INF5153 – Génie logiciel : conception

Introduction

Jacques Berger

Objectifs

Introduire la conception et son vocabulaire

Prérequis

Programmation

Cycle de vie

Entre le début du projet de développement et son retrait définitif, un logiciel traverse plusieurs phases

Cycle de vie

Ces phases dépendent du modèle utilisé, mais on retrouve généralement ces étapes :

Extraction des exigences logicielles
Conception
Réalisation
Tests
Maintenance

Exigences

Compréhension du problème

Proposition d'un ensemble de fonctionnalités répondant aux besoins du client

Documentation des exigences fonctionnelles et non fonctionnelles

Acceptation des exigences par le client

Exigences

En résumé : description du fonctionnement du logiciel à produire

Conception

À partir des exigences, on détermine comment devrait être implémenté le logiciel pour répondre aux exigences logicielles

La conception peut être faite avec différents niveaux de granularité, dépendamment du contexte

Conception

Peut inclure une conception architecturale

Peut inclure une conception détaillée
- Classes, dépendances, relations entre les classes, etc.

Conception

En résumé : on détermine comment le logiciel sera réalisé

Réalisation

La programmation du logiciel

À la fin de la réalisation, nous avons un logiciel fonctionnel pouvant être déployé

Maintenance

Tous changements apportés au logiciel après son déploiement

Corrections de bogues

Adaptations et ajouts de fonctionnalités

Maintenance

La maintenance se termine lorsque le logiciel n'est plus supporté

Cycle de vie

Le cycle de vie peut avoir un nombre variable d'étapes selon les modèles

Voir ISO 12207, section 5.2.1

Cycle de vie

Une conception soignée est importante

Une mauvaise réalisation pourrait saboter une excellente conception

Une excellente réalisation ne pourra pas réparer tous les problèmes d'une mauvaise conception

Communication

La conception est une activité de communication

La conception doit être communiquée au travers de l'équipe de développement

Concepteur

La tâche principale du concepteur : trouver la meilleure solution à un problème et décrire comment la réaliser

Concepteur

Le concepteur devra communiquer sa conception à l'équipe de développement

Il devra aussi communiquer avec les analystes d'affaires pour bien comprendre les exigences logicielles

Concepteur

Puisqu'il communique avec les experts du domaine, le concepteur devra développer une certaine connaissance du domaine du logiciel

Il doit être sensible à tous les canaux de communication

La conception

Ce que peut contenir la conception :

La structure statique du système

La structure des données

Les algorithmes à utiliser

La conception

Le regroupement des packages

Les interactions entre les composants du logiciel

Essentiels

Le logiciel doit répondre à 2 nécessités :

- Le logiciel doit être fonctionnel
- Le logiciel doit effectuer le travail attendu

Essentiels

Le reste, bien qu'important, est secondaire

Ceci inclut :

- La taille
- La rapidité
- La facilité à modifier le logiciel
- Etc.

Essentiels

Une conception élégante n'a aucune valeur si le logiciel ne produit pas les résultats attendus

Une bonne conception

Les qualités recherchées d'une bonne conception

Couplage faible
Cohésion forte
Modularité
Réutilisabilité

Couplage

Le couplage est un lien serré entre 2 classes ou 2 composants

Une dépendance est un couplage

Couplage

Le couplage est inévitable, cependant on cherche à le minimiser autant que possible

Techniques pour diminuer le couplage :

- Injection de dépendance
- Design par interfaces
- Etc.

Couplage

Couplage fort : un changement dans une classe entraînera probablement un changement dans les classes qui l'utilisent

Couplage faible : un changement dans une classe n'aura aucun impact sur les classes qui l'utilisent

Couplage

Un couplage faible permet d'effectuer des changements dans le code sans trop avoir d'effets néfastes sur les autres classes

Cohésion

La cohésion est l'étroitesse du lien entre les différentes fonctionnalités d'une même classe

On cherche à maximiser la cohésion dans une classe

Cohésion

Une cohésion forte indique que toutes les fonctionnalités de la classe sont essentielles à son fonctionnement

Une cohésion faible indique que certaines fonctionnalités de la classe n'ont pas vraiment de lien avec les autres

Cohésion

On maximise la cohésion en créant des classes de petite taille n'ayant qu'une seule responsabilité

Modularité

Concevoir l'application en plusieurs blocs modulaires

Chaque module correspond à un sous-ensemble de fonctionnalités reliées

Réutilisabilité

On cherche toujours à réutiliser des composants logiciels

En ayant un couplage faible, une cohésion forte et une bonne modularité, on favorise la création de composants réutilisables

Patron de conception

Une solution à un problème de conception récurrent

Abstraction

L'abstraction est la simplification d'un problème en retirant certains détails de sa description, tout en conservant ses propriétés importantes

La conception utilise beaucoup ce concept

Plus loin...

Software Design, 2nd edition
David Budgen
Addison-Wesley, 2003
Chapitre 1