TEST BAYES

- Adavya Bhalla
- Avidant Bhagat

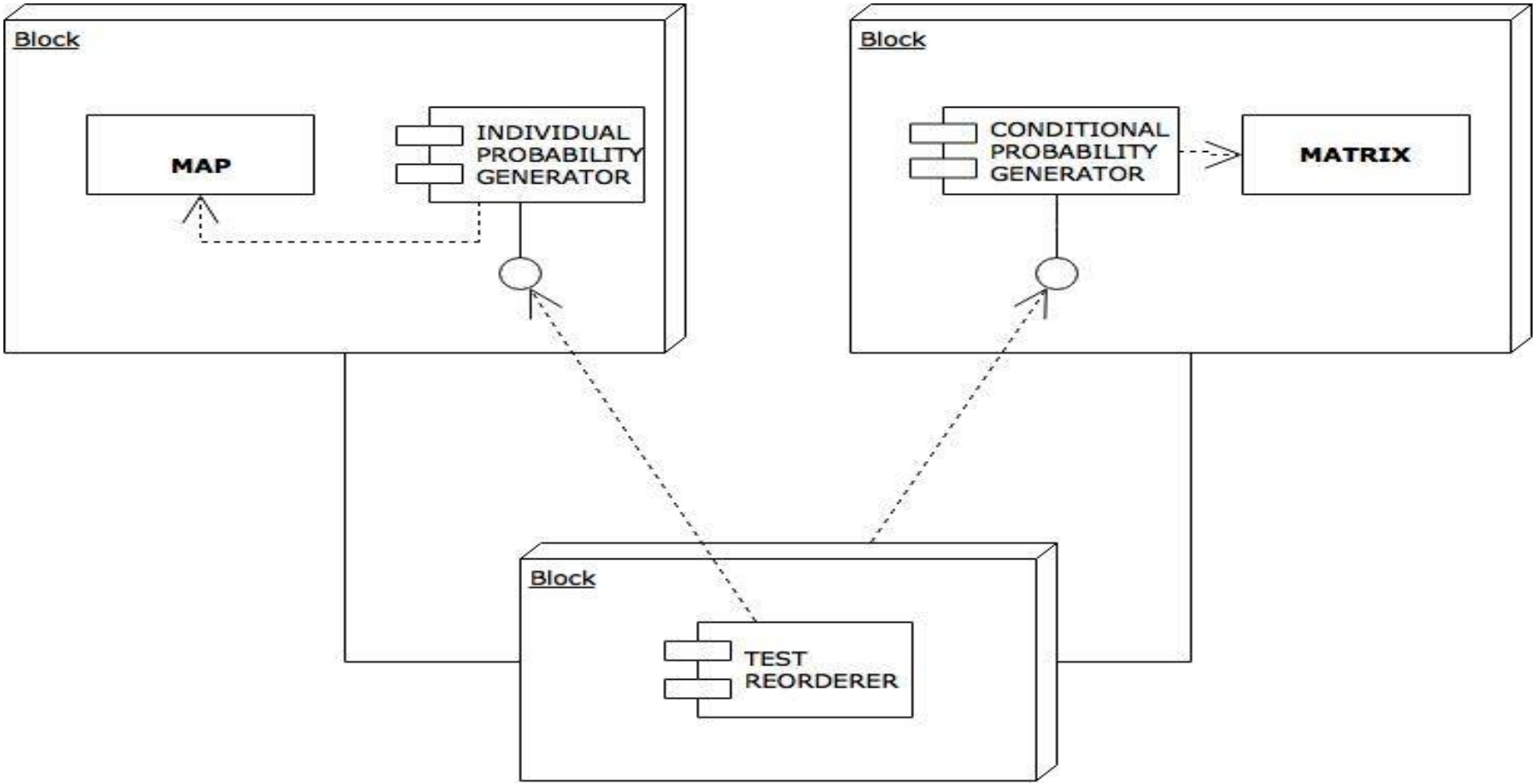
Probabilistic models
for test ordering

Motivation

- Test ordering is hard and often deceptive. Maybe the only failing test is the last one.
- Testing is one of the most critical components of software, typically software will have twice the amount of test code than it has source code.
- It is hard to order tests because we never know what bug we have introduced, but we can speculate based on which tests have passed.
- Today Manual ordering is done based on previous runs to minimize time to failure, but, what if we can automate the process and make it better through mathematical calculations.

Approach

- Extend JUnit and dispatch tests based on probabilistic calculations.
- Our approach takes care of many things, the tests that fail first are tested first, Tests that pass if another test has passed are pushed to the back of the line, tests that are faster are run first.
- We will use conditional probability, test run time, total probability and a moving average to compute which test is most likely to fail and then run it.
- It is a greedy process.


Block
MAP
INDIVIDUAL
PROBABILITY
GENERATOR
Block
CONDITIONAL
PROBABILITY
GENERATOR
MATRIX
Block
TEST
REORDERER

Research Questions

- Our initial hypotheses are:
  - Tests are related and one test's results can affect another test's results
  - Tests that have had a higher probability of failing in the past will have a higher probability of failing now and in the future
  - The amount of time taken to reorder the tests will be significantly lower than the amount of time taken
- We will test these hypotheses using the Evaluation Plan that we will talk about at the end of the presentation. The metrics we will use will be time to failure, and number of tests to failure.

Preliminary results

- We have not found any way to extend JUnit and reorder tests without adding an Annotation to the test classes
- File Storage, Reading, and Writing is easy and allows us to use data collected from multiple machines and runs

Evaluation plan

- We plan on testing the time to failure before and after reordering tests. We will also use a random ordering to see if we do better than random.
- We will test using some injected bugs in CSE 331 projects with large test suites. This will be a nice small case study for our project
- We will also use online repositories and use their test run histories to see whether the time to failure is decreased by using our tools
- We will also see whether using pure conditional probability to greedily test would be better or using all the information we have.
- Once our code base is stable, we will run our code on itself