UNIVERSITÉ DE SHERBROOKE

Département d'informatique

IFT 719 – Processus du génie logiciel

Plan de cours
Automne 2013

Enseignant	**Luc Lavoie**	
	Courriel :	luc.lavoie@usherbrooke.ca
	Local :	D4-2006
	Téléphone :	(819) 821-8000 poste 62015
	Site :	http://info.usherbrooke.ca/llavoie
	Disponibilité :	sur rendez-vous
Horaire		lundi 14:30 à 16:20 (suggestion)
		mercredi 08:30 à 10:20 (suggestion)

Description officielle de l'activité pédagogique[1]

Objectifs : Effectuer l'analyse du processus même de développement des logiciels; utiliser et appliquer les techniques de réingénierie et de réutilisation.

Contenu : Bref aperçu des approches et des normes du développement de logiciels. Étude de quelques cycles de base de développement de logiciels par le paradigme de décision/justification. Illustration sur des exemples. Approches de réingénierie et de rétro-ingénierie des logiciels : limites et perspectives. Techniques de réutilisation des logiciels. Environnements et ateliers de développement assisté des logiciels. Études de cas.

Crédits : 3

Organisation : 3 heures d'exposé magistral par semaine
6 heures de travail personnel par semaine

Particularités : Aucune

Professeur responsable

Luc Lavoie

[1] http ://www.usherbrooke.ca/programmes/cours/IGL/igl601.htm

1 Présentation

Cette section présente les objectifs spécifiques et le contenu détaillé de l'activité pédagogique.

1.1 Mise en contexte

Le génie logiciel traite de la configuration d'une machine universelle (ordinateur) dans le but d'atteindre un objectif spécifique. Le logiciel de configuration peut lui aussi être vu comme une machine, mais il diffère des autres machines en ce sens qu'il est intangible. Le génie logiciel doit son nom et sa constitution comme un domaine de connaissance propre à la tenue d'un séminaire organisé par l'OTAN à Garmisch-Partenkirchen en Allemagne en 1968.

Le logiciel de configuration d'une machine universelle est désigné sous plusieurs appellations différentes, selon la caractéristique mise de l'avant : logiciel (intangibilité), programme (déterminisme), système (complexité).

Puisqu'on construit généralement un système pour atteindre un but donné, il est préférable de déterminer et de détailler d'abord quel est ce but. Ce qui nous amène à l'ingénierie des exigences, la partie du génie logiciel qui permet de déterminer quel système sera développé.

L'élaboration d'une architecture logicielle découle de cette première étape. Elle a pour but de délimiter les unités organiques qui permettront de réaliser les fonctions du logiciel de façon efficace, traçable et modifiable (pour ne nommer que quelques-unes des propriétés recherchées). Même si dans certains cas il est possible d'induire l'architecture de la spécification formelle, la vérifiabilité et la validité du produit résultant nécessitent un examen indépendant. Voilà une troisième étape.

Les composants organiques doivent communiquer entre eux et les humains interagir avec certains d'entre eux. L'ingénierie des interfaces machine-machine et des interfaces personne-machine intervient alors dans une quatrième étape.

La conception globale et la conception détaillée, souvent regroupées sous la désignation de programmation, visent à construire effectivement chacun des composants organiques et des composants d'interface. La conception globale s'intéresse aux algorithmes, aux structures de données et à la complexité algorithmique des solutions proposées pour réaliser les composants logiciels. La conception détaillée s'intéresse à l'expression des solutions élaborées lors de la conception globale à l'aide d'un formalisme particulier, le plus souvent un langage de programmation. Parfois considérées comme deux étapes distinctes, parfois comme une seule étape, la conception globale et la conception détaillée permettent la concrétisation du logiciel effectif qui devient alors produit.

Les frontières entre chacune de ces étapes ne sont pas arbitraires, mais ne sont pas toujours tranchées. L'ordre dans lequel elles sont réalisées peut varier considérablement selon qu'il s'agit d'un projet prédictif ou expérimental. L'agencement des activités et des tâches en phases, en vagues, en cascades, en itérations ou en spirales nécessite de nombreuses adaptations. Plusieurs procédés ont donc vu le jour qui, lorsqu'ils sont contextualisés, déterminent les processus de développement effectivement utilisés.

Ces processus de développement sont le plus souvent réalisés dans le contexte d'un projet dont la gestion nécessite elle-même plusieurs processus (gestion de portée, de durée, de cout, de qualité, des ressources humaines, des risques, etc.). L'interaction entre tous ces processus est alors déterminante pour le succès du projet et, ultimement, du produit.

Par ailleurs, les projets de génie logiciel se distinguent d'autres types de projets notamment par l'importance qu'y occupent les processus d'estimation, de gestion des configurations, de vérification et de validation. Plusieurs techniques utilisées au sein de ces processus sont aussi propres au génie logiciel.

La réalisation de produits d'envergure passe donc par la maitrise de (tous) ces processus, de leurs techniques et de leur planification.

Le cours s'intéresse cependant principalement aux processus techniques, liés au développement proprement dit du logiciel.

1.2 Objectifs spécifiques et compétences

Objectifs spécifiques

1. Choisir, motiver, adapter et planifier les processus requis par le développement logiciel.
2. Optimiser les relations entre les différents processus logiciels.
3. Contrôler, superviser et maitriser les processus du développement logiciel.

Compétences

A. Déterminer les procédés adéquats à la réalisation d'un projet de développement logiciel.
B. Définir les processus de développement et les harmoniser.
C. Piloter les activités de génie logiciel au sein d'un projet.
D. Faire le suivi et l'analyse des activités dans une perspective d'amélioration continue.

1.3 Contenu détaillé

Le contenu du cours peut être revu pour cibler certaines techniques particulières en fonction des intérêts et des acquis des étudiants inscrits.

Tableau 1 – Contenu détaillé de l'activité

N°	Contenu	Heures[2]
1.	Propostion d'un cadre général aux processus de développement logiciel	3
2.	Processus de développement	9
3.	Processus de vérification-validation	9
4.	Processus de gestion des anomalies	3
5.	Processus de gestion des configurations	3
6.	Processus de gestion documentaire	3
7.	Intégration des processus	3
8.	Contrôle et supervision des processus	3
9.	Études de cas	3
	Total	39

2 Organisation

Cette sectionprésente la méthode pédagogique, le calendrier, le barème et la procédure d'évaluation ainsi que l'échéancier des travaux.

2.1 Méthode pédagogique

L'activité se déroulera en deux temps. Dans le premier temps, des exposés magistraux permettront de couvrir la plus grande partie des connaissances théoriques et méthodologiques. Dans la deuxième partie, un projet supervisé permettra de développer les compétences fondées sur ces connaissances. Le projet est réalisé en équipes de trois à six personnes formées par l'enseignant. Chaque équipe est dirigée par un étudiant inscrit à l'activité IFT 719 qui sera responsable de la conduite du projet et de l'équipe. Les autres membres de l'équipe proviendront d'autres activités concomitantes offertes par le Département d'informatique, notamment IGL 601, IFT 592 et IGE 592.

2.2 Calendrier

Le calendrier est structuré selon une répartition des heures de cours à raison de quatre heures par semaine avant la relâche et de deux heures par semaine après de façon à libérer du temps pour la

[2] Répartion des heures de cours, les heures de travail personnel sont intégrés au projet.

réalisation du projet. Au besoin, cette modulation peut être adaptée en fonction des besoins d'approfondissement exprimés par les étudiants et des impératifs de projet.

Tableau 2 – Planification des activités et des lectures

N°	Semaine	Activités	Contenu	Travaux
1.	2013-08-26	cours	1	
2.	2013-09-02	cours[3]	2	
3.	2013-09-09	cours	2	
4.	2013-09-16	cours	2, 3	
5.	2013-09-23	cours	3	
6.	2013-09-30	cours	3	
7.	2013-10-07	atelier	--	définition du projet
8.	2013-10-14	relâche	--	projet
9.	2013-10-21	cours + projet	4	projet
10.	2013-10-28	cours + projet	5	projet
11.	2013-11-04	cours + projet	6	projet
12.	2013-11-11	cours + projet	7	projet
13.	2013-11-18	cours + projet	7, 8	projet
14.	2013-11-25	cours + projet	8	projet
15.	2013-12-02	présentation des projets	9	projet
16.	2013-12-09	clôture des projets	--	bilan du projet
17.	2013-12-16	examen	--	examen

2.3 Évaluation

En plus d'un examen, l'évaluation porte sur un projet réalisé au cours de la deuxième partie de l'activité.

Les examens sont placés sous la responsabilité de la Faculté des sciences et organisés par elle. La durée de l'examen est de trois heures. La documentation personnelle (notes et manuels de cours) est permise; l'usage d'appareils informatiques, électroniques ou de communication (ordinateur, calculatrice, téléphone, etc.) est interdit sauf si l'examen a lieu dans un laboratoire auquel cas seuls les équipements du laboratoire peuvent être utilisés.

Tableau 3 – Sommaire des évaluations

Évaluation	Valeur	Commentaire
examen	50 %	individuel
projet	50 %	équipe de trois à six personnes, livrables individuels
Total	100 %	

Tout étudiant, toute étudiante, qui omet de remettre un travail au moment prescrit doit rencontrer l'enseignant afin de déterminer une nouvelle date de remise. Dans tous les cas, une pénalité de 10 % par jour de retard est imposée.

L'évaluation est faite en tenant compte de la clarté des documents et du respect de la méthodologie du génie logiciel. Conformément à l'article 17 du Règlement facultaire d'évaluation des apprentissages[4], l'enseignant peut retourner à l'étudiante ou à l'étudiant tout travail non conforme aux exigences quant à la qualité de la langue et aux normes de présentation. Toute situation de plagiat sera traitée en conformité avec le Règlement des études[5] de l'Université de Sherbrooke, notamment l'article 8.1.2.

[3] Il n'y a pas cours le lundi 2 septembre, mais il y a cours le lundi 9 décembre.

[4] http://www.usherbrooke.ca/sciences/fileadmin/sites/sciences/documents/Intranet/Informations_academiques/evaluation_apprentissages.pdf

[5] http://www.usherbrooke.ca/programmes/etude

En cas de circonstances extraordinaires au-delà du contrôle de l'Université de Sherbrooke et sur décision de celle-ci, l'évaluation des apprentissages de cette activité est sujette à changement.

2.4 Échéancier des travaux

Le projet doit être présenté sous la forme d'un séminaire lors la dernière semaine de cours et les livrables du projet doivent être remis sous leurs formes définitives au plus tard le dernier jour du trimestre. Les modalités de remises seront établies dans l'énoncé qui sera remis avant la semaine de relâche.

3 Matériel nécessaire pour le cours

Le plan de cours et les présentations utilisées en cours sont disponibles sur le site de l'enseignant[6]. Plusieurs lectures sont tirées de manuels en référence et d'articles proposés en cours. Le tout est disponible à la bibliothèque des sciences et de génie.

4 Références

[Boehm2003]
Barry BOEHM, Richard TURNER;
Balancing Agility and Discipline;
Addison-Wesley, 2003. ISBN 0-321-18612-5

[Braude2011]
Eric J. BRAUDE, Michael E. BERNSTEIN.
Software engineering : modern approaches.
2nd edition, John Wiley & sons, 2011, ISBN 978-0-471-69208-9.

[Bray2002]
Ian K. BRAY.
An Introduction to requirements engineering.
Addison-Wesley, 2002; ISBN 0-201-76792-9; [Cote UdeS QA 76.758 B744, 2002].

[Buckley2004]
Fletcher J. Buckley;
Implementing configuration management : hardware, software, and firmware.
Cote : QA 76.76 C69B83 1996

[Contantinidis2011]
Yves CONTANTINIDIS.
Expression des besoins pour le système d'information.
Eyrolles, 2011. ISBN 978-2-212-12783-6.

[Craig2002]
Rick D. CRAIG ;
Systematic software testing.
Cote : QA 76.76 T48C73 2002

[Davis2007]
M. DAVIS.
Requirements Bibliography.
http://web.uccs.edu/adavis/UCCS/reqbib.htm (consulté le 2007-03-15).

[De Lucia2008]
Andrea DE LUCIA, Filomena FERRUCI, Genny TORTORA, Maurizio TUCCI (ed).
Emerging Methods, Technologies, and Process Manafement in Software Engineering.
John Wiley and sons, 2008. ISBN 978-0-470-08571-4.

[Fairley2009]
Richard E. (Dick) FAIRLEY
Managing AND Leading Software Projects
Wiley, 2009, ISBN 978-0-470-29455-0.

[6] http://info.usherbrooke.ca/llavoie/enseignement/IGL601

[Fournier1993]
Jean-Pierre FOURNIER;
Fiabilité du logiciel : concepts, modélisations, perspectives.
Cote : QA 76.76 R44F68 1993

[GDT]
Grand dictionnaire terminologique.
Office québécois de la langue française.
http://www.grandictionnaire.com (consulté le 2011-12-15).

[Gilb1993]
Tom GILB, Dorothy GRAHAM;
Software inspection.
Cote : QA 76.76 Q35G48 1993

[GLOGUS]
GROUPE Μῆτις
GLOGUS – Recueil de modèles de documents pour le développement logiciel.
http://info.usherbrooke.ca/llavoie/glogus.php
Département d'informatique, Faculté des sciences, Université de Sherbrooke, janvier 2012.

[Hull2011]
Elizabeth HULL, Ken JACKSON, Jeremy DICK.
Requirements Engineering.
3 rd edition, Springer, 2011; ISBN 978-1-84996-405-0.

[IEEE]
Institute of Electrical and Electronics Engineers ;
IEEE software engineering standards collection.
[cédérom] Cote : QA 76.758 I33 2003
les versions les plus récentes sont disponibles au format PDF par le biais du portail du Service des bibliothèques[7] et ce, gratuitement pour les étudiants inscrits à l'Université de Sherbrooke.

[IEEE12207]
Industry Implementation of International Standard ISO/IEC 12207-1995.
IEEE Std 12207.0-1996, IEEE, New York, 1998.

[IEEE1233]
IEEE Guide for Developing System Requirements Specifications.
IEEE Std 1233-1998, IEEE, New York, 1998.

[IEEE830]
IEEE Recommended Practice for Software Requirements Specifications;
IEEE Std 830-1998, IEEE, New York, 1998.

[IGE 401]
Luc LAVOIE ;
IGE 401 – Gestion de projet, Notes complémentaires et synthétiques,
http://info.usherbrooke.ca/llavoie/enseignement/IGE401
Département d'informatique, Faculté des sciences, Université de Sherbrooke, janvier 2012.

[IGL 301]
COLLECTIF GL.
IGL 301 – Spécification et validation des exigences (notes complémentaires et synthétiques).
http://info.usherbrooke.ca/llavoie/enseignement/IGL301
Département d'informatique, Faculté des sciences, Université de Sherbrooke, janvier 2011.

[IGL 601]
Luc LAVOIE;
IGL601 – Présentations et notes de lecture.
http://info.usherbrooke.ca/llavoie/enseignement/IGL601
Département d'informatique, Faculté des sciences, Université de Sherbrooke, septembre 2013.

[INF 755]
Luc LAVOIE.
INF 755 – Méthodes d'analyse et de conception (notes complémentaires et synthétiques).
http://info.usherbrooke.ca/llavoie/enseignement/INF755
Département d'informatique, Faculté des sciences, Université de Sherbrooke, janvier 2012.

[7] http://www.usherbrooke.ca/biblio/trouver/banques-de-donnees/

[Jackson1995]
Michael JACKSON.
Software Requirements & Specifications.
Addison Wesley, 1995; ISBN 0-201-87712-0; [UdeS QA 76.76 D47J33, 1995].
[Jackson2001]
Michael JACKSON.
Problem frames.
ACM Press Book, Addison Wesley, 2001; ISBN 0-20159 627-X; [UdeS QA 76.76 D47J32, 2001].
[Jacobson1994]
Ivar JACOBSON;
Object-Oriented Software Engineering;
ACM Press Book, Addison Wesley, 1994; ISBN 0-201-54435-0.
[Jacobson1999] (traduit en français, voir [Jacobson2000])
Ivar JACOBSON, Grady BOOCH, James RUMBAUGH.
The unified software development process.
Addison-Wesley, 1999; ISBN 0-201-57169-2.
[Jacobson2000] (traduction de [Jacobson1999])
Ivar JACOBSON, Grady BOOCH, James RUMBAUGH.
Le processus unifié de développement logiciel.
Eyrolles, 2000; ISBN 2-212-09142-7; [UdeS 76.76 D47J3514 2000].
[Jacquin1996]
Dominique JACQUIN;
Maîtrisez votre gestion des configurations logicielles : une étape pour la certification ISO 9000.
Cote : QA 76.76 C69J32 1996
[Jalote]
Pankaj JALOTE.
Software Project Management in Practice.
Addison Wesley Professional, 2002.
ISBN 2-7440-1432-X; UdeS QA 76.76 D47J3714 2002 .
[Joliot2012]
Didier JOLIOT.
La conception des avant-projets.
Collection Lavoisier, Hermes, 2012. ISBN 978-2-74623823-7.
[Keyes2004]
Jessica KEYES;
Software configuration management.
[Cote UdeS QA 76.76 C69K49 2004].
[Kotonya1998]
G. KOTONYA and I. SOMMERVILLE;
Requirements engineering: processes and techniques;
John Wiley, 1998; [Cote UdeS QA 76.758 K67 1998].
[Kovitz1998]
B. L. KOVITZ;
Practical Software Requirements: A Manual of Content and Style;
Manning Publications Company, 1998; [QA 76.76 D47K68 1999].
[Kuhn2013]
D. Richard KUHN, Raghu N. KACKER, Yu LEI;
Introduction to Combinatorial Testing;
CRC Press, 2013; ISBN 978-1-4665-5229-6
[Kulak2002]
D. KULAK, E. GUINEY;
Use Cases : Requirements in Context;
2/E, Addison Wesley Professional, 2004.

[Larman2005]
Craig LARMAN.
Applying UML and patterns - an introduction to object-oriented analysis and design and iterative development.
3 rd edition, Prentice-Hall, Upper Sadel River (NJ), 2005;
ISBN 0-13-148906-2 [UdeS QA 76.9 O35L3714, 2005].

[Lauesen2002]
S. LAUESEN;
Software Requirements : Styles and Techniques;
Addison Wesley Professional, 2002; [QA 76.754 L38 2002].

[Leffingwell2003]
D. LEFFINGWELL, D. WIDRIG.
Managing software requirements – A use case approach.
2nd edition, Addison-Wesley, 2003, ISBN 0-321-12247-X; [UdeS QA 76.76 D47L44, 2003].

[Mikkelsen]
Tim MIKKELSEN, Suzanne PHERIGO ;
Practical software configuration management : the latenight developer's handbook.
Cote : QA 76.76 C69M55 1997

[Pezzè2008]
Mauro PEZZÈ, Michal YOUNG ;
Software testing and analysis – Process, principles, and techniques.
John Wiley and sons, 2008.
ISBN-13 : 978-0-471-45593-6

[Pfleeger2010]
Shari Lawrence PFLEEGER, Joanne M. ATLEE.
Software Engineering – Theory and Practice.
4 th edition, Prentice Hall, 2005; ISBN 978-0-13-606169-4.

[Pressman2005]
Roger S. PRESSMAN.
Software Engineering — A practioner's Approach.
6th edition, McGraw-Hill, 2005; ISBN 0-07-301933-X; [UdeS QA 76.758 P73, 2005].

[Printz2012]
Jacques PRINTZ.
Architecture logicielle — Concevoir des applications simples, sûres et adaptables.
Dunod, 3e édition, 2012; ISBN 978-2-10-057865-8.

[Sommerville2011]
Ian SOMMERVILLE.
Software Engineering.
9th edition, Addison-Wesley, 2011; ISBN 978-0-13-703515-1.

[VanVliet2008]
Hans VAN VLIET.
Software Engineering — Principles and Pratice.
3rd edition, Wiley, 2008; ISBN 978-0-470-03146-9; [UdeS QA 76.758 V55, 2008].

[WOL2004]
Wall-On-Line : l'e-gouvernement wallon.
La boîte à outils : 15 méthodes d'implication des utilisateurs.
http://egov.wallonie.be/boite_outils_methodes/index.htm
(version en date du 17 décembre 2004 consultée le 11 mai 2007, disponible maintenant sous http://info.usherbrooke.ca/llavoie/projets/GLOGUS/wall-on-line.pdf)

[Wood1995]
Jane WOOD, Denise SILVER.
Joint Application Development.
2nd edition, John Wiley & sons, 1995. ISBN 0-471-04299-4.

[Xanthakis2000]
Spyros XANTHAKIS, Pascal RÉGNIER, Constantin KARAPOULIOS;
Le test des logiciels.
Cote : QA 76.76 T48X36 2000