DAT265/DIT598 Software Evolution Project - Course Literature

Before the course starts

- Wolfe, J., & Powell, E. (2014, October). Strategies for dealing with slacker and underperforming teammates in class projects. In *Professional Communication Conference (IPCC), 2014 IEEE International* (pp. 1-8). IEEE.

Introduction to Software Evolution

- ADM Task Force. "Architecture-driven modernization scenarios." Object Management Group (*OMG), USA* (2006), http://adm.omg.org/ *
- ISO/IEC 9126-1 Software Engineering Product Quality Part 1: Quality Model (25 pages). *
- Android development basics: https://developer.android.com/training/index.html ("Build your first app", "Building a dynamic UI with Fragments"), https://developer.android.com/training/building-graphics.html
- Further reading:
    - Lehman, Meir M., et al. "Metrics and laws of software evolution-the nineties view." *Software Metrics Symposium.* IEEE, 1997. *
    - Mens, Tom. "Introduction and roadmap: History and challenges of software evolution." *Software evolution*. Springer Berlin Heidelberg, 2008. 1-11 *
    - Bennett, Keith H., and Václav T. Rajlich. "Software maintenance and evolution: a roadmap." *Proceedings of the Conference on the Future of Software Engineering*. ACM, 2000. *

Software Comprehension

- Cornelissen, Bas, et al. "A systematic survey of program comprehension through dynamic analysis." *IEEE Transactions on Software Engineering* 35.5 (2009): 684-702. *
- Roehm, Tobias, et al. "How do professional developers comprehend software?." *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012. *
- Further reading:
    - Siegmund, Janet, and Jana Schumann. "Confounding parameters on program comprehension: a literature survey." Empirical Software Engineering 20.4 (2015): 1159-1192. *
    - Siegmund, Janet. "Program Comprehension: Past, Present, and Future." Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on. Vol. 5. IEEE, 2016. *
    - Jbara, Ahmad, and Dror G. Feitelson. "How programmers read regular code: a controlled experiment using eye tracking." Empirical Software Engineering 22.3 (2017): 1440-1477. *
    - Chikofsky, Elliot J., and James H. Cross. "Reverse engineering and design recovery: A taxonomy." *IEEE software* 7.1 (1990): 13-17. *

Refactoring

- Fowler, M., Catalog of Refactoring, http://refactoring.com/catalog/
- Source Making, Code Smells, https://sourcemaking.com/refactoring/smells/divergent-change
- Refactoring in Eclipse: http://www.ibm.com/developerworks/library/os-ecref/ http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Freference%2Fref-menu-refactor.htm

* Literature available on the internal course page (ping pong)

- Negara, S., Chen, N., Vakilian, M., Johnson, R. E., & Dig, D. (2013, July). A comparative study of manual and automated refactorings. In *European Conference on Object-Oriented Programming* (pp. 552-576). Springer Berlin Heidelberg. *
- Further Reading
    - Silva, Danilo, Nikolaos Tsantalis, and Marco Tulio Valente. "Why we refactor? Confessions of github contributors." Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016. *
    - Murphy-Hill, Emerson, Chris Parnin, and Andrew P. Black. "How we refactor, and how we know it." IEEE Transactions on Software Engineering 38.1 (2012): 5-18. *
    - Kim, Miryung, Thomas Zimmermann, and Nachiappan Nagappan. "A field study of refactoring challenges and benefits." Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012. *
    - Fokaefs, Marios, Nikolaos Tsantalis, and Alexander Chatzigeorgiou. "JDeodorant: Identification and Removal of Feature Envy Bad Smells." *ICSM*. 2007. * https://marketplace.eclipse.org/content/jdeodorant https://users.encs.concordia.ca/~nikolaos/jdeodorant/index.php?option=com_content&view=article&id=45

Clone Detection & Removal

- Rattan, Dhavleesh, Rajesh Bhatia, and Maninder Singh. "Software clone detection: A systematic review." *Information and Software Technology* 55.7 (2013): 1165-1199. *
- Saha, Ripon K., et al. "Evaluating code clone genealogies at release level: An empirical study." *Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on*. IEEE, 2010. *
- Further reading:
    - Georges Golomingi Koni-N'sapu. *A scenario based approach for refactoring duplicated code in object oriented systems*. Diploma Thesis, University of Bern, June 2001. *
    - Roy, Chanchal K., James R. Cordy, and Rainer Koschke. "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach." Science of computer programming 74.7 (2009): 470-495. *
    - Su, Fang-Hsiang, et al. "Identifying functionally similar code in complex codebases." Program Comprehension (ICPC), 2016 IEEE 24th International Conference on. IEEE, 2016. *
    - van Tonder, Rijnard, and Claire Le Goues. "Defending against the attack of the micro-clones." Program Comprehension (ICPC), 2016 IEEE 24th International Conference on. IEEE, 2016. *

Continuous Integration

- Debbiche, Adam, Mikael Dienér, and Richard Berntsson Svensson. "Challenges When Adopting Continuous Integration: A Case Study." *Product-Focused Software Process Improvement*. Springer International Publishing, 2014. 17-32. *
- Nilsson, Agneta, Jan Bosch, and Christian Berger. "Visualizing testing activities to support continuous integration: A multiple case study." *Agile Processes in Software Engineering and Extreme Programming*. Springer International Publishing, 2014. 171-186 *

Other literature depends on the specific topic selected by the student; suggestions will be collected on the course homepage.

* Literature available on the internal course page (ping pong)