

Production-Driven Patch Generation

Thomas Durieux¹, Youssef Hamadi², Martin Monperrus¹

May 25, 2017

¹INRIA & University of Lille, France ²Ecole Polytechnique, France

**Windows sends more than 100 million
error reports per day!**

Automatic Program Repair



Buggy Application



Repair Strategy



Oracle (e.g: Crash)

Offline - Automatic Program Repair



Buggy Program



GenProg,
Nopol, ...



Failure Oracle:
Failing Tests
Regression Oracle:
Passing Tests

Online - Automatic Program Repair



Running
Program



Repair Strategy



Failure Oracle: ?
Regression Oracle: ?

Outline

Itzal Concept

Itzal Architecture

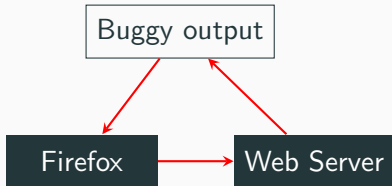
Itzal Prototype

Conclusion

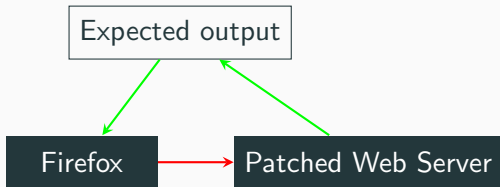
Itzal Concept



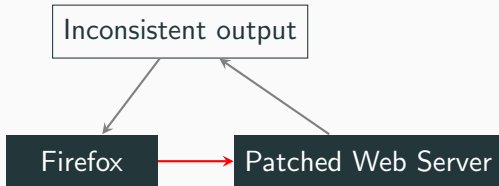
Itzal Concept



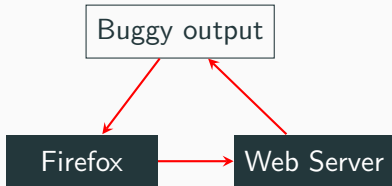
Itzal Concept



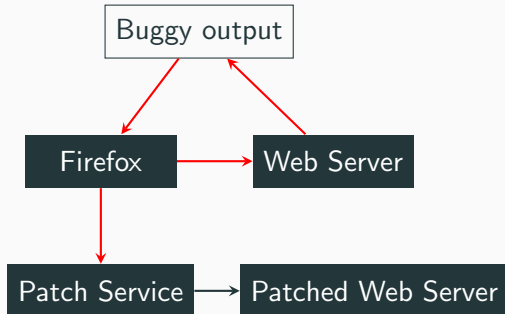
Itzal Concept



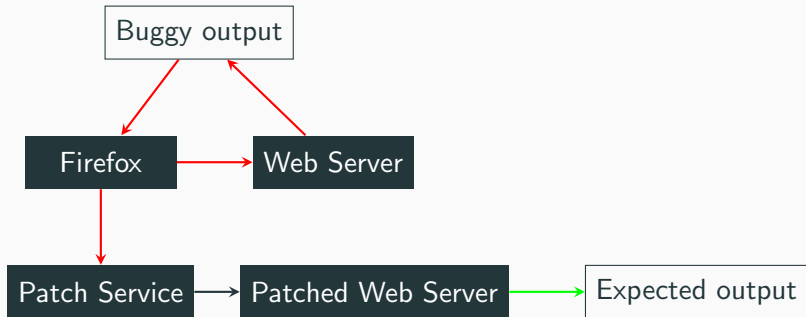
Itzal Concept



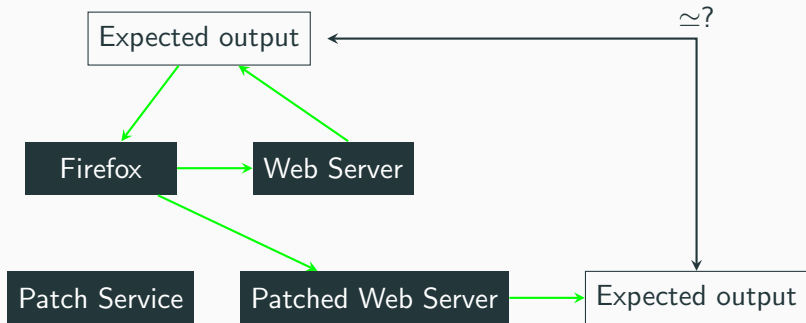
Itzal Concept



Itzal Concept



Itzal Concept



Highlights

- Patch generation in production.
- Patch regression with production inputs.
- Sandboxed repair environment repair the program without affecting the production.

Requirement

A request based application to reproduce the requests in a duplicated environment.

Outline

Itzal Concept

Itzal Architecture

Itzal Prototype

Conclusion

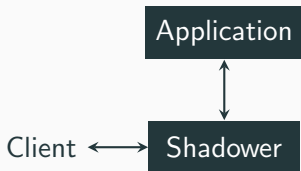
Itzal Architecture



Client: e.g. a browser

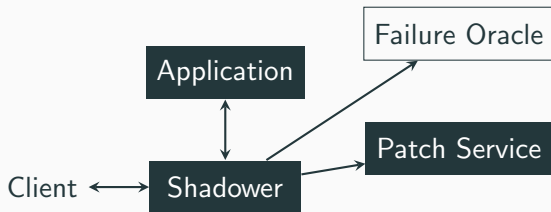
Application: e.g. a web server

Itzal Architecture



Shadower: intercepts and duplicates the requests

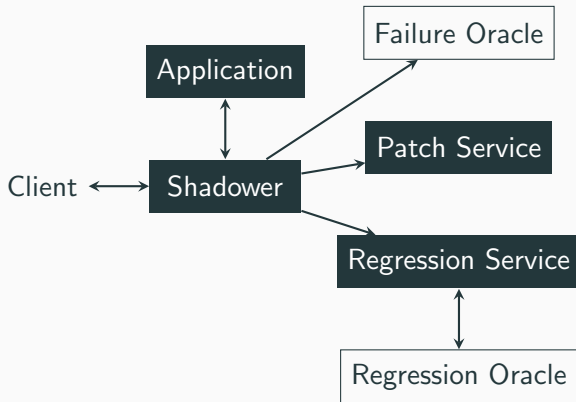
Itzal Architecture



Patch Service: generates patches that fix the requests

Failure Oracle: detects if a request is passing or failing

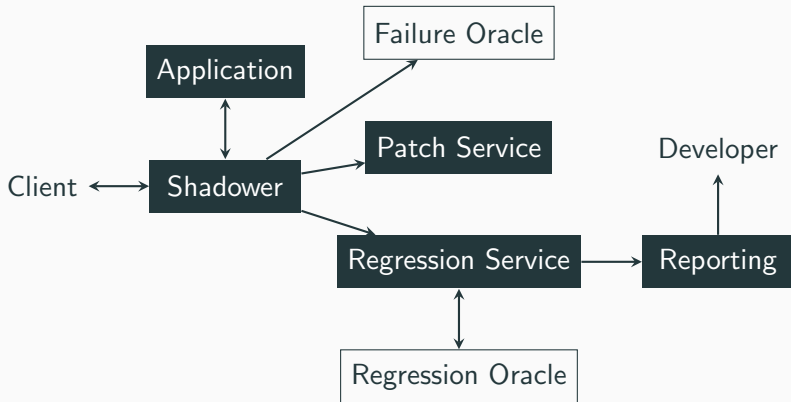
Itzal Architecture



Regression: executes passing request on patched applications

Regression Oracle: compares the output of the application and the patched application

Itzal Architecture



Reporting: communicates the patches to the developers
(Dashboard, Pull Request, ...)

- Failure Oracle: decides if a request is valid or not (e.g. $HTTP_{status} \neq 5xx$)
- Regression Oracle: decides if the patch does not modify the behavior for all non-failing requests.

Outline

Itzal Concept

Itzal Architecture

Itzal Prototype

Conclusion

The screenshot shows the homepage of Mayocat Shop. At the top left is the logo, a stylized cat with a top hat and mustache. To the right is a navigation menu with links for Discover, Community, Documentation, Showcase, and Contact. A GitHub link is also present. The main banner features a red background with the headline "Open source e-commerce and marketplaces made simple". Below this is a yellow box with a cartoon cat illustration and text describing the platform as a next-generation open source marketplace. To the right of the cat is a green button for downloading version 0.30.1, with links for release notes, installation instructions, and getting started. Below the banner is a section titled "State of the art technologies" with a sub-headline about a robust solution based on Java. It features a diagram showing a bidirectional flow between a "Back-office" (Java, PostgreSQL, ElasticSearch) and a "Front-office" (HTML5, Javascript, Mustache / Handlebars) via an "API". To the right of this diagram is a "Keep in touch" section with an email input field and an "OK" button, and a "Get involved" section with text about the open community and a "Read more" link.

Mayocat Shop

Discover Community Documentation Showcase Contact

We're on GitHub

Open source e-commerce and marketplaces made simple

Mayocat Shop is the next generation of **open source** marketplace and e-commerce platform. With Mayocat Shop, developers and businesses can leverage a powerful and innovative platform to build great global marketplace and individual e-commerce websites and provide online merchants with a state of the art selling experience.

[Download Mayocat Shop](#)
version 0.30.1

[Release notes](#)
[Installation instructions](#)
[Getting started with Mayocat Shop](#)

State of the art technologies

Take advantage of a robust solution based on Java, and make e-shops with only HTML knowledge.

Back-office

Front-office

Java

HTML5

API

PostgreSQL,
ElasticSearch,

Javascript,
Mustache / Handlebars

Keep in touch

Your e-mail

or follow us on [Twitter](#) or [GitHub](#)

Get involved


Mayocat Shop is an open and welcoming community. There are opportunities for contributing and learning in a number of different areas.

[Read more](#)

Itzal Prototype

NPE in cart when no price is defined in shipping strategy #231

New issue

 **Closed** jvelo opened this issue on Dec 9, 2014 · 0 comments



jvelo commented on Dec 9, 2014

Owner



No description provided.




jvelo added the **bug** label on Dec 9, 2014



jvelo self-assigned this on Dec 9, 2014



jvelo added a commit that closed this issue on May 29, 2015

 Fixes #231 NPE in cart when no price is defined in shipping strategy

ce04282



jvelo closed this in [ce04282](#) on May 29, 2015

Assignees

 jvelo

Labels

bug

Projects

None yet

Milestone

No milestone

Notifications

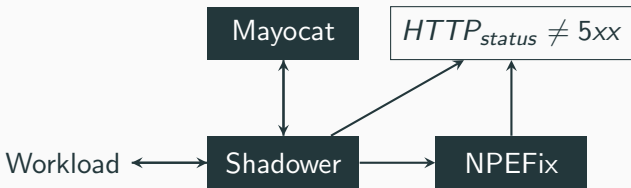
 **Subscribe**

You're not receiving notifications from

Generated Patch for Mayocat

```
@@ FlatStrategyPriceCalculator.java
@@ -37,2 +37,5 @@
+   if (carrier.getPerItem() == null) {
+       return null;
+   }
    price = price.add(carrier.getPerItem().
        multiply(BigDecimal.valueOf(
            numberOfItems)));
```

Itzal Prototype



Outline

Itzal Concept

Itzal Architecture

Itzal Prototype

Conclusion

Conclusion

Take Away

- Patch generation in production.
- Patch regression with production inputs.
- Sandboxed repair environment repair the program without affecting the production.

Future Work

- Evaluation on other bugs (extremely difficult)
- Repair other type of bugs

Summary

Online - Automatic Program Repair



Running
Program



Repair Strategy



Failure Oracle: ?
Regression Oracle: ?

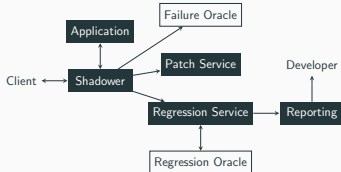
3

Generated Patch for Mayocat

```
@@ FlatStrategyPriceCalculator.java
@@ -37,2 +37,5 @@
+   if (carrier.getPerItem() == null) {
+       return null;
+   }
    price = price.add(carrier.getPerItem().
        multiply(BigDecimal.valueOf(
            numberOfItems)));
```

13

Itzal Architecture



Reporting: communicates the patches to the developers
(Dashboard, Pull Request, ...)

8

Conclusion

Take Away

- Patch generation in production.
- Patch regression with production inputs.
- Sandboxed repair environment repair the program without affecting the production.

Future Work

- Evaluation on other bugs (extremely difficult)
- Repair other type of bugs

16

Summary

