CS 563: Software Maintenance And Evolution

Mining Software Repositories

Oregon State University, Spring 2024

Today’s Plan

- Learn about mining software repositories
- Finish in-class exercise#2

Mining Software Repositories

1. What is a software repository?
2. What is mining software repositories?
3. Why should we mine software repositories?
4. When should we mine software repositories?
5. How should we mine software repositories?

What is a software repository?

What is a software repository?

A ***software repository*** represents a record-keeping database that stores data about artifacts of a complex computer-based system. It tracks changes applied to the artifacts and stores corresponding meta data.

What is a software repository?

A ***software repository*** represents a record-keeping database that stores data about artifacts of a complex computer-based system. It tracks changes applied to the artifacts and stores corresponding meta data.

Types of software repositories?

What is a software repository?

A ***software repository*** represents a record-keeping database that stores data about artifacts of a complex computer-based system. It tracks changes applied to the artifacts and stores corresponding meta data.

Types of software repositories?

- History-Recording Software Repositories
  - VCS, bug/defect/fault repositories and tracking systems, communication archives (emails, stack overflow posts, wiki, …)

What is a software repository?

A ***software repository*** represents a record-keeping database that stores data about artifacts of a complex computer-based system. It tracks changes applied to the artifacts and stores corresponding meta data.

Types of software repositories?

- History-Recording Software Repositories
  - VCS, bug/defect/fault repositories and tracking systems, communication archives (emails, stack overflow posts, wiki, …), etc.
- Run-Time Data Software Repositories
  - Deployment logs, execution trace logs in development and production environments, etc.

What is a software repository?

A ***software repository*** represents a record-keeping database that stores data about artifacts of a complex computer-based system. It tracks changes applied to the artifacts and stores corresponding meta data.

Types of software repositories?

- History-Recording Software Repositories
  - VCS, bug/defect/fault repositories and tracking systems, communication archives (emails, stack overflow posts, wiki, …), etc.
- Run-Time Data Software Repositories
  - Deployment logs, execution trace logs in development and production environments, etc.
- Code Software Repositories
  - SourceForge, Google Code, GitHub, etc. that contain large numbers of independently developed software projects

What is mining software repositories?

- ***Mining software repositories*** aims at examining and analyzing "*the rich data available in software repositories to uncover interesting and actionable information about software systems and projects*"

What is mining software repositories?

- ***Mining software repositories*** aims at examining and analyzing "*the rich data available in software repositories to uncover interesting and actionable information about software systems and projects*"

- It focuses on uncovering relevant information, relationships, and trends about a particular **evolutionary characteristics** of a system
- Adding **time-dimension** for understanding the evolution of codebase

What is mining software repositories?

- ***Mining software repositories*** aims at examining and analyzing "*the rich data available in software repositories to uncover interesting and actionable information about software systems and projects*"

- It focuses on uncovering relevant information, relationships, and trends about a particular **evolutionary characteristics** of a system
- Adding **time-dimension** for understanding the evolution of codebase

- Empirical and Systematic Investigation to answer research questions
- It falls under the **Quantitative Studies** (benefit: reproducible)

Why should we mine software repositories?

To answer **research questions** that fall under the following two categories:

1. **Market-basket question (MBQ)**: *if A occurs then what else occurs on a regular basis?* The answer is a set of rules or guidelines describing situations of trends or relationships. For example, *if A occurs then B and C happen X amount of the time.*
2. **Prevalence questions (PQ)** that include metric and boolean queries. For example, *was a particular function added/deleted/modified?* Or *how many and which of the functions are reused?*

The questions asked indicate the purpose of the mining approach.

Why should we mine software repositories? Research Questions Examples

- **Bug Prediction:** What factors (e.g., code churn, file complexity) contribute to the likelihood of a file containing a bug?
- **Developer Collaboration:** How do developers collaborate on a software project? What are the patterns of communication and collaboration within a development team?
- **Code Review Effectiveness:** How effective are code reviews in finding and fixing defects? What factors influence the effectiveness of code reviews?
- **Developer Productivity:** What factors (e.g., experience, workload) influence developer productivity? Can we identify patterns that lead to increased or decreased productivity?
- **Software Evolution:** How does software evolve over time? What are the patterns of change in software projects, and how do they impact software quality?
- **Defect Prediction:** Can we predict the likelihood of a file containing a defect based on its characteristics (e.g., complexity, size)?
- **Dependency Analysis:** How do dependencies between software components impact software maintenance and evolution? Can we identify patterns of dependency that lead to more maintainable software?
- **Contributor Behavior:** What factors influence the behavior of contributors in open-source projects? How do these factors impact project success and sustainability?
- **Code Ownership:** How is code ownership distributed in software project? How does code ownership impact software quality and maintenance?

Why should we mine software repositories?

Evolutionary task category	Approaches
Evolutionary couplings/patterns	Bieman *et al.* [79], Canfora and Cerulo [56,57], Fischer *et al.* [23,27], Gall *et al.* [20,26,76], Hassan and Holt [69], Kagdi *et al.* [36], Shirabad *et al.* [37–39], Williams and Hollingsworth [48], Zimmermann *et al.* [15,33], Ying *et al.* [34]
Change classification/representation	Antoniol *et al.* [62], German [80], Hindle and German [29], Holt and Pak [75], Kim *et al.* [30], Mockus and Votta [77], Nikora and Munson [73]
Change comprehension	Beyer and Noack [81], Burch *et al.* [82], Chen *et al.* [22], Chen *et al.* [83], Cubranic *et al.* [60,61], Gall *et al.* [76], Görg and Weißgerber [78], Hindle and German [29], Holt and Pak [75], Kim *et al.* [84], Purushothaman and Perry [67,68], Claudio [85], Robles *et al.* [44], Van Rysselberghe and Demeyer [53], Venolia [86]
Defect classification and analysis	Anvik *et al.* [63], German [21], Livshits and Zimmermann [35], Menzies *et al.* [52], Nagappan *et al.* [87], Ostrand and Weyuker [43], Sandusky *et al.* [42], Sliwerski *et al.* [28], Williams and Hollingsworth [45,46]
Source code differencing	Maletic and Collard [31], Neamtiu *et al.* [71], Raghavan *et al.* [70], Sager *et al.* [88]
Origin analysis and refactoring	Dig *et al.* [89,90], Godfrey *et al.* [72,91], Görg and Weißgerber [49,78], Henkel and Diwan [92], Kimand Notkin [55], Ratzinger *et al.* [54], Tu and Godfrey [74], Weißgerber and Diehl [93], Zou and Godfrey [24]
Software reuse	Selby [47], Van Rysselberghe and Demeyer [32], Xie and Pei [94]
Development process and communication	Dinh-Trong and Bieman [40], El-Ramly and Stroulia [95], Hayes *et al.* [59], Huang and Liu [64], Mockus *et al.* [96], Ohba and Gondow [58], Ohira *et al.* [65,66], Ying *et al.* [50]
Contribution analysis	Koch and Schneider [97], Mockus *et al.* [96], Robles *et al.* [41,98]
Evolution metrics	Capiluppi *et al.* [51], Godfrey *et al.* [72,91], Menzies *et al.* [52], Nagappan *et al.* [87], Nikora and Munson [73], Tu and Godfrey [74]

Source: Kagdi et al., *A survey and taxonomy of approaches for mining software repositories in the context of software evolution,* Journal of Software Maintenance and Evolution, 2006

When should we mine software repositories?

- If the research questions can be addressed by MSR.
    - What kind of information you need to mine to answer the question?
    - Does there exists a software repository from which you can mine such information?
    - Can you create a repository if it does not exist?
- Can you mine “sufficient” information to answer your research question.
    - How do determine “sufficient”?
        - **Conduct statistical tests** to ensure that your findings are statistically significant (i.e., they are not happening by chance)
        - **Statistical analysis test results** will provide you **confidence** in your results (e.g., p-value, 95% Confidence Interval).
        - Low confidence means your mined information is not sufficient

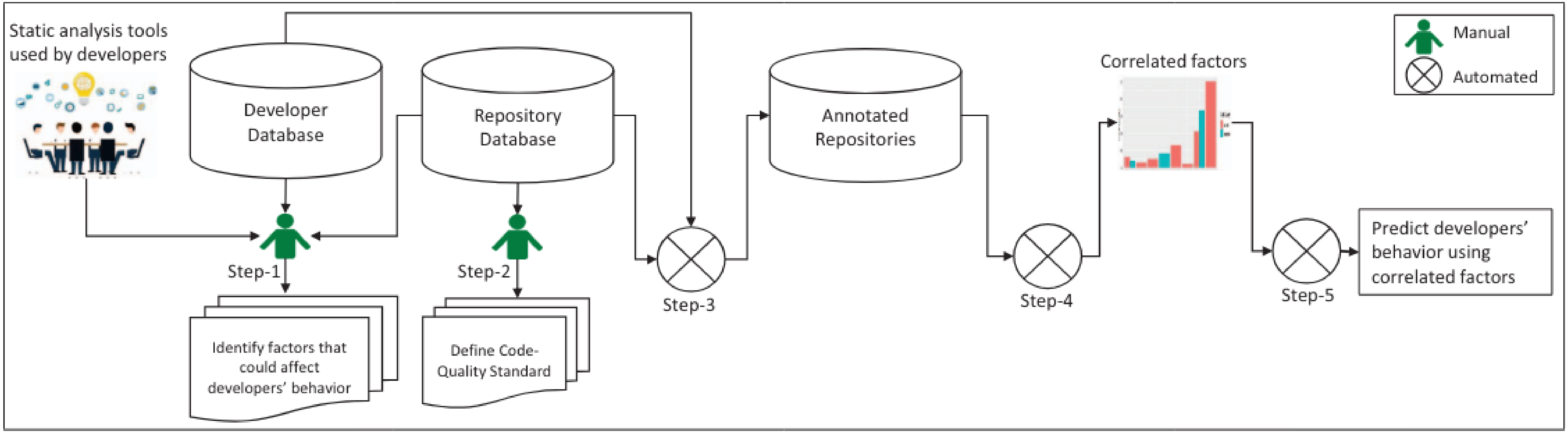
MSR Example Research Study

- Example use case: **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards**

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

- Example use case: **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards**



Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

- Example use case: **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards**
- **Banned API standard at Microsoft**: List of APIs in C/C++ that are unsafe to use and can be potentially exploited

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

- Example use case: **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards**
- **Banned API standard at Microsoft**: List of APIs in C/C++ that are unsafe to use and can be potentially exploited
- **RQ1:** What factors correlate with development teams' adherence to the Banned API Standard?
- **RQ2:** Can the correlated factors predict adherence to the Banned API Standard?
- **RQ3:** Can development team's past activity, encoded in terms of correlated factors, predict future standard adherence?

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

RQ1: What factors correlate with development teams' adherence to the Banned API Standard?

Factor	Overall Distribution preferred	Overall Distribution discouraged	Correlation r	Correlation 95% CI	Correlation p
CareerStageName⋆	20,175	116,592	0.058	[0.051, 0.068]	ϵ
YearsOfMSExperience⊕	50,657	245,438	0.068	[0.062, 0.074]	ϵ
EmployeeLevel⊕	72,069	356,884	−0.104	[−0.108, −0.099]	ϵ
JobTitleName⋆	69,120	352,176	0.080	[0.077, 0.083]	ϵ
StandardTitle⋆	69,092	351,920	0.147	[0.144, 0.150]	ϵ
AuthorChurn⊕	80,243	428,150	−0.109	[−0.113, −0.105]	ϵ
WorkLocation⋆	9,779	54,943	0.204	[0.189, 0.220]	ϵ
Department⋆	9,543	53,330	0.447	[0.438, 0.463]	ϵ
ProjectType⋆	22,451	109,393	0.367	[0.361, 0.377]	ϵ
ProjectName⋆	23,190	136,527	0.407	[0.403, 0.417]	ϵ
IsInternalMSEmployee⋆	93,663	479,946	0.025	[0.023, 0.028]	ϵ
EmployeeType⋆	9,793	55,368	0.009	[0.001, 0.022]	0.141
SourceLOC⊕	93,073	473,892	−0.018	[−0.022, −0.015]	ϵ
RepoLOC⊕	93,675	479,976	−0.064	[−0.068, −0.060]	ϵ
FileChurn⊕	93,423	478,562	0.137	[0.134, 0.141]	ϵ
SourceIntermittentChanges⊕	78,473	394,409	0.061	[0.058, 0.064]	ϵ
TeamSize⊕	71,925	356,287	0.003	[−0.002, 0.008]	0.167
RepositoryOrganization⋆	93,675	479,969	0.117	[0.114, 0.120]	ϵ
RepositoryProject⋆	93,667	479,437	0.190	[0.187, 0.193]	ϵ
Repository⋆	93,612	478,372	0.330	[0.328, 0.333]	ϵ
IsTestCode⋆	93,073	473,892	0.064	[0.061, 0.066]	ϵ
IsActiveCode⋆	93,675	479,976	0.065	[0.062, 0.068]	ϵ
RuleType⋆	93,670	479,974	0.480	[0.477, 0.483]	ϵ
QualityStandardApplicability⊕	93,675	479,976	0.023	[0.019, 0.027]	ϵ
QualityStandardDensity⊕	93,675	479,976	−0.087	[−0.091, −0.083]	ϵ

⋆: categorical or dichotomous factor; ⊕: numerical factor.

Confidence in findings

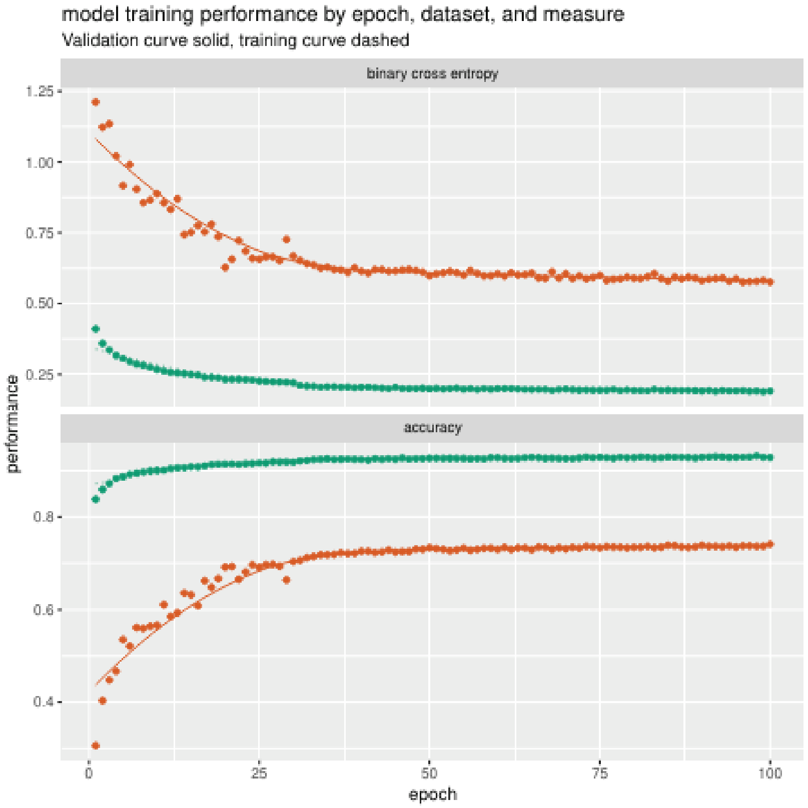
Findings

Factors related to developers' coding experience (role, years of experience, career stage, author churn), complexity (code base size, number of contributors to the source file, file churn), work environment (department, team, project, work location), motivation to adhere to a code-quality standard (test vs. non-test code, and external vs. internal code), and whether the developer is an internal or an external employee significantly correlate with development teams' adherence to the Banned API Standard.

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

RQ2: Can the correlated factors predict adherence to the Banned API Standard?



Confusion Matrix			Evaluation Metrics	
	reference			
prediction	preferred	discouraged	Accuracy:	0.9175
preferred	1,194	226	95% CI:	[0.9113, 0.9234]
discouraged	450	6,322	AUC-ROC:	0.8459
total	1,644	6,548		

prediction over code bases

An artificial-neural-network (ANN)-based model can predict whether developers will use preferred or discouraged APIs with 92% accuracy.

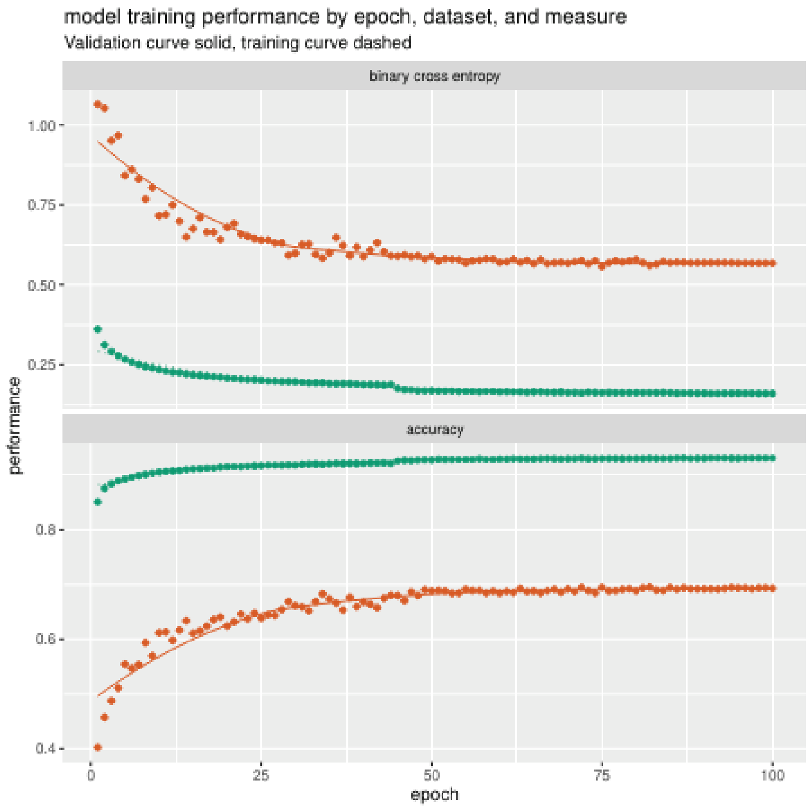
Confidence in findings

Findings

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

MSR Example Research Study

RQ3: Can development team's past activity, encoded in terms of correlated factors, predict future standard adherence?



Confusion Matrix			Evaluation Metrics	
	reference			
prediction	preferred	discouraged	Accuracy:	0.8633
preferred	9,704	1,726	95% CI:	[0.8605, 0.8661]
discouraged	6,431	41,823	AUC-ROC:	0.7879
total	16,135	43,549		

prediction over time on one large code base

For the large code base with a decade of development activity, an ANN-based model can predict if a team will use preferred or discouraged APIs in the future with 86% accuracy based on 7 factors computed from past development activity.

Confidence in findings

Findings

Manish Motwani and Yuriy Brun, **Understanding Why and Predicting When Developers Adhere to Code-Quality Standards, ICSE-SEIP 2023**

How should we mine software repositories?

- **RepoDriller**: a Java framework for mining software repositories
  - Extract information from Git repositories
    - Commits, branches, tags
    - Developers' info
    - Modifications and diffs
    - Source code
  - Quickly export CSV files
  - Integration with Static Analysis tools and Code Parsers (Eclipse JDT, Java Parser)
- **PyDriller**: Python version of RepoDriller
- **Tool-specific APIs** (e.g., Git, GitHub API, GitLab API, Bugzilla API, JIRA API, Apache Spark, etc.)

https://github.com/mauricioaniche/repodriller https://github.com/ishepard/pydriller

MSR Challenge – International Conference

Challenge

The challenge is open-ended: participants can choose the research questions that they find most interesting. Our suggestions include:

1. What types of issues (bugs, feature requests, theoretical questions, etc.) do developers most commonly present to ChatGPT?
2. Can we identify patterns in the prompts developers use when interacting with ChatGPT, and do these patterns correlate with the success of issue resolution?
3. What is the typical structure of conversations between developers and ChatGPT? How many turns does it take on average to reach a conclusion?
4. In instances where developers have incorporated the code provided by ChatGPT into their projects, to what extent do they modify this code prior to use, and what are the common types of modifications made?
5. How does the code generated by ChatGPT for a given query compare to code that could be found for the same query on the internet (e.g., on Stack Overflow)?
6. What types of quality issues (for example, as identified by linters) are common in the code generated by ChatGPT?
7. How accurately can we predict the length of a conversation with ChatGPT based on the initial prompt and context provided?
8. Can we reliably predict whether a developer's issue will be resolved based on the initial conversation with ChatGPT?
9. If developers were to rerun their prompts with ChatGPT now and/or with different settings, would they obtain the same results?

https://2024.msrconf.org/track/msr-2024-mining-challenge?#Call-for-Mining-Challenge-Papers-

Reminders

- **Homework 1 is due on Wednesday! (May 1)**
  - Use lecture notes and readings on software design patterns
  - Open-ended, so don’t wait until last minute!
- **Literature review and project Plan assignment due next Wednesday! (May 8)**
  - You need to read at least 10 papers related to the project you selected.
  - We will discuss how to draft this in next class, but please start working on the assignment if you haven’t started yet

In-class exercise 2: software debugging using git

- Form groups on Canvas (if not done already) and start working
- Submission due by tomorrow end of the day