Résumé d'article

An open component Model and its support in Java

E. Bruneton*, T. Coupaye*, M. Leclerc**, V. Quema**, J.B. Stefani**
France Télécom R&D* et INRIA Rhône-Alpes**

Romain Rincé

Université de Nantes
2 rue de la Houssinière, BP92208, F-44322 Nantes cedex 03,
France

1 Introduction

Le papier « *An open component Model and its support in Java* » présente le modèle à composant FRACTAL réalisé conjointement avec la section R&D de France Télécom et l'équipe INRIA Rhône-Alpes.

L'objectif des auteurs étant de résoudre un certain nombre de limitations présentes dans les différents modèles existants (e.g. EJB, CORBA, .Net) telles que :

- Les faibles possibilités d'extensibilité.
- La difficulté (voire l'impossibilité) d'ajouter dynamiquement des moyens de contrôle sur les composants.
- La nécessité de parfois devoir faire des choix difficiles entre le degré de configurabilité et les performances.

Afin de répondre à ces problématiques, les auteurs présentent leur modèle à composant FRACTAL, ainsi que son implémentation en Java, le framework JULIA.

2 Le modèle à composant FRACTAL

L'objectif du modèle à composants FRACTAL est de pouvoir implémenter ou gérer des systèmes logiciels complexes, et ce même sur des systèmes moins traditionnels tels que l'embarqué. Pour ce faire FRACTAL offre la possibilité d'avoir des composants composite, des composants partagés, des capacités d'introspection et de reconfiguration.

Il est à présent nécessaire de détailler un peu plus précisément le modèle FRACTAL (c.f. Figure 1). Le modèle FRACTAL est principalement composé de deux entités qui sont le contenu et la membrane.

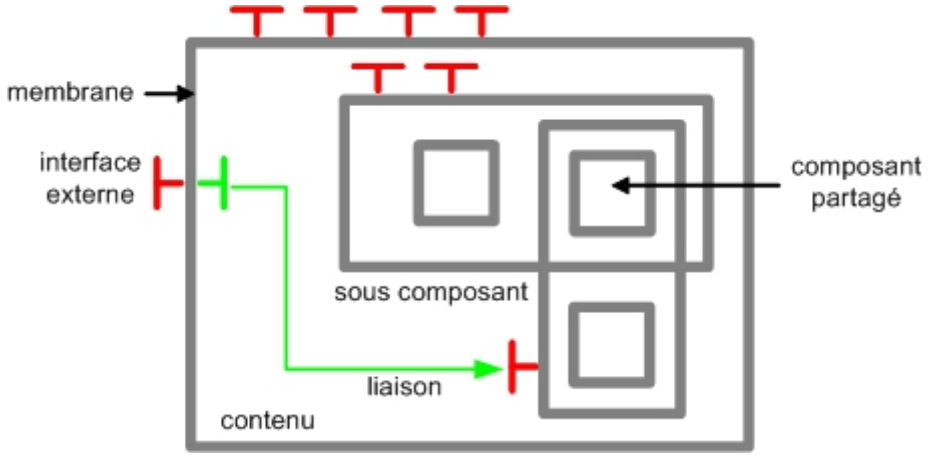

FIGURE 1 – Exemple d'un modèle FRACTAL de composants

Contenu Le contenu peut être soit un ensemble de composants, soit du code du langage cible (e.g. Un objet en java).

Membrane La membrane est l'élément qui permet au modèle d'offrir une organisation en composant. Celle-ci va encapsuler le contenu pour lui offrir les propriétés d'un composant. La membrane dispose en effet d'interfaces externes pour communiquer avec des composants de même niveaux ou avec son parent, et d'interfaces internes pour communiquer avec ses sous-composants. Ces interfaces peuvent être soit de type `Client` (c'est-à-dire qui demandent un service), soit de type `Serveur` (c'est-à-dire qui fournissent un service). Les interfaces `Serveur` peuvent être reliées à plusieurs interfaces `Client` ou aucune. Les interfaces des composants devant bien évidemment être liées pour pouvoir communiquer.

La membrane dispose aussi de contrôleurs qui offrent un certain nombre d'actions sur le composant. Ces contrôleurs peuvent notamment effectuer les actions suivantes :

- Accéder ou modifier les attributs du composant.
- Modifier les liaisons internes au composant.
- Ajouter ou supprimer des sous-composants.
- Agir sur le cycle de vie du composant (e.g. Démarrer ou arrêter l'exécution du composant).

Il s'agit bien sûr d'une liste non-exhaustive des contrôleurs ; le modèle FRACTAL permettant l'extensibilité de ses contrôleurs.

Pour une spécification plus détaillée, vous pouvez vous reporter au rapport technique de FRACTAL[BCS03].

3 Le framework JULIA

JULIA[Jul] est un framework implémentant en Java le modèle FRACTAL. Il fournit donc un ensemble extensible de classes (`Controller` et `Interceptor`) permettant le design des membranes des composants. L'objectif étant de répondre aux problématiques présentées en introduction, JULIA permet une gestion des coûts en performance plus fine grâce à une continuité entre la reconfiguration statique et dynamique. De plus JULIA doit pouvoir fonctionner sur des environnements restraints n'ayant, par exemple, pas de `ClassLoader`, pas d'API de reflexion, pas d'API de collection ...

Présention JULIA dipose donc d'un ensemble de `Controller` et d'`Interceptor` (c.f. Figure 2). Les `Interceptor` gèrent les entrées/sorties sur l'appel de méthodes des interfaces (à l'exception des interfaces de contrôle). Les `Controller` quant-à-

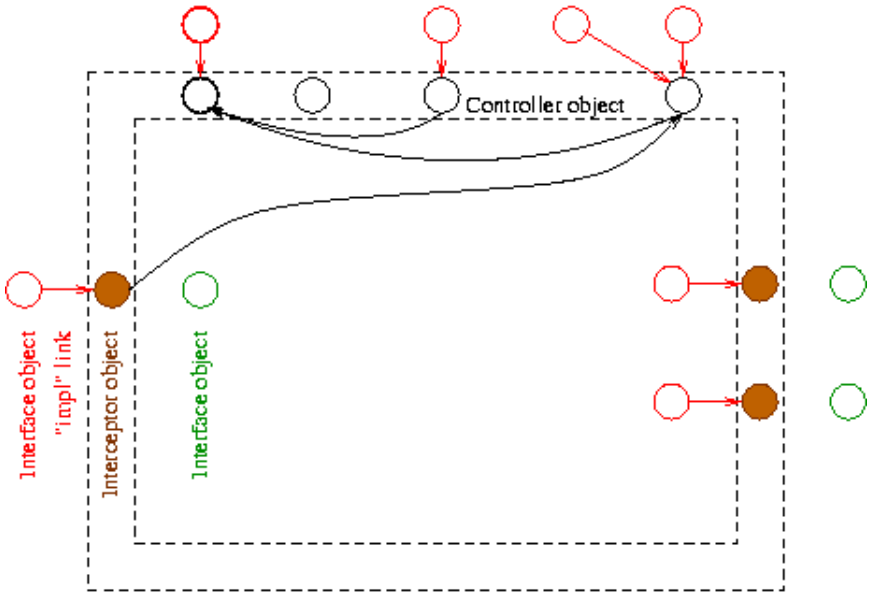

FIGURE 2 – Exemple d'un modèle FRACTAL de composants

eux correspondent, en terme de concept, aux contrôlleurs de FRACTAL. Dans son ensemble un composant est donc un ensemble de trois types d'objets :

- Les objets qui implémentent les interfaces. Il faut noter qu'il est nécessaire de passer par un `Interceptor` pour que la membrane ait accès à cet objet concret (L'interface ayant un lien `impl` vers cet objet) ; les interfaces `Client` ont leur lien `impl` à `null` (Ils ne nécessitent pas d'objets les instanciants).
- Les objets de la membrane (`Controller` et `Interceptor`).
- Les objets implémentants le contenu.

Extensibilité Afin de permettre l'extensibilité de son framework, JULIA fournit un mécanisme de classe abstraite permettant d'étendre les classes offertes de base. Par exemple il est possible d'étendre un contrôlleur gérant le démarrage et l'arrêt d'un composant (i.e. un `LifeCycleController`) pour lui fournir une capacité d'arrêt automatique.

Optimisations Pour améliorer les performances, JULIA effectue divers optimisations

- Des optimisations intra-composant qui consistent à faire fusionner les objets de certains contrôleurs entre-eux.
- Des optimisations inter-composant qui consistant à effectuer des optimisations de référence sur les liens (c.f. Figure 3).

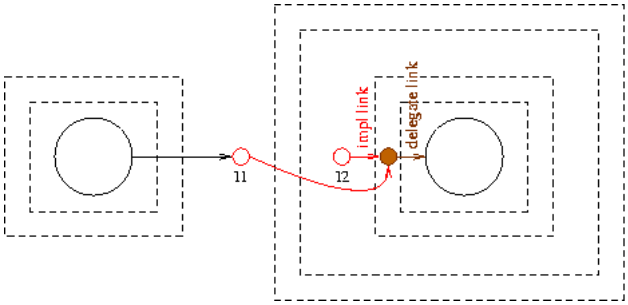

FIGURE 3 – Exemple d'un modèle FRACTAL de composants

4 Qualité du modèle FRACTAL et de JULIA

Dans la dernière partie du papier, les auteurs évaluent leur travail par rapport aux problèmes soulevés en début de rapport. JULIA semble répondre à toutes leurs exigences, tant sur le plan de la modularité et l'extensibilité, que sur les performances (notamment mémoire). De plus JULIA étant relativement léger (45 ko en exécution), il est capable de tourner sur des systèmes extêmement restreints.
Cependant les auteurs soulèvent certaines limitations au niveau de la modularité et de l'exensibilité du framework JULIA ; notamment dû au méchanisme de classes abstraites proposé.

Brève conclusion Les résultats obtenus par les auteurs ainsi qu'une relative fierté de ces derniers vis-à-vis de leurs travaux, laissent à penser que le modèle FRACTAL possède nombre de qualités ; la création d'une communauté développant de nouveaux frameworks autour de celui-ci (AOKell[AOK], Think[Thi], FracTalk[Fra] ...) ne fait que confirmer cette observation.

Bibliographie

[AOK] Aokell. http://fractal.ow2.org/aokell/index.html.

[BCS03] E. Bruneton, T. Coupaye, and J.B. Stefani. The fractal component model. Technical report, ObjectWeb Consortium, http://www.objectweb.org/fractal, 2003.

[Fra] Fractalk. http://csl.ensm-douai.fr/FracTalk.

[Jul] Julia. http://fractal.ow2.org/julia/.

[Thi] Think. http://think.ow2.org/.