Architecture des composants Runes.

On peut retrouver le modèle de composant runes dans trois implémentations, une en C/linux et C/Contiki et une autre en Java. Ces trois versions peuvent différer notamment en C, en fonction de la plateforme cible, nous allons donc étudier les possibilités offerte en java.

La connexion entre composants se fait à l'aide d'interfaces et de réceptacles, un réceptacle représente le composant qui peut offrir un service, l'interface est donc cliente de celui-ci. Un composant a la possibilité d'implémenter plusieurs interfaces et réceptacles. Des architectures complexes peuvent être réalisé à partir de cela, de plus il est possible de réaliser des composants dit composite, c'est à dire composés d'autres composants. On peut ainsi à l'intérieur d'une application définir des composants a fonctionnalité précise, comme par exemple un algorithme de compression pour vidéo ou autre.

La connexion entre les deux entités se fait par l'intermédiaire d'un « Connector » qui est un composant particulier permettant les échanges. Ce dernier permet d'apporter des propriétés de réflexivité à l'architecture, on a entre autre la possibilité d'ajouter des intercepteurs au niveau des réceptacles et des interfaces, on peut ainsi demander des pré et post conditions au moment de l'invocation des composants pour entre autre effectuer des contrôles ou ajouter des sécurités à l'architecture. Au moment de la connexion, le « Connector » conservera l'interface et sera ensuite placé dans le réceptacle.

Si l'on déploie un composant dans un autre dit « Capsule », une possibilité de configuration à l'exécution est ainsi offerte, on pourra entre autre suivre le cycle de vie de celui-ci, mais aussi instancier d'autres composants. La « Capsule » a la particularité de conserver les interfaces et réceptacles de chaque composant instancié, ainsi il est aisé de déconnecter un composant pour le remplacer par un autre.

Les particularités de réflexivité et de configuration à l'exécution sont des éléments important pour le modèle et notamment pour la cible concernée. En effet, prenons le cas d'un serveur qui accepte une certaine quantité de connexion, le composant gérant les connexions pourra par exemple être « débranché » si le nombre maximum d'utilisateurs est atteint.

Les caractéristiques de Runes peuvent se retrouver dans un modèle de composant tel Fractal entre autre, les différences se font au niveau de l'implémentation. En fractal, chaque composant possède un membrane, celle-ci permet entre autre d'introduire dans un composant des contrôleurs et des intercepteurs sur chaque interface et permettre à chaque composant d'avoir des propriétés de réflexivité et de configuration à l'exécution, on se passe donc ici des « Connector » intermédiaire présent avec Runes. Runes offre la possibilité de choisir les composants qui seront configurable à l'exécution avec les « Capsule » tandis que les composants Fractal le sont tous. Ceci n'est pas négligeable car par l'intermédiaire des « Capsule », si l'on désir par exemple obtenir les attributs d'un composant, les composants étant stocké dans une HashTable, il est relativement simple de les retrouver. En Fractal, par contre, il n'est pas rare de devoir remonter parmi toutes les interfaces depuis le composant « root » pour retrouver le composant désiré. En revanche, Fractal possède un langage de définition d'architecture, qui permet de « binder » les composants très facilement et offre ainsi une certaine visibilité par rapport à ce que fait Runes, qui ressemble à ce que l'on peut faire avec Fractal sans extension.

Références

http://www.ist-runes.org/
http://runesmw.sourceforge.net/contiki.html
http://www.ist-runes.org/docs/deliverables/D1_01.pdf
http://www.ist-runes.org/docs/deliverables/D1_05_02.pdf
http://fractal.objectweb.org/specification/index.html