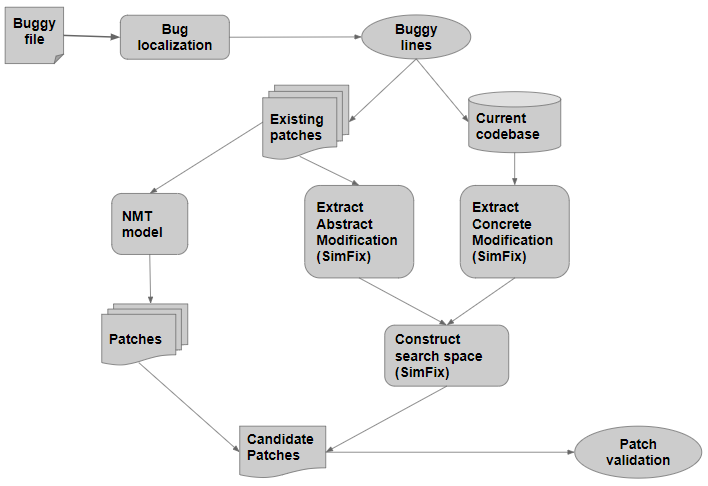
Preliminary Project Proposal (SimFix++)

Yangruibo Ding (yd2447)

Joe Huang (jch2220)

Columbia University, New York, NY

## What we plan to do?



**Fig1. SimFix++ Workflow**

We give our tool a name of “SimFix++” because the tool will be developed based on the SimFix1 tool.

We would like to design an automatic bug fixing tool, which will take advantage of both existing patches and current codebase. The existing patches are to capture the potential bug-fixing patterns and the current codebase is to search the similar correct code. We reserve the full functionalities of SimFix, but we would like to add a neural-machine-translation model to analyze the existing patches. By applying the seq2seq model (one NMT model), we can generate more possible patches directly for the validation phase. This may take some time to validate the patch, but the number of bugs we can fix should increase, especially for those one-line bugs. We use OpenNMT2 tool to implement seq2seq model.

In Fig1, we show the workflow of our tool. The general input of our tool will be the buggy files, and the bug localizer will give us the buggy lines. I want to mention that, as planned before, we hoped to use Natural Bug Finders to localize the bugs and rank the candidate patches by naturalness score. However, after careful evaluation, the improvement of performance by applying this method may not be significant, so we will give this idea a lower priority to implement. After getting the buggy lines, we will use both the original SimFix patch generation and our NMT model to generate patches, where the NMT model will specifically focus on learning existing patches. Finally, we will utilize patches generated from both techniques to validate.

Note: I have successfully built and run both SimFix tool and OpenNMT, and I ensured that they are available.

## How it is relevant to software engineering?

Program bugs are pretty common in daily software development and could cause unexpected output, security problems, or even software crashes. Maintaining a software system or generating patches for the buggy code always needs a lot of human effort and could be very time-consuming. For instance, in order for developers to generate a patch, they have to review the bug reports, try to reproduce the bug, understand the cause, and then fix it. In addition, the patches need to be validated by several tests and human reviewers, making the process more arduous.

Because some fixing patterns are underlying the repetitive fixing process, we want to automate this process. It is true that not every bug fixing follows a pattern and some would need the creativity of human developer. Therefore, our tool aims to cover those which do follow certain patterns.

## Why we are interested in doing this project?

Yangruibo Ding:

Automating some parts of software engineering is trending in the industrial field. More big companies are focusing on freeing their engineers from time-consuming and repetitive work. When I was an intern, I always heard about some complaints from colleagues in the Q&A department, and they complained about the boring and repetitive stuff they are working with for all day. I have strong interests in machine learning fields (NLP mostly), so I hope to automate some parts in software engineering by applying efficient machine learning methods.

Joe Huang:

The idea of automatic program repair really intrigues me. After doing my midterm paper on investigating the technique, I want to see how certain techniques could be improved and Robin’s project idea really interests me. Hopefully, by combining two ideas, our project could make an improvement to the performance of SimFix and make contributions to the field.

## How do we evaluate our tool?

Our tool tends to complement Simfix tool by adding a neural-machine-translation-based functional branch to improve the performance of original Simfix. Therefore, the evaluation will be based on the same bug database with Simfix used, “Defects4J”. In the previous work of Simfix, it can fix 34 bugs in this bug database. In our opinions, our NMT-model should at least fix some one-line bugs inside “Defects4J”, which is not common in the ones fixed by SimFix, so with our NMT-model, the number of successful fixes is expected to be higher.

## How we plan to deliver the code, documentation and other software artifacts for your project

We will post the codes, documentation, and any relevant files to Github repository, which probably is a private one, and I will add the reviewers of my project as collaborators.