Model-based Testing Without a Model: Assessing Portability in the Seattle Testbed

Justin Cappos and Jonathan Jacky

University of Washington
Seattle, Washington USA
justinc@cs.washington.edu, jon@u.washington.edu

with special thanks to Jeff Rasley

Seattle Testbed

Seattle distributed computing testbed
`https://seattle.cs.washington.edu/`

Seattle programs —

- Coded in a Python subset
- Use a special API
- Run in a “sandbox”, a safe execution environment
- Run on many platforms: Windows, Linux, Mac OS X, BSD, various mobile . . .

Portability

Seattle programs should be *portable*: behave the same on all platforms.

Portability has proved to be a problem —

- On other systems (Java, POSIX, . . .)
- On Seattle itself (file system, network . . .)

Seattle API

Simple API: 32 functions

- File I/O: 6 functions
- Locks: 3
- Debugging: 2
- Threads: 3
- Code verification and evaluation: 2
- Network DNS, UDP, TCP: 13
- Accounting: 2
- Random bytes (crypto): 1

Testing the Seattle API

Unit tests are not enough. We need —

- Many interleavings (to check different orders)
- Handle nondeterminism (for network API)

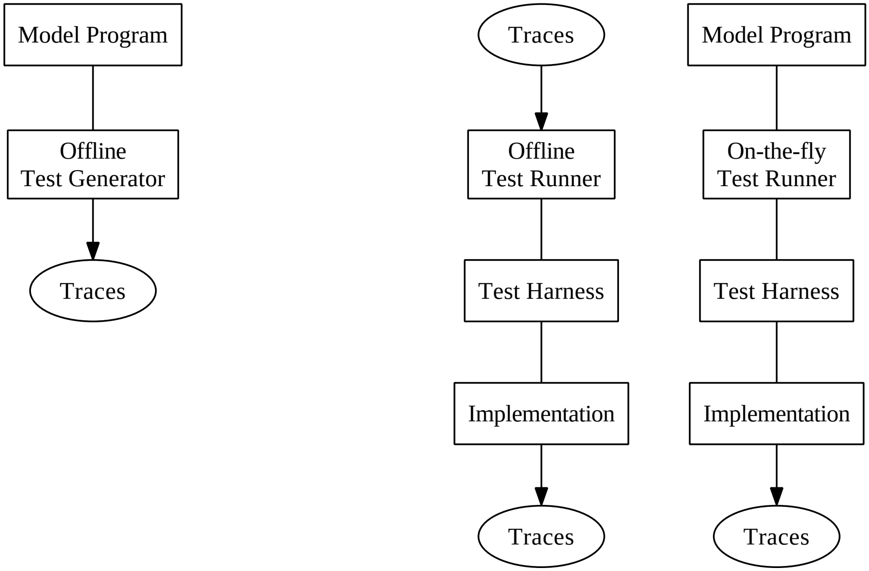
We chose *model-based testing*.

Traces

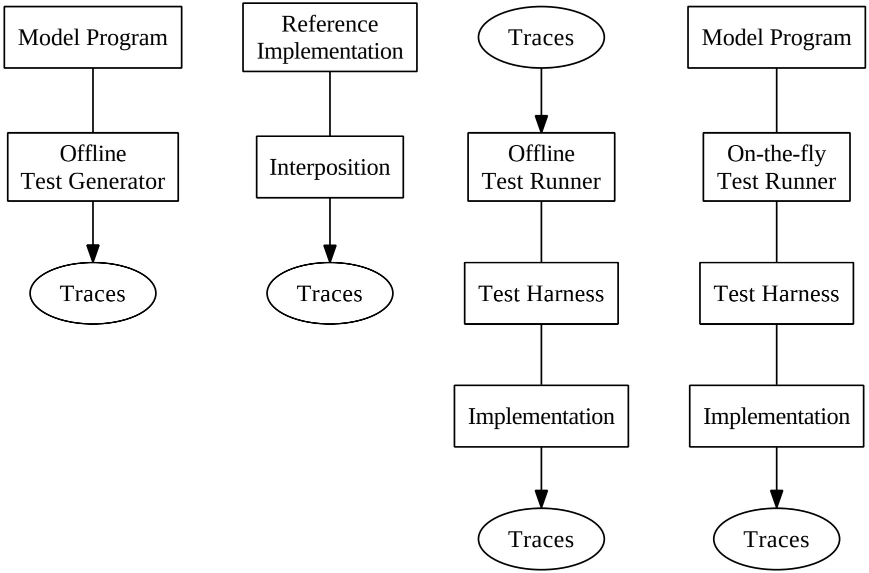
Traces are samples of behavior:

```
(listfiles_start, ()),
(listfiles_finish, (["junk.testfile"],)),
(removefile_start, ("junk.testfile",)),
(removefile_finish, ()),
(openfile_start, ("junk.testfile", True)),
(openfile_finish, (fileobject0)),
(filewriteat_start, (fileobject0, "hello world!!!", 0)),
(filewriteat_finish, ()),
(filewriteat_start, (fileobject0, "ked", 9)),
(filewriteat_finish, ()),
(filereadat_start, (fileobject0, None, 0)),
(filereadat_finish, ("hello worked!!")),
(fileclose_start, (fileobject0)),
(fileclose_finish, ())
```

Model-based testing



Model-based testing with trace capture / replay



Experience

Technology demonstration:

- File system API for next Seattle version
- Capture traces (via interposition) on Windows, Mac, Linux
- Captured traces while executing unit tests for file system API
- Replay traces on Windows, Mac, Linux
- Discovered, fixed several portability problems involving filenames

Future work

- Test with traces captured “in the wild”
- Test entire Seattle API
- Compare to offline testing, on-the-fly testing
- Consider testing other systems with portability requirements

Seattle distributed computing testbed
`https://seattle.cs.washington.edu/`

PyModel model-based testing framework
`http://staff.washington.edu/jon/pymodel/www/`