Smart Debugging

Stijn de Gouw

CWI Amsterdam / SDL Fredhopper

CWI In Bedrijf
Amsterdam, 4 juni 2015

Centrum Wiskunde & Informatica

Smart Debugging

Kosten van software fouten

- Onderzoek Amerikaanse overheid in 2002: $59 miljard (jaarlijks)
- Nieuw onderzoek in 2013 door Cambridge: $312 miljard (jaarlijks)
- Subtiele fouten, maar met enorme gevolgen: vliegtuigen, auto's, medische apparatuur
- Imago schade

Smart Debugging

Kosten van software fouten

- Onderzoek Amerikaanse overheid in 2002: $59 miljard (jaarlijks)
- Nieuw onderzoek in 2013 door Cambridge: $312 miljard (jaarlijks)
- Subtiele fouten, maar met enorme gevolgen: vliegtuigen, auto's, medische apparatuur
- Imago schade

Oplossing: testen, debuggen ?

- Test case: voer algoritme uit op 1 specifieke invoer en vergelijk met de verwachte uitvoer
- Vindt fouten na uitvoeren van programma op test case
- Maar: zeer veel / oneindig veel mogelijke invoeren
- Dus: testen en debuggen houdt nooit op

Smart Debugging

Kosten van software fouten

- Onderzoek Amerikaanse overheid in 2002: $59 miljard (jaarlijks)
- Nieuw onderzoek in 2013 door Cambridge: $312 miljard (jaarlijks)
- Subtiele fouten, maar met enorme gevolgen: vliegtuigen, auto's, medische apparatuur
- Imago schade

Oplossing: testen, debuggen ?

- Test case: voer algoritme uit op 1 specifieke invoer en vergelijk met de verwachte uitvoer
- Vindt fouten na uitvoeren van programma op test case
- Maar: zeer veel / oneindig veel mogelijke invoeren
- Dus: testen en debuggen houdt nooit op

Smart Debugging, recente toepassing: Java, Python, Android

Smart Software
Domain Specific Languages

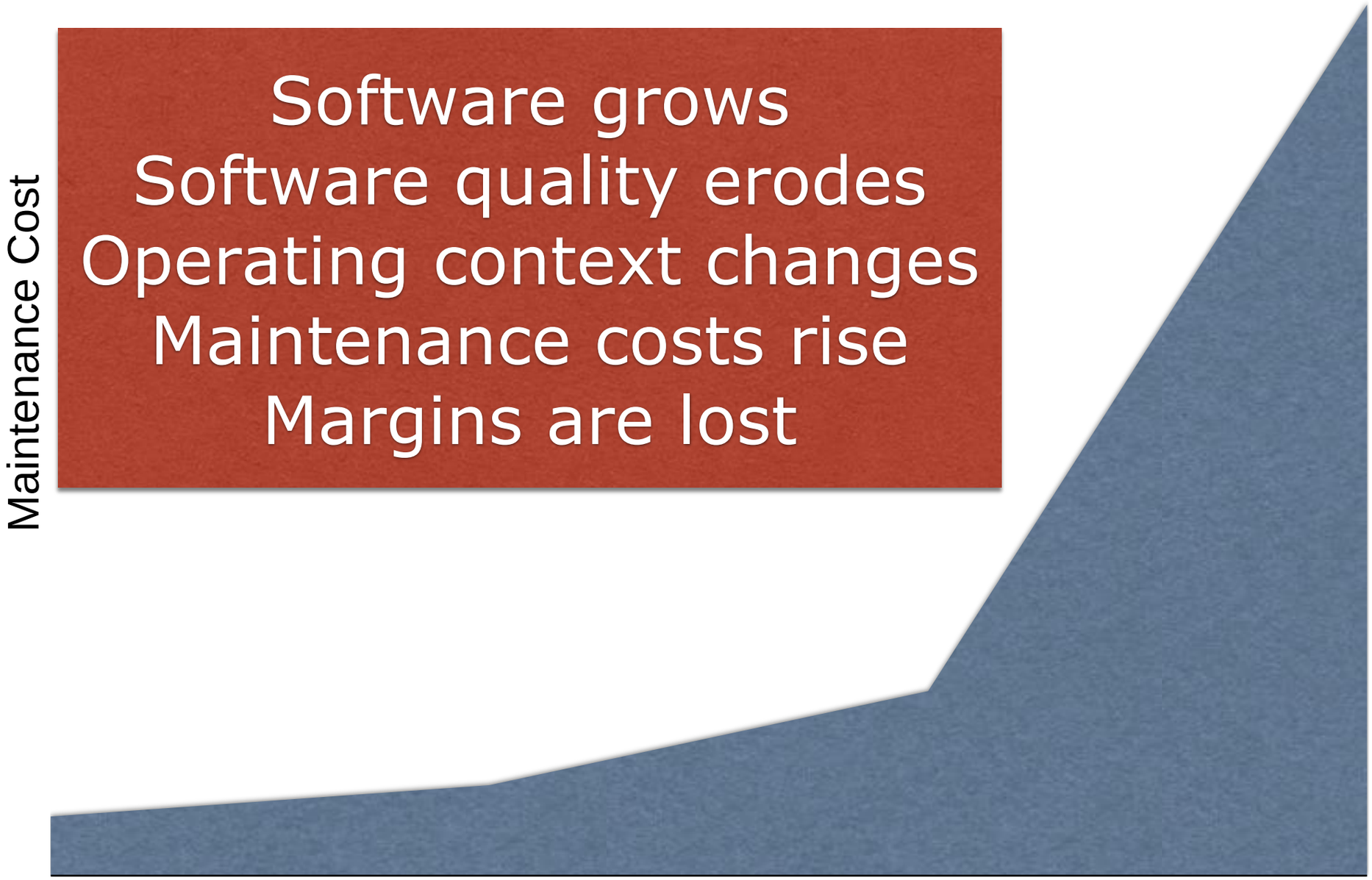
Jurgen J. Vinju
Tijs van der Storm
Jouke Stoel
Davy Landman

Software Economy

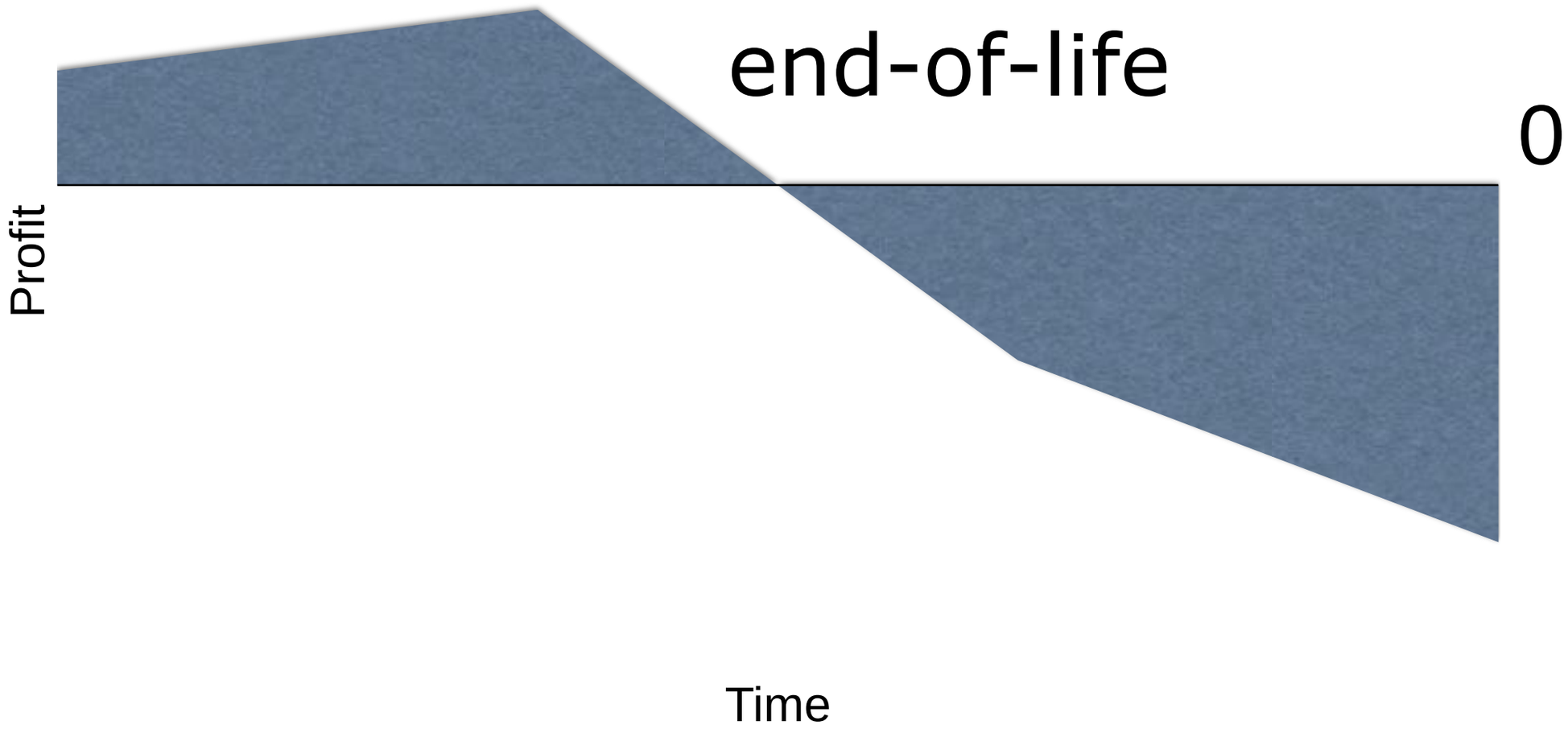
Software investments pay off due to high innovation impact and (relatively) low operating costs

But: Software Evolution

Software grows
Software quality erodes
Operating context changes
Maintenance costs rise
Margins are lost

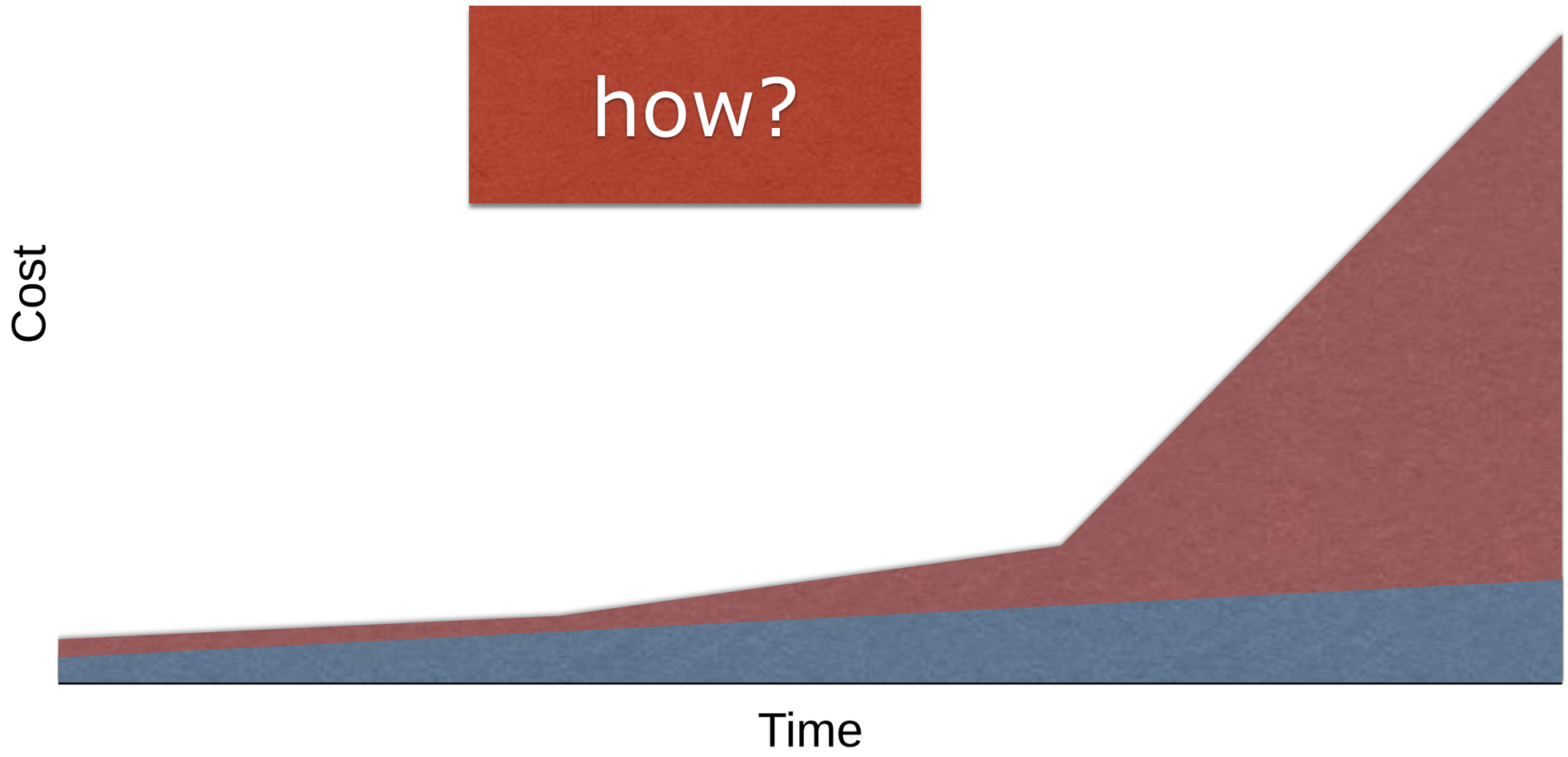


Maintenance costs matter

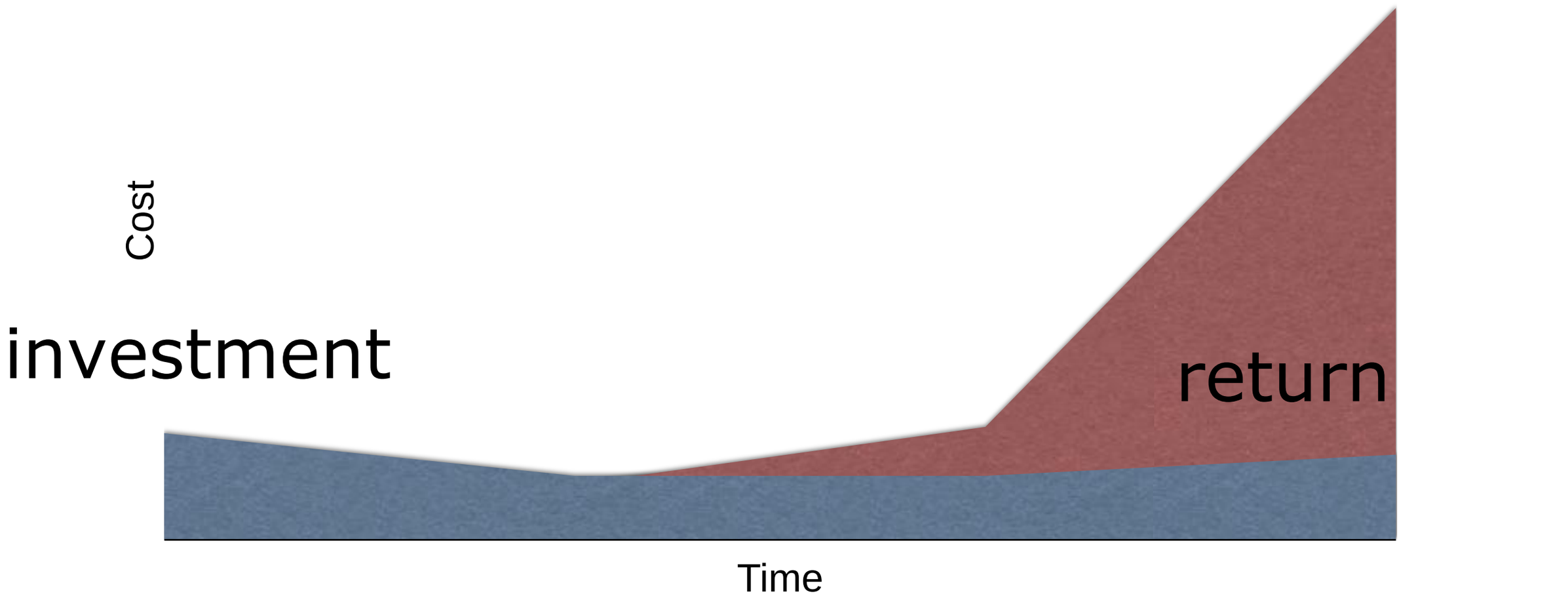


Lowering maintenance costs is a prime factor to longer term ROI of software-based innovation

Avoid maintenance cost



Investing in domain knowledge and automation: domain specific languages



one-to-many

- one DSL program:
  - generates many possible products, now
  - is small and can evolve quickly
  - can have different targets: code, documentation, tests, simulations, …

many-to-one

- Many product instances
  - can be optimized by one DSL
  - can be fixed by fixing one DSL
  - can be ported to a new platform with one DSL

Demonstrations

AimValley
Machino
DSL for platform-specific Statemachine code generator

ING
Rebel
DSL for modeling, analysis and simulation of products

CWI
Marvol
Children's DSL to make a robot dance

NFI
Derric
Optimal code generation for digital forensics

Not-so-secret Ingredient

- http://www.rascal-mpl.org/
- **Lower investment cost** of language and IDE construction
- Enable **fast feedback**/improvement loops

- Rascal is more than DSLs
  - (continuous) code quality assessment,
  - (continuous) code modification,
  - refactoring, …
- *We are interested in more proof-of-concept studies with you*