Antifragile Software and Genetic Improvement

Martin Monperrus
University of Lille & Inria, France

CREST Open Workshop on Genetic Improvement
Jan 2016


Stoke-on-Trent
Nottingham
Derby
Norwich
England
Wales
Ipswich
Swansea
Headington
Bristol
London
Leigh-on-Sea
Cardiff
Reading
Barnstaple
Southampton
Exeter
Portsmouth
Bournemouth
Plymouth
Falmouth
English Channel
Guernsey
Jersey
Le Havre
Caen
Rouen
Amiens
Cambrai
Saint-Quentin
Lannion
St-Malo
Brest
Saint-Brieuc
Rennes
Quimper
Le Mans
Orleans
Lorient
Vannes
Angers
Tours
Nantes
Saint-Nazaire
Cholet
La Roche-sur-Yon
Poitiers
Niort
Bourges
Dijon
Chalon-sur-Saône
Paris
Évry, Essonne
Troyes
Reims
Lille
Bruges
Antwerp
Mol
Brussels
Belgium
Luxembourg
Den Helder
Amsterdam
The Hague
Netherlands
Woensel-Noord
Nord
Cologne
Düsseldorf
Münster
Kernstadt
Bielefeld
Hanover
Brunswick
Germany
Frankfurt
Wiesbaden
Würzburg
Mannheim
Nuremberg
Saarbrücken
Karlsruhe
Stuttgart
Metz
Nancy
Ingolstadt
Ulm
Munich
Freiburg
Konstanz
Basel
Zurich
Innsbruck
Liechtenstein
Emmenbrücke
Switzerland
Crans-Montana
Geneva

```
try {
   prepare_meringue()
   add_cream()
   make_balls()
   add_chocolate_shavings()
}
catch (MissingChocolateEx e) {
  use_nutella();
}
```

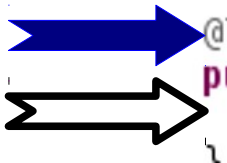
« Anticipated errors »

Expected errors in test suites

Blue and white tests specify anticipated errors
White tests specify resilience

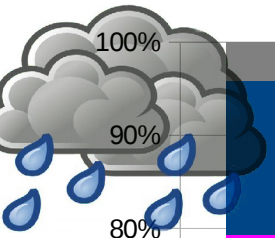
- Test colors
  - Pink
  - Blue
  - White

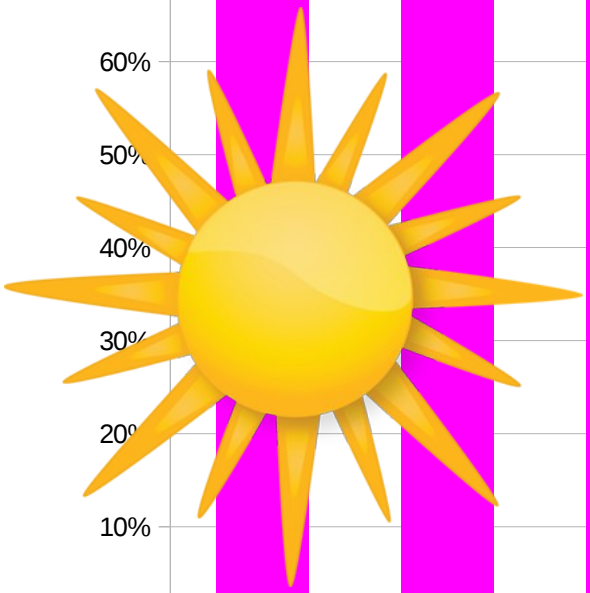
```
public String getProperty(String s) {
    String res = null;
    try{
        res = getPropertyFromFile(s);
    }catch(PropertyNotFoundException pnfe){
        return null;
    }
    return res;
}
```



```
@Test
public void testAbsentProperty(){
    assertNull(getProperty("@$%^ù*µ;?!"));
}
```

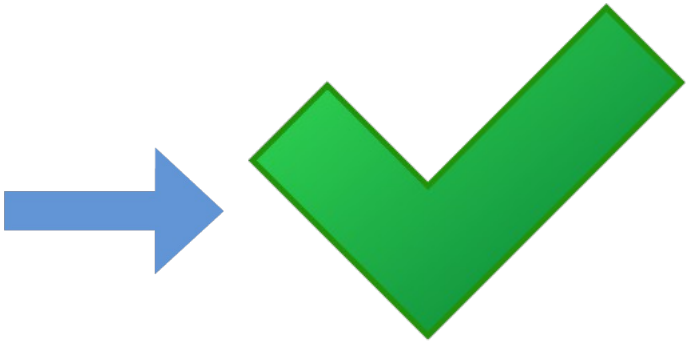
Empirical results

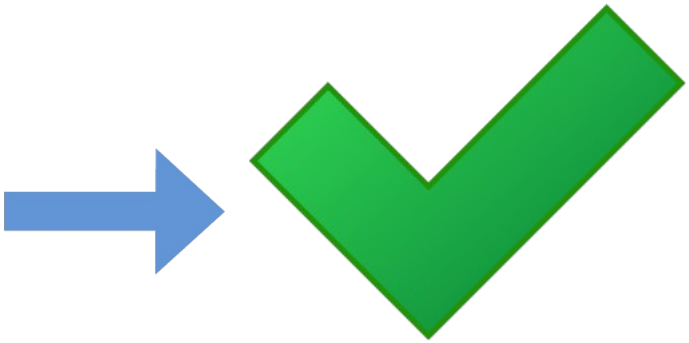




Our intuition: simulating unforeseen errors by perturbing test case execution

Why ?





Short circuit testing: Exception injection in test suites

White test

+ throw new X()

=

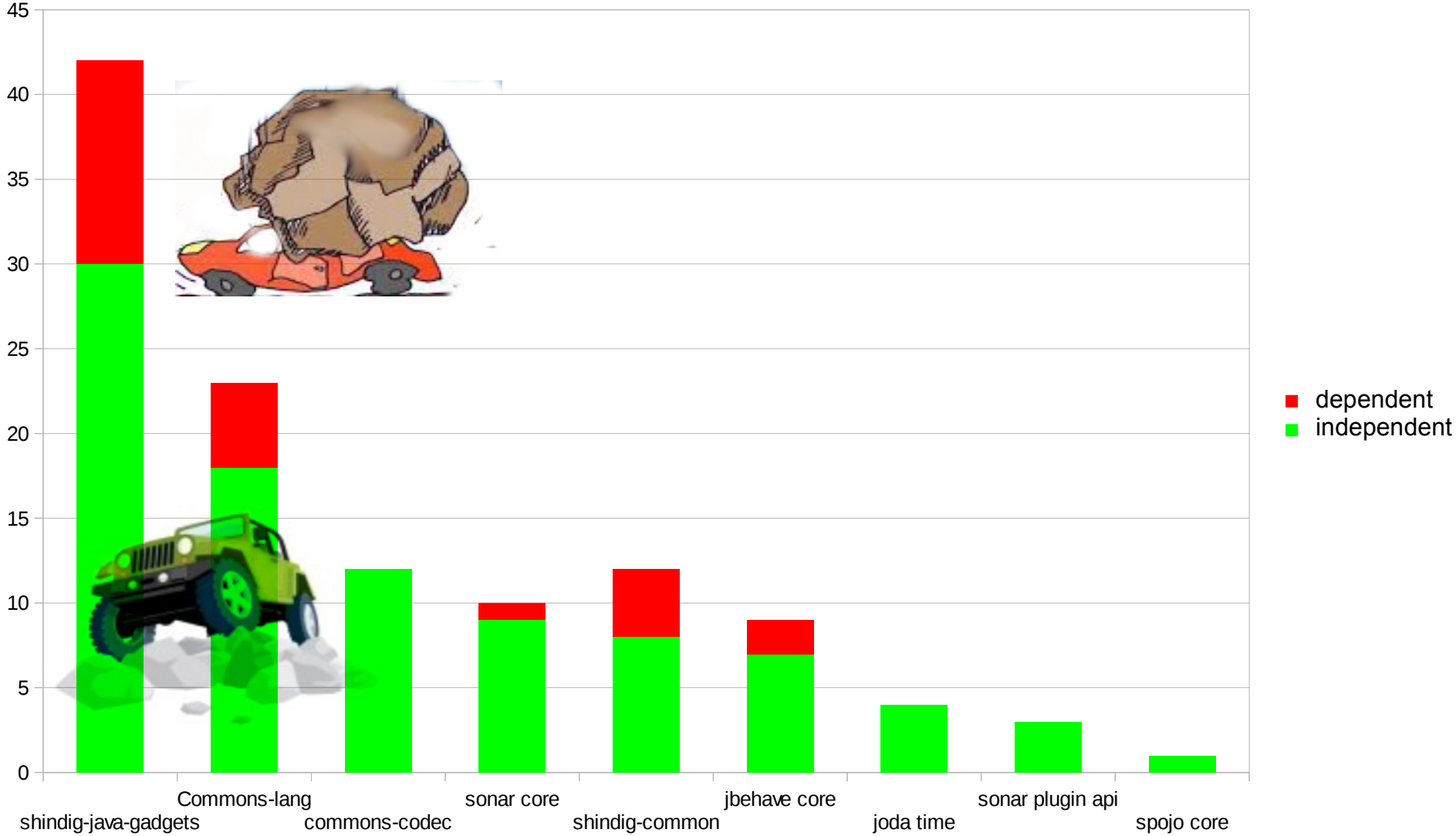
Source-Independence

The try-catch blocks that keep the tests passing under exception injection are capable to handle unanticipated errors.

We call them "source-independent".

Empirical assessment of resilience against unforeseen errors

Empirical Evaluation



Test driver

```
try{
   // instrumentation code
   if(Controller.isCurrentTryCatchWithInjection())
        throw new IOException();
   }

   ... //normal try body
...
} catch (IOException e ) {  // Genetic Improvement
... //normal catch body
}
```

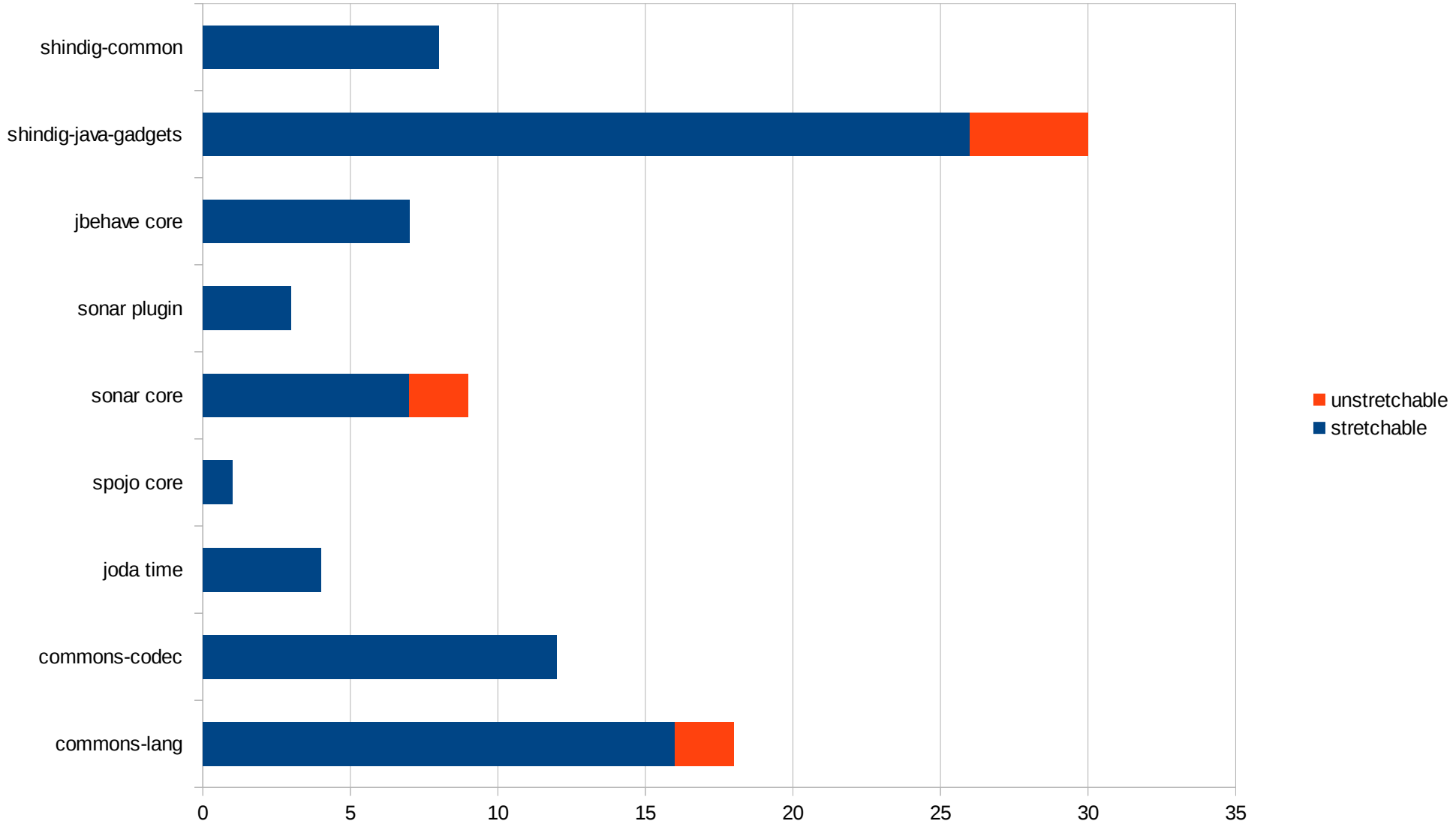
Limit cases

Stack of try-catch at runtime

```
// original
Catch (OufofMemory) {
  Catch (NullPointer) {
    Catch (IOException) {
      throw new NullPointerException()
    }
  }
}
```

```
// stretched
Catch (OufofMemory) {
  Catch (NullPointer) {
    Catch (Exception) {
      }
    }
  }
}
```

Empirical Evaluation



Take-away

"We inject exceptions during test suite execution to assess and improve resilience"

References:

Exception Handling Analysis and Transformation Using Fault Injection: Study of Resilience Against Unanticipated Exceptions (Benoit Cornu, Lionel Seinturier, Martin Monperrus), In Information and Software Technology, Elsevier, 2014.

Definition

A thing is **antifragile** if it becomes

better with stressors, shocks, volatility, noise, mistakes, faults, attacks, failures . . .

(Antifragile, N. N. Taleb, 2013)

A software system is **antifragile** if it becomes

better with failures, attacks, failures, misconfigurations, weird usages . . .

(Principles of Antifragile Software, M. Monperrus, 2014)

Short circuit testing is antifragile:

- Better with exception injection
- Does not try to anticipate or predict

Chaos Monkey

```
// Chaos Engineering
// principlesofchaos.org, 2015

hypothesis ← perturbation, measure
while (true) {
  perturb
  if (measure unacceptable) {
    report failure
  }
}
```

Antifragile software engineering

Core model:

- perturbation model
- perturbation controller
- perturbation cost
- perturbation gain

http://www.monperrus.net/martin/antifragile-software