Final Project

17-712: Fantastic Bugs and How to Find Them (Spring 2023)

The project is one of the main components of this course. It is intended to provide an opportunity for you to get hands-on experience in developing a domain-specific program analysis or automated testing solution.

The project can be performed **individually** or in teams of **up to three**. Please declare the teams at the time of proposal submission.

Proposal Due: March 16, 2023 by 11:59pm
Checkpoint 1: March 30, 2023 by 11:59pm
Checkpoint 2: April 13, 2023 by 11:59pm
Presentation: April 27, 2023 (in class)
Report Due: May 5, 2023 by 11:59pm

Scope

Projects should be related to analyzing software in some specific domain and must have a concrete implementation component, but otherwise the topic can be of the students' choosing. The project is deliberately open-ended, the main requirement being that the problem, technique, or tool be specific to a domain of your choice.

PhD students are expected to pick a project topic that explores an open research question, usually aligning with their own thesis work. Masters and undergraduate students are welcome to perform research, but can also pick an engineering-oriented project as long as it engages with large-scale real-world software: either the software-analysis tooling or the target applications should be in regular widespread use. The teaching staff will help students refine their project scope to ensure it meets learning objectives while being appropriately sized for completion within the semester.

You should expect to spend 30-40 hours on the project per person over six weeks from the proposal due date to the day of presentations. This means that projects involving two or more team members are expected to be of a slightly larger scope than individual projects.

Proposal [20 pts]

Submit your project proposal as one PDF of ~2 pages on Canvas. It should include:

- **Name**(s) of project members (1–3 names). The team should not change after the proposal except in extraordinary circumstances.
- **Selected domain**: For example, “web applications”. It does not have to be a domain that we have covered in class.
- **Target class of bugs**: Are you looking for crashes, correctness issues, SQL injections, performance bugs, etc.? If possible, point to at least one bug report from a target application (not necessarily the one you will analyze) or a news article to show an example of the bug you are hoping to find. This will make sure you know what you are looking for. If you can’t find such an example, document why.
- **Summary of Proposed Task**: Clearly summarize what you are trying to do, in about 1–2 paragraphs. A good rule of thumb is to think of this as writing the description of a GitHub issue that you are assigning to yourself.
    - Define a problem and solution: Briefly mention what exists right now, what you intend to do, and why it is important.
    - The proposed task should be estimable. For example, “I will build a tool to find bugs in smart contracts” is not estimable, because it doesn’t say what concrete things you are going to do.
    - It should be *testable*. If you complete the task, it should be possible to go back to the proposal and validate whether you have achieved what you set out to do.
- **Technical Details**: Expand on your proposed task by providing any more details as necessary. For example, if you plan to add a feature in a tool that reads a configuration file, you can mention here (a) the envisioned format of the config file, and (b) what components in the tool you will modify. Think of this as both a concrete specification for your task and a way for you to avoid getting surprised about something you didn’t think about later (e.g., not being able to find where in the source code to make changes to a tool). It is OK if the specification is refined as you work on the project. Nobody can plan everything perfectly ahead of time.
- **Evaluation Plan**: Describe in about 1–2 paragraphs how you will validate your contribution. Identify at least two target programs that you will run on (though you may change this choice later if you wish). Use your experience from the exploratory assignment to do your homework early; for example, try to build your chosen target program or identify entry points for dynamic analysis if required. How will success be measured? Are you going to collect specific numbers (e.g., coverage or analysis run-time) or will your evaluation be qualitative (e.g., effort required)?
- **Risks**: Identify likely technical issues that may come up which might prevent you from being able to complete your tasks. It is better to document known risks ahead-of-time so that you are prepared for the unknown. For example, if you are doing a researchy project, one risk might be that your idea does not actually end up improving an existing technique. Or perhaps you don’t find the bugs you thought you would find. For each risk, briefly mention a mitigation strategy: what alternative would you try or how would you refine the scope of your project if you encounter this risk?

Checkpoints [5 pts x 2]

Each of the checkpoints will be short and intended to make sure that you are making progress. The course staff will reach out to you if we feel that you are falling behind or struggling. The checkpoints will be text boxes on Canvas that you will fill out with about a 1–2 paragraph summary of: *How is the project going? What have you done so far? What has changed in your task description since the proposal / last checkpoint?*

Presentation [30 pts]

The presentation time limit (est. 5-8 minutes) will be announced later, depending on how the teams form and how many presentations we can fit within a 80-minute class slot.

The structure of the presentation can vary depending on what kind of project you are doing. In general, the presentation should address the following:

- The domain and problem; why is it important
- What are some domain-specific challenges in solving the problem?
- Your proposed solution
- What domain-specific insights, if any, did you use to create your solution?
- Your results (implementation effort, evaluation, findings)
- Any additional insights: what things were unexpectedly hard, what did you try and then later ditch, if you had conversations with any open-source developers then what was their outcome, if you had to do it again what would you do differently, etc.

You will be asked to submit your slides on Canvas and deliver your presentation in class. Each team member is expected to be present and speak at least partially in the presentation. There will be time for a short Q&A after the presentation.

Final Report [40 pts]

The final report is due about a week after the presentation. The report should answer similar questions as the presentation, but provide concrete details that can't be covered in a short talk:

- The domain and problem; why is it important
- What are some domain-specific challenges in solving the problem?
- Your proposed solution
- What domain-specific insights, if any, did you use to create your solution?
- Your results (implementation effort, evaluation, findings)
- Any additional insights: what things were unexpectedly hard, what did you try and then later ditch, if you had conversations with any open-source developers then what was their outcome, if you had to do it again what would you do differently, etc.

You can structure your project (in terms of headings, etc.) in any way you like. We expect project reports to be anywhere from 5 to 10 pages (Letter-sized, 12pt). Feel free to use a conference-paper-style layout if you wish to. If you are doing PhD-level research, you can structure your project report as if it were a submission to a software engineering workshop specialized in the domain where you are working. You are encouraged to actually submit this work for peer review.