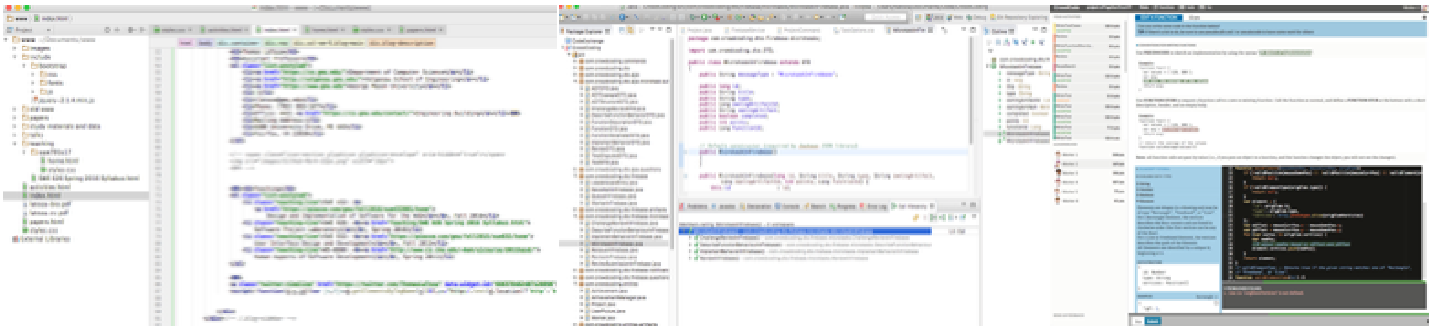
SWE 795: Software Engineering Environments, Fall 2019

Monday 4:30 - 7:10, Enterprise Hall 173



Instructor: Prof. Thomas LaToza (http://cs.gmu.edu/~tlatoza) tlatoza@gmu.edu (https://mail.google.com/mail/?view=cm&fs=1&tf=1&to=tlatoza@gmu.edu)
Twitter: @ThomasLaToza (https://twitter.com/thomaslatoza)
Office: 4431 Engineering Building; (703) 993-1677
Office Hours: Anytime electronically, Wed 3:00 - 4:30, or by appointment

Objective and Learning Outcomes

This course will explore the nature of software engineering environments, investigating programming tools and development environments intended to support programming activities. The focus will primarily be on individual software development, such as what is known about tools for debugging, reuse, program synthesis, refactoring, and managing crosscutting concerns. The course will consider questions such as What makes debugging hard?, What causes defects?, How do developers check if code works?, and explore how tools have been designed to better support these activities. The course will survey the science of studying programming activities and the design of tools designed to help developers work with code more easily and successfully. This will include what is sometimes called "Empirical Studies of Programmers" and the "Psychology of Programming".

At the end of this course, you should be able to:

(1) Conduct a small study of software development practice
(2) Design a development environment feature to address a challenge software developers face
(3) Implement a prototype of a development environment feature within a modern development environment

Readings

We will have weekly readings from relevant research papers. All readings will be posted in the schedule below. Each week, every student will be responsible for reading all 3 papers and writing a brief answer to one of several prompts on Piazza before class.

Additionally, each student will be responsible for being the designated Discussant for one paper once every approximately three weeks. For each paper, the discussant will be responsible for giving a short 6 min presentation briefly summarizing the paper and facilitating 9 mins of discussion about the paper with the class.

Project

The homework in this course will be in the form of a project. All project work will occur in two person groups. Rather than creating a written report, each HW assignment, after HW0, will take the form of an in-class presentation, where all groups members will give a 10-min presentation on their work.

HW0: Project Proposal (50 points)

The project proposal should describe a specific aspect of software development that your project will focus on. The project proposal should clearly identify a specific challenge software developers experience in programming work, including a scenario describing a situation a developer might face. The project proposal should also include (1) a brief description of the type of study you will perform to understand this challenge better and (2) an initial idea of how a tool might address this challenge.

HW1: Review of Literature (100 points)

In this assignment, you will read several papers related to your proposed project idea. You will summarize each of these papers, describing the similarities and differences with the approach you are envisioning. Based on what you learn from these readings as well the feedback you received on HW0 from the instructor, you will prepare a revised plan for your project.

HW2: Study of Current Practice (100 points)

The study of current practice will be a small study examining a specific challenge software developers face in their programming work. Several types of study are possible. You might choose to examine posts on StackOverflow or another online repository. You might choose to perform a simple think aloud usability study and observe 2 or 3 participants perform a programming task. Or you might choose to survey professional software developers about their activities. In any case, the outcome of the study should be a better understanding of a challenge that software developers face in their programming work.

HW3: Tool Sketch (100 points)

Based on the challenge identified in HW1, you should create a sketch of a potential solution. Your sketch should not contain any implementation of your tool. Instead, the sketch should provide a storyboard, depicting a series of screenshots describing the behavior of your tool on two or more examples. Additionally, a description and high-level overview of the tool's design and implementation should be included.

HW4: Tool Prototype (250 points)

Based on your sketch, you will implement a small prototype of your tool, extending an existing development environment such as Eclipse, Visual Studio, WebStorm, or seeCode.run to implement your idea.

Tentative Schedule

1. Course Overview and Conducting Studies (lecture1-overview.pdf) (8/26)

Assigned readings: none

(9/2) - NO CLASS - LABOR DAY

HWs: HW0 due on Sept 3

2. Design Process (Lecture%202%20-%20Design%20Process.pdf) (9/9)

Assigned readings:

- B. A. Myers, A. J. Ko, T. D. LaToza and Y. Yoon, "Programmers Are Users Too: Human-Centered Methods for Improving Programming Tools (http://ieeexplore.ieee.org/document/7503516/)," in Computer, vol. 49, no. 7, pp. 44-52, July 2016. Summary by Thomas LaToza (Programmers%20are%20Users%20Too.pdf)
- Ko, A. J., LaToza, T.D., and Burnett, M. M. (2013). A practical guide to controlled experiments of software engineering tools with human participants. (http://dx.doi.org/10.1007/s10664-013-9279-3) Empirical Software Engineering (ESE), Sept. 2013, 1-32.
- Andrew J. Ko and Brad A. Myers. 2009. Finding causes of program output with the Java Whyline (https://doi.org/10.1145/1518701.1518942). Conference on Human Factors in Computing Systems (CHI), 1569-1578.   Summary by Thomas LaToza (Finding%20Causes%20of%20Program%20Output%20with%20the%20Java%20WhyLine.pdf)

3. Problem Solving (9/16)

Assigned readings:

- Thomas D. LaToza, David Garlan, James D. Herbsleb, and Brad A. Myers. 2007. Program comprehension as fact finding (https://doi.org/10.1145/1287624.1287675). European software engineering conference and the Symposium on the Foundations of Software Engineering, 361-370.
- Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2008. Asking and Answering Questions during a Programming Change Task (https://doi.org/10.1109/TSE.2008.26). IEEE Trans. Softw. Eng. 34, 4 (July 2008), 434-451.

4. Programming as Communication (9/23)

HWs: HW1 due

5. Debugging (9/30)

Assigned readings:

- Diomidis Spinellis. 2018. Modern debugging: the art of finding a needle in a haystack (https://doi.org/10.1145/3186278). Commun. ACM 61, 11, (October 2018), 124-134.
- James A. Jones, Mary Jean Harrold, and John Stasko. 2002. Visualization of test information to assist fault localization (DOI=http://dx.doi.org/10.1145/581339.581397). International Conference on Software Engineering (ICSE), 467-477.
- Marcel Böhme, Ezekiel O. Soremekun, Sudipta Chattopadhyay, Emamurho Ugherughe, and Andreas Zeller. 2017. Where is the bug and how is it fixed? an experiment with practitioners (https://doi.org/10.1145/3106237.3106255). Foundations of Software Engineering (ESEC/FSE), 117-128.

6. Crosscutting Concerns (10/7)

HWs: HW2 due

(10/14) - MONDAY CLASSES MEET ON TUESDAY

7. Navigating Code (10/15) (ONLINE LECTURE - NO CLASS MEETING)

Assigned readings: None

8. Software Visualization (10/21)

HWs: HW3 due

9. Editing Code (10/28)

Assigned readings:

- Young Seok Yoon and Brad A. Myers. 2015. Supporting selective undo in a code editor (https://dl.acm.org/citation.cfm?id=2818784). International Conference on Software Engineering (ICSE), 223-233.
- Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. 2012. How We Refactor, and How We Know It (http://dx.doi.org/10.1109/TSE.2011.41). IEEE Trans. Softw. Eng. 38, 1 (January 2012), 5-18.
- Brian Hempel, Justin Lubin, Ravi Chugh. 2019. Sketch-n-Sketch: Output-Directed Programming for SVG (https://arxiv.org/pdf/1907.10699.pdf). Symposium on User Interface Software and Technology (UIST).

10. Preventing Defects (11/4)

Assigned readings:

- Andrew J. Ko and Brad A. Myers. 2005. A framework and methodology for studying the causes of software errors in programming systems (https://faculty.washington.edu/ajko/papers/Ko2004SoftwareErrorsFramework.pdf). J. Vis. Lang. Comput. 16, 1-2 (February 2005), 41-84.
- Caitlin Sadowski, Edward Aftandilian, Alex Eagle, Liam Miller-Cushon, and Ciera Jaspan. 2018. Lessons from building static analysis tools at Google (https://doi.org/10.1145/3188720). Commun. ACM 61, 4 (March 2018), 58-66.
- Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W. O'Hearn. 2019. Scaling static analyses at Facebook (https://doi.org/10.1145/3338112). Commun. ACM 62, 8 (July 2019), 62-70.

11. Code Reuse (11/11)

Assigned readings:

- Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code (https://doi.org/10.1145/1518701.1518944). Conference on Human Factors in Computing Systems (CHI), 1589-1598.
- Stephen Oney and Joel Brandt. 2012. Codelets: linking interactive documentation and example code in the editor (https://doi.org/10.1145/2207676.2208664). Conference on Human Factors in Computing Systems (CHI), 2697-2706.
- Xin Rong, Shiyan Yan, Stephen Oney, Mira Dontcheva, and Eytan Adar. 2016. CodeMend: Assisting Interactive Programming with Bimodal Embedding (https://doi.org/10.1145/2984511.2984544). Symposium on User Interface Software and Technology (UIST), 247-258.

12. Crowdsourcing (11/18)

Assigned readings:

- Klaas-Jan Stol and Brian Fitzgerald. 2014. Two's company, three's a crowd: a case study of crowdsourcing software development (https://doi.org/10.1145/2568225.2568249). International Conference on Software Engineering (ICSE), 187-198.
- Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. 2017. Codeon: On-Demand Software Development Assistance (https://doi.org/10.1145/3025453.3025972). Conference on Human Factors in Computing Systems (CHI), 6220-6231.
- Thomas D. LaToza, Arturo Di Lecce, Fabio Ricci, W. Ben Towne, and André van der Hoek. 2019. Microtask programming (https://doi.org/10.1109/TSE.2018.2823327). Transactions on Software Engineering.

13. Program Synthesis (11/25)

Assigned readings:

- Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. 2019. Automated Program Repair (http://www.cs.cmu.edu/~clegoues/docs/legoues-cacm2019.pdf). Communications of the ACM (CACM).

- A. Marginean, J. Bader, S. Chandra, M. Harman, Y. Jia, K. Mao, A. Mols, and A. Scott. 2019. SapFix: automated end-to-end repair at scale (https://doi.org/10.1109/ICSE-SEIP.2019.00039). International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 269-278.
- Joel Galenson, Philip Reames, Rastislav Bodik, Björn Hartmann, and Koushik Sen. 2014. CodeHint: dynamic and interactive synthesis of code snippets (https://doi.org/10.1145/2568225.2568250). International Conference on Software Engineering, 653-663.

14. Learning Programming (12/2)

HWs: HW4 due

Resources

This course will use Piazza for posting the schedule and all assignments and announcements. Additionally, we will use Piazza for a discussion board. Grades will be available through Blackboard.

Makeups

As much of the course work will consist of in-class presentations, it will not be possible to submit HW assignments late. HW assignments submitted late will receive a zero. If you will be unable to attend class on the date of a HW assignment, please contact the instructor about this as early as possible.

Grading

Paper responses: 20%
Paper discussant: 20%
Project: 60%

Honor Code

GMU is an Honor Code university; please see the Office for Academic Integrity for a full description of the code and the honor committee process, and the Computer Science Department's Honor Code Policies regarding programming assignments. The principle of academic integrity is taken very seriously and violations are treated gravely. What does academic integrity mean in this course? Essentially this: when you are responsible for a task, you will perform that task. When you rely on someone else's work in an aspect of the performance of that task, you will give full credit in the proper, accepted form. Another aspect of academic integrity is the free play of ideas. Vigorous discussion and debate are encouraged in this course, with the firm expectation that all aspects of the class will be conducted with civility and respect for differing ideas, perspectives, and traditions. When in doubt (of any kind) please ask for guidance and clarification.

Accommodations for Disabilities

If you have a documented learning disability or other condition that may affect academic performance you should: 1) make sure this documentation is on file with Office for Disability Services (SUB I, Rm. 4205; 993-2474; http://ods.gmu.edu (http://ods.gmu.edu)) to determine the accommodations you need; and 2) talk with me to discuss your accommodation needs.

Privacy

Students must use their MasonLIVE email account to receive important University information, including messages related to this class. See http://masonlive.gmu.edu (http://masonlive.gmu.edu) for more information.

Other Useful Campus Resources

Writing Center: A114 Robinson Hall; (703) 993-1200; http://writingcenter.gmu.edu (http://writingcenter.gmu.edu)
University Libraries: Ask a Librarian
Counseling and Psychological Services (CAPS): (703) 993-2380; http://caps.gmu.edu/ (http://caps.gmu.edu/)
University Policies: The University Catalog, is the central resource for university policies affecting student, faculty, and staff conduct in university academic affairs. Other policies are available at http://universitypolicy.gmu.edu/. All members of the university community are responsible for knowing and following established policies.
GMU Academic Calendar