IFT1025 été 2010

Miklós Csűrös

22 juin 2010

9 Conception de classes

9.1 Cycle de vie

Cycle de vie du logiciel : toutes les activités de développement
Phases principales :

- ⋆ Analyse de tâches, spécification
- ⋆ Conception
- ⋆ Implémentation
- ⋆ Tests
- ⋆ Déploiement

Analyse. Quels sont les besoins du client ? On doit définir les tâches, mais non pas la manière dont les faire. Sortie : document spécifiant les fonctionnalités et les critères de performance.

Conception. Analyse de structure du problème. En OO : classes et méthodes principales. Sortie : spécification de classes et méthodes.

Implémentation. Codage.

Tests. Définis par le client et les développeurs. Sortie : déscription de tests et les résultats.

Déploiment. *Packaging*, installation, distribution.

Idéalement, les phases suivent l'une après l'autre, formant un cascade . . .

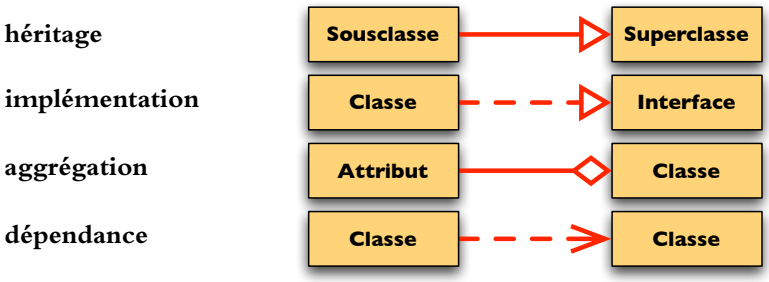
En vérité, il y a beaucoup d'itérations entre les phases (p.e., lors d'implémentation on trouve de meilleures idées de conception, nouvelles tâches) : *modèle de spirale*.

9.2 Conception

Découverte de classes. Une classe devrait représenter un concept en soi (examiner les noms propres dans la déscription de tâches) — **cohésion**.

Découverte de comportement. Extraire les tâches du document (verbes) et assigner les responsabilités aux classes.

Relations entre les classes. Diagramme UML :



Couplage : connectivité du réseau de relations entre les classes (devrait être bas dans un logiciel bien structuré).

9.3 Documentation

Documenter ses classes et ses méthodes à la fin de conception : «squelettes» des fichiers source.

Utiliser `javadoc` : commentaires débutant avec `/**`

Documentation structurée : tags (syntaxe `@nom-de-tag`)

`@param` paramètre d'une méthode
`@return` valeur retournée de la méthode
`@exception` ou **`@throws`** exception lancée par la méthode
`@author` nom du développeur (pour la classe)
`@version` version (classe ou méthode)
`@deprecated` marque la méthode comme dépreciée (avertissement lors de compilation)
`@see` association avec une autre méthode ou classe

Pour toutes les méthodes non-privées : `@param` (chaque paramètre), `@return` (sauf si `void`), `@exception` ou `@throws` (sauf `RuntimeException`)