

# Technical Debt

How not to ignore it

Agile 2008  
Toronto



Henrik Kniberg - Crisp AB

Agile coach & Java guy

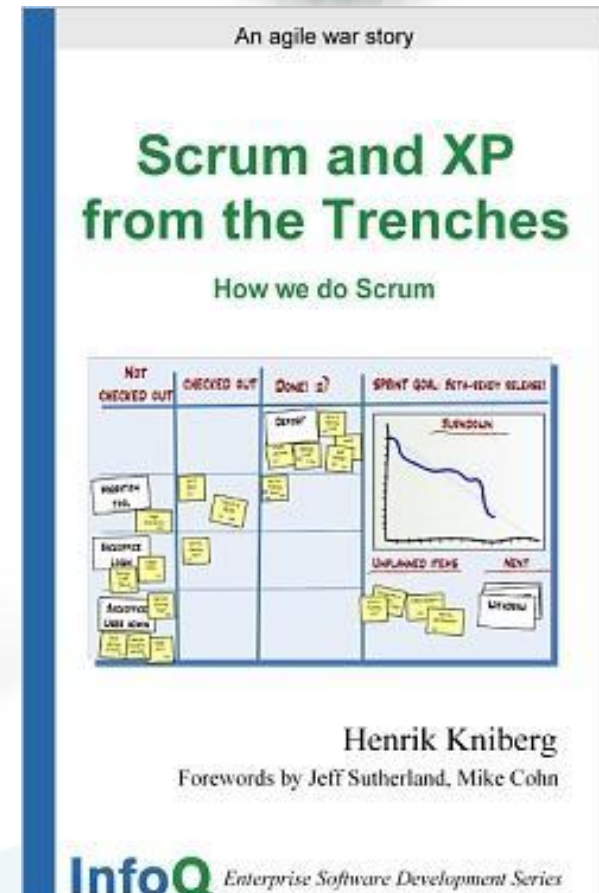
Cofounder / CTO of Goyada (mobile services)  
30 developers

Lead architect at Ace Interactive (gaming)  
20 developers

Chief of development at Tain (gaming)  
40 developers



henrik.kniberg@crisp.se  
+46 70 4925284

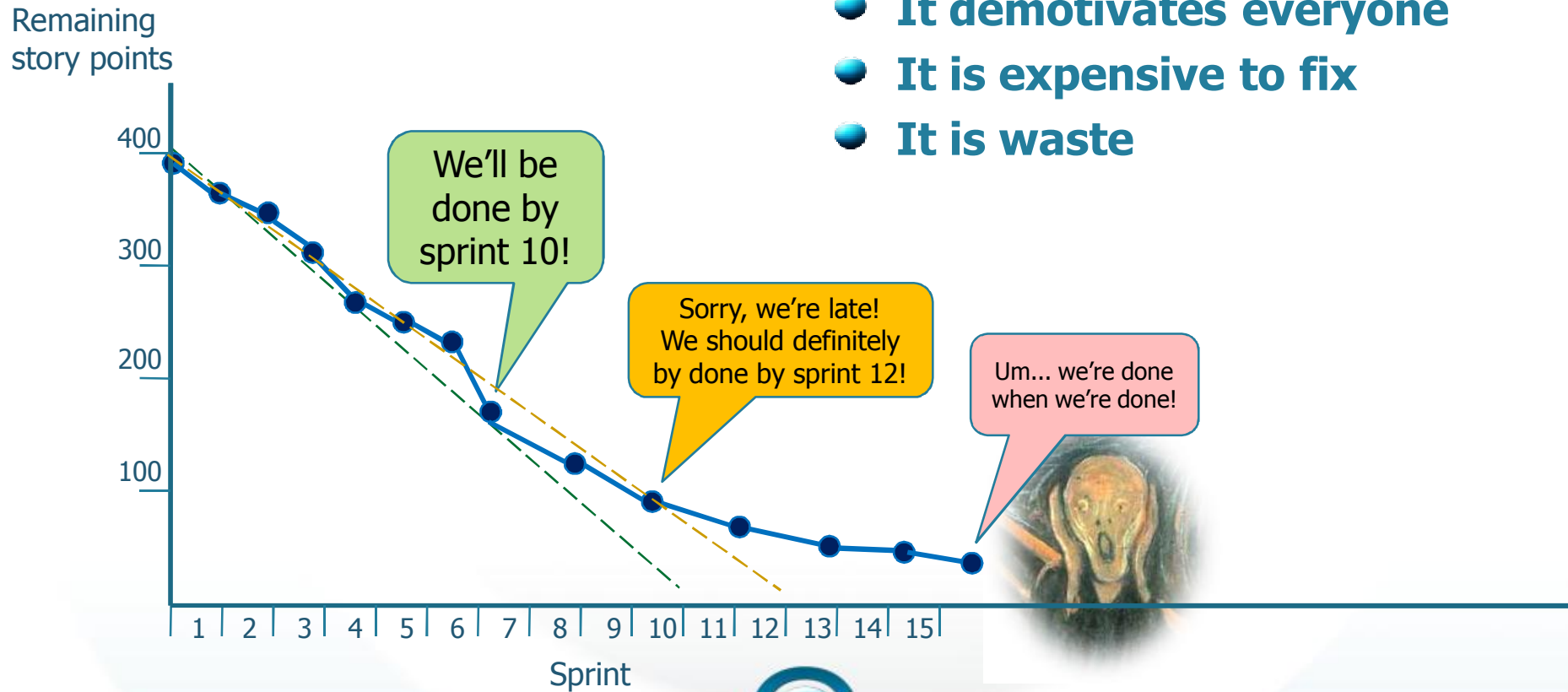


# What is technical debt?

- **Anything that slows down development**
- **For example breaking Kent Beck's rules for Simple Code**
  1. Runs all tests correctly
  2. Contains no duplication
  3. Expresses all the ideas that we had about the program
  4. Minimizes the number of classes and methods
- **More examples:**
  - Not fixing the broken build server
  - Inefficient manual release routines

# Why is it a problem?

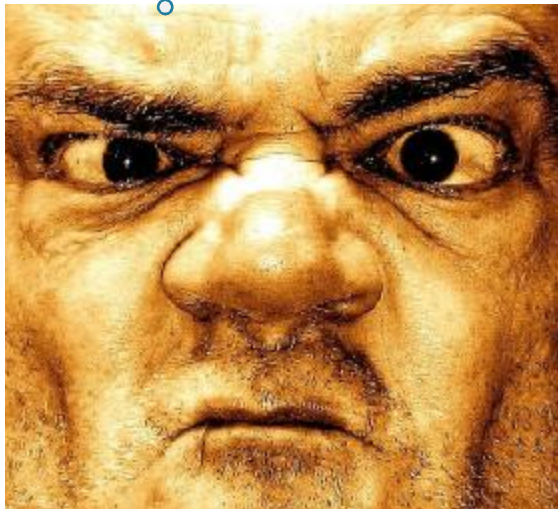
- Almost everyone has it
- It kills release plans
- It kills products
- It kills companies
- It demotivates everyone
- It is expensive to fix
- It is waste



# How can you detect it?

## Customer's gut feel

Feature delivery used to be a lot faster!



Henrik Kniberg

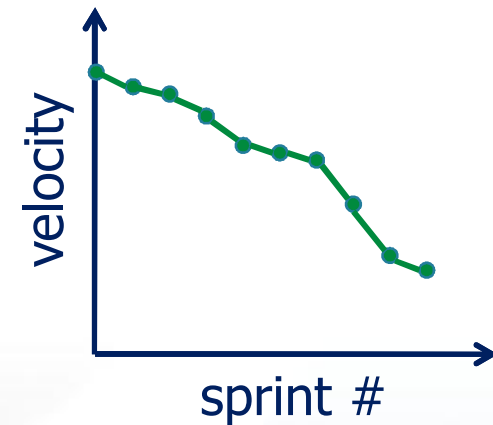
## Developer's gut feel

This code is embarrassing



## Metrics

- Duplication
- Test coverage
- Length of methods & classes
- Velocity



# Why does it happen?

- **Short-term thinking**

"We'll have time to fix it after the release"

- **Pressure & lack of slack & unsustainable pace**

"Everything MUST be finished by X-mas. Just work harder."

- **Not talking about it**

"We don't like whiners here"

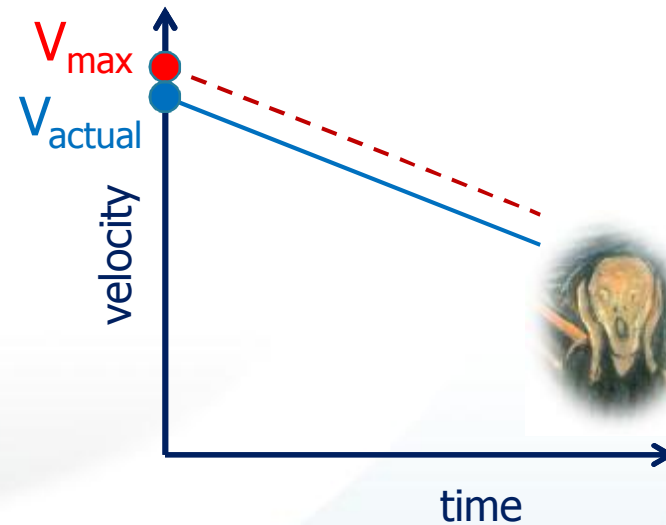
- **Lack of development skills**

"What's wrong with global variables?"

- **Broken window syndrome**

"Who cares, this codebase is crap anyway"

VELOCITY		
ESTIMATED		ACTUAL
<del>30</del>	40	35
<del>25</del>	35	30
<del>20</del>	30	25





# How can you do about it?



- **Option 1: Ignore**

- Let system decline into entropy death and hope that nobody needs it by then.

Risky!

Expensive!

How do you know you won't mess it up again?

- **Option 2: Rebuild**

- Rebuild system from scratch using TDD and other debt-minimizing practices.

Lack of learning.

Not fixing the process

- **Option 3: Throw it over the wall**

- Separate dedicated team fixes the code.

- **Option 4: Incremental improvement**

Preferred approach

- Fix a little bit each sprint, while still delivering business value
- ... or dedicated "Refactoring sprints" once in a while



# Fix the process, not just the product!

- **Technical problems are symptoms of process problems**
- **When a defect slips through a release**
  - Fix the defect
  - Analyze the cause of the defect
  - Improve the process to reduce the risk that this type of defect occurs again.
- **Example: Production server crashes when more than 100 users logged in**

Question	Answer	Improvement
Why did the server crash?	SessionHandler uses a DB connection pool fixed size 100	Gracefully reject incoming requests when connection pool full
Why wasn't the bug detected before release?	No automated performance tests	Write automated performance tests
Why weren't the tests written from start?	Nobody knew how to write such tests	Give people more time to learn
Why didn't the team take the time to learn?	Too much pressure on team.	Reduce pressure by reducing scope of project. Introduce pull scheduling.

# Test automation backlog

Step 1: List your tests

- Change skin
- Security alert
- Transaction history
- Block account
- Add new user
- Sort query results
- Deposit cash
- Validate transfer

# Test automation backlog

Step 2: Classify each test

Test case	Risk	Manual Test Cost	Automation Cost
Change skin	low	0.5 hrs	20 sp
Security alert	high	1 hrs	13 sp
Transaction history	med	3 hrs	1 sp
Block account	high	5 hrs	0.5 sp
Add new user	low	0.5 hrs	3 sp
Sort query results	med	2 hrs	8 sp
Deposit cash	high	1.5 hrs	1 sp
Validate transfer	high	3 hrs	5 sp

# Test automation backlog

Step 3: Choose your priorities & sort the list

For example prioritize by:

1. Manual test cost
2. Risk
3. Automation cost

Test case	Risk	Manual Test Cost	Automation Cost
Block account	high	5 hrs	0.5 sp
Validate transfer	high	3 hrs	5 sp
Transaction history	med	3 hrs	1 sp
Sort query results	med	2 hrs	8 sp
Deposit cash	high	1.5 hrs	1 sp
Security alert	high	1 hr	13 sp
Add new user	low	0.5 hrs	3 sp
Change skin	low	0.5 hrs	20 sp

# Test automation backlog

Step 4: Automate a few tests each sprint

Examples:

- "Each sprint we will implement one test automation story"
- "Each sprint we will implement up to 10 story points of test automation stories"
- "Each sprint we will spend about 10% of our time implementing test automation stories"
- "Each sprint we will finish the product backlog stories first, and then spend the remainder of the time (if any) implementing test automation stories"
- "The product owner will merge the test automation stories into the overall product backlog, and the team will treat them just like any other story."

Test case	Risk	Manual Test Cost	Automation Cost
Block account	high	5 hrs	0.5 sp
Validate transfer	high	3 hrs	5 sp
Transaction history	med	3 hrs	1 sp
Sort query results	med	2 hrs	8 sp
Deposit cash	high	1.5 hrs	1 sp
Security alert	high	1 hr	13 sp
Add new user	low	0.5 hrs	3 sp
Change skin	low	0.5 hrs	20 sp

# Take-away points

- **Talk about it!**
  - "Technical Debt" is a great metaphor when communicating with non-techies!
- **Stop accumulating it!**
  - No matter how bad it already is.
- **Fix it incrementally!**
  - Decide on a viable repayment pace
  - Avoid big-bang approaches or "throw it over the wall"
- **Fix the process, not just the product!**

**That's it! Questions? Discussion?**

