Automatic Workarounds: Exploiting the Intrinsic Redundancy of Software to Improve Reliability

Antonio Carzaniga, Alessandra Gorla, Nicolò Perino, Mauro Pezzè

Faculty of Informatics
University of Lugano
Switzerland

March 22, 2012
Scope of this work:

reliability $\neq$ correctness

reliability $\approx$ fault tolerance

“self-healing. . .”
reliability $\Rightarrow$ redundancy

... **some** redundancy is necessary.
Examples

- N-version programming [Avizenis’75]
- Recovery blocks [Randell’75]
Examples

- N-version programming [Avizenis’75]
- Recovery blocks [Randell’75]

- Many forms of specifications
  - invariants
  - assertions
  - pre/post-conditions
Examples

- N-version programming [Avizenis’75]
- Recovery blocks [Randell’75]

- Many forms of specifications
  - invariants
  - assertions
  - pre/post-conditions

- Data diversity [Ammann&Knight’88]
- Robust data structures [Taylor et al.’80]
- “Rejuvenation” [Garg et al.’96]
- Rx: “bugs as allergies” [Qin et al.’07]
- Micro-reboots [Candea et al.’03]
- …
Examples

- N-version programming [Avizenis’75]
- Recovery blocks [Randell’75]
- Many forms of specifications
  - invariants
  - assertions
  - pre/post-conditions
- Data diversity [Ammann&Knight’88]
- Robust data structures [Taylor et al.’80]
- “Rejuvenation” [Garg et al.’96]
- Rx: “bugs as allergies” [Qin et al.’07]
- Micro-reboots [Candea et al.’03]
- ...
Examples

- N-version programming [Avizenis’75]
- Recovery blocks [Randell’75]
- Many forms of specifications
  - invariants
  - assertions
  - pre/post-conditions
- Data diversity [Ammann&Knight’88]
- Robust data structures [Taylor et al.’80]
- “Rejuvenation” [Garg et al.’96]
- Rx: “bugs as allergies” [Qin et al.’07]
- Micro-reboots [Candea et al.’03]

deliberate redundancy

expensive and possibly ineffective!
Hypothesis:

Software is **intrinsically** redundant
Hypothesis:

Software is *intrinsically* redundant

...and this intrinsic redundancy can be used to deal with faults at runtime.
Hypothesis:

Software is **intrinsically** redundant

...and this intrinsic redundancy can be used to deal with faults at runtime.

*at practically no cost*
Prior Plausibility
Code clones

- pervasive even \textit{in binaries} [Sæbjørnsen et al.:ISSTA’09]
- including \textit{semantic clones} that are \textit{syntactically different} [Gabel et al.:ICSE’08, Jiang & Zu:ISSTA’09]
Prior Plausibility

- Code clones
  - pervasive even in binaries [Sæbjørnsen et al.:ISSTA’09]
  - including semantic clones that are syntactically different [Gabel et al.:ICSE’08, Jiang&Zu:ISSTA’09]

- Design for reusability
  - display functions in JQuery: fadeIn(), show(), fadeInTo(), animate()
  - mutually interchangeable methods in Java SWT: setLocation(Point) and setLocation(int x, int y), setSize(Point) and setSize(int), etc.
  - alternative operations in Java containers: add(Component comp), add(Component comp, int index), add(Component comp, Object constraints), add(Component comp, Object constraints, int index), remove(Component comp), remove(int index), removeAll(), etc.
Prior Plausibility (2)

- Performance optimization
  - in the Apache Ant library `StringUtils.endsWith()` reimplements `java.lang.String.endsWith()`; `CollectionUtils.frequency()` reimplements `java.util.Collection.frequency()`, `SelectorUtils.tokenizePathAsArray()` reimplements `tokenizePath()`, etc.
  - the GNU Standard C++ Library has two implementations of stable sort (insertion-sort used for small sequences, and merge-sort for the general case)
Performance optimization

- in the Apache Ant library `StringUtils.endsWith()` reimplements `java.lang.String.endsWith()`; `CollectionUtils.frequency()` reimplements `java.util.Collection.frequency()`, `SelectorUtils.tokenizePathAsArray()` reimplements `tokenizePath()`, etc.

- the GNU Standard C++ Library has two implementations of stable sort (insertion-sort used for small sequences, and merge-sort for the general case)

Backward compatibility

- 45 classes and 365 methods in the Java 6 standard library are *deprecated*, and they duplicate exactly or almost exactly the functionality of newer classes and methods
polyline.enableDrawing();

Example: Google Maps Issue n. 1305
Example: Google Maps Issue n. 1305

```java
v = polyline.deleteVertex(polyline.getVertexCount()-1);
polyline.insertVertex(polyline.getVertexCount()-1,v);
polyline.enableDrawing();
```
Do Workarounds Exist?
### Do Workarounds Exist?

Analysis of issues recorded in issue-tracking systems

<table>
<thead>
<tr>
<th>system</th>
<th>reported faults</th>
<th>selected “workaround”</th>
<th>confirmed workarounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>≈ 400</td>
<td>63</td>
<td>43</td>
</tr>
<tr>
<td>YouTube</td>
<td>21</td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

- 10% of reported faults in Google Maps admit to a workaround
  - conservative estimate

- 42% of reported faults in YouTube admit to a workaround
Do Automatic Workarounds Exist?
Do Automatic Workarounds Exist?

Which workarounds can be plausibly generated automatically?

<table>
<thead>
<tr>
<th>system</th>
<th>confirmed workarounds</th>
<th>confirmed automatic workarounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>43</td>
<td>14</td>
</tr>
<tr>
<td>YouTube</td>
<td>9</td>
<td>5</td>
</tr>
</tbody>
</table>

- 33% workarounds in Google Maps could be generated automatically.
- 55% of workarounds in YouTube could be generated automatically.
application state space
application state space
application state space
general idea

application state space
application state space
General Idea

application state space
application state space
General Idea

application state space
General Idea

application state space
General Idea

application state space
application state space
application state space
General Idea

application state space

failure detection
General Idea

- Application state space
- Checkpoint recovery
- Failure detection
General Idea

- Checkpoint recovery
- Failure detection
- Workaround

Application state space
An initial “easy” context...
Web Applications

An initial “easy” context...

- *failure detection*: the *user* can do that for us
An initial “easy” context...

- **failure detection**: the user can do that for us

- **checkpoint/recovery**: simply **reload the page**
  - applications are mostly stateless (on the client-side)
Web Applications

An initial “easy” context...

- **failure detection**: the **user** can do that for us

- **checkpoint/recovery**: simply **reload the page**
  - applications are mostly stateless (on the client-side)

- **workarounds**: **alternative sequences**
  - represented as **code-rewriting rules**

  **Example**:
  
  ```
  setTags($X,$Y); → setTags($X); appendTags($Y);
  ```
  
  - implemented as a proxy or as a browser extension
  
  - priority scheme, automatic oracle, ... [Carzaniga et al. FSE’10]
Does It Work?
<table>
<thead>
<tr>
<th>system</th>
<th>rewriting rules</th>
<th>issues considered</th>
<th></th>
<th></th>
<th></th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>known</td>
<td>WA</td>
<td>unknown</td>
<td>WA</td>
<td></td>
</tr>
<tr>
<td>Google Maps</td>
<td>39</td>
<td>14</td>
<td>24</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>YouTube</td>
<td>40</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jQuery</td>
<td>68</td>
<td>25</td>
<td>77</td>
<td>102</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Does It Work?

<table>
<thead>
<tr>
<th>system</th>
<th>rewriting rules</th>
<th>issues considered</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>known WA</td>
<td>unknown WA</td>
</tr>
<tr>
<td>Google Maps</td>
<td>39</td>
<td>14/14</td>
<td>15+9/24</td>
</tr>
<tr>
<td>YouTube</td>
<td>40</td>
<td>5/5</td>
<td>1/1</td>
</tr>
<tr>
<td>jQuery</td>
<td>68</td>
<td>25/25</td>
<td>42+35/77</td>
</tr>
</tbody>
</table>

*(found, not found)*
### Does It Work?

<table>
<thead>
<tr>
<th>system</th>
<th>rewriting rules</th>
<th>issues considered</th>
<th></th>
<th></th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>known WA</td>
<td>unknown WA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google Maps</td>
<td>39</td>
<td>14/14</td>
<td>15+9/24</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>YouTube</td>
<td>40</td>
<td>5/5</td>
<td>1/1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>jQuery</td>
<td>68</td>
<td>25/25</td>
<td>42+35/77</td>
<td>102</td>
<td></td>
</tr>
</tbody>
</table>

*(found, not found)*

- **Google Maps:** 76%
- **YouTube:** 100%
- **jQuery:** 65%
## Does It Work?

<table>
<thead>
<tr>
<th>system</th>
<th>rewriting rules</th>
<th>issues considered</th>
<th>known WA</th>
<th>unknown WA</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>39</td>
<td>14/14</td>
<td>15/24</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>YouTube</td>
<td>40</td>
<td>5/5</td>
<td>1/1</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>jQuery</td>
<td>68</td>
<td>25/25</td>
<td>42/77</td>
<td>102</td>
<td></td>
</tr>
</tbody>
</table>

(found, not found)

<table>
<thead>
<tr>
<th></th>
<th>max iterations</th>
<th>found</th>
<th>not found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps:</td>
<td>76%</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>YouTube:</td>
<td>100%</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>jQuery:</td>
<td>65%</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
## Does It Work?

<table>
<thead>
<tr>
<th>System</th>
<th>Rules</th>
<th>Known WA</th>
<th>Unknown WA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>39</td>
<td>14/14</td>
<td>15+9/24</td>
<td>38</td>
</tr>
<tr>
<td>YouTube</td>
<td>40</td>
<td>5/5</td>
<td>1/1</td>
<td>6</td>
</tr>
<tr>
<td>jQuery</td>
<td>68</td>
<td>25/25</td>
<td>42+35/77</td>
<td>102</td>
</tr>
</tbody>
</table>

(max iterations w/ oracle

<table>
<thead>
<tr>
<th>System</th>
<th>Found</th>
<th>Not Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>76%</td>
<td>2</td>
</tr>
<tr>
<td>YouTube</td>
<td>100%</td>
<td>1</td>
</tr>
<tr>
<td>jQuery</td>
<td>65%</td>
<td>2</td>
</tr>
</tbody>
</table>
Current and Future Work
Current and Future Work

- Supporting general-purpose applications

- *Measuring* software’s intrinsic redundancy

- Approximate redundancy: *almost-equivalent* sequences

- Dealing with multi-threaded applications...
Anything to do with... multiplicity computing?
Anything to do with . . .

**multiplicity computing?**

Yes!

Use available computing power to improve reliability by **exploiting software’s intrinsic redundancy**
right now it’s only a notion

but I think I can get money to make it into a concept

and later turn it into an idea

[Woody Allen ’77, “Annie Hall”]

exploiting intrinsic redundancy by design

extended contracts + infrastructure + runtime analysis

document potential

intrinsic redundancy

“orchestrate”

some development at runtime