CS 520
Final project description

Final projects will be completed in teams of 4 or 5 students. Each team is responsible for a single project.
You should select a team and a project by **Tuesday, October 9, 2018, 9:00AM EDT**.
Your mid-point check-in will be on **Tuesday, November 13, 2018, 9:00AM EST**.
The final project will be due **Tuesday, December 11, 2018, 11:55 PM EST**.
There are five options for a final project (each team will do one):

1. MSR 2019 Mining Challenge
2. Replication study
3. Model Inference for Inferring Processes
4. EleNa: Elevation-based Navigation
5. Self-defined software engineering research project

MSR 2019 Mining Challenge

The Mining Software Repositories conference runs an annual challenge in which they provide a dataset and ask you to answer research questions about the dataset. Read the description of this year's dataset, research questions, and challenge here:
https://2019.msrconf.org/track/msr-2019-Mining-Challenge#Call-for-Mining-Challenge-Papers

Replication study

A replication study takes an existing research paper, replicates its experiments on the same data, and then extends the experiments to expanding that data set on which the experiments are run. For this project, we highly recommend selecting a paper with publicly available dataset and code to execute the experiments. The project involves a write up describing the process of replicating the experiments, deviations in the achieved results from the original ones reported in the paper, and lessons learned from applying the experiments to new data.

Here is a list of several papers that are good candidates for replication:

1. Automatic generation of oracles for exceptional behaviors from Javadoc comments.

   Paper: https://dl.acm.org/citation.cfm?id=2931061

   Source code: https://github.com/albertogoffi/toradocu

2. SimFix: Automated program repair

   Paper: http://sei.pku.edu.cn/~xiongyf04/papers/ISSTA18a.pdf

   Source code: https://github.com/xgdsmileboy/SimFix

   Dataset: https://github.com/rjust/defects4j

3. EvoSuite: Automated test generation

   Paper: https://dl.acm.org/citation.cfm?id=2685612

   Source code: http://www.evosuite.org/ and https://github.com/EvoSuite/evosuite

   Dataset: https://github.com/rjust/defects4j

4. Are mutants a valid substitute for real faults in software testing?

   Paper: https://homes.cs.washington.edu/~mernst/pubs/mutation-effectiveness-fse2014.pdf

   Source code and dataset: https://github.com/rjust/defects4j

Model Inference for Inferring Processes

The goal of this project is to learn how automated model inference works and its limitations, then to apply it to real-world traces to infer models of real-world phenomena, and learn something interesting from the resulting models.

What is model inference?

Model inference uses a set of observations of how a process executes to the produce a model of everything the process can do. For example, imagine watching ten different people, each, bake a pie, and writing down every step each of them takes. What you end up with is ten traces of executions of the pie baking process. Feed these ten traces into a model inference tool, and it will produce a model of pie baking. The model has a start state and an end state, and everything in between is some way of describing different possible traces. Every trace through the model (from the start state to the end state) is a way to bake a pie. Typically (but depending on the tool), this model will include the ten traces you already observed, but it may include others as well. These others are generalizations of the observed traces.

Of course, model inference doesn't just work for pie baking. The more typical approach is to execute a software system many times and record logs of these executions. These logs (traces) could be something like every method that executes, or it could be the logging information developers chose to put into the system. Feeding these log traces into a model inference tool produces a model of possible system behavior, some observed and some unobserved.

Tasks

Your task is to learn about model inference, select a reasonable way to generate traces, develop experiments to evaluate a specific model inference tool (or several tools), use these model inference tools to infer models, and finally, study the models and experiments.

1. Model inference.

   Synoptic and InvariMint are two good tools to know about. Synoptic is relatively simpler, but InvariMint allows running multiple inference algorithms at once. You can start here:

   https://github.com/ModelInference/synoptic/blob/master/README.md

   and also look at two research papers about the tools:

   Synoptic: http://people.cs.umass.edu/~brun/pubs/pubs/Beschastnikh11fse.pdf

   InvariMint: http://people.cs.umass.edu/~brun/pubs/pubs/Beschastnikh15tse.pdf

2. Generating traces.

   Be creative. You can find real-world processes (like baking a pie) for which you can manually write traces. Maybe you can write down plots of movies in a common (small vocabulary) language. Maybe you can use a diary of daily behavior? Or you can find an interesting software system to generate logs.

3. Experiments.

   Lots of things can affect model inference. How many traces there are. How diverse the traces are. How redundant the traces are. And then, there are the model inference tools themselves. Synoptic (and InvariMint) use a set of invariants. Changing this set will affect the models you get. Design experiments that change something about the traces or inference algorithms.

EleNa: Elevation-based Navigation

Navigation systems optimize for the shortest or fastest route. However, they do not consider elevation gain. Let's say you are hiking or biking from one location to another. You may want to literally go the extra mile if that saves you a couple thousand feet in elevation gain. Likewise, you may want to maximize elevation gain if you are looking for an intense yet time-constrained workout.

The high-level goal of this project is to develop a software system that determines, given a start and an end location, a route that maximizes or minimizes elevation gain, while limiting the total distance between the two locations to x% of the shortest path.

Components:

Your software system will most likely have four main components:

1. Data model that represents the geodata.
2. A component that populates the data model, querying, e.g., OpenStreetMap.
3. The actual routing algorithm that performs the multi-objective optimization.
4. A component that outputs or renders the computed route.

While all components are necessary for a working prototype, you may choose to focus on some of them in greater detail. For example:

- If you focus on developing and experimenting with several routing algorithms, it is sufficient to have a simple interface for entering the start and end location and a simple output that represents the route.
- If you focus on a sophisticated UI with proper rendering of the computed route, it is sufficient to have a basic data model and routing algorithm.

Resources:

- The A⋆ algorithm: https://en.wikipedia.org/wiki/A*_search_algorithm
- Dijkstra's algorithm: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- OpenStreetMap wiki: http://wiki.openstreetmap.org/wiki/Main_Page
- The following paper, in particular Section 2, provides a very accessible introduction and overview of metaheuristic search algorithms:
  https://pdfs.semanticscholar.org/9c83/752460cd1024985981d4acaa7bc85e15c0f7.pdf

Self-defined software engineering research project

As the title suggests, this is an open-ended option for student groups who can define and execute their own research project related to software engineering.

Finding a good research project requires reading research papers and thinking of an extension. You could look at the papers published at the latest premier software engineering conferences, e.g.,

- ICSE 2018: https://www.icse2018.org/track/icse-2018-Technical-Papers#event-overview
- ESEC/FSE 2017: http://esec-fse17.uni-paderborn.de/program_research_track.php
- ICSE 2017: http://icse2017.gatech.edu/?q=technical-research-accepted

and either replicate one of those papers' experiments on another dataset, or extend either a technique or the evaluation described in a paper.

This project is appropriate for students already with some research experience.

If you want to see some sample research ideas, you can contact Yuriy via email.