Architecture fonctionnelle

Patrick Moreau
le 14/11/2014

1 – Motivations

L'architecture fonctionnelle d'un logiciel correspond à une description de haut niveau du logiciel.
Cette description facilite les échanges entre les développeurs et les acteurs du transfert car elle établit une vision partagée du logiciel.

Remarque sur l'apport aux chercheurs
Il a été constaté qu'une équipe de recherche ayant réalisé l'architecture d'un logiciel gagne en efficacité tant pour le développement que pour la maintenance du logiciel. Pour les logiciels sous licence libre, l'établissement de l'architecture est un moyen de communication vers la communauté de contributeurs. En général, les architectures des logiciels libres matures sont relativement modulaires et se prête ainsi à un développement collaboratif. L'établissement de l'architecture peut également participer à l'identification de composants logiciels génériques au sein d'un laboratoire de recherche.

Souvent, l'architecture d'un logiciel de recherche est effectuée après le développement du logiciel. Dans l'industrie, celle-ci est élaborée durant la phase de conception. Il est alors considéré que la phase de conception devrait consommer autour de 40 % [1] de l'effort total de développement et devrait être supérieure ou égale, en effort, à la phase de codage.

Le fait que le logiciel existe ne simplifie pas en fait l'établissement de l'architecture du logiciel. Cela amène beaucoup de questions.
- la première est de savoir quelle est la limite exacte du logiciel. En effet, le logiciel est souvent issu de travaux dans le cadre de projets collaboratifs ou de contrats bilatéraux avec des industriels. Des développements peuvent avoir été effectués sans jamais avoir été intégrés dans la version du logiciel. Certains composants sont nécessaires pour exécuter le logiciel mais ne font pas partie du logiciel. On peut par exemple cité le JRE (Java Runtime Environment) ou encore des outils comme MySQL qu'il est nécessaire d'installer séparément pour exécuter certains logiciels que l'on installe sur son propre ordinateur.
- La deuxième est d'avoir une représentation fonctionnelle du logiciel qui est en phase avec ce qui est réellement implémenté dans le logiciel. Il arrive parfois qu'il soit difficile d'extraire dans le code les grandes fonctions du logiciel. L'auteur a pu d'ailleurs constater que souvent la phase de définition de l'architecture s'accompagnait de réarrangement du code...

Cette architecture sert donc
- pour établir la stratégie de transfert du logiciel. Une stratégie différente pourra être définie pour les différentes parties du logiciel, notamment en fonction des parties génériques des parties plus métiers.
- pour effectuer l'analyse juridique du logiciel. L'analyse juridique d'un logiciel permet d'identifier les problèmes éventuels et d'évaluer les risques au regard du schéma de transfert envisagés.

2 – Comment procéder ?

L'architecture fonctionnelle décrit d'une manière symbolique et schématique les différents éléments du logiciel. Nous proposons une découpe en trois niveaux différents :

- Une description générale du logiciel à plus grande échelle
- Une description des différents blocs fonctionnels du logiciel et leurs interactions.
- Une description détaillée de chaque bloc fonctionnel détaillant les différents composants logiciels qui en font partie, ainsi que leur interaction et les dépendances entre eux.

Remarque : Sur des actions de transfert sur des longues durées, il peut être parfois nécessaire de revisiter l'architecture en fonction des évolutions du logiciel.

2.1 - Description générale du logiciel

Tout d'abord, la fonction du logiciel (Qu'est-ce que cela fait ? A quoi cela sert ?) est décrite. Puis, le logiciel et l'architecture du système dans lequel il s'inscrit sont documentés. Le logiciel est alors vu comme un « gros » module monolithique. C'est à ce niveau que sont listés les logiciels tiers important nécessaires au bon fonctionnement du logiciel mais qui ne font pas partie du logiciel.
Cette phase peut ne pas être nécessaire pour certains logiciels comme des librairies par exemple.

2.2 - Description des différents blocs fonctionnels du logiciel et leurs interactions

La définition de l'architecture consiste à décrire l'organisation générale d'un logiciel et sa décomposition en modules ou blocs ou composants (le terme de plugin est parfois utilisé) ainsi que les interactions entre les modules.
Conseil :
Pour un logiciel donné, il est vivement conseillé de s'accorder sur un vocabulaire unique et commun.

Un bloc fonctionnel correspond à une partie spécifique et identifiable du logiciel (par exemple: le noyau, les outils, interface graphique, ...). Un bloc fonctionnel est un ensemble de composants logiciels offrant un ensemble cohérent de fonctions.
Dans un premier temps, seule l'organisation générale du logiciel est utile pour définir la stratégie de transfert.
Il est nécessaire d'identifier pour un logiciel le ou les personnes qui maîtrisent l'architecture. Les acteurs du transfert veilleront à ce que cette architecture soit agréée par l'ensemble de ces personnes. De manière générale, la définition de ces blocs se fait en itérant avec les acteurs du transfert, ce qui conduit soit à regrouper des blocs entre eux car la séparation n'est pas nécessaire d'un point de vue transfert soit à « ouvrir la boite » pour bien séparer des fonctions différentes.

Pour chaque module, les éléments d'information suivants sont demandés :

* Nom du module
* Description de la fonction du module
  - Qu'est-ce que cela fait ? A quoi cela sert ?
  - Est-ce une commodité (un bien commun) scientifique: ex: librairie d'inversion de matrice
  - La « valeur métier » sera identifiée. Par valeur métier, on entend spécificité versus généricité.
* Interaction avec les autres modules du logiciel

2.3 - Description de chaque bloc fonctionnel

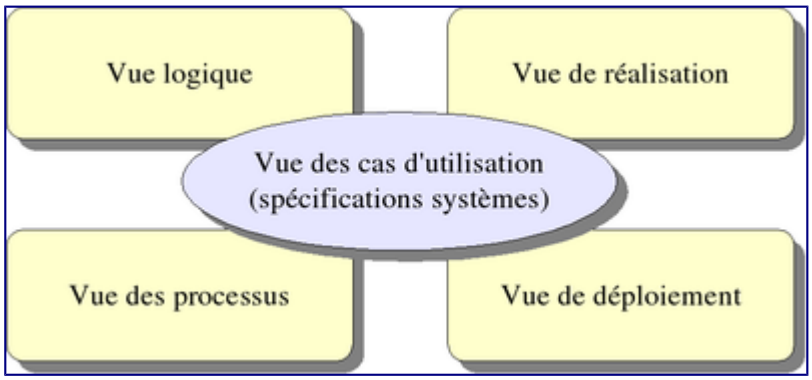
Cette description doit préciser les composantes de chaque bloc fonctionnel . Cela permet d'approfondir la connaissance sur le logiciel et est souvent requise pour une analyse juridique poussée d'un logiciel. Notamment, le fait d'identifier les dépendances avec des logiciels développés par des tiers permet d'avoir un premier niveau d'analyse juridique sur les compatibilités de licence et les ayants-droits. De plus, cela permettra d'établir si il existe des dépendances qui peuvent devenir obsolètes et ainsi introduire du risque sur la maintenabilité/évolutivité du logiciel.
Il est important cependant de bien doser l'effort à fournir, notamment par les chercheurs, en fonction des besoins et des enjeux

Pour chaque composant d'un bloc fonctionnel, les éléments d'information suivants sont demandés :

- Nom du composant
- Version du composant
- Nature du composant parmi les 4 types suivants :
    - le composant a été créé ex-nihilo, sans l'intégration de code pré-existant (code pré-existant appartenant à un tiers ou appartenant à vous, mais provenant d'un autre logiciel)
    - le composant a été créé ex-nihilo et intègre du code pré-existant
    - le composant est un composant pré-existant, qui n'a pas été modifié par les auteurs du logiciel
    - le composant est un composant pré-existante qui a été modifié par les auteurs du logiciel.
- Licence du composant (le cas échéant)
- Règle de composition : il sera indiqué le type de lien / de dépendance existant entre le composant et le reste du logiciel (par exemple: liaison dynamique).

3 - Quel style architectural ?

Il n'y a pas de description générique et universelle. La description est spécifique à chaque logiciel.
Il faut bien avoir en mémoire l'objectif de l'architecture fonctionnelle : un document d'échange partagé. L'architecture ne va pas donc détailler les classes, les modèles ou encore méta-modèles. Autrement dit, même si il existe une architecture mise au point pour la conception, il se peut que celle-ci soit trop complexe, trop détaillée pour être partagée avec des acteurs du transfert souvent non-informaticiens. Par exemple, le modèle des 4 + 1 vues n'est pas adapté à nos besoins :

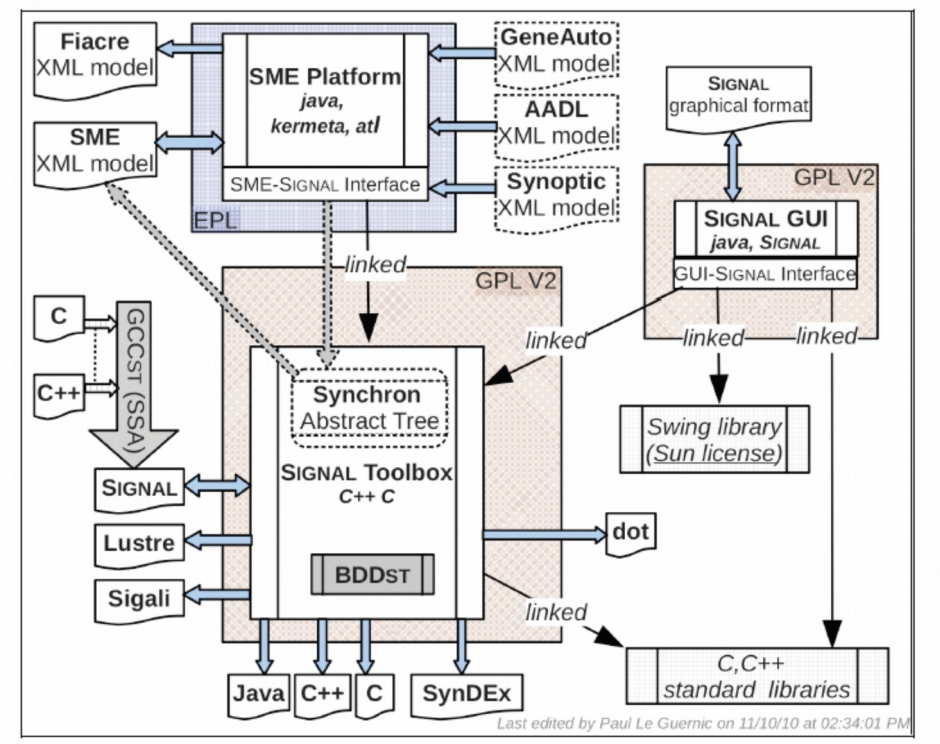


Source wikipedia

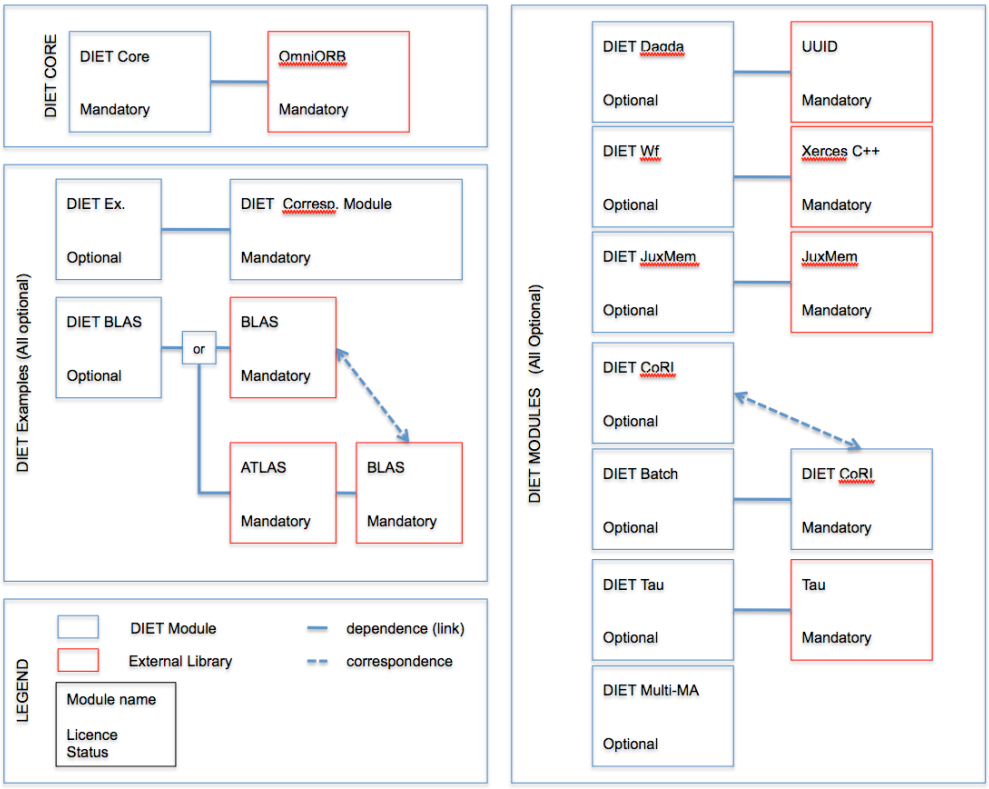
D'après nos expériences, deux grands styles architecturaux se distinguent :

Architecture en appels et retours

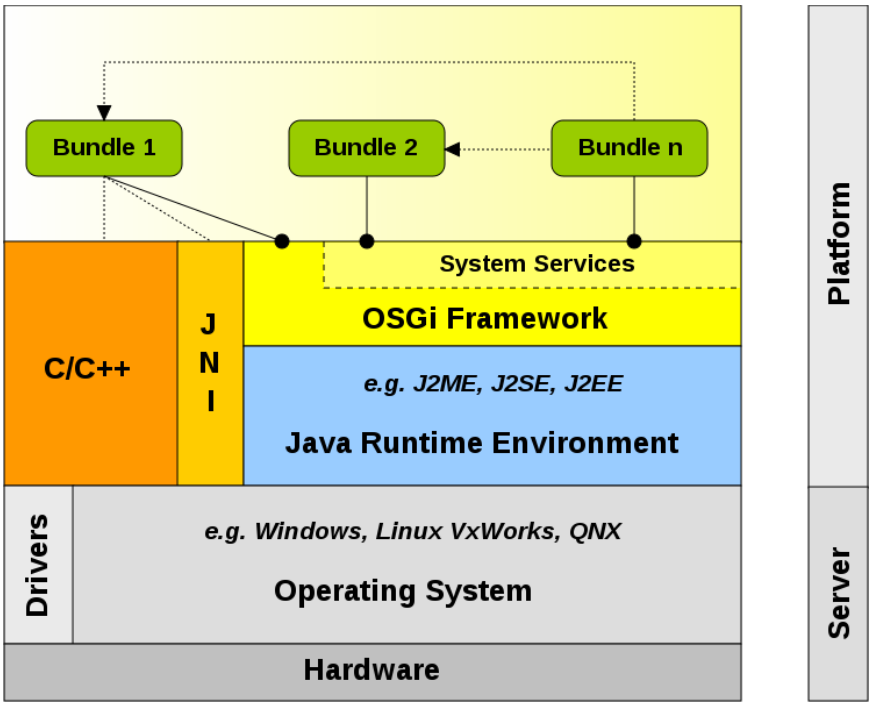
L'architecture en appels et retours consiste à découper une fonctionnalité en sous-fonctionnalités qui sont elles-mêmes divisées en sous sous-fonctionnalités. Elle est également appelée décomposition fonctionnelle. Les deux architectures ci-dessous illustrent ce type d'architecture.



Polychrony, Crédit Inria



Architecture en pile ou en couches
La conception de logiciels nécessite de recourir à des modules. Un module très spécialisé utilise des modules moins spécialisés qui eux-mêmes utilisent des modules génériques.
L'auteur a une préférence pour ce type d'architecture car les composants ayant de la valeur métier sont facilement identifiable.
Cette architecture est une bonne illustration d'une architecture en pile.



Osgi, CC BY-SA 3.0 - Michael Grammling, Bill Streckfus

Références

[1] Pressman R. S., Software Engineering: A Practitioner's Approach, Third Édition. McGraw-Hill. Chapitre 4, p. 107, 1992.
[2] http://fr.wikipedia.org/wiki/Architecture_logicielle
[3] Rapport Qualipso « Proposed IPR traking methodology »