

PROJET RNTL ARCAD*

D3.4 et D4.3 - Démonstration[†]

T0+36

Coordonateur : Michel RIVEILL
Université de Nice / ESSI
930 route des Colles
06903 Sophia Antipolis Cedex

Auteurs : M. Riveill, F. Baude, E. Bruneton, P-C. David,
D. Emsellem, T. Ledoux, M. Morel

8 août 2004

Résumé

Ce document présente brièvement les différentes démonstrations qui ont été réalisées dans le cadre du RNTL Arcad par les différents partenaires du projet. L'objectif de ces démonstrations est de mettre en avant la nécessité d'avoir des plates-formes adaptables et de montrer l'étendue des solutions permettant d'atteindre cet objectif.

1 Introduction

Ce document a pour objectif de décrire quelques démonstrations réalisées par les différents partenaires du projet. Nous rappelons que l'objectif du projet ARCAD était de s'intéresser à l'adaptabilité des plates-formes à composants et par conséquent de mettre en évidence la définition de mécanismes de gestion qui permettent aux applications de choisir des composants techniques en fonctions des besoins spécifiques et de leurs environnements d'exécution. Cet objectif global peut être décomposé de la manière suivante :

*Le projet ARCAD (Architecture Extensible pour Composants Adaptables) a été labellisé en décembre 2000. Les différents partenaires sont France Télécom R&D (équipe ASR - Thierry.Coupage@francetelecom.com), INRIA (projet Oasis - Denis.Caromel@inria.fr, projet Sardes - Daniel.Hagimont@inria.fr), Ecole des Mines de Nantes (équipe OCM - Thomas.Ledoux@emn.fr), le laboratoire I3S commun à l'Université de Nice - Sophia Antipolis et au CNRS (projet Rainbow - Anne-Marie.Pinna@unice.fr). Le coordonateur du projet est Michel.Riveill@unice.fr

[†]Ce livrable intègre les deux livrables initialement prévus : D3.4 - Démonstration complète (T0+21 : interne) et D4.3 - Démonstration mettant en évidence les capacités de la plate-forme (T0+30 : interne - T0+36 : public)

- *Réutilisation du code métier des applications.* Nous voulons fournir des mécanismes qui permettent de configurer les composants techniques utilisés par une application sans modifier son code métier. En effet, la diversification des plates-formes d'exécution, amenée par le développement important des technologies et des réseaux, fait apparaître le besoin d'exécuter les mêmes applications dans différents environnements. La possibilité de réutilisation permettrait d'exécuter les applications dans différents environnements en configurant les composants techniques utilisés sans un processus coûteux de réimplémentation.
- *Réutilisation du code des composants techniques.* Nous voulons fournir des mécanismes qui permettent de réutiliser, sans modification, pour les besoins de différentes applications, le code des composants existants. En effet, la diversification des plates-formes d'exécution fait apparaître des besoins non seulement au niveau des applications mais également au niveau des composants techniques : ils doivent pouvoir être utilisés dans un nombre croissant d'environnements différents. Une réutilisation de leur code existant permettrait de facilement mettre en place le même schéma de gestion dans différents domaines d'application.

2 Adaptation d'une application multimédia répartie

Cette section décrit des scénarios d'adaptation dynamique dans une application multimédia répartie. Rappelons que notre principal objectif consistait à fournir un support permettant de construire des applications multimédia adaptables à leurs environnements d'exécution. Nous avons présenté dans [ref D2.3] un canevas logiciel composé principalement de :

- Un langage de spécification permettant de décrire la structure d'une application multimédia ainsi que les règles de reconfiguration garantissant son évolution correcte face à celles de l'environnement.
- Une plateforme à composant basée sur le modèle Fractal permettant l'automatisation du déploiement des applications et de leurs reconfigurations en cours d'exécution.

Le type d'application auquel nous nous intéressons est basé sur l'utilisation de nœuds intermédiaires (ou proxys) visant à adapter les flux multimédia pour les capacités matérielles et logicielles des terminaux. Nous décrivons dans ce qui suit les expérimentations réalisées dans ce cadre.

L'environnement matériel utilisé comporte des stations de travail et des terminaux mobiles (PDA) reliés par un réseau sans fil. Divers services multimédia sont mis à disposition, dont un serveur Web jouant le rôle d'un serveur de vidéo à la demande en mettant à disposition divers clips vidéo et un serveur d'audio/vidéo conférence multi-usagers. Les scénarios d'adaptation

proposées sont définis comme suit :

Adaptation dans une application de vidéo à la demande : Dans un premier scénario, un utilisateur accède à une vidéo stockée sur le serveur de vidéo à la demande. La vidéo est encodée en MPEG2 avec une résolution de 640*480. L'utilisateur en question est muni d'un PDA ayant de faibles capacités de calcul (processeur à 200 Mhz et 32 Mo de mémoire) et d'affichage (écran de 320*240). Il utilise une application acceptant seulement le format d'encodage MPEG-1. Pour ce faire, la requête du client est envoyée vers un proxy HTTP, placé entre le serveur et le client. Ce dernier se charge ensuite de créer dynamiquement le processus d'adaptation adéquat afin de fournir un contenu adapté à cet utilisateur. Le traitement effectué, décrit par la figure 1, consiste à décoder la vidéo originale, réduire sa résolution et à ré-encoder les données sous le format MPEG-1 avec un débit moins élevé.

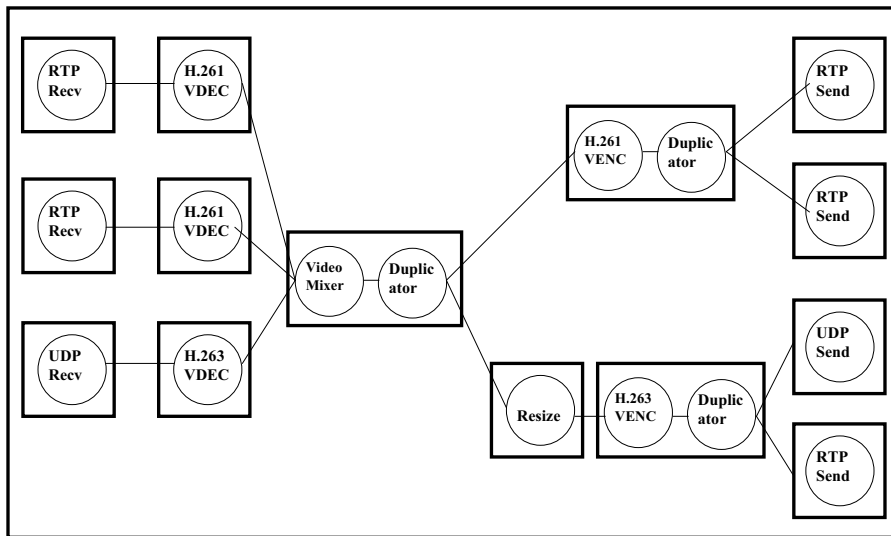


FIG. 1 – Configuration sur un proxy d'adaptation

Adaptation dans une application de vidéoconférence multi-utilisateurs : Le deuxième scénario d'application consiste en une application de vidéoconférence entre quatre participants hétérogènes, utilisant des applications point-à-point (pouvant visualiser un seul flux à la fois). Les deux premiers envoient (et reçoivent) des flux RTP vidéo, encodée sous le format H.263 et une résolution QCIF. Le troisième utilise le

format d'encodage H.261 et UDP comme protocole de transport. Un quatrième participant, muni d'un PDA, souhaite seulement visualiser les vidéos des autres utilisateurs. Le serveur de vidéoconférence, dont la configuration est illustrée dans la figure 2, agit comme une entité centrale dédiée à la coordination des communications entre les divers participants. Il est également responsable de fournir un contenu adapté à chacun d'entre eux. Pour cela, le serveur reçoit les données de chaque participant, effectue un mixage des divers flux vidéo en un seul flux (une matrice 4x4), et envoie le résultat pour chacun des participants.

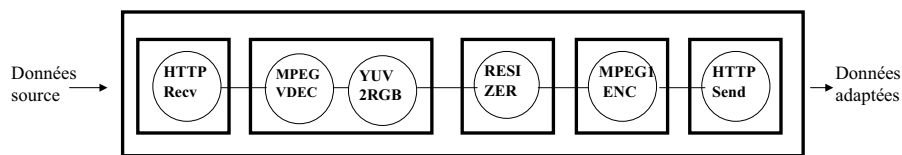


FIG. 2 – Configuration d'un serveur de vidéoconférence

3 Composants répartis Fractal-ProActive

L'objectif de cette description est de démontrer comment des composants Fractal-ProActive sont définis et mis en œuvre dans un cadre réparti (typiquement, de *grille de calcul*). Dans un premier temps, il s'agit d'explicitier la méthode à suivre, de manière la plus pédagogique possible, afin de développer et déployer des composants répartis. Dans un second temps, on souhaite démontrer l'applicabilité d'un tel modèle de programmation des grilles, sur une application scientifique d'envergure, déployée sur clusters de 32/64 processeurs, et/ou sur grille intranet (les 300 machines de bureau de l'UR de l'INRIA Sophia-Antipolis); voir <http://www.inria.fr/oasis/proactive/jem3D.html>.

3.1 Méthodologie

La méthodologie pour développer des composants Fractal-Proactive est détaillée dans les tutoriaux Fractal. L'url ci-après donne une présentation générale de l'implantation de Fractal, effectuée dans le cadre de ProActive : objectifs, fonctionnalités et utilisation. <http://www-sop.inria.fr/oasis/proactive/doc/api/org/objectweb/proactive/doc-files/components/intro.html>

L'API de programmation des composants est conforme à l'API Fractal selon le niveau de conformance 3.2. Ceci signifie que toutes les notions de Fractal sont disponibles, à l'exception des templates.

Pour cibler le contexte du calcul distribué, seules deux extensions ont été apportées au modèle Fractal :

- le déploiement distribué : à la création d'un composant, on peut indiquer comme paramètre supplémentaire un nœud virtuel qui hébergera le composant ; ce paramètre peut être explicitement indiqué dans le code source dans le cas où les composants sont créés par programmation ; sinon, le nœud virtuel peut être indiqué comme un attribut, dans le fichier ADL de l'application, qui décrit les composants et leurs liaisons ; un fichier de déploiement associé à l'application permet de spécifier où, et de quelle manière démarrer les JVMs correspondant aux nœuds virtuels requis par l'application ;
- les composants parallèles : ce type de composant est une spécialisation de la notion de composant composite. La particularité d'un composant parallèle est qu'il encapsule d'autres composants du même type ; les appels entrants sont retransmis aux interfaces internes correspondantes, sur les composants inclus. L'ADL Fractal a été étendu de manière à introduire un nouvel élément *parallel-composite-component*. De même, par programmation, on peut instancier des composants parallèles.

Les objets actifs qui matérialisent les composants sont automatiquement instanciés lors de la création des composants dans l'implémentation des méthodes qui permettent de créer les composants. Pour les composants primitifs, l'objet actif instancié est de la classe indiquée pour l'implantation du composant. Ici, la seule différence avec Julia consiste donc à créer un objet actif ProActive plutôt qu'un objet standard (i.e. enclencher un `ProActive.newActive(...)` plutôt qu'un simple `new`).

Consulter <http://www-sop.inria.fr/oasis/proactive/doc/api/org/objectweb/proactive/doc-files/components/> pour des exemples de fichiers de description d'applications à base de composants. En fait, ce sont des fichiers ADL Fractal standard.

Une fois les composants instanciés, on profite de tout ce que Fractal offre, principalement,

- reconfiguration des liaisons
- modification des inclusions

Et on profite bien sûr de tout ce que ProActive offre, et tout particulièrement :

- appel asynchrone avec futur pour les invocations de services sur les composants, que ce soit point-à-point ou multipoint
- visualisation, pilotage, migration des objets actifs via l'outil IC2D (voir figure 3)

3.2 Jem3D

Il s'agit d'une application de simulation numérique de propagation d'ondes électromagnétiques en 3D, se fondant sur la résolution d'équations de Max-

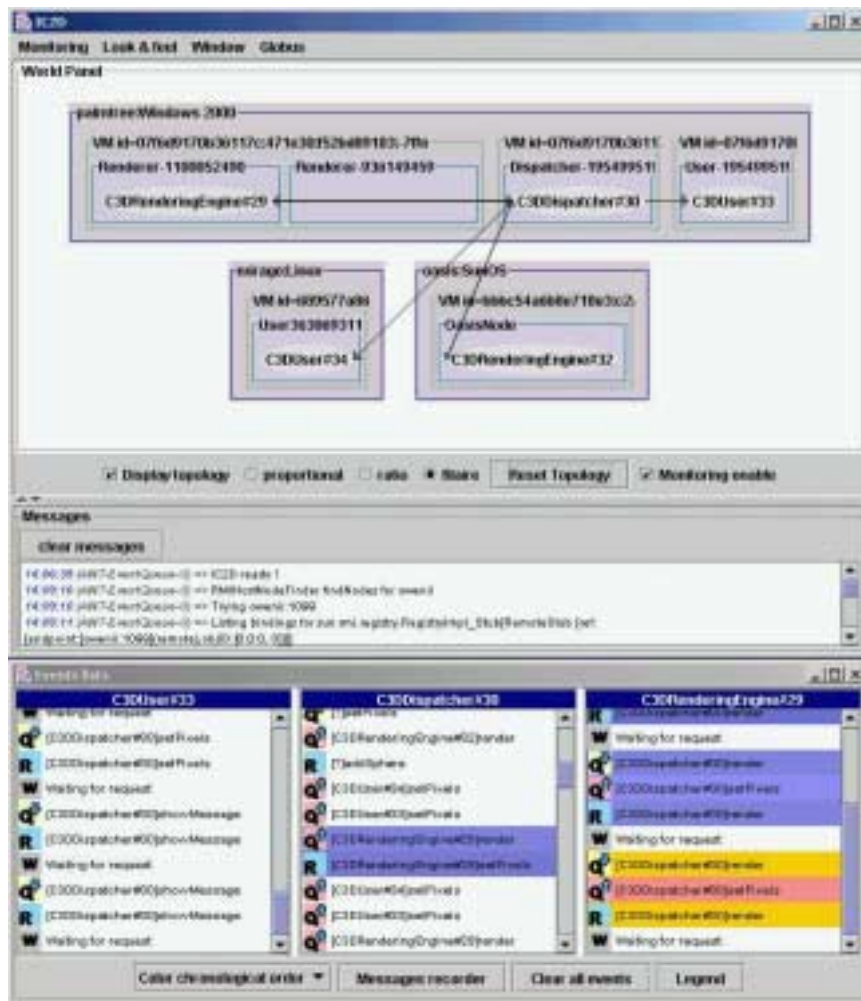


FIG. 3 – Visualisation et pilotage des objets actifs associés aux composants

well [BBC⁺04]. La version démontrée consiste déjà en une véritable évolution vis-à-vis de la version existante en Fortran 77 (EM3D) et parallélisée pour plus de performances, via l'utilisation de MPI, selon un modèle SPMD.

En effet la nouvelle version définit en fait une bibliothèque générique orienté objet, qui permet de définir les méthodes de simulation en se basant sur une décomposition du domaine physique centrée élément ou bien, centrée sommet. De cette bibliothèque générique (Jem3D), on a instancié une version qui consiste à décomposer le domaine physique en volumes tétraédriques centrés élément, et à calculer la propagation des ondes pour chaque pas de temps, via les facettes triangulaires de ces volumes. Pour passer à l'échelle et gagner en performances, on a ensuite artificiellement organisé l'ensemble des tétraèdres constituant le domaine en sous-domaines. Ainsi, des tétra-

hêtres voisins se retrouvent matériellement répartis. Les facettes qu'ils ont en commun et qui matérialisent ces voisinages sont donc sujettes à des accès à distance.

Une telle application est donc intéressante car elle génère un parallélisme relativement important, ainsi qu'un nombre élevé de communications entre sous-domaines, dûes au nombre proportionnel de facettes frontières partagées. Les communications de groupe disponibles en ProActive sont un bon moyen pour simplifier la coordination et les échanges entre les divers objets actifs qui matérialisent les sous-domaines issus de la répartition. Par ailleurs sur un sous-domaine donné, l'ensemble des facettes partagées avec d'autres sous-domaines se prête bien à être représenté comme un groupe d'objets, facilitant ainsi la mise à jour de cet ensemble de facettes avant de passer à la simulation de chaque prochain pas de temps.

Du point de vue composant, cette application est intéressante car elle est composée de parties bien distinctes :

- le solveur lui-même, qui comme on l'a explicité ci-dessus se modélise bien par un composant parallèle ;
- un collecteur, qui pour des raisons de passage à l'échelle, pourrait être étendu en un collecteur hiérarchique, faisant appel à plusieurs collecteurs primitifs, qui, de manière orchestrée pourraient faire remonter l'information calculée à un collecteur maître ;
- une interface graphique et collaborative permettant de brancher plusieurs clients (voir figure 4). Cette interface récolte les résultats auprès du collecteur afin de les présenter en temps le plus proche du réel possible ; elle permet également au(x) client(s) de contrôler le déroulement de la simulation (par exemple, en décidant de zoomer sur telle ou telle partie plus précise du domaine). On a pu tester de la visualisation "classique" rendant les résultats via du Java3D ou du VTK ; on a aussi pu changer le composant interface graphique afin de visualiser les calculs en stéréo 3D immersive (sur le workbench de l'UR INRIA Sophia-Antipolis).

4 Assemblage dynamique de composants logiciels à l'aide des interactions logicielles

L'objectif de cette description est de donner une vue intuitive du service d'interaction, de mettre en évidence la souplesse de son utilisation dans le cadre d'une application simplifiée d'agenda. Ce support nous permettra d'illustrer l'assemblage dynamique de composants logiciel et de services techniques.

Par souci de simplicité, nous décrirons la version RMI de cette application sachant que la même application existe dans sa version EJB utilisant la plateforme JOnAS d'ObjectWeb.

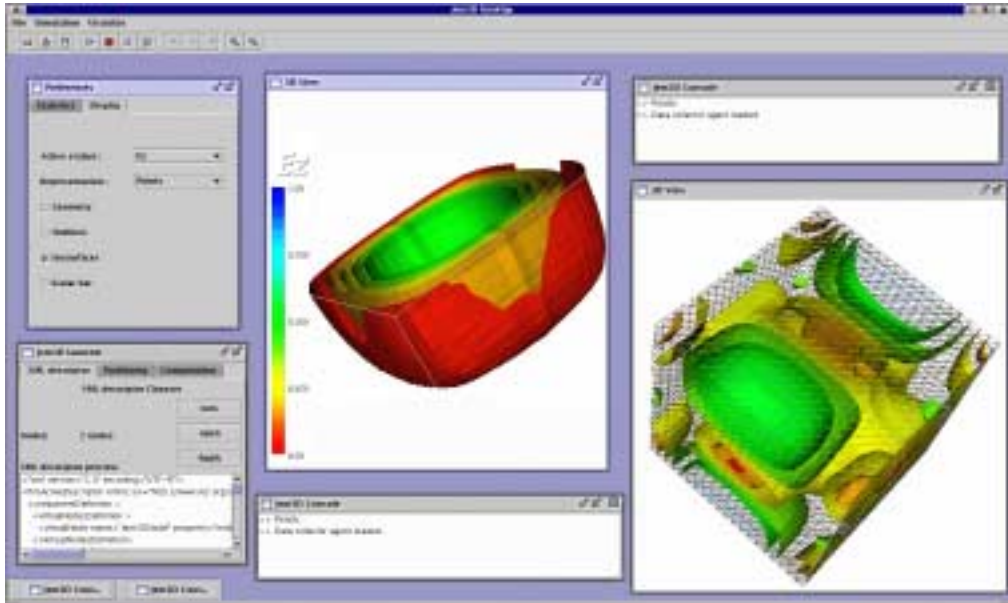


FIG. 4 – Composant d'interface graphique pour l'application Jem3D

Nous ferons l'hypothèse qu'il existe une implémentation RMI d'un composant Display qui permet l'affichage de message, d'un authentificateur qui permet de valider les appels sur un composant, d'un composant simplifié DataBase qui permet de mémoriser les agendas dans une table de hachage en fonction du nom de l'utilisateur de l'agenda et une implémentation RMI d'un composant Agenda.

En cours d'exécution, les instances de ces différents composants sont liées et déliées par des interactions pour s'adapter au contexte d'exécution. Ainsi certaines instances du composant Agenda RMI peuvent être sauvegardées à chaque ajout d'un rendez-vous, un agenda de groupe est créé par la pose d'interactions entre plusieurs agendas, un authentificateur est associé à certains agendas pour vérifier que seuls les utilisateurs autorisés accèdent à l'agenda. La figure 5 visualise un réseau de composants et d'interactions possibles à un moment donné de l'exécution de l'application.

4.1 Définition d'un schéma d'interaction

Dans l'exemple, le développeur de l'application d'agendas veut autoriser ses utilisateurs à lier à l'exécution un agenda à un " afficheur " afin d'être notifié chaque fois qu'un rendez-vous est ajouté dans l'agenda. Pour les agendas RMI, il peut être utile de gérer la persistance par stockage dans une base de données définie par l'utilisateur. Pour cela, le développeur définit des schémas d'interactions, qui sont fournis avec l'application (cf. figure

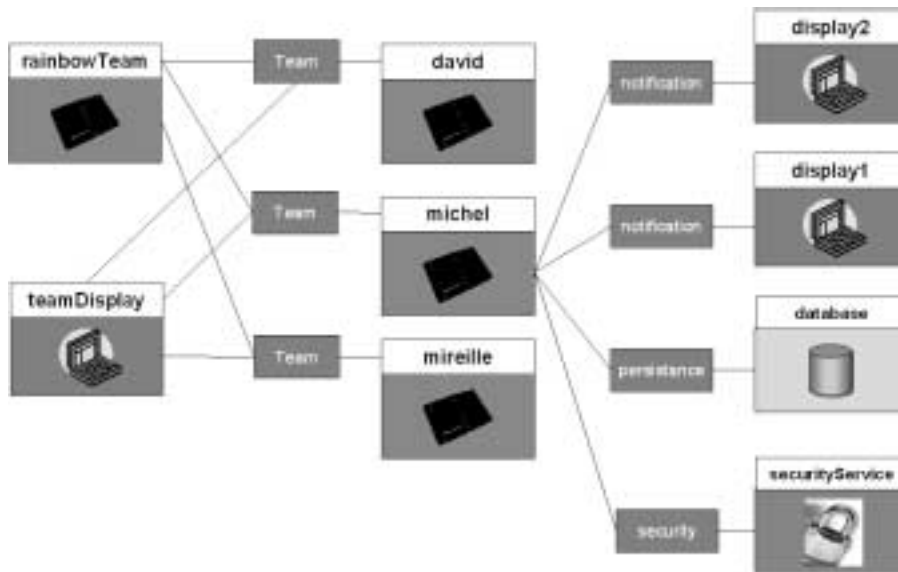


FIG. 5 – Assemblage de composants

6, schémas notification et persistance). Ces schémas définissent des modèles d'interactions que l'utilisateur final pourra utiliser. En cours d'exécution, un utilisateur des agendas peut vouloir ajouter de la notification entre agendas, interaction non prévue par le développeur de l'application, pour par exemple créer des agendas de groupe. De la même manière que précédemment, il définit ses propres schémas d'interactions (cf. figure 6, schéma group).

L'utilisateur exprime les " schémas " d'interactions en utilisant le Langage de Spécification des Interactions : ISL (cf. figure 6) qui est décrit plus amplement dans [Ber02]. Un schéma décrit plusieurs règles d'interactions qui expriment les contrôles qui doivent être opérés sur les objets liés. Une règle est composée d'une partie gauche qui exprime le message notifiant et d'une partie droite la réaction qui correspond à de la réécriture de code.

Le schéma notification (ligne1) peut lier un objet quel que soit son type et un composant de type Display. Il contient une règle (ligne 2 et 3). Elle exprime que tout message (utilisation de l'étoile) reçu par un objet ainsi lié, doit être exécuté dans un ordre indéterminé avec un envoi de message à l'objet display associé. Le schéma group (ligne 6) peut lier 3 composants. Il définit deux règles. La première stipule que le message addMeeting (message notifiant) des agendas correspondants au paramètre team entraînera la réaction (ligne 8) qui après l'exécution du message lui-même, ajoute le rendez-vous à l'agenda d'un membre. Dans le schéma notification, le mot clé `_call` est utilisé pour représenter l'appel du message notifiant (`O._call`), mais aussi pour représenter une réification du message notifiant lorsqu'il est utilisé seul (`_call`) comme paramètre d'une méthode.

```

1 interaction notification(Object O, Display display) {
2     O.* -> O._call //
3         display.notify(_call)
4 }
5
6 interaction group(Agenda team, Agenda member, Display display) {
7     team.addMeeting(java.lang.String title)
8     -> team._call; member.addMeeting(title)
9     ,
10    member.addMeeting(java.lang.String title)
11    -> member._call; display.notify(_call)
12 }
13
14 interaction persistence(Agenda A, Database database) {
15     A.addMeeting(java.lang.String title)
16     , java.lang.String owner
17     -> A._call;
18     owner:=A.getOwner(); database.store(owner,A)
19 }
20
21 interaction security(Object O, SecurityService service) {
22     O.* -> if service.check(_call) then
23         O._call
24     else
25         exception "unauthorized user"
26     endif
27 }

```

28:1

FIG. 6 – Schéma d'interactions

Les schémas sont enregistrés auprès du serveur d'interactions par appel de la méthode `createPattern` qui prend pour paramètre le schéma d'interactions concerné. Le serveur d'interactions stocke ainsi les schémas qui peuvent être récupérés pour modification ou analyse en utilisant la méthode `getPattern(String)`.

4.2 Pose et destruction d'interactions

En cours d'exécution, le programmeur peut lier ou délier les instances de composants entre eux en utilisant les schémas qui sont enregistrés auprès du serveur. Pour cela, il s'adresse au serveur d'interactions par la méthode `createInteraction(String createInteraction String PatternName, Object targets[]) throws Exception`. Le serveur mémorise la nouvelle interaction, renvoie son nom et demande à chacun des composants liés par l'interaction et définissant des messages notifiant de fusionner les nouvelles règles qui le concernent. La prise en compte de ces modifications aura lieu à compter du prochain appel.

Une interface graphique permet également la pose d'interactions. A partir de la liste des schémas d'interactions enregistrés, l'utilisateur sélectionne le schéma d'interaction à instancier (security par exemple) . Une fenêtre propose alors le type des composants à connecter (Object et SecurityService) et les composants interagissant existants de ce type afin de poser une interaction.

Dans notre exemple, si l'utilisateur demande la pose des interactions visualisées dans la figure 5, le comportement des instances de composants liés est modifié pour respecter le résultat de la fusion des différentes règles d'interactions. Le processus de fusion est décrit dans [Ber02].

Par exemple, à la méthode addMeeting de l'agenda AgendaRainbow est associée la règle suivante résultat de la fusion de la règle définie figure 6 à la ligne 7 instanciée successivement entre l'agenda AgendaRainbow et les agendas de AgendaDavid, AgendaAnne-Marie et AgendaMireille :

```
AgendaRainbow.addMeeting(java.lang.String _var0) ->
    AgendaRainbow._call;
    (AgendaDavid.addMeeting(_var0)
    // AgendaAnne-Marie.addMeeting(_var0)
    // AgendaMireille.addMeeting(_var0) )
```

Cette nouvelle règle exprime le fait que ajouter un rendez-vous dans l'agenda de groupe ajoute le rendez-vous dans un ordre non déterminé à chacun des participants. Le retrait d'interactions consiste à demander au serveur d'interactions le retrait d'une interaction par void removeInteraction(String interactionIdentifier) throws Exception. Lors de la destruction d'une interaction, chacun des composants déliés comportant un message notifiant est prévenu afin de retirer la règle qui le concerne et de recalculer la règle résultante.

4.3 Rendre un composant interagissant

Seules les instances de composants " interagissants " peuvent être liées par des interactions comportant des règles où leurs messages apparaissent comme notifiants. Tout composant peut cependant apparaître dans la partie réaction d'une règle.

Pour rendre un composant EJB interagissant, l'utilisateur doit simplement le spécifier dans le fichier de déploiement dont la grammaire XML a été étendue. Pour Jonas [OBJ], le développeur insère la ligne suivante : <jonas-interaction value="true"></jonas-interaction> Pour rendre une classe Java interagissante, il suffit d'appliquer l'outil fourni avec le service d'interaction, " GENINT ", qui modifie directement son byte-code.

4.4 Exécution

Lorsqu'un composant interagissant reçoit un message qui se révèle être notifiant, la règle d'interaction qui lui est associée, est évaluée localement. Les appels entre les composants lors de cette évaluation sont alors directs et ne passent pas par le serveur d'interactions.

4.5 Téléchargement : `http://noah.essi.fr`

5 Architecture pour l'auto-adaptation

Cette section décrit à travers un exemple simple l'utilisation d'un framework destiné à faciliter le développement d'applications auto-adaptatives à base de composants. L'idée centrale de ce framework est de considérer *l'adaptation des applications aux évolutions de leur environnement d'exécution* comme une préoccupation, ou un aspect, qui peut être traité séparément du code purement métier. Pour réaliser cette séparation, notre framework fournit trois grandes fonctionnalités aux programmeurs :

- un modèle de composants (en l'occurrence Fractal légèrement étendu) pour développer le code métier lui-même. L'utilisation de ce modèle rend automatiquement possible la reconfiguration dynamique de l'architecture de l'application. Notre extension de Fractal y ajoute le support de la réflexion comportementale.
- un service d'observation du contexte d'exécution de l'application capable de détecter les changements significatifs de ce contexte qui doivent déclencher une adaptation ;
- un DSL (*Domain Specific Language*) basé sur la notion de règle Événement – Condition – Action, permettant d'exprimer les politiques d'adaptation des composants. Dans l'état actuel, ce DSL n'existe pas encore, et les règles doivent être codées en Java.

L'utilisation de Fractal rend possible un certain nombre de reconfigurations de l'application : paramétrisation (à travers l'interface standardisée `attribute-controller`) et reconfigurations structurelles (modification des liaisons entre composants et des relations de composition). Cependant, ces deux types d'adaptations doivent avoir été anticipées au moment de la conception de l'application : le programmeur d'un composant décide des paramètres de configuration qu'il veut rendre disponibles, et en définissant l'architecture de l'application, il détermine aussi les reconfigurations structurelles possibles. Pour aller plus loin et permettre le support d'adaptations non anticipées, nous avons dû étendre Fractal en y ajoutant le support de la réflexion comportementale, alors que le modèle de base ne supporte que la réflexion structurelle. Concrètement, notre extension se présente sous la forme classique d'un MOP, mais adapté au modèle de composants Fractal. La figure 7 illustre le fonctionnement de cette extension : lorsqu'un compo-

sant réflexif reçoit un message, celui-ci est réifié au niveau du contrôleur, et redirigé vers un sous-composant implémentant une interface générique d'invocation de message (équivalent à un métaobjet). Ce “méta-composant” peut s'il le désire invoquer le composant correspondant au niveau de base.

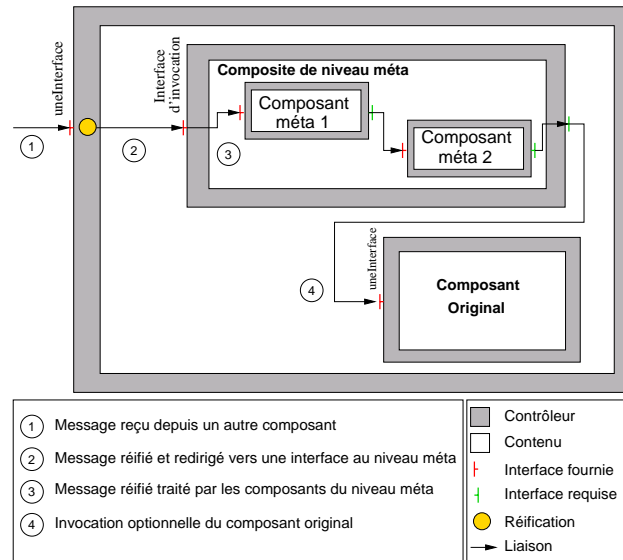


FIG. 7 – Extension Fractal pour la réflexion comportementale

Le reste de cette section décrit l'utilisation concrète du framework sur un exemple simple. Pour une description plus détaillée, voir [DL03].

5.1 Application initiale

L'application prise en exemple est un simple visualiseur d'images. Au cœur de cette application, nous trouvons un composant décodeur responsable d'interpréter le contenu de fichiers images (JPEG, PNG...) sous une forme affichable à l'écran. Ce décodeur dépend d'un autre composant pour charger le contenu des fichiers, soit localement, soit à travers le réseau. La figure 8 représente l'architecture initiale de cette partie de l'application, codée en utilisant Fractal.

5.2 Première adaptation : activation conditionnelle d'un cache

Telle quelle, l'application peut souffrir de problèmes de performances. En effet, les composants qui la constitue ont été conçus de façon à être le plus simple et réutilisables possibles, et en particulier, aucun des composants “chargeurs de fichiers” ne contient de cache.

Dans certaines circonstances, l'ajout d'un cache pourrait pourtant améliorer de beaucoup les performances, et la première adaptation que nous

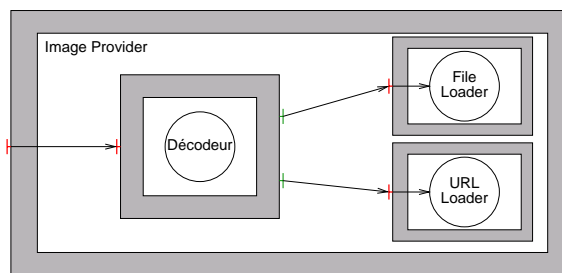


FIG. 8 – Architecture initiale de l’application exemple

allons ajouter va donc consister à ajouter au décodeur un cache transparent si les circonstances indiquent que cela en vaut la peine. Grâce à l’extension que nous avons ajouté à Fractal, il est relativement simple d’implémenter un service de cache générique et de l’ajouter (ou de le retirer) dynamiquement à un composant. La figure 9 montre la structure interne du composant décodeur une fois le cache activé.

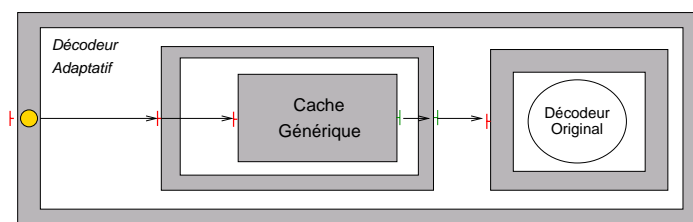


FIG. 9 – Activation conditionnelle du cache

Cette adaptation n’est mise en place que si nécessaire. Cela se traduit par une politique d’adaptation très simple, constituée d’une seule règle :

1. **Quand** le temps moyen d’exécution d’une requête (en l’occurrence chargement puis décodage d’une image) dépasse un certain seuil , **alors** instancier un nouveau “méta-composant” implémentant le service de cache, et l’attacher au composant de décodage.

De cette façon, lorsque le système détecte que les images mettent beaucoup de temps à se charger (parce qu’elles sont très grandes ou accédées à travers le réseau), et uniquement dans ce cas, il s’adapte automatiquement en ajoutant un cache pour éviter de payer plusieurs fois ce coût élevé si l’image est accédée de nouveau.

5.3 Deuxième adaptation : redimensionnement automatique du cache

La deuxième adaptation concerne la taille du cache, que nous n’avons pas précisé jusqu’à maintenant, alors qu’elle a un impact très élevé sur les

performances du système. Si elle est trop petite, le potentiel du cache est mal utilisé ; si elle est trop élevée par rapport à la quantité de mémoire disponible, on risque de payer très cher l'utilisation de la mémoire virtuelle. L'objectif de cette deuxième adaptation est donc d'ajuster la taille du cache à la quantité de mémoire disponible, qui bien sûr varie dynamiquement.

Puisque le cache que nous avons ajouté lors de la première adaptation est lui-même un composant Fractal, nous pouvons lui appliquer les mêmes techniques d'adaptation qu'à n'importe quel composant "métier", c'est-à-dire lui attacher une politique d'adaptation. Nous supposons que la taille du cache est un paramètre de configuration du composant, disponible à travers l'interface standard `attribute-controller` de Fractal. La valeur de ce paramètre sera modifiée dynamiquement par la politique d'adaptation, en fonction de la quantité de mémoire libre. Cette information concernant le contexte d'exécution nous est fournie par un sous-système de notre framework, chargé d'observer le contexte d'exécution et ses évolutions. Elle est disponible au niveau des politiques d'adaptation sous le nom `res :/storage/memory.free` (l'attribut `free` de la ressource nommée `memory` dans la catégorie `storage` du domaine `res`). Les trois règles constituant la politique d'adaptation sont (`free` est utilisé comme raccourci pour `res :/storage/memory.free`) :

1. **Quand** la valeur de `free` change, **si** elle est inférieure à `low`, **alors** mettre la taille du cache à 0.
2. **Quand** la valeur de `free` change, **si** elle est supérieure à `high`, **alors** mettre la taille du cache à `max`.
3. **Sinon, quand** la valeur de `free` change, **alors** mettre la taille du cache à $(\text{free} - \text{low}) \times \frac{\text{max}}{\text{high} - \text{low}}$.

Ces règles indiquent que lorsque la mémoire disponible passe en dessous de `low`, le cache est désactivé en mettant sa taille à 0 (règle 1). Sinon, sa taille augmente linéairement en fonction de la quantité de mémoire disponible (règle 3) jusqu'à atteindre une valeur maximale `max` lorsque la quantité de mémoire disponible est supérieure à `high` (règle 2). De cette façon, la taille du cache est toujours adaptée à la quantité de mémoire disponible, de façon à en tirer le meilleur parti sans risquer de dégradation des performances due à l'utilisation de la mémoire virtuelle.

5.4 Troisième adaptation : sélection d'une politique de remplacement

Lorsque le cache est plein, il utilise une politique de remplacement pour décider lequel des éléments qu'il contient doit être supprimé pour faire de la place. La politique la plus simple et la plus efficace dans le cas général (accès aléatoire) est la politique LRU (Least Recently Used). Cette politique peut cependant poser des problèmes de *trashing* dans les cas où les don-

nées sont accédées en séquence. Pour pallier ce problème, Glass¹ propose l’algorithme de remplacement SEQ. Celui-ci se comporte par défaut comme LRU, mais s’il détecte un début d’accès séquentiel aux données, il change son comportement en MRU (Most Recently Used), ce qui évite le *trashing*. Cet algorithme s’exprime très naturellement dans le cadre de nos politiques d’adaptation, et nous permet d’illustrer le dernier type de reconfiguration que nous supportons : les reconfigurations structurelles.

Nous supposons que le composant qui implémente le cache utilise à travers une connexion un autre composant qui, lui, implémente la politique de remplacement (voir figure 10). Notre framework permet d’attacher au cache une nouvelle politique d’adaptation qui, lorsqu’elle détecte le début ou la fin d’un accès en séquence aux données, modifie la liaison qui relie le cache à sa politique de remplacement, activant alternativement une politique LRU ou MRU.

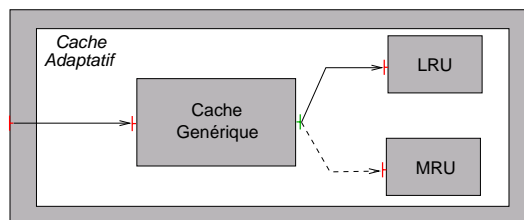


FIG. 10 – Adaptation de la politique de remplacement du cache

Le politique d’adaptation correspondante correspondrait aux règles suivantes :

1. **Quand** la méthode prise en charge par le cache est invoquée, **si** les N dernières requêtes étaient consécutives, **alors** déconnecter le composant LRU et le remplacer par le composant MRU.
2. **Quand** la méthode prise en charge par le cache est invoquée, **si** les N dernières requêtes *n’étaient pas* consécutives, **alors** déconnecter le composant MRU et le remplacer par le composant LRU.

5.5 Conclusion & évaluation

Dans cette section, nous avons montré comment notre framework permet de rendre auto-adaptative une application construite à base de composants Fractal, et cela sans avoir à modifier le code de l’application elle-même. Pour cela, nous avons dû étendre le modèle standard de Fractal pour lui ajouter le support de la réflexion comportementale. Notre framework fournit aussi un service d’observation du contexte d’exécution des applications capable

¹G. Glass and P. Cao, Adaptive Page Replacement Based on Memory Reference Behavior, Proceedings of ACM SIGMETRICS, pp. 115–126, June 1997

de détecter des changements significatifs de cet environnement afin de déclencher les adaptations. Enfin, nous utilisons la notion de règle Événement – Condition – Action (pour l’instant programmées en Java) pour encoder les politiques d’adaptation qui permettent de rendre les composants auto-adaptatifs.

6 Fractal GUI

Le but de FractalGUI est définir des configurations de composants Fractal de façon visuelle et interactive, d’instancier et démarrer les configurations ainsi définies, et de reconfigurer dynamiquement et visuellement les configurations créées précédemment (pas encore implanté).

L’outil offre trois vues possibles : arborescente, graphe, dialogue. La vue graphe permet l’édition de la taille et de la position des composants et des liaisons, montre un niveau hiérarchique variable, et permet de zoomer sur des détails. La vue dialogue permet l’édition des noms et types des composants et des interfaces. L’outil permet également le chargement et enregistrement de et vers l’ADL Fractal.

L’avantage de cet outil est d’offrir une vision concrète de l’architecture des applications (non directement visible en Fractal ADL).

La première démonstration consiste à créer la configuration de l’application HelloWorld, afin de montrer les fonctionnalités de l’outil (cf figure 11).

La deuxième démonstration présente l’architecture de Fractal GUI (cf figures 12 et 13), et quelques variantes possibles, afin de montrer l’avantage des composants logiciels pour obtenir rapidement des configurations et assemblages différents. Fractal GUI est en effet composé d’environ 100 composants Fractal sur 5 niveaux de hiérarchie, dont des composants Model, View et Controller (MVC).

Références

- [ABLR03] M. Auguin, F. Baude, D. Lavenier, and M. Riveill. *Actes des conférences RenPar’15 - CFSE’3 - SympAAA’2003*. INRIA, La Colle sur Loup - France, 15, 16 et 17 octobre 2003. 652 pages. ISBN 2-7261-1264-1.
- [ACC02] I. Attali, D. Caromel, and A. Contes. Security for Distributed and Mobile Active Objects with the ProActive Library. ErcimNews, Avril 2002. http://www.ercim.org/publication/Ercim_News/enw49/.
- [ACC03a] I. Attali, D. Caromel, and A. Contes. Hierarchical and declarative security for grid applications. In *International Conference On High Performance Computing, HIPC, Hyderabad*,

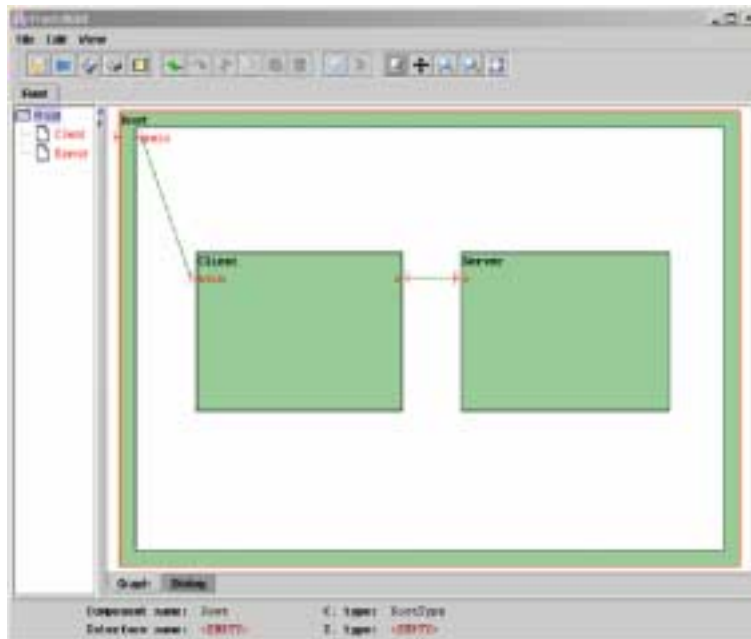


FIG. 11 – Fractal GUI 1

India, December 17-20, volume 2913, pages 363–372, Springer Verlag, 2003. Lecture Notes in Computer Science, LNCS.

- [ACC03b] I. Attali, D. Caromel, and A. Contes. Une architecture de sécurité déclarative et hiérarchique pour les grilles de calcul. In INRIA, editor, *2ème rencontre francophone sur Sécurité et Architecture Réseaux*, pages 203–212, Nancy, France, July 2003.
- [AHN02] S. Alouf, F. Huet, and P. Nain. Forwarders vs. centralized server : An evaluation of two approaches for locating mobile agents. *Performance Evaluation*, 49 :299–319, septembre 2002. ISSN 0166-5316.
- [Bad01] L. Baduel. Communication de groupes efficace pour objets actifs répartis. Master's thesis, Université de Nice - Sophia Antipolis, June 2001.
- [BBC⁺01] F. Baude, A. Bergel, D. Caromel, F. Huet, O. Nano, and J. Vayssière. IC2D : Interactive Control & Debug for Distribution. In *3ème Int. Conference on "Large-Scale Scientific Computations"*, volume 2179 of LNCS, 2001.
- [BBC02a] L. Baduel, F. Baude, and D. Caromel. Communications de Groupes Typés dans ProActive. In *École thématique sur*

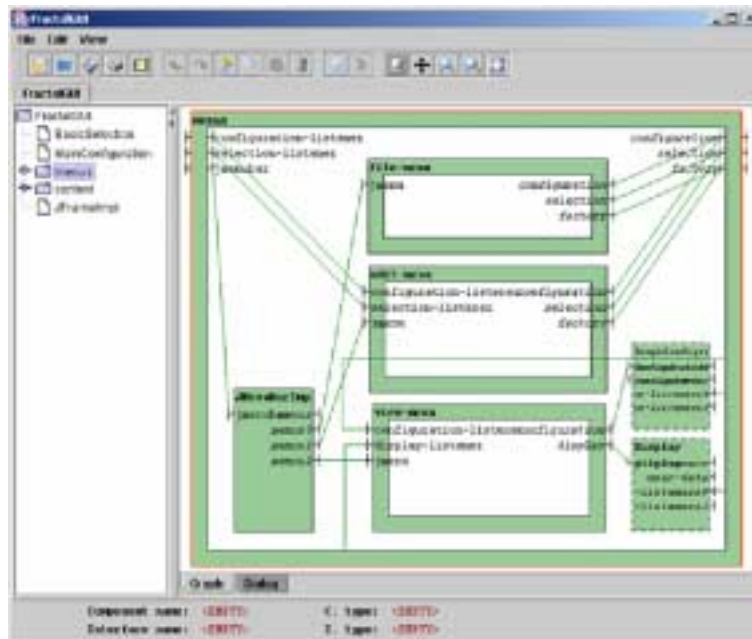


FIG. 12 – Fractal GUI 2

la globalisation des ressources informatiques et des données (GRID), pages 74–84. CNRS, dec 2002.

- [BBC02b] L. Baduel, F. Baude, and D. Caromel. Efficient, flexible, and typed group communications in java. In *Joint ACM Java Grande - ISCOPE 2002 Conference*, pages 28–36, Seattle, 2002. ACM Press. ISBN 1-58113-559-8.
- [BBC⁺04] L. Baduel, F. Baude, D. Caromel, C. Delbe, N. Gama, S. El Kasmi, and S. Lanteri. A parallel object-oriented application for 3d electromagnetism. In *IEEE International Symposium on Parallel and Distributed Computing, IPDPS*, 2004. To appear.
- [BBPH03] S. Bouchenak, F. Boyer, N. De Palma, and D. Hagimont. Can aspects be injected? experience with replication and protection. in proceedings of the international symposium on distributed objects and applications (doa 2003), catania, italy, november 3-7, 2003.
- [BC02] M. Beauvois and T. Coupaye. Composition comportementale d’aspects techniques (behavioural composition of technical services). journées composants 2002 (jc2002), asf (acm sigops france), grenoble, octobre 2002.

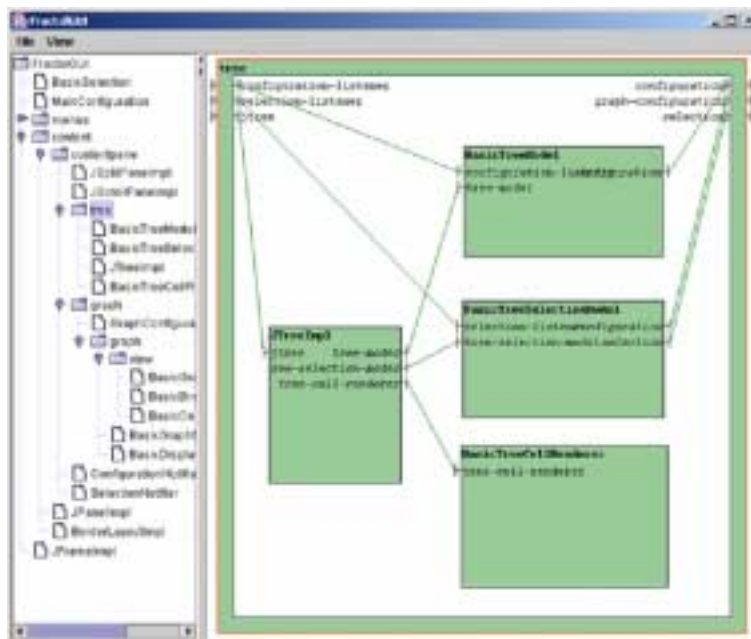


FIG. 13 – Fractal GUI 3

- [BC03] L. Baduel and D. Caromel. Communication de groupe typé pour objets répartis. In Michel Auguin, Françoise Baude, Dominique Lavenier, and Michel Riveill, editors, *Actes de Ren-Par'15*, pages 119–126. INRIA, 2003.
- [BCH⁺02] F. Baude, D. Caromel, F. Huet, L. Mestre, and J. Vayssière. Interactive and descriptor-based deployment of object-oriented grid applications. In *11th IEEE International Symposium on High Performance Distributed Computing HPDC-11*, pages 93–102, Edimburgh, 2002. IEEE Computer Society. ISBN 0-7695-1686-6.
- [BCHV02] F. Baude, D. Caromel, F. Huet, and J. Vayssière. Objets actifs mobiles et communicants. *Technique et science informatiques*, 21(6) :1–36, 2002.
- [BCL⁺04] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J-B. Stefani. An open component model and its support in java. in proceedings of the international symposium on component-based software engineering, edinburgh, scotland, 24-25 may 2004.
- [BCM] F. Baude, D. Caromel, and M. Morel. From distributed objects to hierarchical grid components. 3rd annual ObjectWeb conference, INRIA Rocquencourt, 20-21 novembre 2003.

- [BCM03] F. Baude, D. Caromel, and M. Morel. From distributed objects to hierarchical grid components. In *International Symposium on Distributed Objects and Applications - DOA, Catania, Sicily, Italy, 3-7 November*, volume 2888, pages 1226–1242, Springer Verlag, 2003. Lecture Notes in Computer Science, LNCS.
- [BCS02a] F. Boyer, O. Charra, and A. Senart. Réflexivité pour des environnements adaptables. in *les intergiciels*, pages 73-92. hermès, 2002.
- [BCS02b] E. Bruneton, T. Coupaye, and J-B. Stefani. Recursive and dynamic software composition with sharing. in *proceedings of the 7th ecoop international workshop on component-oriented programming (wcop'02), malaga (spain), june 10th-14th, 2002*.
- [Bea03] M. Beauvois. Brenda : Towards a composition framework for non orthogonal non functional properties. 4th ifip international conference on distributed applications and interoperable systems (dais03), paris, france, november 17-21, 2003.
- [Ber01] L. Berger. *Mise en oeuvre des interactions en environnements distribués, compilés et fortement typés : le modèle MICADO*. PhD thesis, Université de Nice-Sophia Antipolis, octobre 2001.
- [Ber02] L. Berger. Intéraction et modèles de programmation, support des interactions dans les systèmes objets et componentiels. *Numéro spécial de la revue L'OBJET : Coopération dans les systèmes à objets*, 8(3) :9–38, 2002.
- [BFCE⁺04] M. Blay-Fornarino, A. Charfi, D. Emsellem, A-M. Pinna-Dery, and M. Riveill. Software interaction. *Journal of Object Technology*, x(y), z 2004.
- [BFEO⁺02] M. Blay-Fornarino, D. Ensellem, A. Ocelllo, A-M. Pinna-Dery, M. Riveill, J. Fierstone, O. Nano, and G. Chabert. Un service d'interactions : principes et implémentation. In ISBN 2-7261-1229-3 INRIA, editor, *Journée composants : Systèmes à composants adaptables et extensibles, Grenoble, France, 17 et 18 octobre 2002*.
- [BFEPDR04] M. Blay-Fornarino, D. Emsellem, A-M. Pinna-Dery, and M. Riveill. Un service d'interactions : principes et implémentation. *RSTI - série TSI*, 23(2) :175–204, 2004.
- [BFM04] M. Blay-Fornarino and P. Merle, editors. *Interopérabilité des systèmes distribués : des Modèles aux Intergiciels*, volume x, n°y of *Revue TSI*. Lavoisier, 2004.

- [BFPD02] M. Blay-Fornarino and A-M. Pinna-Dery, editors. *Cooperation dans les systèmes à objets*, volume 8, n° 3 of *Revue l'Objet - logiciel, bases de données et réseaux - vol. 8, N° 3*. Lavoisier, 2002.
- [BFPDR02] M. Blay-Fornarino, A-M. Pinna-Dery, and M. Riveill. Towards configuring distributed applications. In *2nd IEEE International Workshop on Aspect Oriented Programming for Distributed Computing Systems, Vienna, Austria*, pages 487–492, July 2-5 2002.
- [BLC02] E. Bruneton, R. Lenglet, and T. Coupaye. Asm : un outil de manipulation de code pour la réalisation de systèmes adaptables (asm : a code manipulation tool for the construction of adaptable systems). journées composants (jc2002), asf (acm sigops france), grenoble, octobre 2002.
- [BMBF⁺01] M. Bartorello, H. Maguin, M. Blay-Fornarino, A.-M. Pinna-Dery, and M. Riveill. Adjonction de services au sein d'un serveur ejb. In *Journées composants : flexibilité du système au langage, Besançon*, 25 et 26 octobre 2001.
- [BMO⁺02] M. Bartorello, H. Maguin, A. Occello, M. Blay-Fornarino, A.-M. Pinna-Dery, and M. Riveill. Intégration de services au sein d'un serveur ejb. *Conférence Langages et Modèles à Objets - LM0 2002, Montpellier France - publié dans la revue revue RTSI - série L'Objet (Lavoisier Eds)*, 8(1-2) :169–184, 28-30 janvier 2002.
- [BR01a] E. Bruneton and M. Riveill. An architecture for extensible middleware platforms. In *Software : Practice and Experience, Volume 31, Issue 13, Pages : 1237-1264, Online ISSN : 1097-024X, Print ISSN : 0038-0644*, 2001.
- [BR01b] E. Bruneton and M. Riveill. Experiments with javapod, a platform designed for the adaptation of non-functional properties. In *REFLECTION 2001, Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, Kyoto, Japan*, pages 52–72. Springer-Verlag - Lecture Notes in Computer Science, September 26-28 2001.
- [BSL01] Noury M. Bouraqadi-Saâdani and Thomas Ledoux. How to weave ? ECOOP 2001 Workshop on Advanced Separation of Concerns, June 2001.
- [BSLS01] N. M. N. Bouraqadi-Saâdani, T. Ledoux, and M. Südholt. A reflective infrastructure for coarse-grained strong mobility and its tool-based implementation. Invited presentation at the *International Workshop on "Experiences with reflective systems"*

- (held in conjunction with Reflection 2001, the “*3rd International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*”), September 2001. also published as TR 01/7/INFO.
- [Car03a] D. Caromel. Objets concurrents, répartis et mobiles. Invited lecture at Ecole Jeunes Chercheurs en Programmation, april 2003.
- [Car03b] D. Caromel. Programming models for future systems vs. mpi, april 2003. <http://csdl.computer.org/comp/proceedings/ipdps/2003/1926/00/1926toc.htm>.
- [CC03] D. Caromel and S. Chaumette, editors. *Java for Parallel and Distributed Computing, Workshop in IPDPS 2003*, 2003. ISBN 0-7695-1926-1.
- [CCFG01] D. Caromel, S. Chaumette, G. Fox, and P. Graham, editors. *Java for Parallel and Distributed Computing*, San Francisco, mai 2001. IPDPSP’01, Workshop in IEEE IPDPS’2001 Proceedings, IEEE CS Press. ISBN 0-7695-0990-8.
- [CCFG02] D. Caromel, S. Chaumette, G. Fox, and P. Graham, editors. *Java for Parallel and Distributed Computing*, Fort Lauderdale, April 2002. IPDPSP’02, Workshop in IEEE IPDPS’2002 Proceedings, IEEE CS Press. ISBN 0-7695-1573-8.
- [CER04] A. Charfi, D. Emsellem, and M. Riveill. Dynamic component composition in .net. *Journal of Object Technology*, 3(2) :37–46, february 2004. Special issue : .NET : The programmer’s Perspective : ECOOP Workshop 2003.
- [CG03] D. Caromel and A. Genoud. Non-functional exceptions for distributed and mobile objects. In Alexander Romanovsky, Christophe Dony, Jorgen Lindskov Knudsen, and Anand Tripath, editors, *ECOOP 2003 Workshop on Exception Handling in Object Oriented Systems : towards Emerging Application Areas and New Programming Paradigms*, July 2003. <http://homepages.cs.ncl.ac.uk/alexander.romanovsky/home.fo-rmal/ehoos2003.html>.
- [Cha03] A. Charfi. Software interaction : Towards dynamic adaptation of .net objects. Master Science Thesis - Technical University of Munich, july 2003.
- [CHS04] D. Caromel, L. Henrio, and B. Serpette. Asynchronous and deterministic objects. In *Proceedings of the 31st ACM Symposium on Principles of Programming Languages*, pages 123–134. ACM Press, 2004.

- [CHV01] D. Caromel, F. Huet, and J. Vayssière. Combining security with meta programming in java. In A. Yonezawa and S. Matsuo, editors, *Proceedings of the third International Conference, Reflection 2001, Kyoto, Japan, September 25-28, 2001*, volume 2192 of *LNCS*, pages 118–125. Springer, 2001.
- [CLB⁺01] T. Coupaye, R. Lenglet, M. Beauvois, E. Bruneton, and P. Déchamboux. Composants et composition dans l’architecture de systèmes répartis. journées composants : flexibilité du système au langage, asf (acm sigops france), besançon, Octobre 2001.
- [CMT04] D. Caromel, L. Mateu, and E. Tanter. Sequential object monitors. In *Proceedings of the European Conference on Object-Oriented Programming, ECOOP’2004*, LNCS. Springer, 2004. A paraitre.
- [Con01] A. Contes. Programmation à objets distribués : Sécurisation de collecticiels. Master’s thesis, Université de Nice - Sophia Antipolis, June 2001.
- [CQ03] D. Caromel and R. Quilici. Proactive : From active objects to components, towards pse. In *EURESCO Conference on Advanced Environments and Tools for High Performance Computing*, EuroConference on Problem Solving Environments and the Information Society, Albufeira, Portugal, june 2003. <http://www.esf.org/euresco/03/pc03139>.
- [Cre01] F. Crevola. Langage de description d’applications. Rapport de DEA Réseau et Systèmes Distribués, Université de Nice-Sophia Antipolis, juin 2001.
- [CS01] O. Charra and A. Senart. Adaptable and extensible bindings in distributed environments in proceedings of the 4th european research seminar on advances in distributed systems (ersads’2001), pages 239-244, bertinoro, italy, may 2001.
- [CS02] O. Charra and A. Senart. Thinkrcx, un noyau de système d’exploitation extensible pour lego rcx. in proceedings of les journées systèmes à composants adaptables et extensibles, pages 239-244, grenoble (france), October 2002.
- [CV01] D. Caromel and J. Vayssière. Reflections on mops, components, and java security. In Jorgen Lindskov Knudsen, editor, *Proceedings of the European Conference on Object-Oriented Programming, ECOOP’2001, Budapest, Hungary, June 18-22*, volume 2072 of *LNCS*, pages 256–274. Springer, 2001.
- [CV03] D. Caromel and J. Vayssière. A security framework for reflective java applications. *Software Practice and Experience, SPE, Wiley & Sons, InterScience*, 33(9) :821–846, 2003.

- [Dav01] Pierre-Charles David. Une infrastructure pour middleware adaptable. Rapport de DEA, École des Mines de Nantes, Université de Nantes, September 2001.
- [Del03] C. Delbé. Tolérance aux pannes pour un modèle à objets actifs. Master's thesis, UNSA, June 2003. Rapport de stage DEA RSD, in french.
- [dF03] M. Didonet del Fabro. Adaptive replication models, an experimentation with ejb. DEA Réseau et Systèmes Distribués, Université de Nice - Sophia Antipolis, July 2003.
- [DL02a] Pierre-Charles David and Thomas Ledoux. Dynamic adaptation of non-functional concerns. In *First International Workshop on Unanticipated Software Evolution (USE'02) at ECOOP 2002*, Malaga, Spain, June 2002.
- [DL02b] Pierre-Charles David and Thomas Ledoux. An infrastructure for adaptable middleware. In R. Meersam and Zahir Tari et al, editors, *On the Move to Meaningful Internet Systems 2002 : CoopIS, DOA, ODBASE 2002*, volume 2519 of *Lecture Notes in Computer Science*, pages 773–790. Springer-Verlag, October 2002.
- [DL03] Pierre-Charles David and Thomas Ledoux. Towards a framework for self-adaptive component-based applications. In *Proceedings of DAIS'03*, Lecture Notes in Computer Science, Paris, November 2003. Federated Conferences, Springer-Verlag.
- [DLBS01] Pierre-Charles David, Thomas Ledoux, and Noury M. Bouraqadi-Saâdani. Two-step weaving with reflection using AspectJ. In *OOPSLA 2001 Workshop on Advanced Separation of Concerns in Object-Oriented Systems*, 2001.
- [ECR03] D. Emsellem, A. Charfi, and M. Riveill. Dynamic component composition in .net. In *ECOOP'2003 Workshop on ".NET : The Programmer's Perspective"*, Darmstat, Germany, 22 July 2003.
- [ER02] D. Emsellem and M. Riveill. Rainbow's interaction. In *First rotor workshop, Cambridge, UK*, July 2002.
- [ER03] D. Emsellem and M. Riveill. Interaction integration into the cli. In *2nd Rotor Workshop*, Pisa, Italy, 23-25 april 2003. Microsoft Research.
- [Fak02] H. Fakih. Extension du modèle client-serveur pour la mobilité : Vers un modèle java rmi mobile et dynamique. Rapport de DEA RACOR (Réseaux Avancés de CONnaissances et oRganisations), Université Technologie de Troyes, septembre 2002.

- [FS01] J-P. Fassino and J-B. Stefani. Think : un noyau d'infrastructure répartie adaptable, cfse-2, avril 2001.
- [FSLM02] J-P. Fassino, J-B. Stefani, J. Lawall, and G. Muller. Think : a software framework for component-based operating system kernels, usenix 2002, monterey, california, usa, 10-15 June 2002.
- [Gen02] Alexandre Genoud. Exceptions fonctionnelles et non fonctionnelles pour objets mobiles et asynchrones. Master's thesis, Université de Nice - Sophia Antipolis, June 2002. Stage DEA Informatique.
- [HB01] D. Hagimont and F. Boyer. A configurable rmi mechanism for sharing distributed java objects. iee internet computing, vol. 5, 1, pp. 36-44, Jan.-Feb. 2001.
- [Hen03] Ludovic Henrio. *Asynchronous Object Calculus : Confluence and Determinacy*. PhD thesis, Université de Nice - Sophia Antipolis – UFR Sciences, November 2003.
- [HI] D. Hagimont and L. Ismail. A protection scheme for mobile agents on java, third acm/ieee international conference on mobile computing and networking (mobicom), budapest, septembre 1997.
- [HL] D. Hagimont and D. Louvegnies. Javanaise : Distributed shared objects for internet cooperative applications, ifip international conference on distributed systems platforms and open distributed processing (middleware'98), the lake district, septembre 1998.
- [HP02] D. Hagimont and N. De Palma. Removing indirection objects for non-functional properties. in proceedings of pdpta 2002, international conference on parallel and distributed processing techniques and applications, las vegas (usa), June 2002.
- [Hue02] Fabrice Huet. *Objets Mobiles : conception d'un middleware et évaluation de la communication*. PhD thesis, Université de Nice - Sophia Antipolis, December 2002.
- [JDL02] Zahi Jarir, Pierre-Charles David, and Thomas Ledoux. Dynamic adaptability of services in enterprise JavaBeans architecture. In *Seventh International Workshop on Component-Oriented Programming (WCOP'02) at ECOOP 2002*, Malaga, Spain, June 2002.
- [Mar02] V. Marangozova. Patrons de conception pour la duplication de composants. in proceedings of asf 2002, journées francophones des jeunes chercheurs en systèmes d'exploitation, hammamet (tunisia), april 10th-13th, 2002.

- [MH] V. Marangozova and D. Hagimont. An infrastructure for corba component replication, rapport technique sirac, novembre 2001.
- [MH01] V. Marangozova and D. Hagimont. Adaptation d'une application répartie pour la disponibilité. deuxième conférence française sur les systèmes d'exploitation (cfse-2), paris (france), Avril 2001.
- [MH02a] V. Marangozova and D. Hagimont. An infrastructure for corba component replication. in proceedings of the first international ifip/acm working conference on component deployment, berlin (germany), june 20th-21st, 2002.
- [MH02b] V. Marangozova and D. Hagimont. An architectural approach to replication configuration. in proceedings of the 6th international conference on principles of distributed systems (opodis'2002), reims (france), December 2002.
- [MH02c] V. Marangozova and D. Hagimont. Non-functional replication management in the corba component model. in proceedings of the 8th international ifip/acm conference on object-oriented information systems, montpellier (france), September 2002.
- [Min01] P. Nhat Minh. Interaction et communication de groupe. Rapport de DEA Réseau et Systèmes Distribués, Université de Nice-Sophia Antipolis, juin 2001.
- [Nan01] O. Nano. Composition de services. Rapport de DEA Informatique, Université de Nice-Sophia Antipolis, juin 2001.
- [NBF03a] O. Nano and M. Blay-Fornarino. Services integration by model annotation and transformation. In Paul Sammut Edited by Andy Evans and James S. Williams, editors, *First International Workshop on Metamodelling for MDA*, pages 77–92, York, England, UK, 24-25 november 2003.
- [NBF03b] O. Nano and M. Blay-Fornarino. Une approche mda pour l'intégration des services dans les plates-formes à composants. In *Journées Objets, Composants et Modèles*, Vannes, 5 février 2003. GDR ARP.
- [NBF03c] O. Nano and M. Blay-Fornarino. Using mda to integrate services in component platforms. In *Eighth International Workshop on Component-Oriented Programming (WCOP 2003) in conjunction with ECOOP2003*, Darmstat, Germany, 21 july 2003.
- [NBFDR02] O. Nano, M. Blay-Fornarino, A-M. Dery, and M. Rivell. An abstract model for integrating and composing services in component platforms. In *Seventh In-*

- ternational Workshop on Component-Oriented Programming in conjunction with ECOOP'2002, Malaga, June 10 2002.*
- [Nie02] Diego Niewbourg. Grid et metacomputing dans proactive. Master's thesis, Université de Nice Sophia-Antipolis, September 2002. Stage DESS Télécommunications.
- [OBFPDR02] A. Occello, M. Blay-Fornarino, A-M. Pinna-Dery, and M. Riveill. Vers une adaptation dynamique cohérente des composants. In ISBN 2-7261-1229-3 INRIA, editor, *Journée composants : Systèmes à composants adaptables et extensibles, Grenoble, France, 17 et 18 octobre 2002.*
- [Occ02] A. Occello. Composants : Vers une adaptation dynamique cohérente. Rapport de DEA Informatique, Université de Nice-Sophia Antipolis, juillet 2002.
- [OPDBFR03] A. Occello, A-M. Pinna-Dery, M. Blay-Fornarino, and M. Riveill. Contrôle des adaptations d'applications à base de composants. In *Journées Objets, Composants et Modèles*, Vannes, 5 février 2003. GDR ARP.
- [Pas01] M. Pasin. Adding continuous availability to enterprise java beans application servers using group communication. In *Proceedings of the DSN-2001, The International Conference on Dependable Systems and Networks* <http://www.dsn.org> - Student Forum, Göteborg, Sweden, June 30th - July 4th 2001.
- [Pas03] M. Pasin. *Réplicas para alta disponibilidade em arquiteturas orientadas a componentes com suporte de comunicação de grupo*. PhD thesis, Universidade do Rio Grande do Sud, maio 2003.
- [PDBFM02] A-M. Pinna-Dery, M. Blay-Fornarino, and S. Moisan. Apport des interactions pour la distribution des connaissances. *Numéro spécial de la revue L'OBJET : Systèmes distribués et connaissances*, 8(4) :85–109, 2002.
- [PR01] M. Pasin and M. Riveill. Implementing fault-tolerance ejbs using group communication system support. In *European Research Seminar in Advanced Distributed Systems, ER-SADS'2001*, Bertinoro, Italy, 14-18 may 2001.
- [PR02] M. Pasin and M. Riveill. A multi-layer architecture for high available enterprise javabeans. In *WTF2002 - III WORKSHOP DE TESTES E TOLERÂNCIA A FALHAS*, Búzios, RJ, Brazil, 20 a 24 maio 2002.
- [PR03] M-C. Pellegrini and M. Riveill. Component management in a dynamic architecture. *Special issue of The Journal of Supercomputing*, 24(2) :151–159, February 2003.

- [PRC03] M. Prochaska, R. Rouvoy, and T. Coupaye. On enhancing component-based middleware with transactions. 5th international symposium on distributed objects and applications (doa03) - poster session, catania, sicily (italy), 3-7 november, 2003.
- [PRW01] M. Pasin, M. Riveill, and T. Silva Weber. Using group communication to establish distributed checkpoint in replicated ejb application servers. In *JAIIO - ASSE* <http://www.ing.unlpam.edu.ar/jaiio2001/>, 2001.
- [PWdFR03] M. Pasin, T. S. Weber, M. Didonet del Fabro, and M. Riveill. A highly-available replicated component-based distributed architecture. In ISBN 2-7261-1264-1 INRIA, editor, *3ème Conférence Française sur les Systèmes d'Exploitation - CFSE*, pages 549–560, La Colle sur Loup, France, 14-17 octobre 2003.
- [PWR03a] M. Pasin, T. S. Weber, and M. Riveill. Alta disponibilidade para aplicações distribuídas transacionais através de réplicas e comunicação de grupo. In *XXIX Conferencia Latinoamericana de Informatica*, La Paz, Peru, 29 septiembre-2 octubre 2003.
- [PWR03b] M. Pasin, T. S. Weber, and M. Riveill. A high available replicated component-based distributed architecture. Technical Report I3S/RR-2003-14-FR, Laboratoire I3S - Université de Nice-Sophia Antipolis, Bâtiment ESSI - BP145 - F-06903 Sophia Antipolis CEDEX, june 2003.
- [Rap01] P. Rapicault. Specifying the usage and composition of software components. In *Poster session, ECOOP01*, 2001.
- [RBR03] M. Rits, K. Boudaoud, and M. Riveill. Security engineering for adaptable software components. In *First ACM Workshop on Business Driven Security Engineering (BIZSEC)*, Fairfax, USA, 31 october 2003.
- [RC03] N. Rivierre and T. Coupaye. Observing component behaviors with temporal logic. workshop on correctness of software composition (cmc) at ecoop 2003, darmstadt, germany, 2003.
- [Rit03] M. Rits. Component adaptability and security. DEA Réseau et Systèmes Distribués, Université de Nice - Sophia Antipolis, july 2003.
- [Riv02a] M. Riveill. Programmation par composition et déploiement d'applications réparties. *Numéro spécial de la revue L'OBJET : Coopération dans les systèmes à objets*, 8(3) :91–107, 2002.

- [Riv02b] M. Riveill, editor. *Systèmes à composants adaptables et extensibles*. Actes des Journées Composants - Grenoble - France - 17 et 18 octobre 2002 - INRIA - ISBN : 2-7261-1229-3, 2002.
- [Riv03] M. Riveill. Assemblage dynamique de composants logiciels. In *Journées Académiques*, Paris, 16-18 mars 2003. Microsoft Research.
- [Riv04] M. Riveill, editor. *Systèmes à composants adaptables et extensibles*, volume 23, n°2 of *RSTI - série TSI*. Lavoisier - ISBN 2-7462-0913-6, 2004.
- [RS02] M. Riveill and A. Senart. Aspects dynamiques des langages de description d'architecture logicielle. *Numéro spécial de la revue L'OBJET : Coopération dans les systèmes à objets*, 8(3) :109–129, 2002.
- [Sag02] F. Sagna. Transactions mobiles. Rapport de DEA Réseau et Systèmes Distribués, Université de Nice-Sophia Antipolis, septembre 2002.
- [Say02] Haris Saybasili. Un modèle de composant hiérarchique pour proactive et les principes de son implémentation avec fractal. Master's thesis, Université de Nice - Sophia Antipolis, September 2002. Stage DEA Réseaux et Systèmes Distribués.
- [SCS02] A. Senart, O. Charra, and Jean-Bernard Stefani. Developing dynamically reconfigurable operating system kernels with the think component architecture. in proceedings of the workshop on engineering context-aware object-oriented systems and environments, in association with oopsla'2002, seattle (usa), november 4th-8th, 2002.
- [SGN] J.B. Stefani, F. Germain, and E. Najm. Elements of an object-based model for distributed and mobile computation, 4th international conference on formal methods for open object-based distributed systems, stanford, ca, usa, september 2000.
- [SL01] François Sarradin and Thomas Ledoux. Adaptabilité dynamique de la sémantique de communication dans Jonathan. In Robert Godin and Isabelle Borne, editors, *LMO 2001 - Langages et modèles à objets*, pages 45–62, Le Croisic, France, January 2001. Hermès. L'Objet, 7(1-2).
- [TBSN01] E. Tanter, N-M. Bouraqadi-Saâdani, and J. Noyé. Reflex : une extension réflexive de Java portable, souple et performante. In Robert Godin and Isabelle Borne, editors, *LMO 2001 - Langages et modèles à objets*, pages 165–180, Le Croisic, France, January 2001. Hermès. L'Objet, 7(1-2).
- [TNCC03] E. Tanter, J. Noyé, D. Caromel, and P. Cointe. Partial behavioral reflection : Spatial and temporal selection of reification.

In Ron Crocker and Guy L. Steele, Jr., editors, *Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA 2003)*, pages 27–46, Anaheim, California, USA, October 2003. ACM Press.

[Vay02]

Julien Vayssière. *Une architecture de sécurité pour les applications réflexives - Application à Java*. PhD thesis, Université de Nice - Sophia Antipolis, November 2002.