

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

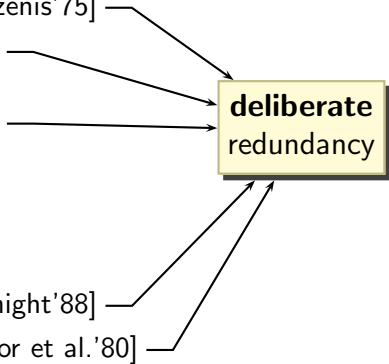
Examples

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

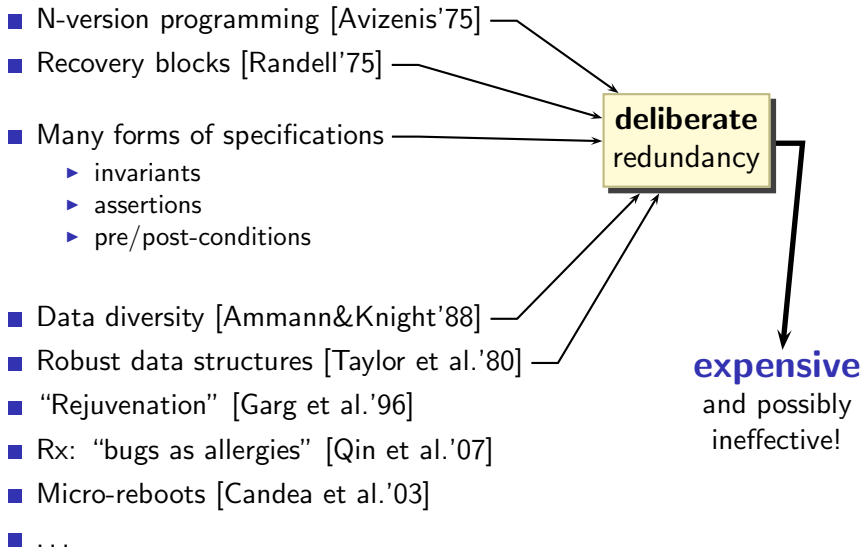
- Many forms of specifications
 - ▶ invariants
 - ▶ assertions
 - ▶ pre/post-conditions

- 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]
 - ...
- 
- The diagram features a central yellow box with a black border containing the text "deliberate redundancy". Eight arrows point towards this box from the following items in the list: "N-version programming [Avizenis'75]", "Recovery blocks [Randell'75]", "Many forms of specifications" (specifically from the "assertions" sub-item), "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]", and "Micro-reboots [Candea et al.'03]".

Examples



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

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]

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*: *fadeOut()*, *show()*, *fadeTo()*, *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)

Prior Plausibility (2)

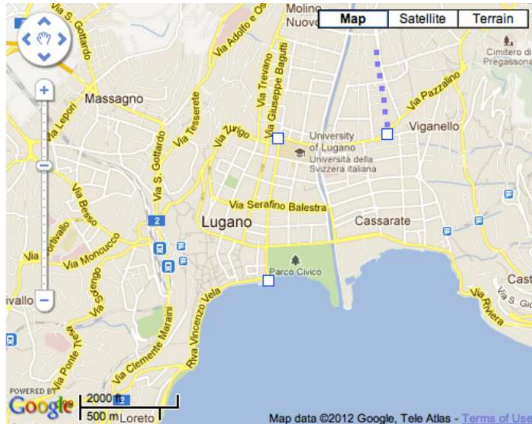
■ Performance optimization

- ▶ 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)

■ Backward compatibility

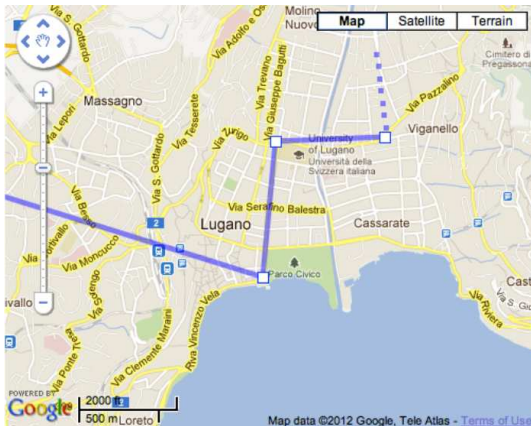
- ▶ 45 classes and 365 methods in the Java 6 standard library are *deprecated*, and they duplicate exactly or almost exactly the functionality of newer classes and methods

Example: Google Maps Issue n. 1305



```
polyline.enableDrawing();
```

Example: Google Maps Issue n. 1305



```
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

system	reported faults	selected "workaround"	confirmed workarounds
Google Maps	≈ 400	63	43
YouTube	21		9

- 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?

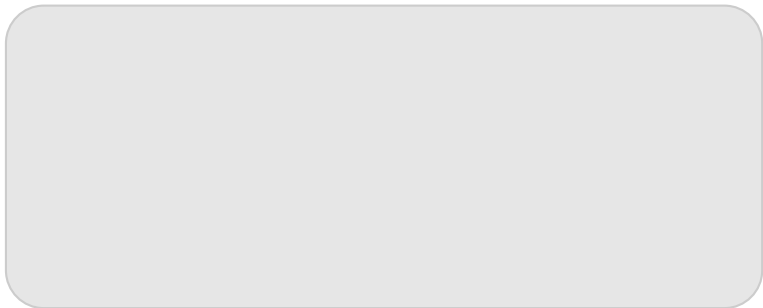
Which workarounds can be plausibly generated *automatically*?

system	confirmed workarounds	confirmed automatic workarounds
Google Maps	43	14
YouTube	9	5

- 33% workarounds in Google Maps could be generated automatically
- 55% of workarounds in YouTube could be generated automatically

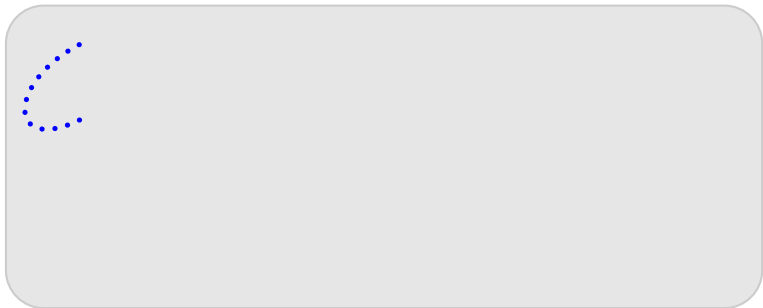
General Idea

application state space



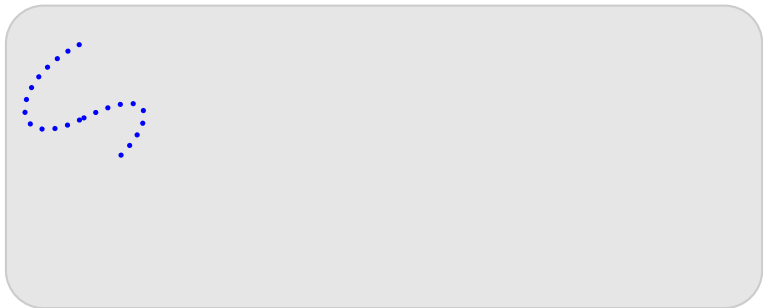
General Idea

application state space



General Idea

application state space



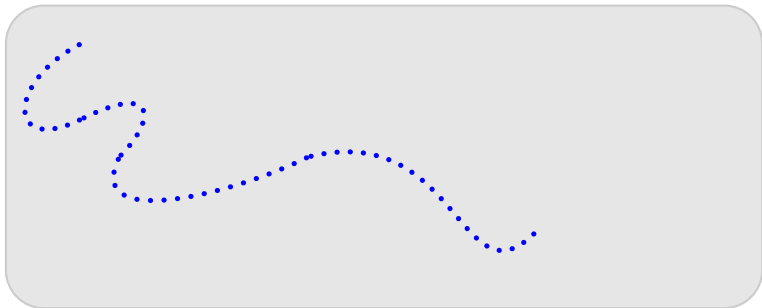
General Idea

application state space



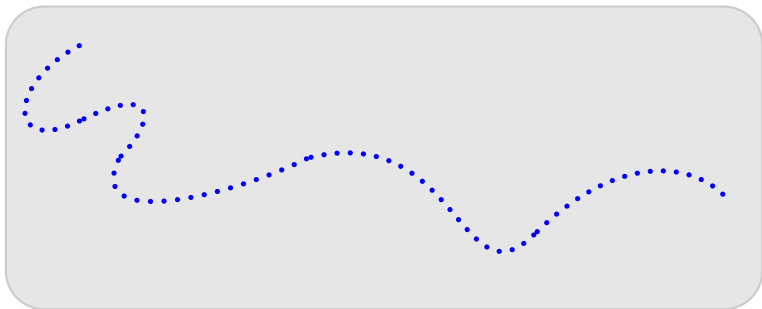
General Idea

application state space



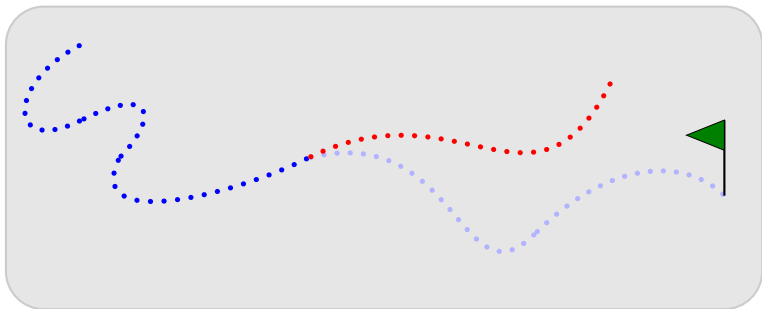
General Idea

application state space



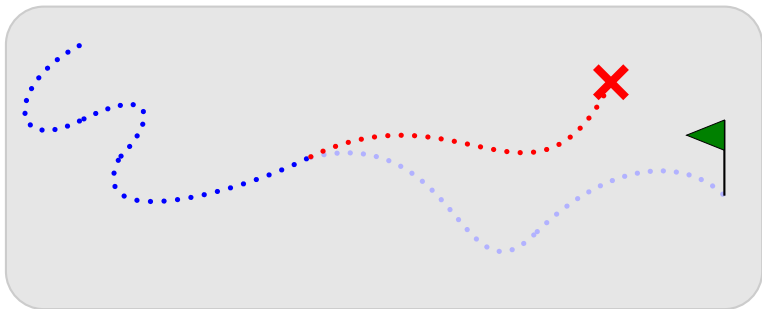
General Idea

application state space



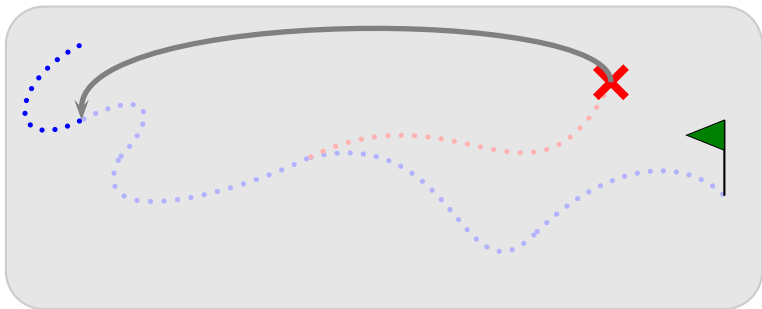
General Idea

application state space



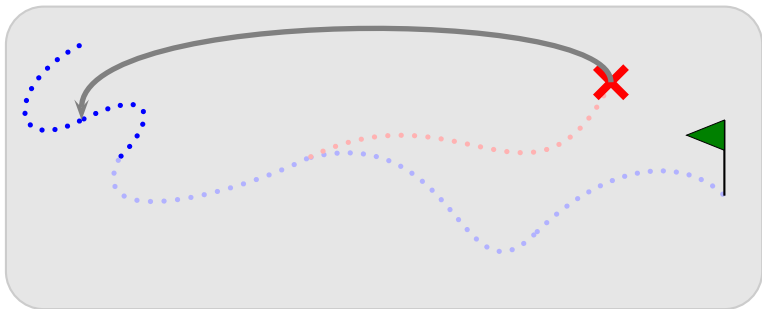
General Idea

application state space



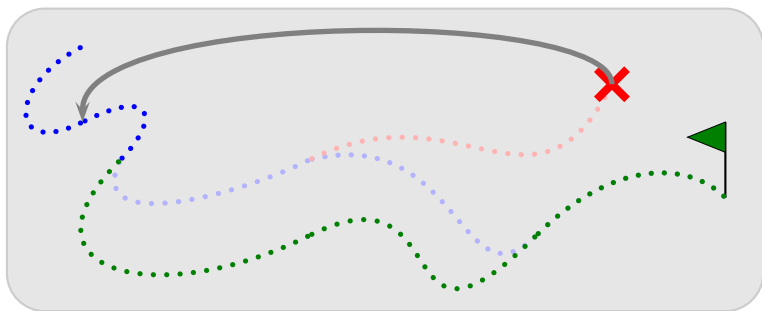
General Idea

application state space

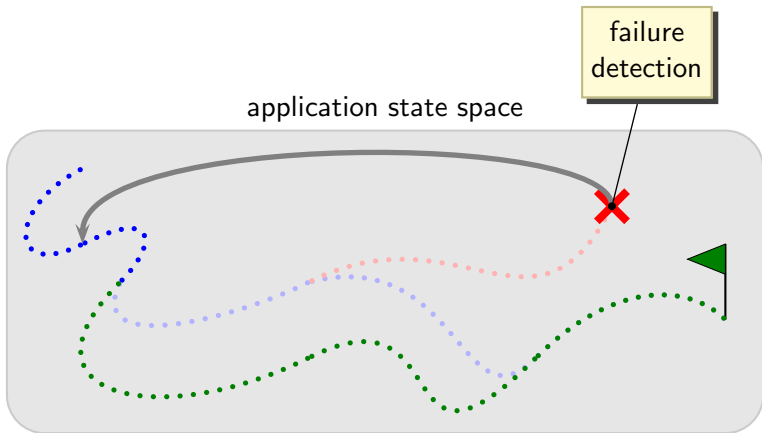


General Idea

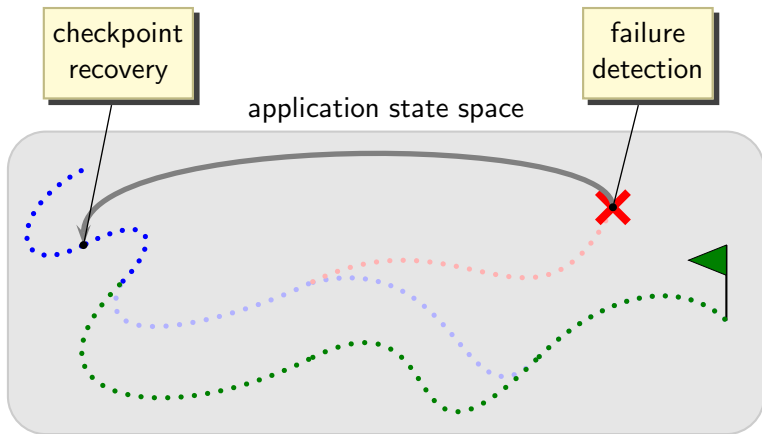
application state space



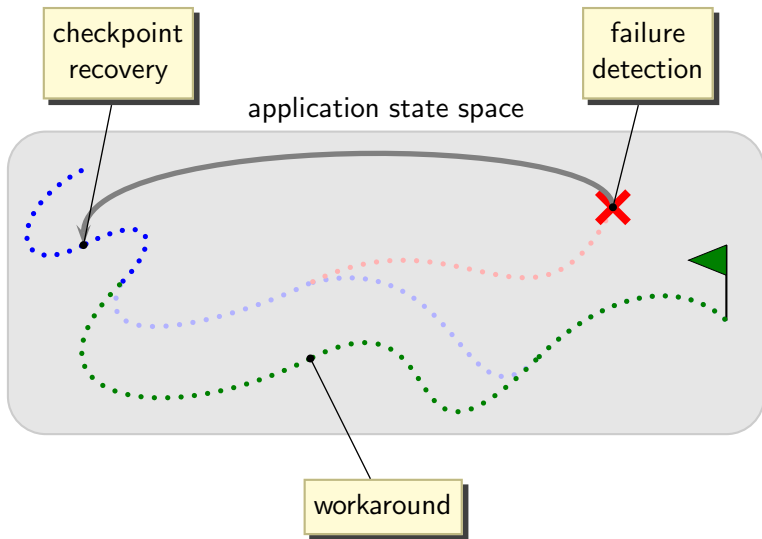
General Idea



General Idea



General Idea



Web Applications

An initial “easy” context...

Web Applications

An initial “easy” context...

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

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)

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?

Does It Work?

system	rewriting rules	issues considered		
		known WA	unknown WA	total
Google Maps	39	14	24	38
YouTube	40	5	1	6
jQuery	68	25	77	102

Does It Work?

system	rewriting rules	issues considered		
		known WA	unknown WA	total
Google Maps	39	14/14	15+9/24	38
YouTube	40	5/5	1/1	6
jQuery	68	25/25	42+35/77	102

(found, not found)

Does It Work?

system	rewriting rules	issues considered		
		known WA	unknown WA	total
Google Maps	39	14/14	15+9/24	38
YouTube	40	5/5	1/1	6
jQuery	68	25/25	42+35/77	102

(found, not found)

Google Maps:  76%

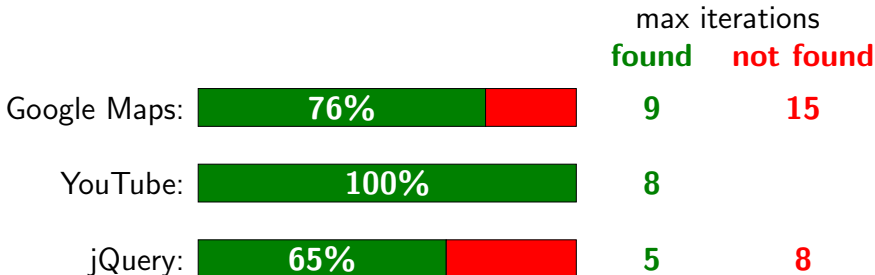
YouTube:  100%

jQuery:  65%

Does It Work?

system	rewriting rules	issues considered		
		known WA	unknown WA	total
Google Maps	39	14/14	15+9/24	38
YouTube	40	5/5	1/1	6
jQuery	68	25/25	42+35/77	102

(found, not found)



Does It Work?

system	rewriting rules	issues considered		
		known WA	unknown WA	total
Google Maps	39	14/14	15+9/24	38
YouTube	40	5/5	1/1	6
jQuery	68	25/25	42+35/77	102

(found, not found)

max iterations w/ oracle

found not found

Google Maps:



2

3

YouTube:



1

jQuery:



2

2

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