COURSE SYLLABUS

CMPT 846: Software Maintenance and Evolution

Catalogue Description:

Studies show that up to 80% of total effort is spent on software maintenance. In practice, software engineers spend busy time in modifying software systems to meet user requirements, fixing bugs and defects or adapting the systems to changing needs. It is also a common practice to reuse existing software components for building new systems rather than building from scratch. Over the last decades, research is therefore aimed at the rapid evolution of software systems in response to business and technological change. Using automated techniques, software evolution and adaptation can be made more cost effective, more timely and less error prone, helping software industries be more competitive and better able to quickly react to changing market demands while addressing critical software quality issues.

This course aims to make students aware of the challenges inherent in the maintenance and evolution of software systems, and to provide a working understanding of some of the techniques and best practices currently in use for changing software safely, efficiently and in a cost effective way.

Class Time & Location:	Mondays, 9: 30 AM -12:20 PM
Website:	Moodle and Dropbox

Instructor Information

Instructor:	Dr. Chanchal Roy
Contact:	Email**:** chanchal.roy@usask.ca
	Office Phone: 966-4163
Office Hours:	Location**:** Thorv 205A
	Hours**:** By appointment and after lecture.

Course Objectives and Course Contents

In this course, we focus on state-of-the-art methods, tools, and techniques to assist the maintenance, understanding and restructuring of very large software systems. Topics include:

- **State of the art tools for supporting software developers and maintenance engineers**
- Program Comprehension
- Software Clone Detection and Analysis

- Mining Software Repositories
- Design Recovery
- Traceability
- Refactoring
- Locating features in source code, concept analysis
- Program Transformation and Migration
- Software Evolution Process Models
- Software Testing during Maintenance and Evolution
- Software Metrics for Maintenance
- Software Reuse
- Maintenance and Evolution of Services Systems
- **Focused on Big Data Analytics for Software Engineering**

Learning Outcomes

At the end of the course, all the students are expected to meet the following learning goals of the course:

- The students are expected to have an introductory but solid ideas on different topics of software maintenance and evolution as outlined above
- The students are expected to frame any research problems in depth, generate possible hypothesis, outline potential methodologies and evaluation strategies, and draw possible conclusions.
- Given a chosen research problem the students are expected to write a well-formatted research proposal, conduct large scale empirical studies, discuss the findings and evaluate the proposed problem and then be able to write the details in the form of a research paper.
- The students are expected to know the most of the latest related work in the area of the proposed research.
- The students are expected to learn of how to present a research work to the participants, focusing on different factors such as what is the problem, why this is a problem, what are potential factors behind it, what could be possible methodologies along with the empirical findings.
- The students are expected to learn of how to critique/review any scientific papers.
- In summary, the students are not only expected to learn the current state of the arts in software maintenance and evolution, but also be able to frame research problems, conduct experiments, write and critique papers.

Student Evaluation

Grading Scheme

Each of the parts of the course is important and counted towards the final grade. The general scheme is as follows.

- Class participation (10%)
  - Active participation in the discussions is encouraged
  - 3 hours/week over the term
- Paper presentations/discussions by you and a partner (15%)
  - Each student (with a partner) will present at least three papers over the term.
- A bi-weekly critique of one of the assigned papers (15%)
  - Each student (with a partner) will provide a critique (1-2 page) of at least five papers over the term.
- A project (+ a couple of small assignments) carried out with a one/two member group (60%)
  - Project proposal (5%)
  - Project proposal presentation (2%)
  - Project progress report (no more than five pages IEEE format, 15%)
  - Project progress presentation (3%)
  - Final project report (no more than 20 pages IEEE format, 30%)
  - Final project presentation (5%)

Information on literal descriptors for grading at the University of Saskatchewan can be found at:
http://students.usask.ca/current/academics/grades/grading-system.php

Please note: There are different literal descriptors for undergraduate and graduate students.
More information on the Academic Courses Policy on course delivery, examinations and assessment of student learning can be found at:

http://policies.usask.ca/policies/academic-affairs/academic-courses.php

The University of Saskatchewan Learning Charter is intended to define aspirations about the learning experience that the University aims to provide, and the roles to be played in realizing these aspirations by students, instructors and the institution. A copy of the Learning Charter can be found at: http://www.usask.ca/university_secretary/LearningCharter.pdf

University of Saskatchewan Grading System (for undergraduate courses)

Exceptional (90-100) A superior performance with consistent evidence of

- a comprehensive, incisive grasp of the subject matter;
- an ability to make insightful critical evaluation of the material given;
- an exceptional capacity for original, creative and/or logical thinking;
- an excellent ability to organize, to analyze, to synthesize, to integrate ideas, and to express thoughts fluently.

Excellent (80-90) An excellent performance with strong evidence of

- a comprehensive grasp of the subject matter;
- an ability to make sound critical evaluation of the material given;
- a very good capacity for original, creative and/or logical thinking;
- an excellent ability to organize, to analyze, to synthesize, to integrate ideas, and to express thoughts fluently.

Good (70-79) A good performance with evidence of

- a substantial knowledge of the subject matter;
- a good understanding of the relevant issues and a good familiarity with the relevant literature and techniques;
- some capacity for original, creative and/or logical thinking;
- a good ability to organize, to analyze and to examine the subject material in a critical and constructive manner.

Satisfactory (60-69) A generally satisfactory and intellectually adequate performance with evidence of

- an acceptable basic grasp of the subject material;
- a fair understanding of the relevant issues;
- a general familiarity with the relevant literature and techniques;
- an ability to develop solutions to moderately difficult problems related to the subject material;
- a moderate ability to examine the material in a critical and analytical manner.

Minimal Pass (50-59) A barely acceptable performance with evidence of

- a familiarity with the subject material;
- some evidence that analytical skills have been developed;
- some understanding of relevant issues;
- some familiarity with the relevant literature and techniques;
- attempts to solve moderately difficult problems related to the subject material and to examine the material in a critical and analytical manner which are only partially successful.

Failure <50 An unacceptable performance

Reading List (tentative): We will primarily focus on the following list of papers over the term as the basis. We then look for which others papers have cited these papers and then look for the latest papers on the topic areas for choosing the right one. In this way, we get to know which topics are more dominant and interesting and what papers we need to focus on. We also get to know the authors in the area and the venues they publish. Students are taught well of how to find the latest papers in a certain topic areas.

[1] Michael W. Godfrey and Daniel M. German. The Past, Present, and Future of Software Evolution. Invited paper in *Proc. of Frontiers of Software Maintenace*track at the *2008 IEEE Intl. Conf. on Software Maintenance* (ICSM-08*)*, October 2008, Beijing, China

[2] J. Maletic and H. Kgdi Expressiveness and Effectiveness of Program Comprehension: Thoughts on Future Research Directions. In *Proc. of the Frontiers of Software Maintenance track at ICSM'08*, pp. 31-37, 2008.

[3] Mik Kersten and Gail Murphy. Using Task Context to Improve Programmer Productivity. In *Proc. of 2006 Conference on Foundations of Software Engineering*(FSE-06), Nov 2006.

[4] Elizabeth Burd and Malcolm Munro. An Initial Approach Towards Measuring and Characterising Software Evolution. In *Proc. of 1999 Working Conf. on Reverse Engineering* (WCRE-99), Atlanta, 1999.

[5] R. Bednarik and M. Tukiainen. An Eye-Tracking Methodology for Characterizing Program Comprehension Processes. In *Proceedings of 2006 symposium on Eye tracking research & applications (ETRA)*, San Diego, California, pp. 125-132, 2006.

[6] S. Yusuf, H. Kagdi, and J.I. Maletic. Assessing the Comprehension of UML Diagrams via Eye Tracking. In *Proceedings of 15th IEEE International Conference on Program Comprehension (ICPC'07)*, Banff, Canada, June 26-29, pp. 113-122, 2007.

[7] C.K. Roy and J.R. Cordy. Scenario-based Comparison of Clone Detection Techniques. In *Proceedings of the 16th IEEE International Conference on Program Comprehension (ICPC’08)*, pp.153-162, IEEE Press, Amsterdam, The Netherlands, June 2008.

[8] Cory J. Kapser and Michael W. Godfrey. `Cloning Considered Harmful' Considered Harmful: Patterns of Cloning in Software. *Empirical Software Engineering*(Springer), vol. 13, no. 6, December 2008

[9] C.K. Roy and J.R. Cordy. Near-miss Function Clones in Open Source Software: An Empirical Study. *Journal of Software Maintenance and Evolution: Research and Practice*, 25 pp., 2009.

[10] Michael W. Godfrey and Lijie Zou. Using Origin Analysis to Detect Merging and Splitting of Source Code Entities. *IEEE Trans. on Software Engineering*, vol. 31, no. 2 (Feb 2005).

[11] Hamid Abdul Basit and Stan Jarzabek. Detecting Higher-level Similarity Patterns in Programs. In *Proc. of the 2005 Intl. Conference on Foundations of Software Engineering* (FSE-05), Lisbon, Portugal, September 2005

[12] B.S. Baker. Finding Clones with Dup: Analysis of an Experiment. *IEEE Transactions on Software Engineering*, Vol. 33(9): 608-621, 2007.

[13] Y. Higo and S. Kusumoto. Significant and Scalable Code Clone Detection with Program Dependency Graph. In *Proc. of the 16th Working Conference on Reverse Engineering*, 4 pp., 2009.

[14] D. Hou, F. Jacob, and P. Jablonski. Exploring the Design Space of Proactive Tool Support for Copy-and-Paste Programming. In Proc. *IBM Conference of the Centre for Advanced Studies on Collaborative Research*, 15 pp., 2009.

[15] E. Juergens, F. Deissenboeck, B. Hummel and S. Wagner. Do Code Clones Matter? In *Proc. of the 31st International Conference on Software Engineering*, pp. 485-495, 2009.

[16] I. Baxter and C. Pidgeon. Software Change Through Design Maintenance. In*Proceedings of International Conference on Software Maintenance (ICSM97)*, Bari,Italy, October 1-3 , 1997.

[17] F. P. Brooks. No Silver Bullet: Essence and Accidents of Software Engineering.*IEEE Computer*, vol. 9, no. 4, pp. 34-42, 1987.

[18] Ahmed E. Hassan . The Road Ahead for Mining Software Repositories. Invited paper in *Proc. of Frontiers of Software Maintenace* track at the *2008 IEEE Intl. Conf. on Software Maintenance* (ICSM-08), October 2008, Beijing, China.

[19] Thomas D. LaToza, Gina Venolia, Robert DeLine . Maintaining Mental Models: A Study of Developer Work Habits. In *Proc. of the 2006 Intl. Conf. on Software Engineering* (ICSE-06), Shanghai, China, May 2006.

[20] Peter C. Rigby and Ahmed E. Hassan. What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List.

In *Proc. of the Fourth International Workshop on Mining Software Repositories*(MSR-07), Minneapolis, May 2007.

[21] Martin Robillard and Barthelemy Dagenais. Retrieving Task-Related Clusters from Change History. In *Proc. of the 2008 Working Conference on Reverse Engineering*(WCRE-08), Antwerp, Belgium, October 2008

[22] Chris F. Kemerer and Sandra Slaughter. An empirical approach to studying software evolution. *IEEE Transactions on Software Engineering*, 25(4):493–509, 1999.

[23] Andreas Zeller. Yesterday, my program worked. today, it does not. why? In*ESEC/FSE-7: Proceedings of the 7th European Software Engineering Conference held jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering,* pp. 253–267, London, UK, 1999. Springer-Verlag.

[24] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. CCFinder: Amultilinguistic token-based code clone detection system for large scale source code.*IEEE Transactions on Software Engineering*, 28(7):654–670, 2002.

[25] Antoniol, G., Canfora, G., Casazza, G., and De Lucia, A. Maintaining Traceability Links During Object-Oriented Software Evolution. *Software - Practice and Experience,* vol. 31, no. 4, April, pp. 331-355, 2001.

[26] Alessandro Orso, Nanjuan Shi, and Mary Jean Harrold. Scaling regression testing to large software systems. In *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth International Symposium on Foundations of Software Engineering*, pages 241–251, New York, NY, USA, 2004. ACM

[27] Xiaoxia Ren, Fenil Shah, Frank Tip, Barbara G. Ryder, and Ophelia Chesley.Chianti: a tool for change impact analysis of java programs. In *OOPSLA '04: Proceedings of the 19th annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 432–448, New York, NY, USA, 2004. ACM.

[28] Danny Dig and Ralph Johnson. Automated detection of refactorings in evolving components. In *ECOOP '06: Proceedings of European Conference on Object-Oriented Programming*, pages 404–428. Springer, 2006.

[29] Martin P. Robillard and Gail C. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 406–416, Washington, DC, USA, 2003. IEEE Computer Society.

[30] Chikofsky, E. J. and Cross, J. H. Reverse Engineering and Design Recovery: A

Taxonomy. *IEEE Software*, vol. 7, no. 1, January, pp. 13-17, 1990.

[31] Eisenbarth, T., Koschke, R., and Simon, D. Locating Features in Source Code. *IEEE Transactions on Software Engineering*, vol. 29, no. 3, March, pp. 210 – 224, 2003.

[32] Michele Lanza and St´ephane Ducasse. Polymetric views-a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering,*29(9):782–795, 2003.

[33] Briand, L. C., Daly, J. W., and Wüst, J. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. In *Proceedings of 4th International Software Metrics Symposium (METRICS'97)*, Albuquerque, NM, November 5-7, pp. 43-53, 1997.

[34] Thomas Reps, Thomas Ball, Manuvir Das, and James Larus. The use of program profiling for software maintenance with applications to the year 2000 problem. In*ESEC '97/FSE-5: Proceedings of the 6th European Conference held jointly with the 5th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 432–449, New York, NY, USA, 1997. Springer-Verlag New York, Inc.

[35] Miryung Kim, Vibha Sazawal, David Notkin, and Gail Murphy. An empirical study of code clone genealogies. In *ESEC/FSE-13: Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 187–196, New York, NY, USA, 2005. ACM.

[36] Wuu Yang. Identifying syntactic differences between two programs. *Software – Practice & Experience*, 21(7):739–755, 1991.

[37] Miryung Kim and David Notkin. Discovering and Representing Systematic Code Changes. In *Proceedings of the International Conference on Software Engineering,*2009.

[38] Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, and Andreas Zeller.Mining version histories to guide software changes. *IEEE Transactions on Software Engineering*, 31(6):429–445, 2005

[39] P. Tonella. Concept Analysis for Module Restructuring. *IEEE Transactions on Software Engineering*, vol. 27, no. 4, July, pp. 351-363, 2001.

[40] John Anvik, Lyndon Hiew and Gail C. Murphy. Who Should Fix This Bug? In*Proc. of the 2006 Intl. Conference on Software Engineering* (ICSE-06), Shanghai, May 2006.

[41] Wilde, N., Buckellew, M., Page, H., Rajlich, V., and Pounds, L. A comparison of methods for locating features in legacy software. *The Journal of Systems and Software*, vol. 65, pp. 105-114, 2003.

[42] Antoniol, G., Fiutem, R., and Cristoforetti, L. Using Metrics to Identify Design Patterns in Object-Oriented Software. In *Proceedings of 5th IEEE International Symposium on Software Metrics (METRICS'98)*, Bethesda, MD, November 20-21, pp. 23 – 34, 1998.

[43] Mary Shaw. What Makes Good Research in Software Engineering? Invited presentation given at *ETAPS 2002*, Grenoble, France.

[44] Hongyu Zhang. Exploring Regularity in Source Code: Software Science and Zipf'sLaw. In *Proc. of the 2008 Working Conference on Reverse Engineering* (WCRE-08), Antwerp, Belgium, October 2008

[45] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM,* 15(12):1053–1058, 1972.

[46] Lewis, G. A., and Smith, D. B. Service-Oriented Architecture and its Implications for Software Maintenance and Evolution. In *Proc. of the Frontiers of Software Maintenance track at ICSM'08*, pp. 1-10, 2008.

[47] J.R. Cordy. The TXL Source Transformation Language. *Science of Computer Programming* 61,3 (August 2006), pp. 190-210.

[48] Daniel M. German, Ahmed E. Hassan and Gregorio Robles. Change impact graphs: Determining the impact of prior code changes. *Information and Software Technology.* In Press, May, 2009.

[49] Omar Alam, Bram Adams and Ahmed E. Hassan. A Study of the Time Dependence of Code Changes. *In Proceedings of the 25th IEEE Working Conference on Reverse Engineering (WCRE)*, Lille, France, October 15-18, 2009.

[50] Nicolas Bettenburg, Weyi Shang, Walid Ibrahim, Bram Adams and Ying Zou and Ahmed E. Hassan. An Empirical Study on Inconsistent Changes to Code Clones at Release Level. In *Proceedings of the 25th IEEE Working Conference on Reverse Engineering (WCRE)*, Lille, France, October 15-18, 2009.

[51] Adam Kieżun, Philip J. Guo, Karthick Jayaraman, and Michael D. Ernst. Automatic creation of SQL injection and cross-site scripting attacks. In *ICSE'09, Proceedings of the 30th International Conference on Software Engineering*, (Vancouver, BC, Canada), May 20-22, 2009.

[52] Surafel Lemma Abebe, Sonia Haiduc, Andrian Marcus, Paolo Tonella,

GiulianoAntoniol. Analyzing the Evolution of the Source Code Vocabulary. In *Proc. ofCSMR 2009*, pp. 189-198, 2009.

[53] Stefan Gueorguiev, Mark Harman, Giuliano Antoniol. Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In *GECCO 2009*, pp. 1673-1680.

[54] Bénédicte Kenmei, Giuliano Antoniol, Massimiliano Di Penta: Trend Analysis and Issue Prediction in Large-Scale Open Source Systems. In *CSMR 2008*, pp. 73-82.

[55] Marc Eaddy, Alfred V. Aho, Giuliano Antoniol, Yann-Gaeal Gueheneuc: CERBERUS: Tracing Requirements to Source Code Using Information Retrieval, Dynamic Analysis, and Program Analysis. In *ICPC 2008*, pp. 53-62.

[56] Chiara Di Francescomarino, Alessandro Marchetto, Paolo Tonella: Reverse Engineering of Business Processes exposed as Web Applications. In *CSMR 2009*,pp. 139-148.

[57] Shin Yoo, Mark Harman, Paolo Tonella, Angelo Susi: Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge. In *ISSTA 2009*, pp. 201-212.

[58] Gerardo Canfora, Luigi Cerulo, Massimiliano Di Penta: Ldiff: An enhanced linedifferencing tool. In *ICSE 2009*, pp. 595-598.

Incomplete Course Work and Final Grades

"*When a student has not completed the required course work, which includes any assignment or examination including the final examination, by the time of submission of the final grades, they may be granted an extension to permit completion of an assignment, or granted a deferred examination in the case of absence from a final examination.*

Extensions past the final examination date for the completion of assignments must be approved by the Department Head, or Dean in non-departmentalized Colleges, and may exceed thirty days only in unusual circumstances. The student must apply to the instructor for such an extension and furnish satisfactory reasons for the deficiency. Deferred final examinations are granted as per College policy.

In the interim, the instructor will submit a computed percentile grade for the class which factors in the incomplete coursework as a zero, along with a grade comment of INF (Incomplete Failure) if a failing grade.

In the case where the student has a passing percentile grade but the instructor has indicated in the course outline that failure to complete the required coursework will result in failure in the course, a final grade of 49% will be submitted along with a grade comment of INF (Incomplete Failure).

If an extension is granted and the required assignment is submitted within the allotted time, or if a deferred examination is granted and written in the case of absence from the final examination, the instructor will submit a revised assigned final percentage grade. The grade change will replace the previous grade and any grade comment of INF (Incomplete Failure) will be removed.

A student can pass a course on the basis of work completed in the course provided that any incomplete course work has not been deemed mandatory by the instructor in the course outline and/or by College regulations for achieving a passing grade." (http://policies.usask.ca/policies/academic-affairs/academic-courses.php)

For policies governing examinations and grading, students are referred to the Assessment of Students section of the University policy "Academic courses: class delivery, examinations, and assessment of student learning" (http://policies.usask.ca/policies/academic-affairs/academic-courses.php)

Academic Honesty

The University of Saskatchewan is committed to the highest standards of academic integrity and honesty. Students are expected to be familiar with these standards regarding academic honesty and to uphold the policies of the University in this respect. Students are particularly urged to familiarize themselves with the provisions of the Student Conduct & Appeals section of the University Secretary Website and avoid any behavior that could potentially result in suspicions of cheating, plagiarism, misrepresentation of facts and/or participation in an offence. Academic dishonesty is a serious offence and can result in suspension or expulsion from the University.

All students should read and be familiar with the Regulations on Academic Student Misconduct (http://www.usask.ca/secretariat/student-conduct-appeals/StudentAcademicMisconduct.pdf) as well as the Standard of Student Conduct in Non-Academic Matters and Procedures for Resolution of Complaints and Appeals (http://www.usask.ca/secretariat/student-conduct-appeals/StudentNon-AcademicMisconduct.pdf) Academic honesty is also defined and described in the Department of Computer Science Statement on Academic Honesty (http://www.cs.usask.ca/undergrad/honesty.php).

For more information on what academic integrity means for students see the Student Conduct & Appeals section of the University Secretary Website at: http://www.usask.ca/secretariat/student-conduct-appeals/forms/IntegrityDefined.pdf

Examinations with Disability Services for Students (DSS)

Students who have disabilities (learning, medical, physical, or mental health) are strongly encouraged to register with Disability Services for Students (DSS) if they have not already done so. Students who suspect they may have disabilities should contact DSS for advice and referrals. In order to access DSS programs and supports, students must follow DSS policy and procedures. For more information, check http://students.usask.ca/health/centres/disability-services-for-students.php, or contact DSS at 966-7273 or dss@usask.ca.

Students registered with DSS may request alternative arrangements for mid-term and final examinations. Students must arrange such accommodations through DSS by the stated deadlines. Instructors shall provide the examinations for students who are being accommodated by the deadlines established by DSS.