Architecture logicielle : quelques éléments

Licence professionnelle IDSE
2012-2013
http://anubis.polytech.unice.fr/iut/2012_2013/lp/idse/gl/management

Mireille Blay-Fornarino
blay@unice.fr

Architecture logicielle

- L'architecture informatique définit la structuration d'un système informatique (i.e. matériel et logiciel) en termes de composants et d'organisation de ses fonctions.

François Trudel, ing., M.Sc.A.
Président Fondateur
francois.trudel@agilexx.com
université de Sherbrook

L'ARCHITECTURE LOGICIELLE EN PRATIQUE

Qu'est-ce qu'une architecture logicielle ?

- Contrairement aux spécifications produites par l'analyse fonctionnelle
  - le modèle d'architecture ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications.
  - L'analyse fonctionnelle décrit le « quoi faire » alors que l'architecture décrit le « comment le faire »

Description d'une architecture logicielle ?

La définition de l'architecture logicielle consiste à:

- Décrire l'organisation générale d'un système et sa décomposition en sous-systèmes ou composants
- Déterminer les interfaces entre les sous-systèmes
- Décrire les interactions et le flot de contrôle entre les sous-systèmes
- Décrire également les composants utilisés pour implanter les fonctionnalités des sous-systèmes
  - Les propriétés de ces composants
  - Leur contenu (e.g., classes, autres composants)
  - Les machines ou dispositifs matériels sur lesquels ces modules seront déployés

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une **vue de haut-niveau** de leur structure et de leurs contraintes. Les **motivation des choix de conception** sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise **l'identification des éléments réutilisables**, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un **plan de haut-niveau** du développement et de l'intégration des modules en mettant en évidence les composants, les *interactions et les dépendances*. Elle doit permettre aux développeurs de **travailler sur des parties individuelles du système en isolation**
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La **séparation composant/connecteur** facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la **cohérence**, **test de conformité**, analyse des **dépendances**
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale

Pourquoi une architecture logicielle [Garlan 2000]

- **Compréhension** : facilite la compréhension des grands systèmes complexes en donnant une vue de haut-niveau de leur structure et de leurs contraintes. Les motivation des choix de conception sont ainsi mis en évidence
- **Réutilisation** : favorise l'identification des éléments réutilisables, parties de conception, composants, caractéristiques, fonctions ou données communes.
- **Construction** : fournit un plan de haut-niveau du développement et de l'intégration des modules en mettant en évidence les composants, les interactions et les dépendances. Elle doit permettre aux développeurs de travailler sur des parties individuelles du système en isolation
- **Évolution** : met en évidence les points où un système peut être modifié et étendu. La séparation composant/connecteur facilite une implémentation du type « plug-and-play»
- **Analyse** : offre une base pour l'analyse plus approfondie de la conception du logiciel, analyse de la cohérence, test de conformité, analyse des dépendances
- **Gestion** : contribue à la gestion générale du projet en permettant aux différentes personnes impliquées de voir comment les différents morceaux du casse-tête seront agencés. L'identification des dépendance entre composants permet d'identifier où les délais peuvent survenir et leur impact sur la planification générale