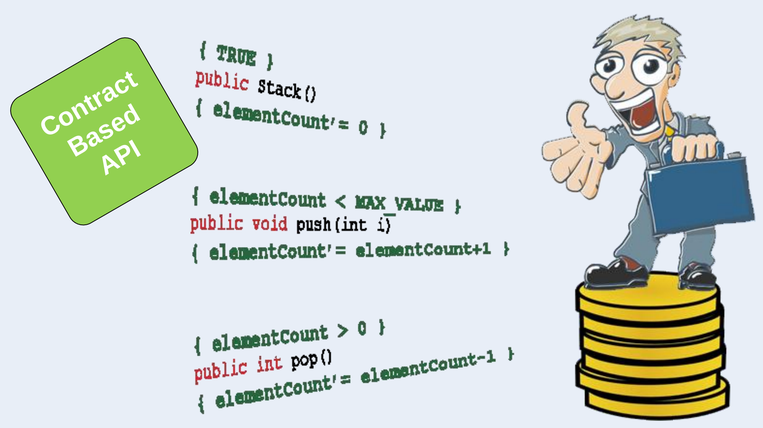
API Testing Using Contract Abstractions

Hernán Czemerinski – hczemeri@dc.uba.ar
Departamento de Computación, FCEyN, Universidad de Buenos Aires

Why Testing Matters?

Goal: To Find Bugs

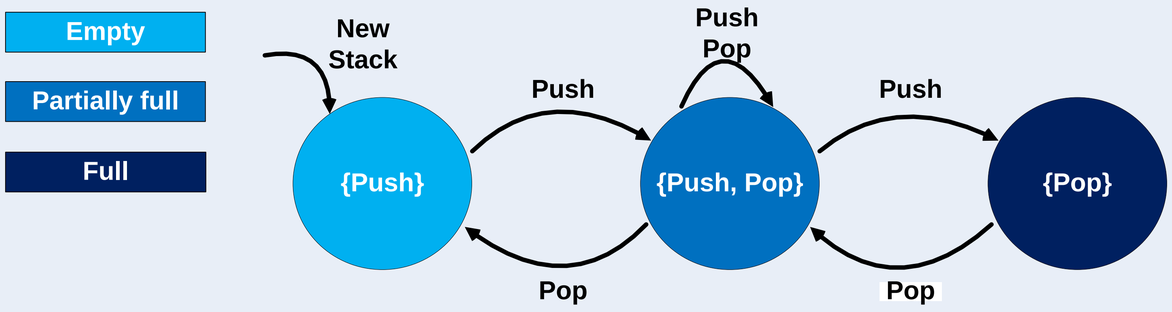
Contract-Based Software



Enabledness Models

API specifications are often given in terms of contracts:
- ✓ they say a lot about what each function does, but...
- ✗ don't say much about how it should be used as a whole; i.e. its protocol

For having such understanding we use enabledness models



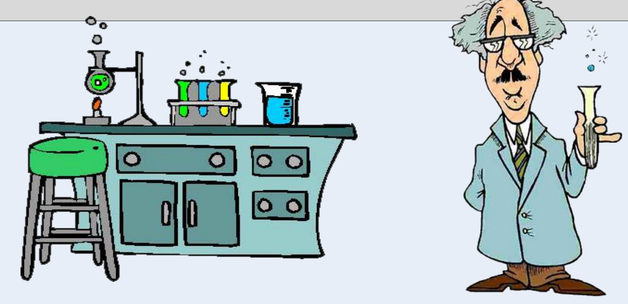
- each state of the model represents a particular set of enabled/disabled actions
- a transition from state *A* to state *B* labeled with *c* means that when the available actions of the API are those that *A* represents, after the execution of *c* the new set of enabled actions could eventually be those that *B* represents
- we build the enabledness model by using the Contractor tool, which takes a contract-based API specification as input and produces the model as output

Hypothesis

Test suites with *higher coverage of the enabledness model* are likely to:
- ✓ **H1**: be more effective at catching faults
- ✓ **H2:** obtain a higher code coverage

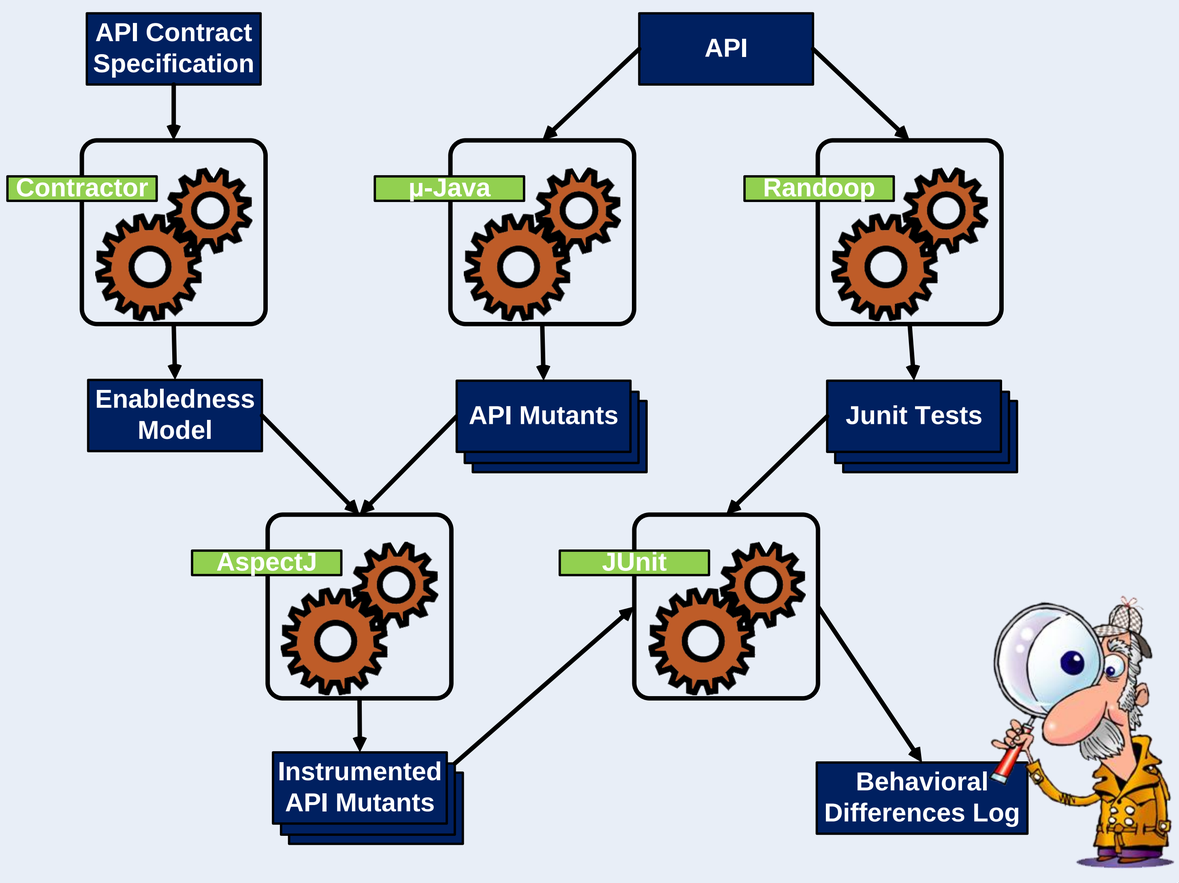
But… what do we mean by *higher coverage of the enabledness model*?

We consider two types:
- state coverage: it is important to reach each state, but not necessarily how it is reached; could be satisfied without executing every action
- transition coverage: it is important to execute every action everywhere it is possible; it implies the execution of all actions



Experimental Setup

In order to obtain experimental data for contrasting our hypothesis we choose Java APIs and:
- generate enabledness models from a contract-based specification
- generate randomly JUnit Tests using Randoop tool for the API
- generate mutants of the API by executing the Mu-Java framework
- instrument API mutants for logging how they exercise the model and the percentage of code coverage they achive
- execute unit tests on mutated versions of the API and detect behavioral differences found with respect to the oracle (API original version)



Done and Next

What's been done:

We've been working on the enabledness model of the following Java APIs
- ResultSet interface: we've used the HyperSQL database implementation
- Java Email Sever: an SMTP server
- Java Digital Signature: from the standard JDK 1.4 implementation

At **test suite level**, we've found that the number of covered transitions is highly correlated with both, the capability of finding bugs and the code coverage achieved. In contrast, the correlation given by covered states was low, which suggests that it is not a promising criterion. We report the Spearman' correlation rank for each of the case studies for the transition coverage criteria:

coefficient	JDBC	SMTP server	Signature
bug finding	0.77	0.73	0.36
code coverage	0.78	0.48	0.35

What's next:

We are currently running the experiment on the Java Socket and Java List Iterator of the JDK 1.4. implementation. Next steps would be:
- analyze gathered data for mentioned case studies
- build a Test Case Generation Tool that uses the information of the enabledness model for guiding the generation of unit test

So far, the experimental results have provided good evidence that for guiding the design of tests, the coverage of transitions of the enabledness model criterion would be very effective for achieving a high percentage of code coverage and for detecting faults of non-trivial protocol software.