Software Diversity

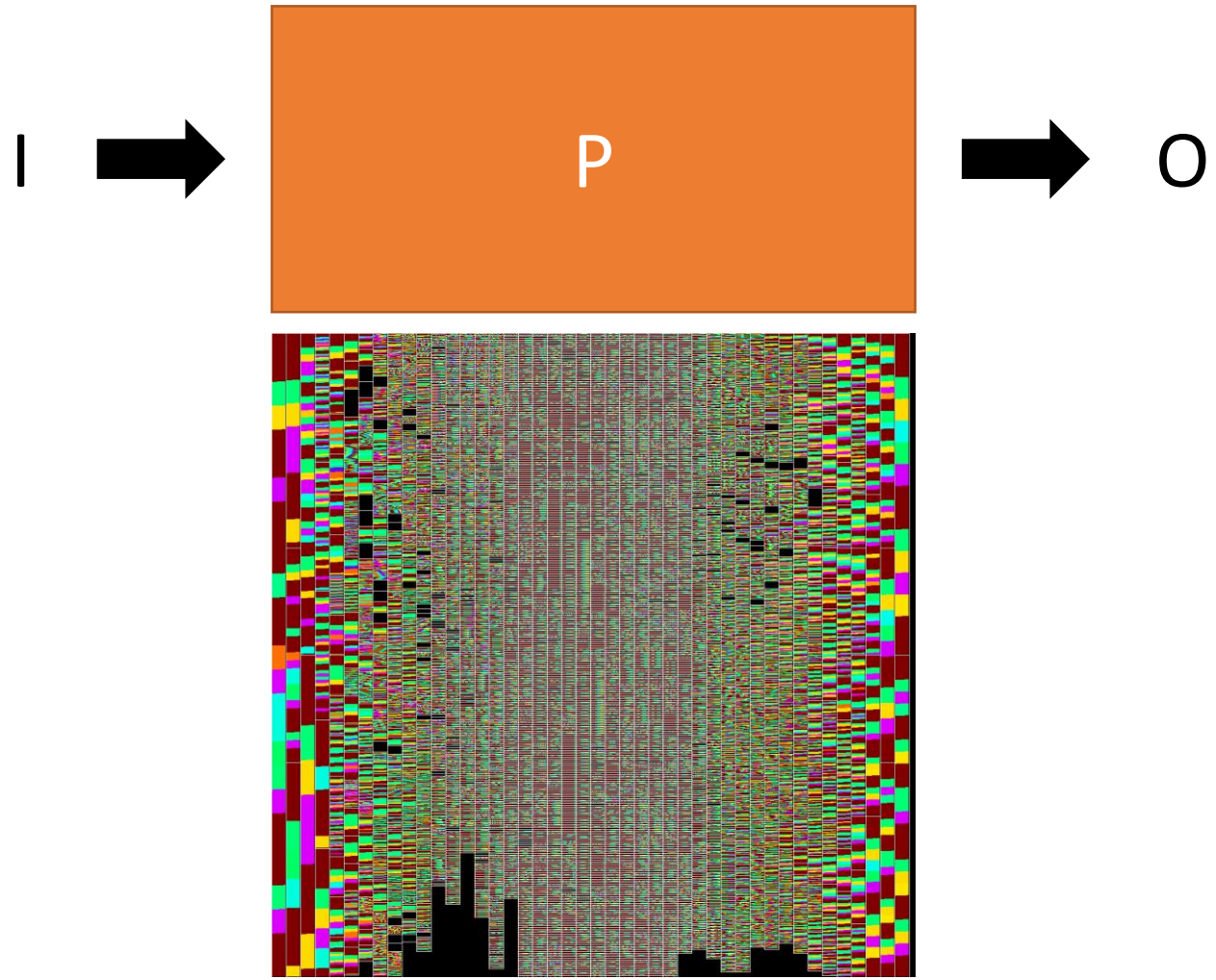
1 concept and 10 papers I Love

Benoit Baudry

Professor, KTH

CASTOR
Software Research Centre


I
P
O

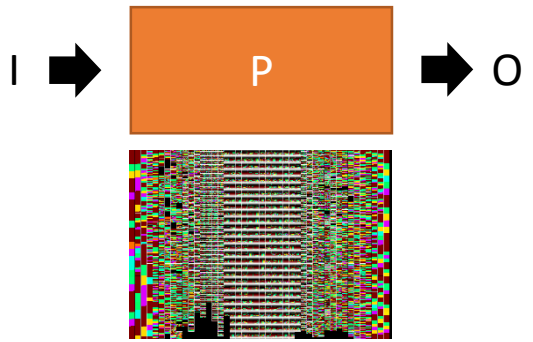

I
P
O

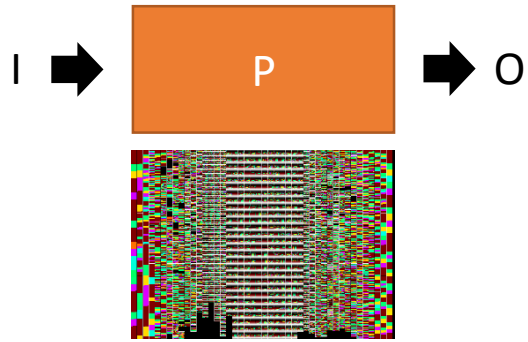

I
P
O

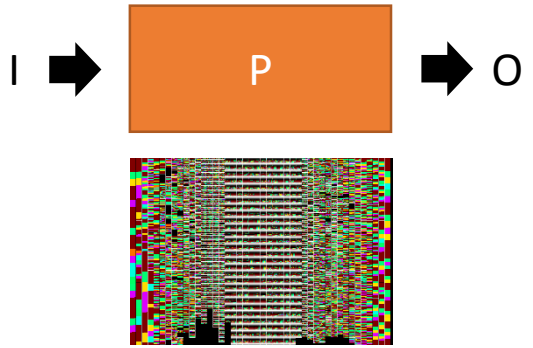

I
P
O

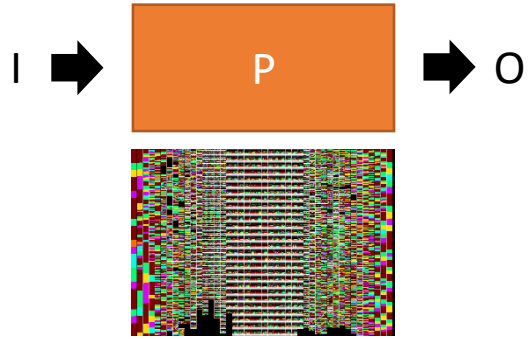

I
P
O


I
P
O

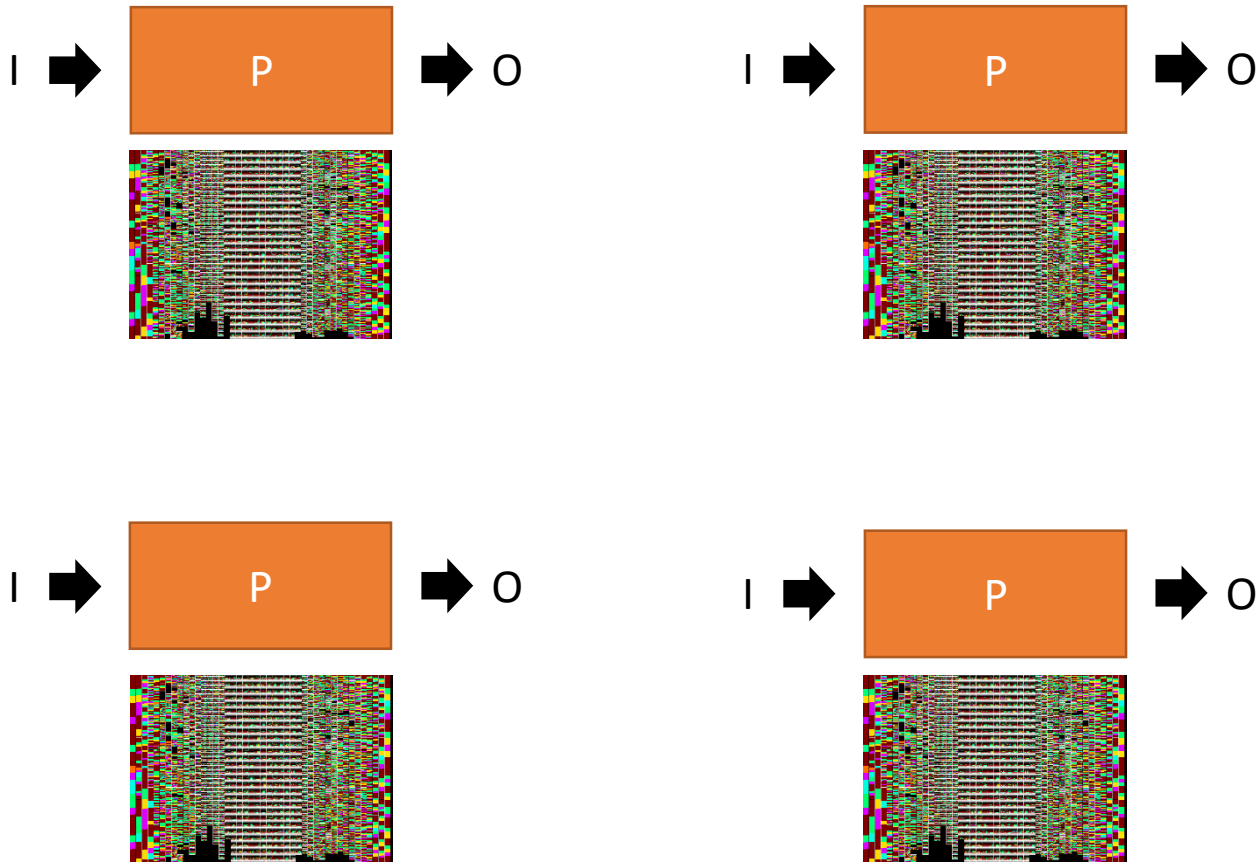

I
P
O

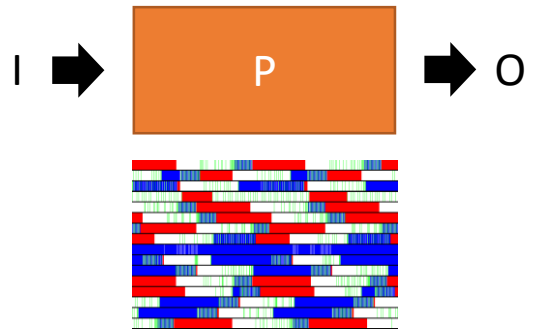

I
P
O

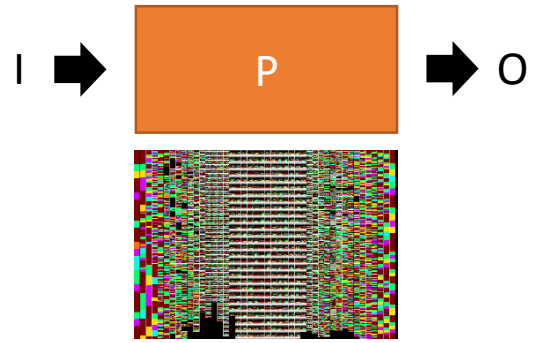

I
P
O

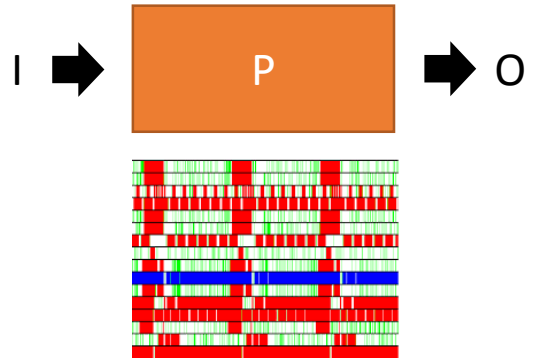

I
P
O

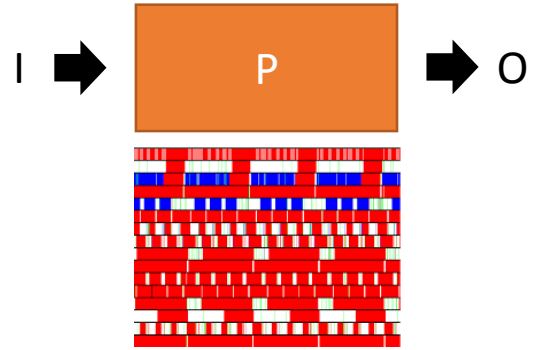
Risks of monoculture
- No specialization
- Same bugs
- Same vulnerabilities



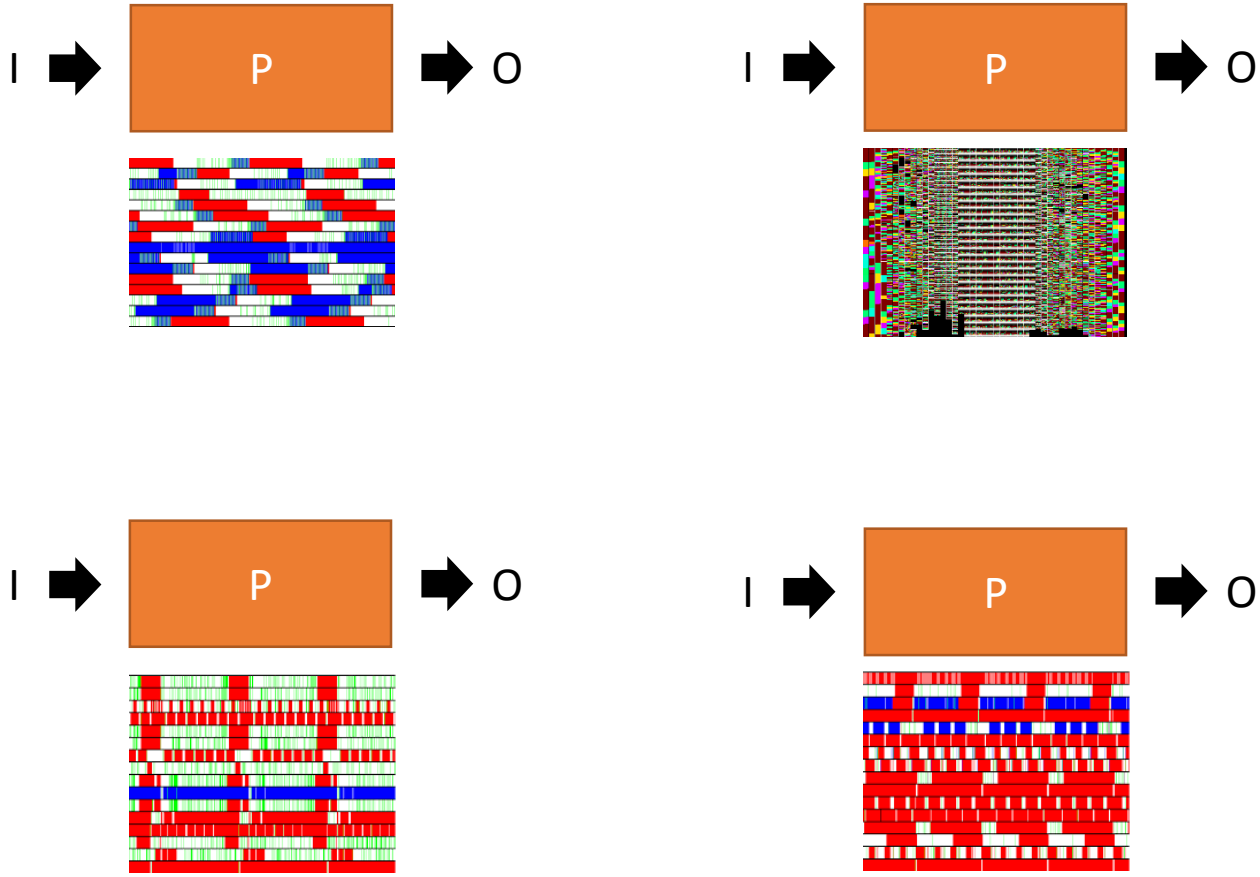

I
P
O


I
P
O


I
P
O


I
P
O

Software diversity mitigates the risks of software monoculture with diverse behaviors



Software diversity

SRSLSLRSRLLSSRRLRL

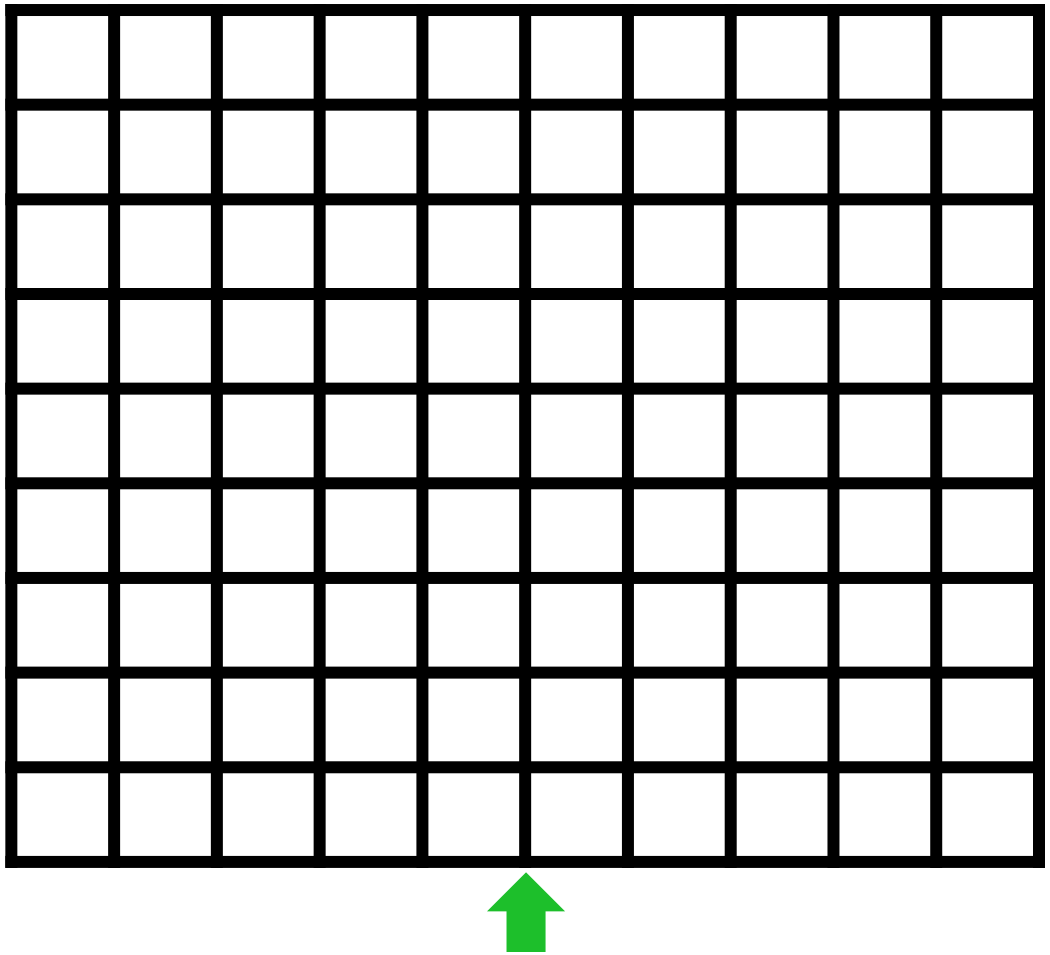
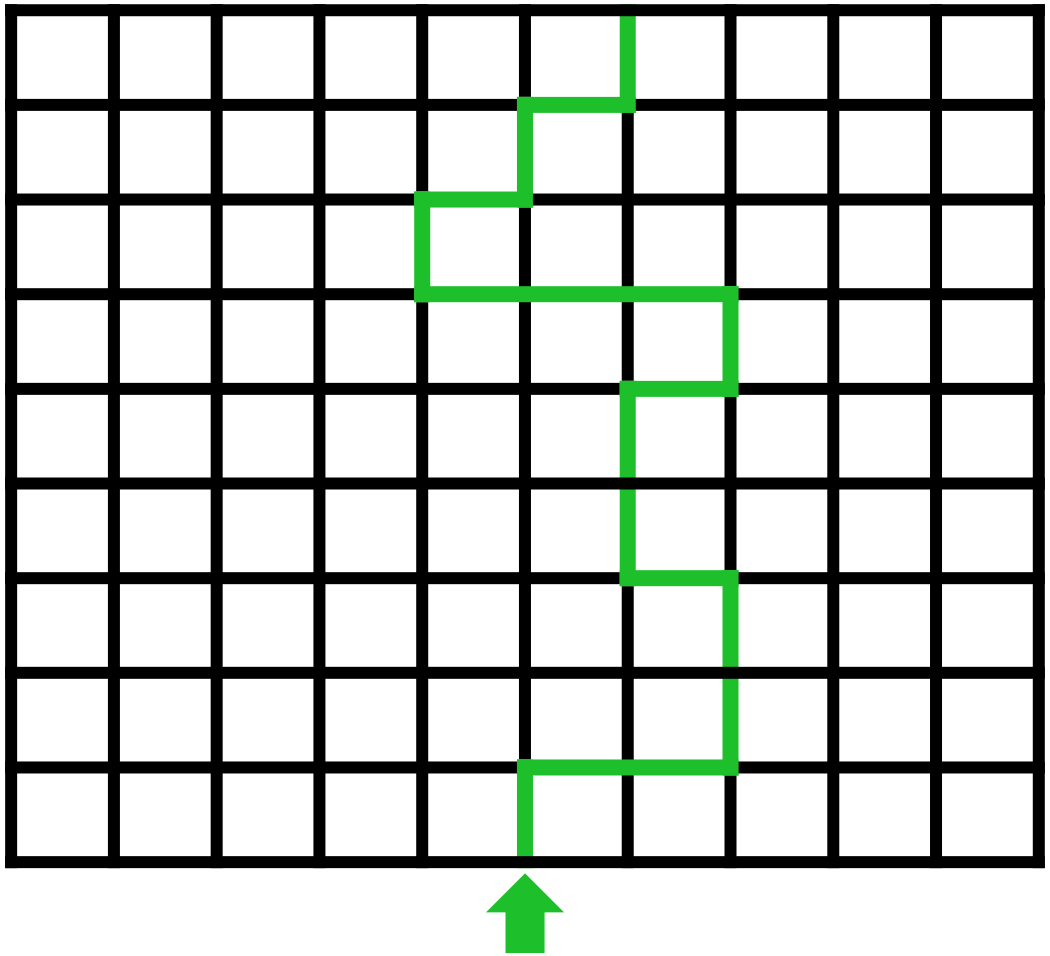


Illustration inspired by G. Berry. « A la chasse aux bugs, la maladie du certain » (8 juin 2011)

Software diversification

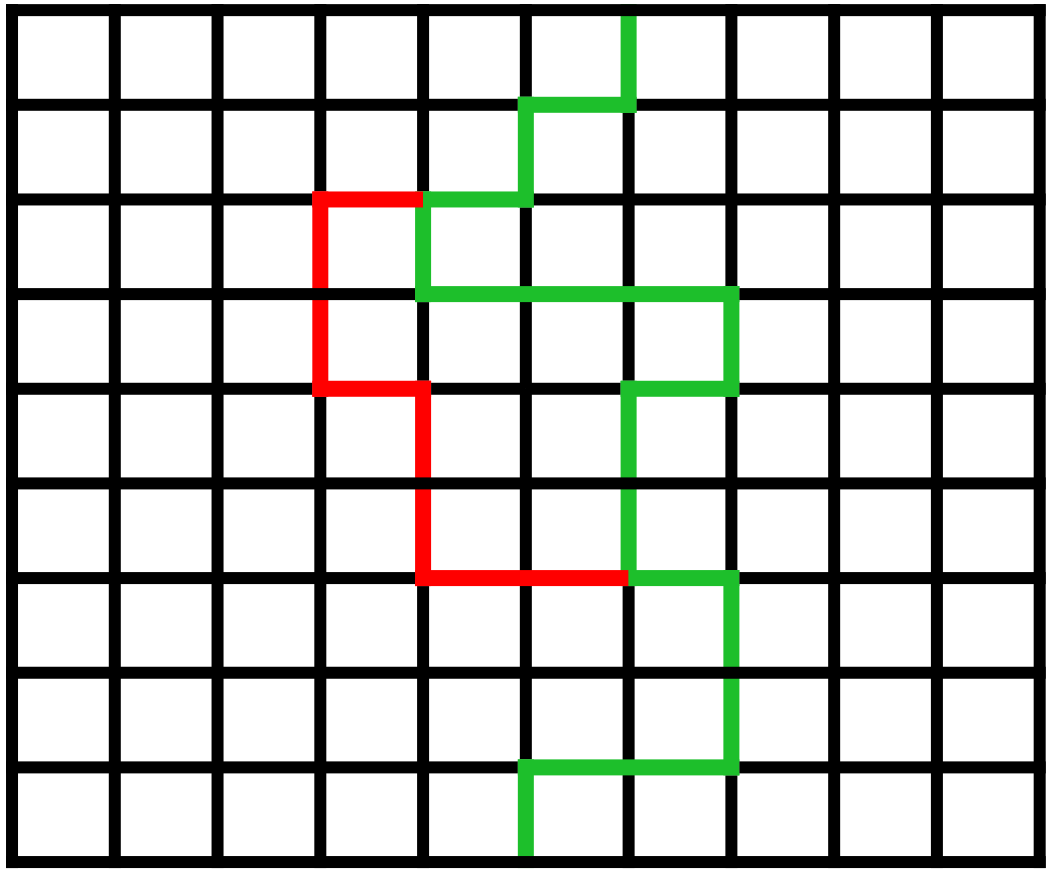
SRSLSLRSRLLSSRRLRL



Software diversification

SRSLSLRSRLLSSRRLRL

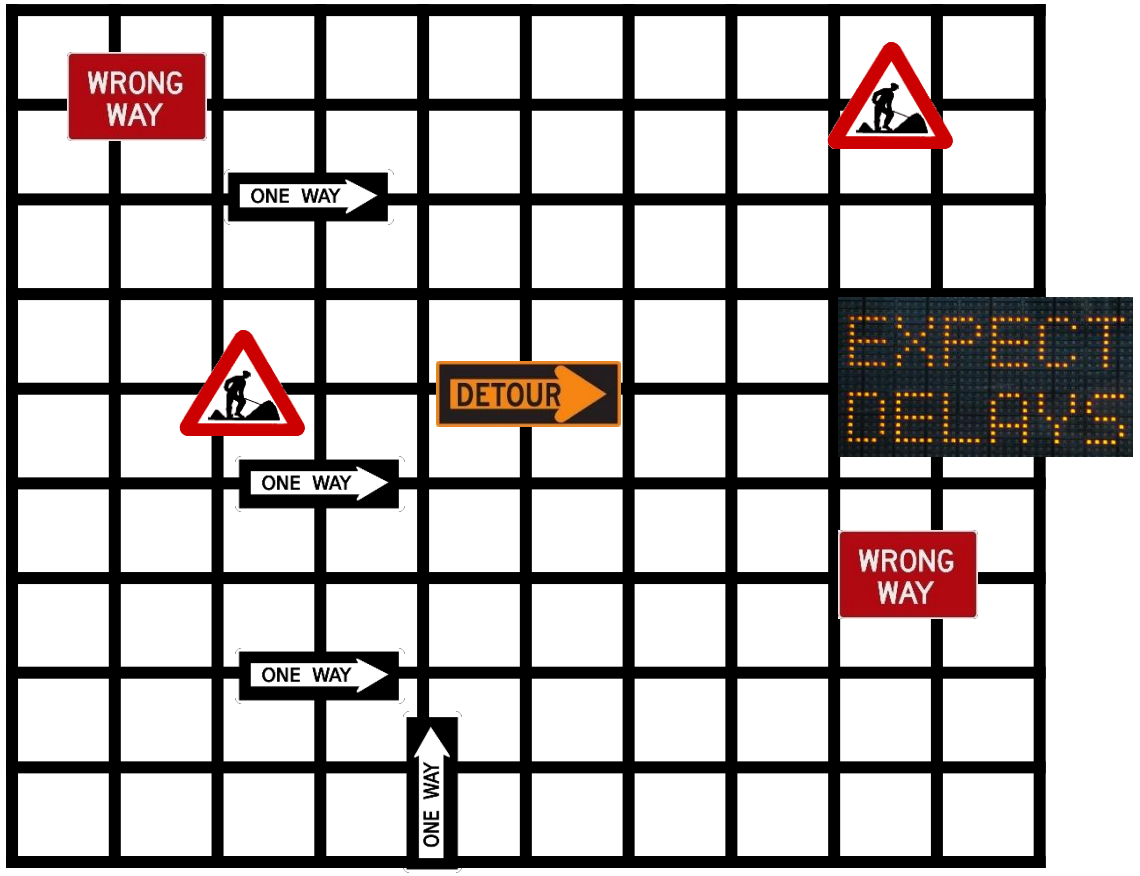
variant



Software diversification

SRSLSLRSRLLSSRRLRL

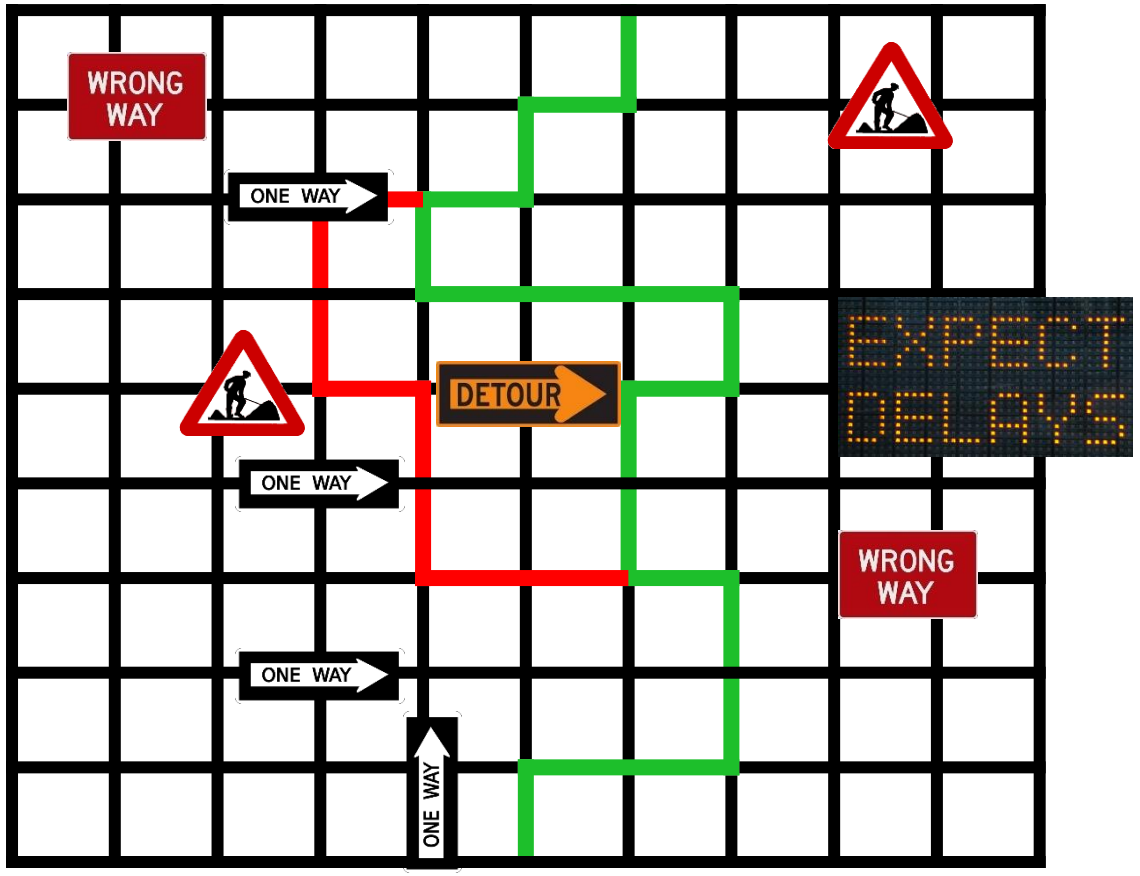
variant

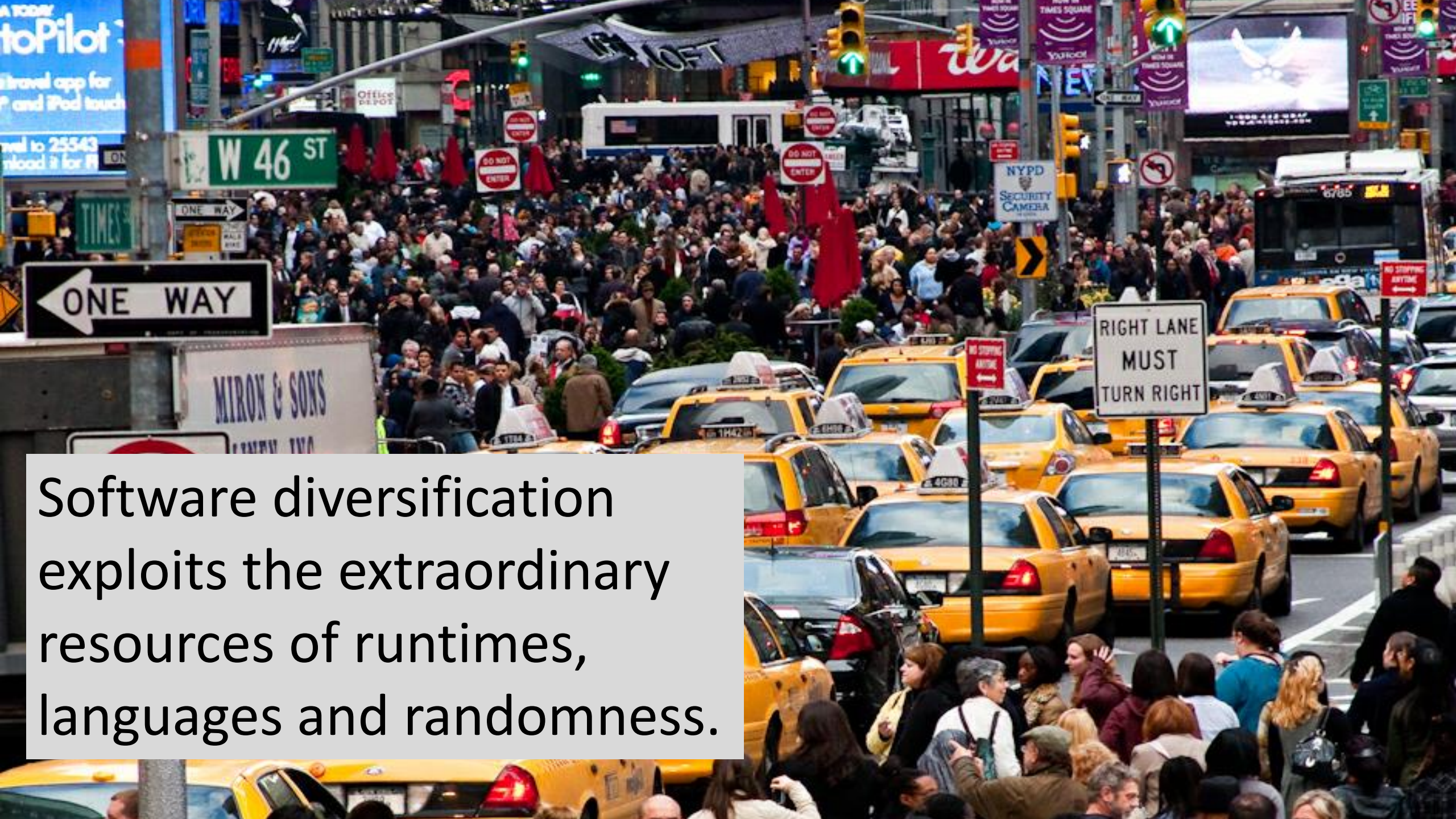


Software diversification

SRSLSLRSRLLSSRRLRL

SRSLSLSSRLLSSRRLRL





Software diversification exploits the extraordinary resources of runtimes, languages and randomness.

A journey into software diversity

Precursors

- S. Yau. Design of self-checking software. 1975.
- Brian Randell. System structure for software fault tolerance. 1975.
- A. Avizienis. The N-version approach to fault-tolerant 1985.

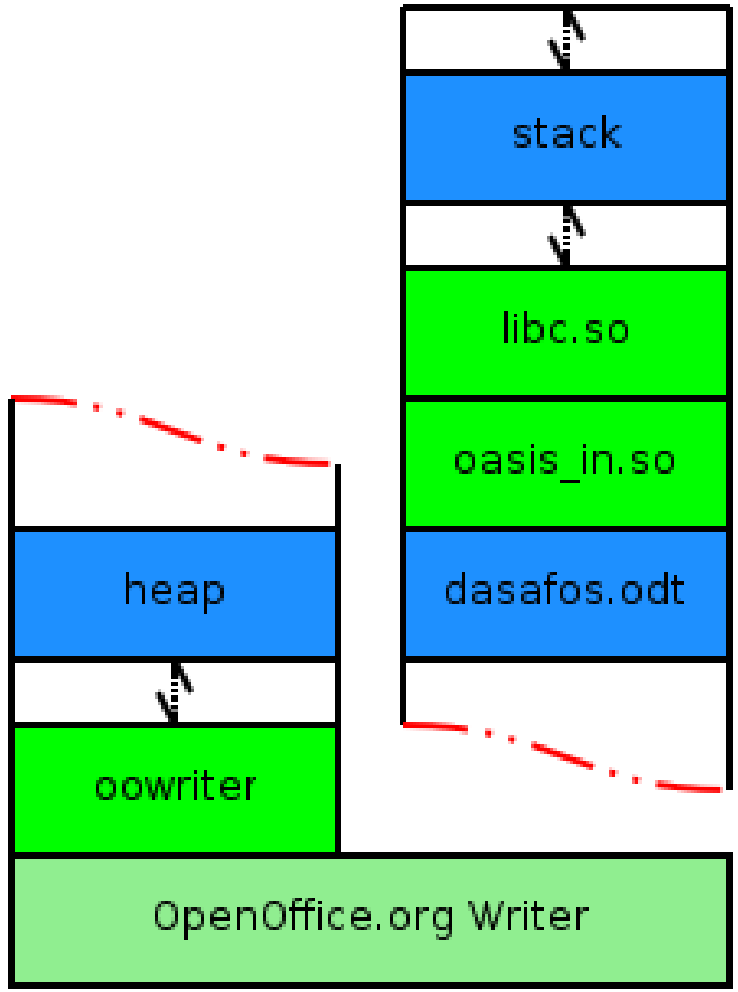
Pioneers of automatic diversification

- Fred Cohen, 1993
  - Increase the costs of attacks
  - Program transformations
  - Pioneer: reordering, garbage insertion, function mix
- Stephanie Forrest, 1997
  - Biological inspiration
  - Avoid unnecessary consistency
  - Pioneer : NOP insertion, random memory padding
  - Prototype of randomized stack layout

Fred Cohen. Operating system protection through program evolution. 1993.
Stephanie Forrest, Anil Somayaji, David Ackley. Building Diverse Computer Systems. 1997.

Address space layout randomization

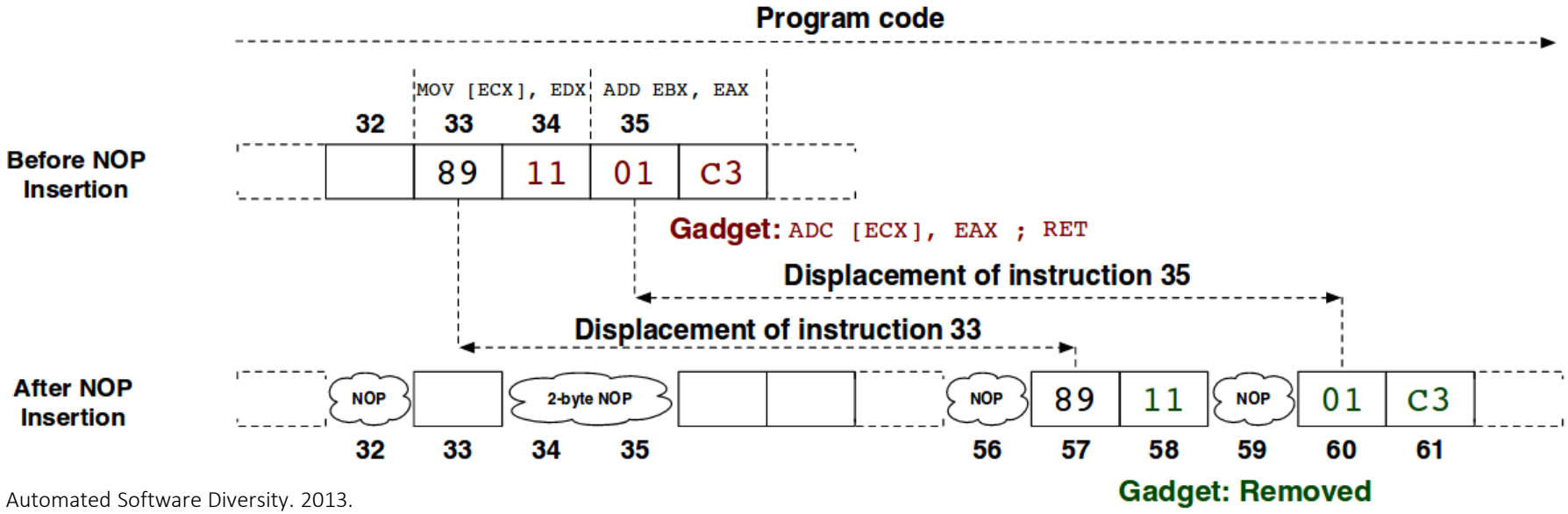
- PaX Linux kernel patch. 2000.
  - Separate readable data pages and executable code pages
  - Address space layout randomization: heap, stack and libraries
- ASLR is now in all main Oss
  - Mitigates ret-to-libc and stack smashing



https://en.wikipedia.org/wiki/PaX
Shacham, H. and colleagues. On the Effectiveness of Address-Space Randomization. 2004

NOP insertion

- Compiler-based diversification
  - Randomly insertr NOPs in the generated binary
  - One different binary at each compilation
- Mitigates return oriented programming



Homescau and colleagues. Profile-guided Automated Software Diversity. 2013.

Good enough software

- Functionality removal, computation discard
- Mitigate homogeneous performance


```
for (i = 0; i < n; i++) { … }
```

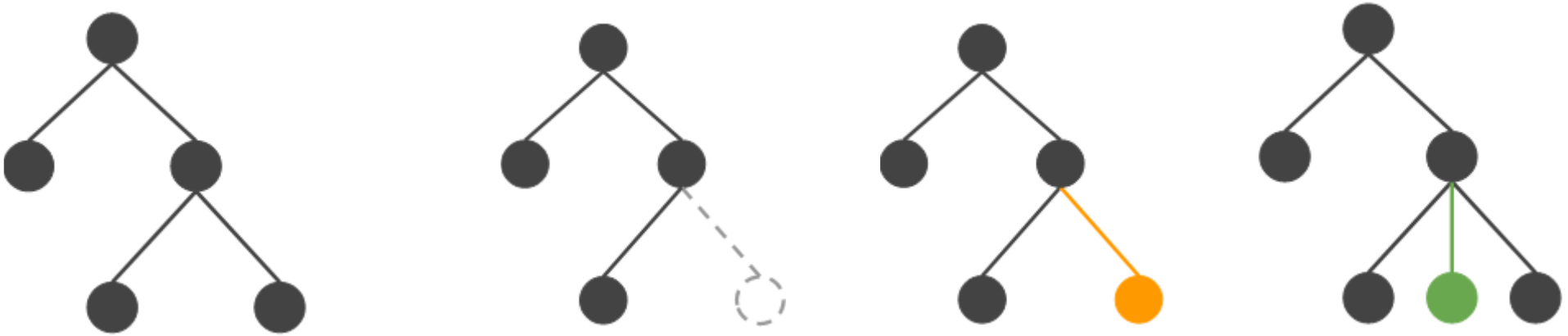
⇩

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


Rinard. Obtaining and Reasoning About Good Enough Software. 2012.

Mutational robustness

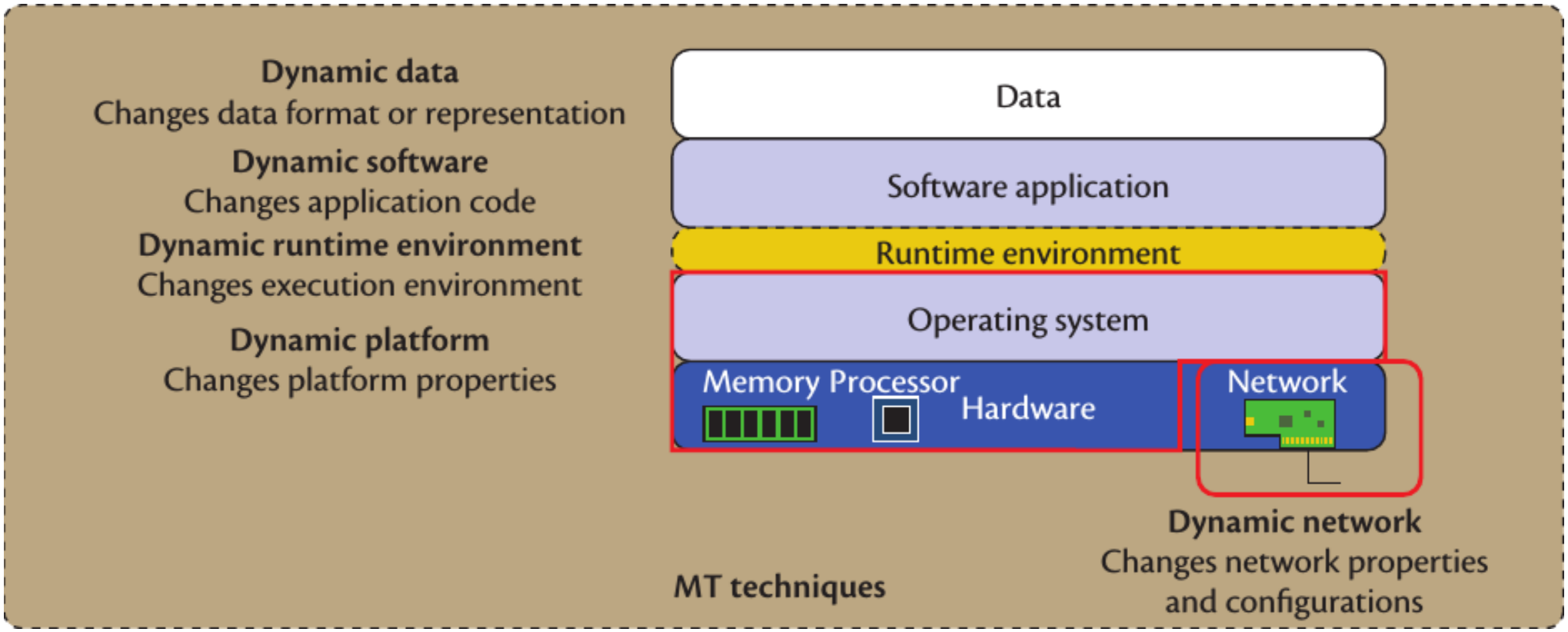
- Source can be randomly transformed with speculative transformations
- Empirical evidence of software mutational robustness
- Mitigates risks of bug and vulnerability monoculture



Schulte and colleagues. Software mutational robustness. 2013.

Moving Target Defenses

- Runtime evolution + diversity



Okhravi and colleagues. Finding Focus in the Blur of Moving-Target Techniques. 2014

Conclusion

- The forces of monoculture are strong
  - Technical standards (e.g., JSON)
  - Socio-technical networks (e.g., Github)
  - The penetration of software in society (e.g., Wordpress)
- Extraordinary challenges to fuel software diversity
  - Remodel the natural diversity of code strata
  - Embrace evolution with DevOps
  - Explore the space of short-lived data and programs