Génie Logiciel

Robert Strandh

strandh@labri.fr

Laboratoire Bordelais de Recherche en Informatique

(LaBRI)

Organisation

- Mercredi matin : 1h30 de cours, 2h30 de TP
- Vendredi matin : 1h30 de cours, 2h30 de TP

Rappel rapide

Définition du Génie Logiciel

Techniques et méthodes pour développer des logiciels :

- de taille importante,
- avec peu de moyens

de manière à ce que le résultat soit :

- fonctionnel
- de bonne qualité
- économique en ressources

Qualité du logiciel

La *qualité externe* est la qualité visible par l'utilisateur.

Le logiciel doit être :

- correct (conforme à la spécification)
- robuste
- économique en ressource (temps UC, mémoire)

Qualité du logiciel

La *qualité interne* sert à diminuer le coût du développement.

Le logiciel doit être :

- maintenable
- modulaire
- réutilisable

Méthodes traitées ici

Nous allons regarder :

- comment diminuer le coût du développement en écrivant du code plus maintenable
- comment diminuer le coût du développement en écrivant du code plus modulaire
- comment améliorer la productivité du développeur sur des outils existants
- comment choisir des outils plus performants

Maintenance du logiciel

La maintenance représente souvent plus de 75% du coût total du logiciel.

Il y a plusieurs types de maintenance :

- corrective
- adaptative
- évolutive
- préventive

Modularité et types abstrait

Un type abstrait est un type défini par les opérations admises sur une instance du type.

C’est le contraire d’un type concret défini par la disposition en mémoire d’une instance du type.

Types concrets

Exemples de types concrets :

- structure,
- tableau,
- arbre binaire,
- liste chaînée,
- etc.

Types abstraits

- compte bancaire,
- train,
- client,
- adresse,
- etc.

Importance des types abstraits

Le type abstrait permet d'avoir une petite spécification (interface) pour une implémentation importante.

Cela permet au code client de ne pas tenir compte de l'implémentation.

Ce qui permet de maintenir séparément l'implémentation du type.

La réutilisation est également favorisée.

Conteneurs

Plusieurs types abstraits sont destinés à contenir d’autres objets. Il s’agit des *conteneurs*.

Conteneurs

Exemple de conteneurs :

- pile (empiler, dépiler, vide ?),
- file (enfiler, défiler, vide ?),
- dictionnaire (insérer, supprimer, membre ?),
- file de priorité (insérer, supprimer min, vide ?),
- etc.

Difficulté des types abstraits

La notion de type abstrait nécessite un langage de programmation capable d'exprimer des types paramétrés. On parle également de *généricité*.

Plusieurs mécanismes dans un langage moderne existent pour exprimer la généricité (héritage, templates, etc).

En C, le seul mécanisme à notre disposition est le pointeur générique (`void *`).

Autre difficulté

La gestion de la mémoire est un problème.

Il faut savoir qui est chargé d'allouer de la place pour un objet et qui est chargé de la libérer.

Le problème arrive lorsque le même objet est référencé depuis plusieurs autres.

Exemple de type abstrait

Exemple de type abstrait : `dictionnaire`

TP : Modularité

Makefile

Permet de recompiler seulement les fichiers modifiés d’un logiciel, ou d’un document arbitraire.

La commande ’make’ lance les commande dans le Makefile.

Makefile

L'option `-MM` à gcc permet d'automatiser les dépendances.

CVS

Permet de gérer les versions d'un ensemble de fichiers "sources".

Capable de résoudre des conflits.

Marche à travers un réseau.

Ramasse-miettes

TP : Modularité + Makefile