TESTING WEB SERVICES WITH WEBDRIVER AND QUICKCHECK

Alex Gerdes

(joint work with Thomas Arts, John Hughes,
Hans Svensson and Ulf Norell)

QuviQ AB

QuviQ

PROWESS

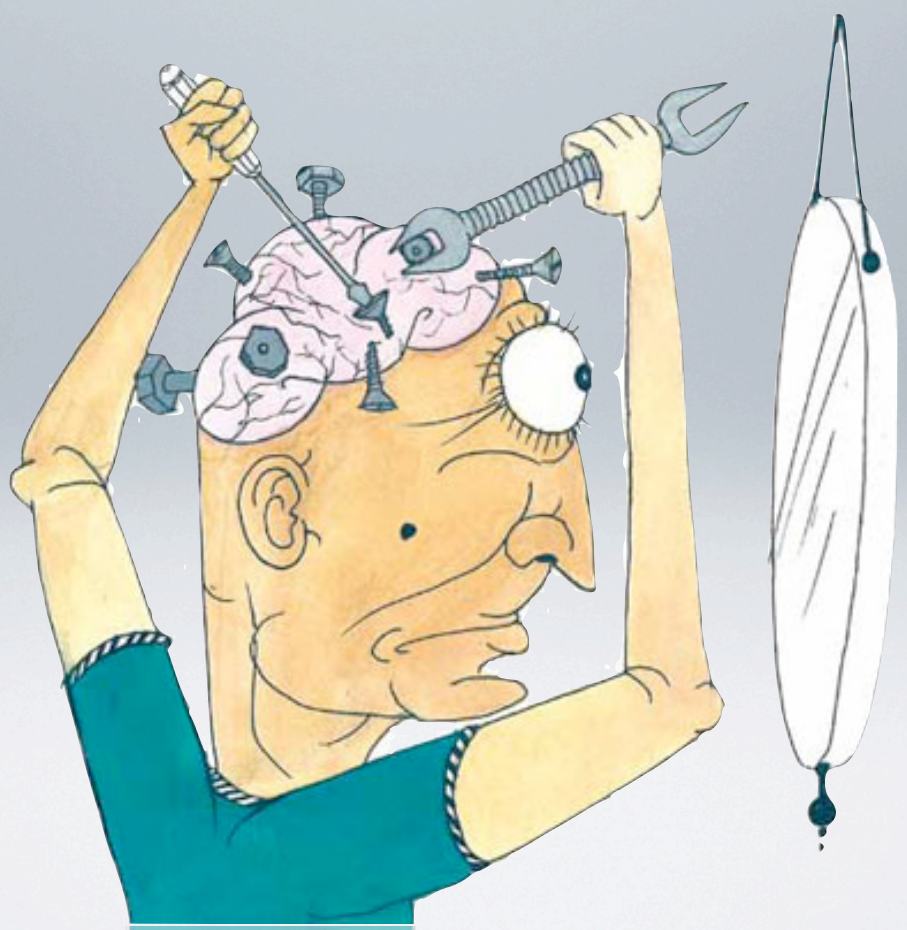
ABOUT ME

- From the Netherlands, now living in Sweden
- Worked at Ericsson and ASTRON
- Did a PhD study: developed a FP tutor for Haskell

- Started last year at QuviQ to work on/with QuickCheck

Q...

CHANGE YOUR MIND ABOUT TESTING!



QUICKCHECK

- Originally developed by John Hughes and Koen Claessen
- Property based testing
- Controlled randomness
- Shrinking
- QuickCheck libraries in many languages

QUVIQ QUICKCHECK

- Written in Erlang
- Many extensions:
  - Statem: testing using finite state machines
  - Pulse: finding race conditions
  - Symbolic test case generation
  - Mocking (C and Erlang)
  - Testing C-code
  - Composable components

EXAMPLE

```
% reverse a list
rev(Xs) ->
  lists:foldl(fun(X, Acc) -> [X|Acc] end, Xs, []).



% Properties
prop_rev() ->
  ?FORALL(Xs, list(int()), rev(rev(Xs)) == Xs).

prop_model() ->
  ?FORALL(Xs, list(int()),
          eqc:equals(rev(Xs), lists:reverse(Xs))).

prop_append() ->
  ?FORALL({Xs, Ys}, {list(int()), list(int())},
          rev(Xs ++ Ys) == rev(Ys) ++ rev(Xs)).
```

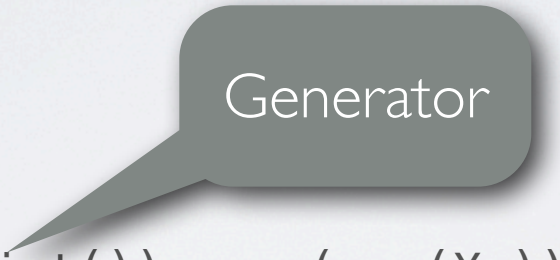
EXAMPLE

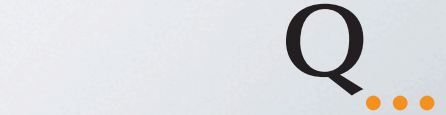
```
% reverse a list
rev(Xs) ->
  lists:foldl(fun(X, Acc) -> [X|Acc] end, Xs, []).


% Properties
prop_rev() ->
  ?FORALL(Xs, list(int()), rev(rev(Xs)) == Xs).

prop_model() ->
  ?FORALL(Xs, list(int()),
          eqc:equals(rev(Xs), lists:reverse(Xs))).

prop_append() ->
  ?FORALL({Xs, Ys}, {list(int()), list(int())},
          rev(Xs ++ Ys) == rev(Ys) ++ rev(Xs)).
```





EXAMPLE

```
% reverse a list
rev(Xs) ->
  lists:foldl(fun(X, Acc) -> [X|Acc] end, Xs, []).


% Properties
prop_rev() ->
  ?FORALL(Xs, list(int()), rev(rev(Xs)) == Xs).

prop_model() ->
  ?FORALL(Xs, list(int()),
          eqc:equals(rev(Xs), lists:reverse(Xs))).

prop_append() ->
  ?FORALL({Xs, Ys}, {list(int()), list(int())},
          rev(Xs ++ Ys) == rev(Ys) ++ rev(Xs)).
```

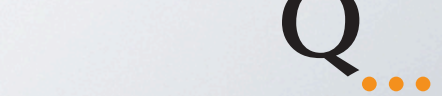
Generator

Properties

Demo

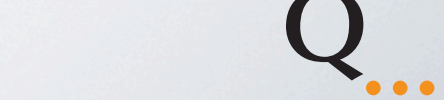
TESTING WEB SERVICES

- Move from desktop to web applications/services
- Often created using agile development processes
- Many different environments
- Testing of web services lags behind
- Use property based testing!



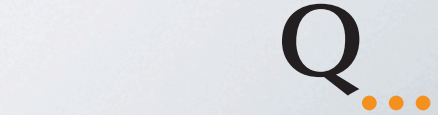
SELENIUM

"Selenium automates browsers. That's it. What you do with that power is entirely up to you. Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well."



SELENIUM

- Create and run scripts
- Supported by a wide range of browsers and operating systems:
  - Chrome, Internet Explorer, Opera, HtmlUnit and Firefox
  - Windows, Mac OS X, Linux, and Solaris
- Language support for: C#, Java, Python, Ruby, and partial support for Perl and PHP
- Widely used to create unit-tests and regression testing suites for web services



SELENIUM WEBDRIVER

- In version 2, Selenium introduced the WebDriver API
- Via WebDriver it is possible to drive the browser natively
- The browser can be local or remote – possible to use a grid test
- Languages implementing driver: C#, Java, Python, Ruby... and Erlang!

WEBDRIVER PROTOCOL

- What if your preferred language is not in the list?
- All WebDriver drivers communicates via HTTP using the WebDriver Wire Protocol
- It is a RESTful web service using JSON over HTTP
- The Wire Protocol consists of ~80 different commands controlling different aspects

WEBDRIVER IN ERLANG

- Implementation consists of:
  - webdrv_cap – handle web browser capabilities
  - webdrv_wire – purely functional implementation of the wire protocol
  - webdrv_session – wrapper module for WebDriver sessions
  - json – JSON library, written by Tony Garnock-Jones http://github.com/tonyg/erlang-rfc4627

WEBDRIVER COMMANDS

- Windows/Tabs – open, close, resize, set_position, maximize, ...
- Page elements – find, find relative, click, send_keys, submit, ...
- Navigation – back, forward, refresh, set_url, ...
- Timeouts – script timeout, page load timeout, ...
- Cookies – set, get, delete, ...

Demo

QUICKCHECK & WEBDRIVER

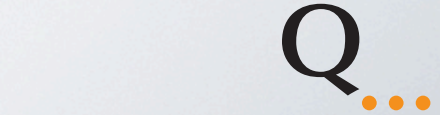
- Combine QuickCheck and WebDriver
- Insert data generators and state properties
- Use QuickCheck (finite) state machine to
  - store system state
  - generate valid sequences of selenium calls
- Rely on shrinking to find minimal counter example
- Play back a counter example

DUDLE

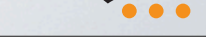
- Open source version of Doodle
- Polling for a time slot or opinion
- Multilingual
- Written in Ruby
- Clear and well defined interface

TESTING DUDLE

- Create polls
- Add and remove options
- Add and remove participants
- Vote for options

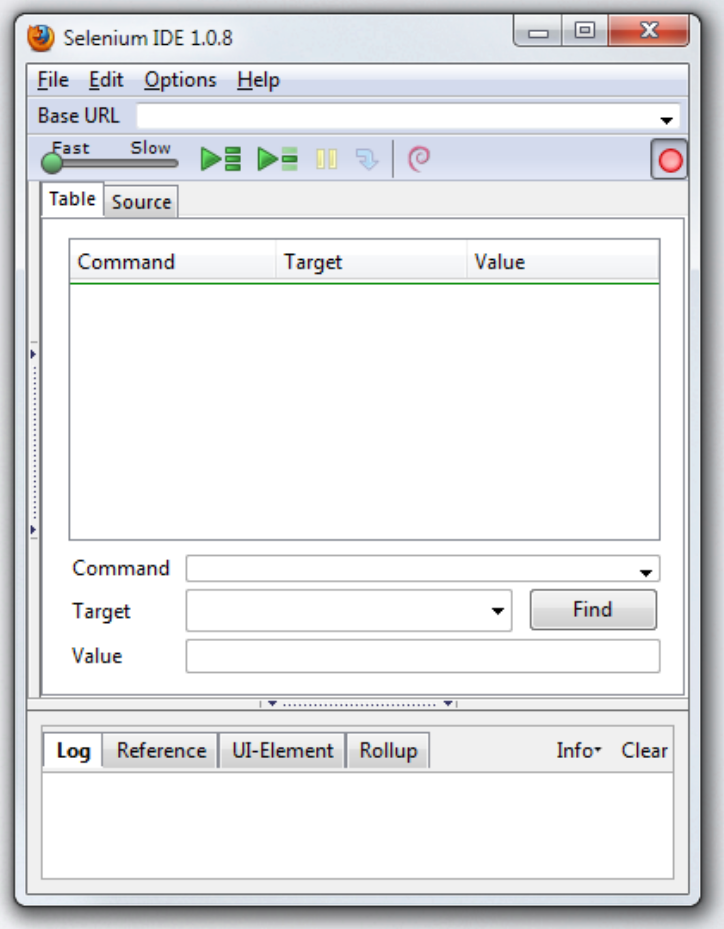


Lets test



SELENIUM IDE

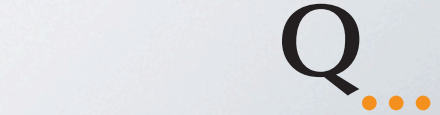
- A GUI tool to construct Selenium test cases
- Plugin for Firefox
- Record, store, and play back



Demo

PARSING SELENESE

- Find elements is tiresome
- Record test cases using Selenium IDE
- Translate those to webdrv in Erlang
- Adapt where necessary



EPILOGUE

- Available on GitHub: http://github.com/Quviq/webdrv
- Try it! If it breaks: fix it!
- Future work: extract QC finite state machines from EUnit tests
- Possible exercises:
  - Use webdrv on your (own) favourite website
  - Extend the dudle testing