Including mutation testing as part of a continuous integration workflow

Tim Waterson

Supervisor: Louis Rose

Read title. Starts with definitions (i.e. recap lit review) then project goals and motivations.

What is mutation testing?

Mutation testing measures the effectiveness of a program's test suite.

Most programs now have automated test suites.
Test suites measure program quality.
MT can measure the measurer, can give an indication of test suite quality.
Similar to code coverage which you may have heard of.
That's the **what it does**, now for the **how it works**.

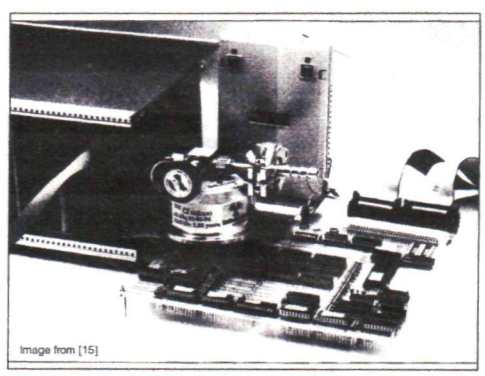
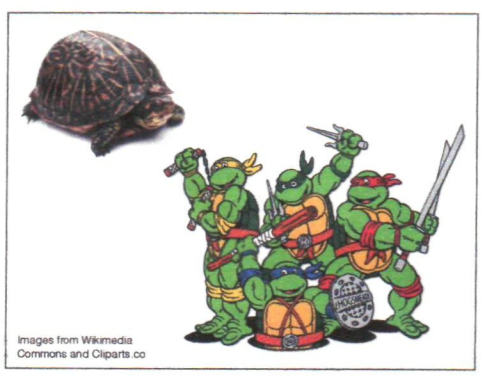


Image from [15]

This machine was constructed to inject faults into hardware.
It bombards CPUs with radiation to trigger bit-flips, sees whether the system can recover.
MT is the equivalent for software — Injects faults into program source code.



Images from Wikimedia Commons and Cliparts.co

Goal is to see whether or not the test suite can distinguish between the original and mutant versions.
If it can't then we don't know which was originally intended.
May point to a system bug.
Produces a mutation score, which is the number of mutants detected, or killed.

A high mutation score is strongly correlated with test suite effectiveness.

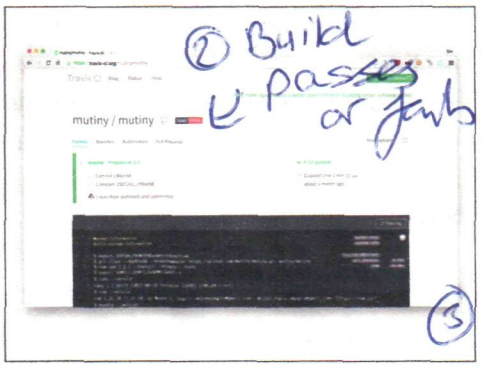
Citations [21] and [22]

This justifies mutation testing's relevance.
Also of interest is the fact that this is not true for code coverage.

What is continuous integration?

The other main theme for this project.

Means for regularly / continuously evaluating software quality.



② Build ↙ passes or fails

Every ~~one~~ pushed commit, ① triggers a build which results in a report.

③ Build log, output from ~~een~~ commands run

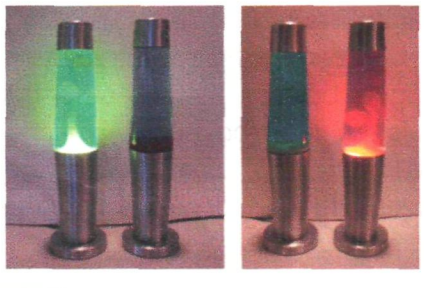


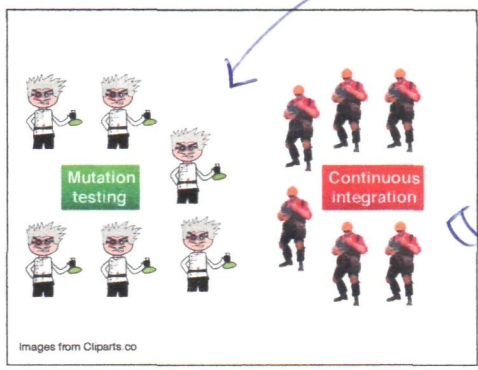
Image from [31]

↙ shows most recent build

~~Empressies~~ binary outcome

Bringing the two concepts together

That's the goal for this project.



Images from Cliparts.co

MT has received a lot of academic interest

CI widely deployed in industry

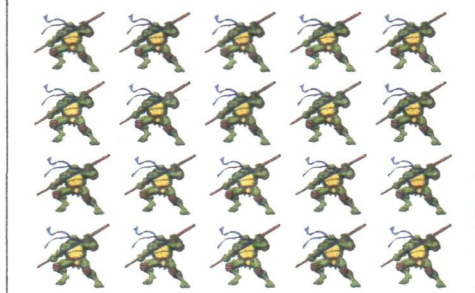
MT isn't. This project hopes to address that by ~~finding~~ enabling MT to be run as CI build

CI builds need to be fast.

Mutation testing is slow.

ID challenge is doing that is MT is slow

CI 10 mins rule. Need to be fast



Images from Cliparts.co

∴ lots of mutants generated

e.g. over for a statement prog.

CI builds need to either pass or fail.

Mutation testing produces a percentage score.

Can't just use 100%.

- equiv. mutants undecidable

(no time to explain fully)

"All respectable software engineering research should have the eventual goal of helping real programmers build better software"

–Jeff Offutt

Quote from [12]

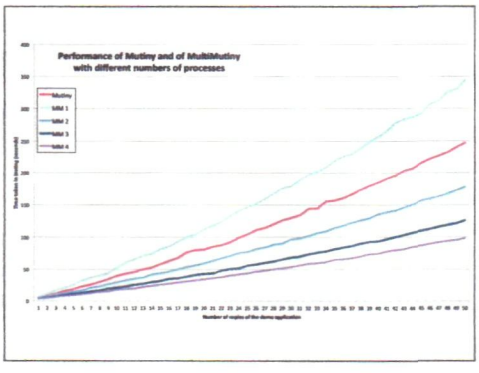
↑ justification for prev aim

Analysis

This section gives the reasons behind several key decisions in the project.

Just dive to explain in this presentation.

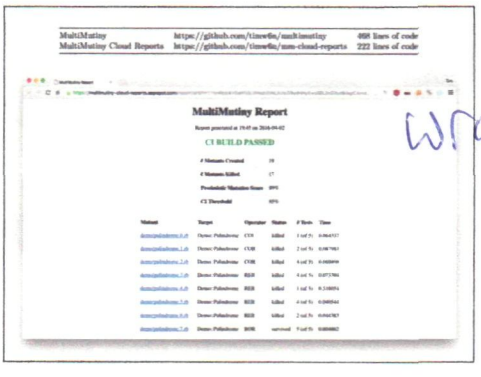
CI build again.
Note this is all the info there is. Already know loads of maths, would be too unwieldy to be in text log... ...hence could compound that won't be seen later.



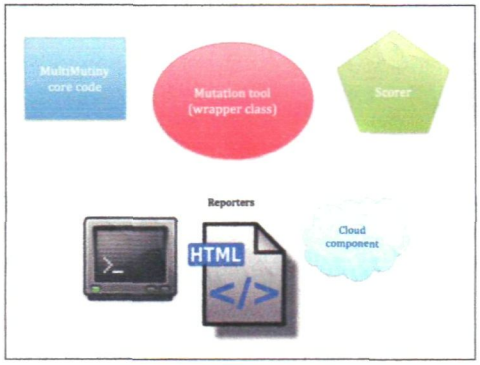
Parallelisation to ~~improve~~ speed improve
Red line original
Purple line for parallel processes.

Design & Implementation

This section gives details regarding the structure of the software produced as part of this project.



2 pkgs, Main and Cloud.
Wrapsaround the Mutation tool.
Multiple different types of reports - cloud one shown here.
Can click on item for details



Tool designed to be highly modular.

[Go through each in turn]

Evaluation

Project	GitHub repository	Description (from GitHub)
event_bus	kevinrutherford/event_bus	A simple pubsub event bus.
full_name	aglicker/full_name	Adds `full_name` method for classes that provide a `first_name` and a `last_name`!
lumberjack	bdurand/lumberjack	A simple, powerful, and very fast logging utility that can be a drop in replacement for `Logger` or `ActiveSupport::BufferedLogger`.

Aim was to support MT for real projects.

So here's three real projs doing real things for real people.

It works!

Project	Mutants generated	Mutants killed	Pessimistic mutation score
event_bus	57	51	89%
full_name	51	51	100%
lumberjack	565	480	85%

Image from Cliparts.co

In short, it functions. Summary statistics given here.

And it's fast!

Project	Mutiny	MultiMutiny	MultiMutiny as a % of Mutiny
event_bus	17.32s	7.20s	41.59%
full_name	15.30s	6.55s	42.79%
lumberjack	142.68s	51.78s	36.29%

Image from Cliparts.co

Node 60%. Speedup as expected.

Conclusion & Further Work

Better pall'n

not decreasing

threshold?

Any Questions?

All square-bracketed references refer to items in the main project Bibliography.