Dynamic Analysis

Software Reengineering
(COM3523 / COM6523)

The University of Sheffield

Lab follow-up

GitLab Login Difficulties

Tech support are currently investigating the log-in problems.

- Log-in appears to work for some students but not for others.

Ensure that you are using the correct GitLab server: stugitlab.dcs.shef.ac.uk

- Requires your university of sheffield login ID (e.g. ac1nw) and password.
- Requires VPN if used remotely.

Repository Mining Task

parent directory of zxing

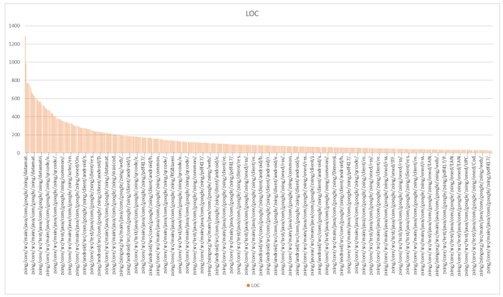
```
ac1nw@TENB88584C2F899 MINGW64 /u/Teaching/Reengineering/2022/practicals
$ reengineering-toolkit/reengineering-toolkit/src/main/scripts/repMining.sh zxing
```

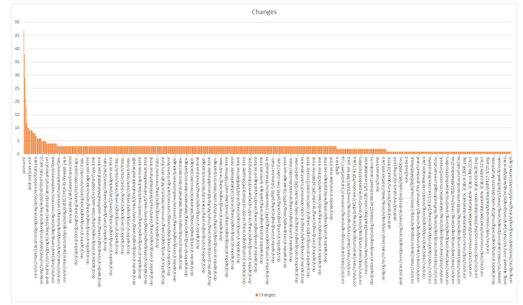
If it takes too long, pressing ctrl-C after a few minutes will produce a curtailed CSV file.

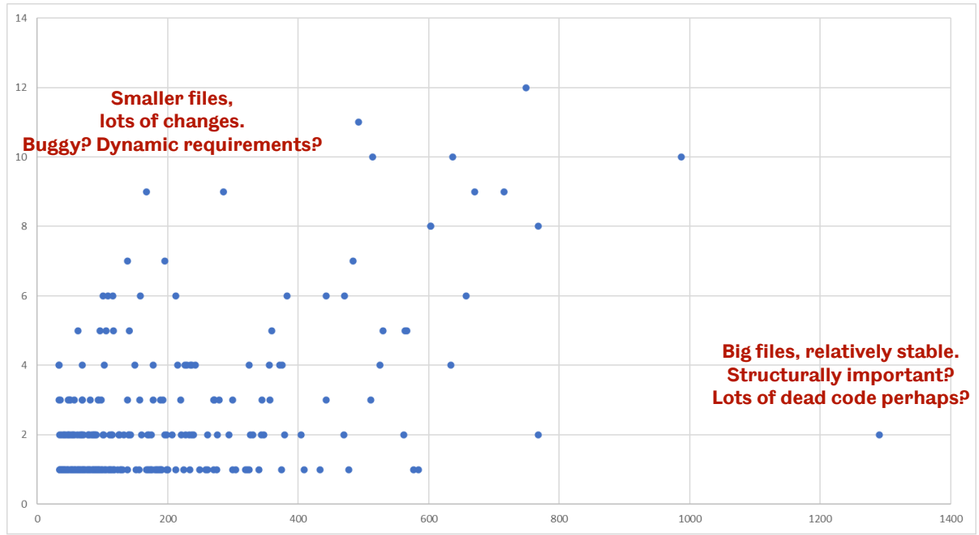
directory containing git repo

```
ac1nw@TENB88584C2F899 MINGW64 /u/Teaching/Reengineering/2022/practicals
$ reengineering-toolkit/reengineering-toolkit/src/main/scripts/listFileSizesForType.sh zxing '*.java' > zxingFileSizes.csv
```

pipe output to different csv file







Dynamic Analysis

The Problem with Static Analysis

Too much information.

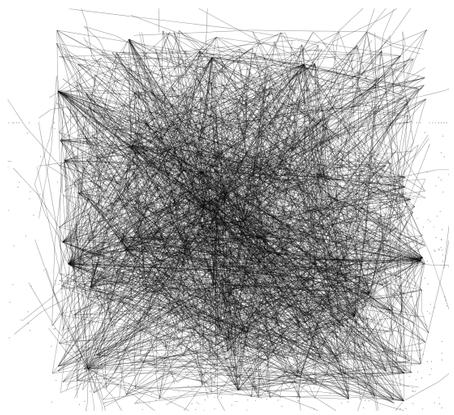
- ZXing contains >510 classes.
- What if we are only interested in a particular use-case?

Inherently **conservative**.

- Can produce lots of relations that are spurious or infeasible.
- E.g. - Produce method calls that are not possible for any actual program execution.

Many things are impossible to determine from source code analysis alone.

- Performance-related questions.
- Issues wrt. particular usage scenarios.



Call graph for Apache Commons Math.

Dynamic Analysis

Extracting information from **program executions**.

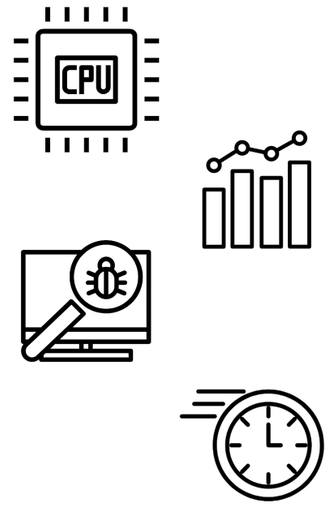
- Function calls, input-events, network-signals, ...
- Call stack depth, object states, output values, parameter inputs, memory usage, clock-time, etc.

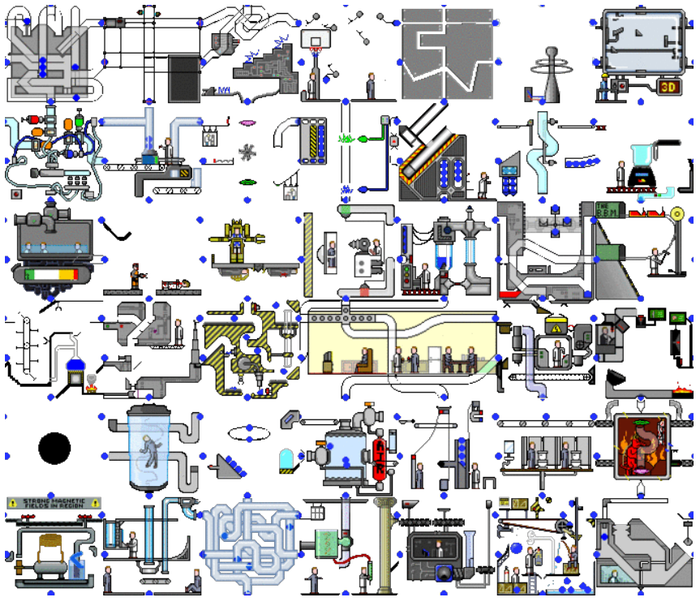
Provide information about "software behaviour".

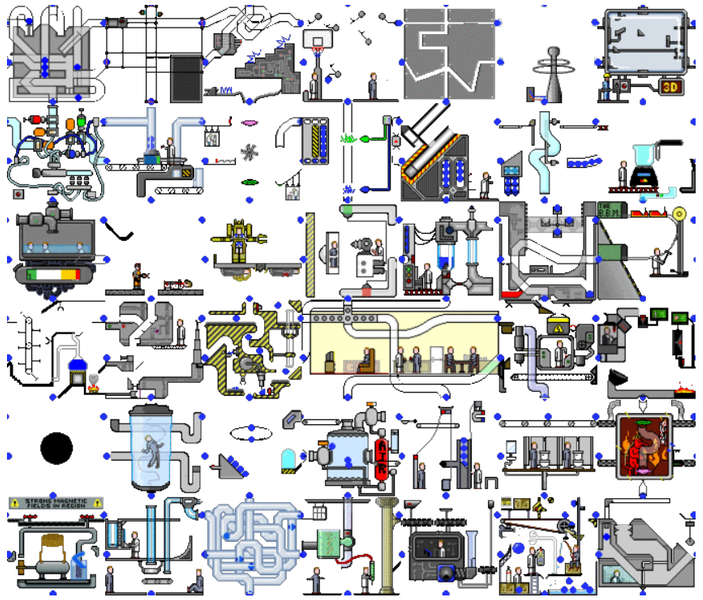
- Which elements are executed, when, for how long, ...

Common dynamic analyses:

- Testing
- Profiling
- Logging
- Debugging







Software Reengineering (COM3523 / COM6523) The University of Sheffield *Dynamic Analysis* 9

Execution Traces

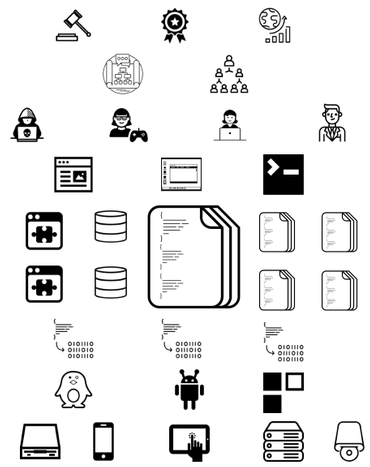
Making a “record” of an execution.

A trace is a collection of time-stamped data-points.

- Precise contents depend on the level at which the trace is collected.
- Can be at any level in the system - from hardware to business process.
- Commonly comprise an event (e.g. a method call), coupled with some data (e.g. system state information, performance data)

Can apply to distributed systems, or monolithic ones.

- Reasoning about order of trace-elements becomes harder in distributed / concurrent systems.



Loggers

Tools to automate the collection of runtime data.

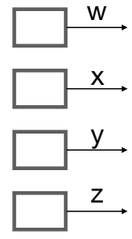
- Need a better alternative than System.out.println(...)!

Can specify:

- **Where** information is funnelled - console, file, database, ...
  - *Logging instructions tend to be inserted into code by developers.*
- **When** logging occurs - for info, debugging, errors,...
- **How** information is presented / formatted.

Lots of logging frameworks

- java.util.logging, slf4j, log4j, ...



Loggers

Tools to automate the collection of runtime data.

Need a better alternative than System.out.println(...)!

Can specify:

Where information is funnelled - console, file, database, ...

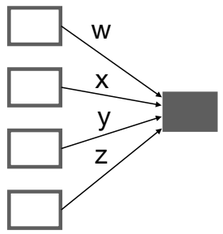
Logging instructions tend to be inserted into code by developers.

When logging occurs - for info, debugging, errors,...

How information is presented / formatted.

Lots of logging frameworks

java.util.logging, slf4j, log4j, ...



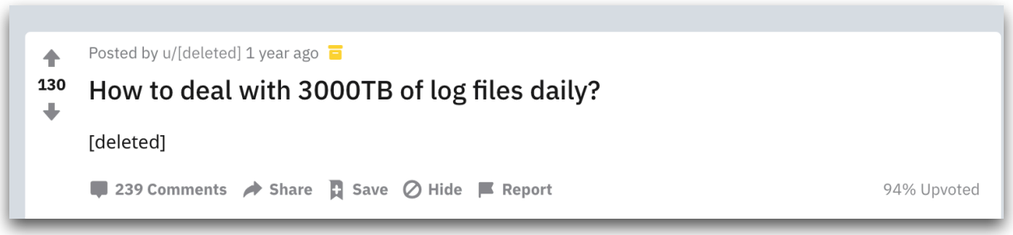
Precision is important

The size of a trace file is impossible to predict in general.

See “Halting Problem”.

Large traces are harder to analyse and manage.

Important to be selective about where, when and what to log.



Trade-offs galore

	Human effort to set-up / remove	Detail / level of information about application	Performance impact	Trace volume

Trade-offs galore

	Human effort to set-up / remove	Detail / level of information about application	Performance impact	Trace volume
Print-statements inserted by developer	😭	🤨	🙂	🙂

Trade-offs galore

	Human effort to set-up / remove	Detail / level of information about application	Performance impact	Trace volume
Print-statements inserted by developer	😭	🤨	🙂	🙂
Using dTrace or bTrace to record OS-level events	🙂	🤨	🙂	🤨

Trade-offs galore

	Human effort to set-up / remove	Detail / level of information about application	Performance impact	Trace volume
Print-statements inserted by developer	😭	🤨	🙂	🙂
Using dTrace or bTrace to record OS-level events	🙂	🤨	🙂	🤨
Automatically instrumenting code	🙂	🤩	🤨	🤨

Trade-offs galore

	Human effort to set-up / remove	Detail / level of information about application	Performance impact	Trace volume
Print-statements inserted by developer	😭	🤨	🙂	🙂
Using dTrace or bTrace to record OS-level events	🙂	🤨	🙂	🤨
Automatically instrumenting code	🙂	🤩	🤨	🤨
Recording inputs and outputs - truly "black-box"	🤩	🤨	🤩	🤩

The Heisenbug

A bug that disappears / alters its behaviour when you attempt to study it.

The Heisenbug

A bug that disappears / alters its behaviour when you attempt to study it.

Named after the “Heisenberg Effect” in quantum mechanics.

The act of observing a system inevitably changes its state.

The Heisenbug

A bug that disappears / alters its behaviour when you attempt to study it.

Named after the “Heisenberg Effect” in quantum mechanics.

The act of observing a system inevitably changes its state.

A risk when the dynamic analysis interferes with the application environment.

The Heisenbug

A bug that disappears / alters its behaviour when you attempt to study it.

Named after the “Heisenberg Effect” in quantum mechanics.

The act of observing a system inevitably changes its state.

A risk when the dynamic analysis interferes with the application environment.

Examples:

- Time-sensitive programs - dynamic analysis can slow the program down.
- Programs that rely on disk access - trace storage can interfere with this.
- Programs that rely on compiler optimisations - trace instrumentation can alter the optimisations,
- ...

The Heisenbug

A bug that disappears / alters its behaviour when you attempt to study it.

Named after the “Heisenberg Effect” in quantum mechanics.

The act of observing a system inevitably changes its state.

A risk when the dynamic analysis interferes with the application environment.

Examples:

- Time-sensitive programs - dynamic analysis can slow the program down.
- Programs that rely on disk access - trace storage can interfere with this.
- Programs that rely on compiler optimisations - trace instrumentation can alter the optimisations,
- ...

Relationship to Static Analysis

Static analysis:

Conservative - considers every possible (and many impossible) executions.

Imprecise (e.g. lots of redundant calls in a call graph).

Sound. Every possible call is in the call graph.

Dynamic analysis:

Partial - only considers a single execution.

Precise - it it occurs in the trace it **must** be possible.

Unsound - observations may not generalise to **every** execution.

Ernst, Michael D. "Static and dynamic analysis: Synergy and duality." *WODA 2003: ICSE Workshop on Dynamic Analysis*. 2003.

Relationship to Static Analysis

Static analysis:

Conservative - considers every possible (and many impossible) executions.

Imprecise (e.g. lots of redundant calls in a call graph).

Sound. Every possible call is in the call graph.

Dynamic analysis:

Partial - only considers a single execution.

Precise - it it occurs in the trace it **must** be possible.

Unsound - observations may not generalise to **every** execution.

Ernst, Michael D. "Static and dynamic analysis: Synergy and duality." *WODA 2003: ICSE Workshop on Dynamic Analysis*. 2003.

Automatically Instrumenting Java Code

Aspect-Oriented Programming

A way to modularise software in terms of **"cross-cutting concerns"**.

Motivated by the high amount of duplicate code in systems.

Capture code that deals with a particular 'aspect' (known as an **"advice"**).

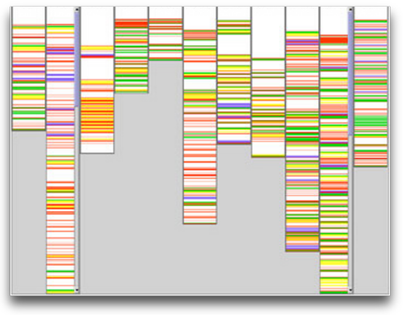
Fragment of code to be used throughout a program.

E.g. File-handling, data-base management, logging...

Advice is **"woven"** into software execution via **"join points"**.

E.g. method entry or exit points, field accesses, etc.

A specification that links advice to a join-point is known as a **"point cut"**.



Cross-cutting concerns from a 19k component. (from Bruntink, van Duersen and Tourwé)

Using Transformer Agents

There are many dedicated Aspect-Oriented Programming Libraries that could be used.

- AspectJ is one of the main ones for Java.
- Too heavyweight.

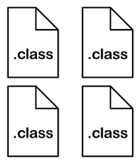
Java includes the ability to write “Transformers” that modify classes during execution.

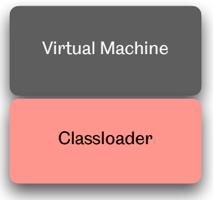
- Provide the infrastructure to implement basic aspect-oriented concepts, including automated logging.

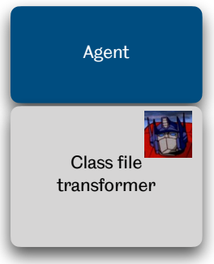
Deployed as “agents”.

- “Attached” to a Java program when it is run, have the ability to alter a class when it is loaded.

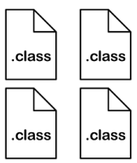
How it works

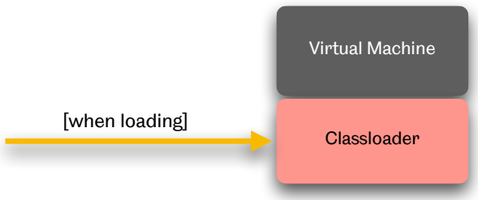


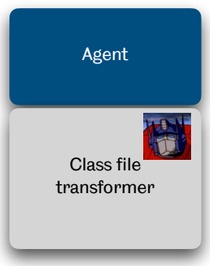




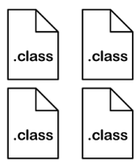
How it works

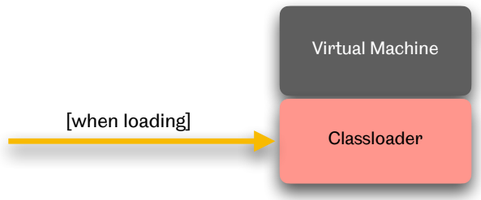


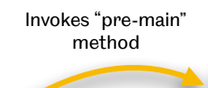


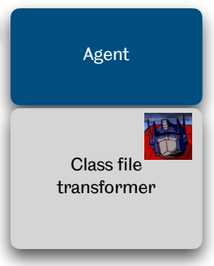


How it works

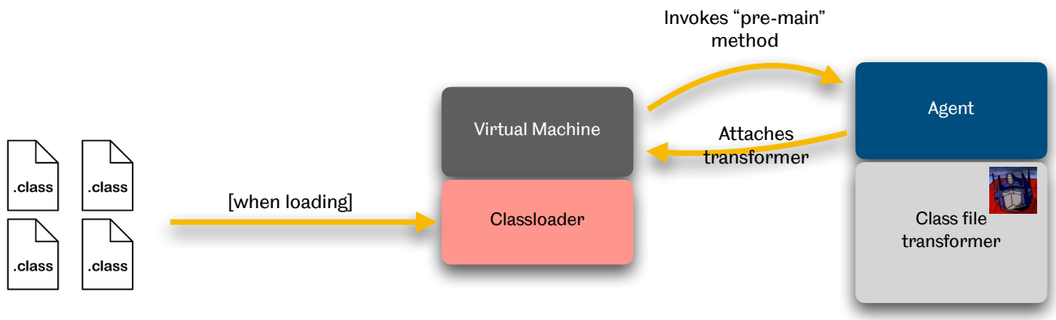




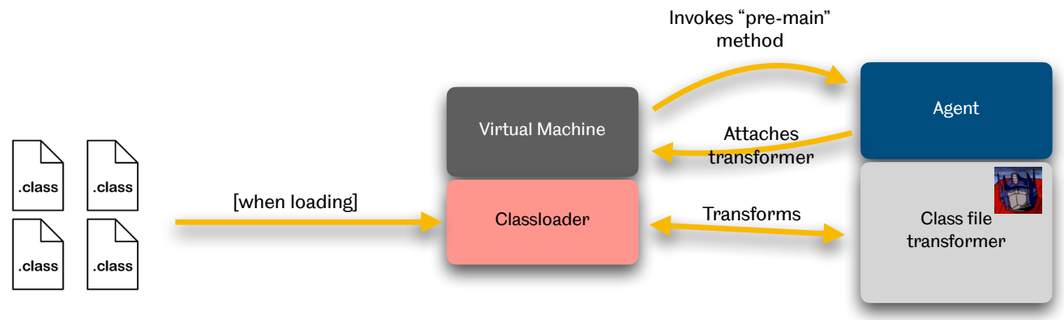




How it works



How it works



Demo

Key Takeaways

Dynamic analysis is the process of recording runtime data.

Commonly operates on “execution traces”.

- Recordings of software executions, captured as sequences of events.

Can be much more precise than static analysis, but can also be less sound.

- Relies on the analyst “covering” all of the relevant software behaviour.

Traces can be very large.