INF3135

Construction et maintenance de logiciels

Hiver 2005 (groupe 20)

> *Software construction is a fundamental act of software engineering: the construction of working, meaningful software through a combination of coding, validation, and test (unit testing) by a programmer* [BGM01].

Professeurs: Vladimir Makarenkov (#3870) et Aziz Salah (#1485)

Courriels: makarenkov.vladimir at uqam.ca et salah.aziz at uqam.ca

Démonstrateur: Julien Gauthier – **courriel**: julg at sympatico.ca

Sites web du cours : http://www.labunix.uqam.ca/~makarenv/INF3135.html
et http://www.labunix.uqam.ca/~salah/INF3135.html

Horaires du cours :

• **Cours théorique**: Mardi et Jeudi, 09:00–10:30 au SH-2140

• **Laboratoire**: Jeudi 11:00–13:00 au PK-S570

Description (*selon l'annuaire*)

Initier les étudiants à la programmation à l'aide d'un langage impératif et procédural. Familiariser les étudiants à la construction professionnelle de logiciels et à leur maintenance.

Notions de base de la programmation procédurale et impérative en langage C sous environnement Unix/Linux (définition et déclaration, portée et durée de vie, fichier d'interface, stuctures de contrôle, unités de programme et passage des paramètres, macros, compilation conditionnelle). Décomposition en modules et caractéristiques facilitant les modifications (cohésion et couplage, encapsulation et dissimulation de l'information, décomposition fonctionnelle). Style de programmation (conventions, documentation interne, gabarits). Déboggage de programmes (erreurs typiques, traces, outils, par ex., gdb). Assertions et conception par contrats. Tests (unitaires, intégration, d'acceptation, boite noire vs. boite blanche, mesures de couverture, outils d'exécution automatique des tests, par exemple, xUnit, scripts). Évaluation et amélioration des performances (profils d'exécution, améliorations asymptotiques vs. optimisations, outils). Techniques et outils de base pour la gestion de la configuration (par exemple, make, cvs). Introduction à la maintenance de logiciels (types de maintenance, techniques de base, par exemple, remodelage, automatisation des tests de régression).

Ce cours comporte une séance obligatoire de laboratoire (2 heures).

Objectifs

Généraux : Ce cours vise à introduire les étudiants à la construction professionnelle de logiciels. Il vise aussi à familiariser les étudiants avec la programmation procédurale et impérative.

Spécifiques : À la fin du cours, l'étudiant devrait être capable :

• de développer et modifier des composants logiciels écrits dans un langage impératif et procédural;

• de bien maitriser le langage C et le compilateur du C sous UNIX/Linux;

• d'utiliser les notions de module, de cohésion et couplage, de complexité structurale, de dissimulation de l'information, etc., pour évaluer la qualité d'un composant logiciel;

• d'expliquer et d'utiliser les principales techniques de modularisation : décomposition fonctionnelle, dissimulation de l'information, filtres et pipelines;

• d'utiliser des assertions (pré/post-conditions, invariants) pour documenter des composants logiciels et assurer leur bon fonctionnement;

• de faire une inspection formelle de code pour détecter les erreurs ou le code de mauvaise qualité;

• de déboguer un programme à l'aide de techniques, stratégies et outils appropriés;

• de vérifier le bon fonctionnement d'un composant logiciel à l'aide de tests fonctionnels et structurels, et d'évaluer à l'aide des notions et outils appropriés la qualité des tests (par ex., complexité cyclomatique, mesures de couvertures des tests);

• d'évaluer de façon empirique à l'aide d'outils appropriés (par ex., profils d'exécution) les performances d'un composant logiciel de façon à pouvoir, si nécessaire, en améliorer les performances;

• d'utiliser divers outils (outil de gestion de configuration, fichiers makefile [LO97], langage de scripts) pour organiser le développement de programmes comportant plusieurs composants ou modules;

• d'expliquer les notions de base de la maintenance des logiciels et d'appliquer certaines techniques de maintenance (tests de régression exécutés automatiquement, remodelage de programmes).

Évaluation

Examens:

• Examen intra : 30 %

• Examen final : 40 %

Travaux pratiques:

• Trois (3) travaux pratiques: 30 %

L'utilisation de notes de cours personnelles et de matériel provenant du site *web* du cours est permise aux examens. Par contre, l'utilisation de livres ou manuel ne l'est pas. L'examen final a lieu le jeudi 28 avril 2005. Une moyenne d'au moins 50% aux examens est exigée pour réussir le cours.
Les travaux pratiques peuvent être réalisés individuellement ou en équipe de deux (2) personnes. Par contre, les examens doivent évidemment être faits de façon individuelle. Une pénalité de 10% par jour de retard sera appliquée pour la remise des travaux pratiques. La qualité du français sera prise en considération (jusqu'à 10 % de pénalité).

Il n'y a pas de reprise d'examen s'il y a absence aux dates prévues. Un étudiant absent à un examen se verra normalement attribuer la note zéro pour cet examen. Cependant, une attestation d'un médecin en bonne et due forme, présentée au plus tard deux semaines après l'examen et confirmant que l'étudiant était dans l'impossibilité de se présenter à l'examen pour des raisons de santé pourra être considérée comme une justification d'absence valable. L'attestation du médecin traitant doit obligatoirement être complétée sur le formulaire du Département d'informatique prévu à cette fin.

Dates importantes

Examen intra : jeudi le 10 mars 2005 de 9:00 à 12:00 au SH-3720 et SH-3760
Examen final : jeudi le 28 avril 2005 de 9:00 à 12:00 au SH-3720 et SH-3760

Date limite pour la remise du TP1 : jeudi le 17 février 2005
Date limite pour la remise du TP2 : jeudi le 17 mars 2005
Date limite pour la remise du TP3 : jeudi le 21 avril 2005

Calendrier (par semaine)

1. Présentation du plan de cours. Langage C : présentation générale, noms des variables et fonctions, instruction for, variables globales et visibilité.

2. Langage C (suite) : types de base, conversions, expressions et priorités des opérateurs, expressions conditionnelles, boucles, fonction main. Style de programmation.

3. Langage C (suite) : variables externes et statiques, préprocesseur, pointeurs et tableaux.
Librairies standard et non-standard (exemple : bibliothèque CII) du langage C.

4. Débogage. Langage C (suite) : tableaux multi-dimensionnels, allocation dynamique de mémoire, arguments du programme.

5. Gestion de la configuration : cvs, make. Langage C (suite) : structures, typedef, union.

6. Outils Unix. Langage C (suite) : listes chaînées, arbres binaires, fichiers et entrées/sorties standards.

7. Révision pour l'examen Intra.

8. Semaine de relache.

9. *Examen Intra - le jeudi 10 mars de 9:00 à 12:00.*

10. Programmation défensive et assertions.

11. Tests.

12. Tests (suite). Conception.

13. Conception (suite). Évaluation des performances.

14. Évaluation des performances (suite). Maintenance.

15. Révision pour l'examen final.

16. *Examen final - le jeudi 28 avril de 9:00 à 12:00.*

Note: Les cours des semaines 1, 2, 3, 11 (Conception), 12 et 13 seront enseignés par M. Vladimir Makarenkov. Les cours des semaines 4, 5, 6, 9, 10, 11 (Tests) seront enseignés par M. Aziz Salah. Les séances de révision et les examens (semaines 7, 8, 14 et 15) seront assurées de façon conjointe par les deux enseignants. Notez que cette répartition des cours est approximative et peut être modifiée.

Bibliographie

Les références essentielles sont mis en gras :

[Bec03] K. Beck. *Test-Driven Development — By Example*. Addison-Wesley, 2003.

[BGM01] T. Bollinger, P. Gabrini, and L. Martin. *Guide to the Software Engineering Body of Knowledge (Trial Version)*, chapter 4. Software Construction, pages 53–67. IEEE Computer Society Press, 2001.

[Bla04] C. Blaess. *Scripts sous Linux* — Shell Bash, Sed, Awk, Perl, Tcl, Tk, Python, Ruby.... Eyrolles, 2004. [QA76.76O63B4899.2004].

[Han97] D. R. Hanson. *C* Interfaces and Implementations — Techniques for Creating Reusable Software. Addison-Wesley, 1997.

[HT00] A. Hunt and D. Thomas. *The Pragmatic Programmer — From journeyman to master*. Addison-Wesley, 2000. [QA76.6H858].

[KP99] B. W. Kernighan and R. Pike. *The Practice of Programming*. Addison-Wesley, 1999.

[KP01] B. W. Kernighan et R. Pike. *La programmation — En pratique*. Vuibert, 2001. [QA76.6K48814].

[KR90] B. W. Kernighan et D. M. Ritchie. *Le langage C*, deuxième édition, Masson, Paris, 1990, 280 pp.

[LG01] B. Liskov and J. Guttag. *Program Development in Java*. Addison-Wesley, 2001.

[LO97] M. Loukides and A. Oram. *Programming with GNU* Software. O'Reilly, 1997.

[Mag95] S. Maguire. *L'art du code*. Microsoft Press, 1995.

[McC04] S. McConnell. *Code Complete — A Practical Handbook of Software Construction (Second Edition)*. Microsoft Press, Redmond, WA, 2004.

[PJ88] M. Page-Jones. *The Practical Guide to Structured Systems Design, Second edition*. Prentice-Hall, 1988.

[Ray04] E. S. Raymond. *The Art of UNIX* Programming. Addison-Wesley, 2004.

[Som01] I. Sommerville. *Software Engineering (Sixth Edition)*. Addison-Wesley, 2001.

[Spi03] D. Spinellis. *Code Reading — The Open Source Perspective*. Addison-Wesley, 2003.

[TG96] A. A. Takang and P. A. Grubb. *Software Maintenance — Concepts and Practice*. International Thomson Computer Press, 1996. [QA76.76S64T25].

[TH04] D. Thomas and A. Hunt. *Pragmatic Version Control Using CVS*. The Pragmatic Bookshelf, 2004.

Remarques générales sur les travaux pratiques

Les travaux pratiques devront être réalisés individuellement ou en équipe de deux. Si vous aimez travailler avec des collègues surtout faites-le, on apprend généralement mieux en s'expliquant mutuellement ce que l'on comprend, en cherchant ensemble ce que l'on ne comprend pas. Pour être acceptés, les travaux doivent être remis à la date prévue, le jour du laboratoire, en début de séance de laboratoire. Les dates de remise sont spécifiées ci-dessus. Les travaux pratiques doivent être remis au **démonstrateur**. N'oubliez pas d'inscrire votre **nom** et votre **nom d'usager** sur le document remis au démonstrateur. Vous devez remettre au démonstrateur un **listing documenté du programme** (la qualité de la documentation sera un critère d'évaluation important). Si vous avez travaillé en équipe de deux, le nom et le nom d'usager de chaque étudiant doivent apparaître sur la copie remise.

Les **exécutables** (les commandes compilées) pour les travaux pratiques doivent être crées sur le serveur **arabica** (Sun). Si vous avez travaillé en équipe de deux, un seul étudiant doit envoyer l'exécutable et le code source par intermédiaire de la commande **rendre_tp** (voir le manuel pour cette commande – *man rendre_tp* – disponible sur le serveur arabica). Les exécutables pour les TP1, TP2 et TP3 doivent s'appeler respectivement TP1.out*, TP2.out*, TP3.out*. Les fichiers source pour les TP1, TP2 et TP3 doivent s'appeler respectivement TP1.c, TP2.c, TP3.c. Tous ces fichiers doivent être envoyés au répertoires suivants : /usagers/inf3135/tp1 pour le TP1, /usagers/inf3135/tp2 pour le TP2 et /usagers/inf3135/tp3 pour le TP3.

EXEMPLE :

Disons que votre nom d'usager est ab778899. Pour nous remettre votre tp1 les commandes suivantes doivent être exécutées :

rendre_tp /usagers/inf3135/tp1 /usagers/ab778899/TP1.c

rendre_tp /usagers/inf3135/tp1 /usagers/ab778899/TP1.out

Pour vérifier si votre TP1 a été bien remis faites :

ls /usagers/inf3135/tp1/ab778899.TP1.c

ls /usagers/inf3135/tp1/ab778899.TP1.out

Les commentaires du code et les autres textes doivent être en français. La qualité du français constitue un critère d'évaluation pour un maximum de 10% de la note. Les cas de plagiat seront référés au comité de discipline. Format des rapports de laboratoire :

1. Décrire les objectifs du travail,
2. Faire la description des algorithmes utilisés,
3. Pour chaque fonction indiquer clairement son but, les entrées et les sorties et leurs types respectifs,
4. Présenter des exemples d'exécution,
5. Donner les références si besoin est,
6. Fournir le listing (code source) détaillé et abondamment commenté des programmes.

Rappelez-vous qu'il va de la rédaction de rapports comme des vins de la SAQ: la modération a bien meilleur goût! Alors soyez brefs et concis. Le format du code est 80 caractères maximum par ligne, sans débordement de ligne; le code doit être imprimé avec une police non proportionnelle, (i.e. une police de caractères de largeur constante, i.e. une police moderne) avec marges, en-têtes et pieds de page.