The Test Case as Executable Example

CWI Lectures – Understanding Software
June 14, 2012

Arie van Deursen
@avandeursen

TU Delft
20th International Conference on Program Comprehension

Study of
the human activity of comprehending software;
the processes and technologies for supporting it
Programming by Example

Daniel Conrad Halbert
Ph.D. in Computer Science
Department of Electrical Engineering and Computer Sciences
Computer Science Division
University of California, Berkeley
November, 1984
Sponsor: Xerox Corporation
Susan L. Graham, Committee Chair

Abstract: Programming by example is a way of programming a software system in its own user interface. The user of the system writes a program by giving an example of what the program should do. The system records the sequence of actions, and can perform it again. Programming by example allows a user to create programs without doing conventional programming.

Programming by example was incorporated into a simulation of an office information system. As the system evolved, it became clear that the basic concept of programming by example needed to be extended: certain aspects of program creation are better done by program modification than by using the language's own mechanisms. This third extension includes a method for implementing many of the mechanisms that are used in the conventional language's built-in mechanisms. The third extension includes a macro mechanism that is independent of the example mechanism.

Spreadsheet Data Manipulation Using Examples

Sumit Gulwani
Microsoft Research
Redmond, WA, USA
sumitg@microsoft.com

William R. Harris
Univ. of Wisconsin
Madison, WI, USA
wharris@cs.wisc.edu

Rishabh Singh
MIT CSL
Cambridge, MA, USA
rishabh@csail.mit.edu

ABSTRACT

Millions of computer end-users need to perform tasks over large spreadsheet data, yet lack the programming knowledge to do such tasks automatically. We present a programming by example methodology that allows end-users to automate such repetitive tasks. Our methodology involves designing a domain-specific language and developing a synthesis algorithm that can learn programs in that language from user-provided examples. We present instantiations of this methodology for particular domains of tasks: (a) syntactic transformations of strings using restricted forms of regular expressions, conditionals and loops, (b) semantic transformations on data, such as translating names or phone-numbers or dates from one format to another, cleaning data, or extracting data from several text files or web pages into a single document. Excel allows users to write macros using a rich built-in library of string and numerical functions, or to write arbitrary scripts in Visual Basic or .NET programming languages. However, since end-users are not proficient in programming, they find it too difficult to write detailed macros or scripts. Moreover, even skilled programmers might hesitate to write a script for a one-off repetitive task.

We performed an extensive case study of spreadsheet help forms and observed that string and table processing is a common class of problems end-users struggle with. Our system is designed to learn macros for such tasks.

1. OVERVIEW

In this section, we outline a general methodology that we have used for developing interactive synthesizers for end-user programming tasks, along with how a user can interact with the synthesizers. In the first step of our methodology, we identify a domain of useful tasks that end-users struggle with...
The Testing Perspective

- A test case is an executable example of system behavior
- Stakeholder communication needs examples
- Throughout the full development cycle
Feature: Log in and out
  As an administrator
  I want to restrict access to certain portions of my site
  In order to prevent users from changing the content

Scenario: Logging in
  Given I am not logged in as an administrator
  When I go to the administrative page
  And I fill in the fields
    | Username | admin |
    | Password | secret |
  And I press "Log in"
  Then I should be on the administrative page
  And I should see "Log out"

Scenario: Logging out ...
Plugin / JUnit Testing in Eclipse.
1000s of tests
## Test Suite *Information Needs*

<table>
<thead>
<tr>
<th>ID</th>
<th>Need</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>Understanding test (source) code</td>
<td>$P_{11,17,18}$</td>
</tr>
<tr>
<td>N2</td>
<td>Understanding the reason (requirements) for a test</td>
<td>$P_{3,7,11,12,19}$</td>
</tr>
<tr>
<td>N3</td>
<td>Identifying what is tested by a test, test plug-in and (assembled) test suites</td>
<td>$P_{8,11,12,14}$</td>
</tr>
<tr>
<td>N4</td>
<td>Identifying blank spots</td>
<td>$P_{3,7,10,14}$</td>
</tr>
<tr>
<td>N5</td>
<td>Getting an overview of test suites</td>
<td>$P_{7,10,14}$</td>
</tr>
<tr>
<td>N6</td>
<td>Understanding integration with other plug-ins</td>
<td>$P_{7,10}$</td>
</tr>
<tr>
<td>N7</td>
<td>Understanding test organization</td>
<td>$P_{8,10,12,13,18,19}$</td>
</tr>
<tr>
<td>N8</td>
<td>Locating test code</td>
<td>$P_{13,19}$</td>
</tr>
<tr>
<td>N9</td>
<td>Identifying what influences the test execution environment</td>
<td>$P_{6,8}$</td>
</tr>
</tbody>
</table>

*Test Confessions: A Study of Testing Practices for Plug-In Systems*

Michaela Greiler, Arie van Deursen, Margaret-Anne Storey, ICSE 2012
Challenges

• Set of examples is **incomplete**

• Set of examples is **too large**
  – Some grouping needed
  – Redundancy

• Set of examples is **disconnected**
  – Requirements traceability
  – Backward coverage

Test Similarity
Measuring Test Case Similarity to Support Test Suite Understanding
Michaela Greiler, Arie van Deursen, Andy Zaidman, TOOLS 2012
Measuring Test Case Similarity

Test-aware instrumentation → Test Traces → Shared word count → Tests ranked by similarity

JUnitTestAll
Runs: 73/73  Errors: 0

aspectj
Preliminary Evaluation

- 75 unit tests
- 14 acceptance tests
- Ground truth by hand

- 37 unit
- 6 end to end
- Ground truth from book

- Most similar match: #1 or #2 in ground truth
- Top 10: all (but one) explainable & useful
- Surprises: mostly point to issue in test suite.

- Work in progress:
  Analyzing Eclipse test suites.
Rethinking “Test Cases as Examples”

- Languages for writing examples
  - Different levels of abstraction / stakeholders
  - Executable
  - Analyzable
  - Connected to execution traces

- Modularization and grouping of examples
  - Different levels of abstraction / stakeholders
  - Requirements / design / code coverage
  - Multiple views
  - Connected via execution traces
In Conclusion

• A test case is an executable example
• Never enough, yet also too many

• Test case connection via trace similarities
• Language engineering opportunities ahead
Photo credits

• Passau: Andreas Brandl
  http://icpc12.sosy-lab.org/
• Programming by Example, picture JanHeering (CWI). Photo Eelco Visser.
  http://eelcovisser.org/post/91/example-driven-research
• The Testing Perspective: Photo “Reaching for Understanding”, Flickr
  http://www.flickr.com/photos/elizabethsalib/2741807789/
• Andromeda Galaxy
• Colliding galaxies
• Paul Klint, lecturing.
  http://www.frankwatching.com/archive/2009/05/18/het-nationaal-ict-event/