# Yeager: An Annotation-based Framework for the Generation of Automated Long Sequence Regression Tests in Python

Casey Doran

Florida Institute of Technology

*cdoran2011@my.fit.edu*

November 27, 2017

# Overview

# Play Along at Home

- github.com/elementc/yeager
- github.com/elementc/monica-tests-traditional
- github.com/elementc/monica-tests-yeagerized
- github.com/elementc/thesis
- github.com/monicahq/monica

# Why Automate?

- Save time.
- Save money.
- Computers don't get bored.
  - Testing is boring work.
- Doesn't overlook test cases.

# How Do You Automate?

- Write functions that exercise the system under test.
- Put these functions in a format that can be consumed by a test runner.
- Call test runner.
- Interpret test runner's output.

# Kinds of Test Automation

- Unit- Verify individual code functions work as expected.
- Integration- Verify modules are working together.
- Regression- Verify stuff that worked isn't broken and stuff that's broken before haven't broken again.
- Functional- Validate that system as a whole conforms to requirements (eg, works to a user's eye).
- Many others- see CSE3411 & CSE4415.

# Languages

- *Unit frameworks (CUnit, JUnit, CppUnit, etc.) enable the practice, though they're useful for far more than unit testing.
- Testers are usually, unintuitively, **less** trained as programmers.
- Consequently, they prefer "easier" scripting languages like Python or Ruby.
- This discussion will center around Python. All of it can happen in Ruby.

# Frameworks

- Include a suite of assertion convenience methods, logging/reporting facilities, and a runner.
- Python: `unittest`, `nose`, `pytest`.
- `unittest` is in the Python Standard Library, we'll use it.

# Glass Box Testing

- Test code interacts directly with the program's source.
- Can probe quite deeply.
- Use mock interfaces shims to accomplish testing goals.
- Unit testing and integration testing are automated through Glass Box methods.

# Black Box Testing

- Test code interacts with the user (or some other non-transparent) interface into the running program.
- Use external toolkits like Selenium to enable driving user interfaces.
- Usually in a special test environment but otherwise the unmodified software.
- Regression testing and functional testing are automated through Black Box methods.

# Selenium

- Programmatic control of web browsers for testing and other automation.
- Driver class allows navigation (get this URL) and document queries (get this node for me to read from or click on or type into).
- Node class allows interaction (click here, type this), and data retrieval (What's the text body? What' site does this form POST to?) and limited Driver-like queries for children.

# HTML (summary)

- XML- based documents for the web.
- Tree-structured.
- Nodes have properties, including text, in addition to children.

# CSS (summary)

- Language for styling HTML documents.
- Format- `selector: rule;`
- Selectors: strings that identify one, many, or none of the nodes in an HTML document.
- Rules: Specific styling rules to apply to each node matched by preceeding rule.

# Monica: A Personal CRM

- Open-Source.
- Life-tracker.
- Friend-keeper.
- Journal.
- In the cloud.

# Contacts Book On Steroids

- We've all seen a contact list as an example in a database course.
- Monica extends that to the extreme. Per-contact notebooks, relationship tracking (How many wives did he have?), reminders, etc.
- Screenshots TBD

# Page Object Modeling

- Each page on a site corresponds to a Python class.
-

# How Test Suites Come Together

- TBD

# Running Tests

- TBD

# What Traditional Testing Finds

- TBD

# What Traditional Testing Doesn't Find

- TBD

# How To Find What Traditional Testing Doesn't Find

- TBD

# Examples of The Bugs We Want To Find

- TBD

# Software Is A State Machine

- TBD

# Testers Write Based On The System's States

- TBD

# State Models Can Help Us Plan New Tests

- TBD

# Context: What State Models Don't Capture

- TBD

# Random Walks: Generating New Test Plans Automatically

- TBD

# What Bugs Look Like From A Modeling Perspective

- TBD

# Prior Art: Model Based Testing

- TBD

# Weaknesses in PyModel

- TBD

# What Is Yeager?

- TBD

# Yeager's API Fits On A Notecard

- `import yeager`
- `@yeager.state_transition()`
- `yeager.walk()`
- Tweak: `yeager.add_state_to_blacklist()`, `yeager.add_transition_to_blacklist()`, and removal versions of each.
- Debug: `yeager.enumerate_transitions()`, `yeager.reachable_states()`, `yeager.orphaned_states()`

# Write a Function

- TBD

# Annotate the State Transition

- TBD

# Repeat

- TBD

# Debug Your Models

- TBD

# Plan a Test Run

- TBD

# Run It

- TBD

# Let's Test Monica

- TBD

# Monica's Intuitive States

- TBD

# States Necessitate Transitions

- TBD

# Boy, These Look Familiar

- Emulates the Page Object Models.

# Write Some Glue and Go

- TBD

# Example Suite's Model

- TBD

# Give It A Run!

- TBD

# What It Looks Like When Everything's Good

- TBD

# What It Looks Like When The Model Is Wrong

- TBD

# What It Looks Like When The Software Is Wrong

- TBD

# What Is High Volume Test Automation

- TBD

# Generators

- TBD

# Interface

- TBD

# Oracle

- TBD

# Loggers and Diagnostics

- TBD

# Context

- TBD

# Scalability

- TBD

# Inventors One Through Three

- TBD

# Inventors Four Through Six (And Beyond)

- TBD

# Everybody Thinks It's A Trade Secret

- TBD

# A Call For Academic Consideration

- TBD

# Family Tree: Exploring (And Abusing) The Anatomy

- TBD

# Long Sequence Regression Testing

- TBD

# State Model Testing

- TBD

# Exhastive Testing

- TBD

# Fuzz Testing

- TBD

# Load Testing

- TBD

# Testing In Production

- TBD

# A/B Testing

- TBD

# Model-Based LSRT

- TBD

# Quick To Implement

- TBD

# Good Enough Detail

- TBD

# Benefit From Existing Test Code

- TBD