

# PROJET RNTL ARCAD

## D2.1 - Propriétés non-fonctionnelles à appliquer aux composants

T0+15 :document interne

Coordonnateur : Denis Caromel  
Univ. de Nice Sophia Antipolis – INRIA  
2004 Rt. des Lucioles, BP 93  
F-06902 Sophia Antipolis Cedex

8 Mars 2002

### Résumé

Ce document a pour objectif de définir un ensemble de propriétés non-fonctionnelles à appliquer aux composants. C'est à dire celles entrant dans le domaine de l'étude.

## 1 Introduction

Ce document définit les propriétés non-fonctionnelles des composants. Plus spécifiquement, les propriétés visées sont directement liées à la nature de plus en plus distribuée et mobile des composants logiciels (migration, communication asynchrone, mode déconnecté, etc.).

Une application est dite mobile lorsque l'un de ses constituants (matériels, logiciels, utilisateurs) change de localisation physique en cours d'exécution. Ces applications se développent très rapidement en raison des nouveaux modes de travail et des possibilités techniques : communication sans fil, appareils portables (ordinateurs et téléphones mobiles, PDA). Même lorsque la mobilité ne concerne que les utilisateurs, il se pose le problème de fournir un environnement uniforme à un utilisateur qui change de point d'accès.

Dans l'architecture extensible qui doit supporter l'exécution des différents composants, objectif de ce projet RNTL, la partie déploiement et assemblage de composants est primordiale. Les propriétés non-fonctionnelles que nous

allons aborder dans ce sous-projet devront pouvoir être configurées lors du déploiement, mais également dynamiquement lors de l'exécution.

## 2 Propriétés non-fonctionnelles

Nous souhaitons traiter les propriétés non-fonctionnelles suivantes.

### 2.1 Communication asynchrone, synchronisation par futurs

Les infrastructures habituelles offrent pour la plupart une communication synchrone. Une communication asynchrone permet bien souvent une plus grande indépendance entre les composants, avec en particulier une meilleure résistance aux inter-blocages. Nous souhaitons donner la possibilité de désynchroniser facilement les communications entre deux composants, et donc il nous semble nécessaire d'avoir une telle propriété.

Par contre, les communications asynchrones souffrent d'une manière évidente d'un manque de synchronisation, en particulier en cas de dépendances fonctionnelles entre composants. Pour cette raison, la propriété de communication asynchrone pourra être couplée avec la possibilité d'avoir une synchronisation par futurs entre composants.

### 2.2 Communication en mode déconnecté

Cette propriété prolonge d'une certaine manière la précédente. Dans le cas d'une communication asynchrone, on peut demander aux deux composants d'être tous deux activés et connectés au moment de la communication ; cela permet entre autre de garantir certaines propriétés. Mais on peut, dans certains cas c'est une contrainte forte de l'application, vouloir permettre une communication dite en "Mode déconnecté". Il n'est plus alors nécessaire que les deux composants soient connectés au même instant.

Notons que dans ce cas, les synchronisations par futurs sont également un aspect pertinent et fort utile.

### 2.3 Communication de groupe

La capacité à participer à une communication de groupe est une propriété importante dans le cadre de composants devant servir à la construction d'applications collaboratives, de commerce électronique, ou encore de services télécoms. Il est donc important de définir les modalités d'une telle commu-

nication comme un aspect non-fonctionnel configurable, et d'en implémenter les primitives nécessaires.

Par rapport aux autres sous-projets, cette propriété servira de support pour la gestion de *données dupliquées* (réplication), et pour la communication de groupe de plus haut niveau d'abstraction nécessaire aux *aspects interactions* (Sous-projet 3).

## 2.4 Migration

Les techniques de migration et de mobilité sont une des bases permettant aux applications de s'adapter aux changements de localisation de ses différents composants et de se reconfigurer. La migration peut se faire au moyen de la mobilité de code, de calcul, ou de donnée. La mobilité des données, associée à des techniques de gestion de caches répartis, permet à la fois de diminuer la latence d'accès aux informations et de modifier dynamiquement l'environnement d'exécution d'une application pour répondre à des besoins changeants. La mobilité du code et de calculs permet par exemple de déplacer dynamiquement l'exécution d'un processus client vers un serveur de données pour remédier à la variabilité des performances d'un réseau, ou à une déconnexion. L'aspect migration de composants est donc une propriété non-fonctionnelle fondamentale. Il faudra donc définir les caractéristiques et contraintes nécessaires pour permettre la migration d'un composant, et définir une ou plusieurs sémantiques de migration (faible, forte, etc.).

## 2.5 Création, placement dynamique, et accès à distance

Un composant est habituellement créé et placé de façon statique dans un container. Avec les nouvelles techniques de programmation d'architecture répartie mises en oeuvre dans ce projet, et en particulier les aspects réflexifs, il devrait être possible de créer dynamiquement des composants à partir d'objets Java standard, de les rendre ainsi accessible à distance par d'autres composants pré-existants, voir même de les placer dynamiquement dans d'autres containers.

## 2.6 Sécurité

Les techniques utilisant la mobilité du code et des calculs impliquent l'exécution sur un serveur de programmes provenant d'une autre machine. En l'absence de mesures de protection appropriées, une telle exécution présente un danger potentiel. Des aspects sécurité doivent donc être pris en compte.

Définir la sécurité comme une propriété non-fonctionnelle devrait permettre une configuration dynamique de cet aspect qui est absolument nécessaire en présence de migration. En effet, lorsque des composants se déplacent d'un domaine administratif à un autre, les attributs de sécurité à utiliser lors des communications changent. Il est donc impératif d'adapter dynamiquement les caractéristiques de la communication point à point entre chaque paire de composants (authentification, intégrité, confidentialité, etc.).

D'autre part, les applications *coopérantes* nécessitent l'adjonction de mécanismes spécifiques. Plusieurs utilisateurs physiques ("*principals*") collaborant par le biais d'une application, il est nécessaire, d'une part de les identifier, d'autre part d'associer à différentes entités (personne physique, ordinateur) un *contrôle d'accès* spécifique.

### 3 Conclusion

En résumé, les propriétés non-fonctionnelles appliquées aux composants seront les suivantes :

- création, placement dynamique, et migration,
- communication asynchrone,
- synchronisation par futurs,
- communication en mode déconnecté,
- communication de groupe, et gestion de données dupliquées,
- interactions (voir Sous-projet 3),
- sécurité (authentification, intégrité, confidentialité et contrôle d'accès),

Globalement, les approches à base d'interception et à base de tissage de code seront souvent comparées, simultanément au sein d'une équipe ou par le biais de travaux complémentaires menés par plusieurs équipes.