. Cette dernière met en évidence une organisation en trois couches :

- ◇ La spécification de haut niveau avec *Devices* qui fournit les opérations de communication synchrone et asynchrone aux applications. La généricité sur le type des données lues et écrites permet de s'adapter aux types de messages. Deux paquetages internes assurent le suivi des communications et la gestion des liens de communications.
- ◇ Le gestionnaire de périphérique (*"Peripheral_Manager"*) assure l'aiguillage des appels du composant de haut niveau vers les interfaces des composants réseaux. Cet aiguillage correspond à un codage en dur de l'appel de la primitive réseau et de ce fait tout ajout de composants de communication nécessite une modification du code de l'aiguillage et donc une recompilation de ce composant.
- ◇ Des composants qui réalisent des interfaces vers les services du réseau et différents protocoles Unix. Ces composants peuvent être utilisés indépendamment des composants de plus haut niveau et fournissent une alternative intéressante aux appels directs des fonctions des services Unix.

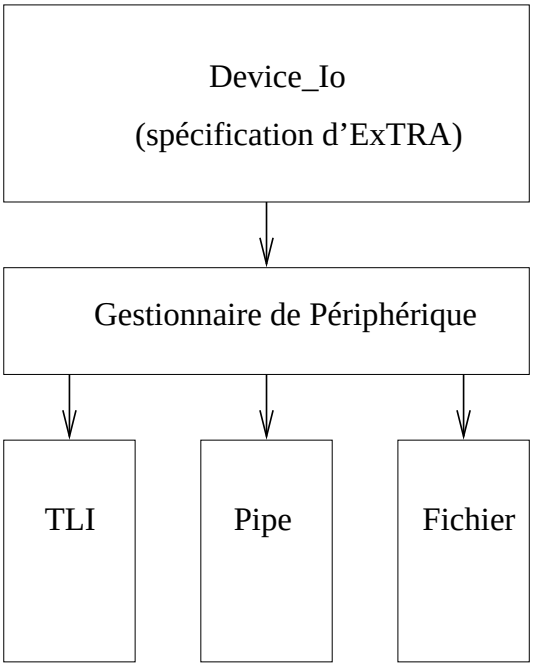


FIG. 2.6: Structure du paquetage de communication

2.4.2.2 Le composant GARLIC et Ada 95

GARLIC *"Generic Ada Reusable Library for Interpartition Communication"* [34] suit la lignée des composants de communication que nous avons proposés avec ExTRA. Dans le cadre du projet GLADE avec NYU, nous devions réaliser la première mise en oeuvre de l'annexe des systèmes distribués et donc proposer un couplage entre le modèle Ada 95 pour la distribution

et les différents médias de communication. Nous avons donc retenu le principe de la définition d'un composant de communication de haut niveau avec comme principaux critères de conception :

- ◇ La compatibilité avec les autres approches utilisées pour les systèmes distribués. Pour cela, nous avons suivi les recommandations et les principes de MPI [35] qui fournit des spécifications de haut niveau pour la communication.
- ◇ La modularité. La conception d'un composant comme GARLIC nécessite une interface de haut niveau (en lien avec les applications) mais aussi un ancrage avec les protocoles de bas niveau (le réseau). Cette dualité impose une abstraction des différentes couches et des interfaces précises.
- ◇ La portabilité. Les services de communication de bas niveau sont fortement liés aux systèmes d'exploitation ainsi qu'à l'architecture et protocoles sous-jacents. Nous devons donc abstraire ces particularités et permettre d'adapter les composants existants ou d'en intégrer un nouveau, si besoin.
- ◇ L'efficacité. Cette contrainte impose une gestion précise de la mémoire et des ressources, d'éviter des situations de concurrence non anticipées (*"race conditions"*), et de réaliser des appels directs aux primitives de communication de bas niveau.

Ces contraintes sont parfois antagonistes à première vue et nous avons centré le composant GARLIC sur la notion de protocole et utiliser les différentes facilités du langage Ada 95. Nous indiquons ci-après quelques éléments de conception de ce composant et le lecteur intéressé trouvera dans [34] des informations complémentaires :

- ◇ Le composant de communication, que nous proposons, reprend la notion de protocole telle que présentée par G. Coulouris :

  > *'a protocol is used to refer to a set of precisely-defined rules and conventions used for communication between similar software modules running on the different computers in a network" [36].*

  Chaque protocole apparaît dès lors comme une entité indépendante et nous avons utilisé la notion de hiérarchie de paquetages pour la définition de propriétés communes.
- ◇ Le protocole peut être représenté sous la forme d'un type abstrait (*"abstract"* selon la terminologie Ada 95) doté d'un ensemble d'opérations primitives, qui sont les différentes opérations qu'un médium de communication doit fournir. Ce type est également extensible (*"tagged"* selon la terminologie Ada 95) pour prendre en compte des informations ou opérations complémentaires pour ajouter un nouveau protocole (certains protocoles fonctionnent avec la notion d'états).
- ◇ Le composant GARLIC regroupe un ensemble de services de communication ainsi que des opérations de contrôle. Dans le composant présenté précédemment, nous avions un aiguillage logiciel qui permettait de se rediriger vers le bon protocole réseau. Pour GARLIC, les différents services de bas niveaux s'enregistrent (fournissent les adresses mémoire des sous-programmes associés) directement dans une structure de données liée au type abstrait.

- ◇ Les différents services de ce composant admettent comme paramètre un objet de type *Protocol_Type'Class* afin de prendre en compte les différents protocoles définis ou utilisés dans l'application. Les services de communication admettent des flux de données (*stream*) en tant que paramètres.
- ◇ Un objet interne appelé Synchronizer, évite les conflits lors d'accès aux données partagées et fournit également différents services lors de la définition des protocoles.

```
 type Opcode is
   (Remote_Call,     --  Normal remote call
    Shared_Memory,   --  Shared memory message
    Message_Passing, --  Free message passing
    ...);

type Protocol_Type is
   abstract tagged limited private;

 procedure Initiate_Send
   (Partition : in  Partition_ID;
    Length    : in  Stream_Element_Count;
    Protocol  : out Protocol_Access;
    Operation : in  Opcode);

 procedure Send
   (Protocol  : in Protocol_Type'Class;
    Partition : in Partition_ID;
    Stream    : access Params_Stream_Type);

 procedure Complete_Send
   (Protocol  : in Protocol_Type;
    Partition : in Partition_ID) is abstract;

 procedure Initiate_Receive
   (Partition : out Partition_ID;
    Length    : out Stream_Element_Count;
    Protocol  : out Protocol_Access;
    Operation : in  Opcode);

 procedure Receive
   (Protocol  : in Protocol_Type'Class;
    Partition : in Partition_ID;
    Stream    : access Params_Stream_Type);

 procedure Complete_Receive
   (Protocol  : in Protocol_Type;
    Partition : in Partition_ID) is abstract;
```

L'architecture des composants logiciels de communication est présentée dans la figure 2.7 et illustre la composition hiérarchique du composant GARLIC et des sens paquetages

enfants, qui prennent en charge les protocoles spécialisés (et concrets). La figure 2.8 présente les différents points d'entrée de notre composant central de communication.

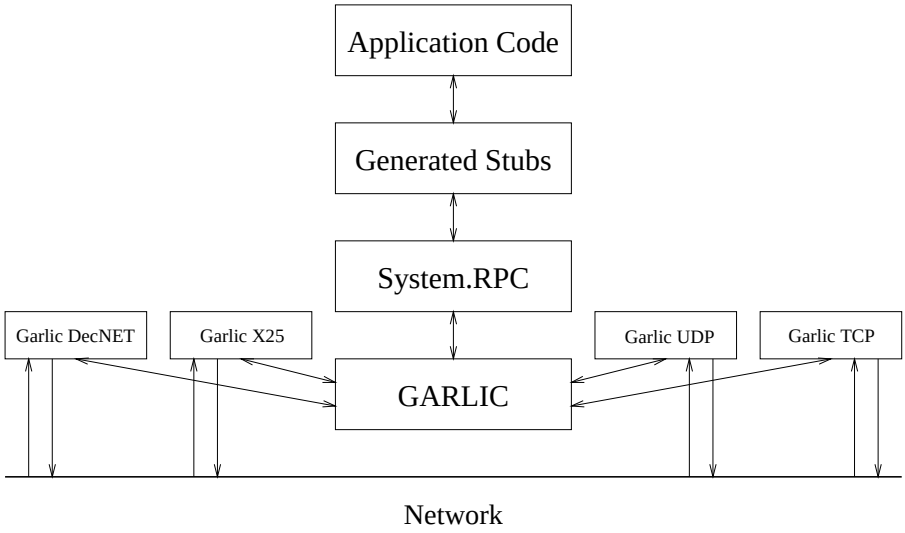


FIG. 2.7: Architecture de composants logiciels de communication

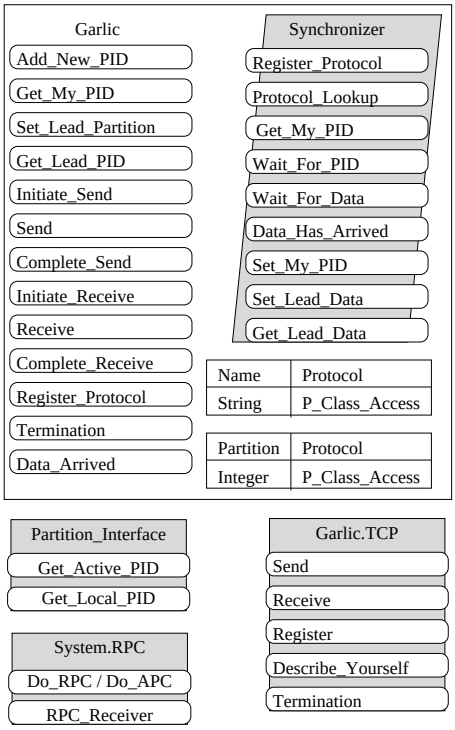


FIG. 2.8: Architecture du composant GARLIC

2.4.3 Composants pour le contrôle distribué

À partir des composants de communications que nous avons présentés dans les sections précédentes, nous avons pu réaliser l'implémentation de l'approche de distribution de Ada 95 dans l'environnement GNAT et étoffer notre bibliothèques de composants. En effet, avec

les composants liés à ExTRA ou à GARLIC, nous avons défini une interface pour l'accès aux différents protocoles réseaux du monde Unix. Ces composants de communication correspondent naturellement au paradigme du passage de message.

Nous avons souhaité compléter l'offre des paradigmes de communications et d'interactions et enrichir ainsi l'environnement de développement des applications distribuées. Plusieurs composants ont été ainsi réalisés et tous reprennent les principes de génie logiciel que nous avons énoncés précédemment :

- ◇ Le premier composant met en oeuvre le modèle Linda [37] [38] proposé par N. Carriero. Le concept de mémoire associative est intéressant et extrêmement puissant pour la synchronisation des applications distribuées, comme l'indique [39]. La mise en oeuvre de ce paradigme en Ada se présente sous la forme d'un composant, qui utilise le composant de communication d'ExTRA et nous le présentons dans [40].
- ◇ Le deuxième composant logiciel met en oeuvre le paradigme de mémoire partagée, tel que proposé par [17]. Il propose un espace d'adressage commun avec différents mécanismes de gestion de cohérence et est décrit dans [41]. Ce composant prend également en compte le contexte particulier des systèmes embarqués et apporte une gestion novatrice pour la gestion des défauts de pages.
- ◇ Enfin, ces différents composants et le travail réalisé autour de GNAT permettent au développeur d'applications distribuées de retrouver le modèle des objets distribués.

2.4.4 Assemblage et composition

Nous avons présenté dans les sections précédentes deux approches de composition et d'assemblage de composants :

- ◇ La première approche que nous avons utilisée est celle de l'aiguilleur logiciel utilisé par le composant de communication d'ExTRA. L'assemblage de composants écrits avec Ada 83 pose le problème de la difficulté d'extension et de rajout de nouvelles spécialisations des composants. En effet, l'ajout d'un composant logiciel nécessite, comme nous l'avons indiqué, la réécriture de l'aiguilleur et donc la recompilation de l'ensemble.
- ◇ L'approche suivie pour la conception de GARLIC utilise les nouvelles fonctionnalités offertes par Ada 95 avec la notion de type abstrait, de type étiqueté et de paquetages hiérarchiques. L'interconnexion des composants se réalise en enregistrant les services des couches inférieures dans des descripteurs de niveau supérieur.

2.4.5 Synthèse sur l'utilisation de composants métiers Ada

Nous avons réalisé un ensemble de composants logiciels en vue de la communication et du contrôle des applications distribuées. Ces composants répondent aux critères et traits des composants logiciels présentés en début de ce chapitre et ont été intégrés avec succès dans des applications et dans d'autres composants.

La démarche, que nous avons retenue dès le début, insiste sur l'analyse au préalable de l'existant et sur la capacité d'abstraction et de généralisation. La définition du composant GARLIC est révélatrice de cet état puisque nous avons proposé un modèle de composant qui unifie des protocoles bien différents.

2.5 Synthèse sur les composants réutilisables et perspectives

2.5.1 Synthèse sur les composants Ada

Nous avons présenté dans ce chapitre notre approche et nos réalisations de composants logiciels réutilisables pour les applications distribuées. A partir d'une démarche pragmatique, de définition de composants pour la simulation, nous avons étendu le contexte d'utilisation de ces composants en prenant en compte différents réseaux de communication et différents paradigmes de programmation des applications distribuées.

Nous ne reprenons pas dans le détail les avantages d'une conception par composants logiciels mais insistons sur deux points qui nous apparaissent fondamentaux :

- **La prise en compte de la réutilisabilité lors de la conception de composant logiciel est difficile.**
  Il faut en effet anticiper des contextes futurs d'utilisation du composant et nous avons vu que ceci était difficile à réaliser dans le cadre d'une analyse des besoins. Nous pensons avoir réussi à définir des composants réutilisables de communication (*Devices* et GARLIC) principalement du fait de notre connaissance du domaine et surtout d'une exploration des standards. Cette connaissance nous a permis de nous abstraire des couches basses de communication et de dégager l'essentiel d'un composant de communication. Le processus de conception de ce composant a été itératif puisque, après chaque proposition de composant, nous le confrontions avec la réalité en l'intégrant dans des contextes divers et variés. Cette itération nous a permis de plus de proposer un composant de qualité, puisque testé dans de nombreuses configurations.

- **L'assemblage de composants est difficile.**
  Nous avons vu en particulier que l'utilisation de la généricité permettait d'élargir le contexte d'utilisation de composants. Ce dernier est précisé lors de l'instanciation en indiquant les différents connecteurs. Cependant, ces assemblages sont difficilement perçus sans l'aide de diagrammes (qui ne présentent qu'une vue du problème). La deuxième solution que nous avons proposée avec GARLIC consiste à définir un type abstrait et ce sont les différentes spécialisations (ou concrétisations du composant) qui viennent alors s'enregistrer. L'approche objet et ses mécanismes d'héritage et de liaison

dynamique apportent une solution technologique élégante.

Enfin, nous pensons que les résultats obtenus n'auraient pas pu être atteints sans cette démarche composants que nous avons retenue. Les différents composants sont en effet le fruit du travail d'une petite équipe formée autour de L. Pautet et de moi-même dans le cadre d'une association avec l'équipe de New York University.

2.6 Perspectives

Mes actions de recherche et développement dans le cadre des composants logiciels réutilisables ont commencé avec mes études en doctorat et se sont poursuivies dans le cadre du projet GNAT et de la définition et réalisation d'un environnement pour la conception d'applications distribuées avec Ada 95. Au cours de ces travaux, une base importante de composants a été réalisée et réutilisée dans des contextes divers et variés. Enfin, ces différents composants ont été utilisés avec succès dans un contexte d'apprentissage et ont permis à plusieurs générations d'étudiants de se sensibiliser aux thématiques liées à la conception d'applications distribuées.

Il nous semble intéressant de poursuivre ce travail dans trois directions principales :

- ◇ La première concerne la description d'un composant logiciel. En effet, si la base des composants se développe et regroupe plusieurs centaines de composants, le problème de la recherche de composants se pose et donc celui de l'indexation de ces composants. Nous pensons qu'une indexation est possible et pourrait reprendre les propriétés présentées pour les services distribués dans le dernier chapitre de ce document. Une telle description pourrait présenter plusieurs vues du composant allant de ses propriétés statiques jusqu'à la description de son comportement et de son interface.
- ◇ La deuxième concerne l'assemblage des composants. Nous avons commencé à aborder ces thèmes dans le cadre du projet RNTL COTRE mais nous pensons qu'il faut aller plus loin et proposer un véritable support pour la composition et l'assemblage de composants logiciels.
- ◇ La troisième et dernière direction qui nous paraît indispensable à explorer est celle de la vérification de la cohérence des assemblages. Nous avons montré dans les précédentes sections que le typage permet de vérifier certaines propriétés statiques mais il nous faut aussi être capable de vérifier des propriétés liées au comportement. Dans le cadre du projet RNTL COTRE, le consortium s'est principalement intéressé aux vérifications formelles et ceci constituerait une première contribution à cette action.

CHAPITRE 3

Enseignement et composants d'apprentissage pour IST CANDLE

Dans ce chapitre, je présente le projet IST CANDLE et mes contributions. Dans un premier temps, je présente le contexte du projet, puis ses objectifs. Je continuerai ensuite par une définition du composant d'apprentissage qui forme le cœur de notre approche de réutilisation et de partage. Je présenterai ensuite mes contributions au projet et justifierai nos choix et nos approches. La cinquième partie présentera quelques questions ouvertes auxquelles je fournirai quelques premiers éléments de réponse. Enfin, en guise de conclusion de ce chapitre, je présenterai les perspectives pour une suite de nos activités de recherche et développement.

3.1 Le contexte du projet

Le projet CANDLE (*"Collaborative And Networked Distributed Learning Environment"*) est issu du groupe EUNICE, un réseau d'universités européennes spécialisées dans le domaine des télécommunications et des réseaux informatiques. Ce groupe s'est créé il y a une dizaine d'années afin de favoriser les échanges entre les partenaires tant en enseignement qu'en recherche. La prise en compte conjointe des aspects liés à l'enseignement et à la recherche souligne l'originalité de ce groupe et surtout l'importance donnée à la formation par la recherche. Les membres d'EUNICE se sont enfin engagés par une charte (disponible depuis http ://www.eunice-forum.org) à favoriser les échanges entre enseignants et étudiants afin d'améliorer l'enseignement des télécommunications en Europe et de coordonner les différentes activités de recherche. Dès la naissance du groupe, les activités en recherche ont été nombreuses et se sont concrétisées par des actions diverses et variées : depuis l'établissement d'écoles d'été annuelles pour les doctorants jusqu'à l'élaboration de projets de recherche européens en passant par l'établissement de réseaux d'excellence (*"Network of Excellence"* du 6ème PCRD).

Les activités liées à l'enseignement étaient par contre limitées et seuls des accords d'échanges d'étudiants et de reconnaissance mutuelle de diplômes et de formation ont pu être établis entre les partenaires. Ces accords institutionnels se sont naturellement accompagnés d'échanges entre les enseignants et les étudiants. Je coordonne le groupe EUNICE depuis 1998 et avec les représentants des différents membres, nous avons souhaité aller plus loin et établir des objectifs plus ambitieux. Le programme européen IST [1] nous a donné un cadre et les moyens de l'atteindre.

La thématique commune des partenaires d'EUNICE, qui est celle des télécommunications et plus particulièrement des réseaux, fournit un domaine bien délimité. Ce domaine se caractérise aussi par une forte évolution des technologies et concepts. Ces derniers doivent naturellement être pris en compte par les enseignants dans leurs formations supérieures afin de préparer les étudiants à leurs futures activités professionnelles. Il en est de même pour les employés d'entreprises de haute technologie, qui ne peuvent ignorer les avancées techniques, que ce soit au niveau concepts et approches ou celui des plates-formes, par exemple.

Or nous le savons tous, construire un nouveau cours nécessite un temps important pour sa préparation et l'accomplissement de nombreuses tâches, parmi celles-ci nous trouvons par exemple :

- ◇ La collecte d'information que ce soit à partir de livres, sur le WEB, auprès de constructeurs et de fournisseurs de technologies, de centres de recherches, de documents et de bibliographie.
- ◇ Ce grand volume d'information, une fois collecté, doit ensuite être sélectionné, trié et synthétisé.
- ◇ La préparation d'un nouveau cours ou d'une nouvelle formation est alors possible après sélection d'une approche pédagogique en lien avec les pratiques de l'enseignant, de son établissement, du public visé et des objectifs désirés pour la formation, etc.
- ◇ Ce cours peut enfin s'accompagner de simulations et de démonstrations, de travaux pratiques et d'études de cas, afin de mieux visualiser un phénomène et de favoriser ainsi son appréhension par les étudiants.
- ◇ Enfin, après ces étapes de préparation et l'organisation de la formation, le cours peut être délivré selon un mode direct (face-à-face) ou à distance ou suivant une association de ces deux modes. Une fois le cours terminé, une évaluation est nécessaire afin de prendre en compte l'impact de la formation sur son public et de la modifier éventuellement : en rajoutant des sections manquantes ou des liens vers des aspects théoriques qui se sont révélés être des freins lors de l'apprentissage, de reformulation et de correc-

[1] À partir de 1998, l'Union Européenne lance le 5ème PCRD dans le cadre des actions IST (*"Information Society Technologies"*) avec un volet consacré aux technologies pour le partage des savoirs et leurs accès. Ce thème correspond enfin à des actions fortes de la politique européenne pour l'apprentissage tout au long de la vie, et l'accès aux savoirs (cf déclaration du Conseil Européen de Lisbonne sur *http* : *//europa.eu.int/comm/education/policies/*2010*/et_*2010*_en.html*).

tion de tout ou partie du cours, ou de retrait de sections inutiles car trop complexes ou ... trop triviales.

Toutes ces tâches sont naturellement itératives et dans un domaine aussi dynamique que celui des réseaux et des télécommunications, il est presque illusoire d'imaginer un cours figé et définitif et, surtout, comme le résultat du travail d'une seule personne. Ces étapes de conception et production de nouveaux cours nécessitent naturellement un travail important de préparation et se déroulent, bien souvent, dans le cadre d'une équipe ... et seront aussi engagées par plusieurs équipes dans plusieurs universités. L'approche de ce projet est justement de favoriser la collaboration entre enseignants en vue de la production de supports de cours actuels et de bonne qualité, et qui puissent évoluer en tenant compte de la dynamique du domaine et des avancées scientifiques, et être modifiées par son auteur ou un autre.

"To me, teaching is synonymous with sharing." D. Wiley [42]

La question posée alors est de déterminer ce qu'une communauté d'apprenants et d'enseignants peut partager et déterminer les approches et outils qui supporteront un tel partage.

3.2 Les objectifs du projet IST CANDLE

Le projet CANDLE (*"Collaborative And Networked Distributed Learning Environment"*) s'inscrit tout naturellement dans la thématique du partage et de la réutilisation de cours (tout ou partie) et de pratiques dans une communauté bien définie afin d'améliorer la qualité de la formation des étudiants en formation mais aussi de la formation permanente (apprentissage et formation tout au long de la vie). Cet objectif se décline naturellement en plusieurs thèmes et sous objectifs et nous en indiquons quelques-uns ci-après :

- ◇ Proposer une méthodologie et approches pédagogiques pour la création de cours à partir de composants partagés.
- ◇ Proposer des approches innovantes afin de diffuser des cours composés d'éléments réutilisés.
- ◇ Outiller ces approches et offrir ainsi un atelier permettant de composer des cours, d'assurer leur diffusion et de partager des composants de cours.
- ◇ Concevoir et mettre en place une architecture distribuée afin d'assurer le partage et l'accès de ces composants de cours pour les enseignants mais aussi pour les étudiants.
- ◇ Contribuer aux échanges entre enseignants et étudiants et améliorer les échanges et coopérations entre eux. Cette dimension collective permet enfin de renforcer les liens entre équipes et personnes.

La définition et l'établissement de pratiques et d'approches communes nécessitent naturellement un consensus afin qu'elles puissent être appropriées par ses futurs utilisateurs. Ceci

nécessite également une compatibilité afin que ces dernières puissent co-exister avec les pratiques individuelles précédentes. Le défi rencontré est donc de pouvoir proposer une évolution (et non pas une révolution ou un changement radical) et surtout de démontrer tous les bénéfices, tant individuels que collectifs, associés à ces nouvelles pratiques.

3.3 La notion de composant d'apprentissage

3.3.1 Une définition du composant d'apprentissage ?

> "Learning objects are elements of a new type of computer-based instruction grounded in the object-oriented paradigm of computer science. Object-orientation highly values the creation of components (called objects) that can be reused in multiple contexts" (D. Wiley - [43])

La notion même de composants d'apprentissage (*"learning object"*) [44] a pris une actualité particulière du fait de l'arrivée du web et des facilités qu'il offre pour le dépôt, l'accès et le partage des ressources. Des ressources numériques (des polycopiés, des transparents, des travaux pratiques, des simulations, etc.) peuvent être aisément transmises à une ou plusieurs personnes, voire mises à la disposition de toute une communauté, afin d'être réutilisées dans un autre contexte.

> "In some cases, learning activities can be turned into templates with original content stripped out and new content filled in. Thus the term 'learning object' embraces both content and processes, stored and transmitted in digital format, and having the potential for sharing in other instructional contexts." (G. Richards in [45])

Cette notion d'objet réutilisable est directement liée à celle que nous connaissons en informatique et en génie logiciel, en particulier. Comme pour un programme informatique qui pourrait être réalisé par assemblage de composants sur étagères, l'approche visée avec les composants d'apprentissage est celle de fabriquer des matériaux pédagogiques (des cours, des modules par exemple) à partir de briques de base composées d'éléments aussi variés que d'images, d'explications, de tutoriaux ou de simulations, par exemple.

Cette approche est naturellement plus qu'attrayante pour la communauté enseignante mais aussi pour un public bien plus vaste. Elle contribuerait aux impératifs de qualité et de réactivité pour la production de cours dans des domaines innovants et dynamiques. Cependant, nous connaissons aussi les limites de telles approches de réutilisation en informatique et les difficultés associées à la conception de composants réutilisables (il faut en particulier anticiper les contextes d'utilisations futures et possibles) et les intégrer avec d'autres.

Griff Richards [45] identifie ainsi cinq points critiques afin de favoriser la notion de composants d'apprentissage et leur réutilisation. Ces questions sont naturellement difficiles et de

nombreux efforts de standardisation ont été entrepris pour y répondre plus ou moins complètement :

- ◇ L'interopérabilité : un composant créé sur une plate-forme donnée peut-il être intégré et réutilisé sur une autre plate-forme ?
- ◇ L'utilisation des méta données : comment indexer un composant afin que d'autres, et son auteur, puissent le retrouver ?
- ◇ Les bibliothèques : comment stocker les composants d'apprentissage et leurs méta données afin de les rendre visibles et accessibles par d'autres ?
- ◇ La pédagogie : comment créer un objet et signifier les processus pédagogiques implicites et le contexte de sa création et de son utilisation ?
- ◇ L'évaluation : comment assurer la qualité des composants et identifier des critères de qualité ?

Dans le contexte de IST CANDLE, nous devons aussi prendre en compte ces questions, y apporter une réponse, et prendre en considération les facteurs humains et les relations complexes qui existent entre les domaines d'enseignement, les enseignants et les élèves.

3.3.2 Thématiques de recherche liées à la notion de composant d'apprentissage

De nombreuses actions de recherche et de développements dans le domaine des nouvelles technologies ont été lancées ces dernières années. Des approches mais aussi des outils et plates-formes ont été proposés à partir des travaux initiaux, des premières expérimentations ont rendu possibles une première évaluation et de modifier les prototypes avant les déploiements à grande échelle que nous connaissons actuellement (WebCT, Explora et les travaux du LICEF [46] [47], en particulier).

Il reste cependant de très nombreux thèmes de recherche à défricher et à explorer. Nous en indiquons quelques-uns afin d'illustrer la diversité et la complexité de ces domaines :

- ◇ comment assurer que des briques de cours puissent former quelque chose de cohérent pour l'élève et pour l'enseignant alors que les contextes de création et de réutilisation peuvent être bien différents ?
- ◇ comment décrire un composant d'apprentissage et quels critères faut-il prendre en compte ?
- ◇ comment indexer un composant afin de favoriser sa recherche et son accès ?
- ◇ comment le rendre accessible, voire contrôler son accès ?
- ◇ comment favoriser la réutilisation et la rendre acceptable ?
- ◇ gérer les aspects droits d'auteurs et propriété intellectuelle ?
- ◇ comment définir des thésaurus et autres langages communs de description de domaine tout en tenant compte des aspects multilingues et évolution de ces domaines techniques ? idem pour la définition des méta données [48].

- ◇ peut-on gérer ses propres index et autres méta données tout en gardant une partie commune du domaine ?
- ◇ comment assurer l'évolution du modèle de méta données ?

D'autres questions ouvertes et pertinentes sont présentées par Alain Derycke [50]. Elles ont trait à l'impact des exceptions et particularités nationales, aux nouvelles pratiques associées en pédagogie, et aux nouvelles relations qui s'établissent vis-à-vis de l'enseignant et aux nouvelles fonctions des universités (virtuelles).

3.3.3 Le point sur la granularité : que peut-on réutiliser ?

La notion de granularité [51] est centrale pour la structuration de documents pédagogiques et pour la création de ces documents. Le choix de la granularité, c'est-à-dire les critères retenus et utilisés pour trouver le bon grain d'apprentissage, influence la réutilisabilité de ces composants. De plus, afin de permettre la réutilisation, des exigences particulières doivent être respectées et apparaissent comme autant de contraintes sur la définition et la création de ces composants. Ces décisions sont, en général, intuitives et liées directement à l'auteur, mais naturellement les difficultés apparaissent lorsque nous nous plaçons dans une communauté d'auteurs appelés à s'échanger des composants d'apprentissage. Dans ce cas, la notion de granularité doit apparaître comme une notion commune et partagée.

Plusieurs organisations liées à la définition de standards en enseignement à distance (IEEE LOM [52] et IMS [53], en particulier) définissent la notion de granularité (ou niveau d'agrégation) en lien avec la hiérarchie d'un cours : un cours ayant alors la plus grande granularité et un élément (une image par exemple) étant le plus petit grain. De telles vues sur la granularité sont généralement centrées média.

D'autres définitions mettent plus l'accent sur le contenu et la densité du composant. Par exemple, afin de déterminer la granularité, on peut se poser les questions suivantes : quels sont les éléments du modèle, la stratégie pédagogique choisie, quelle est la durée nécessaire pour le lire et/ou le comprendre, quels thèmes aborde-t-il, etc. Plus il y a d'éléments dans le composant, plus celui-ci sera dense et naturellement aura un grain important.

> "When creating learning objects for instructional use, two design issues are preeminent : granularity and combination. Succintly stated, granularity refers to the size of the learning object while combination refers to the manner in which objects are assembled into larger structures to facilitate instruction. Granularity and combination issues are in many ways analoguous to the scope and sequence issues with which instructional designers grapple requlary." *David Wiley* [43]

3.3.4 Critères de ré-utilisabilité

La question sous-jacente au thème de la réutilisation est de déterminer ce qui rend plus aisé de réutiliser et d'adapter un composant existant plutôt que de créer un nouveau composant ? Pourquoi et comment un document créé pour un contexte d'apprentissage donné peut-il être intégré dans un autre contexte d'apprentissage ? Afin de répondre à cette question, il est nécessaire d'identifier des propriétés et critères de qualité pouvant être utilisés pour qualifier un contexte d'apprentissage afin ensuite de comparer ces deux contextes d'apprentissage (le contexte initial dans lequel le composant a été créé et le nouveau) et de déterminer leur compatibilité.

Pour IST CANDLE, nous avons ainsi identifié trois critères de qualité ou propriétés : solidité (*"soundness"*), utilité (*"usefulness"*) et intérêt (*"worthwhileness"*). Réutiliser devient alors possible et intéressant lorsque ces trois propriétés sont vérifiées. Nous donnons ci-après quelques détails sur ces propriétés :

- La solidité d'un composant est évaluée par l'utilisateur en fonction de l'utilisation correcte de :
  - la terminologie du domaine,
  - des règles et procédures définies par la communauté, etc. Ce critère nécessite donc l'établissement de consensus au sein d'une communauté comme celle d'EUNICE.
- L'utilité d'un composant se définit dans un contexte d'apprentissage : domaine abordé, pédagogie retenue, liens vers des domaines connexes, présentation et qualité. En d'autres termes, le composant est utile s'il remplit un rôle / une fonction dans un contexte d'apprentissage. Pour être considéré comme utile, le composant doit au moins couvrir le domaine indiqué dans son descriptif avec un niveau de précision suffisant.
- Le composant sera enfin intéressant si le coût de son acquisition et de son adaptation est moindre que celui de le créer. Cette dimension économique et pratique renforce ici la nécessité de trouver le 'bon' composant et ceci à partir d'une description la plus complète possible.

Ces critères prennent en compte différents aspects : le domaine (le contexte sémantique), le contexte de création et de réutilisation, et des contraintes de présentation. Afin que le composant soit réutilisable, il doit satisfaire simultanément les trois critères mentionnés ci-dessus.

Enfin, ces critères de qualité sont très subjectifs et très liés aux contextes. Le projet IST CANDLE bénéficie d'un cadre intéressant puisque la communauté EUNICE partage déjà des expériences communes en recherche, dispose des mêmes domaines d'enseignement et ceci à des niveaux très similaires (master du LMD et doctorants, en particulier).

3.3.5 Caractérisation des éléments d'apprentissage

Dans la perspective de partage de ressources et de composants d'apprentissage, le thème de leur accessibilité par les étudiants et les enseignants se pose naturellement. Un enseignant qui cherche des contenus en ligne précisera dans sa requête des informations comme :

- ◇ une description du contenu afin de répondre aux objectifs pédagogiques [54],
- ◇ une caractérisation de l'apprenant (son âge, ses connaissances, ou sa langue de travail, par exemple) ,
- ◇ l'approche pédagogique suivie.

Toutes ces informations sont autant de critères pertinents pour la recherche (et le dépôt) de composants d'apprentissage. Ces derniers peuvent être caractérisés et décrits plus ou moins complètement avec un modèle de méta données

Les méta données sont des données sur les données et se présentent sous la forme d'un ensemble de rubriques d'information qui décrivent des ressources de types divers. La composante "méta" du mot "méta données" révèle une volonté d'abstraction à un niveau supérieur et introduit aussi la notion de réflexivité. Les méta données doivent donc compléter l'information relative aux données à un niveau d'abstraction supérieur tout en étant capables de se décrire elles-mêmes. La composante "données" indique que les méta données sont aussi des données et peuvent donc être structurées et interrogées.

Dans un contexte d'échanges intensifs du fait d'Internet, plusieurs équipes ont travaillé à la définition de standards communs en vue de la définition de ces modèles de méta données. A partir des travaux des équipes du NIST (National Institute for Standards and Technology), IEEE LTSC (Learning Technology Standards Committee) et ARIADNE en Europe [55], IEEE LOM (Learning Object Meta data) [52] réalise une synthèse de ces travaux en vue de la normalisation.

Les objectifs principaux de IEEE LOM regroupent les fonctionnalités attendues d'un système d'indexation, c'est-à-dire de :

- ◇ permettre aux apprenants et aux enseignants de rechercher, évaluer, acquérir et utiliser des composants d'apprentissage,
- ◇ permettre le partage et l'échange de composants d'apprentissage entre plates-formes,
- ◇ permettre le développement de composants d'apprentissage qui peuvent ensuite être combinés et décomposés,
- ◇ permettre à des agents informatiques (logiciels et moteurs de compositions, par exemple) de composer automatiquement et dynamiquement des cours personnalisés à un individu (le lien avec le web sémantique est immédiat avec ce point),
- ◇ fournir un standard simple mais aussi extensible à différents domaines.

Le Web sémantique évoque la notion d'annotations pour caractériser une indexation par méta données. Le terme d'annotation est lié à l'information complémentaire qui est associée au contenu. Enfin, ces annotations sont destinées à l'humain mais aussi à la machine pour un traitement automatique.

3.4 Approches proposées et contributions dans IST CANDLE

Au sein du projet IST CANDLE, j'assumais la responsabilité du lot (*"work package"*) consacré aux interfaces avec l'utilisateur et avais en charge la coordination d'une évaluation mettant en jeu la collaboration entre enseignants et étudiants. En tant que membre du comité de pilotage du projet, je participais à la coordination globale, aux revues avec les experts désignés par l'UE et avais, de ce fait, un état global d'avancement du projet pour les différents lots. L'équipe de l'ENST Bretagne avait pris la responsabilité de la conception et du développement des outils d'interfaces, un thème que nous connaissions mais dans un contexte applicatif (les nouvelles technologies pour l'enseignement) nouveau pour l'équipe.

Un tel projet implique bien évidemment des aspects technologiques conséquents mais aussi et surtout des investigations en pédagogie et sciences de l'éducation, évaluation et mise en situation, adoption de consensus et propositions de nouvelles pratiques. Chacun de ces thèmes fait lui-même l'objet d'études, de recherches et de décisions, en relation avec les autres équipes du projet.

Nous devions apparaître en tant que concepteurs de logiciels mais du fait de notre position dans le projet (aux interfaces) nous avons dû proposer de nouvelles approches et concevoir les outils tout en prenant en compte les exigences d'un tel projet :

- ◇ des aspects innovants compatibles avec un programme de recherche européen
- ◇ des exigences de qualité afin que les outils et approches soient utilisés par les enseignants
- ◇ des exigences liées aux engagements contractuels.
- ◇ des impératifs de calendrier afin de respecter l'échéancier (livraison de logiciels à date contrainte afin de mener des évaluations sur des groupes d'étudiants en formation, par exemple) et des contraintes financières.

Nous avons relevé ces défis avec l'outil CAT (CAT est l'acronyme de CANDLE Authoring Tool et ce nom ne reflète qu'une partie des fonctionnalités de l'outil - celles présentes dans sa première version livrée - Les fonctionnalités du logiciel se sont étoffées et CAT apparaît désormais comme un ensemble d'outils intégrés et destinés aux auteurs).

3.4.1 Cadre théorique de notre approche de conception

L'outil CAT joue un rôle central pour le concepteur de cours et ceci dès le démarrage du projet CANDLE. En effet, après une étude de l'existant et des scénarios d'usage du système réalisés par l'ensemble des partenaires [57], nous devions outiller un tel système de gestion de composants d'apprentissage afin de mener à bien les expérimentations et les analyses d'usage. Parmi ces fonctionnalités, nous avions prévu les supports pour réaliser les tâches en vue de production de cours et de composants d'apprentissage :

- ◇ créer un cours et mettre en place le ou les modèles pédagogiques de CANDLE ad'hoc,
- ◇ favoriser des approches à base de réutilisation en permettant, en particulier, de rechercher des composants et de les insérer dans d'autres contextes,
- ◇ s'interfacer avec les bases de composants d'apprentissage (repositories et autres bases de données) qui peuvent déjà exister sur les différents établissements des partenaires du projet,
- ◇ sauvegarder localement ou à distance les documents pédagogiques réalisés,
- ◇ partager des éléments de cours et favoriser une collaboration entre les différents acteurs.

La pédagogie joue naturellement un rôle central et prépondérant dans CAT. La première tâche que nous avons eu à réaliser était de déterminer ce qui se cachait derrière ce terme et cet art.

La pédagogie peut être comprise en tant qu'interactions qui ont lieu entre les enseignants et les apprenants, et les facteurs qui créent et dirigent ces interactions. Elle prend en compte le contexte et les aspects humains des relations qui structurent l'apprentissage et l'enseignement, que ce dernier soit en face-à-face ou à distance. Toute tentative de modélisation de ces facteurs doit aussi prendre en compte un éventail très large de croyances et de pratiques qui sont souvent présentes dans des environnements éducatifs, qu'ils soient formels ou informels. Respectivement, un modèle doit aussi prendre en compte l'impact de l'environnement éducatif sur les croyances et pratiques [60].

L'hypothèse principale de notre approche est d'affirmer que les activités pédagogiques sont orientées vers des finalités puisque contrôlées par les intentions des individus impliqués, organisées temporellement en séquences d'actions, et liées sémantiquement par des relations rhétoriques et ceci dans un contexte précis. Une des difficultés pour une telle modélisation est l'existence de jeux et relations subtiles entre contextes, buts, séquences d'actions et rhétorique.

Concevoir de nouvelles pratiques pédagogiques et ensuite les outiller nécessite une prise en compte de la complexité du domaine et surtout de la diversité des relations qui relient les acteurs, les contextes. La conception de ces outils devait donc être précédée d'une analyse

préliminaire exhaustive et pour cela nous avons fait appel à deux théories :

◇ **La théorie de l'activité**
Elle a été proposée par L. Vygotsky à partir de travaux collectifs démarrés dans les années 1920. Cette théorie reconnaît l'unité essentielle des systèmes, des utilisateurs et de leurs objectifs, et des contextes, y compris la communauté dans laquelle se déroule l'activité. Les interrelations entre tous les éléments d'une activité sont des supports pour la description de système de comportement complexe comme la pédagogie. Cette théorie est largement utilisée pour la conception des interfaces hommes-machines, comme par exemple dans les travaux de K. Kuutti [61], ou pour la conception de systèmes [62]. La figure 3.1 présente, par exemple, la structure de base d'une activité au sein d'une communauté : elle représente l'idée qu'un individu (*le sujet*) est assisté d'*outils* pour atteindre un *objectif* et peut accepter certaines *règles* afin de travailler dans une *communauté*, laquelle contribue à l'objectif via une *répartition du travail*. Selon Kaptelinin [63]

> 'One of the most important claims of activity theory is that the nature of any artefact can be understood only within the context of human activity - by identifying the ways people use this artefact, the needs it serves and the history of its development.'

◇ **La théorie de la structure rhétorique** (ou RST) [64], [65].
Cette théorie, initialement prévue pour la traduction automatique de textes, s'intéresse aux formes et organisations particulières afin d'atteindre des objectifs par un discours. Elle propose en particulier une explication de la cohérence des textes. Pour toute partie d'un texte cohérent, il existe une fonction, une raison plausible à sa présence, qui soit évidente pour le lecteur, et par ailleurs, le lecteur n'a pas le sentiment que des parties manquent à l'ensemble. Dans notre contexte d'enseignement, elle permet d'exprimer comment une activité pédagogique est établie et structurée par des contenus, des liens entre ces contenus et des modes spécifiques d'interactions entre l'enseignant et l'apprenant. Tout ceci naturellement est établi afin d'atteindre les objectifs pédagogiques fixés.

3.4.2 Proposition d'un modèle pédagogique pour IST CANDLE

Au sein de CANDLE, l'Institut de l'Education (*"Institute of Education"*) de Londres avait retenu la théorie de l'activité pour l'évaluation et cette sélection a également été réalisée pour la définition du contexte pédagogique. De plus, les relations complexes existant dans un système d'apprentissage ont fait l'objet de nombreuses études avec par exemple les systèmes d'activité (*activity systems*) de Y. Engestrom [66] [67] et les structures d'activités de Leont'ev [68] .

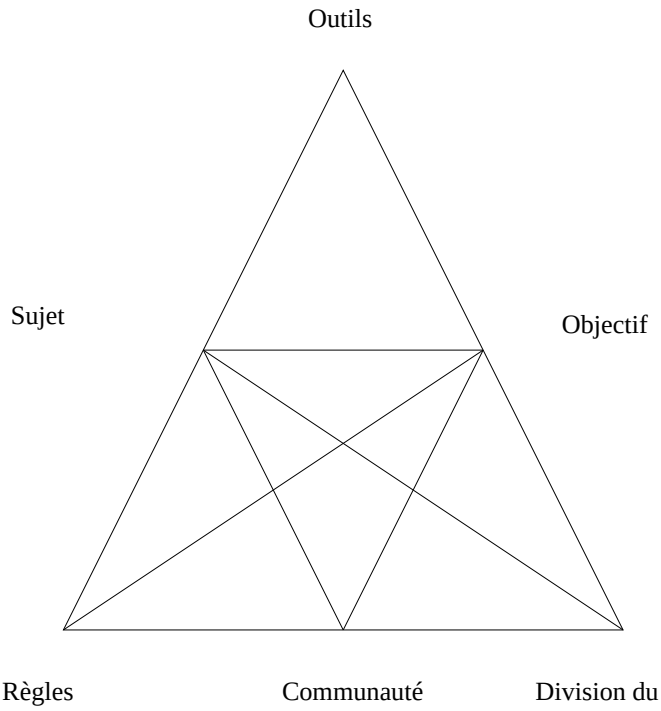


FIG. 3.1: Structure de base d'une activité d'une communauté (d'après Kuutti)

Les structures d'activités se réfèrent à la décomposition en trois niveaux des processus outillés. Les activités, ayant une finalité bien identifiée, peuvent être décomposées en une série d'actions qui représentent une séquence de processus discrets aux objectifs clairs en regard de l'activité globale, et qui sont nécessaires pour atteindre l'objectif final. Prises ensemble, ces actions ont une séquence, un enchaînement et une structure qui caractérisent l'activité. Enfin, il nous faut considérer les conditions pratiques qui permettent à ces séquences d'actions de se dérouler et de se terminer avec succès. Chaque niveau d'une activité interagit avec les autres pour former un ensemble de relations dynamiques qui se conditionnent mutuellement.

Les activités d'enseignement et d'apprentissage peuvent être exprimées dans ce cadre théorique. R. Lewis, dans [70] détaille cette approche dans le contexte d'une communauté d'apprenants et décrit les différentes structures d'activités associées. Chacune des activités pédagogiques a un but, avec un ensemble de résultats attendus(*"outcomes"*) en termes d'apprentissage, qui détermine l'organisation de l'activité et sa mise en œuvre. Ces activités ont donc une structure qui les caractérise et qui peut être représentée par une séquence d'actions à mener afin d'atteindre le but. Enfin, les conditions matérielles de l'activité (avec les rôles des intervenants et les outils qu'ils utilisent) sont importantes car conditionnent l'efficience de l'activité.

Un des buts de la RST est donc de décrire des textes et de proposer une explication de la cohérence du texte en établissant des liens entre les différentes parties ou fragments. Deux fragments, souvent adjacents, sont ainsi reliés de telle sorte que l'un joue un rôle spécifique

par rapport à l'autre. Mann et Thompson ont proposé une vingtaine de rôles ou de relations. La définition de ces relations repose sur deux arguments : le fragment noyau et le fragment satellite. Chaque relation est définie par des contraintes sur chacun de ses arguments et se traduit par un effet sur le lecteur. Prenons l'exemple de la relation "élaboration" : son effet sur le lecteur est de fournir dans le fragment satellite des éléments complémentaires (des détails ou une autre formulation) des propos contenus dans le fragment source de la relation. Son utilisation dans un cadre éducatif est évident. D'autres relations de la RST sont cependant moins adaptées à notre contexte car trop enracinées dans la traduction de textes.

L'intérêt d'un cadre formel tel que la RST, pour l'expression des relations sémantiques, repose sur l'utilisation d'un vocabulaire commun (celui des liens), dont la sémantique est unique, ce qui améliore la compréhension de la structure narrative De plus, elle favorise le partage de composants d'apprentissage car elle fournit en plus du contenu un fil directeur et les intentions de l'auteur.

Nom de la relation	**Noyau**	**Satellite**
Arrière-plan (*"Background"*)	texte dont la compréhension est facilitée	texte servant à faciliter la compréhension
Élaboration (*"Elaboration"*)	information de base	information supplémentaire
Préparation (*"Preparation"*)	texte qui va être présenté	texte préparant le lecteur à anticiper et à interpréter le texte qui va être présenté
Motivation (*"Motivation"*)	une action	information destinée à accroître chez le lecteur le désir d'accomplir l'action

TAB. 3.1: Quelques relations de la RST

Le modèle pédagogique sous-jacent proposé dans CANDLE associe la RST et la théorie de l'activité. Comme indiqué ci-dessus, ce modèle permet de décrire les différents éléments qui structurent les activités pédagogiques. Nous avons vu enfin que la référence aux structures d'activités permettait l'introduction de trois niveaux et de caractérisations associées (les aspects intentionnels, structure et matériel qui correspondent aux activités, actions et opérations).

Ces spécificités liées au domaine et aux approches de CANDLE doivent naturellement être décrites et nous avons alors le modèle pédagogique suivant :

Niveau	**Modèle pédagogique**
Activité - but	Buts, résultats, domaine étudié, évaluation
Action - structure et séquence de l'activité	Structure des actions et liens entre organisation et outils selon le système d'activités
Opérations - conditions	Contexte, ressources, règles, conditions matérielles

TAB. 3.2: Modèle pédagogique proposé pour CANDLE

Ce modèle simple correspond bien aux approches classiques suivies par les enseignants et surtout pouvait être traduit directement avec CAT.

3.4.3 Les méta données et leurs supports

Dans le cadre de IST CANDLE, nous aurions pu adopter directement les propositions de IEEE LOM, de CanCore [71] ou de ARIADNE. Cependant, nous étions dans une configuration différente. En effet, ces standards sont généralistes et de ce fait abordent de nombreux domaines. Dans le projet IST CANDLE, nous sommes dans une configuration spécialisée avec des approches spécifiques notamment en pédagogie. Parmi les éléments spécifiques nous pouvons citer en particulier :

- ◇ Le domaine est bien délimité : celui des réseaux des télécommunications. La description du contexte devra tenir compte au mieux de cette restriction et surtout pourra être utilisée pour "naviguer" (explorer) et trouver ainsi des documents à la thématique proche. Ceci impose en particulier une représentation du domaine des télécommunications et des moyens pour l'utiliser. Cette notion de domaine doit enfin être considérée comme un paramètre du système afin de pouvoir anticiper des adaptations à d'autres domaines scientifiques ou non.
- ◇ Les modèles pédagogiques de CANDLE prennent une part importante dans le projet et, du fait de la multiplicité des modèles, nous souhaitons les décrire afin que l'utilisateur puisse, par exemple, trouver un composant avec une approche pédagogique donnée. Ce dernier point permettra en particulier de mieux décrire le contexte de création (ou de l'utilisation) du composant et apportera ainsi des indices précieux à des futurs utilisateurs potentiels.
- ◇ Enfin, et surtout, le modèle pédagogique proposé dans CANDLE fait référence aux trois niveaux décrits dans la section précédente et doivent d'une manière ou d'une autre être décrits lors de la description et de l'indexation de chaque composant.

Le modèle proposé s'inscrit dans la lignée des travaux de normalisation entrepris par IEEE LOM : l'exigence d'interopérabilité (de compatibilité) avec les autres répertoires existants

peut ainsi être assurée. Les principales extensions de notre modèle de méta données sont liées aux spécificités du modèle proposé par CANDLE pour la pédagogie. Les méta données liées à la pédagogie sont naturellement dérivées depuis le modèle précédent que nous avons proposé (cf dans la table 3.2. Chacun des niveaux du modèle pédagogique se décline en autant de groupes de méta données spécifiques détaillées dans le tableau 3.3.

Niveau	**Modèle pédagogique**
Activité - but	Buts : description de l'activité, avec des mots clés et des buts Résultats : connaissances, compréhension, compétences acquises Domaine : précisé par l'ontologie du domaine Méthode d'évaluation : examen, projet ...
Action - structure et séquence de l'activité	Structure des actions Rôle : explication, élaboration, description, etc. Outils : communication, simulation
Opérations - conditions	Contexte : lieu, niveau, participants Ressources utilisées Prérequis : connaissances et compétences spécifiées par l'ontologie

TAB. 3.3: Modèle de méta données pédagogiques proposé pour CANDLE

Ce modèle simple correspond bien aux approches classiques suivies par les enseignants. Le modèle complet des méta données proposées pour IST CANDLE est présenté dans [72].

En ce qui concerne les outils et supports pour les méta données, nous avons deux catégories bien distinctes :

- ◇ Les outils d'indexation : qui permettent d'attacher des valeurs de méta données à du contenu. Dans notre cas particulier, il s'agit de décrire les composants d'apprentissage selon les différentes vues du modèle de méta données.
- ◇ Les outils de recherche qui permettent d'explorer une base de composants d'apprentissage et d'exprimer des critères de sélection.

Notre contexte de conception et de développements d'outils, et pour les outils supports aux méta données en particulier, était particulièrement instable. En effet, le modèle de méta données ne s'est pas figé dès le démarrage du projet mais il est le résultat de plusieurs itérations et de consensus. Les méta données liées à la pédagogie sont apparues la deuxième année du projet. Ce besoin d'évolution rendait donc impératif une solution paramétrable pour laquelle le modèle de méta données pouvait être modifié sans refaire les outils logiciels.

3.4.4 Ontologies et CAT

Dès le démarrage du projet IST CANDLE, nous avons souhaité proposer une architecture innovante nous permettant, d'une part, d'atteindre nos objectifs et les résultats attendus, et, d'autre part, de mener des recherches et d'explorer de nouvelles approches. Pour cela, nous avons retenu à partir de 1998, des techniques à base de représentation de connaissances et d'ontologies en particulier. Ceci correspond d'une part au domaine d'expertise et d'étude du département IASC mais aussi apparaissait comme un terrain porteur et favorable à l'extension de nos travaux de recherche. Le lecteur intéressé pourra trouver dans [58] une justification plus détaillée du choix des ontologies.

Un des problèmes soulevés porte sur la signification des concepts lors d'un travail collaboratif ou lors de partage de connaissances et de pratiques. Par exemple, il est nécessaire qu'un même concept soit utilisé pour représenter la même chose dans l'indexation de différents documents, or ceci est difficile à obtenir avec une indexation en texte libre. La solution est obtenue avec l'un des composants du Web Sémantique : les collections d'informations appelées ontologies [73]. Une définition d'ontologie est donnée par Gruber [74] : "une ontologie est une spécification explicite d'une conceptualisation". Elle est constituée d'un ensemble de définitions, de primitives, de représentation de connaissances spécifiques à un domaine particulier exprimées à l'aide d'un formalisme particulier. Dans ce formalisme on trouve les notions de concepts, de relations entre ces concepts, de fonctions, etc. Gruber définit l'ontologie comme le fait de conceptualiser l'univers du discours et les relations pertinentes dans cet univers.

L'architecture de CAT repose sur une architecture modulaire à base d'ontologies et utilise, en particulier, trois ontologies : un schéma de méta données, un modèle du domaine et une ontologie des liens sémantiques. La gestion des ontologies repose sur OntoBroker [75] [76] qui possède un système de requêtes fondé sur un moteur d'inférence. OntoBroker est un logiciel de gestion de connaissances qui permet le filtrage et la recherche d'information dans l'espace d'information comme dans les ontologies. Les trois ontologies principales définies et utilisées par CAT sont les suivantes :

- **Le modèle de domaine :**
  Le modèle de domaine - ou ontologie de domaine - représente l'ensemble des concepts du domaine de l'application ainsi que les relations entre ces concepts. Pour CANDLE, le domaine des cours concerne les réseaux et les télécommunications. Le modèle de domaine permet de décrire le contenu des fragments au sein des méta données qui lui sont associées (voire les connaissances du lecteur au sein d'un modèle utilisateur [77] [78]). Le modèle du domaine peut aussi être considéré comme une ressource pédagogique et être présentée à l'apprenant.
- **Le modèle de méta données :**
  L'ontologie code les différentes rubriques de méta données et les domaines de valeurs possibles pour chaque champ. Le codage du modèle à l'aide d'OntoBroker permet de

plus d'obtenir une série d'outils pour l'indexation des composants d'apprentissage et de réagir aux modifications du modèle sans avoir à réécrire les outils.

- ◇ **Les liens de la RST :**
  Initialement, nous avons repris les différents liens de la RST, et donc adaptés au texte narratif. La définition de différents modèles pédagogiques a mis en évidence la nécessité de nouveaux liens afin de traduire de nouvelles relations entre composants. Ici aussi, l'avantage principal du codage de ces liens par une ontologie dans OntoBroker permet l'évolution de la nature des liens.

Cette architecture nous a permis de développer très rapidement des prototypes et surtout de prendre en compte les diverses modifications du modèle de méta données principalement sans avoir à modifier complètement les outils : le modèle de méta données apparaît en tant que paramètre et peut être modifié si besoin. De plus, ces ontologies nous ont permis d'étendre les outils pour la composition de cours (travaux de Ph. Tanguy [79] et S. Iksal [80] [81] autour des documents virtuels personnalisables).

3.4.5 Création de cours

Il s'agit ici de l'opération de création de contenu et ceci à partir des outils classiques utilisés par les enseignants (utilisation d'outils de type *PowerPoint* et *Word*, documents en HTML ou pdf, par exemple). Classiquement, à chacun de ces documents est associé un stockage sous la forme d'un fichier, géré localement ou sur une machine distante.

Implicitement, ces documents contiennent :

- ◇ une structure : séquentielle par exemple pour un diaporama powerpoint ou plus libre avec un document HTML. La structure séquentielle impose par exemple une lecture, ou une visualisation, continue des transparents l'un à la suite de l'autre.
- ◇ des intentions pédagogiques : ce document est destiné à une formation, a priori, avec des objectifs bien déterminés.
- ◇ des données diverses liées par exemple au support utilisé, à sa version, sa taille, etc.

Cette organisation en fichiers correspond aussi à une certaine granularité retenue par l'auteur, et qu'il souhaite voir apparaître dans un module plus important ou être réutilisé par d'autres.

Une autre situation, que nous n'abordons pas dans ce document, est la ré ingénierie de documents pédagogiques. Elle consiste, à partir d'un bloc monolithique existant, de le fragmenter en autant de documents réutilisables. Une approche spécifique est proposée dans le cadre de IST CANDLE pour y répondre [82].

Enfin, l'enseignant peut aussi récupérer des objets d'apprentissage en provenance de l'extérieur et qu'il a sélectionnés parce qu'ils répondent à ses attentes.

Dans toutes ces situations, l'auteur ou l'enseignant se retrouve avec un ensemble de documents (un ensemble de fichiers) qu'il faut assembler / composer afin de produire un cours et atteindre un objectif pédagogique donné.

3.4.6 Assemblage et composition

L'assemblage et la composition de cours consiste à rassembler les différents fragments existants et produire un document pédagogique cohérent. Pour cette étape, l'utilisation de la RST permet au créateur d'expliciter une structure de cours que nous appelons structure narrative. Une structure narrative est un graphe orienté sans circuit, représenté par un ensemble de composants et de relations. L'un des composants a le statut particulier de point d'entrée dans la structure.

Les composants sont reliés entre eux par une relation sémantique extraite de la RST. La figure 3.2 donne un aperçu de l'utilisation de CAT lors de la création d'un cours : les différents écrans présentent la hiérarchie des contenus, qui peuvent être ou non visualisés.

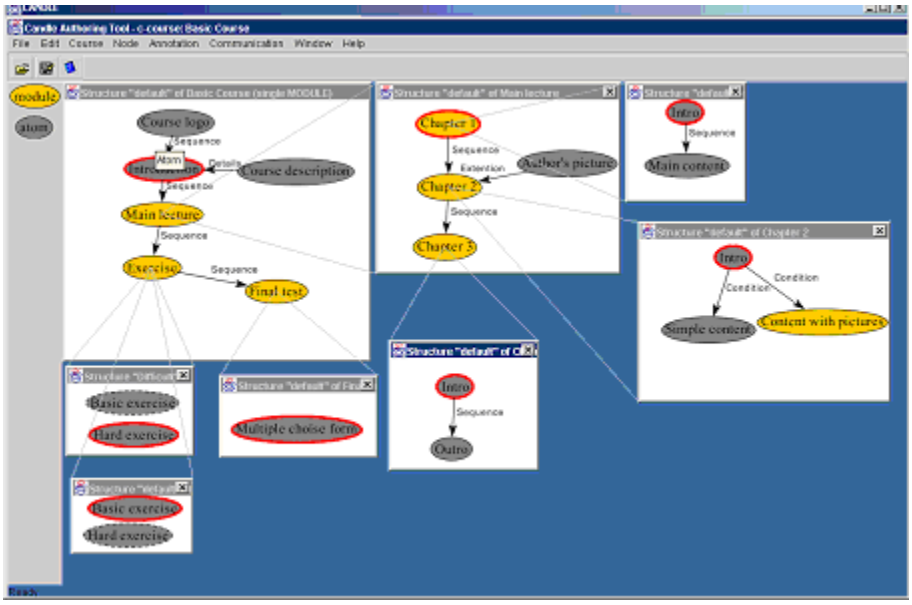


FIG. 3.2: Un exemple de création de cours avec CAT

Cette structure, avec une représentation à l'aide d'un graphe, permet de saisir l'organisation générale du document pédagogique et surtout les intentions du créateur. Nous verrons aussi que son association à un modèle pédagogique permet de prendre en compte différentes approches pédagogiques.

Enfin, cette structure est récursive. Un composant peut lui-même se composer de composants structurés en interne par un réseau de liens sémantiques.

Cette approche de la pédagogie par la présentation de la structure du cours présente de nombreux avantages :

- ◇ Au premier niveau, l'utilisateur peut obtenir une vue globale du composant, de son contenu, des liens et de leurs natures entre ces composants.
- ◇ Le graphe présente un formalisme clair et compréhensible pour les enseignants scientifiques mais aussi pour les experts en sciences de l'éducation : les différents nœuds représentent les différentes parties du cours ou de l'activité et les liens sont étiquetés par les relations de la RST représentent l'organisation et l'enchaînement de ces différentes parties.
- ◇ La structure récursive permet de se concentrer sur un niveau supérieur et ensuite d'explorer les différents composants, si besoin, et présente ainsi un niveau d'abstraction élevé.
- ◇ Le modèle se comprend aisément et surtout correspond aux pratiques des enseignants et créateurs de cours.
- ◇ Enfin, une structure de cours peut être proposée, elle constitue une lecture et un parcours entre les différents constituants. Notre approche rend possible l'existence de plusieurs parcours, qui pourront, par exemple, constituer des parcours pédagogiques adaptés à un type de lecteurs ou à un niveau particulier.

3.4.7 Indexation d'éléments de cours

> "Le développement de ressources éducatives utilisables au travers de réseaux de type Internet suppose que de telles ressources soient correctement décrites pour un référencement adéquat par les moteurs de recherche, que ces moteurs soient internes à une communauté d'utilisateurs ou généraux sur le Web." Monique Grandbastien [83].

L'indexation de documents est déjà largement utilisée par les bibliothèques et autres centres de ressources et ceci depuis de très nombreuses années ... L'enjeu est ici de prendre en compte les particularités du document pédagogique vis-à-vis des autres documents. En effet, implicitement ou explicitement, il fait référence à :

- ◇ un niveau d'étude ou de qualification, ou à un diplôme en cours de préparation.
- ◇ des prérequis qui sont autant de contraintes sur les connaissances et compétences de l'apprenant pour pouvoir accéder au document.
- ◇ des effets ou conséquences : la lecture du document et sa compréhension modifie, a priori, l'étendue des connaissances de l'apprenant ou du lecteur. Ces acquisitions lui permettront en outre d'accéder à d'autres ressources ou documents.
- ◇ un modèle pédagogique suivi par le créateur du document.

L'indexation est une opération critique pour assurer l'accès, la recherche et finalement la réutilisation des composants d'apprentissage : les différents champs et index doivent être renseignés précisément afin de décrire au mieux le composant d'apprentissage, sa nature, son contexte d'utilisation, etc. A ce titre, elle fait partie des critères de qualité qui doivent être pris en compte pour le composant d'apprentissage. L'utilisateur doit donc être assisté pour la réaliser et ceci implique par exemple :

- ◇ une présentation du modèle et des différentes rubriques,
- ◇ des contraintes pour ne pouvoir remplir que les valeurs anticipées,
- ◇ des traitements automatiques.

Ces aides ont été mises en œuvre dans le cadre du projet et ceci à partir des travaux de S. Iksal autour d'Obhutt.

- ◇ L'outil d'indexation présente sous forme de menus déroulants les différentes rubriques du modèle : chaque champ est ensuite présenté par un court texte, parfois trop compact, et par la liste des valeurs possibles. La personne, qui indexe, choisit alors la ou les valeurs retenues à partir d'une sélection de valeurs possibles présentées dans une liste. Cette contrainte de choix limite les erreurs dues par exemple à des zones de texte libre.
- ◇ L'indexation de nombreux composants se présente surtout comme une tâche fastidieuse, voire répétitive. La structure d'un document réalisé avec CAT apparaît sous une forme hiérarchique et de ce fait certaines méta données peuvent se déduire ou s'hériter d'un niveau supérieur. Nous avons proposé à cette fin la notion de patron (ou *"template "*) et de valeurs par défaut qui permettent de définir, pour certains champs uniquement, des valeurs par défaut qui sont insérées automatiquement dans les méta données à moins que l'utilisateur n'indique d'autres valeurs.

3.4.8 Visualisation et accès

Le document et ses composants une fois créés par l'auteur, rendus accessibles pour la communauté, doivent aussi être accédés par les utilisateurs dans un contexte d'apprentissage. L'auteur a prévu un parcours pédagogique qui doit au final être parcouru par l'apprenant.

Il ne s'agit pas ici d'un système complet de délivrance de cours (*"course delivery system"*) mais d'un composant d'un tel système. Pour notre cahier des charges, nous avions imposé la contrainte suivante : le poste de l'étudiant ne doit pas nécessiter d'outil particulier et seul un browser web est nécessaire.

A partir des travaux de Ph. Tanguy [79] et S. Iksal [80] [81] pour le projet ICCARS [84][85] et la conception de dossiers thématiques de journalistes, nous avons adapté les outils existants et permis ainsi de visualiser un parcours pédagogique.

La figure 3.3 présente une copie d'écran obtenue lors de la visualisation d'un cours. L'organisation de l'écran laisse apparaître l'organisation suivante :

- ◇ La partie centrale se compose du texte initial tel que rédigé par son auteur. Ce contenu est extrait du nœud courant lors du parcours (de la navigation). Les "ascenseurs" permettent de déplacer la zone de visualisation.
- ◇ La partie de gauche présente un aperçu de l'organisation du document pédagogique. Le nœud courant apparaît en surligné et surtout permet de visualiser (et de s'y déplacer directement si besoin) les nœuds voisins mais aussi les prédécesseurs dans la hiérarchie. Ce thème de localisation nous semble pertinent dans un parcours pédagogique mais son efficacité n'a pu être évaluée.
- ◇ Le bandeau supérieur fournit une vue plus limitée de la navigation avec le nœud courant, au centre, le nœud précédent et le suivant.

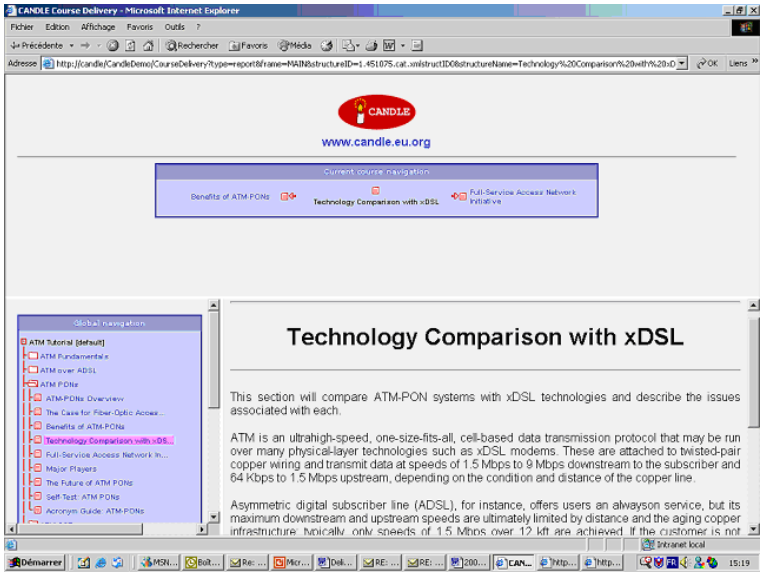


FIG. 3.3: Copie d'écran lors d'un accès à un cours

3.4.9 Synthèse et évaluation de CAT

La conception de CAT puis sa mise en œuvre lors du projet IST CANDLE ont permis de proposer une démarche outillée pour la création et le partage de cours. L'association conjointe de la théorie de l'activité et de la RST donne à la communauté EUNICE un cadre intéressant qui favorise les pratiques communes et de ce fait les échanges aussi bien de contenus que d'approches.

Enfin, l'utilisation sous-jacente des ontologies a nécessité des confrontations intéressantes dans la communauté des enseignants et ceci a abouti à la réalisation d'une ontologie du domaine des réseaux.

La figure 3.4 présente les différents aspects et fonctions de l'outil CAT :

- ◇ L'écran supérieur et central présente la structure du cours ou de l'activité pédagogique,

avec ses différents composants et ses liens de la RST.

- ◇ L'écran inférieur de gauche présente l'outil d'association de méta données à un composant. Les différents onglets correspondent aux grandes sections du modèle de méta données retenu pour IST CANDLE.
- ◇ L'écran inférieur à droite présente le raccrochement de contenu aux différents nœuds de la structure. Le créateur dispose en particulier d'un outil qui lui permet d'explorer une arborescence de fichiers.

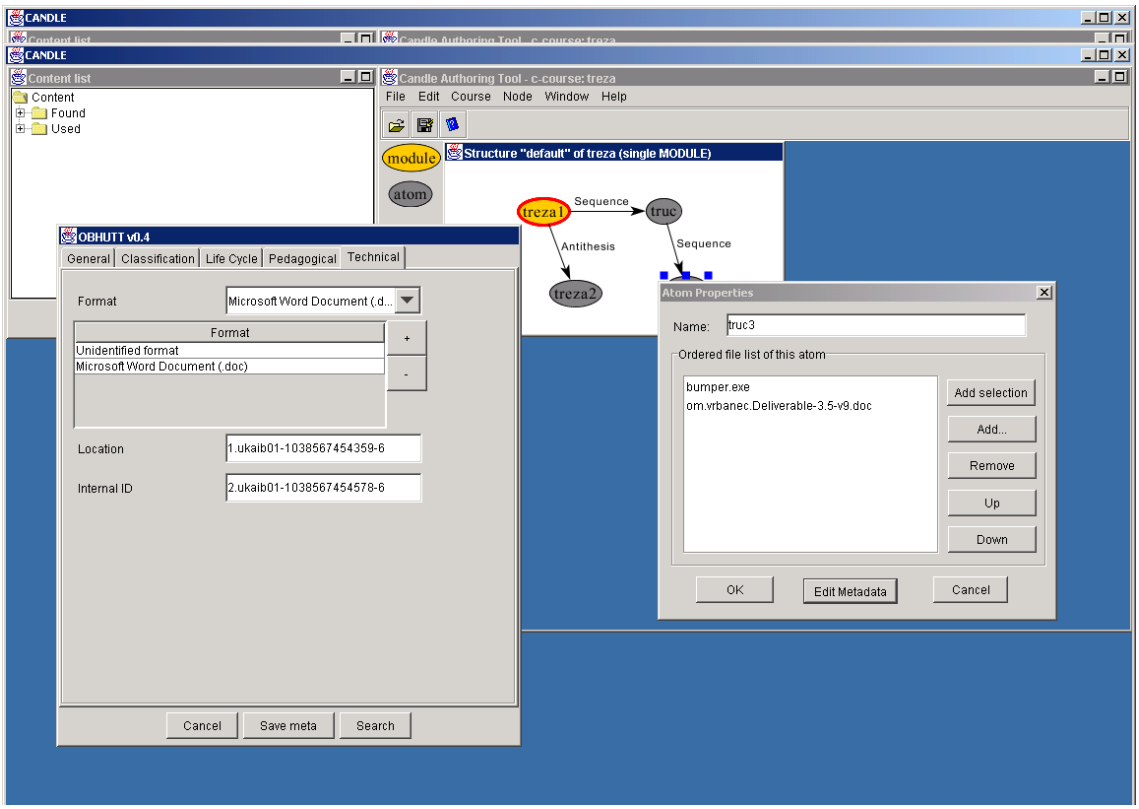


FIG. 3.4: Fonctionnalités de CAT

3.5 Question ouvertes et premiers éléments de réponse

- ◇ **La cohérence de l'assemblage ou peut-on tout assembler ?**
  Cette question est difficile surtout dans un contexte d'enseignement. En effet, l'enseignant peut être amené à provoquer des ruptures ou des digressions importantes, soit pour provoquer l'élève, soit au contraire pour lui permettre d'établir des liens avec des thèmes ou domaines connexes mais non directement liés. Il est donc difficile d'assurer la notion de cohérence thématique en examinant uniquement la cohérence des thèmes abordés par les différents composants. Il en est naturellement de même pour assurer la cohérence pédagogique d'un assemblage : la notion de rupture est bien identifiée dans certaines approches pédagogiques (comme la confrontation) justement afin de susciter l'attention de l'élève. Si la définition même de ces règles semble difficile dans

un cas général, il est naturellement envisageable d'assurer la vérification de telle ou telle propriété retenue par l'enseignant. L'architecture sous-jacente de CAT, à base d'ontologies, permet l'expression de telles règles en logique du premier ordre et fournit également le moteur de vérification.

◇ **Extraction du contexte et recopie des méta données**
Un composant d'apprentissage est créé par un certain auteur et dans un contexte : une partie de ce contexte est exprimée par les méta données ainsi que son contexte pédagogique de création ou d'utilisation. Ce composant apparaît aussi dans une structure narrative créée aussi par l'auteur : il s'inscrit ainsi dans un contexte particulier avec des liens éventuellement de dépendances avec d'autres composants. Son extraction, sa copie pure et simple, pose le problème de l'identification de son contexte intial de création. En effet, si les méta données sont copiées lors de l'extraction du composant, il n'en est pas de même pour la structure dans laquelle il apparaissait initialement. Dans la version actuelle de CANDLE, on perd ainsi des informations précieuses sur son contexte d'utilisation.
Cette structure initiale véhicule beaucoup d'implicite qui pourrait être codée dans une extension du modèle des méta données. On pourrait alors prévoir un codage du voisinage du composant et des liens qui unissent ces composants. La question est de savoir jusqu'où aller surtout s'il s'agit d'une structure complexe composée de plusieurs niveaux hiérarchiques.
Avec une telle approche, le composant apparaîtrait alors comme une entité avec des ports d'entrée / sortie : les entrées modélisant les prérequis et autres dépendances avec des modules et les sorties représentant les concepts et compétences abordées dans le composant. Cette analogie nous permettrait alors de nous resituer dans une thématique similaire à celle exprimée pour les composants logiciels Ada, présentée au chapitre précédent.

◇ **Duplication ou non du composant réutilisé**
Lorsqu'un composant d'apprentissage est réutilisé (par son auteur ou un autre), il est automatiquement dupliqué. Ce choix pour IST CANDLE est lié au modèle de stockage retenu : les composants peuvent résider sur des machines différentes et distantes et toute panne ou indisponibilité du réseau rendrait alors inaccessibles ces composants. Afin d'éviter cette situation, le composant est donc dupliqué et ne garde qu'une référence vers son composant initial. Nous n'évoquons pas ici le cas de la destruction pure et simple du composant initial qui lui génèrerait d'autres types de difficultés. Il se pose naturellement le problème de la cohérence des copies et de la prise en compte des modifications. Du fait de l'existence d'un lien vers le composant d'origine conservé comme méta donnée, celui qui duplique peut vérifier l'existence d'une version plus récente du composant. La réciproque n'est pas vraie à moins de garder des traces de toutes les opérations de copie d'un composant mais cette copie n'implique pas que le

composant soit effectivement réutilisé.

◇ **Le problème des droits et de la propriété intellectuelle**
Ce problème est abordé avec beaucoup de verve par D. Wiley dans son article *"The teacher's outrageous claim of intellectual property"* [42]. Un composant d'apprentissage nécessite un travail non négligeable et son auteur a donc investi son temps, son énergie afin de le produire ... et son employeur (Université ou entreprise) son salaire et une expertise. Avoir son cours, ou une partie de cours reproduite et insérée dans d'autres cours est une reconnaissance implicite des qualités et mérites de son auteur (que ce soit pour le contenu, pour la pédagogie utilisée ou plus simplement parce que ce cours existe !). Cependant, cette vision 'idéale' se heurte à la réalité : la difficulté de partager et de voir ses propres efforts directement intégrés ailleurs, et surtout les réticences des employeurs à voir ainsi leurs produits disparaître sans rétribution.
Initialement, le projet IST CANDLE impliquait les partenaires d'EUNICE pour lesquels des intentions d'échanges et de partages apparaissent dans la charte signée par ses membres. Si les échanges entre personnes ne posent généralement pas de problème, il n'en est pas de même au niveau institutionnel. Enfin, les partenaires industriels du projet ne sont pas membres de EUNICE et de ce fait sont dégagés de ces engagements. Enfin, malgré les injonctions de notre coordonnateur de projet à Bruxelles (*"project officer"*) nous n'avons pas pu atteindre la rédaction commune d'un IPR ("Intellectual Property Right") en trois ans.
Une solution consisterait alors à s'engager sous le manifeste du "Libre" avec la licence GNU.

◇ **Collaboration en ligne**

Nous n'évoquons ici que les outils de collaboration liés à la conception de cours ou de composants d'apprentissage par une communauté d'enseignants. Les outils de collaboration classiques sont en effet disponibles dans de nombreux environnements informatiques.
L'acronyme de CANDLE fait directement référence à la collaboration mais n'a pas été réellement abordé dans le cadre du projet. À partir d'une expérience impliquant les Universités de Twente, Barcelone, Trondheim et l'ENST Bretagne, dans le cadre d'une expérimentation (*"testbed"*) de CANDLE, nous avons établi des activités communes en vue de la réalisation d'un système de messagerie instantanée (depuis la conception jusqu'à son implémentation en passant par sa validation et son observation) et un programme de cours commun. Le projet impliquait donc les enseignants dans une première phase, puis les enseignants et les étudiants, qui devaient travailler à partir de spécifications fournies par une autre équipe et produire des documents et logiciels à destination d'une autre équipe. Les informations plus précises sont disponibles sur http ://iasc.enst-bretagne.fr/PEX/ et, pour la partie norvégienne, sur

http ://www.item.ntnu.no/˜rolvbra/CandleAssignment/). Pour la mise en place de ces activités, les quatre partenaires ont utilisé CAT pour structurer les étapes du projet, tout en restant physiquement dans leurs pays respectifs. L'utilisation conjointe de CAT et de NetMeeting s'est révélée particulièrement fructueuse.
Dans une précédente version de CAT, nous avions aussi installé un système à base d'annotations qui permet de sauvegarder les traces d'interactions entre les différents acteurs. Ces traces constituent une source d'informations pour ceux qui accèdent aux composants, ou au cours, pour la première fois, car véhiculent des informations complémentaires à celles présentes dans les méta données.

◇ **Adaptation à l'usager**

Tous les utilisateurs (enseignants ou apprenants) sont différents, que ce soit au niveau de leurs objectifs, de leurs préférences ou de leurs connaissances. Ce qui signifie qu'ils ont tous des attentes différentes quant aux documents qui leur sont fournis, à la manière dont ils sont présentés (tant sur la forme que sur leur enchaînement). L'adaptation / personnalisation d'un document nécessite de disposer d'informations sur l'utilisateur, qui pourront être décrites par un modèle utilisateur : il donne une représentation des caractéristiques de l'utilisateur sous un angle (un point de vue). Il doit comporter à la fois des informations permettant de filtrer et de sélectionner l'information pertinente pour l'utilisateur, mais aussi des informations concernant son utilisation du système et les documents ou fragments consultés. Ce modèle utilisateur est enfin mis à jour afin de tenir compte de l'évolution de ses compétences, par exemple.
S. Iksal [81] a proposé dans sa thèse une approche de composition de documents personnalisables dans le contexte de dossiers thématiques. Une mise en œuvre dans CANDLE pourrait être réalisée rapidement mais il reste à définir le modèle de l'utilisateur (c'est-à-dire comment caractériser l'apprenant dans son parcours d'apprentissage). Comment en particulier évaluer ses connaissances et ses compétences ? Le seul fait de consulter ou de lire un théorème ne suffit pas, hélas, à le comprendre ou à le maîtriser.
Un tel profil - modèle utilisateur - doit donc être couplé avec les phases d'évaluation de l'apprenant. La société Aleks (www.aleks.com) gère ainsi de tels profils bâtis à partir des théories de JC Falmagne et de ses espaces de connaissances [86].
Ce profil peut aussi être utilisé pour l'expression de pré conditions à l'accès à telle ou telle partie du cours : de telles conditions portent sur les connaissances ou compétences de l'apprenant et lui permettent, par exemple, progressivement, d'élargir son espace de connaissance.
En attendant, la notion de parcours que nous proposons dans CANDLE apporte un début de réponse en prévoyant des lectures du document en fonction de grandes classes prévues par l'auteur : débutant, expert, par exemple.

3.6 Synthèse du projet IST CANDLE

Le projet IST CANDLE s'est terminé en septembre 2003 et nous avons atteint les objectifs fixés initialement dans le contrat. L'architecture distribuée pour la gestion des composants de cours est fonctionnelle et permet les échanges entre les différents partenaires du projet. Les modèles pédagogiques proposés sont accompagnés d'outils supports. Cependant, de nombreux points noirs subsistent et sont de différentes natures. Nous indiquons ci-après quelques éléments qui restent à évaluer :

- ◇ Le schéma de méta données est compatible avec IEEE LOM et les extensions au modèle pédagogique ont été favorablement reçues par le comité de standardisation.
- ◇ La définition d'un schéma d'indexation et en particulier la définition du modèle de méta données est avant tout le résultat d'un consensus ... et s'il peut obtenir le consentement d'un groupe, son utilisation effective confronte l'utilisateur avec une réalité dont il n'avait pas nécessairement conscience. Nous avons donc expérimenté dans le projet IST CANDLE des réactions de réveil lors de l'utilisation effective du schéma de méta données.
- ◇ Le contenu initial de la base de CANDLE comprend environ 5000 composants d'apprentissage qui ont tous été étiquetés avec les méta données. Une analyse plus fine devra cependant évaluer la qualité des étiquettes utilisées.
- ◇ Le modèle de méta données tel que proposé par IST CANDLE est réduit mais se compose cependant de nombreux champs (environ 80) : chacun de ces champs fait l'objet d'une description complète dans un livrable [72] ainsi que de documents complémentaires (un guide d'utilisation et un manuel consacré à la pédagogie) lorsque ces champs apparaissent comme complexes, voire ambigus. Les méta données liées au modèle pédagogique apparaissent comme étant les plus difficiles à utiliser.
- ◇ RST et liens : plusieurs liens de la RST ont été rajoutés lors des évaluations et utilisations de CAT. Il reste cependant à figer le modèle. Le modèle de la RST apparaît d'une utilisation simple et compatible avec les pratiques des auteurs de cours. A partir de plusieurs cours réalisés, il apparaît cependant que la relation "séquence" de la RST est majoritairement utilisée. Ceci est révélateur d'une non prise en compte des modèles pédagogiques possibles avec IST CANDLE par CAT et surtout qu'il faut dépasser le simple séquencement d'activités.
- ◇ Nous avions défini des critères de qualité applicables pour les divers produits du projet, dont les composants d'apprentissage. Nous avons en particulier défini des critères à prendre en compte pour la validation de composants (voir section 3.3.4). A partir de l'expérience acquise, ce processus de validation doit être affiné et une grille de critères doit être définie comme par exemple celle définie dans le cadre du projet CLOE (http ://cloe.on.ca) [87] et [88].
- ◇ Collaborer entre enseignants est finalement difficile et nous avons rencontré de nombreux obstacles, non anticipés, comme ceux liés à la confidentialité et à la propriété intellectuelle. Les expériences et évaluations ont cependant montré une réelle implica-

tion des partenaires et surtout l'approbation d'une approche de partage qui demeure la seule à ce jour pour aborder la thématique de la formation en réseaux et télécommunications.

3.7 Perspectives sur un après IST CANDLE

Nous avons montré tous les avantages des composants de cours réutilisables et nous avons démontré la faisabilité dans le cadre du projet IST CANDLE. Les différents outils réalisés sont principalement destinés aux auteurs et nous pensons que notre approche de composition de documents de cours pourrait être étendue afin de développer un véritable environnement pour le lecteur. Les travaux de thèse de S. Iksal [81] pourraient être repris et adaptés afin de prendre en compte le profil utilisateur et réaliser ainsi des cours adaptés aux différents lecteurs.

Le réel développement de pratiques de réutilisation nécessite enfin des pratiques communautaires, que nous avons activées au sein du projet IST CANDLE lors des expérimentations entre quatre établissements universitaires. Cependant, il reste à continuer les expériences et surtout passer à l'échelle lors de validations plus étendues. Les pratiques à base de composants réutilisables sont principalement intéressantes si la base de composants est étendue : c'est-à-dire que la base doit comprendre de nombreux composants qui abordent les différentes thématiques du domaine concerné avec des vues différentes ou complémentaires.

Le projet IST CANDLE a défini des approches et pratiques communes pour l'enseignement en dernier cycle universitaire. Nous pensons continuer ces actions de partage et de coopération dans un cadre de recherche. Pour cela, nous souhaitons mettre en place une école doctorale virtuelle dans le domaine de la recherche en interfaces hommes machines et en sciences cognitives. Cet objectif se décline dans les thèmes suivants :

- ◇ Créer une communauté d'utilisateurs : il s'agit d'apporter aux différents intervenants de l'école doctorale virtuelle les moyens et approches leur permettant d'interagir et de partager leurs travaux de recherche.
- ◇ Constituer au sein de l'école doctorale virtuelle une ressource permettant des actions de recherche et de mise à disposition d'informations pour les différents intervenants.
- ◇ Offrir aux enseignants des outils simples et puissants leur permettant d'interagir avec des étudiants à distance dans le cadre de séminaires, ateliers et travaux de recherche.
- ◇ Constituer autour des enseignants et des étudiants des réseaux créateurs de contenus et d'acteurs de leur réutilisation / capitalisation.

Nous pensons que nous pouvons atteindre ces objectifs à partir de notre expérience avec IST CANDLE. Ce thème s'est concrétisé par la proposition de deux projets de recherche et développement : le projet IST DOCTE (soumis dans le cadre du 6ème PCRD, placé en *"short*

list" mais non retenu) et le projet VIRDOC en cours d'évaluation dans le cadre des actions MEGALIS des régions Bretagne et Pays de Loire.

CHAPITRE 4

Indexation, recherche et composition de services

4.1 Présentation de la thématique

Les systèmes distribués forment désormais l'ossature de nombreux systèmes informatiques et ceci depuis l'introduction massive des réseaux de télécommunications. En effet, les impératifs liés à la disponibilité des ressources, à la répartition de ces ressources sur des sites parfois éloignés rendent incontournables de telles architectures. Si cette présence massive permet de résoudre certains problèmes, elle en génère aussi de nouveaux jusqu'alors inconnus pour les systèmes centralisés. Nous ne reprendrons pas ici une analyse détaillée des avantages et inconvénients de ces approches, qui sont largement détaillés dans la littérature [89] [90].

Une des approches pour favoriser l'utilisation des systèmes distribuées est celle proposée par l'introduction d'intergiciel (ou *middleware*, ou bien encore appelé *enabling technologies*). Parmi les principales solutions technologiques dans ce domaine, nous avons en particulier étudié PVM [91] et MPI [35] puis OSF DCE [92] [93] et OMG CORBA [94] [95], ce dernier propose en particulier un ensemble de services qui sont mis à disposition des développeurs d'applications [96].

Les modèles de programmation ont naturellement évolué au fur et à mesure des développements des plates-formes mais aussi des avancées en génie logiciel. Le paradigme de la communication par message très vivace dans les premiers temps du fait de son adéquation avec le réseau sous-jacent, a progressivement laissé place à des paradigmes de plus haut niveau à base de RPC dans un premier temps, puis d'objets distribués [97] [95] [98] ou d'objets liés à une mémoire associative comme dans Linda [37].

Ces paradigmes de programmation à base d'objets distribués nécessitent des mécanismes de désignation afin de pouvoir référencer ou appeler des objets ou services distribués. Un objet ou un service peut ainsi être désigné par une adresse unique qui associe des

informations réseaux (numéro IP de machine, protocoles et ports, par exemple) et système, utiles pour accéder à la ressource. Cette adresse permet aux clients d'accéder à l'objet.

Une des difficultés du développement des applications distribuées est justement de mettre en place des services d'échanges d'adresses afin que les objets et services de l'application distribuée puissent communiquer et inter-agir. Ce thème présente une importance accrue dans un environnement mobile où les objets, services et clients se déplacent mais désirent toujours interagir entre eux.

La thématique que nous abordons dans ce chapitre est celle de la recherche et de l'indexation des objets et services dans les systèmes distribués. La thèse de O. Kassem Zein s'inscrit dans cette thématique et propose une approche flexible pour la recherche de services et l'étend avec des facilités pour la création de nouveaux services à partir de techniques de compositions.

Dans un premier temps, nous présentons les fonctions et services rendus par les serveurs de noms proposés par des intergiciels. Puis nous détaillons le modèle de méta données qui permet d'indexer des objets et services. Nous illustrons ensuite la mise en œuvre opérationnelle de notre approche dans le cadre d'un trader. Nous détaillons ensuite notre trader étendu qui associe au sein de la même entité des supports pour une description détaillée des services et objets et qui nous permet de composer des services. Après une synthèse de l'état de nos travaux, nous présentons des perspectives d'actions futures en lien avec ces travaux.

4.2 Services de nommage et d'annuaires

4.2.1 Service de noms pour les systèmes distribués

Le service de nommage de ressources est un élément clé dans tous les systèmes informatiques. C'est un annuaire de type pages blanches qui définit un espace de désignation pour retrouver les objets ou les services à partir d'un nom symbolique. Cet espace peut être structuré par un graphe de répertoires contenant des références sur les objets. Un serveur associe d'une façon unique un nom symbolique à un objet et un client utilise ce nom pour obtenir une référence sur l'objet en question.

4.2.2 Service de courtage pour les systèmes distribués

Le besoin pour les objets de se découvrir et de se localiser mutuellement est une nécessité. Le service de nommage y contribue en permettant aux clients/serveur de découvrir/publier les objets ou les services en leur associant des noms. Un autre service, qui fournit un mécanisme de recherche de services plus flexible que le service de nommage, est celui de

courtage (ou *"trading"*).

Il a été étudié en 1990 par ANSA [99]. Le groupe de travail ODP (*Open Distributed Processing*) a défini des spécifications de ce service [100]. En 1997, OMG a ajouté ce service à l'architecture de CORBA, ce qui renforce son rôle et son importance. Il joue un rôle primordial dans les systèmes distribués depuis qu'il lie les clients et les serveurs [101].

L'idée principale du service de courtage est d'avoir un médiateur qui agit comme un courtier entre les clients et les serveurs. En effet, c'est un annuaire de type pages jaunes qui permet à un service ou objet d'être découvert via une description basée sur des propriétés (caractéristiques). Un trader est un objet qui implémente le service de courtage.

Le principe de fonctionnement du trader se résume comme suit : un serveur, nommé aussi exportateur, enregistre les services qu'il désire rendre disponibles au niveau du trader en les caractérisant par un ensemble de propriétés. Les valeurs de ces propriétés sont stockées sous la forme d'offres (services offerts) dans le trader. Un client, nommé aussi importateur, qui désire un service offert interroge le trader en donnant les caractéristiques attendues.

Traditionnellement, le trader enregistre les services offerts dans ses bases de données. Il traite les requêtes de clients par une interface avec SQL ou un langage booléen. Il cherche, dans ses bases de données, les services demandés et il sélectionne les offres appropriées satisfaisant les requêtes de clients. Une fois que les services sont retournés avec leurs références, les clients et les serveurs communiquent indépendamment du trader. Le trader permet ainsi la recherche multi-critères pour découvrir des services.

Des implémentations de trader sont disponibles dans presque tous les intergiciels puisqu'il assure une fonction de médiation entre les clients et les serveurs pour l'accès aux ressources, aux objets ou aux services. Ces implémentations, comme celles d'Orbacus [102] ou de JacOrb [103], reposent sur une architecture de bases de données relationnelles avec une interface de type SQL. Or une telle utilisation du modèle relationnel nécessite la définition d'un schéma de base de données, sous la forme d'un ensemble d'attributs qui stockeront les différentes propriétés. La mise à disposition de primitives de définition du schéma de la base permet naturellement de prendre en compte différents modèles et de s'adapter ainsi au contexte de l'application.

4.3 Proposition d'un modèle de méta données pour les services

4.3.1 Classification des services

Avant de commencer la caractérisation et la description de services, l'intérêt de leur classification est double : d'une part, elle décrit ce qu'un service signifie, et d'autre part elle structure l'espace de services dans des classes qui peuvent être facilement caractérisées. Ces classes constituent un ensemble de méta données associé au service.

Plusieurs approches sur la classification de services ont été proposées dans le domaine de *Marketing* [104]. La classification de Lovelock [105] est l'une des approches appropriées à la description de service. Cette classification est basée sur un ensemble de questions dont nous en citons certaines :

- Qui est le destinateur direct du service ? Est-il une personne, un objet physique ou un logiciel ? Réciproquement, nous pouvons poser la même question sur qui délivre et fournit le service. De cette façon, nous obtenons les classes des services suivantes : humain-à-humain (*"human-to-human"*), humain-à-objet (*"human-to-object"*), objet-à-humain (*"object-to-human"*), objet-à-objet (*"object-to-object"*), logiciel-à-humain (*"software-to-human"*) comme un moteur de recherche, etc.
  Ces différentes classes montrent qu'il n'est pas nécessaire que la livraison d'un service implique une personne mais peut être une organisation.
- Quelle est la relation entre un fournisseur du service et ses clients ? Par exemple, est-ce qu'un fournisseur de service demande à ses clients une inscription pour pouvoir accéder au service ?
- Quelle est la nature de la demande de service ? Est-ce que les clients doivent faire une réservation ou sont ils servis d'une manière quelconque, par exemple *FIFO* (First Input First Output) ? En d'autres termes, quelles sont les politiques et les contraintes à prendre en compte pour pouvoir utiliser le service ?
- Comment le service peut-il être délivré ? Physiquement ou électroniquement ?

La classification de services pose donc plusieurs questions. Les réponses à ces questions permettent de caractériser et de décrire un service. Par exemple, les propriétés de fournisseur du service, les moyens d'accès au service (électronique ou physique), le service demande une réservation ou non, etc. répondent aux questions posées et peuvent être utilisées comme des index qui décrivent un service.

La description de service a pris une actualité importante du fait de l'émergence des web services et des architectures qui utilisent ces approches à grande échelle. Lors de la conception de notre modèle d'indexation nous n'avions pas pu prendre en compte le modèle proposé par

DAML-S [106] [107] [108]. Ce dernier semble compatible, pour les grandes lignes, avec nos travaux.

4.3.2 Comment décrire un service ?

Pour rechercher un service, fourni par un fournisseur, dans un répertoire, un utilisateur doit identifier un ensemble de caractéristiques de service, et peut donner une valeur désirée à chacune d'entre elles. En se basant sur ces valeurs, une liste de services offerts est retournée et permet au client de choisir ceux qui sont appropriés.

La description ou la caractérisation d'un service est donc nécessaire pour l'interrogation et la sélection de services et doit être prise en compte dans l'étape de l'indexation et la publication. Si le service est bien indexé, l'interrogation sera possible et permettra ainsi de rechercher le ou les services désirés.

4.3.3 Description statique d'un service

Afin de caractériser un service, nous avons besoin de répondre à plusieurs questions que nous avons citées dans la section précédente. Par exemple, un client doit savoir ce que fournit le service (une description de la fonction du service), son fournisseur, sa localisation et comment un client peut y accéder. Le prix de service, les moyens de paiement, les outils de demande et de livraison, la qualité de service de point de vue utilisateur peuvent être pris en compte. La délivrance du service peut être aussi garantie et décrite. Le modèle complet de méta données que nous proposons est décrit dans [109].

4.3.4 Description statique par interface d'un service

L'interface du service définit les moyens et les méthodes pour interagir et communiquer avec le service. La description de l'interface signifie la description de ses opérations et leurs paramètres, ses attributs, etc. Elle permet aux utilisateurs de découvrir les opérations disponibles et de comprendre comment ils peuvent formuler et exécuter les requêtes d'invocation du service lors de l'exécution.

Deux modes d'invocation de service existent : statique et dynamique. Dans le mode statique, un utilisateur connaît l'interface du service lors de la compilation et peut déterminer les opérations nécessaires à appeler. La description de l'interface est utilisée comme un contrat par l'utilisateur du service et sert à la génération des différentes souches. De plus, certaines vérifications peuvent être réalisées à la compilation et des erreurs pourront donc être détectées puis corrigées.

Dans le mode dynamique, un utilisateur ne connaît pas l'interface du service et doit construire et formuler ses requêtes d'invocation dynamiquement. La description de l'interface du service est nécessaire puisqu'elle permet aux utilisateurs de découvrir l'interface à l'exécution et de bâtir leurs requêtes. OMG CORBA propose un tel mode avec DII (*"Dynamic Invocation Interface"*) mais qui est peu utilisé du fait de sa complexité.

Nous choisissons le langage IDL défini par OMG CORBA comme support pour décrire l'interface de service. Il permet de définir l'interface indépendamment de son langage d'implémentation. En effet, ce langage permet d'exprimer, sous la forme de contrat IDL, la coopération entre les fournisseurs et les utilisateurs de services, en séparant l'interface de l'implémentation des services et en masquant les problèmes liés à l'interopérabilité, l'hétérogénéité et la localisation de ceux-ci.

L'interface repository *IR* de OMG CORBA permet d'enregistrer cette description et de répondre aux demandes des utilisateurs. Pour y accéder, une bibliothèque spécifique existe et fournit des primitives bien différentes de celles du trader. Notre approche est donc de fournir des services similaires à ceux proposés par *interface repository*, en enrichissant le trader. La description de l'interface apparaissant alors comme une caractéristique supplémentaire de l'objet ou du service.

4.3.5 Description du service par son comportement

Les deux premières descriptions de service sont bien connues et largement utilisées, mais ne couvrent qu'une petite partie de la caractérisation d'un service. En effet, il nous manque une description de ce que fait le service et de son comportement. Il nous faut donc une telle description qui soit exploitable aussi bien par la machine que par l'homme. De plus, notre approche d'indexation vise à annoter le service par des méta données ou autres, qui pourront ensuite servir pour la recherche. Nous devons donc proposer un formalisme qui soit adapté à une forme d'interogation et de recherche.

4.4 Description du comportement d'un service

Après avoir découvert un service en interrogeant ses propriétés statiques, un client peut découvrir son interface et connaître ainsi les opérations disponibles. Ces opérations déterminent ce qu'un client peut réaliser avec le service. En d'autres termes, elles indiquent les fonctionnalités du service. Dans ce contexte, le client a besoin de déterminer comment utiliser et invoquer le service. Cette notion conduit vers la définition du comportement du service : il indique ce que le service peut faire et comment il fonctionne en interne. Il détermine les ordres par lesquels les opérations du service peuvent être exécutées.

Pour qu'un client puisse utiliser le service, il doit découvrir son comportement. Donc, la description comportementale du service devient nécessaire. Cette description, que nous appelons encore description par des propriétés dynamiques, doit fournir des informations permettant au client d'utiliser et d'invoquer le service. Elle comporte, par exemple, les enchaînements et les séquences correctes par lesquels les opérations doivent être invoquées. Dans ce sens, nous devons définir les fonctionnalités des opérations du service qui permettent de déterminer comment les opérations peuvent être enchaînées. La description du comportement du service inclut donc la description des fonctionnalités de chaque opération et la possibilité de la connecter avec d'autres opérations.

Pour implémenter le comportement de service, nous devons utiliser un mécanisme qui permet d'enchaîner ses opérations en fonction de leurs fonctionnalités. Nous avons choisi les formalismes SDL (*"Specification and Description Language"*) [110] et l'automate d'interface (*"Interface Automata"*) [111] qui sont utilisés dans le domaine des télécommunications. Ces approches permettent de décrire le comportement d'un composant logiciel par un automate d'états finis. Ils décrivent les fonctionnalités d'une opération par ses entrées et ses sorties, c'est-à-dire, ce que l'opération peut prendre en entrée et ce qu'elle peut fournir en sortie. L'opération est donc vue comme une fonction d'entrées/sorties. À partir de cette description, un automate peut être construit en associant des états et des transitions aux opérations et leurs fonctionnalités et peut aussi être décrit comme une fonction d'entrées/sorties.

Dans le domaine des services Web, le langage WSDL [112] permet de décrire le comportement d'un service Web. Il décrit les opérations par des messages des entrées/sorties et par des post-conditions et pré-conditions qui indiquent les conditions dans lesquelles une invocation est faisable ou non. À chaque invocation d'une opération, il s'agit d'évaluer ses pré-conditions. L'interogation et l'invocation de services Web sont donc des tâches complexes. En utilisant les automates, nous allons proposer un modèle plus simple pour la description du comportement du service du point de vue utilisateur et pour la découverte de services par interrogation.

Ce niveau de description est complémentaire à ceux fournis par les méta données statiques et par l'interface. Ces trois niveaux fournissent des index décrivant un service d'une manière globale et peuvent être utilisés par les clients pour interroger les services. De ce fait, le mécanisme de la recherche de services est devenu de plus en plus élaboré et permet de combiner des critères différents.

Dans cette section, nous présentons les formalismes SDL et l'automate d'interface et les approches utilisées pour décrire un composant logiciel par un automate. Nous décrivons notre approche qui est basée sur ces deux formalismes pour décrire le comportement d'un service et la possibilité de l'interroger.

4.4.1 SDL

ITU-T (International Telecommunications Union-Telecommunications) a défini le langage SDL qui est un langage formel pour spécifier et décrire le comportement des systèmes de télécommunications. Les spécifications et les descriptions doivent être formelles, c'est-à-dire qu'il doit être possible de les analyser et de les interpréter sans ambiguïté.

La puissance de SDl est sa capacité à décrire la structure, le comportement et les données d'un système.

La description de comportement est fondée sur les machines à états finis (automates) étendues communiquant par message. Les domaines d'application principaux pour lesquels SDL a été conçu sont nombreux (les systèmes distribués, les systèmes temps-réels, les services de télécommunications, etc.) et les environnements ont été développés en y incluant des systèmes de vérification et de validation.

Un système SDL contient les concepts relatifs aux principes généraux de structuration du langage. Il s'agit des concepts de système, de bloc, de processus et de procédure. En effet, les blocs sont connectés entre eux à l'environnement par des canaux. À l'intérieur de chacun des blocs, il y a un ou plusieurs processus. Ces processus communiquent entre eux par des signaux et sont supposés s'exécuter en parallèle. SDL supporte l'appel de procédure à distance qui permet à un processus d'appeler une procédure s'exécutant sur un autre processus.

Le comportement d'un système SDL est décrit par le comportement de ses processus et la communication entre eux. Dans notre étude de la description comportementale d'un service, nous nous intéressons donc à la notion de processus défini par SDL puisqu'un service peut être considéré comme un ou plusieurs processus communicants.

4.4.1.1 Le comportement d'un processus SDL

La description du comportement d'un processus SDL est basée sur une machine à états finis (automate) étendue. Elle est constituée d'un ensemble d'états et de transitions connectant ces états. Un état de l'automate détermine quel comportement aura le processus quand il reçoit un message donné. Suite à la réception d'un message donné appelé message d'entrée, le processus réalise une action appelée aussi transition, chaque fois que le processus se trouve dans un certain état. La réalisation d'une transition fournit un ou plusieurs messages appelés messages de sortie et aboutit au passage du processus à un autre état, qui n'est pas nécessairement différent de l'état d'origine.

La notion "étendue" dans la définition de la machine à états finis vient de la possibilité d'ajouter des données variables. Elle permet de définir par exemple des décisions pour exécuter des transitions en se basant sur les valeurs d'une variable donnée. En effet, quand un

message d'entrée spécifique est reçu, l'état suivant ne dépend pas seulement de l'état courant et du message d'entrée. Une décision peut aussi utiliser des informations existant dans les messages pour déterminer la transition qui doit être faite.

4.4.1.2 Description

Le schéma 4.1 présente un exemple simple de la structure de la machine à états finis étendue décrivant un processus SDL.

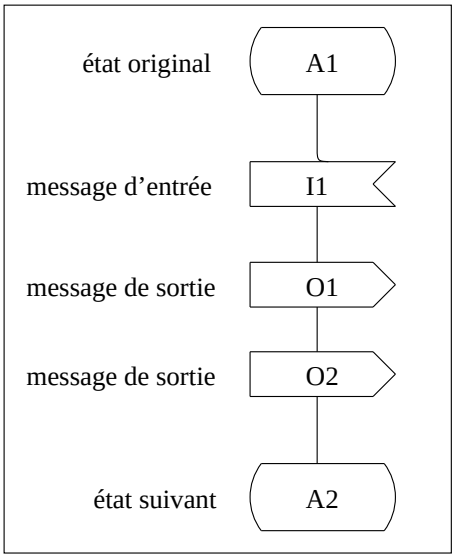


FIG. 4.1: Exemple de machine à états finis

Chaque état est défini par des messages d'entrée et des messages de sortie. Comme la figure indique, si le processus est dans l'état "A1" et il reçoit "I1" en entrée, alors il fait une transition vers l'état "A2" et fournit "O1" et "O2" en sortie. À son tour, l'état "A2" sera l'état courant du processus qui peut recevoir des entrées spécifiques, réalise des transitions vers d'autres états et fournit des sorties. Dans ce contexte, nous parlons d'un automate d'états finis déterministe où, pour un état donné, un état suivant unique est déterminé par la réception d'un seul message d'entrée.

Le comportement d'un processus est donc caractérisé par la production des messages de sortie en consommant des messages d'entrée et en se déplaçant d'un état à un autre. Ainsi, un automate peut être vu comme une fonction de messages d'entrée/sortie, c'est-à-dire qu'il peut fournir des sorties spécifiques à partir des entrées données en exécutant un ensemble de transitions. Cette notion inclut la description de chaque état de l'automate en fonction de ses messages d'entrée/sortie et puis les interactions entre ces états. Elle était notre source d'inspiration pour décrire le comportement de service que nous proposons dans une section prochaine de ce chapitre.

4.4.2 Automate d'interface : *Interface Automata*

Un autre formalisme qui permet de décrire le comportement d'un composant logiciel et ses interactions avec son environnement via un canal de communication est l'automate d'interface (*Interface Automata*) [111]. Comme son nom l'indique, il décrit le comportement d'un composant logiciel par un automate.

Nous avons choisi ce formalisme puisqu'il est utilisé dans le domaine des services télécoms. Il modélise la description des interfaces et CORBA s'intéresse bien à ces descriptions.

Un composant logiciel fonctionne en exécutant des actions ou les méthodes définies dans son interface selon les messages qu'il reçoit. Le principe de l'automate d'interface est d'utiliser un automate pour :

1. capturer les messages d'entrée qui indiquent les ordres par lesquels les méthodes d'un composant logiciel doivent être exécutées et les valeurs de retour du canal de communication.
2. capturer les messages de sortie qui déterminent les ordres par lesquels le composant doit appeler des méthodes externes (via le canal de communication) et les valeurs attendues par le composant.

Ce formalisme propose encore une approche pour la composition d'automates. Deux automates peuvent être composés si leurs actions sont disjointes sauf qu'une sortie de l'un peut coïncider avec une entrée de l'autre. Nous nous sommes intéressés par cette approche afin de composer des services dynamiquement sauf que la structure de l'automate dans notre approche de description comportementale d'un service est différente de celle définie par l'automate d'interface.

La composition de deux composants logiciels signifie la composition d'automates décrivant leurs comportements. D'une manière algorithmique, l'automate composé est construit en combinant les états compatibles des automates composants. Deux états sont compatibles si l'un a une entrée égale à une sortie de l'autre.

Nous allons présenter la description et la structure de l'automate d'interface et ensuite la composition de deux automates en donnant quelques exemples.

4.4.2.1 Description

Nous illustrons le fonctionnement et la structure de l'automate d'interface en prenant comme exemple, le comportement d'un composant logiciel qui implémente un service de

transmission de messages [111] (figure 4.2) (*Automate 1*).

Le composant a la méthode "msg" qui est utilisée pour envoyer des messages. Chaque fois que cette méthode est appelée, le composant retourne soit "ok", soit "fail". Pour exécuter ce service, le composant est lié par une interface au canal de communication qui fournit la méthode "send" pour émettre les messages. Les deux valeurs de retour possibles sont "ack" qui indique un succès de la transmission, et "nack" qui indique un échec de la transmission. Quand la méthode "msg" est invoquée, le composant essaye d'envoyer le message, et de le ré-envoyer si la première transmission échoue. Si les deux transmissions échouent, alors le composant retourne un échec ("fail") ; autrement, il retourne un succès ("ok").

Comme la figure indique, en étant dans un état quelconque, l'automate n'accepte que des entrées spécifiques. Par exemple, le message "msg" est accepté seulement si l'automate est dans l'état 0. En étant dans l'état 0, le composant va attendre comme sortie soit "ok" ou soit "fail" avant de faire un autre appel de "msg".

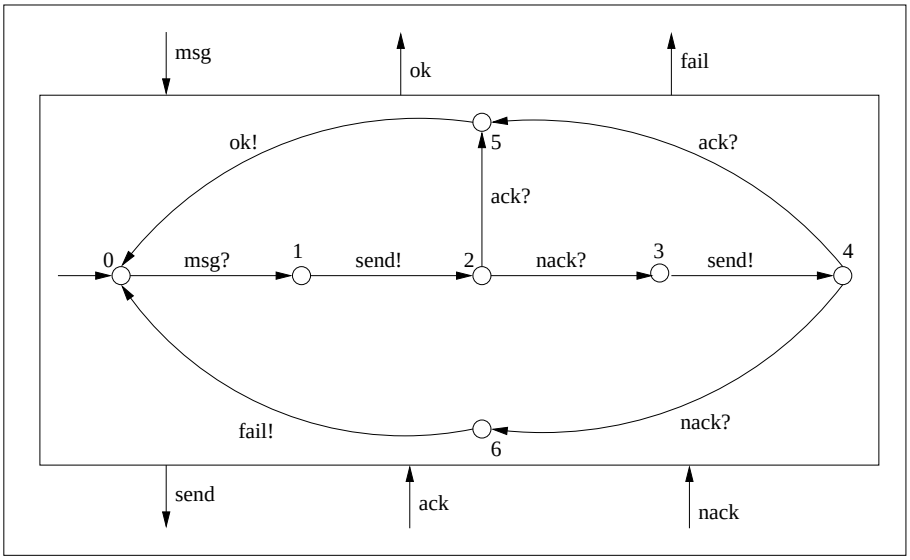


FIG. 4.2: Automate1 : un exemple d'automate d'interface

Les messages ("msg", "ack", "nack") sont considérés comme entrées puisqu'ils déterminent les méthodes du composant à appeler, leurs ordres d'invocation et le retour du canal de communication. Les messages ("send", "ok", "fail") sont considérés comme sorties puisque "send" représente la méthode externe à exécuter et ("ok","fail") définissent les valeurs de retour attendues par le composant.

À noter que les messages d'entrée sont désignés par le symbole ("?"), les messages de sortie par le symbole ("!") et les messages internes par (";"). Ce formalisme est donc très proche de celui utilisé par CSP [113]. Nous nous sommes intéressés seulement aux messages d'entrées/sorties pour décrire le comportement d'un composant logiciel.

Alors, nous pouvons décrire l'automate comme une boîte qui prend comme entrées ("msg", "ack", "nack") et comme sorties ("ok", "fail", "send").

4.4.2.2 Composition

Le but de la composition est de créer un composant qui a comme comportement la combinaison des comportements d'autres composants. Deux automates d'interface peuvent être composés si leurs actions sont disjointes et l'un a une entrée qui coïncide avec une sortie de l'autre.

Par définition, deux automates d'interfaces P et Q sont composables si :

$$A_P^H \cap A_Q = \emptyset \qquad A_P^I \cap A_Q^I = \emptyset$$
$$A_P^O \cap A_Q^O = \emptyset \qquad A_Q^H \cap A_P = \emptyset$$
$$A_P^I \cap A_Q^O \neq \emptyset \quad \text{or} \quad A_Q^I \cap A_P^O \neq \emptyset$$

Où A_P^O, A_P^I, A_P^H sont respectivement les ensembles de messages de sortie, de messages d'entrée et de messages internes de l'automate P. Comme nous remarquons, il reste qu'il y a une intersection entre l'ensemble d'entrées de l'un avec l'ensemble de sorties de l'autre.

Nous allons présenter la composition des deux automates d'interfaces dans l'exemple ci-après.

La figure 4.3 (*Automate3*) décrit un exemple d'un composant implémentant un service de transmission de messages. Ce composant prévoit que les messages seront émis avec succès et il ne traite pas les échecs de la transmission. Le composant est modélisé comme suit : après l'appel de méthode "msg", il accepte le retour "ok", mais il n'accepte pas le retour "fail". Ce composant est conçu pour être utilisé seulement dans les services de transmission de messages qui émettent les messages sans échec. Il prend comme sortie "msg" et comme entrée "ok".

La figure 4.4 (*Automate4*) représente la composition de *Automate1* avec l'automate décrit dans la figure 4.2 (*Automate2*). Les états d'Automate3 sont composés des états d'Automate1 et Automate3 qui ne présentent pas de conflits, c'est-à-dire un fonctionnement illégal. Les états qui ont les mêmes entrées/sorties sont combinés. En effet, l'état 0 d'Automate3 est constitué de la composition de l'état 0 de Automate1 et Automate2. Ces deux états sont composables puisqu'ils sont la même entrée/sortie ("msg").

Les étapes d'Automate3 sont les suivantes : une étape commune aux deux automates

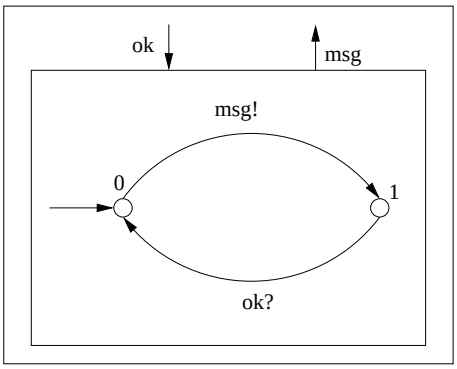


FIG. 4.3: Automate2 : un autre exemple d'automate d'interface

composants, qui est l'appel de "msg" pour envoyer le message, une étape aussi commune aux deux automates composants qui est la réception du message "ok" qui représente la terminaison de l'émission, des étapes de l'Automate2 qui sont un appel à la méthode "send" pour émettre le message ou une réception de retour "ack" ou "nack".

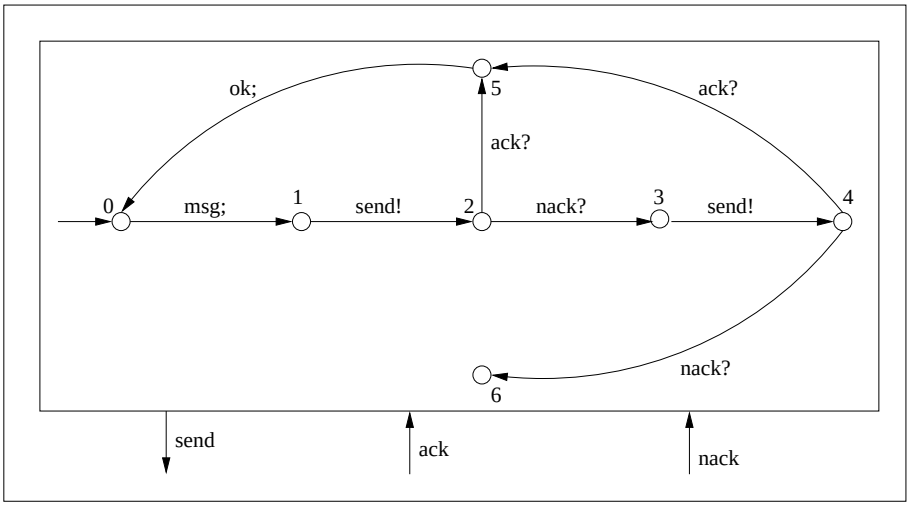


FIG. 4.4: Automate3 : la composition de Automate1 et Automate2

Considérons la séquence suivante des événements. Automate2 appelle la méthode "msg", puis Automate1 appelle deux fois la méthode "send" et reçoit deux fois la valeur de retour "nack", qui indique l'échec de la transmission. Cette séquence conduit vers l'état 6 de Automate3 qui correspond à l'état 1 d'Automate2 et l'état 6 d'Automate1. Dans l'état 6, Automate1 essaye d'envoyer le retour "fail", mais Automate2 n'accepte pas ce retour dans l'état 1. L'état 6 est considéré comme illégal.

Dans ce contexte, le formalisme interface d'automate considère que ces deux automates sont composables et compatibles en supprimant tous les états illégaux. La composition de ces deux automates, en se basant sur la formalisme *Interface Automata*, est indiquée par la

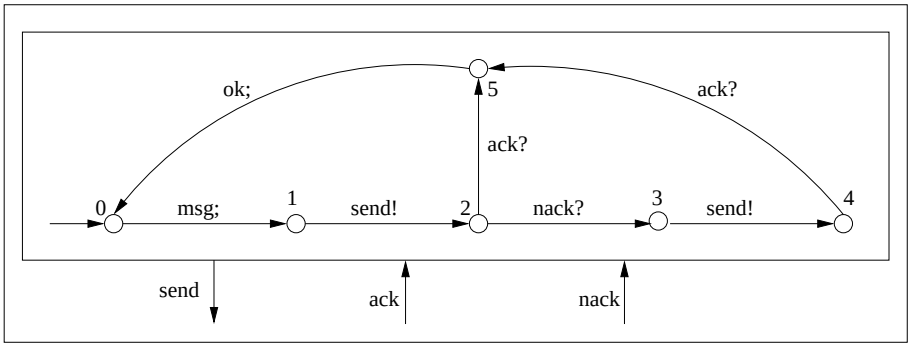


FIG. 4.5: Automate4 : la composition de Automate1 et Automate2 proposée par le formalisme interface d'automate

figure 4.5 (*Automate4*).

Alors, nous pouvons décrire Automate4 comme une boîte qui prend ("ack", "nack") en entrée et produit ("send") en sortie. Puisque les messages "ok" et "msg" sont communs aux deux automates, c'est-à-dire qu'ils sont définis comme messages d'entrée dans l'un et de sortie dans l'autre, alors ils ne sont pas pris en compte dans la description du comportement de l'automate composé.

Après cette présentation sur la description comportementale d'un composant logiciel par l'automate d'interface et SDL, nous allons présenter dans la section suivante notre approche de la description comportementale d'un service en utilisant les automates.

4.4.3 Notre approche

Le comportement d'un service définit les ordres corrects par lesquels les opérations du service doivent être exécutées. En se basant sur SDL et le formalisme de l'automate d'interface, les ordres de l'exécution des opérations d'un service dépendent de leurs entrées/sorties. Un service peut être considéré comme un composant logiciel. Par exemple, un client ne peut pas appeler un service bancaire pour déposer un montant sans qu'il ait ouvert un compte. Il doit exécuter une opération d'ouverture d'un compte bancaire avant qu'il n'exécute une opération de dépôt d'un montant.

Ainsi, le comportement d'un service permet aux clients d'utiliser et d'interagir avec le service en indiquant comment ils peuvent invoquer ses opérations. Il s'agit de savoir comment modéliser et décrire le comportement. En se basant sur SDL et l'automate d'interface, nous avons utilisé des automates étendus pour décrire le comportement. Nous allons présenter la structure et la construction de l'automate dans notre approche et puis sa description par des propriétés pour permettre aux clients de l'interroger.

4.4.3.1 Structure de l'automate étendu

Nous associons à chaque opération des entrées et des sorties qui indiquent ce qu'un service peut prendre en entrée et ce qu'il fournit comme résultats de son exécution. En d'autres termes, les entrées décrivent ce qu'un client doit avoir comme entrée pour pouvoir invoquer l'opération et les sorties décrivent les résultats. Par exemple, pour invoquer une opération qui permet de transformer un fichier "pdf" vers un fichier "ps", l'opération doit avoir une entrée indiquant aux clients qu'ils ont besoin en entrée d'un fichier "pdf".

Les états de l'automate représentent les opérations du service. Pour un état S, un état suivant est celui qui a une entrée égale à une sortie de l'état S. Une transition de l'état S à un autre état T est faite en exécutant l'opération représentée par l'état S et en fournissant une sortie qui doit être une entrée à l'état T.

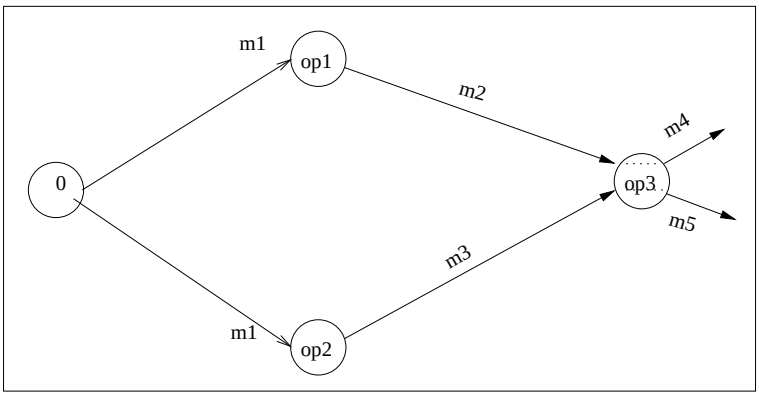


FIG. 4.6: Un exemple d'automate décrivant le comportement d'un service

La figure 4.6 donne l'exemple d'un automate qui décrit le comportement d'un service. Cet automate est composé de trois états qui représentent les opérations du service : "op1", "op2" et "op3". "op1" prend en entrée "m1" et fournit en sortie "m2". "op2" prend en entrée "m1" et donne en sortie "m3". "op3" prend une parmi deux entrées possibles : "m2" ou "m3". Pour une entrée "m3", "op3" fournit "m5" en sortie et pour une entrée "m3" il fournit "m4" en sortie.

Nous constatons que deux états sont liés s'ils ont une même entrée/sortie. Par exemple, "op1" est connecté à "op3" puisqu'ils ont une entrée/sortie commune "m2". En étant par exemple dans l'état "op1", la transition à l'état "op3" est faite en recevant une entrée "m1", en exécutant l'opération "op1" et en fournissant la sortie "m2" qui est une entrée à l'état "op3". Un client qui désire invoquer les opérations de ce service, doit suivre les ordres décrits par la figure 4.6. À chaque invocation, le client doit avoir la ou les entrées correspondantes.

Cet automate peut être implémenté du côté du client comme suit :

état_courant = 0 ;

while (...)

{

if symbole_lu = “m1” and état_courant = 0 {

switch(choix)

case 1 : appel méthode op1 ; symbole_lu = ”m2” ; état_courant := op3 ;

case 2 : appel méthode op2 ; symbole_lu = ”m3” ; état_courant := op3 ; }

elseif symbole_lu = “m2” et état_courant = op3 {

appel méthode op3 ; symbole_lu = ”m4” ; état_courant := final ; }

elseif symbole_lu = “m3”et état_courant = op3 {

appel méthode op3 ; symbole_lu = ”m5” ; état_courant := final ; }

}

Comme nous le remarquons, cet automate est utilisé par le client, qui peut donc déterminer les séquences d’appels possibles en fonction de son état courant. Cet automate qui décrit le comportement d’un service est fourni par le fournisseur du service et chaque client qui désire utiliser ce service doit l’invoquer en respectant les ordres d’interactions entre les opérations fournies par l’automate.

4.4.3.2 Description

Nous avons décrit la structure et l’architecture de l’automate décrivant le comportement d’un service et son implémentation. La question qui se pose maintenant est comment un client peut interroger un service par son comportement.

Nous pouvons décrire un service comme une boîte noire qui est une relation entre des entrées et des sorties. Dans l’exemple précédent, le service peut prendre en entrée "m1" et produire en sortie ("m2", "m3", "m4", "m5"). D’une manière plus précise, de l’entrée "m1", nous pouvons avoir les sorties ("m2" et "m3" et "m4" et "m5"), de l’entrée "m3", nous pouvons avoir la sortie "m5" et de "m2" nous pouvons avoir "m4". Alors pour chaque automate nous pouvons construire une table qui décrit pour chacune de ses entrées, les sorties possibles. La figure 4.7 présente la description de la table (boîte noire) associée à l’exemple précédent. Elle est composée de trois attributs "entrée", "sortie", et "type_service". Les clients peuvent utiliser ces attributs pour interroger le service par son comportement en envoyant une requête de type : je voudrais un service qui, à partir d’une entrée donnée, fournit une sortie précise. L’attribut "type_service" indique le type de service et le distingue des autres.

Après qu’un client découvre un service par ses entrées/sorties, il a besoin de savoir comment il pourra obtenir le résultat demandé. Cela signifie qu’il a besoin de savoir quels sont le

entrée	sortie	type_service
m1	m2	service
m1	m3	service
m1	m4	service
m1	m5	service

FIG. 4.7: La table décrivant l'automate comme une boîte noire

ou les chemins d'interactions des opérations qui fournissent les résultats. Par exemple, pour avoir la sortie "m4" à partir de l'entrée "m1", il faut suivre le chemin "op1" puis "op3". S'il y a plusieurs chemins, alors le client peut en choisir un qui lui convient en fonction de ses critères comme le coût, le chemin le plus court, etc.

Dans ce contexte, nous pouvons définir l'automate comme une boîte blanche qui indique les chemins d'interactions et les enchaînements des opérations d'un service. Cette boîte peut être décrite par une table qui a comme attributs : opération, entrée, sortie et le type de service. Pour chaque opération, ce qu'elle peut prendre en entrée et fournit en sortie. Puis en comparant les entrées avec les sorties, le client peut découvrir tous les chemins qui peuvent conduire d'une entrée donnée à une sortie demandée.

La figure présente la description de la table (boîte blanche) associée à l'exemple précédent.

opération	entrée	sortie	type_service
op1	m1	m2	service
op2	m1	m3	service
op3	m2	m4	service
op3	m3	m5	service

FIG. 4.8: La table décrivant l'automate comme une boîte blanche

Comme exemple, nous remarquons que "op1" a une entrée "m1" et une sortie "m2" et "op3" a une entrée "m2" et une sortie "m4". Comme "op1" et "op2" ont une même entrée/sortie "m2", alors nous constatons qu'à partir de l'entrée "m1", si nous invoquons "op1" puis "op2", alors nous aurons la sortie "m4".

4.4.4 Comparaison avec SDL et l'automate d'interface

Notre approche décrit le comportement d'un service par un automate étendu qui est compatible avec ceux définis par SDL et le formalisme de l'automate d'interface. En effet, nous avons pris la structure de l'automate tel qu'il est défini par SDL sauf que les actions représentent dans notre contexte les opérations du service. Dans notre approche, deux états

peuvent être connectés s'il ont une même entrée/sortie alors que dans SDL deux états peuvent être connectés sans avoir une même entrée/sortie.

L'automate d'interface décrit le comportement par un automate d'entrées/sorties où l'appel aux opérations est modélisé comme un message. Alors que dans notre approche, nous avons décrit les opérations comme des états de l'automate. Dans ce contexte, le client peut interroger un service en fonction de ses entrées/sorties et peut découvrir les opérations à invoquer et leurs ordres d'interactions pour atteindre sa demande (boîte blanche). Il y a une séparation entre les entrées/sorties et les opérations. Par contre, avec l'automate d'interface, si un client interroge un service en fonction de ses entrées/sorties alors il ne peut pas déterminer précisément les enchaînements des opérations qu'il peut suivre pour atteindre sa demande.

4.5 Conception et implémentation d'un trader

4.5.1 Architecture du trader à base d'ontologies

Comme pour le projet CANDLE, nous avons retenu une architecture à base d'ontologies. Ce choix est principalement lié aux travaux de notre équipe dans le domaine mais surtout aux potentialités, que nous avons évaluées initialement, de la combinaison de techniques à base de représentation de connaissances pour les systèmes distribués. L'ontologie nous sert à décrire le modèle de méta données et autres modèles de domaines, à exprimer des règles et garantir ainsi la cohérence des contenus. Enfin, l'interface permet aisément d'insérer des nouvelles entrées (c'est-à-dire une référence à un objet ou à un service et ses propriétés), et d'interroger.

4.5.2 Définition de la structure de l'ontologie d'un trader

La définition de la structure d'une ontologie se compose de deux parties :

- La définition des concepts, aussi appelés termes ou classes de l'ontologie. Ils correspondent à la définition des caractéristiques d'un domaine particulier décrit par l'ontologie. Chaque concept regroupe un ensemble de propriétés et peut contenir d'autres concepts.
- La définition des relations, qui traduisent les associations existant entre les concepts de l'ontologie. Ces relations incluent des associations comme : sous-classe-de (héritage), partie-de (composition), etc. Ces relations permettent de déterminer la structuration et l'interrelation des concepts, les uns par rapport aux autres.

Cette définition de la structure de l'ontolgie peut aussi s'accompagner de la définition de règles qui permettent d'inférer de nouvelles informations ou d'assurer la cohérene et l'inté-

grité du contenu. Ces règles s'expriment sous la fome d'un couple (pré-condition, conclusion) exprimé en logique du premier ordre : si la pré-condition est vraie alors la conclusion doit être vraie.

4.5.3 Indexation de services

L'indexation d'un nouveau service consiste à insérer une nouvelle information (un fait) dans l'ontologie. Cette information renseignera les différents champs et sections des méta données ainsi que l'adresse réseau qui permettra au client de contacter le service.

Cette opération d'indexation est réalisée par le service lorsqu'il s'enregistre auprès du trader ou lorsqu'il souhaite modifier les informations déjà stockées.

4.5.4 Recherche de services avec le trader

Pour interroger et rechercher des informations d'une ontlogie, nous avons besoin d'un langage lié à la logique du premier ordre [114]. Pour notre trader qui repose sur OntoBroker, nous utilisons le langage F-Logic [115] et nous donnons quelques exemples dans la section suivante.

4.5.5 Le trader en situation

La figure 4.9 décrit la structure du trader et ses interactions avec les autres entités (clients, serveurs, OntoBroker). Les étapes sont les suivants :

1. Un serveur d'application envoie au trader un fait rédigé en F-Logic qui décrit un service offert incluant une référence et les propriétés du service.
2. Le trader contacte OntoBroker pour ajouter ce fait à l'ontologie.
3. Le client qui désire un service interroge le trader en envoyant une requête en F-Logic et en spécifiant les propriétés désirées.
4. Le trader appelle OntoBroker pour chercher les services offerts demandés.
5. OntoBroker recherche les services demandés dans l'ontologie et il renvoie leurs références au trader.
6. Le trader transmet au client ces résultats, c'est-à-dire, les références.
7. Le client peut interagir avec le serveur indépendamment du trader en invoquant les services choisis.
8. Le serveur d'application répond à la demande du client.

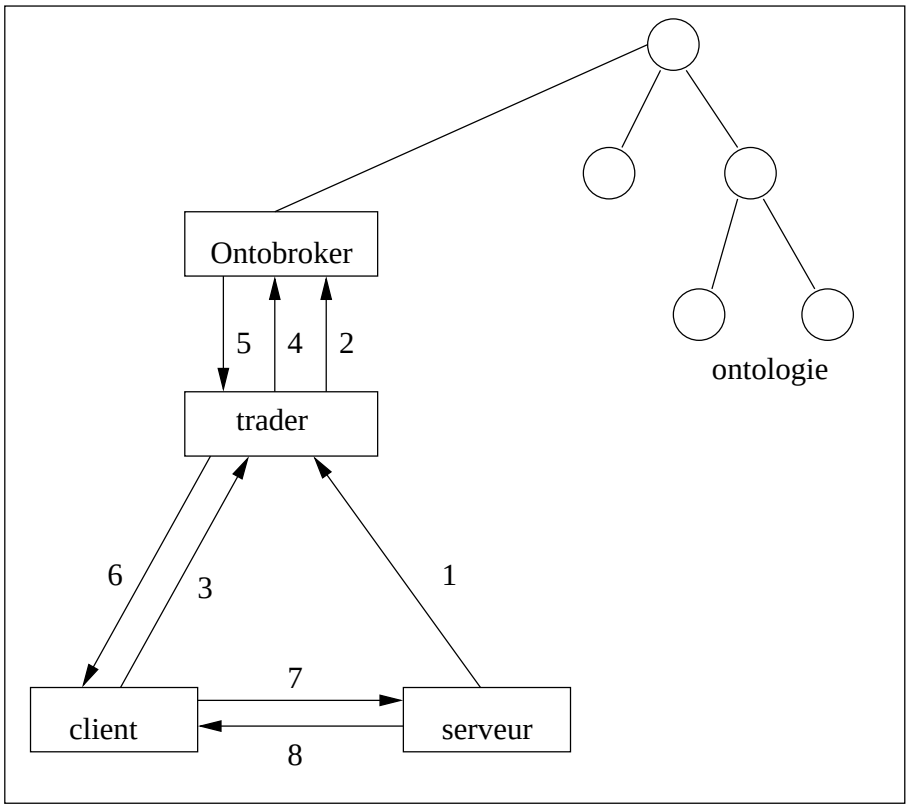


FIG. 4.9: Les interactions entre un client, un serveur, OntoBroker et l'ontologie

4.5.6 Un exemple de définition d'un trader et de son utilisation

Nous présentons dans la figure 4.10 un exemple d'une ontologie qui décrit les restaurants et les hôtels. Nous déclarons un concept "hôtel" qui décrit un hôtel. Nous attachons les propriétés suivantes à ce concept : un nom, le nombre de chambres, le nombre d'étoiles, l'adresse, etc. Nous déterminons un autre concept "restaurant" qui décrit un restaurant. Nous pouvons attacher les caractéristiques suivantes à ce concept : un nom, les spécialités, l'adresse, heure d'ouverture, etc.

Nous pouvons utiliser la relation d'héritage et définissons des sous-concepts qui fournissent des vocabulaires et des relations qui pourront être utilisés pour modéliser le domaine décrit par l'ontologie. Nous déclarons deux autres concepts "rest_sans_hôtel" (restaurant sans hôtel) et "rest_hôtel" (restaurant avec hôtel). Le premier hérite seulement de "hôtel" (il hérite de ses propriétés) et le deuxième hérite de "restaurant" et de "hôtel" (héritage multiple). Chacun de ces concepts a des propriétés spécifiques. Par exemple, pour un hôtel sans restaurant, il peut être utile de préciser la distance des restaurants voisins, qui sera alors un attribut spécifique.

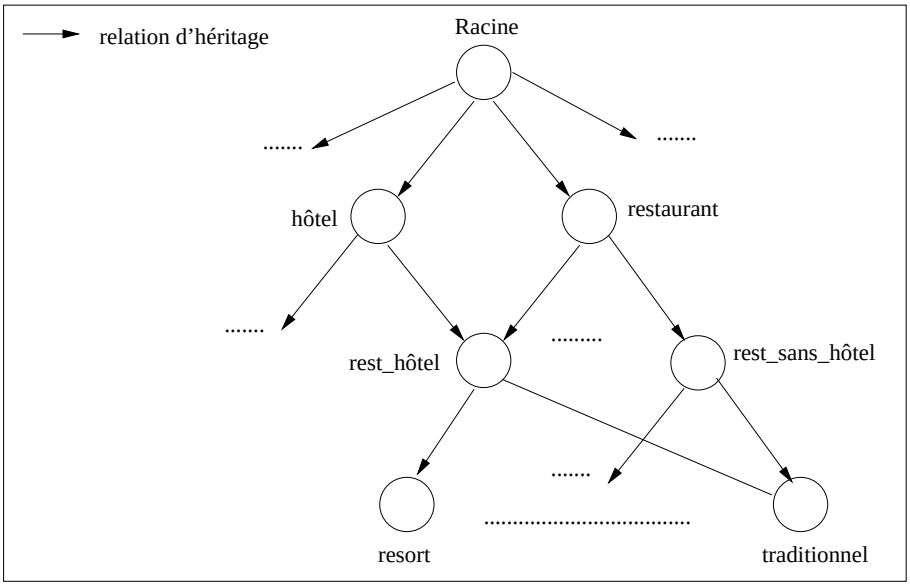


FIG. 4.10: Graphe d'un exemple d'une ontologie

Nous définissons un concept "traditionnel" qui hérite de "rest_hôtel" et "rest_sans_hôtel" et un autre concept "resort" qui hérite de "rest_hôtel". Chacun de ces concepts contient aussi des attributs spécifiques. Le graphe décrivant la structure de l'ontologie constituée de ces concepts est indiqué dans la figure 4.10.

Naturellement, chaque concept fait l'objet d'une définition et nous donnons un exemple. Le concept "restaurant" définit les attributs suivants : "nom", "spécialité" qui doivent être des chaînes de caractères (type *String*) et "adresse" qui doit être une instance du concept "Localisation". La définition en F-Logic est la suivante :

restaurant [

nom=>>String ;

spécialité=>>String ;

adresse=>>Localisation ;

...].

Nous pouvons ensuite définir des instances de ces concepts appelées aussi des faits. Par exemple :

chinois :restaurant [

nom->>"Shanghai" ;

spécialité->>"chinoise" ;

adresse->>ad :Localisation[nom_ville->> "Brest" ; code_postal->>29200 ;nom_rue-

>>"Siam"]

].

Cette expression signifie que "chinois" est une instance du concept "restaurant" qui a "Shanghai" comme "nom", "chinoise" comme "spécialité" et "ad" comme "adresse". "ad" est une instance du concept "Localisation" (voir section 3.3.1) qui a "Brest" comme nom de ville, "29200" comme code postal et "Siam" comme nom de rue.

Nous pouvons aussi définir des règles comme dans l'exemple suivant :

FORALL x,y x[spécialité->>"Française"] <- x :rest_hôtel and x[adresse->>y] and y[nom_ville->>"Brest"] and y[nom_rue->>"Siam"].

Cette règle signifie que pour chaque instance x de "rest_hôtel" ayant "Brest" comme nom de ville et "Siam" comme nom de rue, x offre comme spécialité "Française". Chaque instance du concept "rest_hôtel" ayant comme localisation "Brest" (la ville) et "Siam" (la rue), elle a par défaut la valeur "Française" de l'attribut "spécialité".

Une requête en F-Logic peut être considérée comme un type particulier d'une règle sauf que la tête d'une requête comporte seulement les variables permettant d'obtenir le résultat de la requête. Comme exemple :

Forall X <- X :restaurant and X[spécialité ->>"Française"].

FORALL X,K <- X :K and K : :restaurant.

La première requête retourne toutes les instances X de "restaurant" ayant "Française" comme valeur de l'attribut "spécialité". La deuxième requête retourne toutes les instances X de concepts (rest_hôtel et rest_sans_hôtel) qui héritent de "restaurant".

Par exemple, un client envoie une requête au trader pour découvrir tous les services offerts du "restaurant" :

FORALL x <- x :restaurant.

S'il y a de nombreuses offres, le trader indique au client qu'il y a un grand nombre de réponses et lui propose d'affiner sa requête. Il lui présente tous les attributs du concept "restaurant", ses super-concepts ("Racine") et ses sous-concepts ("rest_hôtel" et "rest_sans_hôtel"). Le client peut utiliser ces attributs et ces concepts pour reformuler sa requête pour rechercher

les services dans l'ontologie d'une manière plus précise. Par exemple :

FORALL x <- x :rest_hôtel and x[spécialité->>"Française"].

De la même façon, s'il y a un grand nombre de réponses, le trader présente au client les attributs de "rest_hôtel", ses super-concepts ("hôtel et restaurant") et ses sous-concepts (resort). Chaque fois, le client peut utiliser les concepts et les attributs retournés par le trader pour reformuler sa requête. Il y a ainsi une assistance et guidage de l'utilisateur qui est rendu possible grâce à l'ontologie. Ceci impose naturellement que la structuration de l'ontologie soit conçue de manière correcte et prenne en compte les requêtes possibles des utilisateurs.

D'autre part, supposons que le client demande un service et ce service n'existe pas. Par exemple :

FORALL x,y <- x :restaurant and x[adresse ->> y] and y[nom_ville->>"Brest"] and y[nom_rue->>"Siam"].

Si aucune offre ne répond aux critères du client, le trader retourne au client les sous-concepts ("rest_hôtel" et "rest_sans_hôtel") et les super-concepts ("Racine") du concept "restaurant". Il lui propose d'interroger ces concepts car ils permettent d'obtenir les services les plus proches du service demandé. Supposons qu'il y ait des restaurants avec hôtel qui offrent la spécialité chinoise et se localisent à Brest. En interrogeant le concept "rest_hôtel", le client peut obtenir le service demandé ou le service le plus proche du service demandé :

FORALL x,y <- x :rest_hôtel and x[adresse ->> y] and y[nom_ville->>"Brest"] and y[nom_rue->>"Siam"].

En se basant sur la règle définie précédemment, les hôtels retournés par cette requête offrent comme spécialité "Française". Tous les services offerts de type (rest_hôtel) ayant comme adresse (ville :Brest et rue :Siam), ont par défaut la valeur "Française" pour "spécialité".

Cet exemple présente les intérêts et la puissance de la structure de l'ontologie et surtout la relation d'héritage et la définition des règles dans la recherche de services. Notre trader permet la recherche de services d'une manière flexible et aide les clients à naviguer dans l'ontologie et à découvrir les services satisfaisant leurs exigences.

4.6 Vers un trader aux fonctionnalités étendues

Nous avons présenté dans la section précédente notre trader de base en utilisant les ontologies. Il permet la recherche de services d'une manière flexible, grâce à la structure de l'ontologie. Dans cette section, nous allons présenter une extension de ce trader pour adresser :

- La notion d'adaptation et de personnalisation de services selon les besoins du client ("profil utilisateur"). Il peut être décrit par un concept dans l'ontologie et pris en compte lors de la recherche de services.
- La description et l'interrogation du comportement du service et ainsi son interface en utilisant les ontologies.
- Quelques exemples d'interrogation.
- Une comparaison avec le trader standard.

4.6.1 Profil utilisateur

Pour adapter les services aux besoins de différents utilisateurs, un "profil utilisateur" est défini et utilisé. Le profil utilisateur "*est une source, une base de données sur un utilisateur*" [116]. Il contient des informations ou des propriétés qui caractérisent chaque utilisateur pour des tâches bien spécifiées. Il est utilisé dans le processus d'adaptation des services. Il peut contenir des caractéristiques principales sur l'utilisateur comme : des informations personnelles sur l'utilisateur (son âge, sa localisation, sa profession, etc.), ses préférences, ses expériences et ses connaissances dans le domaine de l'application, etc.
Les clients peuvent avoir des caractéristiques différentes. En adressant une même requête pour interroger des services via le trader, les clients, en fonction de leurs caractéristiques, peuvent attendre des résultats différents.

Les informations contenues dans le profil utilisateur peuvent être prises en compte lors de la recherche de services. Elles permettent de filtrer les services retournés pour chaque client et de sélectionner ceux qui correspondent à ses besoins. Ce phénomène s'appelle l'adaptation des services aux besoins de l'utilisateur.

Dans notre approche de trader, nous avons choisi d'intégrer le profil de l'utilisateur et de stocker cette description dans l'ontologie. Cette approche favorise une utilisation des profils et de leur intégration dans les requêtes initiales.

4.6.2 Description du comportement et de l'interface

Nous pouvons étendre l'ontologie que nous avons définie dans la section précédente pour prendre en compte la description du comportement du service et ainsi de son interface.

Nous avons décrit le comportement d'un service par un automate. Cet automate peut être considéré comme une boîte noire ou une fonction qui prend des entrées et fournit des sorties et comme une boîte blanche qui indique les chemins d'interactions entre les opérations du service pour fournir une sortie à partir d'une entrée donnée. Dans notre approche nous avons décrit les entrées/sorties par des chaînes de caractères, alors qu'elles peuvent être décrites par d'autres types. Dans ce contexte, nous pouvons définir deux concepts pour décrire le comportement de service comme une boîte blanche et noire :

```
Boîte_noire[
    type_service =>> Description_service ;
    entrée =>> STRING ;
    sortie=>> STRING
].
```

Ce concept contient un attribut indiquant le type de service et deux autres attributs pour décrire les entrées/sorties.

Le concept décrivant la boîte blanche peut être présenté comme suit :

```
Boîte_blanche[
    type_service =>> STRING ;
    opération =>> STRING ;
    entrée =>> STRING ;
    sortie =>> STRING
].
```

Ce concept contient des attributs concernant le type de service, le nom de l'opération et entrées/sorties.

Après avoir publié, les services qu'il offre en se basant sur leurs propriétés statiques, un fournisseur de services peut utiliser ces concepts pour décrire et publier les comportements de ses services. Pour chacun d'eux, il utilise le concept "Boîte_noire" pour indiquer ses entrées/sorties. Il utilise le concept "Boîte_blanche" pour décrire les entrées/sorties de chaque opération du service. Ce concept permet aux clients de construire l'automate implémentant le comportement du service. En effet, les états de l'automate représentent les opérations du service et chaque opération est liée à une autre s'ils ont les mêmes entrées/sorties. Cette description est nécessaire aux clients puisqu'elle leur permet de découvrir le comportement et le fonctionnement du service. Un client peut utiliser ces deux concepts pour interroger un service comme une relation d'entrées/sorties et pour découvrir les chemins d'interactions

entre les opérations du service qui permettent d'avoir une sortie à partir d'une entrée donnée. Dans une section suivante, nous présentons un exemple sur l'interrogation de ces deux concepts et comment un client peut découvrir les chemins disponibles dans l'automate.

La dernière étape de la description et l'indexation du service dans l'ontologie est celle de son interface. Nous avons retenu le langage IDL comme support de la description. Nous créons des concepts qui permettent de décrire les opérations de l'interface et leurs paramètres, les exceptions, etc. Ces informations permettent aux clients de découvrir l'interface du service et d'invoquer les opérations dynamiquement en utilisant par exemple l'interface DII de CORBA.

Nous définissons un concept "interface" pour décrire l'interface du service. Il peut être considéré comme un attribut du concept "Service" (défini dans la section précédente) puisque chaque service a une interface. Il est décrit comme suit :

```
interface[
    nom =>> STRING;
    opération =>> OpérationDéf;
    exceptions =>> ExceptionsDéf;
    attribute =>> AttributeDéf;
    typedef =>> TypedefDéf;
    constante =>> ConstantDéf
].
```

Ce concept contient la définition de tous les composants constituant une interface IDL. En effet, "OpérationsDéf", " ExceptionsDéf", "AttributeDéf", "TypedefDéf" et "ConstantDéf" sont les concepts qui permettent de décrire les composants de l'interface IDL. Le concept "OpérationDéf" permet d'indexer les opérations de l'interface IDL comme suit :

```
OpérationDéf[
    nom =>> STRING;
    nbr_paramètres =>> INTEGER;
    paramètre =>> ParamètreDéf;
    return_type =>> STRING;
    opération =>> OpérationDéf
].
```

"ParamètreDéf" est le concept permettant de définir la liste des paramètres de l'opération.

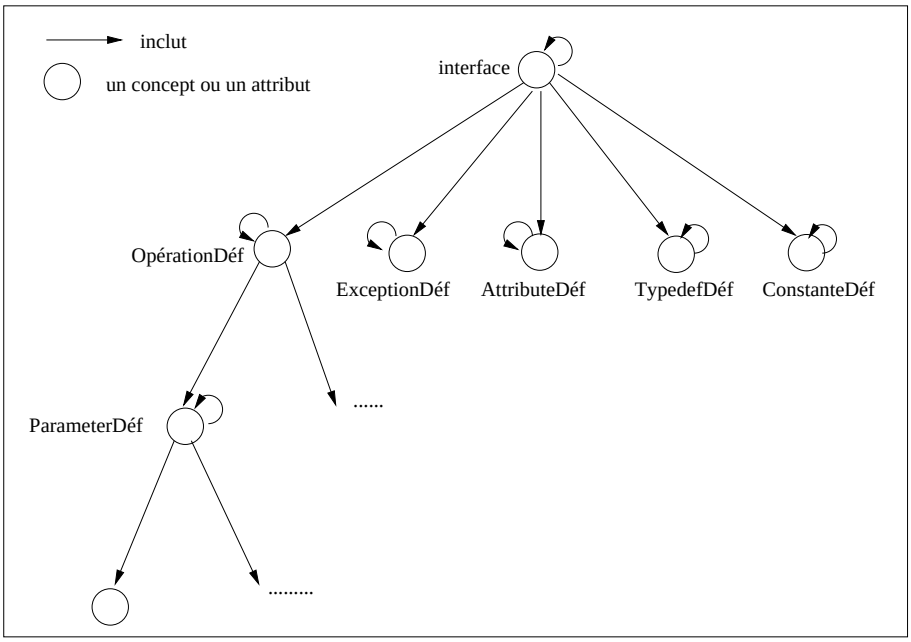


FIG. 4.11: La description de l'interface de service par une ontologie

Les attributs sont : le nom du paramètre, son type, sa position (le premier paramètre, le deuxième, etc.), son mode (entrée, sortie, entrée/sortie). Les paramètres d'une opération peuvent être définis par une liste. Dans ce contexte, nous pouvons définir un attribut "paramètre" de type "ParamètreDéf" pour indiquer la récursivité de la définition des paramètres de l'opération ou la liste des paramètres. De la même manière, nous pouvons définir les autres concepts concernant les exceptions, les attributs, les constantes et les typedefs. La description de l'interface dans l'ontologie est présentée dans la figure 4.11.

Les fournisseurs peuvent utiliser ces concepts pour décrire les interfaces de leurs services. Afin de simplifier le travail et au lieu que chaque fournisseur décrive les interfaces de ses services en créant les instances convenables, nous avons réalisé un traducteur qui permet de traduire du IDL vers F-Logic. Ce traducteur prend comme entrées un fichier IDL et fournit en sortie une ontologie contenant des instances décrivant les interfaces, les opérations, les exceptions, etc. contenues dans le fichier IDL. Un fournisseur de services, qui désire publier les interfaces de ses services offerts, utilise ce traducteur en indiquant le fichier IDL contenant les définitions des interfaces. Un client qui ne connaît pas l'interface du service et désire invoquer un service, peut interroger l'ontologie et récupérer toutes les informations nécessaires qui lui permettent de construire une requête d'invocation dynamique en utilisant par exemple, l'interface DII de CORBA.

Nous avons présenté dans cette section l'ontologie qui permet d'indexer et d'enregistrer

les caractéristiques d'un service. Elle contient des concepts décrivant les propriétés statiques, le comportement et l'interface d'un service. Un fournisseur de services peut utiliser ces concepts pour publier ses services offerts qui sont des faits ou des instances de ces concepts. Un client peut découvrir des services offerts en interrogeant les différents concepts de l'ontologie.

4.6.3 Exemples

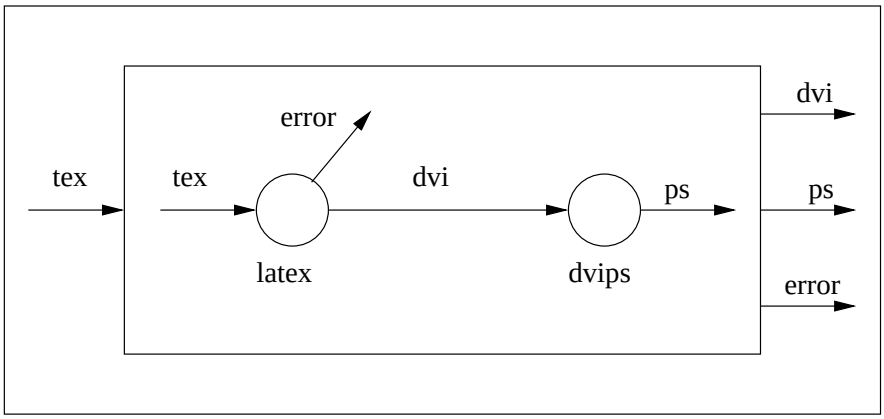


FIG. 4.12: Un exemple du comportement d'un service de transformation des fichiers

Nous prenons comme exemple un service de transformation des fichiers (le type de ce service est "file_transf") [117]. Ce service a des propriétés statiques et un comportement (figure 4.12). Comme indiqué dans la figure, ce service a deux opérations "latex" et "dvips" et permet de transformer un fichier de type "tex" en un fichier de type "ps". L'automate décrivant le comportement a deux états qui correspondent aux opérations. L'état "latex" prend en entrée un "fichier" de type "tex" et fournit en sortie soit un fichier de type "dvi" soit une erreur "error". Le deuxième état "dvips" prend en entrée un fichier de type "dvi" et fournit en sortie un fichier de type "ps". "latex" et "dvips" sont liés car ils ont la même entrée/sortie ("dvi").

Comme une boîte noire, cet automate prend en entrées "tex" et produit en sortie ("dvi", "ps", "error"). Comme une boîte blanche, les chemins d'interactions entre les opérations sont présentés dans la figure 4.12.

4.6.3.1 Exemples d'interrogation utilisés par la fonction d'importation

Un bénéfice principal que notre trader a apporté aux clients est d'avoir la possibilité de découvrir un service en interrogeant ses trois niveaux de description par les même interfaces.

Nous donnons quelques exemples sur les requêtes utilisées par la fonction d'importation pour interroger un service de l'ontologie en utilisant F-Logic.

Le premier exemple est une requête simple qui permet d'interroger le service ("file_transf") en indiquant les propriétés statiques du service demandé :

FORALL x <- x :Service and x[nom_fournisseur->>"France_Telecom"] and x[type_service->>"file_transf"].

D'une manière ensembliste, cette requête peut être exprimée sous la forme :

{x :service/ x.nom_fournisseur='France_telecom' and x.type_service=' file_transf'}

Les résultats de cette requête sont toutes les instances "x" du concept "Service" ayant "France_telecom" comme nom du fournisseur de service et "file_transf" comme type de service.

Nous pouvons étendre la requête pour interroger le comportement du service ayant "file_transf" comme type de service. Nous pouvons interroger le service comme une relation entre des entrées et sorties (le concept *Boîte_noire* (voir l'ontologie définie dans la section précédente)). Nous indiquons les entrées données et les sorties désirées comme des critères de recherche dans la requête.

FORALL x <- EXISTS y x :Service and x[nom_fournisseur->>" France_telecom "] and x[type_service->>"file_transf"] and y :Boîte_noire and y[entrée->>"tex"] and y[sortie->>"ps"] and y[type_service->>"file_transf"].

Cette requête retourne toutes les instances "x" du concept "Service" ayant "France_telecom" comme nom du fournisseur de service, "file_transf" comme type de service et prend en entrée un fichier de type "tex" et produit en sortie un fichier de type "ps" qui représente la transformation du fichier d'entrée "tex" en format "ps".

Nous pouvons interroger le concept *Boîte_blanche* pour découvrir de proche en proche les séquences des opérations que le client doit utiliser pour atteindre le résultat.

FORALL x,y,z,k <- x :Boîte_blanche and x[entrée->>"tex"] and x[sortie->>k] and x[type_service ->>"file_transf"] and x[y->>z].

Si le client reçoit "ps" comme sortie (valeur de "k"), il connaît l'opération (une seule opération à invoquer) qui lui permet d'obtenir le résultat (valeur obtenue par "y" et "z").

Sinon, il étend sa requête en comparant les sorties avec les entrées de deux opérations possibles :

FORALL x,y,m,n,l,k,i,j <- x :Boîte_blanche and y :Boîte_blanche and x[entrée->>"tex"] and x[sortie->>k] and y[entrée->>k] and y[sortie->>l] and x[type_service ->>"file_transf"] and y[type_service ->>"file_transf"] and x[m->>n] and y[i->>j].

Si le client reçoit "ps" comme une sortie (valeur de "l"), le client connaît les opérations qui lui permettent d'obtenir le résultat (deux opérations à invoquer) : une opération indiquée par "x" (valeur obtenue par "m" et "n") et une autre par "y" (valeur obtenue par "i" et "j"). Sinon, le client essaye d'obtenir le résultat en faisant des requêtes récursivement qui impliquent des étapes additionnelles : il fait des requêtes avec 3, 4, etc. étapes (opérations) jusqu'à l'obtention de la sortie "ps". Puis, il peut invoquer les opérations dans l'ordre retourné par les étapes d'interrogation. Ceci permet au client d'obtenir les chemins des interactions entre les opérations implémentées dans l'automate et qui permettent de produire la sortie "ps" à partir de l'entrée "tex".

Le client peut aussi interroger l'interface du service pour invoquer ses opérations dynamiquement (le concept "interface"). Il peut étendre sa requête pour découvrir les services désirés et les opérations fournies par ces services :

FORALL x,y,z,k,l <- x :Service and x[nom_fournisseur->>"France_telecom "] and x[type_service->>"file_transf"] and x[interface->>y] and y[opération->>z] and z[k->>l].

Cette requête retourne la première opération de la liste des opérations de l'interface du service. "z" désigne ainsi une opération de "y", définie comme interface de "x". Le client peut découvrir les autres opérations en traversant la liste. Pour chaque opération, le client peut étendre sa requête pour découvrir la liste de ses paramètres. Par exemple :

FORALL x,y,z,p,m,n <- x :Service and x[nom_fournisseur->>"France_telecom "] and x[type_service->>"file_transf"] and x[interface->>y] and y[opération->>z] and z[paramètre->>p] and p[m->>n].

Cette requête permet de retourner la liste des paramètres de la première opération contenue dans la liste des opérations du service.

Le lecteur intéressé pourra obtenir des informations plus précises concernant la description et l'interrogation de l'interface du service dans [118].

4.7 Composition de services

4.7.1 Composer de nouveaux services

La communication et la collaboration entre les fournisseurs de services deviennent une nécessité pour répondre aux besoins des clients. Chaque service fournit une fonctionnalité qui peut être indépendante d'une autre. Plusieurs services peuvent être combinés et assemblés en un seul service ayant comme fonctionnalité (comportement) la composition de celles de ses services composants. L'idée est donc de créer des nouveaux services en combinant des services existants et augmenter ainsi l'environnement des utilisateurs et l'ensemble des services disponibles.

La composition de services permet donc d'enrichir le modèle des services existants en créant des nouveaux services. En effet, si un client demande un service en interrogeant son comportement et si ce service n'existe pas, alors une combinaison des fonctionnalités des deux ou plusieurs services pourrait, peut-être, répondre à la demande du client. Dans ce contexte, la composition donne une valeur ajoutée à la recherche de services en permettant aux clients de bénéficier des services supplémentaires où chaque service peut ne pas être fourni par un seul fournisseur mais par une collaboration entre deux ou plusieurs fournisseurs.

Les types de composition de services [119] les plus répondus sont : la composition statique et dynamique. Dans la composition statique, les services à composer sont sélectionnés lors de la compilation (lors du déploiement des services). Dans la composition dynamique, les services à composer sont précisés à l'exécution (lors de l'exécution des services). Un autre type de composition appelé "semi-statique" ou "semi-dynamique" [119] sera présenté dans la suite de ce document.

En se basant sur la description comportementale d'un service par un automate, nous allons présenter dans cette section notre approche pour la composition statique, semi-dynamique et dynamique de services. La composition de services nécessite la composition d'automates décrivant leurs comportements. Nous allons décrire les techniques que nous avons utilisées pour implémenter la composition d'automates d'une manière statique et dynamique en donnant quelques exemples.

4.7.2 La composition statique

La composition statique de services permet de créer de nouveaux services à partir des services existant à la compilation lors du déploiement des services. Nous parlons de la composition au niveau service offert, c'est-à-dire, la composition de deux ou plusieurs services offerts. En d'autres termes, un service composé peut être considéré comme un service offert qui est constitué de deux ou plusieurs services offerts pouvant être fournis par

différents fournisseurs.

Pour construire un service composé, un plan qui définit les services offerts composants et l'ordre de leur enchaînement doit être déterminé lors de la définition du service composé. L'ordre de l'enchaînement des services dépend des entrées/sorties des automates décrivant leurs comportements. En se basant sur l'interface d'automate, deux automates peuvent être composés s'ils ont une ou plusieurs entrées/sorties en commun. Cela signifie que la composition de services prend en compte plus particulièrement la description d'un service par une boîte noire puisqu'elle décrit les entrées/sorties de chaque service.

En se basant sur la description de service par une boîte noire, nous pouvons déterminer les services qui peuvent être composés en comparant leurs entrées/sorties. Le service composé peut être décrit par une boîte noire qui est la composition de boîtes noires des services composants.

Pour assurer une certaine cohérence de l'ensemble, les services offerts qui sont composés doivent être filtrés : on utilise pour cela leurs propriétés statiques et les méta données. Sans cela, il y a un risque d'assembler tout et n'importe quoi.

Le service composé peut aussi être décrit par une boîte blanche qui est la composition de boîtes blanches des services composants et qui correspond à la description de sa boîte noire. En d'autres termes, la boîte blanche décrit les chemins d'interactions entre les opérations qui permettent d'avoir les entrées/sorties indiquées dans la boîte noire.

Dans notre approche de la recherche et découverte de services en utilisant les ontologies, le service composé peut être ajouté à l'ontologie décrivant les services. Ce service est décrit par son comportement (boîte blanche et boîte noire). Il est décrit par un connecteur qui peut être considéré comme son fournisseur. Le connecteur est un serveur qui décrit l'automate illustrant le comportement du service qui est composé des automates des services composants. La communication avec ce service se fait via le connecteur qui à son tour communique avec les fournisseurs de services composants pour exécuter les services.

Un client peut interroger le service composé par son comportement et invoquer ses opérations. Pour invoquer une opération du service composé, le connecteur dirige la requête vers le service composant qui implémente cette opération pour l'exécuter et puis il retourne le résultat au client.

4.7.2.1 Exemple

Nous présentons un exemple de deux services offerts que nous désirons composer dans un seul service en utilisant les automates. Le premier est un service offert de type "file_transf" que nous avons présenté dans la section 3.6 (figure 4.12). Il permet de transformer un fichier

de type "tex" en un fichier de type "ps".

Le deuxième est un service offert de type "ps_pdf" qui permet de transformer un fichier de type "ps" en un fichier de type "pdf". L'automate décrivant le comportement de ce service est présenté dans la figure 4.13. Ce service a une seule opération "pstopdf" qui prend en entrée un fichier de type "ps" et fournit en sortie un fichier de type "pdf".

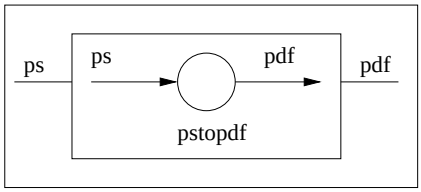


FIG. 4.13: Automate associé au service de transformation de fichier "ps" en "pdf"

Si un client a un fichier "tex" et désire le transformer en un fichier "pdf", alors il peut interroger le trader pour trouver un service ayant comme entrée "tex" et comme sortie "pdf". Aucun de ces deux services ("file_tansf" et "pstopdf") ne peut répondre à la demande du client. Mais si le client invoque le premier service en donnant "tex" comme entrée, alors il pourra avoir la sortie "ps". S'il prend cette sortie comme entrée et invoque le deuxième service, alors il pourra avoir la sortie "pdf".

Donc, si ces deux services sont composés dans un seul service et que ce service est enregistré dans l'ontologie, alors ce service peut répondre à la demande du client pour transformer le flux d'entrée "tex" en "ps". Dans ce type de problème, la composition de services devient une nécessité pour les clients.

Ces deux services peuvent être composés puisqu'ils ont une même entrée/sortie ("ps"). L'automate décrivant le service composé est présenté dans la figure 4.14. Il est constitué des états des automates composants, tel que un état d'un automate composant est connecté à un état d'un autre s'ils ont une même entrée/sortie. Donc, l'état "dvips" du premier automate est lié à l'état "pstopdf" du deuxième automate par l'entrée/sortie "ps".

Cet automate peut être implémenté par le connecteur et les clients peuvent communiquer avec le service en invoquant les opérations existantes dans l'automate et en respectant l'ordre de leur enchaînement. Par exemple si un client désire invoquer l'opération "latex", alors le connecteur appelle le fournisseur du service offert qui implémente cette opération pour l'exécuter. Le fournisseur exécute l'opération et retourne le résultat au connecteur qui à son tour l'envoie au client. La communication entre le connecteur et les fournisseurs de services est implicite pour le client.

Comme une boîte noire, le service composé prend en entrée "tex" et produit en sortie ("dvi","ps","error","pdf"). Comme une boîte blanche, les chemins d'interactions entre les opé-

rations sont présentés dans la figure 4.14.

Ce service peut être enregistré dans l'ontologie et considéré comme un service normal en indiquant son comportement (boîte blanche et noire). Les clients peuvent découvrir ce service via le trader en interrogeant son comportement.

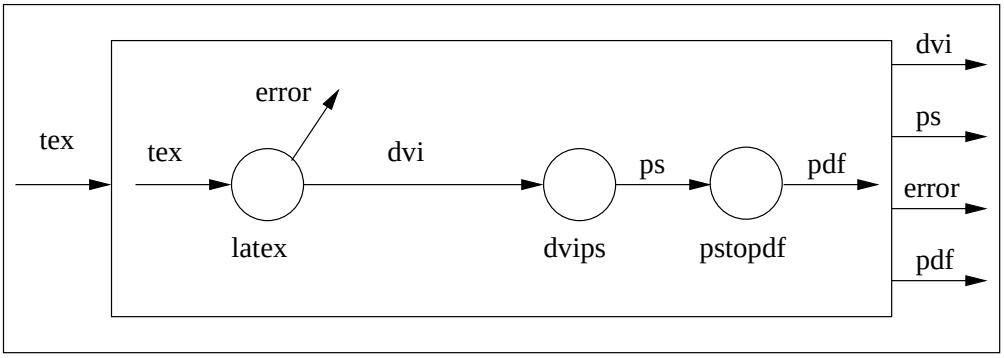


FIG. 4.14: Automate associé au service composé

4.7.3 Composition dynamique de services

La composition dynamique permet de générer des services composés à partir des services existants à l'exécution. Les services composés sont définis au fur et à mesure en se basant sur la demande du client.

Par exemple, le client peut demander un service en fonction des ses entrées/sorties en interrogeant son comportement (*Boîte_noire*) via le trader. Le trader cherche le service demandé dans l'ontologie et si ce service n'existe pas, alors il essaye de combiner des services pour répondre à la demande du client. L'idée est donc de trouver un service qui admet l'entrée demandée et un autre qui admet la sortie demandée et d'essayer de les connecter s'ils ont une même entrée/sortie (deux services à composer). Sinon, il s'agit d'étendre la composition pour adresser trois, quatre, ou plus, services à composer (d'une manière récursive) en connectant leurs entrées/sorties, pour répondre à la demande du client. Pour cela, le trader prend l'entrée donnée par le client et il cherche les services ayant cette entrée. Il prend les sorties fournies par ces services et il les compare avec les entrées des autres services. Si ces autres services fournissent la sortie demandée, alors le trader retourne au client les plans possibles des services à composer permettant de répondre à la demande du client. Chaque plan contient deux services. Sinon, le trader étend la requête pour adresser d'autres services en comparant leurs entrées/sorties et retourne au client les plans possibles des services à composer s'ils existent.

Chaque plan peut avoir un coût (*score*) et le client peut choisir le plan qui a le meilleur coût. En effet, nous pouvons associer à chaque service un coût. Le coût d'un plan et la somme des coûts des services qui le constituent.

Dans ce type de composition, nous pouvons adresser soit la composition au niveau service offert, soit la composition au niveau type de service (selon les besoins du client). Les services composés sont donc déterminés au fur et à mesure et ne sont pas enregistrés dans la base (par exemple, l'ontologie). Les services composés sont définis pour répondre à la demande et aux besoins de chaque client lors de l'exécution.

Chaque plan contient l'ordre de l'enchaînement des services et les entrées/sorties qui les lient. Le client peut exécuter le plan des services en commençant par le premier. Il indique l'entrée qui est l'entrée initiale, il invoque le premier service et récupère la sortie mentionnée dans le plan. Il prend cette sortie comme entrée, appelle le deuxième service et récupère la sortie et ainsi de suite jusqu'à la fin de l'exécution du plan.

Pour invoquer un service, le client doit déterminer l'entrée/sortie demandée dans le plan. Il peut interroger le trader (le concept *Boîte_blanche*) pour obtenir les chemins d'interactions entre les opérations permettant de fournir la sortie demandée à partir de l'entrée donnée.

La composition dynamique comporte la possibilité qu'un client peut appeler un service défini à l'exécution et de le composer avec d'autres services. Cela signifie que le client peut ne pas connaître l'interface du service à la compilation mais il doit pouvoir la découvrir à l'exécution pour invoquer ses opérations.

Dans ce contexte, le client doit invoquer dynamiquement les services en utilisant par exemple l'interface d'appel dynamique (DII) de CORBA. Nous avons utilisé l'approche que nous avons développée dans le chapitre précédent sur la description du service par son interface et de l'invocation dynamique de service. En effet, un fournisseur de service qui désire publier ses services en les décrivant par des propriétés statiques et dynamiques, doit aussi publier leurs interfaces. Il peut décrire l'interface de chaque service par ses opérations et leurs paramètres, ses exceptions, etc. (voir 3.6.1.1) et les enregistrer dans l'ontologie via le trader.

Un client qui désire utiliser et composer ces services avec d'autres services peut récupérer leurs descriptions d'interfaces via l'ontologie pour construire des requêtes d'invocation dynamique en utilisant DII.

Dans ce type de composition, si une interface d'un service est mise à jour par exemple en modifiant ou en ajoutant des opérations, alors le client n'a pas besoin d'être mis à jour ou recompilé. La modification de l'interface n'a aucun effet sur le client.

Ainsi, en fournissant la possibilité de composer des services définis à l'exécution et la possibilité de modifier les interfaces des services d'une manière implicite par rapport au client, alors la composition est devenue dynamique au sens large pour le client. Il reste cependant que cette composition pose une question de sécurité du fait de l'assemblage à l'exécution et surtout est coûteuse en temps et réduit donc la performance.

4.7.4 Composer à partir de spécifications de services

La composition semi-dynamique permet de créer des nouveaux types de services à partir des types de services existants lors de la compilation. En effet, nous considérons que tous les services offerts qui sont des instances d'un même type de service ont le même type d'interface et par conséquent le même comportement. Donc, chaque service ayant comme type de service un type composé de deux ou plusieurs autre types peut être considéré comme un service composé. À noter que les services composés sont toujours décrits par leurs comportements et ne peuvent pas être décrits par des propriétés statiques.

L'intérêt de la composition semi-dynamique est qu'elle permet aux clients de ne pas être limités à des services offerts précis mais ils utilisent des types de services. Cela leur permet donc de préciser les propriétés attendues des services à composer. Par exemple, dans le contexte de la mobilité, si un client compose un service offert de type "météo" avec un autre service offert de type "cinéma" et si l'environnement change, alors cette composition s'adapte à ce changement. Le client peut toujours composer des services offerts de types "météo" et "cinéma" dans le nouvel environnement en indiquant les propriétés désirées.

Pour construire un type de service composé, un plan qui détermine les types de services composants doit être déterminé lors de la définition du type de service composé. Deux types de services peuvent être composés si les automates décrivant leurs comportements ont une ou plusieurs entrées/sorties en commun. La composition de types de services signifie la composition d'automates associés à leurs comportements.

Le type de service composé peut être décrit par une boîte noire et une boîte blanche et enregistré dans l'ontologie comme un type de service normal. L'automate décrivant le comportement du type de service composé est implémenté par un connecteur qui peut être considéré comme le fournisseur de tous les services offerts de ce type de service.

La communication avec ce type de service se fait par l'intermédiaire de connecteur. En effet, les clients peuvent découvrir ce type service en interrogeant son comportement. Une fois qu'un client découvre ce service, il peut communiquer avec le connecteur pour l'invoquer. Le connecteur indique au client que le service demandé a un type composé. Il lui retourne les types de services composants et lui propose d'interroger le trader pour choisir les services offerts de ces types de services qui répondent à ses exigences en se basant sur leurs propriétés statiques. Le client choisit les services offerts convenables et il les retourne vers le connecteur. À son tour, le connecteur peut interagir avec les fournisseurs de ces services pour exécuter les opérations demandées et retourne le résultat au client.

Les clients ne sont pas limités à des services offerts précis comme dans le cas de la composition statique, mais ils peuvent interroger les services offerts qui correspondent aux types de

services composants en se basant sur les propriétés statiques. Ainsi, la composition devient de plus en plus élaborée et riche.

Nous prenons l'exemple présenté dans la section précédente pour montrer comment nous pouvons le faire en utilisant la composition semi-dynamique. En effet, nous supposons que les deux automates présentés par les figures 4.12 et 4.13 décrit les comportements des services offerts ayant "file_transf" et "ps_pdf" comme types de services respectivement. L'automate décrivant le type de service composé est présenté dans la figure 4.14. Ce type de service est enregistré dans l'ontologie en indiquant son comportement (boîte noire et boîte blanche). Il peut être exprimé comme suit : "file_transf"+"ps_pdf".

Un client peut interroger le trader pour trouver un service permettant de transformer un fichier de type "tex" en un fichier de type "pdf". Le trader peut retourner au client le service ayant comme type ("file_transf"+"ps_pdf") qui répond à sa demande. Le client connaît que ce service a un type composé de "file_transf" et "ps_pdf". Il appelle son fournisseur qui est le connecteur pour invoquer les opérations de ce service en respectant leurs ordres d'enchaînements décrits par l'automate (boîte blanche).

Supposons par exemple que le client désire invoquer l'opération "latex", alors le connecteur en fonction de l'opération demandée, informe le client par le type de service qui l'implémente ("file_transf") et propose au client d'interroger le trader pour trouver un service offert ayant comme type de service "file_transf" pour exécuter l'opération. En se basant sur les propriétés statiques, le client peut interroger le trader pour trouver les services offerts satisfaisant ses exigences et qui a comme type "file_transf". Il choisit un service offert et il l'indique au connecteur en lui envoyant son adresse. Le connecteur peut interagir avec le fournisseur du service offert choisi pour invoquer l'opération "latex". Après l'invocation, le connecteur retourne le résultat au client qui peut suivre ses appels pour invoquer d'autres opérations.

La communication entre le connecteur et les fournisseurs de services est toujours implicite aux clients.

Dans ce type de composition, les clients peuvent découvrir les services composés en interrogeant leurs comportements et peuvent choisir les services composants en interrogeant leurs propriétés statiques. Cela leur permet de ne pas être limités aux services offerts composants comme dans le cas de la composition statique mais de sélectionner les services offerts qu'ils souhaitent. Ce phénomène était la base du nommage de ce type de composition par "semi-dynamique" : "dynamique" puisque les clients peuvent interroger et choisir les services offerts et "semi" signifie que la composition n'est pas dynamique puisque les types de services composants sont définis d'une manière statique, c'est-à-dire, à la compilation lors du déploiement du service composé.

4.8 Synthèse générale

Dans ce chapitre, nous avons décrit l'approche que nous avons utilisée pour indexer et enregistrer les caractéristiques d'un service, c'est à partir de techniques de représentation de connaissances et d'ontologies. Les propriétés statiques, le comportement et l'interface d'un service pouvent être décrits par des concepts. Nous avons présenté la structure de l'ontologie et les techniques utilisées comme la définition des règles, la relation d'héritage entre les concepts, etc.

Pour interroger les ontologies, nous avons besoin d'un langage logique. Nous avons présenté un langage que nous avons utilisé, c'est et nous avons retenu *F-Logic*. Il permet aux clients de bénéficier de la puissance de la logique de premier ordre pour interroger et découvrir un service et permet ainsi d'introduire la flexibilité dans les requêtes.

Nous avons décrit notre approche du trader basé sur les ontologies. Il fournit les mêmes outils pour interroger les trois niveaux de description d'un service. Nous avons présenté la puissance de notre trader par rapport au trader standard proposé par ODP et OMG CORBA. Le trader standard est basé sur les bases de données où le schéma est statique. Le client utilise un langage booléen ou SQL pour interroger les services via le trader. En utilisant les ontologies et un langage logique, notre approche de trader permet aux clients d'interroger et de découvrir les services d'une manière plus flexible que le trader standard.

Nous avons présenté aussi notre approche du profil utilisateur qui peut être utilisée pour adapter les services selon les besoins du client. En effet, chaque client peut définir son profil et l'enregistrer dans l'ontologie via le trader. Si un client interroge un service et s'il y a beaucoup de réponses, alors le trader utilise le profil du client pour étendre la requête. Cela lui permet d'avoir les services les plus appropriés satisfaisant les exigences du client.

Finalement, nous avons décrit notre approche pour la composition de services. Nous avons présenté trois types de composition : statique, semi-dynamique et dynamique. La composition statique permet de composer des services offerts lors de la compilation et d'enregistrer les services composés comme des services normaux dans l'ontologie. La composition semi-dynamique est similaire sauf que la composition se fait au niveau types de services. Les clients ont la possibilité d'interroger et de choisir les services offerts à composer qui correspondent aux types de services composés. La composition dynamique est le type le plus puissant puisqu'il permet aux clients de composer les services qu'ils souhaitent et de les invoquer lors de l'exécution.

4.9 Perspectives et questions de recherche

Comme perspectives, notre travail et les travaux de thèse de O. Kassem Zein peuvent être intégrés dans le domaine des services Web (*web services*). En effet, le modèle de méta données que nous avons défini peut être utilisé pour décrire un service Web. Notre trader et les outils de recherche et d'adaptation (profil utilisateur) peuvent être intégrés avec UDDI [120] pour découvrir les services Web. Ainsi, les trois types de la composition de services peuvent être utilisés pour assembler et créer de nouveaux services à partir de services existants ou de spécifications de services. Nous pensons également que notre approche de description du comportement de service peut être compatible avec les langages de description comme WSFL et de ce fait pourrait être également utilisée pour une extension aux services web.

Notre travail peut être étendu et utilisé dans la découverte de services dans les réseaux sans fil spontanés. En effet, chaque service peut publier une abstraction de son fonctionnement. Les services peuvent être découverts dynamiquement par des applications. L'ensemble de services et d'applications forme un environnement spontané qui s'adapte au client en prévoyant ses besoins et en l'accompagnant dans ses tâches et déplacements. À son tour, l'utilisateur peut superviser le réseau et créer des applications à la demande par composition de services.

Ce travail de thèse peut aussi être étendu et utilisé pour prendre en compte les propriétés dynamiques des réseaux. A partir de mesures réalisées par des sondes ou autres agents comme présentés dans la thèse de Martin Heusse [121], il est possible de les synthétiser et ensuite de les indexer, les enregistrer dans l'ontologie et de les mettre à jour périodiquement. Les clients peuvent découvrir les services qui leur conviennent en fonction de ces mesures. Ces informations sur l'état des ressources et voies de communication apporteraient des informations complémentaires et utiles lors de la recherche de services.

CHAPITRE 5

Conclusions et perspectives

J'ai présenté dans ce document les différentes activités que j'ai menées dans les thèmes de la définition et de l'assemblage de composants. J'ai illustré ces thèmes en retenant trois contextes bien différents : les composants logiciels, les composants d'apprentissage et enfin les services distribués. Chacun de ces contextes est exploré dans les chapitres de ce document et j'ai présenté pour chacun d'eux une synthèse de mes activités et une proposition de thèmes à explorer.

Ces trois contextes présentent des similarités intéressantes puisque, pour mener à bien une approche de réutilisation, il faut principalement :

- ◇ Evaluer les pratiques et besoin d'une communauté. En effet, pour qu'une approche de réutilisation puisse être retenue, il faut définir des pratiques et procédures communes qui soient compatibles avec les pratiques antérieures tout en proposant une valeur ajoutée. Dans le cadre des composants d'apprentissage, il s'agit, par exemple, de proposer différents modèles pédagogiques en lien avec les pratiques des enseignants tout en permettant de compléter ou d'adapter (personnaliser) ces modèles. La notion de communauté est aussi importante puisque elle seule peut garantir une qualité des différents composants en imposant des contraintes normatives.
- ◇ Définir des modèles d'indexation. En effet, partager des composants réutilisables implique de les rendre accessibles par une communauté. Un créateur de composant doit donc être capable de décrire de différentes manières son composant et de ce fait doit l'indexer. Les différents membres de cette communauté doivent enfin être capables d'explorer et de rechercher la base de composants afin de retrouver un composant répondant à leurs attentes. Par exemple, pour la description de services web, nous avons proposé un modèle de méta données qui fournit des informations sur le service, son interface et son comportement.
- ◇ Définir des approches d'assemblages et de compositions. Ce thème est naturellement lié à l'intégration du composant dans un environnement et de ses interactions avec cet environnement ou d'autres composants. L'accent peut être aussi mis sur la vérification de la cohérence de l'assemblage. Nous avons, par exemple, présenté tous les bénéfices

de la généricité d'Ada 95 pour l'assemblage de composants logiciels.

Ces différents thèmes de recherche et de développement ont été explorés de différentes manières mais je pense qu'ils ont pu être menés à terme avec succès car :

- ◇ J'ai eu la chance de travailler dans le cadre d'équipes et de projets. Chacun de ces thèmes est vaste et dépasse les capacités d'un individu. Le travail en équipe permet d'explorer des thèmes plus étendus et surtout de confronter des expériences et pratiques diverses.
- ◇ J'ai pu aussi travailler avec des collègues de domaines différents. Il y a certes des informaticiens (terme qui regroupe aussi bien les spécialistes en systèmes distribués ou en compilation en passant par les techniques de représentation de connaissances), des experts en sciences de l'éducation, des sociologues, des spécialistes des couches basses des réseaux, etc. Les interactions avec ces différents interlocuteurs se sont révélées fructueuses et surtout m'ont permis d'aborder des thèmes sous des angles que je n'aurais pas anticipé ou exploré.
- ◇ Les différents projets imposaient des résultats. Un tel contexte apporte de nombreuses contraintes du fait des exigences liées aux échéances mais aussi des financements nous permettant de renforcer nos équipes, des contextes d'application concrets qui nous permettent de mettre en pratique des concepts qui resteraient sinon dans des cartons.
- ◇ Les transferts vers des applications finalisées constituent enfin une valorisation des travaux théoriques tout en motivant les développements puisque les outils sont attendus et destinés à être utilisés.

J'ai identifié des thèmes de recherche qui pourraient être développés et parmi ceux-ci j'en ai retenu trois qui me semblent importants et que je souhaiterais poursuivre et explorer :

- ◇ Le premier est lié aux composants logiciels. Je pense qu'il est possible de décrire les composants logiciels aussi finement que nous l'avons fait pour les services. Une telle description servirait naturellement à indexer les composants et surtout à les retrouver. La réutilisation est certes un acte volontaire du créateur du composant et ceci afin de le rendre réutilisable mais nécessite aussi que les utilisateurs aient une approche de recherche de l'existant. Les techniques d'assemblages de composants et les vérifications de cohérence de l'assemblage me semblent aussi intéressants. Nous espérons continuer les travaux entrepris dans ce thème avec Airbus Industries afin de poursuivre les travaux entamés avec le projet RNTL COTRE et autour d'AADL.
- ◇ Le second thème est celui de l'école doctorale virtuelle. Ce thème est la prolongation de mes activités dans le domaine des nouvelles technologies pour l'enseignement. En effet, si la collaboration des enseignants s'avère indispensable pour la production de cours de qualité et est largement acceptée par les communautés d'enseignants, nous pensons que la collaboration entre chercheurs pourrait aussi bénéficier de tels modèles de coopération. La définition d'une démarche outillée d'exploration d'un espace de

recherche présente de nombreux challenges mais aussi apparaît plus que nécessaire, que ce soit pour les étudiants ou pour les encadrants.

- Le dernier thème est celui des web services dans le contexte des nouvelles technologies pour l'enseignement. Les travaux menés lors de la thèse de O. Kassem Zein sur la recherche et la composition de services pourraient trouver un cadre d'application dans un environnement pour l'enseignement à distance. La recherche et la composition de services permettraient de mieux prendre en compte la mobilité avec le modèle de composition semi-dynamique. Un service est alors décrit par une spécification et, lors de son utilisation effective, un médiateur irait le chercher en prenant en compte les caractéristiques de l'environnement (les services disponibles, les propriétés des voies de communication, etc.) ou préférences de l'utilisateur.

Références bibliographiques

[1] Microsoft Corporation and Digital Equiment Corporation, URL http ://www.microsoft.com/com/resources/comdocs.asp. *Component Object Model specification*, October 1995.

[2] B. Meyer. The significance of components. Software Development online URL :http ://www.sdmagazine.com/documents/s=7207/sdm9911k/, November 1999.

[3] C. Szyperski. *Component Software, Beyond Object-Oriented Programming*. Addison Wesley, 1998.

[4] B. Meyer. On to components. *Computer*, 32(1) :139–143, January 1999.

[5] D. Petit. *Génération automatique de composants logiciels sûrs à partir de spécifications formelles B*. PhD thesis, Université de Valenciennes et du Hainaut Cambrésis, Décembre 2003.

[6] I. Crnkovic and M. Larsson (editors). *Building Reliable Component-Based Software Systems*. Artech House Publishers, July 2002.

[7] G. Haines et al. Component-based software development / cots integration. Technical report, Carnegie Mellon - Software Engineering Institute, 1997.

[8] Alsys. *Reference manual for the Ada programming language*. Alsys, La Celle St Cloud, France, 1983.

[9] E. Schonberg et al. GNAT : the GNU - NYU Ada translator, a compiler for everyone. In *Proceedings of the TRI Ada conference*, Baltimore, Maryland, november 1994. ACM.

[10] Intermetrics. *Ada 95 reference manual*. International Standard ANSI / ISO / IEC-8652 :1995, January 1995.

[11] J.G. Barnes. *Programming in Ada 95*. Addison-Wesley Publishing Company, 1995.

[12] N.R. Mehta, N. Medvidovic, and S. Phadke. Towards a taxonomy of software connectors. In *Software Engineering, 2000*, pages 178–187, June 2000.

[13] M. Raynal. *Algorithmes distribués et protocoles*. Eyrolles, 1985.

[14] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communication of ACM*, 21(7) :558–565, july 1978.

[15] G. Ricart and A. Agrawala. An optimal algorithm for mutual exclusion in computer networks. *Communication of ACM*, 24(1) :9–17, january 1981.

[16] F. Mattern. Virtual time and global states of distributed systems. In *Parallel and distributed algorithms*, pages 215–226. Elsevier Science Publishers B.V., North Holland, 1989.

[17] K. Li and P. Hudak. Memory coherence in shared virtual memory systems. *ACM transactions on computer systems*, 7(4) :321–359, november 1989.

[18] JM Helary, C. Jard, N. Plouzeau, and M. Raynal. Detection of stable properties in distributed applications. In ACM, editor, *Sixth ACM symposium on principles of distributed computing*, pages 125–136, Vancouver, August 1987.

[19] E. Dijkstra, W. Feijen, and A. Van Gasteren. Derivation of a termination detection algorithm for distributed computation. *Information processing letters*, 16 :217–219, June 1983.

[20] R. Buhr. *System design with Ada*. Prentice Hall, 1984.

[21] S.J. Goldstack, A.R. Volz, A. Wellings, and R.K. Power. Virtual nodes/distributed systems working group. *Ada Letters*, pages 66–96, Spring 1990.

[22] R. Volz, R. Thierault, Y. Kermarrec, L. Pautet, S. Tardieu, and G. Smith. Ada 95 distribution annex implementation for GNAT. Technical report, Texas A&M University, College Station, Texas, November 1995.

[23] A. Gargaro, Y. Kermarrec, L. Nana, L. Pautet, G Smith, S. Tardieu, R Thierault, and R Volz. ADEPT technical report : Ada 95 distributed execution and partitionning toolset for GNAT. Technical report, Texas A&M University, April 1996. (http ://www.cs.tamu.edu/research/ADEPT/).

[24] L. Pautet. *Intergiciels schizophrènes : une solution à l'interopérabilité entre modèles de répartition*. PhD thesis, Université Pierre et Marie Curie - Paris VI, December 2001.

[25] Jesús M. González-Barahona, José Centeno-González, Pedro de las Heras-Quirós, Francisco J. Ballesteros-Cámara, and Luis López-Fernández. Teaching network programming with Ada and lower layer.

[26] Jörg Kienzle. Network applications in Ada 95. In *Proceedings of TRI-Ada'97, St. Louis, MO, USA, November 1997*, pages 3–9. ACM Press, 1997.

[27] Kristina Lundqvist and Goran Wall. Using object-oriented methods in Ada 95 to implement linda. In *Ada-Europe*, pages 211–222, 1996.

[28] Thomas Wolf. Fault tolerance in distributed Ada 95. In *Proceedings of the 8th International Real-Time Ada Workshop, Ravenscar, April 1997*, number XVII(5), Sept./Oct. 1997, pages 106–110. ACM Press, 1997.

[29] S. A. Moody. Object-oriented real-time systems using a hybrid distributed model of Ada 95's built-in DSA capability (Distributed Systems Annex-E) and CORBA. *ACM SIGADA Ada Letters*, 17(5) :71–76, /1997.

[30] ExTRA working group. Proposed draft standard for real-time Ada extensions, may 1992. specific Input/Output chapter, release 1.0, version 1.0.

[31] CIFO working group. Asynchronous co-operation mechanisms (with reference memory model). *Catalog of Interface Features and Options for Ada run time environment*, july 1991. Realease 3.0.

[32] Y. Kermarrec and L. Pautet. Ada communication components for distributed and real time applications. In *Proceedings of TRI-Ada'92*, pages 530–536, Orlando, Florida, November 1992. ACM SIGAda, ACM Press.

[33] W. Richard Stevens. *Unix network programming*. Prentice Hall Software series, 1990.

[34] Y. Kermarrec, L. Pautet, and S. Tardieu. GARLIC : Generic Ada Reusable Library for Interpartition Communication. In *Proceedings of TRI-Ada'95*, pages 263–269, Anaheim, California, USA, November 1995. ACM Press.

[35] Message Passing Interface Forum. MPI : a message passing interface standard. Technical Report 230, CS Department, University of Tenessee, Knoxville, April 1994.

[36] G.F. Coulouris and J. Dolimore. *Distributed systems : concepts and design*. Addison Wesley, 1988.

[37] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32(4) :444–458, April 1989.

[38] N. Carriero and D. Gelernter. How to write parallel programs : a guide to the perplexed. *ACM Computing Surveys*, 21(3) :323–358, September 1989.

[39] N. Carriero and D. Gelernter. Applications experience with Linda. In *Proc. ACM Symposium on Parallel Programming*, July 1988.

[40] Y. Kermarrec and L. Pautet. Ada-Linda : A powerful paradigm for programming distributed Ada applications. In *Proceedings of TRI-Ada'94*, pages 438–445, Baltimore, Maryland, October 1994. ACM Press.

[41] Y. Kermarrec and L. Pautet. Integrating page replacement in a distributed shared virtual memory. In *Proceedings of the 14th international conference on distributed computing systems*, pages 355–362, Poznan, Poland, June 1994. IEEE Computer Society Press.

[42] David Wiley. The teacher's outrageous claim of intellectual property. URL http ://wiley.ed.usu.edu/docs/teachers_claim.html, 2000.

[43] David Wiley. *Learning object design and sequencing theory*. PhD thesis, Brigham Young University, June 2000.

[44] David Wiley. *Learning objects – a definition*, chapter Educational technology : an encyclopedia. In A. Kovalchick and K. Dawson (Eds., URL http ://wiley.ed.usu.edu/docs/ocw.pdf, 2002.

[45] Griff Richards. The challenges of the learning object paradigm. *Canadian Journal of learning and technology - Special Issue on learning Objects*, 28(3), 2002.

[46] G. Paquette, I. de la Teja, and A. Dufresne. Explora : An open virtual campus. In *EdMEdia'2000 conference*, pages 844–849, 2000.

[47] A. Dufresne. Conception d'une interface adaptée aux activités de l'éducation à distance - exploragraph. *Sciences et techniques éducatives*, 8(3) :301–320, 2001.

[48] M. Hatala and G. Richards. Global vs. community metadata standards : Empowering users for knowledge exchange. In Horrocks and J. Handler (Eds., editors, *paper presented at the 5th IASTED conference on computers and advanced technologies in education*, pages 292–306. Springer Verlag, 2002.

[49] P. Tchounikine. Environnements informatiques pour l'apprentissage humain. In *Symposium technologies informatiques en éducation*, February 2002.

[50] A. Derycke. Sept questions sur le e-learning. In *Symposium technologies informatiques en éducation*, URL http ://archive-edutice.ccsd.cnrs.fr/docs/00/00/17/71/PDF/Symposium.pdf, February 2002.

[51] D. Wiley, Gibbons, and Recker. A reformulation of the issue of learning object granularity and its implications for the design of learning objects. availble fromhttp ://www.reusability.org/granularity.pdf, 2000.

[52] IEEE. Draft standard for learning object metadata. Technical report, IEEE, http ://ltsc.ieee.org/wg12/, 2002.

[53] IMS Global Learning Consortium. IMS. Technical report, URL http ://www.imsglobal.org, 2004.

[54] T. Carey, J. Swallow, and W. Oldfield. Educational rationale metadata for learning learning objects. *Canadian Journal of learning and technology - Special Issue on learning Objects*, 2002.

[55] ARIADNE consortium. ARIADNE Educational Metadata Recommendation. Technical report, ARIADNE foundation, URL http ://www.ariadne-eu.org, 2002.

[56] Advance Distributed Learning Initiative. SCORM : Sharable Content Object Reference Model. Technical report, URL http ://www.adlnet.org, 2004.

[57] S. Garlatti, Y. Kermarrec, and E. Wiederhold. Specifications and requirements for the authoring, indexing and navigation tools of IST CANDLE. Technical Report deliverable 4.1, CANDLE consortium, 2000.

[58] S. Garlatti, Y. Kermarrec, and P. Wrona. Second prototype of the information broker applications. Deliverable 4.3 of IST CANDLE project, Candle Consortium, December 2002.

[59] S. Garlatti, Y. Kermarrec, and C. Ragnard. Final version of the reference user client. Deliverable 4.5 of IST CANDLE project, Candle Consortium, May 2003.

[60] C. Watkins. and P Mortimore. *Pedagogy : What Do We Know ?*, chapter Understanding Pedagogy and its impact on learning. Paul Champman Publishing Limited, London, 1999.

[61] K. Kuutti. *A framework for HCI research*, chapter In Nardi, B. (Ed.) Context And Consciousness. Activity Theory and Human-Computer Interaction. The MIT Press, 1996.

[62] P. Collins, S. Shukla, and D. Redmiles. Activity theory and system design : a view from the trenches. *Computer supported coopérative work*, 11 :55–80, 2002. URL http ://www.fjeld.ch/litt/collins.pdf.

[63] V. Kaptelinin. *Computer-Mediated Activity : Functional Organs in Social and Developmental Contexts*, chapter In Nardi, B. (Ed.) Context And Consciousness. Activity Theory and Human-Computer Interaction. The MIT Press, 1996.

[64] B. Mann. An Introduction to Rhetorical Structure Theory (RST). http ://www.sil.org/linguistics/rst/rintro99.htm, August 1999.

[65] W.C. Mann and S.A. Thompson. Rhetorical structure theory : Toward a functional theory of text organization. *Text*, 8(3) :243–281, 1988.

[66] Y. Engestrom. *Learning by expanding : an activity theoretical approach to developmental research*. Orienta-Konsultit, Helsinki, 1987.

[67] Y. Engeström, R. Miettinen, and RL. Punamäki, editors. *Perspectives on Activity Theory*. Cambridge University Press, 1999.

[68] A.N. Leont'ev. *Activity, consiousness and personality*. Englewood Cliffs : Prentice-Hall, 1978.

[69] T. Reeves. *Evaluating what realy matters in computer-based education*, chapter Computer education : new perspectives, pages 219–246. in M. Wild and D. Kirkpatrick (Eds), Perth, Australia, 1994.

[70] R. Lewis. Apprendre conjointement : une analyse, quelques expériences et un cadre de travail. In JF. Rouet and B. de la Passardière, editors, *Quatrième colloque hypermédias et apprentissages*, pages 11–28, URL http ://archive-edutice.ccsd.cnrs.fr/docs/00/00/26/51/PDF/HyperAp4p011.pdf, 1998.

[71] N. Friesen and A. Roberts. Cancore : learning object metadata. *Canadian Journal of learning and technology - Special Issue on learning Objects*, 2002.

[72] W. Jarret, E. Mendes, and O. Prnjat. Complete metadata model, ontology and guidelines for the subject domain. Technical Report deliverable 2.4, CANDLE consortium, July 2002.

[73] T. Berners-Lee. Semantic web on XML, December 2000. http ://www.w3.org/2000/Talks/1206-xml2k-tbl.

[74] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 1993.

[75] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker : Ontology based access to distributed and semi-structured information. In R. et al. Meersman, editor, *Conference on Database Semantics*, Rotorua, New Zealand, 1999. Kluwer Academic Publishers.

[76] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker : Or how to enable intelligent access to the www. In *Proceedings of the 11th Knowledge Acquisition for Knowledge-Based System Workshop*, Banff, Canada, April 1998. KAW'98.

[77] N. Kamolvilassatian, M. Vazguez, S. Taank, and S. Banerjee. Multi-domain recommender system : Utilizing ontology information and two-level user model representation in personalization. Technical report, University of Texas, Austin, TX 78712 USA, May 2001. Url : http ://www.cs.utexas.edu/users/noppadon/datamining/index.html.

[78] N. Balacheff. Environnements informatiques et apprentissage humain. In *Symposium technologies informatiques en éducation*, February 2002.

[79] P. Tanguy. Réalisation d'une plate-forme de composition sémantique de dossiers thématiques de presse. Master's thesis, Mémoire de fin d'études CNAM, June 2002.

[80] S. Iksal, S. Garlatti, F. Ganier, and P. Tanguy. Semantic composition of special reports on the web : A cognitive approach. In J.-P. Balpe, S. Leleu-Merviel, I. Saleh, and J.-M. Laubin, editors, *Hypertextes - Hypermédias : Nouvelles écritures, nouveaux langages*, pages 363–378. H2PTM'01, Hermes, October 2001.

[81] S. Iksal. *Spécification déclarative et composition sémantique pour les documents virtuels personnalisables*. PhD thesis, EHESS et ENST Bretagne, December 2002.

[82] R. Bræk, G. Hasnes, and O. Brevik. Author's workbench usage report : the CANDLE RE-Engineering Method - CREEM. Technical Report deliverable 5.1, CANDLE consortium, December 2002.

[83] M. Grandbastien. Indexation et recherche de ressources pédagogiques sur le web. In *Symposium technologies informatiques en éducation*, URL http ://archive-edutice.ccsd.cnrs.fr/docs/00/00/17/71/PDF/Symposium.pdf, February 2002.

[84] S. Garlatti and S. Iksal. Concept filtering and spatial filtering in an adaptive information system. In P. Brusilovsky, O. Stock, and C. Strapparava, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 315–318. AH 2000, Springer Verlag (LNCS 1892), August 2000.

[85] S. Garlatti and S. Iksal. Documents virtuels personnalisables pour des systèmes d'information en ligne. In *Workshop sur les Documents Virtuels Personnalisables*, pages 22–26. IHM'99 - Workshop DVP, November 1999.

[86] J.-P. Doignon and J.-Cl. Falmagne. *Knowledge Spaces*. Springer, Berlin, 1999.

[87] CLOE Co-operative Learning Object Exchange Consortium. Proposed CLOE review process. URL http ://cloe.on.ca/cloeprocess.pdf, June 2004.

[88] CLOE Co-operative Learning Object Exchange Consortium. CLOE guideline for authors. URL http ://cloe.on.ca/CLOE_Guidelines_for_Authors.doc, June 2004.

[89] G.F. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems : Concepts and Design*. Addison-Wesley, third edition, 2001.

[90] A. Tanenbaum and M. van Steen. *Distributed Systems : Principles and Paradigms*. Prentice Hall, 2002.

[91] Al Geist et al. *PVM : Parallel virtual Machine*. The MIT Press, 1994. also available from ftp netlib2.cs.utk.edu.

[92] Ward Rosenberry, David Kenney, and Gerry Fisher. *Understanding DCE*. O'Reilly and associates, inc, 1993.

[93] Ch. Knouse. *Practical DCE programming*. Prentice Hall PTR, 1995.

[94] J Siegel et al. *CORBA fundamentals and programming*. OMG Press, 1996.

[95] Sean Baker. *CORBA distributed objects using ORBIX*. ACM Press and Addison Wesley, 1997.

[96] DEC, HP, and et al. CORBAservices : Common object services specification. Technical Report OMG 95-3-31, Object Management Group and X Open, March 1995.

[97] H. Bal and A. Tannenbaum. Orca : a language for distributed object-based programming. Technical Report IR 140, Vrije University, Amsterdam, December 1987.

[98] W. Emmerich. *Engineering distributed objects*. Wiely, 2000.

[99] A.P.M Ltd. *The ANSA Reference Manual Release 01.00*. APM Cambridge Limited, UK, March 1989.

[100] ISO/IEC JTC1/SC21/WG7/N743. *Working Document on Topic 9.1 - ODP Trader*. November 1992.

[101] CORBAservices. *Common Object Services Specification*. OMG Document, December 1997.

[102] www.orbacus.com.

[103] G. Brose. JacORB - a Java Object Request Broker. Technical Report B-97-02, Freie Universität Berlin, April 1997.

[104] H. Kasper, P.V. Helsdingen, and W.D. Vries. *Services Marketing Management : an International Perspective*. John Willey, Chichester, UK, 1999.

[105] C. Lovelock. Classifying Services to Gain Strategic Marketing Insights. *Journal of Marketing*, pages 47 : 9–20, 1983.

[106] M. Merz, M. Witthaut, and S. McConnel. *Catalogue and Service Architecture*. http ://osm-www.informatik.uni-hamburg.de/osm-www/public, 1997.

[107] DAML Services. http ://www.daml.org/services/.

[108] W. Keong-Ng, G. Yan, and E-P. Lim. Heterogeneous Product Description in Electronic Commerce. *SIGecom Exch.*, 1(1) :7–13, 2000.

[109] O. Kassem Zein and Y. Kermarrec. A metadata model for service description and user profile and facilities of service retrieving in distributed systems. Tampere, Finlande, June 2004. IFIP/IEEE Workshop on Advances in fixed and mobile networks.

[110] ITU-T Recommendation Z.100. *Specification and Description Language (SDL)*. 1989.

[111] L. De Alfaro and T. A. Henzinger. Interface Automata. In ACM Press, editor, *The Ninth Annual Symposium on Foundations of software Engineering*, 2001.

[112] E. Christensen, F. Cubera, G. Meredith, and S. Weerawarana. *Web Services Description Language (WSDL)*. http ://www.w3.org/TR/wsdl, March 2001.

[113] C. Hoare. Communicating sequential process. *CACM*, 21(8) :666–677, august 1978.

[114] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4) :741–843, 1995.

[115] OntoPrise. *How to Write F-Logic Programs - A Tutorial for the Language F-Logic*. OntoPrise GmbH, http ://www.ontoprise.de/download/f_logik_tutorial.zip, 2002.

[116] M. F. McTear. User modelling for adaptive computer systems : a survey of recent developments. *Artificial Intelligence Review*, pages 7 : 157–184, 1993.

[117] O. K. Zein and Y. Kermarrec. An Approach for Describing and Querying Service Behavior in Distributed Systems. In *6èmes Journées Doctorales Informatique et Réseau*, Lannion, France Télécom R&D, 2-4 novembre 2004.

[118] O. K. Zein and Y. Kermarrec. An Approach for Describing User Service Interfaces in Distributed Systems. In *The International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA*, pages 1441–1446, Las Vegas, USA, June 23 - 26 2003.

[119] J. Yang, M. Papazoglou, and W-J van den Heuvel. Tackling the Challenges of Service Composition in E-Marketplaces. In *The 12th IEEE International Workshop on Research Issues in Data Engineering : Engineering e-Commerce/ e-Business Systemes (RIDE 2002)*, 2002.

[120] *Universal Description, Discovery, and Integration (UDDI)*. http ://www.uddi.org.

[121] Martin. Heusse. *Routage et équilibrage de charge par agents dans les réseaux de communication*. PhD thesis, EHESS et ENST-Bretagne, 2001.

ANNEXE

A Curriculum Vitae

A.1 Renseignements administratifs

- ◇ Yvon Kermarrec
- ◇ Maître de conférences à l'ENST Bretagne
- ◇ né le 12 avril 1962 à Plouzévédé (Finistère)
- ◇ Marié et père de deux enfants
- ◇ Résidant au 5 rue de l'Harteloire à Brest
- ◇ Téléphone au bureau 02 29 00 12 85 et au domicile 02 98 80 52 90

A.2 Groupes et activités de Recherche

- ◇ Doctorant à l'ENST Bretagne et à l'IRISA de 1985 à 1988 :
  "Une approche pour la simulation des systèmes distribués : les composants logiciels en Ada". Dans le cadre de cette thèse, les aspects suivants ont été étudiés :
  - Etude et comparaison des langages d'expression du parallélisme (Ada, Occam, Csp et Fdt-Estelle).
  - Etude des techniques du contrôle distribué et des méthodes de conception de ces systèmes.
  - Etude et comparaison de machines parallèles (Transputer, iPSC, FPS-T en particulier).
  - Programmation en Ada d'un simulateur d'une machine à base de Transputers et mise en œuvre d'un mécanisme réparti du contrôle de la simulation.

- ◇ Chercheur au Courant Institute, New York University (NYU) - 1988 à 1990
  - Améliorations du compilateur Ada de NYU afin qu'il puisse être validé par les tests de l'ACVC (Ada Compiler Validation Capability).
  - Conception et mise en œuvre d'une nouvelle représentation des objets Ada lors de l'exécution (contrôle des contraintes sur les sous-types plus efficaces, par exemple).
  - Collaboration avec IBM pour la conception de compilateurs Ada pour les systèmes RT et RP3 (une machine hautement parallèle).
  - Collaboration avec IBM en vue de la conception d'une interface entre le frontal du compilateur Ada et un optimiseur de code développé par IBM.

  - Cycle de conférences au Courant Institute sur l'algorithmique pour le contrôle distribué.

◇ Stage de recherches post-doctorales à l'Université de York (Grande-Bretagne) - Mars 1988 à Octobre 1988 - équipe de I. Wand et A. Wellings
  - Finalisation et extension du simulateur de réseaux de Transputers.
  - Etude des différentes solutions en vue de la répartition de programmes Ada sur machines distribuées.
  - Mise en œuvre de la détection de propriétés stables (terminaison et interblocage) dans le cas d'une exécution de programmes Ada sur machines multi-processeurs (projet York Distributed Ada)

◇ Maître de conférences à l'ENST Paris puis à l'ENST Bretagne depuis 1991
  - Activités d'enseignement, de recherche et d'administration
    - Au département Informatique de l'ENST Paris
    - Au département Informatique et Réseaux de l'ENST Bretagne
    - Au département Intelligence Artificielle et Sciences Cognitives de l'ENST Bretagne
    - Chef de projets de Recherche et Développement et responsable de contrats.
    - Responsabilités collectives au niveau des écoles du GET

A.3 Activités d'enseignement

◇ Depuis décembre 1991. Enseignant chercheur à Télécom Bretagne et Télécom Paris.
  - Responsabilité de la formation troisième année en option CHMEST (de 1998 à 2000).
  - Responsabilité de modules d'enseignement en troisième année (systèmes distribués, systèmes multi-agents, compilation)
  - Enseignements de première année : algorithmique et introduction au langage Ada.
  - Enseignements de deuxième année : langages formels, bases de données et parallélisme.
  - Enseignements de troisième année : compilation, algorithmique distribuée, systèmes multi-agents, systèmes et plates-formes distribuées.
  - Enseignements en formation continue : Transputer et Occam, langage Ada, langage Ada 9X, outils pour le calcul intensif sur machines en réseau, outils avancés du monde Unix.
  - Responsable des stages pour le département.
  - Encadrement de nombreux projets d'étudiants : études bibliographiques, projets d'ingénieurs (d'une durée de 5 mois associant deux départements de l'ENST Bretagne et un industriel), projets cours.
  - Encadrement de stages de DEA et suivi de stages d'ingénieurs.

◇ De mars 1990 à décembre 1991. Enseignant chercheur à Télécom Paris.

  - Création et responsabilité de trois modules d'enseignement de troisième année : compilation, algorithmique et machines parallèles, algorithmique du contrôle distribué.
  - Cours d'Ada et de génie logiciel en deuxième année et aux Mastères.
  - Rédaction d'un polycopié pour le module de compilation.
  - Organisation et enseignement dans le cadre d'une Ecole Erasmus - 6 Angles (Aachen).

◇ Enseignant vacataire à l'Université de Bretagne Occidentale
  - de 1985 à 1987 : Création d'un module et enseignement (cours, TD, TP) en bases de données en maîtrise d'information (SMI)
  - de 1997 à 2002 : enseignement (cours) en temps réel en DESS informatique
  - depuis 2001 : enseignement (cours et TD) en 3ème année d'IUP informatique consacré à la programmation des systèmes temps-réel avec Ada

◇ Depuis septembre 1998. Enseignant à l'ISEB (groupe ISEN)
  - Création et responsabilité d'un module systèmes distribués (cours, TD, TP)

◇ Depuis septembre 2001. Enseignant à l'ENSIETA
  - Enseignements (cours, TD et TP) dans un module systèmes embarqués

◇ Depuis septembre 2002. Enseignant à l'Université de Bretagne Sud
  - Cours en DEA IHM en Systèmes Multi Agents
  - Cours en DESS ASIR (commun au DEA IHM)

◇ Depuis mars 1992. Enseignant à l'Ecole Navale de Brest.
  - Création et responsabilité d'un module langages et compilation.
  - Création et responsabilité d'un module systèmes embarqués.
  - Présidence des Jurys de fin d'étude des élèves officiers de l'option Informatique depuis 1999

◇ Cours à l'International
  - Tutoriaux en conférences internationales : conférences ACM et IFIP.
  - Cours dans des Universités en Suède, Pologne et États-Unis.

A.4 Diplômes

◇ **Doctorat en informatique** - Juin 1988 - Mention "très honorable" - Université de Rennes I - Jury composé de Michel Raynal (Rennes I), André Schiper (EPFL Lausanne), Françoise André (Rennes I), Jean Ferrié (Université de Montpellier), Raymond Marie (Rennes I), Robert Rannou et Daniel Herman (Rennes I).

◇ **DEA d'informatique** - Juin 1985 - option systèmes distribués et bases de données -

Mention AB.

A.5 Activités internationales

- ◇ Coordonateur depuis 1999 du réseau EUNICE : un groupe de 23 Universités Européennes dans le domaine des réseaux.
- ◇ Représentant du GET au Réseau d'excellence Kaléidoscope (6ème PCRD) consacré à l'enseignement à distance
- ◇ Research Visitor à University of York, Grande-Bretagne
- ◇ Research Scientist au Courant Institute of Mathematical Sciences de New York University, Etats-Unis
- ◇ Séjour sabbatique à New York University (collaboration en recherche autour du système GNAT : 2 mois en 1993 et 3 semaines en 1994)
- ◇ Un an en séjour sabbatique industriel avec Raytheon à Vancouver (projet CAATS et MAATS - Canadian Automated Air Traffic Systems - Military Automated Air Traffic Systems). J'avais en charge l'architecture distribuée de la plate-forme et l'optimisation des ressources (mémoire, temps de réponse et de calcul, en particulier).
- ◇ Membre et point de contact du forum HP-European Universities (HP OVUA - Open View University Association).
- ◇ Organisation d'une Ecole européenne dans le cadre ERASMUS (Delft University, University College London, Aachen University, Université de Louvain la Neuve, Leuven University) avec le thème de l'intelligence artificielle et du parallélisme.
- ◇ Enseignement en Anglais à Uppsala, Suède (master informatique) et Poznan (programme Tempus).

A.6 Participation à des Comités scientifiques

- ◇ Membre du comité scientifique de l'IRENAV : Laboratoire de Recherche de l'Ecole Navale.
- ◇ Depuis 1997 : Membre du comité de programme des conférences internationales sur les technologies logicielles sûres (international conference on reliable software technologies - Ada Europe). ACM Sig Ada
- ◇ Depuis 1999, membre des comités de programmes des conférences EUNICE - Affiliées IEEE, IFIP et WATM
- ◇ Membre des comités de programme des conférences ACM Tri Ada en 96, 97 et 98
- ◇ Membre du bureau d'Ada France / AFCET (en 1997 et 1998) et membre du comité de programmes des conférences Ada France
- ◇ Membre du comité de programme et d'organisation de la conférence DVP de 2002
- ◇ Membre du comité de programme des conférences RTS (Real Time Systems) en 1994 et 1995

- ◇ Membre du comité de programme du premier workshop franco chinois sur les outils informatiques pour la sécurité maritime - juin 2002 (Ecole Navale et Shanghai Maritime University).
- ◇ Evaluation de projets pour la Suisse et de candidatures pour la Suède.
- ◇ Chairman de sessions des conférences mentionnées ci-dessus.

A.7 Administration et responsabilités collectives

- ◇ Coordonateur du réseau EUNICE : un groupe de 23 universités européennes dans le domaine des réseaux de télécommunications.
- ◇ Membre du Réseau d'Excellence Kaleidoscope (*"Network of Excellence"* du 6ème PCRD) et représentant du GET
- ◇ Responsablibilité et animation des activités de recherche dans le domaine des services télécoms.
- ◇ Responsabilité de contrats industriels et de projets de recherche : montage, négociation, suivi et réalisation :
  - Etablissement de contrats au niveau national dans le cadre RNRT et RNTL, au niveau européen (programme IST des 5ème et 6ème PCRD) et international (New York University, Texas A&M University et LICEF -Canada)
  - Etablissement de contrats avec des industriels (Dassault Electronique, Thomson CSF, TNI, Airbus Industries).
- ◇ Coordination de modules d'enseignement, organisation de séminaires et accueil des intervenants.
- ◇ Responsabilité de l'interface entre la logistique informatique et les chercheurs du département informatique de l'ENST Bretagne (de 1992 à 1997).
- ◇ Responsabilité des stages à l'étranger et établissement de collaborations avec des Universités étrangères (Carleton U. à Ottawa, New York University, York University, Carnegie Mellon University, Aachen, Delft, Lund, Uppsala)
- ◇ Animation d'un groupe de recherche d'universitaires européens dans le cadre du forum HP Universités.
- ◇ Présidence des jurys de fin d'études des élèves officiers de l'Ecole Navale.
- ◇ Participation à plusieurs jurys de thèse.
- ◇ Rapporteur (*"Opponent"*) de la thèse de L. Bjornfot à Uppsala University.

A.8 Organisation de congrès

J'ai organisé quatre éditions du *Workshop on Software methods and tools for Ada 95*. Cette manifestation internationale a regroupé à chacune de ses éditions 80 experts dans le domaine des grands projets logiciels et de l'utilisation des technologies autour du langage Ada 95. Les participants provenaient du monde industriel et académique. Le programme comprenait des

tutoriaux de personnalités reconnues dans le domaine et de présentations de grands projets. Cette manifestation scientifique a été organisée avec l'UBO, TNI et l'Ecole Navale.

Parmi les principaux intervenants, nous avions invité en particulier :

- ◇ E. Schonberg - Professeur New York University, USA
- ◇ R. Dewar - Professeur New York University, USA
- ◇ A. Strohmeier - Professeur EPFL - Suisse
- ◇ L. Moody - Chef de projet Boeing Defense - Seattle, USA
- ◇ L. Asplund - Professeur Uppsala University, Suède
- ◇ A. Wellings - Professeur University of York, GB
- ◇ V. Celier - Chef de projet Raytheon - Vancouver
- ◇ J. Sherrill - concepteur des exécutifs temps réel RTEMS, USA
- ◇ J. Barnes - un des principaux concepteurs du langage Ada, GB
- ◇ JP Rosen - consultant
- ◇ L. Pautet, ENST Paris
- ◇ M. Voytko, Boeing GB
- ◇ P. Farail, Airbus Industries, France

Chaque édition de ce congrès était consacrée à un thème identifié comme étant exploratoire en recherche et qui correspondait à des besoins majeurs des industriels et grands équipementiers. Les thèmes successivement abordés :

- ◇ 1996 : la nouvelle norme du langage Ada et la transition de Ada 83
- ◇ 1997 : les applications distribuées et temps réel avec Ada 95
- ◇ 1999 et 2001 : la conception des grandes applications logicielles et le modèle objet

Je coordonnais le comité de programme et le comité d'organisation et assurais l'équilibre financier des opérations par la recherche de subventions (régionales et européennes) et de sponsors industriels. Chaque édition nécessitait un budget d'environ 100 000 euros.

A.9 Participation à des jurys de thèse

- ◇ Membre du jury de thèse de Martin Kruger à l'ENST Paris. Titre de sa thèse : Méthode d'analyse d'algorithmes d'optimisations stochastiques à l'aide d'algorithmes génétiques" - 1993. Directeur de thèse : Irène Charron Fournier
- ◇ Membre du jury de thèse de Laurent Pautet à l'ENST Paris Titre de la thèse : "Conception et réalisation de composants logiciels pour applications distribuées temps réel" Directeur de thèse : Anne Germa.
- ◇ Membre du jury de thèse de L. Nana à l'Université de Rennes I. Titre de la thèse : "Ada 95 et les systèmes distribués : la tolérance aux fautes". Directeur de thèse : M. Raynal
- ◇ Opponent de la thèse de L. Bjornfot. Université d'Uppsala en Suède. Titre de la thèse : "Ada and timed automata". Thèse dirigée par L. Asplund.
- ◇ Membre du jury de thèse de Martin Heusse. Titre de la thèse : "Routage et équilibrage

de charge par agents dans les réseaux de communication". Directeur de thèse : JP Barthelemy.

◇ Membre du jury de thèse de Cyril Ray : "Atlas une plate-forme pour la modélisation et la simulation de systèmes désagrégés". Thèse dirigée par Ch. Claramunt à Rennes I. 2003.

A.10 Thèses encadrées ou dirigées

◇ **Laurent Pautet**

- Titre de la thèse : "conception et réalisation de composants logiciels pour applications distribuées temps réel"
- Période de thèse : 1991 à 1994
- Composition du jury : E. Schonberg (New York University), A. Strohmeir (EPFL), R. Rannou (ENST Bretagne), C. Goethals (Dassault), C. Kayser (CNAM), A. Germa (ENST Paris) et Y. Kermarrec
- Résumé :
  L'objectif principal de cette thèse a porté sur la spécification et la conception de composants pour les applications temps réel distribuées. Les travaux en standardisation dans ces domaines ont été suivis afin de permettre la ré-utilisabilité et l'évolution de ces composants. Une large palette d'algorithmes a été explorée afin de fournir une grande panoplie d'outils de contrôle ou de cohérence répartie. Ces outils, ces algorithmes et ces propositions préservent les considérations et les contraintes du temps-réel.
  Enfin ces travaux de thèse ont permis de compléter un environnement de découvertes et de supports à l'enseignement des systèmes distribués. Ces composants, de par leur niveau d'abstraction et leur fonctionnalités, permettent d'établir un lien entre une approche système (trop souvent de bas niveau) et une approche théorique des systèmes distribués.

◇ **Laurent Nana Tchamnda**

- Titre : "Ada 95 et les systèmes distribués : la tolérance aux fautes"
- Période de thèse : 1994 à 1997
- Composition du jury : M. Raynal (Université de Rennes I), E. Schonberg (New York University), D. Herman (Université de Rennes I), L. Marcé (Université de Bretagne Occidentale), Y. Kermarrec et R. Rannou (ENST Bretagne),
- Résumé :
  Le travail de thèse a porté essentiellement sur l'étude et l'intégration de mécanismes complémentaires de tolérance aux fautes logicielles dans les langages de programmation existants et leurs outils associés (compilateurs, préprocesseurs), et plus particulièrement dans le langage Ada 95 et le compilateur GNAT (GNU NYU Ada Trans-

lator). Ces travaux avaient deux objectifs principaux : d'une part, fournir un environnement minimal pour le développement d'applications tolérantes aux fautes, qu'elles soient centralisées ou distribuées, et d'autre part, expérimenter les nouvelles structures du langage Ada 95 dans le cadre de la tolérance aux fautes et de la programmation d'applications distribuées. Le choix du langage Ada 95 comme langage support n'a pas été le fait du hasard. En effet, de nombreux travaux avaient été effectués antérieurement par l'équipe RIST, dont certains avaient conduit à la réalisation d'une bibliothèque de composants pour le développement d'applications temps réel distribuées en Ada 83. L'étude de mécanismes de tolérance aux fautes était un prolongement logique de ces travaux, compte tenu de l'importance de tels mécanismes dans les systèmes temps réel distribués. Par ailleurs, le langage Ada a été initialement conçu pour le développement d'applications critiques où la sûreté et la sécurité sont les principaux objectifs et il se prête bien au développement de larges applications de génie logiciel.

Le travail s'est organisé en quatre phases principales : l'étude et l'intégration des blocs de recouvrement en environnement centralisé, la mise en œuvre de l'annexe distribuée du langage Ada 95 dans le compilateur GNAT, l'extension du modèle initial de blocs de recouvrement pour permettre leur exécution en environnement distribué, et la conception d'un modèle pour la réplication et la reconfiguration d'applications distribuées Ada 95.

◇ **Zied Choukair**

- Sujet : "Inter-opérabilité des objets distribués : extension temps réel du modèle CORBA et applictaion avec Ada 95"
- Période de thèse : 1994 à 1997
- Composition du jury : J. Bézivin (Nantes), M. Feldman (G. Washington University), Y. Kermarrec et R. Rannou (ENST Bretagne), J. Rouillard (ESIM Marseille), G. Vidal-Naquet (Paris 11 - Orsay)
- Résumé :

  Le travail de thèse s'est intéressé à l'interopérabilité entre les objets issus de mondes différents. Dès le début de cette thèse, le modèle CORBA est apparu comme une approche intéressante pour assurer la communication et les interactions entre objets et nous avons essayé de la rapprocher du modèle Ada pour la distribution et les systèmes temps-réel. La principale contribution de ces travaux est la proposition de COREMO qui est un modèle issu de CORBA et compatible avec Ada 95.

◇ **Martin Heusse**

- Période de thèse : 1998 à 2001
- Sujet : "Routage et équilibrage de charge par agents dans les réseaux de communica-

tion".

- Composition du jury : R. Euler (UBO), F. Kordon (Paris VI), P. Rolin (France Telecom R&D), JP Barthelemy (ENST Bretagne), B. Lecler (EHESS), Y. Kermarrec
- Résumé :

  Le routage par agents est le produit de l'application au domaine des réseaux de techniques d'optimisation distribuées issues de l'intelligence artificielle. Ce travail de recherche se révèle nécessaire du fait de l'évolution des utilisations et des mécanismes implantés sur les réseaux, qui créent une demande d'évolution du routage qui soit adaptée aux nouvelles possibilités et aux besoins de ces derniers.

  Un nouvel algorithme de routage est proposé dans cette thèse : il repose sur l'utilisation d'agents, ou sondes, aptes à établir des tables de routage adaptées, en particulier, à l'équilibrage de charge. Avec cette approche, la résolution des problèmes liés au routage n'est plus confiée à des entités logicielles liées aux routeurs mais à des «agents», qui collectent des informations variées sur l'état du réseau et les répercutent sur les routeurs au cours de leurs déplacements.

◇ **Oussama Zein**

- Période de thèse : 2001 à 2004
- Résumé :

  Les travaux de thèse portent sur l'élaboration des nouvelles approches permettant l'indexation et la recherche multicritères de services dans le cadre des systèmes distribués. L'approche est de proposer une recherche multi-critères étendue et d'élargir les fonctionnalités des annuaires standards comme celui d'ODP et d'OMG CORBA dont l'architecture sous-jacente repose sur des bases de données. L'approche retenue consiste à intégrer le domaine des ontologies et celui de la représentation de connaissances dans le domaine de la description et la découverte de services distribués.

  Dans le cadre de ces travaux de thèse, les études suivantes ont été menées :

  - Un modèle de méta données associé à la description de services a été proposé . Ce modèle prend en compte les aspects statiques de services (son fournisseur, sa localisation, etc.), leurs comportements (leurs fonctionnalités) et leurs interfaces (leurs opérations, leurs paramètres, etc.). Il peut être pris en compte lors de la recherche de services par les clients.
  - Un trader élaboré basé sur les ontologies et la représentation de connaissances permettant la recherche flexible de services rend accessible ce modèle de méta données. Il permet de chercher les services d'une manière plus sophistiquée que le trader standard en fournissant aux clients un langage logique pour interroger les services. Ce trader fournit les mêmes outils d'interrogation pour les trois niveaux de description de services.
  - Une approche d'adaptation de services selon les besoins des clients a été proposée afin de prendre en compte des aspects dynamiques de la configuration d'un

système distribué.
- Ce travail s'est ensuite poursuivi avec la proposition d'une approche sur la composition de services. Elle permet de créer de nouveaux services à partir des services existants. Elle enrichit le modèle de services existants et permet aux clients d'avoir des services supplémentaires pouvant répondre à leurs demandes.
- La contribution majeure de cette thèse est de pouvoir décrire de diverses manières les services et leurs caractéristiques et de pouvoir ensuite le rechercher à l'aide d'une interface souple et flexible. L'association des ontologies et de techniques de représentation de connaissances nous permettent d'atteindre de tels objectifs.

A.11 Stages de DEA encadrés ou dirigés

- ◇ Laurent Guerby - DEA Rennes 1 (avec E. Schonberg, NYU) - "Simulation de systèmes temps réel en utilisant l'annexe systèmes distribués d'Ada 95" - 1996
- ◇ Arnaud Schach - DEA UBS - "Définition du méta modèle pour COTRE" - 2001
- ◇ Lian Chiang - DEA Rennes 1 - "Le modèle Linda pour la programmation distribuée avec Ada" - 1994
- ◇ Alain Le Guennec - DEA Rennes 1 (avec L Asplund, University of Uppsala) - "Modèle ASIS (Ada Semantic Interface Specifications) et GNAT" - 1996
- ◇ Dominique Le Campion - DEA Rennes 1 - "Systèmes embarqués avec GNAT et RTEMS" - 1996
- ◇ Mikael Rolander - Master Uppsala - "Routage adaptatif pour les réseaux actifs" - 2000
- ◇ Thi Ngo : architectures pour les services webs - DEA Paris 8 - 2003

A.12 Les projets en Recherche et Développement

Les projets ci-dessous, plus applicatifs, donnent une image complémentaire de mes activités en recherche et développement menées depuis mon doctorat. Ils illustrent, par ailleurs, certaines formes d'intégration que j'ai poursuivies à travers les thématiques prioritaires des axes de recherche des différentes équipes auxquelles j'ai été attaché. J'assumais pour l'ensemble de ces projets les fonctions de responsable (soit au sein de l'ENST Bretagne, soit vis-à-vis du commanditaire) et ai assuré toutes les tâches depuis les phases de montages et de négociation jusqu'à la recette et livraison finales. Le lecteur intéressé trouvera ci-après une présentation synthétique des projets avec leurs objectifs et les partenaires impliqués, ainsi que des informations complémentaires.

A.12.1 Projet *AMARRAGE*

A.12.1.1 Description succincte

Ce projet vise à expérimenter et à déployer des applications complexes sur un réseau actif. Les réseaux actifs constituent une approche d'architecture de réseau orientée programme dans laquelle les messages transportent les données et le code qui peuvent s'exécuter dans le réseau. Les objectifs du projet AMARRAGE sont de deux ordres : d'une part définir, concevoir, développer et valider la faisabilité d'un nouveau concept de « Réseau Actif » sur deux applications significatives (la communication multimédia et l'administration de réseau) et, d'autre part, de mettre en place et en œuvre une plate-forme de démonstration à l'échelle géographique de la France à l'horizon de 24 mois pour la communauté scientifique. La communication multimédia correspond à une visiophonie en point à point ou multipoint, ainsi qu'à une application de diffusion au format MPEG4. L'administration de réseau a pour objectif la configuration dynamique et la supervision de service, greffées aux nœuds actifs. La plate-forme réseau étendu se déploiera sur la plate-forme de réseau expérimental de France Télécom R & D et traitera des contenus de la Cité des Sciences. La plate-forme réseau entreprise sera exhibée à Thomson.

A.12.1.2 Contexte du projet

- ◇ *Commanditaire :* Réseau National de Recherche en Télécommunications (projet RNRT)
- ◇ *Partenaires Industriels :* Thomson-CSF Communications, France-Télécom R & D, Synchronix Software, GET.
- ◇ *Partenaires Académiques :* Institut Galilée - Université Paris 1, LAAS Toulouse, Laboratoire Informatique de Paris 6, LORIA - INRIA et PRiSM - Université de Versailles
- ◇ *Durée :* 2000–2002 *Financement pour notre équipe :* 200 000 Francs

A.12.2 Projet *IST CANDLE*

A.12.2.1 Description succincte

CANDLE est un projet européen IST du 5ème PCRD. Il a pour objet l'utilisation d'Internet pour améliorer la qualité de l'enseignement en Europe tout en réduisant les coûts. Il s'agit d'utiliser le WEB, la technologie multimédia et de permettre la coopération entre les universités et l'industrie dans la création, le partage et la réutilisation de matériaux pédagogiques. D'un point de vue recherche, il s'agit de mettre en œuvre des outils de création pour les auteurs de cours, ainsi qu'un système de production de cours adaptés aux apprenants suivant différentes approches pédagogiques. Nous avons présentés les objectifs du projet et nos contributions dans ce document.

A.12.2.2 Contexte du projet

- ◇ *Projet européen :* 5ème PCRD - programme IST ("Information Society Technologies")
- ◇ *Partenaires Industriels :* BT (British Telecom) et Siemens
- ◇ *Partenaires Académiques :* Université de Karlsruhe, UPC Barcelone, UCL Londres, UT Torino, NTNU Norvège, Université de Stuttgart, IOE (Institute of Education) Londres, University of Twente
- ◇ *Durée :* 2000–2003
- ◇ *Montant :* 252 000 euros pour notre équipe de l'ENST Bretagne - 2,3 M-euros pour le projet dans sa globalité.
- ◇ *Adresse WEB :* http ://www.candle.eu.org/

A.12.3 Projet *CARISM*

A.12.3.1 Description succincte

Le projet a pour objectif de définir et démontrer une plate-forme intergicielle dédiée aux réseaux ambiants et systèmes mobiles. Cette plate-forme comportera, notamment, les fonctions de découverte de services, de vérification de leur interopérabilité, et de reconfiguration de l'intergiciel pour la prise en compte de la qualité de service. Dans le contexte que nous envisageons, les composants applicatifs découvrent dynamiquement les services offerts puis les soumettent aux intergiciels pour assemblage et exécution. En fonction des besoins spécifiques à ces assemblages, les intergiciels doivent alors se reconfigurer pour assurer la qualité de service requise. Ce projet vise aussi à développer une expertise permettant au GET de faire valoir une expérience dans les domaines des systèmes répartis, temps réel et mobiles pour réseaux ambiants. Cette expertise sera valorisée auprès de réseaux d'excellence du 6ième PCRD (ARTIST, CaberNet) ou encore auprès des consortiums français (ObjectWeb).

A.12.3.2 Contexte du projet

- ◇ *Contexte :* projet incitatif inter établissements du GET
- ◇ *Montant pour notre équipe :* 20 000 euros
- ◇ *Partenaires Académiques :* ENST de Paris, INT
- ◇ *Durée :* 2002–2004

A.12.4 Projet *DCE*

A.12.4.1 Description succincte

DCE (Distributed Computing Environment) issu de l'OSF propose les services nécessaires au développement des applications distribuées. Les services offerts par DCE abordent un spectre large : de la communication aux processus légers en passant par la sécurité. Le but de

ce projet est d'évaluer DCE dans un contexte télécommunications et de proposer des extensions en fonction des besoins des utilisateurs.
DCE est une plate-forme qui permet de transformer un ensemble de machines en une ressource unique et de programmer des applications pour une telle cible. Les avantages qui en résultent sont classiques et les applications peuvent bénéficier de puissance de calcul (course aux performances), de disponibilité (cas de la tolérance aux fautes) ou de services comme la migration et la communication-synchronisation.
DCE se veut une plate-forme et offre des services de base qui permettent au concepteur de s'affranchir de nombreux détails de bas niveau (gestion de flots d'instructions, communication de bas niveau avec le réseau ou codage des données dans un système hétérogène). Une étape initiale de ce projet a constité à évaluer une mise en œuvre de DCE en prenant en compte : les interactions des services avec les systèmes d'exploitation, l'évaluation aux limites dans les cas de saturation des services, en particulier. Cette phase s'est poursuivie vers une étude des besoins des utilisateurs en particulier dans le domaine du déploiement des applications distribuées :

A.12.4.2 Contexte du projet

- ◇ *Contexte :* CTI avec France Telecom et le CNET de Lannion
- ◇ *Partenaires académiques :* University Brown et University of Ann Harbor (Projet Pilgrim du CITI)
- ◇ *Partenaire industriel :* HP Research Labs, Bristol
- ◇ *Durée :* 1994–1996
- ◇ *Financement pour notre équipe :* 600 000 Francs de France Telecom et 300 000 Francs en dotation (matériel et logiciel) de HP France.

A.12.5 Projet *GNAT et programmation Ada distribuée*

A.12.5.1 Description succincte

Ce projet s'inscrit dans le cadre de coopérations entre l'Université de New York University (équipe de R. Dewar et E. Schonberg) et nos équipes du GET (ENST et ENST Bretagne). Lorsque l'équipe de NYU a été retenue pour le développement de compilateurs et d'environnement Ada, nous avons collaboré sur la partie distribution de la nouvelle norme du langage Ada. En effet, Ada 95 est le premier langage à inclure un modèle de distribution au niveau de la définition du langage, donc de façon portable et independante des systèmes d'exploitation ou des exécutifs sous-jacents.

Dans un premier temps, nous avons procédé à l'étude de la proposition de la norme Ada pour la partie distribuée tout en mettant en œuvre les concepts proposés. Cette étude conjointe nous a permis d'affiner le modèle et la proposition et interagir directement avec l'équipe de normalisation (équipe dirigée par Tucker Taft).

Puis, nous avons enfin défini l'architecture générale pour le support des applications Ada distribuées et proposé, mis en œuvre les extensions tant au niveau du compilateur que de l'exécutif. Nous avons ainsi été la première équipe à réaliser un tel environnement, qui sera ensuite développé et utilisé par de nombreux projets tant en recherche qu'en projets industriels. L'environnement GLADE est désormais commercialisé et maintenu par la société Ada Core.

Le projet GNAT, associé à sa large diffusion, et notre implantation initiale du modèle Ada distribué nous a ensuite permis d'explorer des domaines connexes et surtout d'enrichir le modèle initial. Nous avons en particulier proposé le langage GNATDIST qui permet de configurer une application distribuée, proposé et réalisé une extension pour le traitement des fautes et la reconfigurabilité.

A.12.5.2 Contexte du projet

- ◇ *Commanditaire :* New York University
- ◇ *Partenaires Académiques :* New York University - Texas A&M University
- ◇ *Partenaire industriel :* Computer Science Corporation et US DoD
- ◇ *Durée :* 1992-1998

A.12.6 Projet *Dassault Electronique*

A.12.6.1 Description succincte

Ce projet est directement lié aux travaux de thèse de Laurent Pautet dont le sujet de thèse a été présenté précédemment.

A.12.6.2 Contexte du projet

- ◇ *Durée :* 1991-1994
- ◇ *Partenaire Industriel :* Dassault Electronique
- ◇ *Montants pour notre équipe :* 200 000 Francs - bourse CIFRE.

A.12.7 Projet *RoDyReF*

Le projet RODYREF (Routage dynamique et répartition des Flux dans les réseaux de télécommunications) a été le point de départ des activités dans le domaine des réseaux.

Les travaux de recherche menés dans le cadre de ce projet ont pour objet l'application des études sur l'intelligence collective des insectes sociaux à des problèmes difficiles. Les algorithmes proposés démontrent en effet toute leur puissance dans le cas de problèmes dont l'énoncé, les données ou les paramètres varient dans le temps : dans de tels cas, un système à base d'agents distribués implantant l'algorithme s'adapte en trouvant dynamiquement, par

l'interaction des agents, de nouvelles solutions approchées. Le but de ce projet était donc d'appliquer ces approches et techniques pour la résolution d'un problème de routage dynamique dans des réseaux de télécommunication. Formellement, le problème consistait à déterminer des chemins (optimaux selon des critères divers de fiabilité, de coûts, de charge, ...) entre des sous-ensembles de nœuds qui doivent échanger des informations. L'accent était mis sur la garantie et la qualité de service et la meilleure utilisation du réseau.

A.12.7.1 Contexte du projet

- ◇ *Commanditaire :* Région Bretagne - Nortel Research
- ◇ *Partenaire Industriel :* Nortel Research (Harlow, GB)
- ◇ *Partenaires Académiques :* Université Libre de Bruxelles
- ◇ *Montants pour notre équipe :* 128 000 Francs (de la Région), 230 000 Francs (de Nortel Research) et 120 000 Francs en dotation matérielle (HP Research Labs, Bristol).

A.12.8 Projet *RAM QoS*

A.12.8.1 Description succincte

Le routage joue un rôle primordial et central pour les réseaux de télécommunications, et sa résolution est très liée au type de service qu'un réseau donné peut offrir. Plus le réseau se diversifie et offre des services de haut niveau, plus le routage doit être élaboré. Dans le cas des réseaux IP, les algorithmes utilisés sont ainsi adaptés à l'offre d'un service de type « best effort » mais ils ne correspondent pas à des solutions génériques conduisant à des algorithmes efficaces lorsque l'on cherche à étendre les fonctionnalités des réseaux. Or les technologies émergentes des télécommunications et le développement des nouveaux usages font apparaître un réel besoin de nouvelles solutions dans ce domaine. On peut ainsi citer les standards récemment apparus (RSVP, Diff-Serv, QOSPF) qui sont des extensions récentes aux algorithmes de routage classiques déjà anciens, et dont l'apparition reflète les nouveaux besoins tout en apportant une réponse souvent partielle aux problèmes posés.

Les objectifs du projet RAM QoS (Routage par agents mobiles - applications à la la qualité de service) sont de deux ordres. D'une part, cette étude vise à étudier les différentes propositions en cours dans le domaine du routage, à comprendre les évolutions en cours et futures et à prendre le recul nécessaire afin de mesurer leurs impacts. D'autre part, nous souhaitons proposer une démarche originale pour le routage dans des réseaux dynamiques avec la prise en compte de plusieurs critères pour l'évaluation de l'état (charge et statut, par exemple) de chaque lien et de chacun des équipements du réseau.

Ces travaux seront réalisés en tenant compte des spécificités et contraintes matérielles et logicielles des différents équipements de réseaux ; tout en tenant compte des domaines d'applications de ces nouvelles techniques. Nous avons en particulier retenu le routage et

l'équilibrage des charges dans les réseaux GSM, le routage dans les réseaux optiques et la gestion des VPN.

A.12.8.2 Contexte du projet

- ◇ *Contexte* : Contrat avec la Direction Scientifique de France Telecom
- ◇ *Partenaires en recherche :* France Telecom R&D
- ◇ *Durée :* 2000–2002
- ◇ *Financement :* 560 000 Francs de France Telecom

A.12.9 Projet *ELCAD* : Environnement Logiciel de Conception d'Applications Distribuées

A.12.9.1 Description succincte

Les applications logicielles pour les systèmes à longue durée de vie (spatial, transport, télécommunications, par exemple) sont complexes et leur conception difficile. Leur taille mais aussi les interactions existant entre les différents composants et les exigences de qualité imposent l'utilisation de méthodes et d'outils afin d'aider lors des différentes phases de conception de ces applications. Il s'agit ici de prendre en considération l'environnement informatique dans lequel l'application évoluera : systèmes multi-sites, multi-machines, caractéristiques des liaisons de communication.

Dans ce cadre, en 1987, l'Agence Spatiale Européenne a développé la méthode HOOD et l'utilise pour ses développements logiciels. TNI a développé l'outil HOOD afin d'offrir aux développeurs un environnement complet de développement d'applications logicielles.

La méthode et l'outil assistent le concepteur des applications mais ne permettent pas de prendre en compte les aspects distribution de l'exécution. Actuellement, cette étape doit être réalisée manuellement et rompt de ce fait la continuité systématique de la conception. Enfin, la plupart des organismes cités gèrent des projets pour lesquels les problèmes de répartition de code doivent être traités indépendamment de la conception logique de l'application. HOOD permet cette démarche, mais les concepts proposés actuellement par la méthode et les facilités implémentées dans STOOD sont insuffisants pour répondre efficacement à ce réel besoin.

Notre objectif est d'enrichir les fonctionnalités d'une méthode de génie logiciel adaptée à la conception de systèmes et de logiciels complexes, et de développer le logiciel interactif associé à la méthode. Nous souhaitons ainsi obtenir un outil complet qui puisse prendre en compte les caractéristiques de l'architecture cible et automatiser les différentes étapes liées à l'exécution de l'application sur une plate-forme embarquée.

A.12.9.2 Contexte du projet

- ◇ *Commanditaire :* Région Bretagne - Programme ITR (Information Télécom et Réseaux)
- ◇ *Partenaire Industriel :* TNI
- ◇ *Financement :* 680 000 Francs pour l'équipe de l'ENST Bretagne
- ◇ *Durée :* 1995–1997

A.12.10 Projet *COTRE*

A.12.10.1 Description succincte

L'objectif du projet est de définir une démarche outillée pragmatique de modélisation et de validation d'architecture de logiciels temps réel permettant de combiner les approches formelles et semi-formelles dans un processus industriel garantissant la continuité/traçabilité de la phase de conception jusqu'à celle d'implantation du logiciel sur la cible réelle. Cette démarche sera proposée comme candidate à une évolution des standards en la matière : nous ciblons plus particulièrement la méthode HOOD (largement reconnue par la communauté industrielle Européenne du logiciel temps réel) et UML mais aussi les travaux actuels de standardisation de langages de description d'architecture par le SAE (Avionics Architecture Description Language).

A.12.10.2 Contexte du projet

- ◇ *Commanditaire :* RNTL
- ◇ *Partenaire Industriel :* Airbus France et TNI
- ◇ *Partenaires Académiques :* ONERA, LAAS et IRIT
- ◇ *Finanement :* 105 000 euros pour l'ENST Bretagne
- ◇ *Durée :* 2001–2003
- ◇ *Adresse WEB :* http ://www.laas.fr/COTRE

ANNEXE B

Publications

B.1 Revues Nationales ou Internationales

[1] P. Dissaux, L. Pautet, Y. Kermarrec, and D. Lecampion. Communication and distribution tools for embedded distributed applications : a case study with Ada 95 and its distributed systems annex. *ACM Ada Letters*, 17(5) :40–44, 1997. Selected paper from the 8th International Real-Time Ada Workshop.

[2] M. Heusse, D. Snyers, and Y. Kermarrec. Adaptative routing in communication networks : comparison with classical methods. *Advances in Complex Systems*, 2(3) :209–219, 2000.

[3] Y. Kermarrec. Some experiments with Ada. *Ada User Journal*, 9(2) :79–82, 1988.

[4] Y. Kermarrec, L. Nana, and L. Pautet. Implementing an efficient fault tolerance mechanism in Ada 9X : an early experiment with GNAT, November 1994. Selected paper from the Ada Beligium conference - ACM / SIGAda and Université Libre de Bruxelles.

[5] Y. Kermarrec, L. Nana Tchamnda, M. Farine, and P. Dissaux. Report on workshop on methods and tools for ada 95. *Ada User Journal*, 21(1) :79–82, April 2000.

[6] R. Rannou and Y. Kermarrec. Le langage Occam. *Bigre + Globule*, (52) :25–66, Décembre 1986.

[7] O. K. Zein and Y. Kermarrec. An Approach For Discovering and Indexing Services for Self-Management in Autonomic Computing Systems. *Annals of Telecommunications (à paraitre dans numéro spécial consacré à l'Autonomic Computing)*, 2005.

B.2 Contributions à des ouvrages collectifs

[1] G. Coppin, M. Heusse, and Y. Kermarrec. *Routage par agents dans les réseaux de télécommunications*, pages 243–258. Hermés, Editeur R. Mandiau et al., 2002.

[2] D. Hérin and Y. Kermarrec. *WebServices et enseignement à distance*. CNRS - monographie - à paraître, 2005.

B.3 Actes de colloques édités

[1] Yvon Kermarrec, editor. *Workshop on software methods and tools for Ada 95*. ENST Bretagne, France, 1996.

[2] Yvon Kermarrec, editor. *Workshop on software methods and tools for Ada 95*. ENST Bretagne, France, 1997.

[3] Yvon Kermarrec, editor. *Workshop on software methods and tools for Ada 95*. ENST Bretagne, France, 1999.

[4] Yvon Kermarrec, editor. *Workshop on software methods and tools for Ada 95*. ENST Bretagne, France, 2001.

B.4 Actes de colloques nationaux ou internationaux avec comité de lecture

[1] J. Batlogg, R. Braek, C. Fowker, L. Gutierrez, Y. Kermarrec, L. Sacks, and J Wetteling. CANDLE : an European e-Education project to improve teaching on the internet. l'Aquila- Roma, Italy, August 2000. International Conference on Advances in Infrastructure for Electronic Business, Science and Education on the Internet (SSGRR 2000).

[2] A. Beugnard, Z. Choukair, and Y. Kermarrec. COREMO : a CORBA real time extension model and its Ada 95 implementation. In *Proceedings of the conference on TRI-Ada '96*, pages 255–268, Philadelphia, Pennsylvania, United States, 1996. ACM Press.

[3] Z. Choukair and Y. Kermarrec. Mémoire virtuelle partagée distribuée pour des machines parallèles à base de Transputers. In *les actes de la conférence RenPar'6*, ENS Lyon, 1994.

[4] P. Dissaux, L. Pautet, L. Bjornfot, Y. Kermarrec, and D. LeCampion. Communication and distribution tools for embedded distributed applications : a case study with Ada 95 and its distributed systems annex. In *Proceedings of the eighth international workshop on Real-Time Ada*, pages 40–44, Ravenscar, United Kingdom, 1997. ACM Press.

[5] A. Gargaro, Y. Kermarrec, L. Pautet, and S. Tardieu. PARIS : Partitionned Ada for Remotely Invoked Services. In Eurospace, editor, *Ada Europe Conference, Franckfurt Germany*, Heidelberg, October 1995. CNES and European Space Agency, Lectures Notes in Computer Sciences.

[6] M. Heusse and Y. Kermarrec. A new routing policy for load balancing in communication networks. In *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, pages 267–272, Beyrouth, 26–29 juin 2001.

[7] Y. Kermarrec. One experience with Ada : a transputer network simulator. In *Proceedings of the conference on CONPAR 88*, pages 306–313, UMIST, Manchester, United Kingdom, 1989. Cambridge University Press.

[8] Y. Kermarrec. La programmation d'applications distribuées avec Ada. In *Actes des journées Objet -organisée par l'IRISA*, ENST Bretagne, May 1997.

[9] Y. Kermarrec. CORBA vs. Ada 95 DSA : a programmer's view. In *Proceedings of the 1999 annual ACM SIGAda international conference on Ada*, pages 39–46, Redondo Beach, California, United States, 1999. ACM Press.

[10] Y. Kermarrec and Z. Choukair. Distributed object oriented programming for Ada 95 with CORBA. In *Proceedings of the 2nd Plenary Workshop of the HP OpenView University Association*, Munich, 1995.

[11] Y. Kermarrec and Z. Choukair. Distributed object oriented programming and interoperability for Ada 95 : an OMG CORBA approach. In Marcel Toussaint, editor, *Ada in Europe, Second International Eurospace - Ada-Europe Symposium, Frankfurt/Main, Germany, October 2-6, 1995, Proceedings*, volume 1031 of *Lecture Notes in Computer Science*, pages 217–227. Springer, 1996.

[12] Y. Kermarrec and P. Dissaux. Méthodes et outils pour applications embarquées et distribuées. In *Proceedings of Real Time Systems'97 - Salon des solutions informatiques pour les systèmes temps réels*, pages 131–143, Paris, France, 1997.

[13] Y. Kermarrec and Y. Gourhant. Evaluation de l'interaction de services de OSF DCE : le cas de la saturation de serveurs. In *Colloque International sur les nouvelles technologies de la répartition - NOTERE*, Pau, November 1997. Université de Pau, Elf, France Telecom et EDF.

[14] Y. Kermarrec and M. Heusse. CAF : cooperative agents for routing in telecommunication networks. In *Proceedings of the 5th Plenary Workshop of the HP OpenView University Association*, Bologna, Italy, June 1999.

[15] Y. Kermarrec and M. Heusse. A new approach for routing in networks. In *Proceedings of the 5th Plenary Workshop of the HP OpenView University Association*, Bologna, Italy, June 1999.

[16] Y. Kermarrec and M. Heusse. Adaptive routing and load balancing of ephemeral connections. In *Proceedings of the 1st IEEE European Conference on Universal Multiservice Networks ECUMN'2000*, pages 100–108, Colmar, France, October 2000.

[17] Y. Kermarrec, M. Heusse, and M. Rolander. Agent based routing over an active network. In *Proceedings of the 7th Plenary Workshop of the HP OpenView University Association*, Santorini, Greece, June 2000.

[18] Y. Kermarrec and L. Nana. Intégration des blocs de recouvrement dans les langages de haut niveau. In *Actes des Journées Doctorales Informatique et Réseaux (JDIR'96)*, ENST, Paris, September 1996.

[19] Y. Kermarrec, L. Nana, and L. Pautet. Implementing an efficient fault tolerance mechanism in Ada 9X : an early experiment with GNAT. In *Ada Belgium conference in Brussells*, printed in the Ada-Belgium Newsletter, Volume 3, Numbers 2+3, Winter 1994, November 1994. In co-operation with ACM/SIGAda and Université Libre de Bruxelles.

[20] Y. Kermarrec, L. Nana, and L. Pautet. Implementing recovery blocks in GNAT : a powerful fault tolerance mechanism and transaction support. In *Proceedings of TRIAda'95*, pages 462–466, Anaheim CA, USA, November 1995. ACM Press.

[21] Y. Kermarrec, L. Nana, and L. Pautet. GNATDIST : A configuration language for distributed Ada 95 applications. In *Proceedings of the conference on TRI-Ada '96*, pages 63–72, Philadelphia, Pennsylvania, United States, 1996. ACM Press.

[22] Y. Kermarrec, L. Nana, and L. Pautet. Providing fault-tolerant services to distributed Ada 95 applications. In *Proceedings of the conference on TRI-Ada '96*, pages 39–47, Philadelphia, Pennsylvania, United States, 1996. ACM Press.

[23] Y. Kermarrec, L. Nana, and R. Rannou. Reconsidering our switch from Ada 83 to C++, experience in system and real-time programming. In *10 th Annual ASSET Symposium*, pages 7–18, Embry-Riddle Aeronotical University, Prescott, Arizona, June 1996.

[24] Y. Kermarrec and L. Nana Tchamnda. Integrating recovery blocks in high level language. In A. Dahbura, editor, *IEEE International workshop on embedded fault-tolerant systems*, pages 355–362, Dallas, Texas, September 1996. IEEE Computer Society (technical comittee on fault tolerant computing) and IFIP WG 10.4 on dependable computing.

[25] Y. Kermarrec and L. Pautet. Ada communication components for distributed and real time applications. In *Proceedings of TRI-Ada'92*, pages 530–536, Orlando, Florida, November 1992. ACM SIGAda, ACM Press.

[26] Y. Kermarrec and L. Pautet. Outils de communication avec ExTRA. In *Proceedings of Ada-France'92*, Paris, France, October 1992. AFCET.

[27] Y. Kermarrec and L. Pautet. Composants logiciels de communication pour applications distribuées temps réel écrites en Ada. In *Proceedings of the Real Time Systems conference*, Paris, France, January 1993. BIRP, Neuf associés.

[28] Y. Kermarrec and L. Pautet. A distributed shared virtual memory for Ada 83 and Ada 9X applications. In Charles B. Engle, Jr., editor, *Proceedings of TRI-Ada'93*, pages 242–252, Seattle, WA, USA, September 1993. ACM Press.

[29] Y. Kermarrec and L. Pautet. Ada-Linda : A powerful paradigm for programming distributed Ada applications. In *Proceedings of TRI-Ada'94*, pages 438–445, Baltimore, Maryland, October 1994. ACM Press.

[30] Y. Kermarrec and L. Pautet. Ada reusable software components for education in distributed systems and applications. In J.L. Diaz-Herrera, editor, *Proceedings of the 7th Software Engineering Institute conference on Software Engineering Education*, Lectures Notes in Computer Science, pages 77–96, San Antonio, Texas, USA, January 1994. Springer Verlag.

[31] Y. Kermarrec and L. Pautet. Implementing the distributed features of Ada 9X with PVM. In *EuroPVM conference*, Rome, Italy, October 1994. ENS Lyon, Universita di Roma and IBM.

[32] Y. Kermarrec and L. Pautet. Integrating page replacement in a distributed shared virtual memory. In *Proceedings of the 14th international conference on distributed computing systems*, pages 355–362, Poznan, Poland, June 1994. IEEE Computer Society Press.

[33] Y. Kermarrec and L. Pautet. Programming distributed systems with both Ada 95 and PVM. In M. Toussaint, editor, *Ada in Europe*, pages 206–216. Springer, Berlin, 1995.

[34] Y. Kermarrec and L. Pautet. Using the distributed system annex in the curriculum : experiences with GLADE. In *proceedings of the 14th Washington Ada Symposium (WAdaS) ACM conference*, Washington, DC, June 1997. ACM Press.

[35] Y. Kermarrec, L. Pautet, and S. Tardieu. GARLIC : Generic Ada Reusable Library for Interpartition Communication. In *Proceedings of TRI-Ada'95*, pages 263–269, Anaheim, California, USA, November 1995. ACM Press.

[36] Y. Kermarrec and R. Rannou. Simulation de systèmes distribués avec Ada. In *Actes des journées Ada AFCET, le parallélisme en Ada*, pages 62–84, ENST Paris, December 1987.

[37] Y. Kermarrec and R. Rannou. Composants logiciels pour la simulation. In *proceedings Ada Belgium conference*, December 1988.

[38] Y. Kermarrec and R. Rannou. Simulation experiments with Ada. In *proceedings of the 19th modeling and simulation conference, IEEE and SCS conferences*, University of Pensylvania, Pittsburgh, May 1988.

[39] Y. Kermarrec and R. Rannou. A transputer network simulator. In T. Muntean, editor, *Parallel Programming of Transputer Based Machines*, pages 281–296, Amsterdam, NL, 1988. IOS Press.

[40] Y. Kermarrec and I. Simpson. TIC et international à l'ENST Bretagne. In *Actes des journées d'études internationales : les technologies de l'information et de la communication au service des relations internationales des établissements d'enseignement supérieur et de recherche*, ENST Bretagne, November 1999.

[41] Y. Kermarrec and A. Soleto. Managing document consistency over the web or managing documents duplication. In *Proceedings of the 4th Plenary Workshop of the HP OpenView University Association*, Madrid, April 1997.

[42] Y. Kermarrec and R. Sancho Villa. Compilation analysis of parallel Occam programs : Enforcing determinacy and communication correctness. In Andrew Veronis, E. Shore, and Yakup Paker, editors, *Transputer Research and Applications 5*, pages 134–148, Baltimore, USA, 1992. IOS Press.

[43] O. Zein and Y. Kermarrec. An elaborate and flexible trader based on ontologies. In *IFIP WG 6.7 and Eunice Summer School on Adaptable Networks and Teleservices*, pages 103–108, Trondheim, 2–4 septembre 2002.

[44] O. K. Zein and Y. Kermarrec. An Approach for Describing and Querying Service Behavior in Distributed Systems. In *6èmes Journées Doctorales Informatique et Réseau*, Lannion, France Télécom R&D, 2-4 novembre 2004.

[45] O. K. Zein and Y. Kermarrec. An Approach for Describing and Querying Service Behavior in Distributed Systems. In *The IEEE international Symposium on Applications and the Internet*, Trento, Italy, January 2005.

[46] O. K. Zein and Y. Kermarrec. An Approach for Service Description and a Flexible Way to Discover Services in Distributed Systems. In *IEEE International Conference on Information Technology*, Las Vegas, April 2005.

[47] O. Kassem Zein and Y. Kermarrec. An approach for describing user service interfaces in distributed systems. In *in the proceedings of PDPTA conference*, pages 1441–1446, LAS VEGAS, Nevada, June 2003.

[48] O. Kassem Zein and Y. Kermarrec. An approach for dynamic composition of services in distributed systems. pages 57–61, Budapest, Hungary, September 2003. IFIP/IEEE Workshop on Next Generation Networks.

[49] O. Kassem Zein and Y. Kermarrec. An approach for describing and querying service dehavior in distributed systems. Banff, Canada, July 2004. IASTED International conference on Communication Systems and Applications.

[50] O. Kassem Zein and Y. Kermarrec. An approach for describing, discovering services and for adapting them to the needs of users in distributed systems. In K Sycarra and T Payne, editors, *In the proceedings of AAAI Spring Symposium on Semantic Web Services*, Stanford, California, March 2004.

[51] O. Kassem Zein and Y. Kermarrec. A metadata model for service description and user profile and facilities of service retrieving in distributed systems. Tampere, Finlande, June 2004. IFIP/IEEE Workshop on Advances in fixed and mobile networks.

B.5 Publications sous format électronique

[1] D.Feneuille, JP Rosen, S. Tardieu, S. Rivière, and Y. Kermarrec. Enseigner ada / choisir un langage entre le tentant et le raisonnable. URL http ://d.feneuille.free.fr/enseignerada.htm, December 2003.

[2] Y. Kermarrec. Active networks : state of the art and perspectives with the RNTL AMARRAGE. In *les journées Télécom - comité Richelieu*, ENST, Brest, 10–11 May 2000.

[3] Y. Kermarrec. Distributed systems : new directions and research issues. In *First Franco-Chinese workshop on computing tools to increase navigation security*, Maritime University of Shanghai, 3–6 June 2002.

B.6 Séminaires

[1] A. Gargaro, Y. Kermarrec, L. Pautet, and R. Volz. ADEPT : Ada Distributed Execution and Partionning Tools. Tutorial, ACM Conference, Anaheim, California, November 1995.

[2] Y. Kermarrec. Parallel programming : models, languages and methodologies. Actes de l'Ecole Erasmus organisée pour les étudiants en Master de Delft, Aachen, Louvain, Leuven, Imperial, Mines Paris, ENST Paris, March 1992.

[3] Y. Kermarrec. Distributed systems programming. Semaine de cours et conférences à Poznan, Programme Tempus - étudiants du master télécommunications et doctorants - Professeur Jerzy Brzenzinski, March 1995.

[4] Y. Kermarrec. Ada 95 et les systèmes distribués. Séminaire Thomson Brest, April 1996.

[5] Y. Kermarrec. Distributed computing environment : architecture and issues. Semaine de cours à Poznan, Programme Tempus - Professeur Jerzy Brzenzinski, March 1996.

[6] Y. Kermarrec. Distributed systems programming with Ada 95. Cours en mastère Informatique Université d'Uppsala - invitation du Professeur Lars Asplund, May 1996.

[7] Y. Kermarrec. GNAT DIST : dealing with distributed systems with Ada. Séminaire - Université d'Uppsala - departmenet of computer systems (DOCS) - Professeur Lars Asplund, May 1997.

[8] Y. Kermarrec. cycle de conférences autour d'Ada 95. Séminaires Raytheon et MC Donald-Dettwiller Associates, Vancouver - Dr C Thompson, 1998.

[9] Y. Kermarrec. "CORBA and real time systems". Séminaire Raytheon, Vancouver - Dr C Thompson, March 1999.

[10] Y. Kermarrec. Distributed execution of Ada programming. Séminaire Boeing Defense, Seattle - Dr S. Moody, March 1999.

[11] Y. Kermarrec. Routing algorithms for telecommunication networks. Seminar - Université de Poznan - invitation du Professeur Jerzy Brzenzinski, June 2001.

[12] Y. Kermarrec. Cycle de conférences sur mes travaux en r&d. Université Libanaise à Beyrouth - invitation du doyen le Professeur M. Zoaeter, October 2004.

[13] Y. Kermarrec and L. Pautet. programming distributed systems with Ada 95 and an inside look at GNATDIST. In *Tutorials of the international conference on reliable technologies - Ada Europe 97*, London, UK, June 1997.

[14] Y. Kermarrec, L. Pautet, and S. Tardieu. Ada 95 and distributed systems. In *Tutorials of TriAda'97*, St-Louis, USA, November 1997. ACM Press.

[15] Y. Kermarrec and A. Skryniarz. Systèmes multi-agents : thématique et architecture. Séminaire Thomson, Brest (pour le projet SMA2), March 2000.

B.7 Rapports techniques

[1] M. Beggaz, A. Bouabdallah, M. Gontas, Y. Kermarrec, and A. Schach. Méta modèle de COTRE. Deliverable F211, RNTL project COTRE, Consortium COTRE, August 2003.

[2] P. Dissaux, S. Guérin, M. Heusse, Y. Kermarrec, and D. Snyers. RODYREF : Routage dynamique et répartition des flux dans les réseaux de télécommunications. Rapport de fin de contrat ITR Région Bretagne, Télécom Bretagne, July 1999.

[3] P. Dissaux, Y. Kermarrec, and D. Le Campion. Environnement logiciel de conception d'applications distribuées. Technical report, ENST de Bretagne, December 1996. Rapport de fin de contrat ITR Bretagne - Projet ELCAD.

[4] S. Garlatti, Y. Kermarrec, and C. Ragnard. Final version of the author's workbench. Deliverable 4.4 of IST CANDLE project, Candle Consortium, May 2003.

[5] S. Garlatti, Y. Kermarrec, and C. Ragnard. Final version of the reference user client. Deliverable 4.5 of IST CANDLE project, Candle Consortium, May 2003.

[6] S. Garlatti, Y. Kermarrec, and Ph. Tanguy. First version of the reference user client. Deliverable 4.2 of IST CANDLE project, Candle Consortium, September 2001.

[7] S. Garlatti, Y. Kermarrec, and E. Wiederhold. Specifications and requirements for the authoring, indexing and navigation tools of IST CANDLE. Technical Report deliverable 4.1, CANDLE consortium, 2000.

[8] S. Garlatti, Y. Kermarrec, and P. Wrona. Second prototype of the information broker applications. Deliverable 4.3 of IST CANDLE project, Candle Consortium, December 2002.

[9] Y. Kermarec and L. Pautet. Termination detection of distributed ada 9X applications. Unpublished note - Available on request from the authors, 1994.

[10] Y. Kermarrec. Evaluation de OSF DCE en cas de saturation des serveurs. Rapport de fin de contrat CNET DEST, Télécom Bretagne, July 1996.

[11] Y. Kermarrec and M. Heusse. Le routage dans les réseaux de communication. Rapport d'avancement du projet RamQos - France Telecom R&D Lannion - CTI AB431, Télécom Bretagne, July 2001.

[12] Y. Kermarrec, M. Heusse, and M. Rollander. Routage par agents pour les réseaux actifs. Rapport final du projet AMARRAGE, Télécom Bretagne, July 2001.

[13] Y. Kermarrec and L. Pautet. Evaluation of the distributed systems annex of Ada 9X and its implementation in GNAT. Technical report, Courant Institute of Mathematical Sciences, New York University, 715 Broadway, New York NY 10012, 1994.

[14] Yvon Kermarrec, Laurent Pautet, and Ed Schonberg. Design Document for the Implementation of Distributed System Annex of Ada 9X in GNAT. Technical report, New York University, Courant Institute, 715 Broadway, New York NY 10012, March 1995.

[15] R. Volz, R. Thierault, Y. Kermarrec, L. Pautet, S. Tardieu, and G. Smith. Ada 95 distribution annex implementation for GNAT. Technical report, Texas A&M University, College Station, Texas, November 1995.

B.8 Thèses encadrées ou dirigées

[1] Zied Choukair. *Inter-opérabilité des objets distribués*. PhD thesis, Université d'Orsay, 1997.

[2] Martin. Heusse. *Routage et équilibrage de charge par agents dans les réseaux de communication*. PhD thesis, EHESS et ENST-Bretagne, 2001.

[3] Laurent Pautet. *Conception et réalisation de composants logiciels pour applications distribuées temps réel*. Thèse de doctorat, École nationale supérieure des télécommunications, January 1994.

[4] Laurent Nana Tchammnda. *Ada 95 et les systèmes distribués*. PhD thesis, Université de Rennes I, 1997.

[5] O. Kassem Zein. *Indexation, recherche et compositions de services distribués*. PhD thesis, UBS Université de Vannes et ENST-Bretagne, 2004.