



Technical Debt

Why you should care

Felipe Rubim

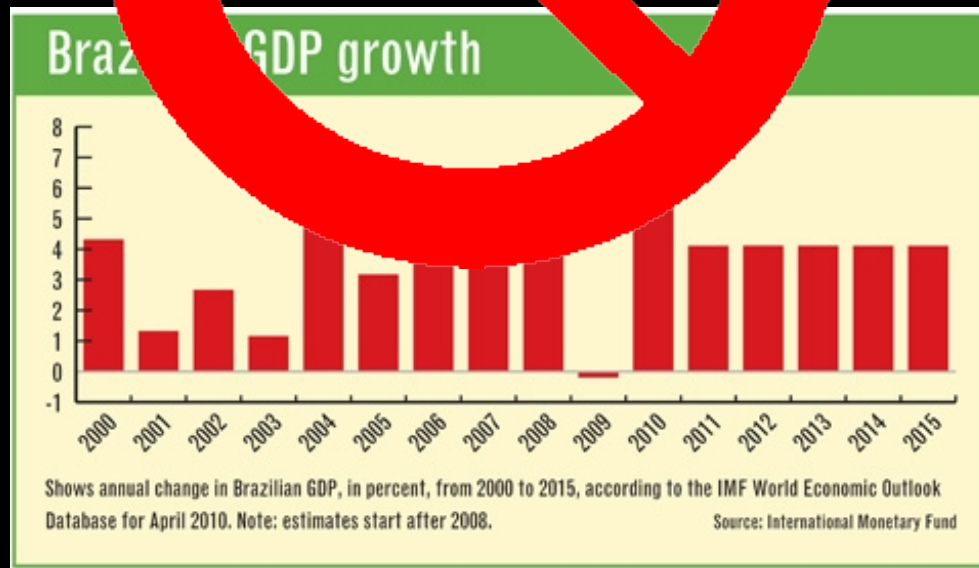
Architect and Team Lead at Ci&T

frubim@ciandt.com

[@frubim](#)

www.ciandt.com







**Oh, no! Another
session about
technical
debt?!?**

} Audience survey

1. Developers or Architects?
2. Product Owners/Project Managers?
3. Involved with Agile projects?

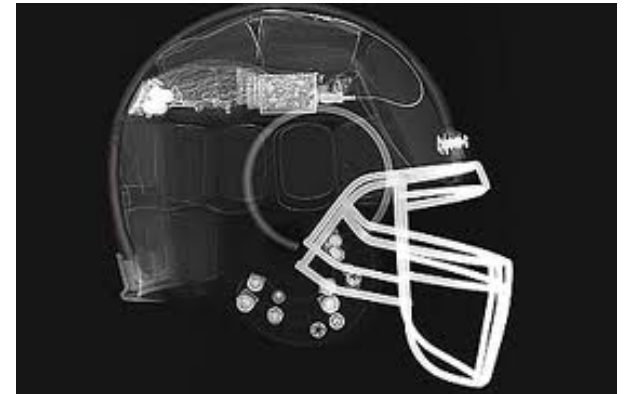
} Different perspectives on technical debt

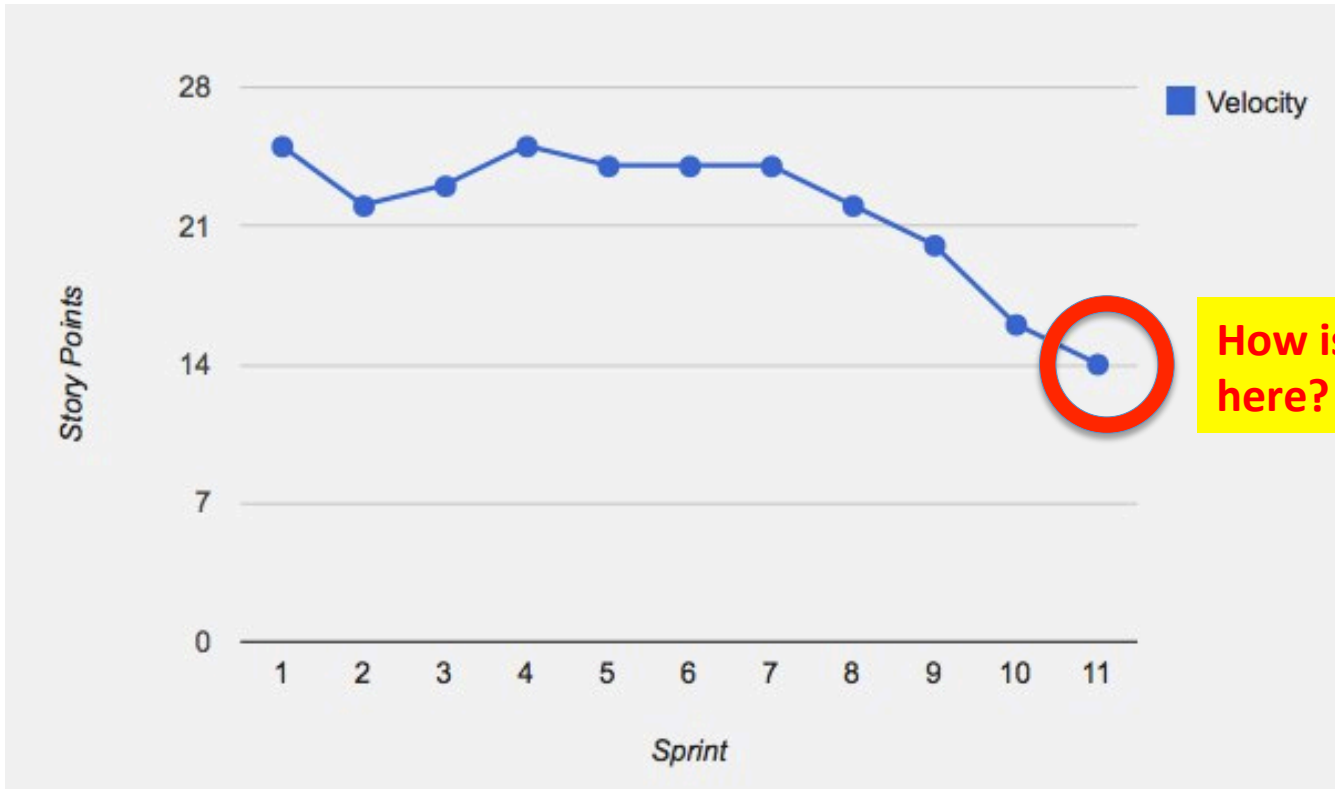
1. **Domain knowledge** – *~Sprint 3: “Now that we know more about this product, we should do something about that code we shipped”*
2. **Prudent design decisions** – *Sprint 1: “Let’s use this framework now and adapt to a better framework on the next sprints”*
3. **Sacrifice of good software development practices** – *“No time for refactoring. We will deal with this later”*



Face the brutal facts:

It will happen!





How is your P.O. here?

what the hell is happening with this project!?



which option would you go for?

1. the easy and quick way, but the code and solution doesn't seem quite proper...

Future looks dark...



2. a proper design and implementation along with refactoring, but it will take longer to ship.

No clouds ahead!



really?

18

*

DEADLINE

19

25

A close-up photograph of a middle-aged man with a balding head, wearing a white dress shirt and a striped tie. He is holding a black mobile phone to his ear with his left hand and another black mobile phone in his right hand, which is held up as if he is shouting into it. His mouth is wide open, showing his teeth, and his facial expression is one of intense anger or frustration. The background is a plain, light-colored wall.

really, really?

All right...



the metaphor

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

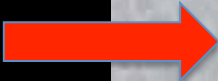
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas



the metaphor...

- every time you choose the first option (the “dark side one”) you create debt (like a financial debt)
 - if it is a debt, it incurs interest
- you may pay only the interest
- or you may pay down the principal, reducing the interest in the future



“Technical debt is everything that makes your code harder to change.”
(Tom Poppendiek - Lean Software Development)

such as....

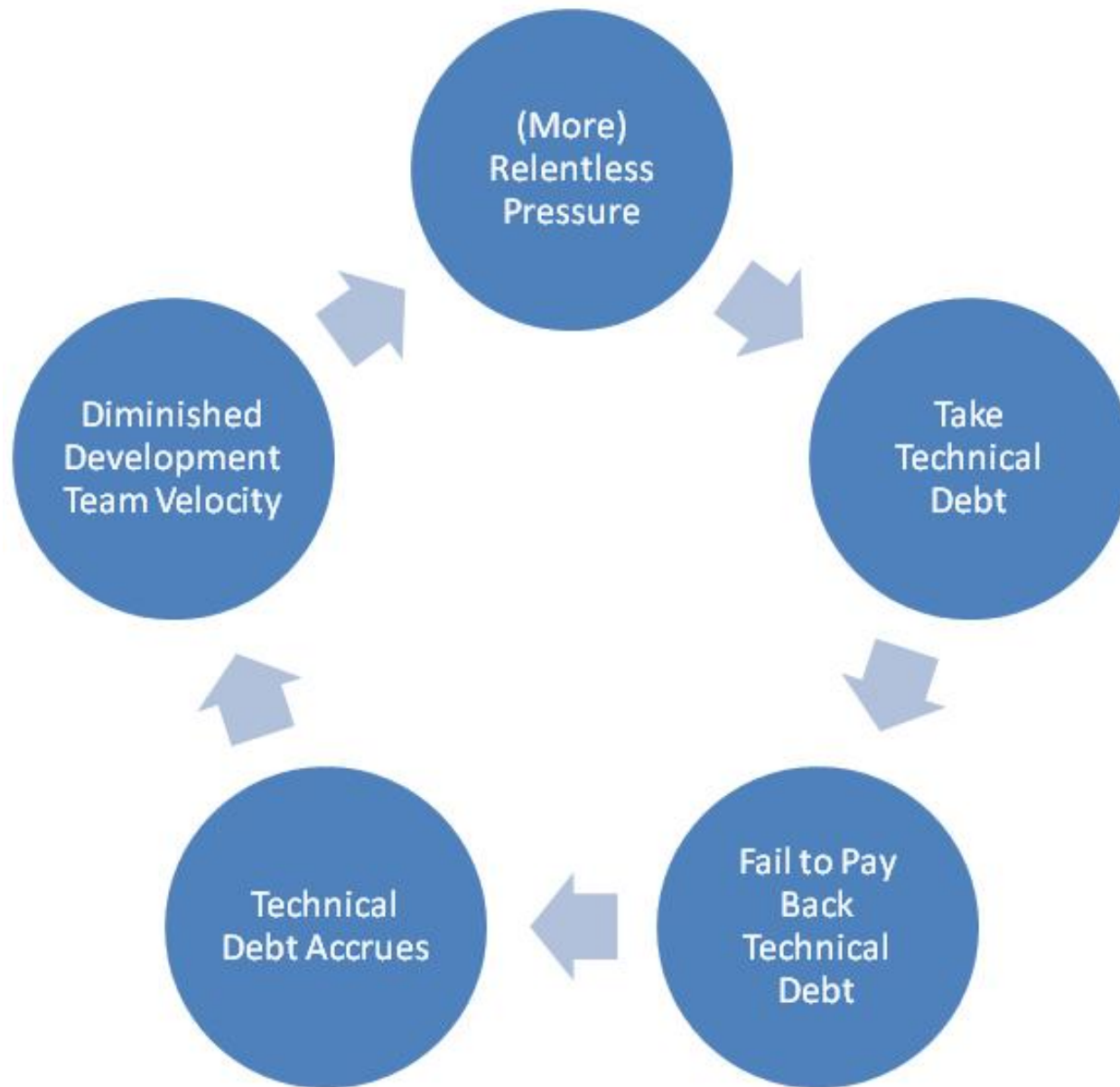
- Not the right design choices
- duplicated code
- lack of tests
- “workarounds”
- Unnecessary complexity
- ...

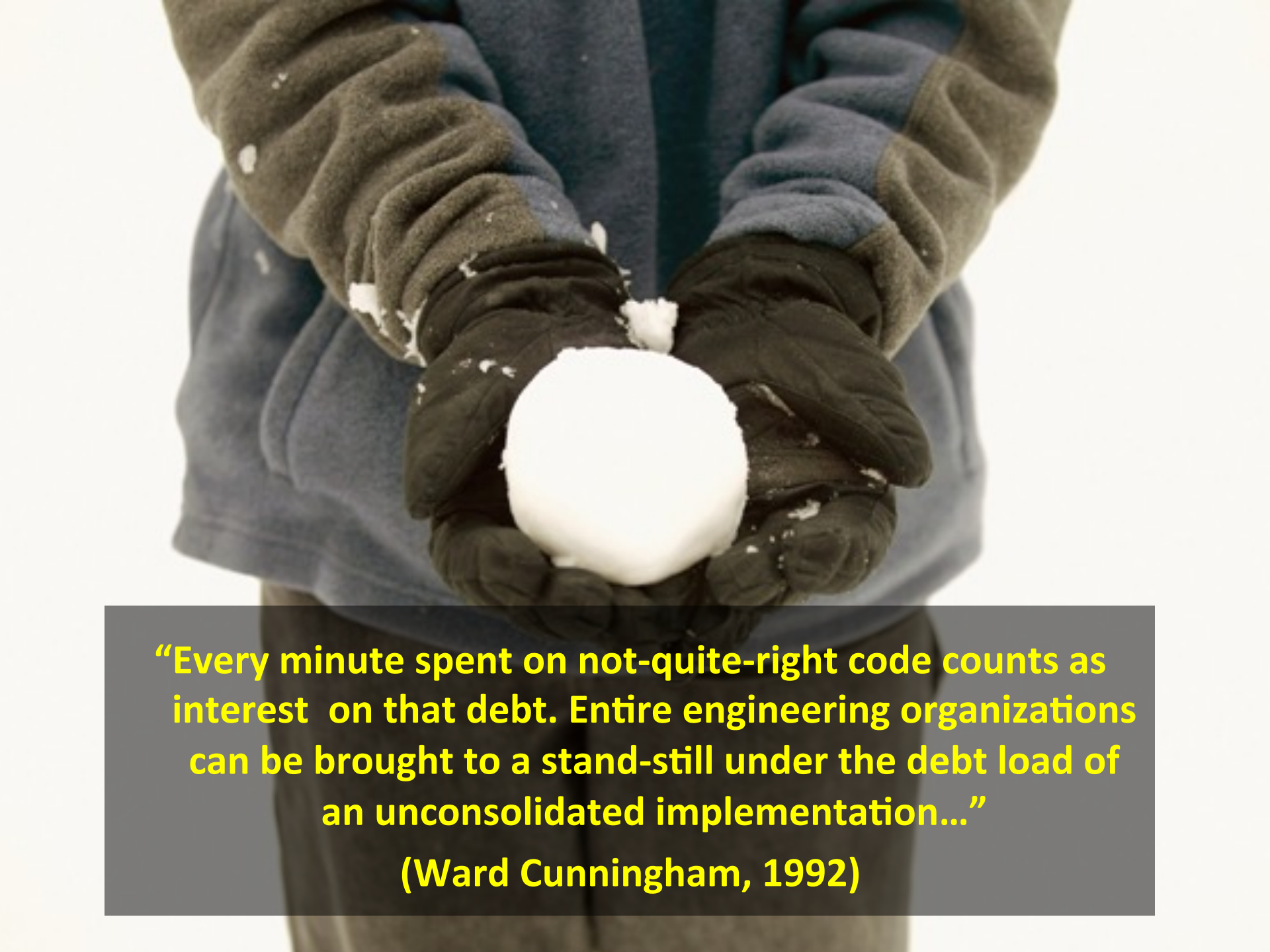


interest?!?

- bugs
- extra effort
- slower velocity
- milestone delays





A close-up photograph of a person's hands, wearing black gloves, holding a single, perfectly round snowball. The person is wearing a blue jacket with brown sleeves. The background is a plain, light color.

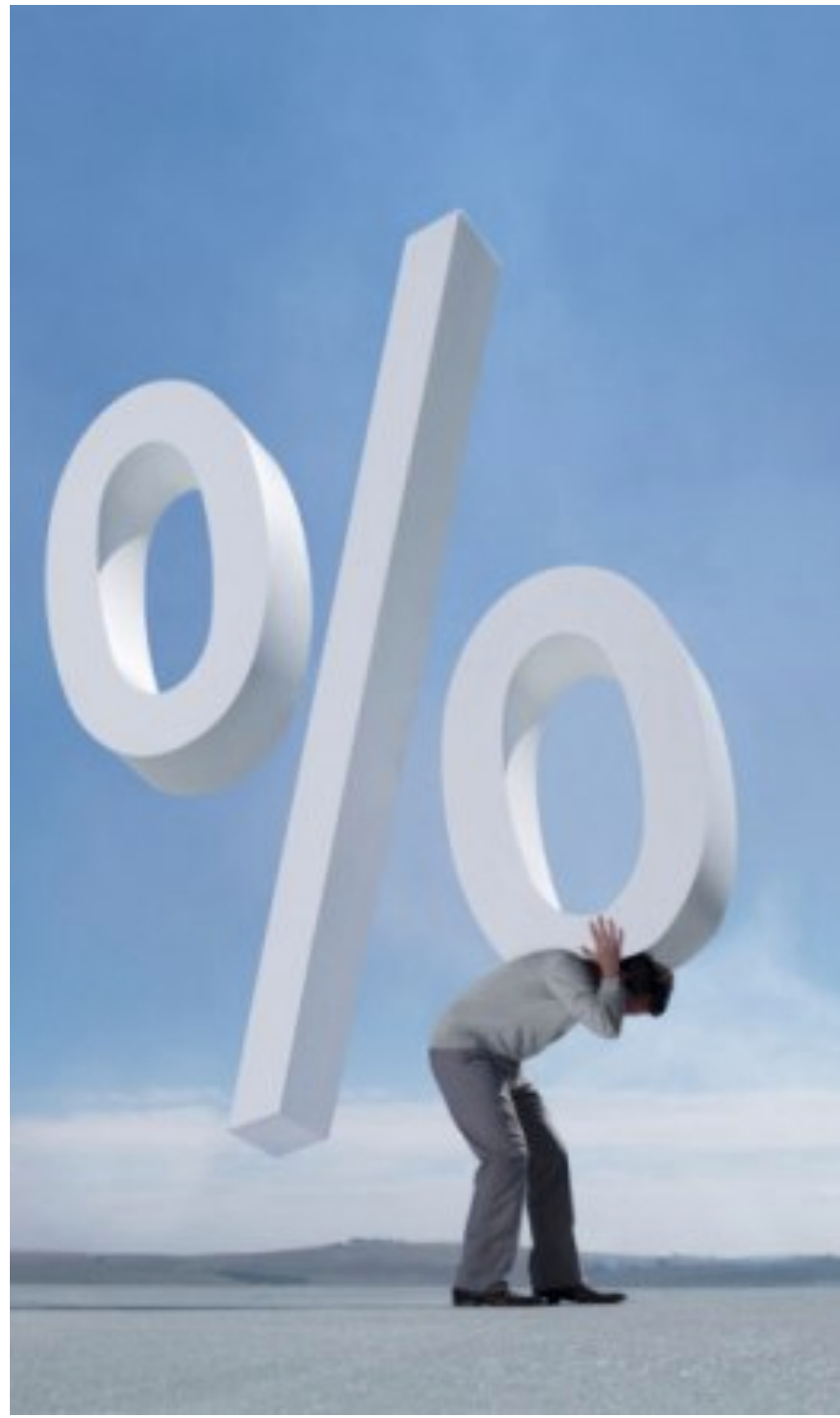
“Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unconsolidated implementation...”

(Ward Cunningham, 1992)

why you should care?

interest?!?

- bugs
- extra effort
- slower velocity
- delays

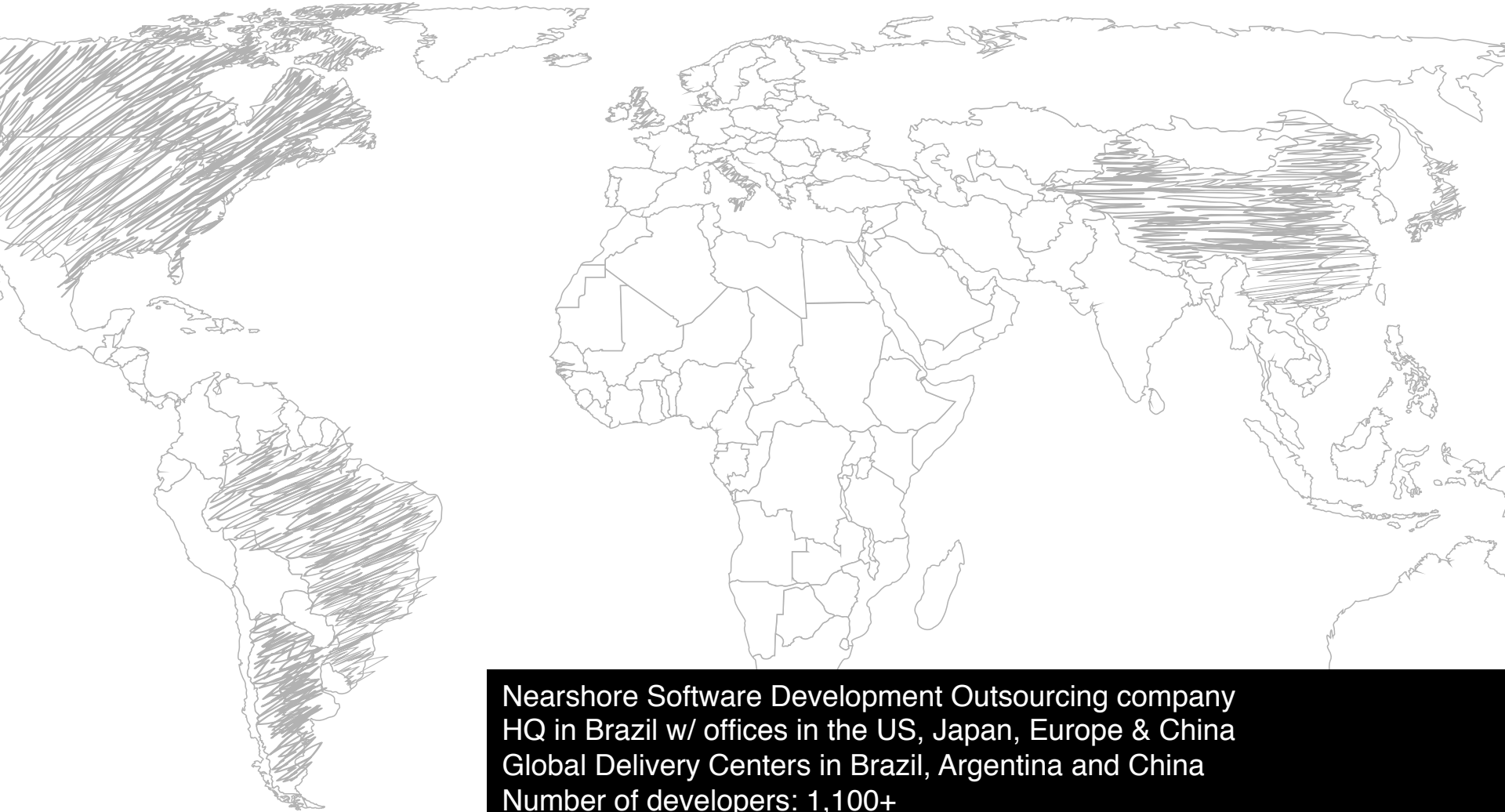








a little context



Nearshore Software Development Outsourcing company
HQ in Brazil w/ offices in the US, Japan, Europe & China
Global Delivery Centers in Brazil, Argentina and China
Number of developers: 1,100+
100% agile projects

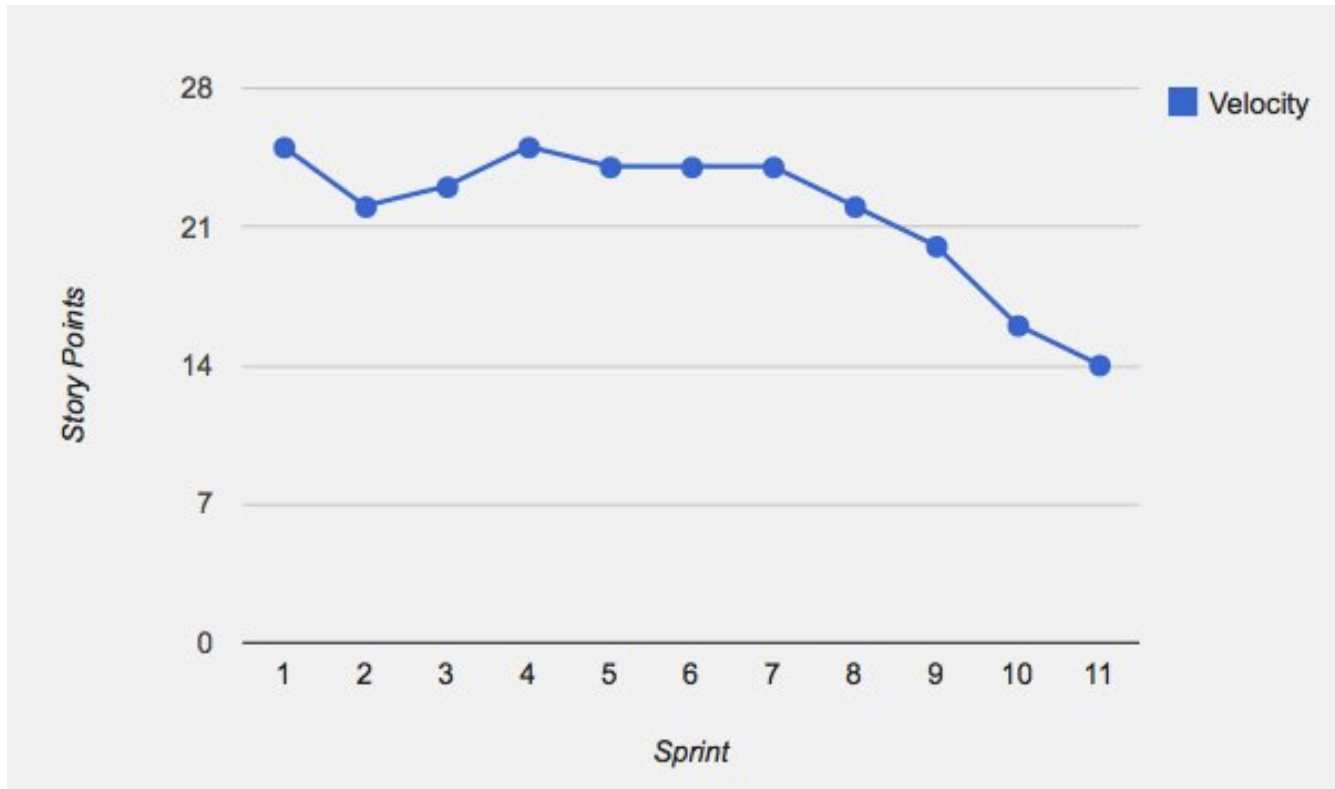
} What do we deal with

- Different platforms/technologies/frameworks
- New developers (different levels)
- New customers
- Aggressive timelines
- One Production System across the organization

some sad stories

(and fortunately other very happy ones!)

} do you remember this chart?

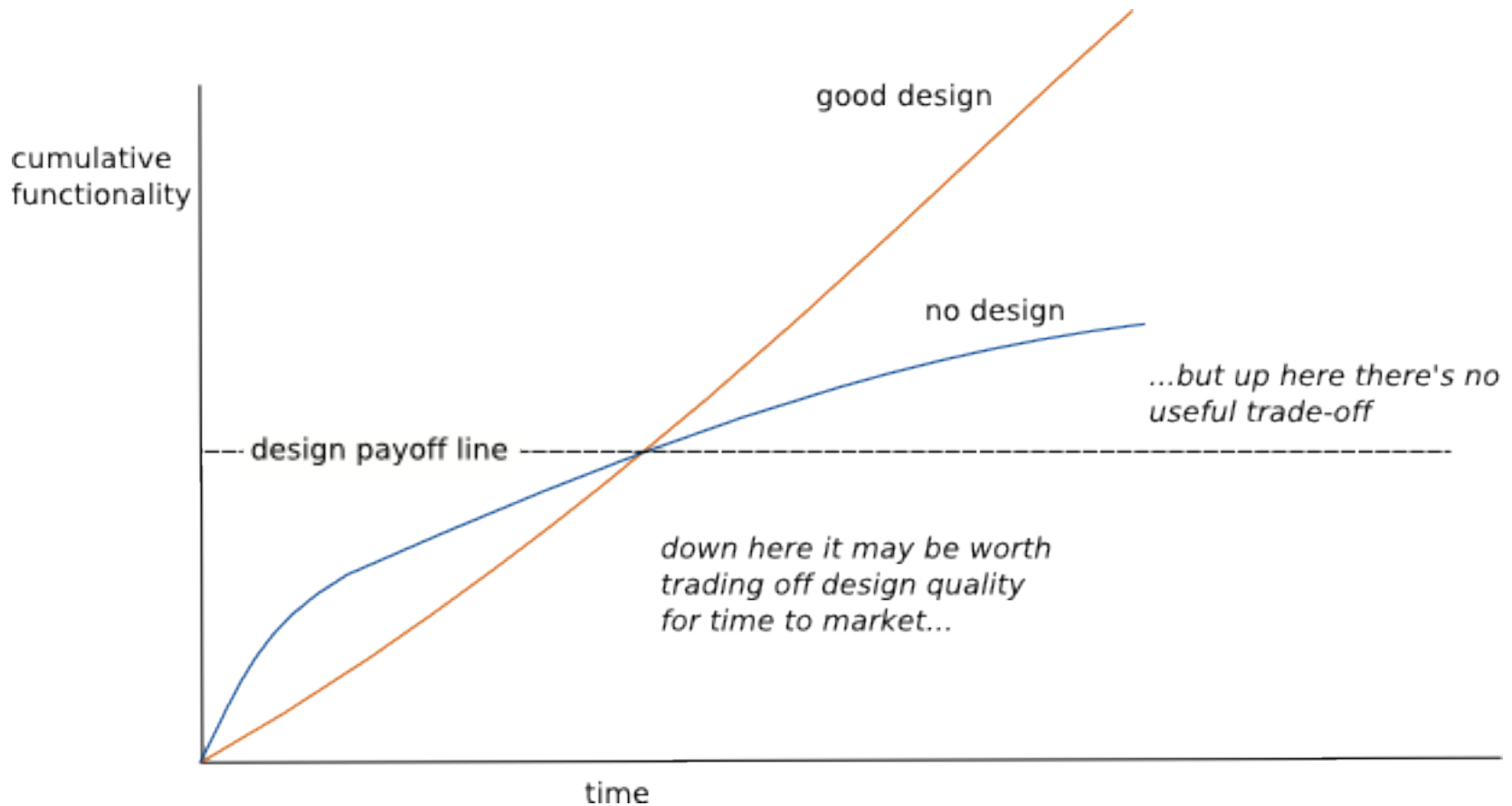


what the hell is happening with this project!?

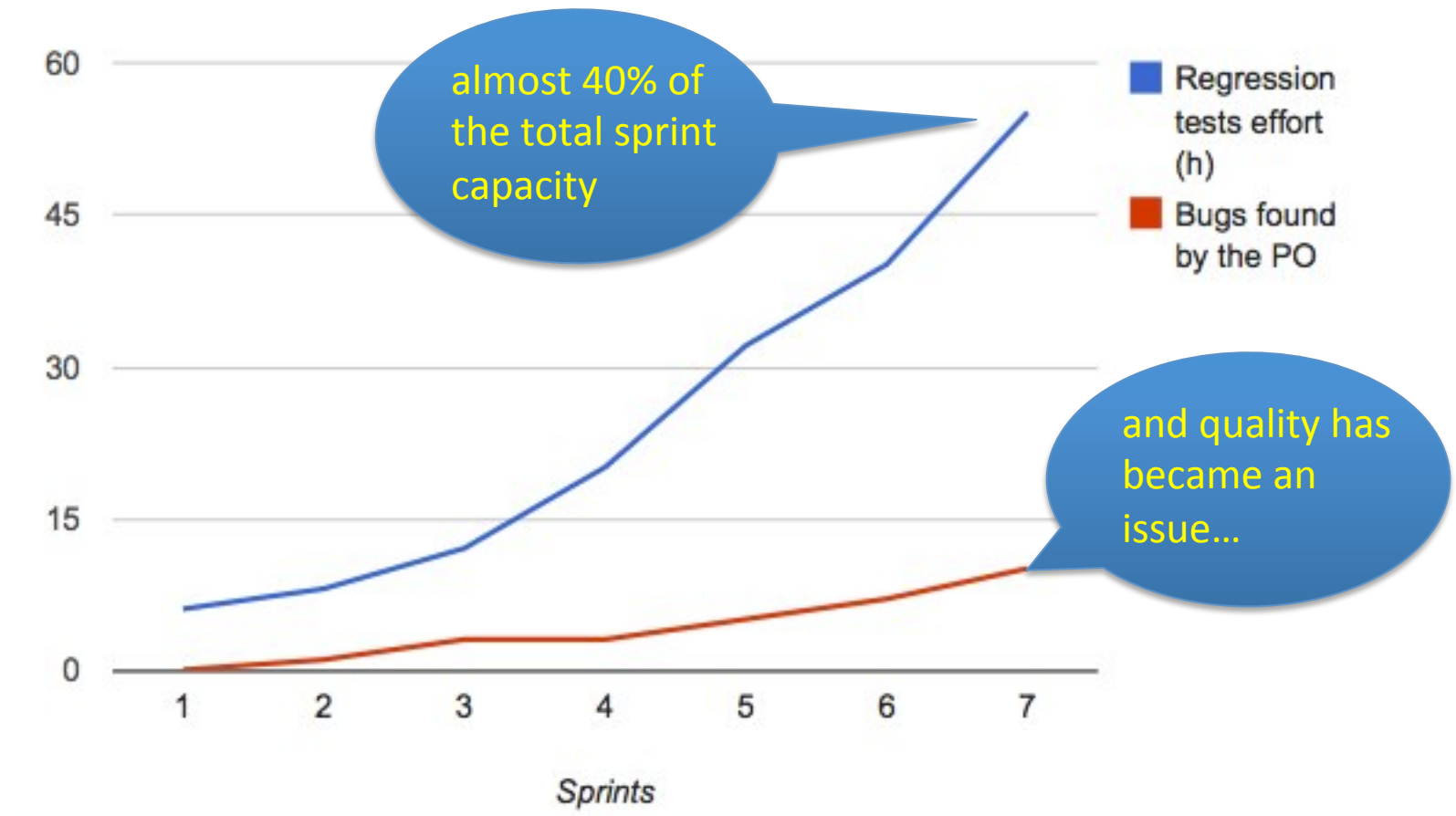
it is a true project... 😞

- Aggressive timeline to ship the product
 - prudent technical debt (“let’s fix it later!”)
 - yes! We met the milestone! 😊
- But.... a new “critical” milestone was set...
 - reckless technical debt
 - things got more difficult (extra effort, extra bugs, frustrated team, unhappy customer...)
 - two sprints spent on refactoring, almost no new functionality delivered...
 - we didn’t meet the second milestone... 😞
 - a long time to recover credibility.

Design Stamina Hypothesis (by Martin Fowler @ Agile Brazil 2010)



let's look another example...



what a hell is happening with this project?!?

lack of automated tests and continuous integration!

- complex billing system for an insurance company
 - several integrations with legacy systems
 - automated tests were not easy to be created (and kept working later!) → not prioritized
 - Product Owner wouldn't accept to spend sprint capacity with what he saw as “no value”
- after some time (and PO not so happy) team was able to create a minimal test suite (one entire sprint spent on that!)
 - keep this testing code up and running had become part of the definition of done (as it should have been since day 0)
 - in the following sprint, regression tests effort dropped half as well as the bugs found by the PO! 😊

a good reference now!

- a large construction company
- 18 months project
- High business complexity
- company board as the main stakeholders (even PO)
- Ci&T team: 12 people
- monthly shipping

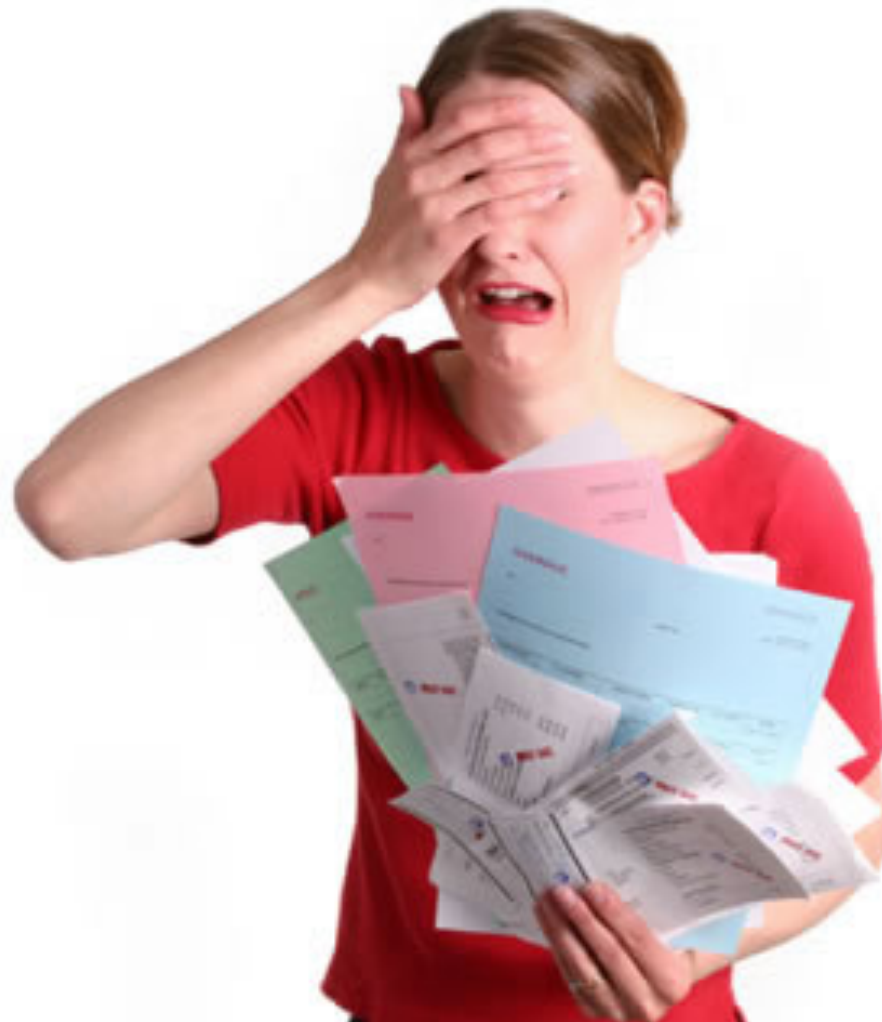


technical debt under control!

- code standards / design / code reviews / frequent refactoring
 - cost to add new features is almost the same since sprint 3
- continuous integration / automated tests
 - daily builds deployed on customer environment
 - completely automated deploy process

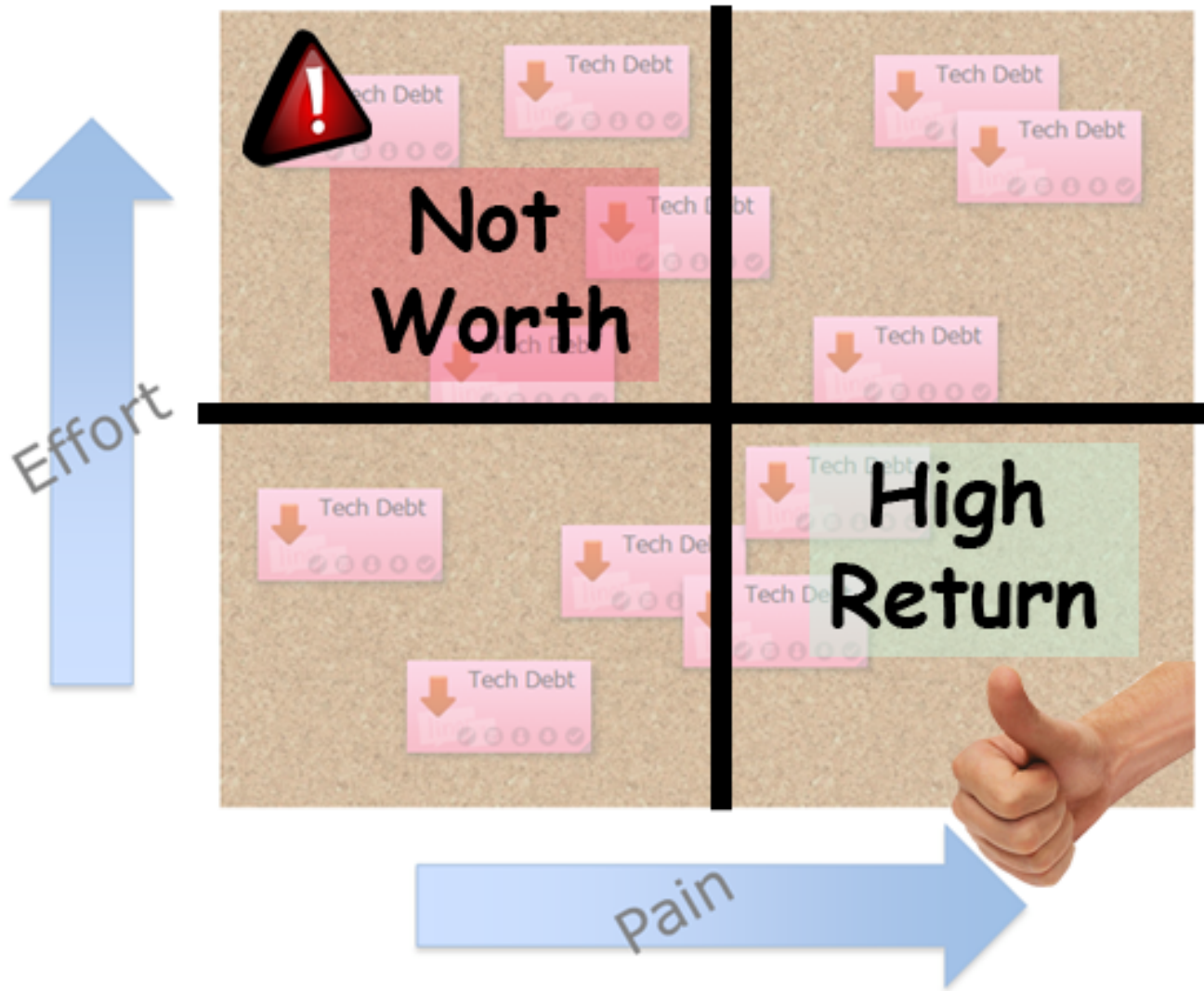
but how?!?

how to pay the debt?



OUR NATIONAL DEBT:
\$6,461,877,931.831.
YOUR *Family Share* \$70,544.
THE NATIONAL DEBT CLOCK

1 – register



2 – evaluate and prioritize



**3 – do not
accumulate**

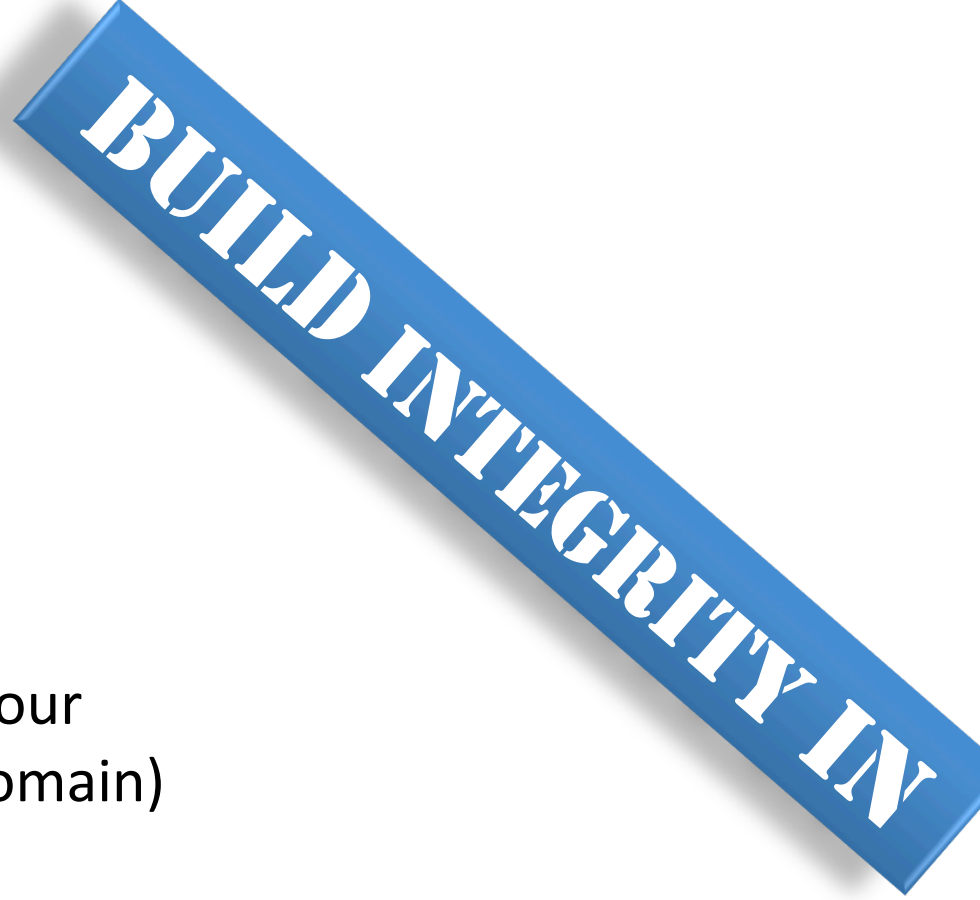
continuous integration

TDD

code reviews

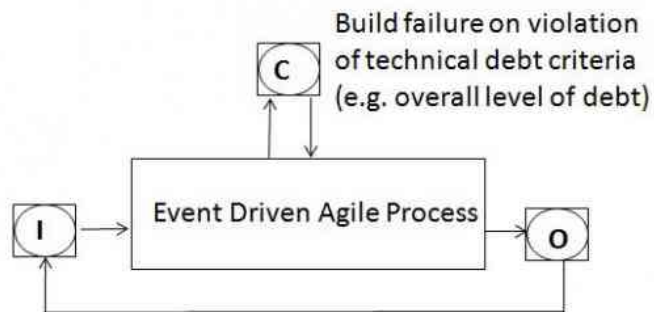
Refactoring (including reflecting your new knowledge about business domain)

Review your design decisions and refactor to reflect them



4 – pay

And then add it to your software development process as a technical debt reduction framework



Legend:

I=Input=(Requirements)

C=Control Unit= ('Stop the line' & convene a team meeting)

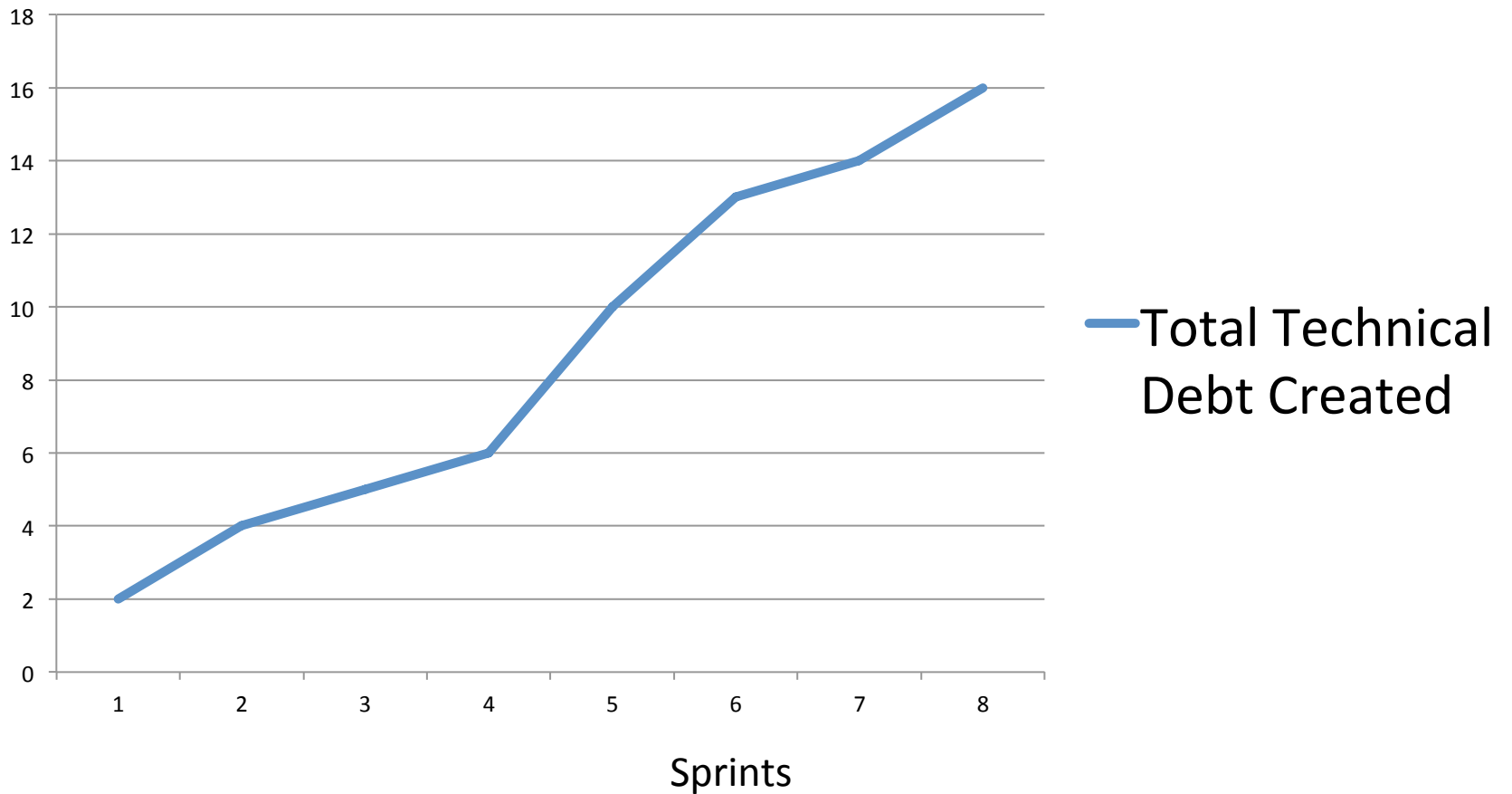
O=Output=(Code Increment in the build)

e.g.: part of
Continuous Integration,
Definition of done,
Daily reviews
Retrospectives,
...

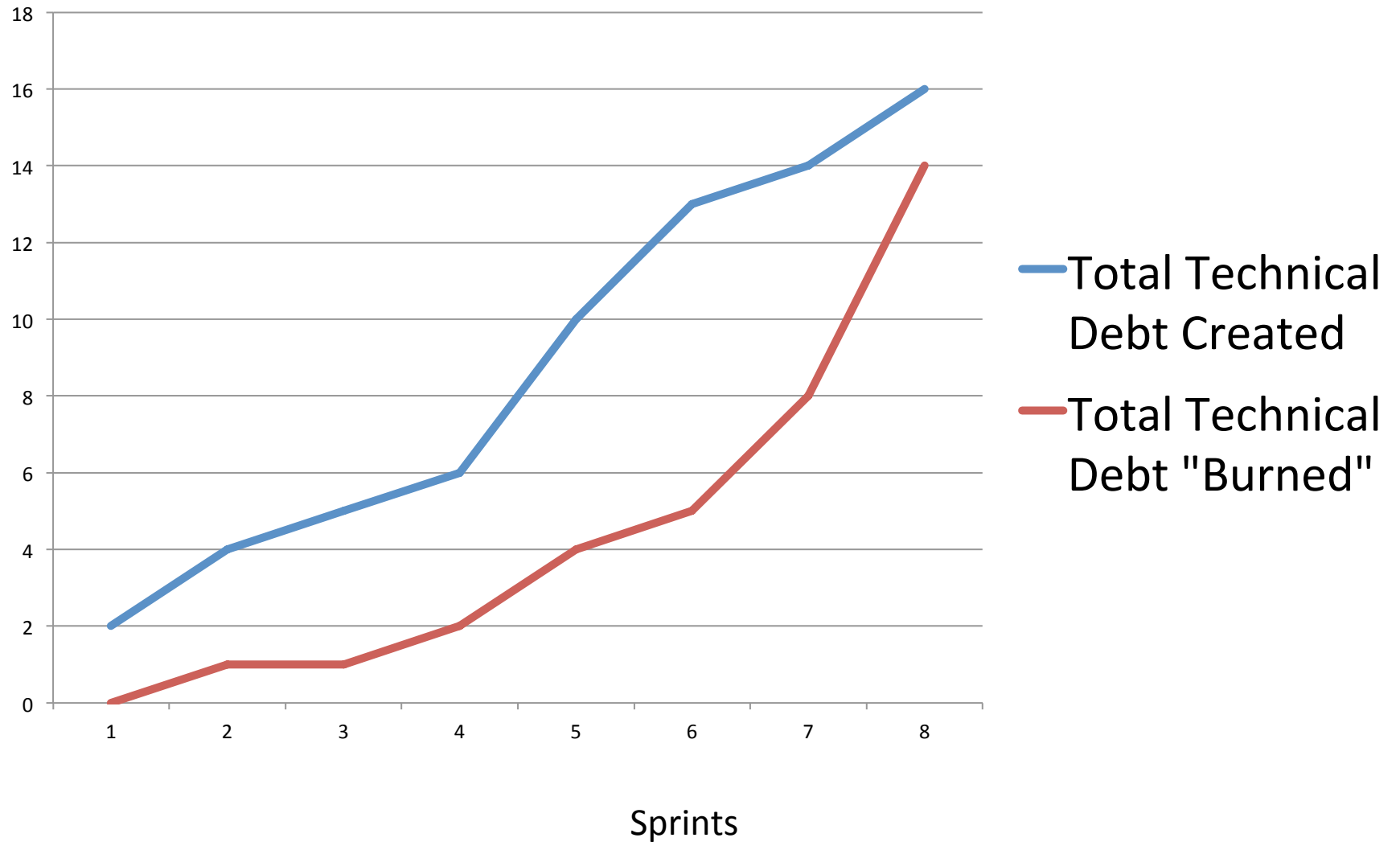
© Copyright 2010 Israel Gat

Measure it

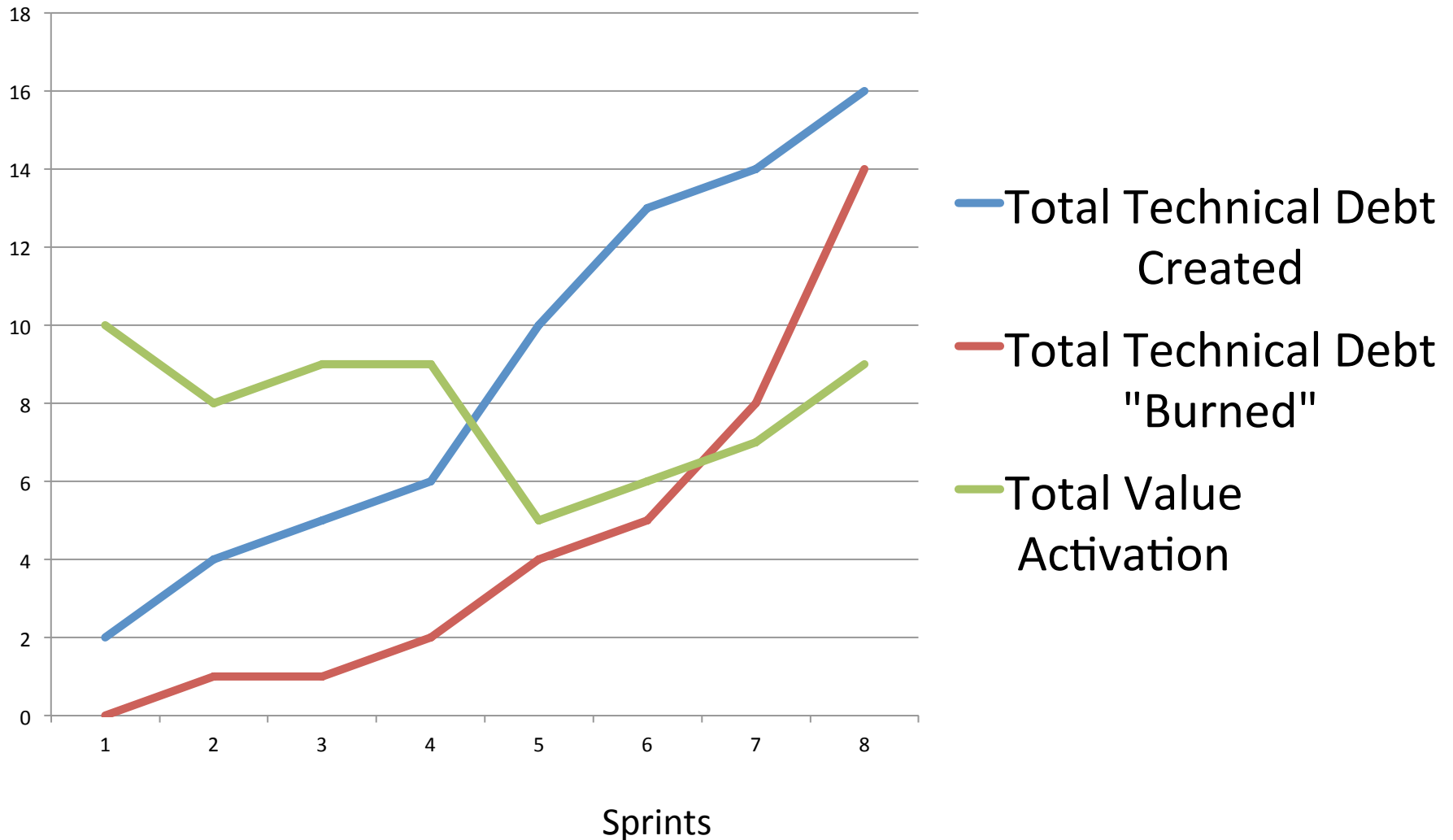
Total Technical Debt Created



Measure it



Measure and try and correlate it

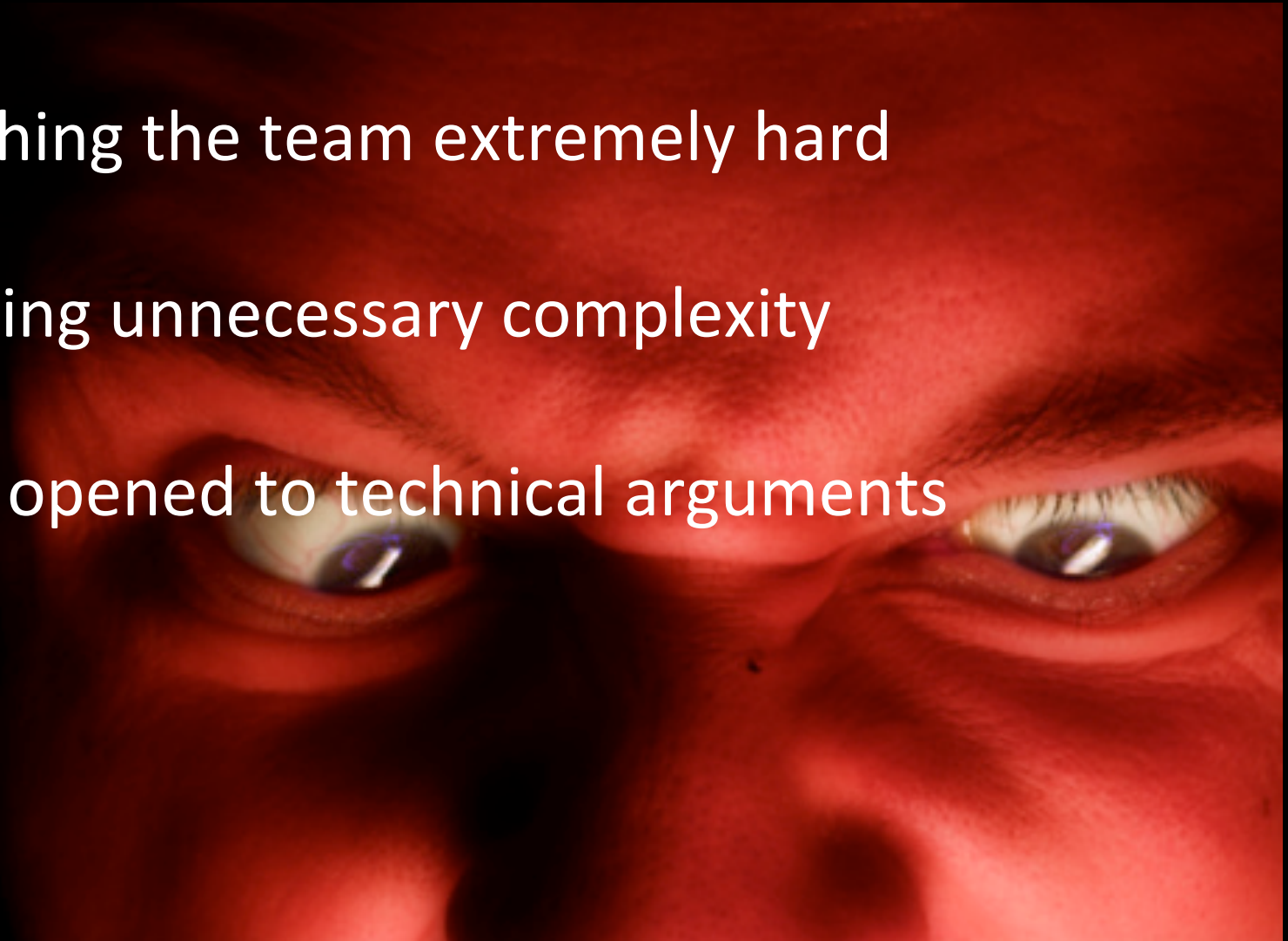


and how about the product
owner?

(because it seems he is somehow
important, doesn't it?)

you may see him as the bad guy...

- pushing the team extremely hard
- adding unnecessary complexity
- not opened to technical arguments



but he is the one who will have to pay
for the debt eventually!



it is all about communication and transparency

- The product owner – usually - does not know what is refactoring, code review, TDD, etc
- He/she needs to know what is the size of the debt and make conscious decisions about when and how to pay it

Technical Debt – To summarize:

- It's expensive to fix, but much more expensive to ignore
- Stick it to everyone's mind, from developers, managers to P.Os. The metaphor is fantastic to any level in the organization
 - Establish a technical debt reduction framework – Measure, correlate and act

Thanks!

Felipe Rubim

 frubim@ciandt.com

 frubim



references

Sites

- <http://www.controlchaos.com/>
- <http://www.mountangoatsoftware.com/scrum>
- <http://jeffsutherland.com/scrum/>
- <http://www.scrumalliance.org/articles>
- <http://www.agilechronicles.com/>

Books

- Agile Project Management with Scrum - by Ken Schwaber
- Lean Software Development: An Agile Toolkit - By Mary Poppendieck, Tom Poppendieck
- Agile and Iterative Development: A Manager's Guide - By Craig Larman
- Agile Software Development - by Alistair Cockburn

Articles

- CMMI® or Agile: Why Not Embrace Both! – by Hillel Glazer, Jeff Dalton, David Anderson, Mike Konrad and Sandy Shrum
- Practical Roadmap To Great Scrum - Jeff Sutherland, Ph.D., October 20, 2009
- Scrum and CMMI - Going from Good to Great, Carsten Ruseng Jakobsen, Jeff Sutherland, Ph.D.