

---

# Trading perfection for robustness in extraordinary software



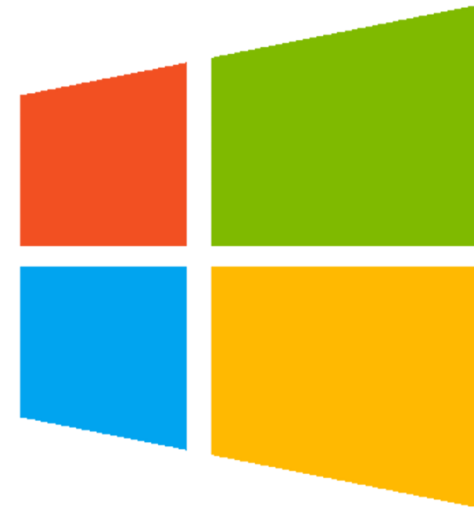
Benoit Baudry (EPI DiverSE)

Journées scientifiques 2015 – June, 19 2015.



# Extraordinary software

---



# Unstable environment

---

- Users
  - Customization, extensions, add-ons
- Malicious users
  - Complex attack surfaces, self-adaptive viruses, weird machines
- Network
  - Concurrent access, bandwidth, server crash, etc.
- Hardware
  - millions of devices, multi-core chips hard to predict, etc.
- Software environment
  - OS, other applications, updates, etc.

# Unstable environment

---

- Users
  - Customization, extensions, add-ons
- Malicious users
  - Complex attack surfaces, self-adaptive viruses, weird machines
- Network
  - Concurrent access, bandwidth, server crash, etc.
- Hardware
  - mill
- Software
  - OS

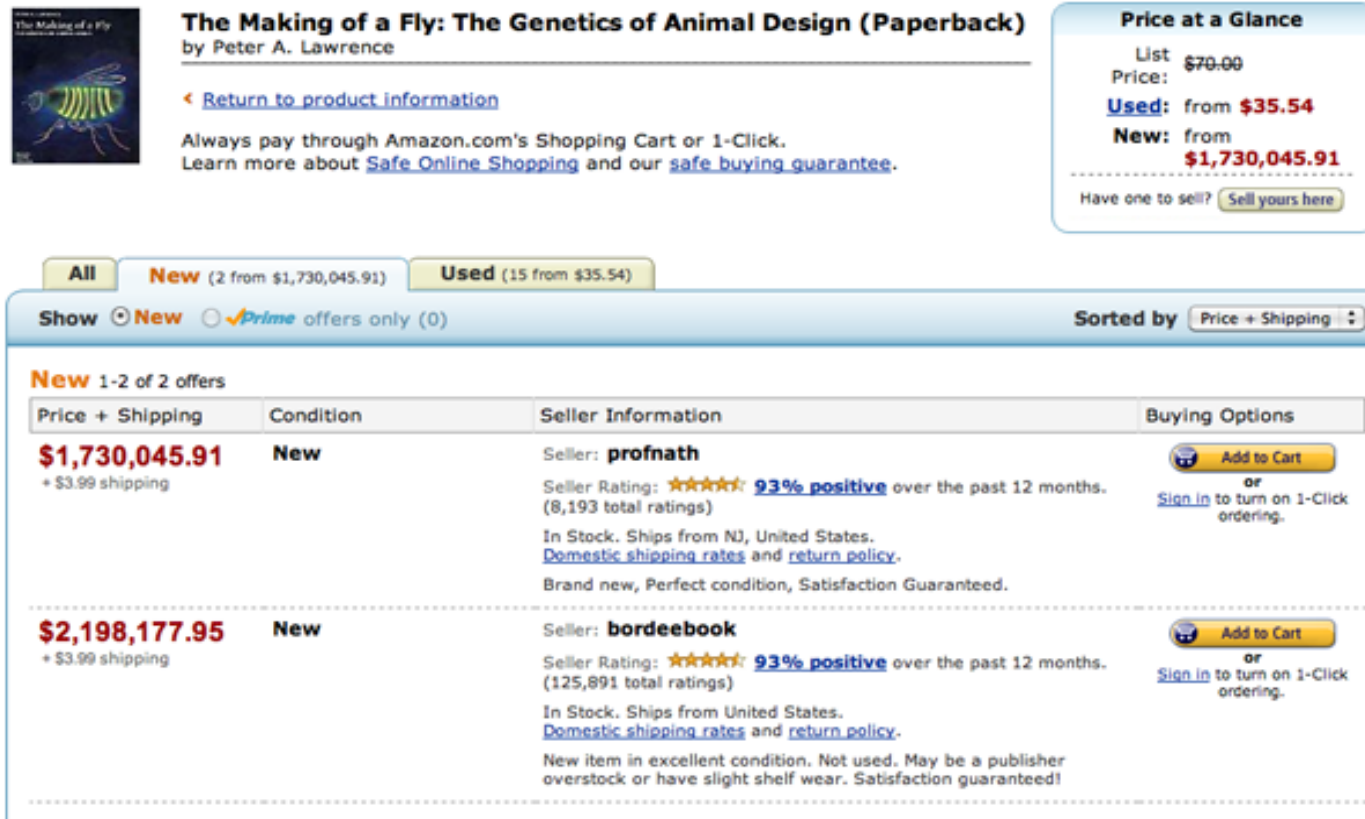
Extraordinary software cannot be perfect in a specific context, it must be acceptable in many contexts that cannot be predicted

# Trading perfection for robustness

---

- How to engineer robust systems
  - that are noisy?
  - that are prone to neutral variations?
  - that are extremely diverse?
  - that are not perfect: they must between different qualities

# Amazon's \$23,698,655.93 book about flies



**The Making of a Fly: The Genetics of Animal Design (Paperback)**  
by Peter A. Lawrence

[Return to product information](#)

Always pay through Amazon.com's Shopping Cart or 1-Click.  
Learn more about [Safe Online Shopping](#) and our [safe buying guarantee](#).

**Price at a Glance**  
List Price: ~~\$70.00~~  
**Used:** from **\$35.54**  
**New:** from **\$1,730,045.91**  
Have one to sell? [Sell yours here](#)

**All** **New** (2 from \$1,730,045.91) **Used** (15 from \$35.54)

Show ☒ New ☐ Prime offers only (0) Sorted by Price + Shipping

**New** 1-2 of 2 offers

Price + Shipping	Condition	Seller Information	Buying Options
<b>\$1,730,045.91</b> + \$3.99 shipping	<b>New</b>	Seller: <b>profnath</b> Seller Rating: ★★★★★ <b>93% positive</b> over the past 12 months. (8,193 total ratings) In Stock. Ships from NJ, United States. <a href="#">Domestic shipping rates</a> and <a href="#">return policy</a> . Brand new, Perfect condition, Satisfaction Guaranteed.	<a href="#">Add to Cart</a> or <a href="#">Sign in</a> to turn on 1-Click ordering.
<b>\$2,198,177.95</b> + \$3.99 shipping	<b>New</b>	Seller: <b>bordeebook</b> Seller Rating: ★★★★★ <b>93% positive</b> over the past 12 months. (125,891 total ratings) In Stock. Ships from United States. <a href="#">Domestic shipping rates</a> and <a href="#">return policy</a> . New item in excellent condition. Not used. May be a publisher overstock or have slight shelf wear. Satisfaction guaranteed!	<a href="#">Add to Cart</a> or <a href="#">Sign in</a> to turn on 1-Click ordering.

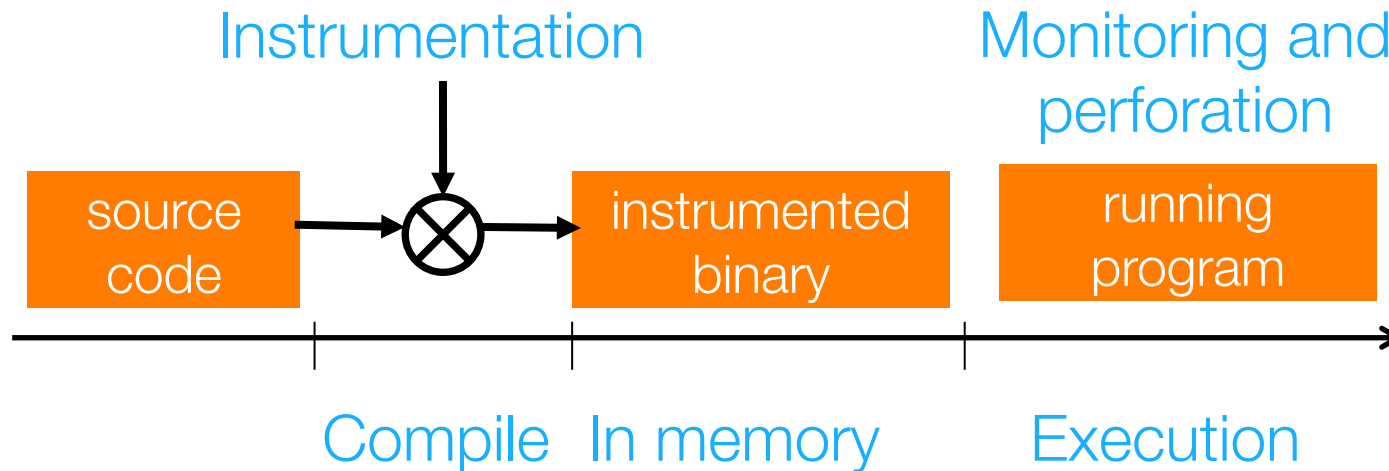
- Algorithmic pricing:
  - Once a day profnath set their price to be 0.9983 times bordeebook's price, then bordeebook "noticed" profnath's change and elevated their price to 1.270589 times profnath's higher price.

# Engineering robust software systems

---

- Obtaining and Reasoning About Good Enough Software
  - M. Rinard. 2012.
- Building Robust Systems. An essay.
  - G.J. Sussman. 2007.
- Self-healing: softening precision to avoid brittlenes
  - M. Shaw. 2002.
- Building Diverse Computer Systems.
  - S. Forrest, A. Somayaji, D. Ackley. 1997.
- Design of self-checking software
  - S. Yau and R. Cheung. 1975.

# Loop perforation



```
for (i = 0; i < n; i++) { ... }
```



```
for (i = 0; i < n; i += 2) { ... }
```



# Loop perforation

---

- Experiment on the PARSEC benchmark
  - video encoding / decoding
  - data mining
  - computer vision
  - monte-carlo simulation
- Some loops can be perforated with 1.5 speedup and minimal quality loss
- Approximate correctness reduces computation time

# Failure oblivious computing

---

- Keep the system running after an out-of-bound access
- When the program attempts to read an out of bounds array element or use an invalid pointer to read a memory location, the implementation manufactures a value
- Successfully prevent crash in the presence of well-known out-of bound errors
  - on 3 different email servers
- Acceptable overhead (due to bound checks)

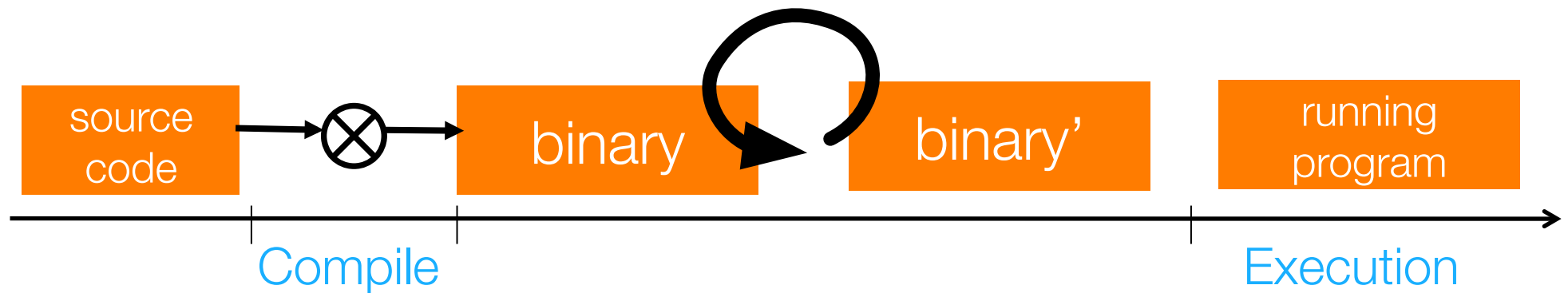
# Adapting binary code for a HW chip

---

- Energy consumption of hardware chips is very difficult to predict statically
  - Necessary energy is a complex interplay between the app code and the hardware architecture
  - Compilers cannot have generic strategies to optimize energy cost of binary code

# Adapting binary code for a HW chip

- Energy consumption of hardware chips is very difficult to predict statically
  - Necessary energy is a complex interplay between the app code and the hardware architecture
  - Compilers cannot have generic strategies to optimize energy cost of binary code



# Results

---

- PARSEC benchmark
- Runtime energy reduction
  - between 10% and 80%
  - most reductions on CPU-bound programs, rather than IO-bound
- Transformations impact
  - the structure of control flow
  - removal of unnecessary computation
  - branch prediction

# Approximate computation

---

- New hardware approximations
  - Voltage overscaling introduces errors in SRAM read/write in exchange of energy savings
  - Bit-width reduction reduces Mantissa bits in exchange of energy savings
- How can we write programs that exploit these approximations?

# Approximate computation

---

- EnerJ and FlexJava extend Java
- Identify what can be approximated
  - approximate data stored in the approximate sections of memory
  - approximate operations are computed in the approximate sections of the CPU

# Approximate computation

---

```
float computeLuminance (float r, float g,  
float b) {  
    float luminance = r * 0.3f + g * 0.6f  
+ b * 0.1f;  
  
    loosen(luminance);  
  
    return luminance;  
}
```



# Approximate computation

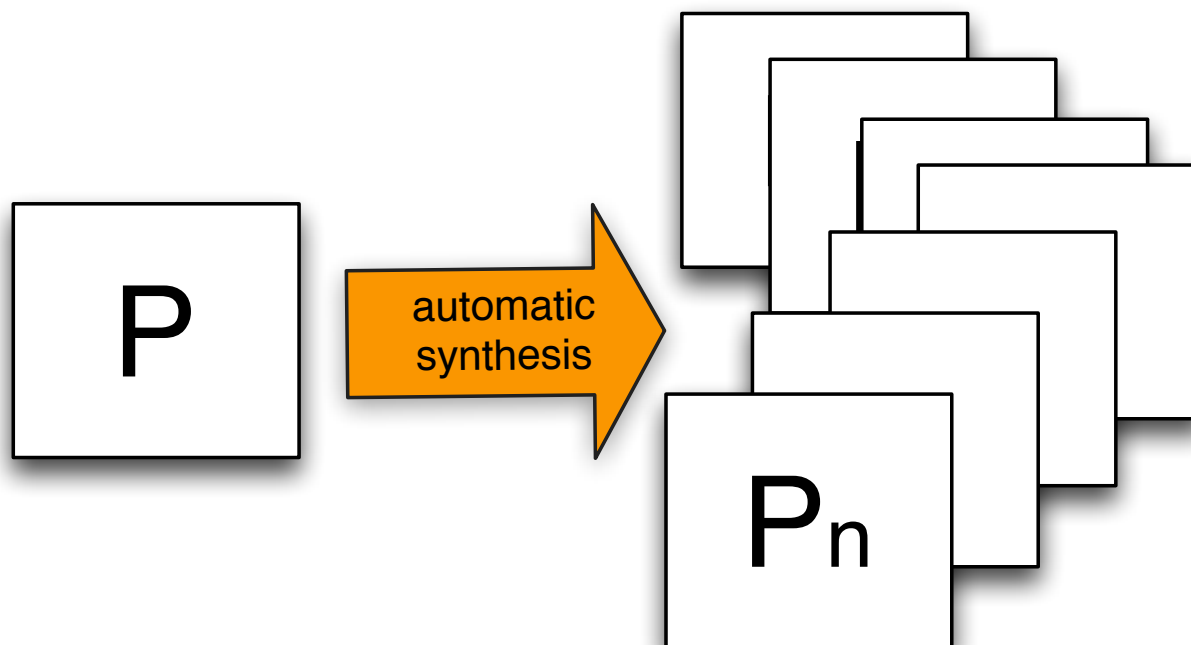
---

- Evaluation
  - programs that tolerate approximate outcomes
  - data mining, image recognition, image encoding
- Between 10 and 40% energy savings for tolerable accuracy loss

# Application-level software diversity

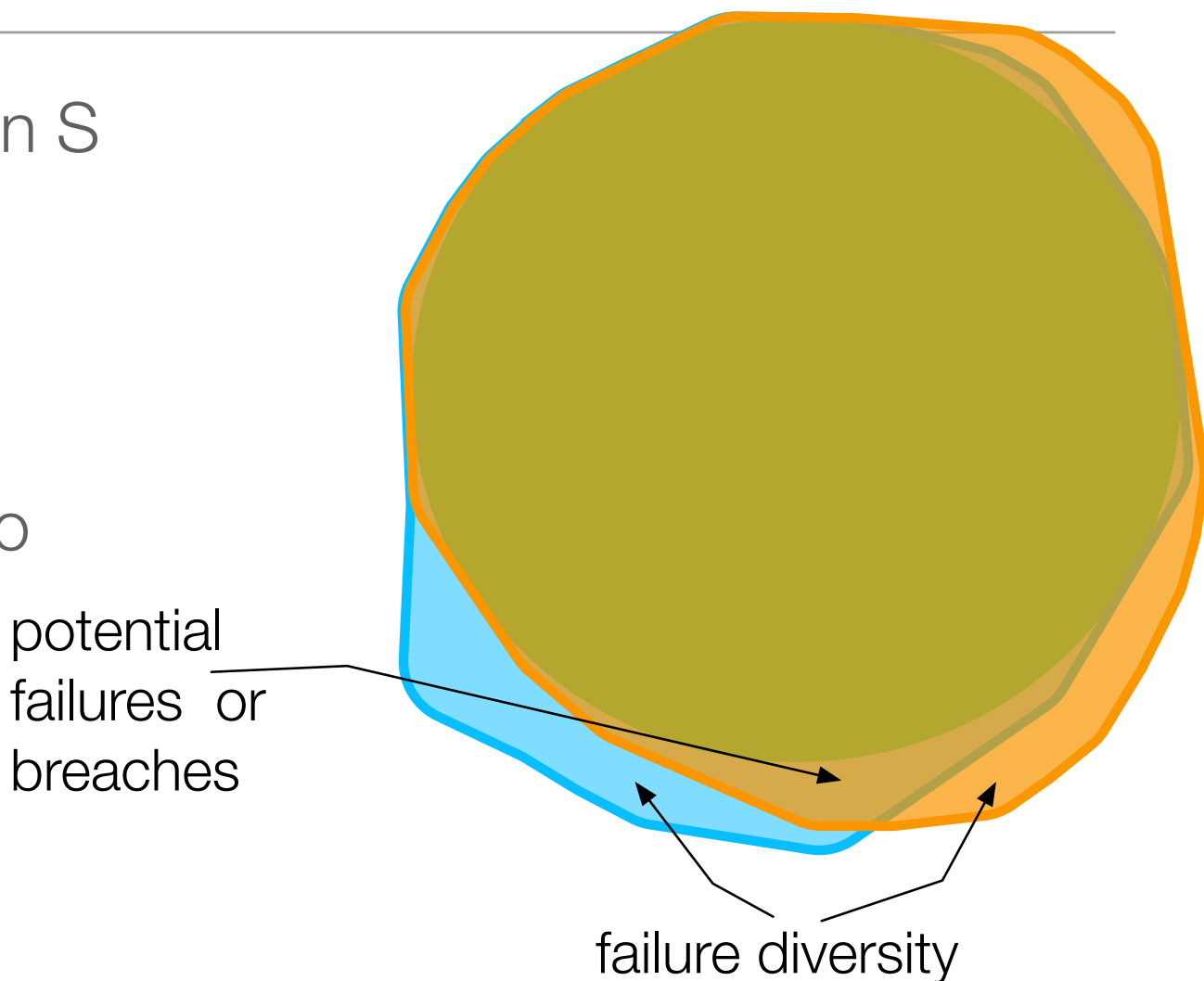
program

diversity of  
functionally  
similar programs



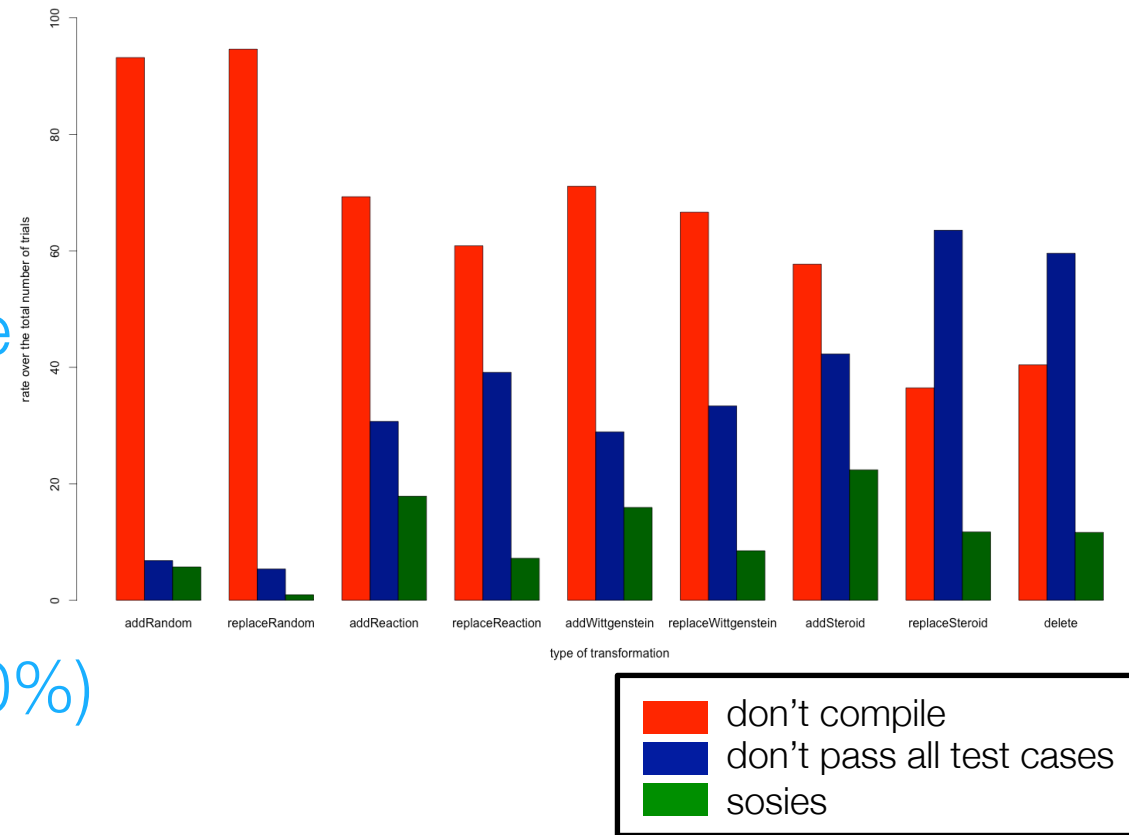
# sosie program

- Given a specification  $S$
- Given a program  $P$  that conforms to  $S$
- A sosie of  $P$  is a variant of  $P$  that also conforms to  $S$



# sosie programs

- 9 Java libraries
  - ~ 150K LoC
  - replace/delete/rename statements
  - nb of trials: 298938
  - nb of sosie: 28805 (10%)



# sosie program

---

```

public static boolean isAssignable(Class<?>[] classArray,
Class<?>[] toClassArray, final boolean autoboxing) {
    if (ArrayUtils.isSameLength(classArray, toClassArray) ==
false)
        {return false;}
    if (classArray == null) {
        classArray = ArrayUtils.EMPTY_CLASS_ARRAY;
    }
    if (toClassArray == null) {
toClassArray = ArrayUtils.EMPTY_CLASS_ARRAY;
    }
    for (int i = 0; i < classArray.length; i++) {
        if (isAssignable(classArray[i], toClassArray[i],
autoboxing) == false) {
            return false;
        }
    }
    return true;
}

```

# Netflix's simian army

---

- Induce failure regularly
  - break production code to check the system's ability to react
- Chaos monkey
  - "to randomly shoot down instances and chew through cables"
- Latency monkey
  - artificial delay in RESTful clients
- Chaos Gorilla
  - simulate shut down of an entire region
- Open source
  - <https://github.com/Netflix/SimianArmy>

NETFLIX



# Conclusion

---

- Different techniques for robust ordinary software
  - unsound repair; accuracy / energy trade-off; diversity injection; fault injection
- The software engineering community develops new approaches for the construction of robust applications
  - that is good enough
  - that is safe enough
  - that runs continuously

# Foundations

---

- Obtaining and Reasoning About Good Enough Software
  - M. Rinard. 2012.
  - <http://people.csail.mit.edu/rinard/paper/dac12.pdf>
- Building Robust Systems. An essay.
  - G.J. Sussman. 2007.
  - <http://groups.csail.mit.edu/mac/users/gjs/essays/robust-systems.pdf>
- Self-healing: softening precision to avoid brittlenes
  - M. Shaw. 2002.
  - <http://www.cs.cmu.edu/afs/cs/project/compose/ftp/pdf/shaw-homeostasis-fin.pdf>
- Building Diverse Computer Systems.
  - S. Forrest, A. Somayaji, D. Ackley. 1997.
  - <http://iar.cs.unm.edu/~forrest/publications/hotos-97.pdf>
- Design of self-checking software
  - S. Yau and R. Cheung. 1975.



# References

---

- FlexJava: Language Support for Safe and Modular Approximate Programming
  - J.Park. 2015
- Multi-tier diversification in Web-based software applications
  - S. Allier. 2015
  - <https://hal.archives-ouvertes.fr/hal-01089268/document>
- Tailored Source Code Transformations to Synthesize Computationally Diverse Program Variants
  - B. Baudry. 2014
  - <https://hal.archives-ouvertes.fr/file/index/docid/938855/filename/sosies.pdf>
- Post-compiler Software Optimization for Reducing Energy
  - E. Schulte. 2014.
  - <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.8820&rep=rep1&type=pdf>
- Automatic Runtime Error Repair and Containment via Recovery Shepherdng
  - F. long. 2014.
  - <http://people.csail.mit.edu/fanl/papers/rcv-pldi14.pdf>
- Managing Performance vs. Accuracy Trade-offs With Loop Perforation
  - S. Sidiroglou. 2011.
  - <http://people.csail.mit.edu/misailo/papers/fse2011.pdf>
- Netflix's Simian army
  - <http://techblog.netflix.com/2011/07/netflix-simian-army.html>