SWE 795: Software Engineering Environments, Spring 2017

Tuesdays 4:30 - 7:10, 2026 Art and Design Building

Syllabus and Schedule

Instructor: Prof. Thomas LaToza

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

Required Textbooks

None

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
(4) Evaluate a development environment feature using a small think-aloud usability study

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 two weeks, serving as discussant for a total of 6 papers over the semester. For each paper, the discussant will be responsible for giving a short 5 min presentation briefly summarizing the paper and facilitating 10 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 be 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: 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.

HW2: 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.

HW3: 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.

HW4: Tool Evaluation (100 points)

To understand the effect of your tool on addressing a challenge programmers face, you will conduct a small study evaluating your tool with a few participants.

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

Tentative Schedule

1. Course Overview and Conducting Studies (1/24)

Assigned readings: none

2. Analyzing Data (1/31)

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.
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 '09), 1569-1578.

HWs: HW0 due

3. Information Needs (2/7)

Assigned readings:
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.
J. Lawrance, C. Bogart, M. Burnett, R. Bellamy, K. Rector and S. D. Fleming, "How Programmers Debug, Revisited: An Information Foraging Theory Perspective (https://doi.org/10.1109/TSE.2010.111)," in IEEE Transactions on Software Engineering, vol. 39, no. 2, pp. 197-215, Feb. 2013.
Thomas D. LaToza and Brad A. Myers. 2010. Developers ask reachability questions (https://doi.org/10.1145/1806799.1806829). International Conference on Software Engineering, 185-194.

HWs: none

4. Debugging (2/14)

Assigned readings:
Mark Weiser. 1981. Program slicing (http://dl.acm.org/citation.cfm?id=802557). International conference on Software engineering, 439-449.
Piramanayagam Arumuga Nainar and Ben Liblit. 2010. Adaptive bug isolation (https://doi.org/10.1145/1806799.1806839). International Conference on Software Engineering, 255-264.
T. D. LaToza and B. A. Myers, "Visualizing call graphs (https://doi.org/10.1109/VLHCC.2011.6070388)," 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Pittsburgh, PA, 2011, pp. 117-124.

HWs: HW1 due

5. Crosscutting Concerns (2/21)

Assigned readings:
D. L. Parnas. 1972. On the criteria to be used in decomposing systems into modules (https://doi.org/10.1145/361598.361623). Commun. ACM 15, 12 (December 1972), 1053-1058.
Peri Tarr, Harold Ossher, William Harrison, and Stanley M. Sutton, Jr. 1999. N degrees of separation: multi-dimensional separation of concerns (https://doi.org/10.1145/302405.302457). International conference on Software, 107-119.
Gail C. Murphy, Mik Kersten, Martin P. Robillard, and Davor Čubranić. 2005. The emergent structure of development tasks (https://doi.org/10.1007/11531142_2). European conference on Object-Oriented Programming, 33-48.

HWs: none

6. Mental Models (2/28)

Assigned readings:
Rist, R. S. (1989), Schema Creation in Programming (http://csjarchive.cogsci.rpi.edu/1989v13/i03/p0389p0414/MAIN.PDF). Cognitive Science, 13: 389–414.
A. Von Mayrhauser and A. M. Vans, "Program comprehension during software maintenance and evolution (https://doi.org/10.1109/2.402076)," in Computer, vol. 28, no. 8, pp. 44-55, Aug 1995.
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.

HWs: HW2 due

7. Software Visualization (3/7)

Assigned readings:
Ron Baecker, Chris DiGiano, and Aaron Marcus. 1997. Software visualization for debugging (https://doi.org/10.1145/248448.248458). Commun. ACM 40, 4 (April 1997), 44-54.
Robert DeLine, Gina Venolia, and Kael Rowan. 2010. Software development with code maps (https://doi.org/10.1145/1787234.1787250). Commun. ACM 53, 8 (August 2010), 48-54.
Pei-Yu (Peggy) Chi, Yang Li, and Björn Hartmann. 2016. Enhancing Cross-Device Interaction Scripting with Interactive Illustrations (https://doi.org/10.1145/2858036.2858382). Conference on Human Factors in Computing Systems, 5482-5493.

HWs: none

(3/14) – NO CLASS – Spring Break

8. Editing Code (3/21)

Assigned readings:
Don Roberts, John Brant, and Ralph Johnson. A refactoring tool for Smalltalk (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.9246&rep=rep1&type=pdf). Theory and Practice of Object Systems, 3(4):253–263, 1997.
Michael Toomim, Andrew Begel, and Susan L. Graham. 2004. Managing Duplicated Code with Linked Editing (https://doi.org/10.1109/VLHCC.2004.35). Symposium on Visual Languages - Human Centric Computing, 173-180.
Andrew J. Ko and Brad A. Myers. 2005. Citrus: a language and toolkit for simplifying the creation of structured editors for code and data (https://doi.org/10.1145/1095034.1095037). Symposium on User interface software and technology, 3-12.
HWs: none

9. Preventing Defects (3/28)

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.
J. R. Larus et al., " Righting software (https://www.microsoft.com/en-us/research/wp-content/uploads/2004/05/ieee_sw_righting_software.pdf)," in IEEE Software, vol. 21, no. 3, pp. 92-100, May-June 2004.
Al Bessey, Ken Block, Ben Chelf, Andy Chou, Bryan Fulton, Seth Hallem, Charles Henri-Gros, Asya Kamsky, Scott McPeak, and Dawson Engler. 2010. A few billion lines of code later: using static analysis to find bugs in the real world (https://doi.org/10.1145/1646353.1646374). Commun. ACM 53, 2 (February 2010), 66-75.

HWs: none

10. Reuse (4/4)

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, 1589-1598.
L. Martie, T. D. LaToza, and A. van der Hoek, " CodeExchange: Supporting Reformulation of Internet-Scale Code Queries in Context (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7371992)", International Conference on Automated Software Engineering (ASE), 2015, pages 24-35.
Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: design mining the web (https://doi.org/10.1145/2470654.2466420). Conference on Human Factors in Computing Systems, 3083-3092.

HWs: none

11. Program Synthesis (4/11)

Assigned readings:
Jiun-Hung Chen and Daniel S. Weld. 2008. Recovering from errors during programming by demonstration (https://doi.org/10.1145/1378773.1378794). International conference on Intelligent user interfaces (IUI), 159-168.
Yalin Ke, Kathryn T. Stolee, Claire Le Goues, and Yuriy Brun. Repairing Programs with Semantic Code Search (http://www.cs.cmu.edu/~clegoues/docs/searchRepair-ase15.pdf). In Automated Software Engineering (ASE), 2015, pp. 532-543.
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.

HWs: none

12. Visual Programming Languages (4/18)

Assigned readings:
T. R. G. Green, M. Petre. When Visual Programs are Harder to Read than Textual Programs (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.57.1633&rep=rep1&type=pdf). In Human-Computer Interaction: Tasks and Organisation, Proceedings ECCE-6 (6th European Conference Cognitive Ergonomics) (1992).
Avraham Leff and James T. Rayfield. 2007. Webrb: evaluating a visual domain-specific language for building relational web-applications (https://doi.org/10.1145/1297105.1297048). Conference on Object-oriented programming systems and applications, 281-300.
Jonathan Edwards. 2007. No ifs, ands, or buts: uncovering the simplicity of conditionals (http://www.subtext-lang.org/OOPSLA07.pdf). Conference on

Object-oriented programming systems and applications, 639-658.

HWs: HW3 due

13. Crowdsourcing (4/25)

Assigned readings:
Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. 2010. What would other programmers do: suggesting solutions to error messages (https://doi.org/10.1145/1753326.1753478). Conference on Human Factors in Computing Systems, 1019-1028.
Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and André van der Hoek. 2014. Microtask programming: building software with a crowd (https://doi.org/10.1145/2642918.2647349). Symposium on User interface software and technology, 43-54.
Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 2015. Apparition: Crowdsourced User Interfaces that Come to Life as You Sketch Them (https://doi.org/10.1145/2702123.2702565). Conference on Human Factors in Computing Systems, 1925-1934.

HWs: none

14. Learning Programming (5/2)

Assigned readings:
M. J. Lee et al., "Principles of a debugging-first puzzle game for computing education (https://doi.org/10.1109/VLHCC.2014.6883023)," Symposium on Visual Languages and Human-Centric Computing, 2014, pp. 57-64.
M. Gordon and P. J. Guo, "Codepourri: Creating visual coding tutorials using a volunteer crowd of learners (https://doi.org/10.1109/VLHCC.2015.7357193)," Symposium on Visual Languages and Human-Centric Computing, 2015, pp. 13-21.
Jill Cao, Scott Fleming, Margaret Burnett, and Christopher Scaffidi. Idea Garden: Situated Support for Problem Solving by End-User Programmers (https://doi.org/10.1093/iwc/iwu022). Interacting with Computers 27(6), 640-660, November 2015.

HWs: HW4 due