What's hard about being an agile developer?

JAOO, Aarhus, Denmark
2008-10-01

Henrik Kniberg - Crisp AB

Agile coach & Java guy

Cofounder / CTO of Goyada (mobile services)
30 developers

Lead architect at Ace Interactive (gaming)
20 developers

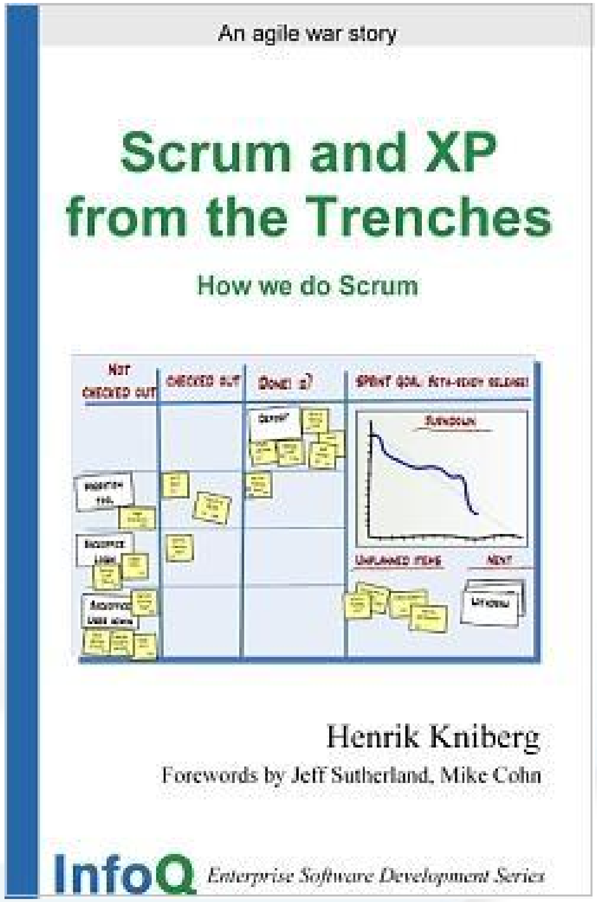
Chief of development at Tain (gaming)
40 developers

Agile coach at various companies

CERTIFIED Scrum Trainer

henrik.kniberg@crisp.se
+46 70 4925284

crisp



An agile war story

Scrum and XP from the Trenches

How we do Scrum

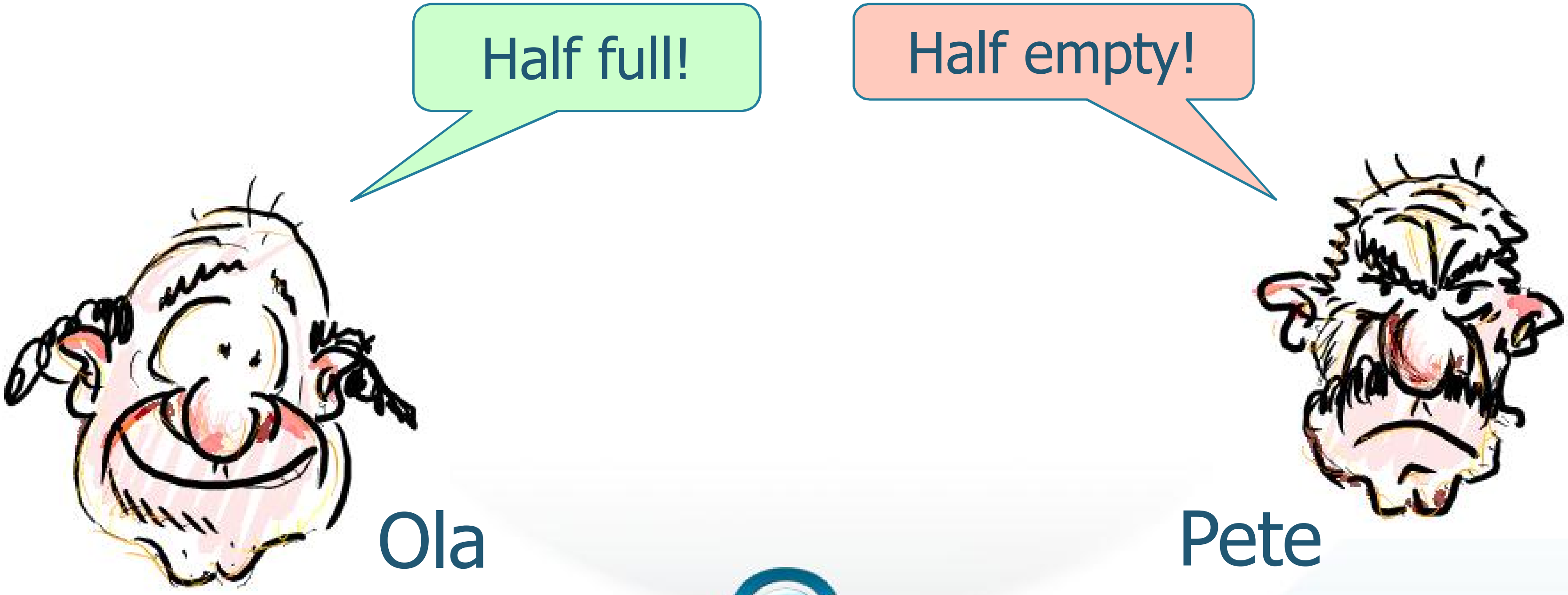
Henrik Kniberg

Forewords by Jeff Sutherland, Mike Cohn

InfoQ Enterprise Software Development Series

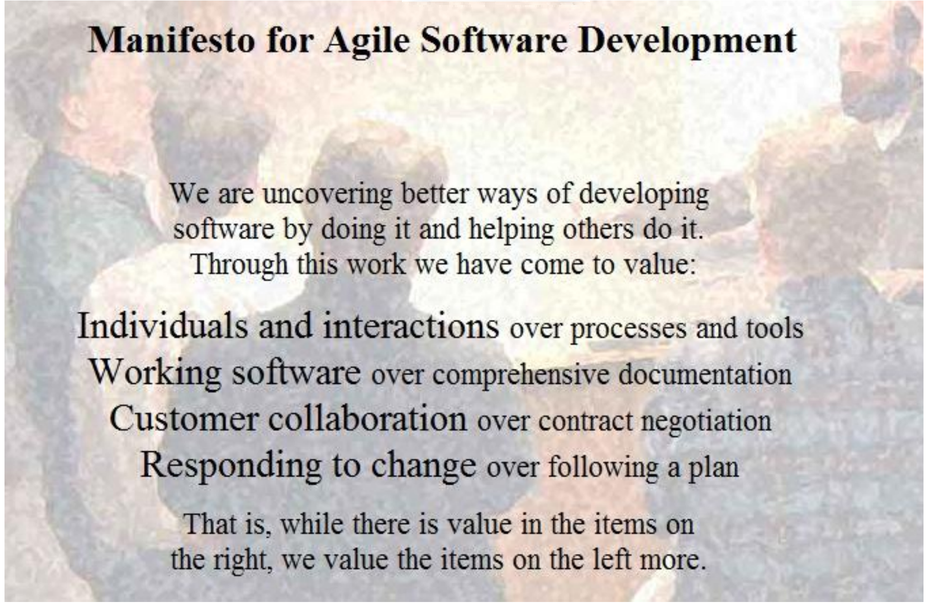
Optimist or pessimist?

```
String[] goodies = new String[4];
goodies[0] = ”apple”;
goodies[1] = ”banana”;
```



Being an agile developer can be hard because you have to:

Think about how you work



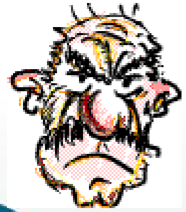
Manifesto for Agile Software Development

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.

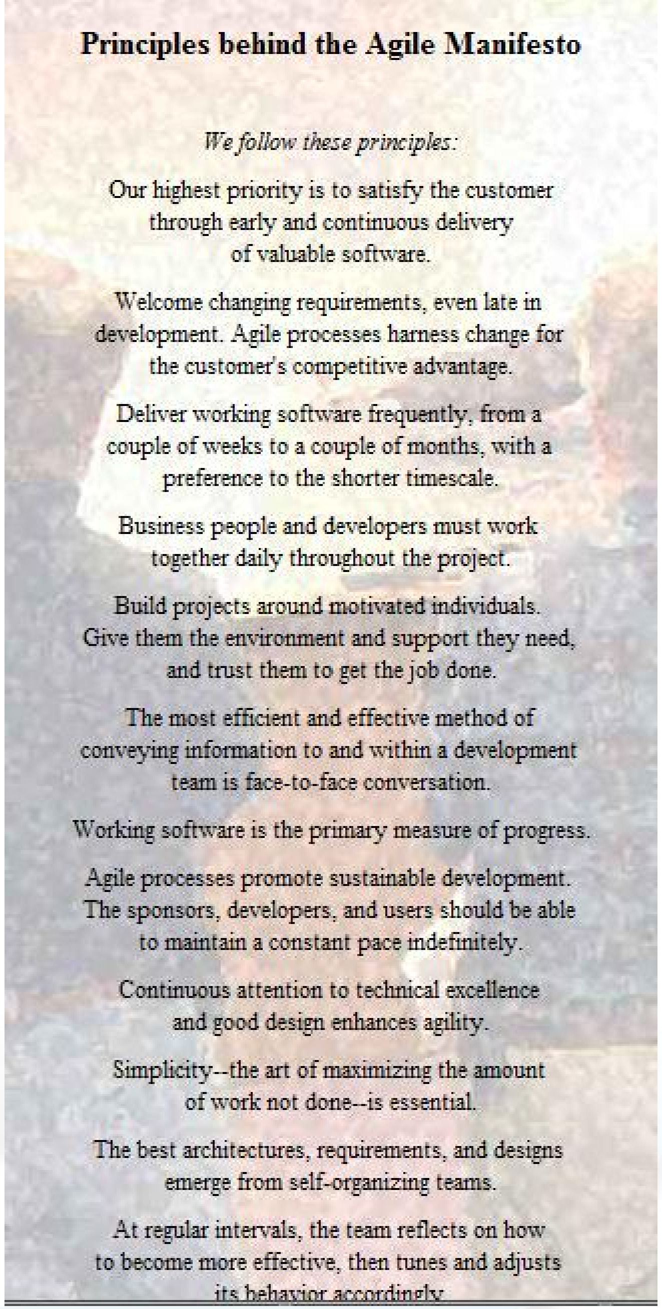
Agile is simple!



Too vague

Well go read some books then!

Henrik Kniberg



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

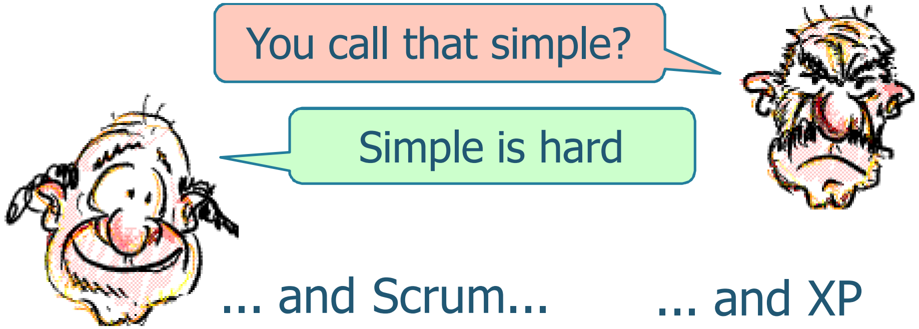
Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile books

A few books on Agile...



... and Scrum...

... and XP

... and maybe some on DSDM and Crystal and Lean while you're at it.

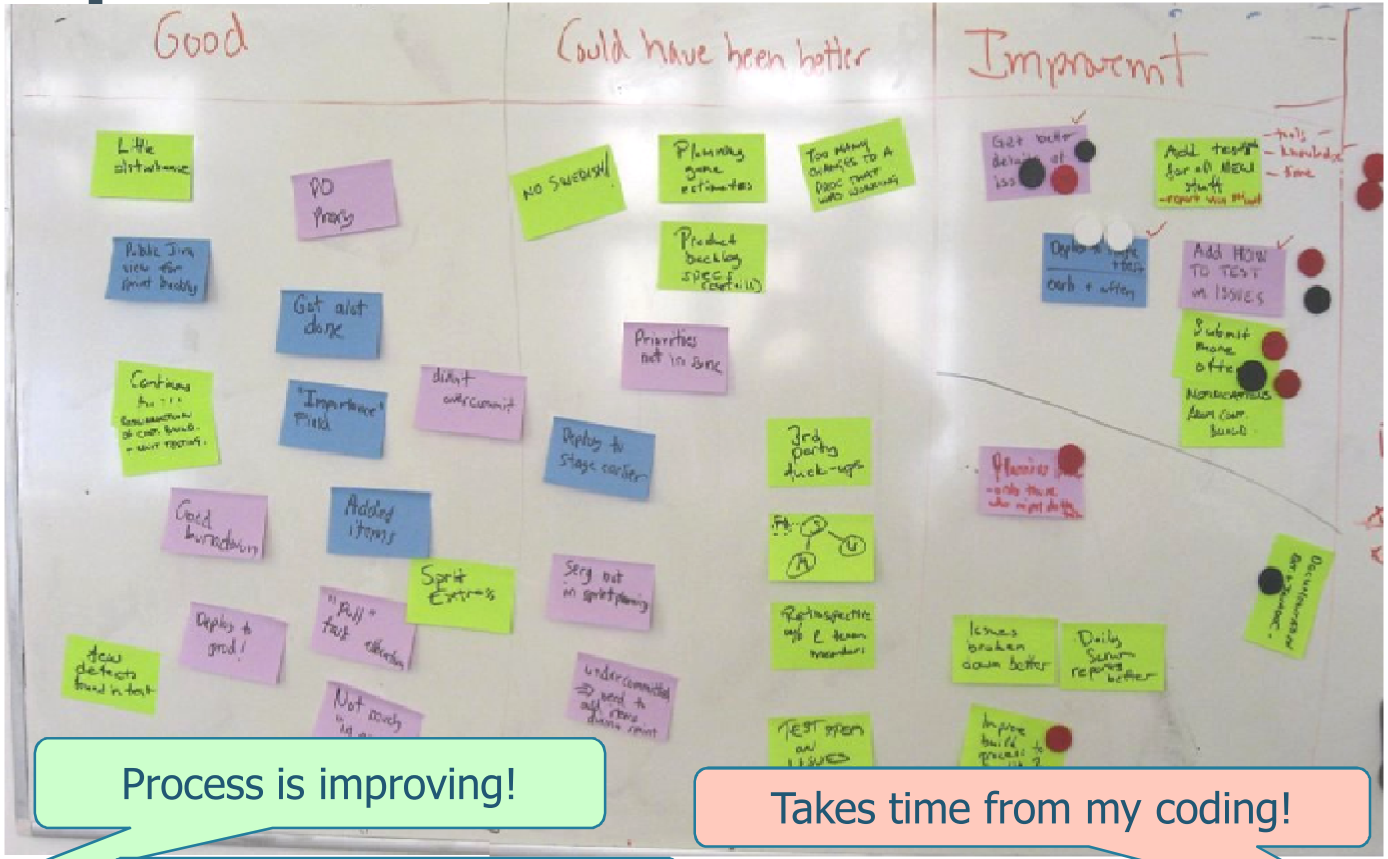
Agile Development
Agile & Iterative Development: A Manager's Guide
Craig Larman
Agile Java
Agile Software Development
Agile Project Management
Agile Documentation
Agile Estimating and Planning
User Stories Applied
Agile Software Development: Principles, Patterns, and Practices
Planning and Budgeting for the Agile Enterprise
Agile Principles, Patterns, and Practices in C#
Agile Patterns
agile modeling
Scott W. Ambler
Building the Agile Enterprise
Agile Database Techniques
Agile Management for Software Engineering

The Enterprise and Scrum
Agile Software Development with Scrum
Ken Schwaber
Scrum and XP from the Trenches
Henrik Kniberg
InfoQ
Scrum
SCUM Manifesto
Valerie Solanas

Extreme Programming Installed
eXtreme Programming in Action
Planning Extreme Programming
Extreme Programming Pocket Guide
Extreme Programming and Agile Methods – XP/Agile Universe 2003
Testing Extreme Programming
Extreme Programming for Web Projects
XP Refactored: The Case Against Extreme Programming
Sams Teach Yourself Extreme Programming in 24 Hours
Extreme Programming Explained
Pro .NET Extreme Programming
Java Extreme Programming Cookbook
Extreme Programming Applied

Henrik Kni

Retrospectives

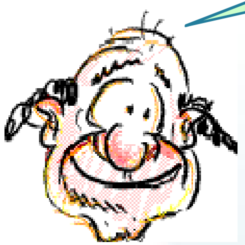


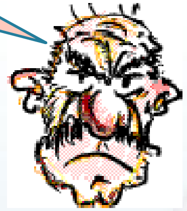
Process is improving!

Less & less waste!

Takes time from my coding!

Too touchy & feely.

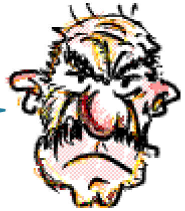




Being an agile developer can be hard because you have to:

Work with the customer

3: Customer collaboration over contract negotiation



I have to talk to the customer and learn his domain

... and follow HIS priorities

I have to demo stuff

Takes time from my coding!

I get to show my stuff to someone who cares!

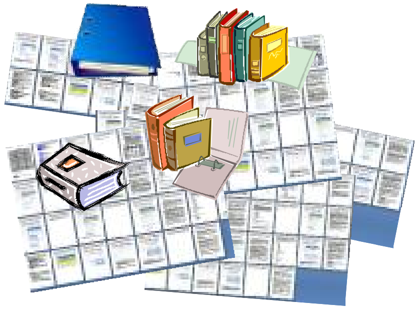
I get to propose improvements!

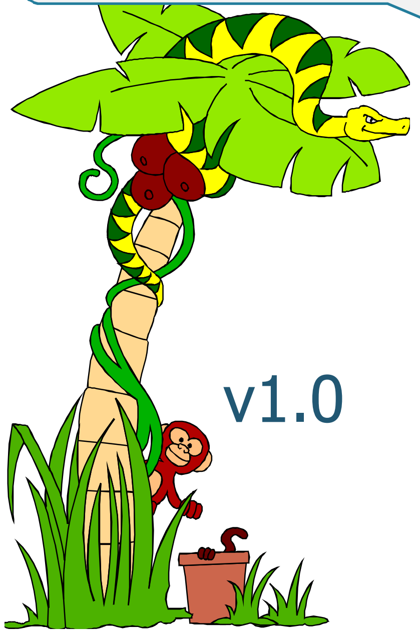
Being an agile developer can be hard because you have to:

2: Working code over comprehensive documentation

Deliver incrementally

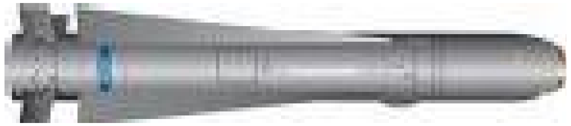
Big bang delivery



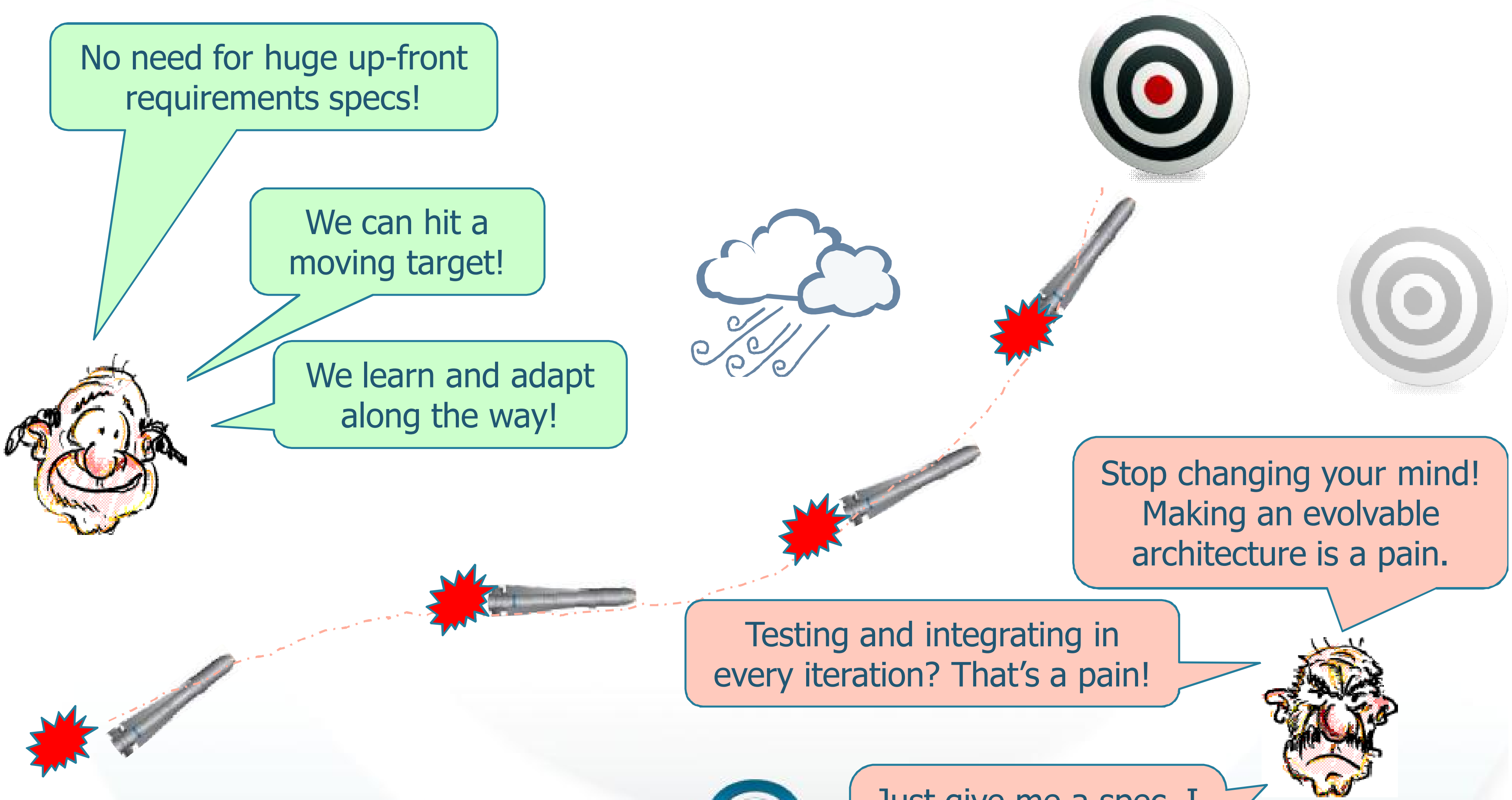


Incremental delivery

Waterfall is like a cannonball



Incremental is like a homing missile



No need for huge up-front requirements specs!

We can hit a moving target!

We learn and adapt along the way!

Stop changing your mind! Making an evolvable architecture is a pain.

Testing and integrating in every iteration? That's a pain!

Just give me a spec, I don't have time to waddle around

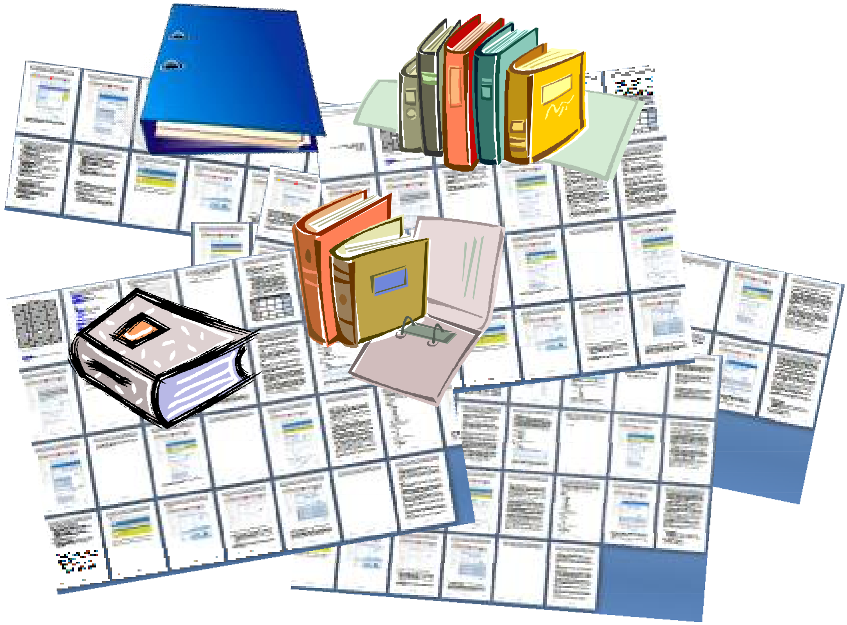
Being an agile developer can be hard because you have to:

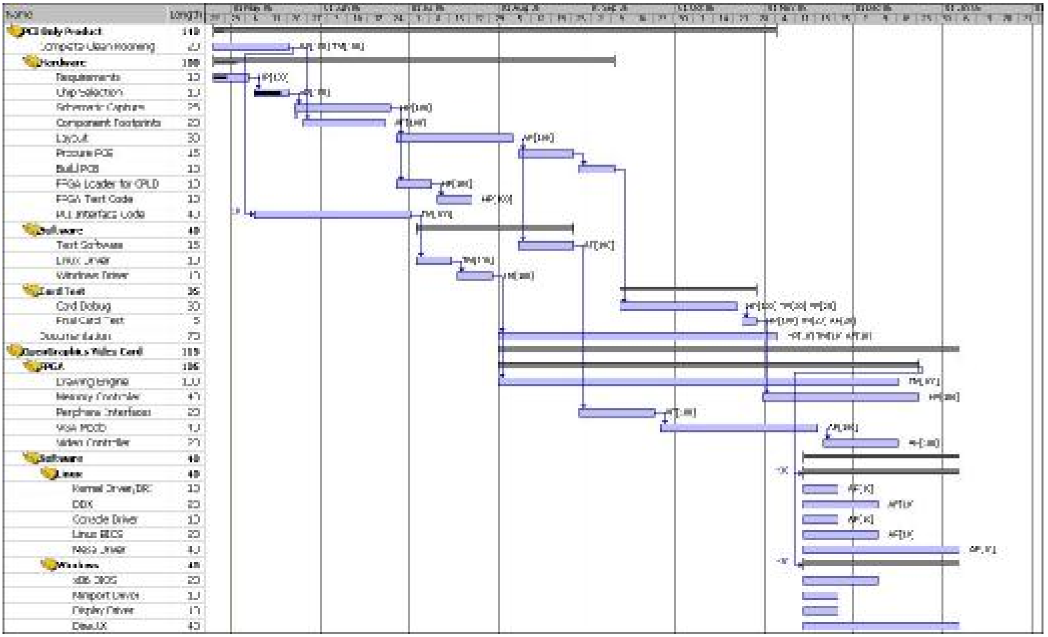
Plan continuously

... but never quite trust the plan

4: Responding to change over following a plan

Planning – the traditional way





Name	Length
PCI Only Product	110
Complete Clean Rooming	20
Hardware	100
Requirements	10
Chip Selection	10
Schematic Capture	25
Component Footprints	20
Layout	30
Procure PCB	15
Build PCB	10
FPGA Loader for CPLD	10
FPGA Test Code	10
PCI Interface Code	40
Software	40
Test Software	15
Linux Driver	10
Windows Driver	10
Card Test	35
Card Debug	30
Final Card Test	5
Documentation	70
OpenGraphics Video Card	115
FPGA	105
Drawing Engine	110
Memory Controller	40
Peripheral Interfaces	20
VGA Mode	10
Video Controller	20
Software	40
Linux	40
Kernel Driver/DRI	10
DDX	20
Console Driver	10
Linux BIOS	20
Mesa Driver	40
Windows	40
x86 BIOS	20
Miniport Driver	10
Display Driver	10
DirectX	40

Planning – the agile way

Simpler!

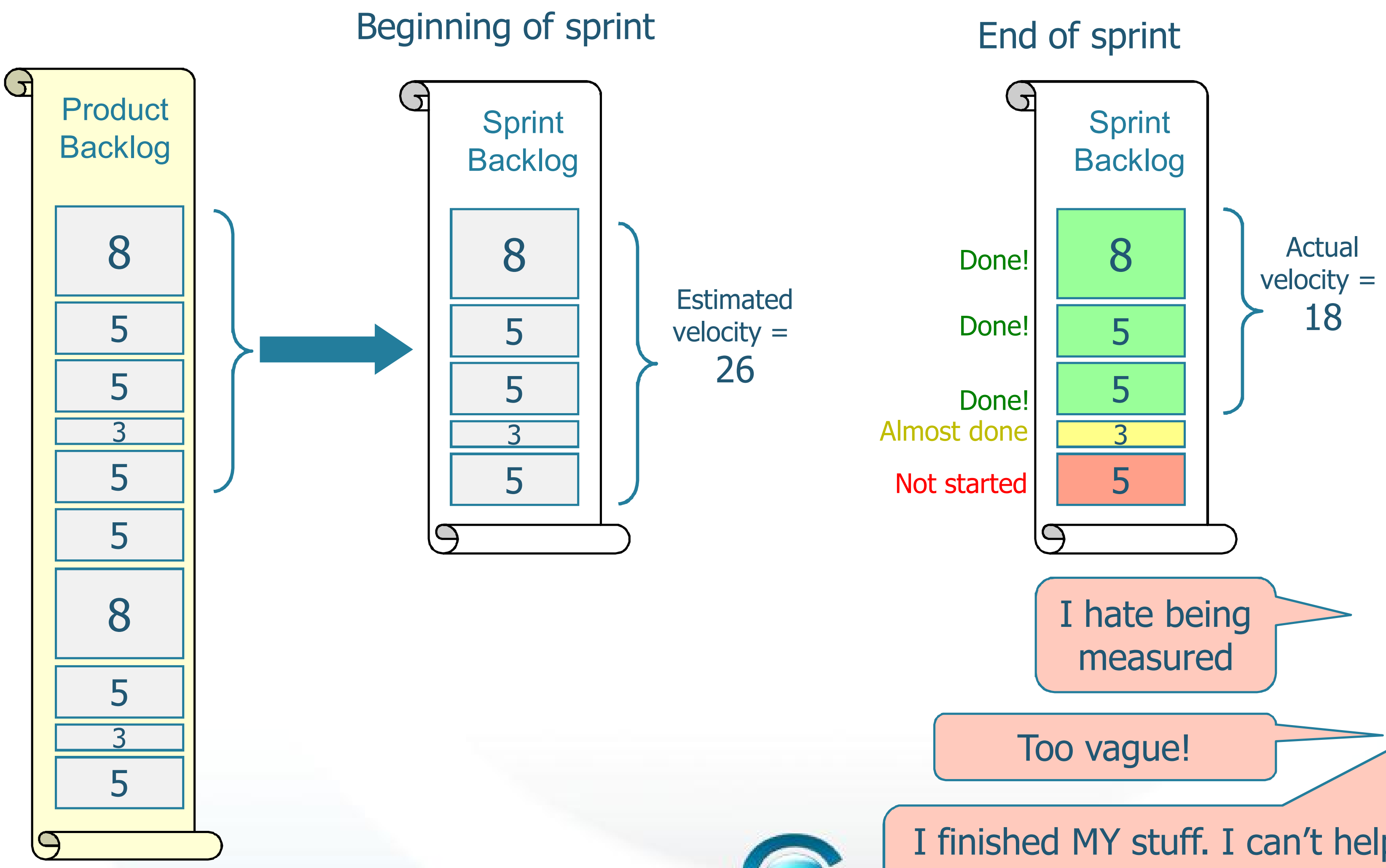
Flexible!

- ~~ADMINISTRATE USERS~~ ~~13~~
  - REGISTER NEW USER 5
    - Write failing test 4h
    - Create DB schema 2h
    - Write form validation 2h
    - Do GUI design 1h
    - Write server-side logic 8h
    - Do integration test 4h
  - FIND USER 3
  - EDIT EXISTING USER 2
  - DELETE USER 8
- VIEW INVOICE IN HTML, PDF, OR EXCEL FORMAT 3
- AS A HELPDESK OPERATOR I WANT TO SEE WHO IS LOGGED IN 8
- OPERATIONS MANUAL 5

Sloppier!

Takes time from my coding!

Measuring velocity



Beginning of sprint

Product Backlog

8
5
5
3
5
5
8
5
3
5

Sprint Backlog

8
5
5
3
5

Estimated velocity = 26

End of sprint

Sprint Backlog

Done! 8

Done! 5

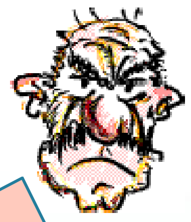
Done! 5

Almost done 3

Not started 5

Actual velocity = 18

Realistic measure of progress!



I hate being measured

Too vague!

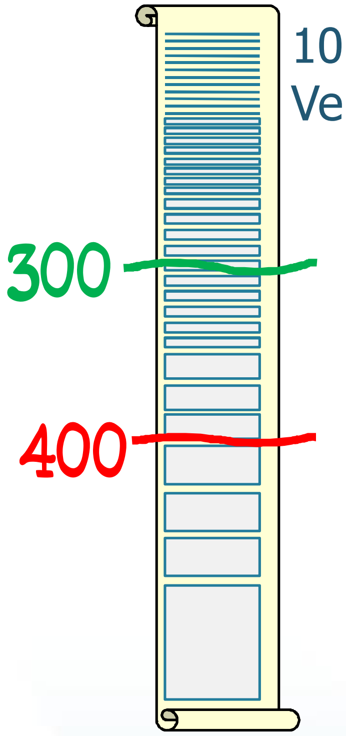
I finished MY stuff. I can’t help that the &%@# tester didn’t finish HER part

Release planning & followup

We can see if we are on track!

Fixed date

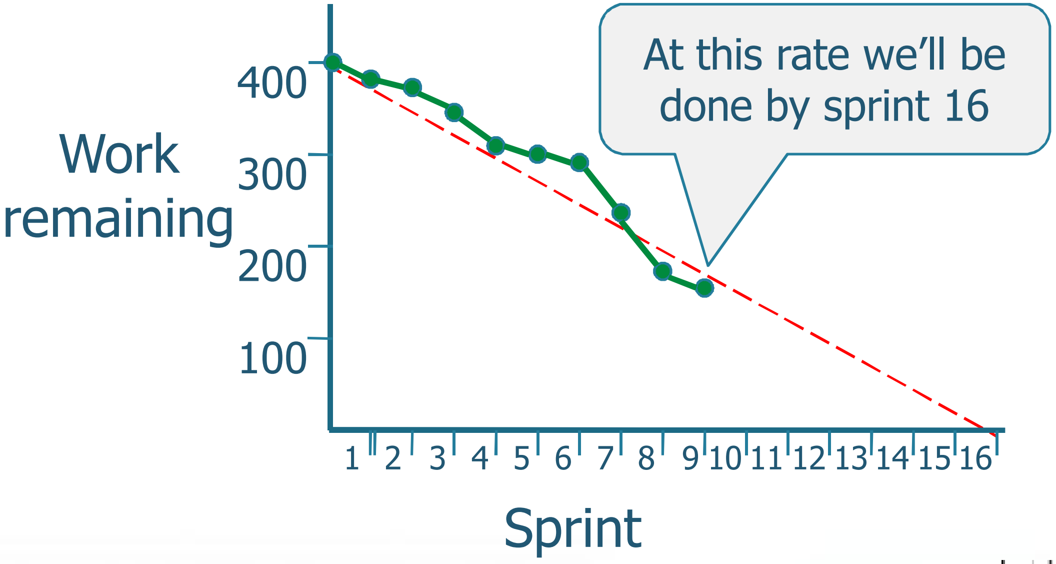
"What will be done by X-mas?"



10 sprints from now
Velocity = 30-40

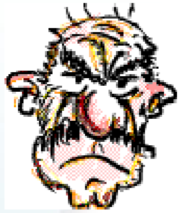
Fixed scope

"When will everything be done?"



At this rate we'll be done by sprint 16

Stressful



Common planning & followup meetings

- Story creation meeting
- Story estimation meeting
- Sprint planning meeting
- Daily Scrum
- Sprint review/demo
- Sprint retrospective

Meetings are boring!

Only if they are ineffective.

We should code instead.

We don’t need *lots* of code, we just need the *right* code. The meetings help us figure out what to build and what not.

The meetings take 5-10% of our time. That gives us 90% time to focus on coding!

Being an agile developer can be hard because you have to:

Write good code

5: Clean code over crap

Dog.java v0

Dog.java v1.0
Quick & dirty

```
public class Dog {
   public static void main(String[] args) {
      System.out.println("WOOF 1!");
      System.out.println("WOOF 2!");
   }
}
```

Dog.java v1.1
Big & hairy

```
import java.sql.Connection;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class AddressDb {
private Executor executor = Executors.newFixedThreadPool(18);
private int CACHE_SIZE = 50;

public AddressDb()
{
try
{
     Class.forName("oracle.jdbc.ThinDriver");
     connection = DriverManager.getConnection("jdbc:oracle:thin:@prod", "admin",
"beefhead");
     statement = connection.prepareStatement("insert into AddressEntry values (?, ?, ?)");
     } catch (ClassNotFoundException e) {}

new Thread().start();
}

public void addPerson(Person person) {
Connection connection = null;
PreparedStatement statement = null;
try {
     connection = DriverManager.getConnection("jdbc:oracle:thin:@prod", "admin",
"beefhead");
     statement = connection.prepareStatement("insert into AddressEntry values (?, ?, ?)");
     statement.setLong(1, System.currentTimeMillis());
     statement.setString(2, person.getName());
     statement.setString(3, person.getPhoneNumber().getNumber());
     statement.executeUpdate();
     }
     }
}
} Connection a = DriverManager.getConnection("jdbc:oracle:thin:@prod", "admin", "beefhead");
b =   a.prepareStatement("select * from AddressEntry where name = '" + name + "'");
  c = b.executeQuery();
if (c.next()) {
String foundName = c.getString("name");
     PhoneNumber phoneNumber = new PhoneNumber(c.getString("phoneNumber"));
     Person person = new Person(foundName, phoneNumber);
return person;
} else {
     return new Person("", null);
}

} catch (SQLException e) {
     return null;
} catch (IllegalArgumentException x) {
     throw x;
     }
}

lic List<Person> getAll() {
     connection = DriverManager.getConnection("jdbc:oracle:thin:@prod", "admin",
efhead");
     statement = connection.prepareStatement("insert into AddressEntry values (?, ?, ?)");
     statement.setLong(1, System.currentTimeMillis());

(statement != null) {
(c.next()) {
ing foundName = c.getString("name");
     PhoneNumber phoneNumber = new PhoneNumber(c.getString("phoneNumber"));
     Person person = new Person(foundName, phoneNumber);
turn person;
} else {
```

~~Code is an asset~~
All code is cost!
Some code is value.

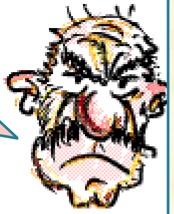
Dog.java v1.2
Clean & simple

```
public class Dog {
    private final String name;
    private int woofCount = 0;

    public Dog(String name) {
        this.name = name;
    }

    public void woof() {
        ++woofCount;
    }
}
```

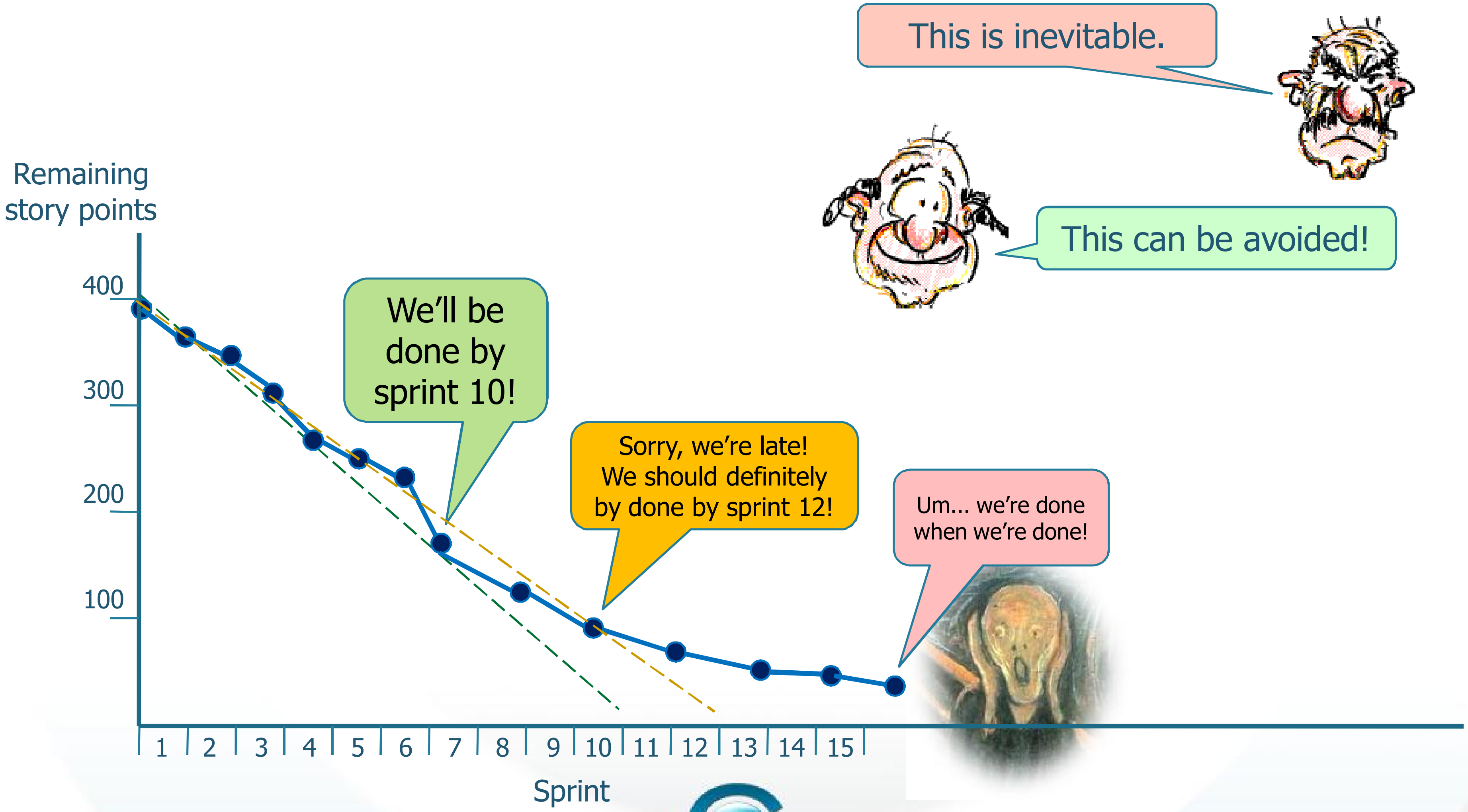
Getting to v1.2 takes too long!



Simple is hard. But worth the effort.

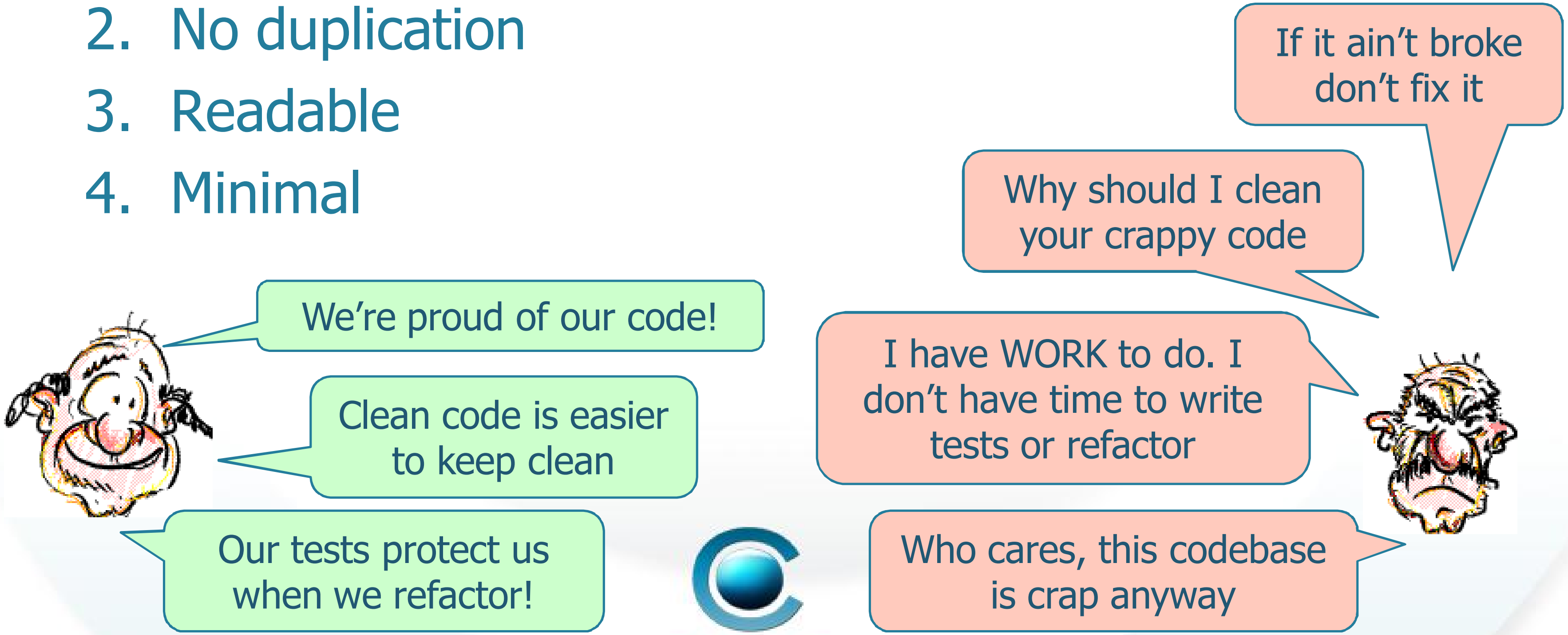
Henrik Kniberg

Technical debt



Sustainable pace & simple code

- **Limit work to capacity**
- **Continuously refactor to keep the code clean & simple**
  1. Passes all tests
  2. No duplication
  3. Readable
  4. Minimal



We're proud of our code!

Clean code is easier to keep clean

Our tests protect us when we refactor!

If it ain't broke don't fix it

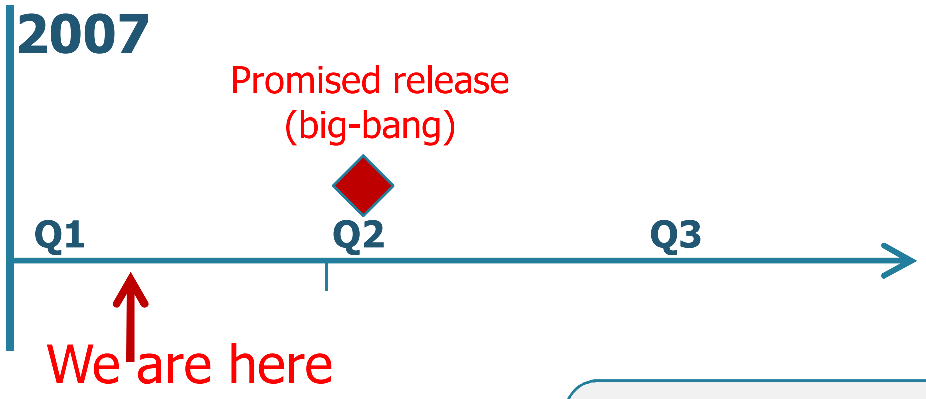
Why should I clean your crappy code

I have WORK to do. I don't have time to write tests or refactor

Who cares, this codebase is crap anyway

Being an agile developer can be hard because you have to:

Be brave

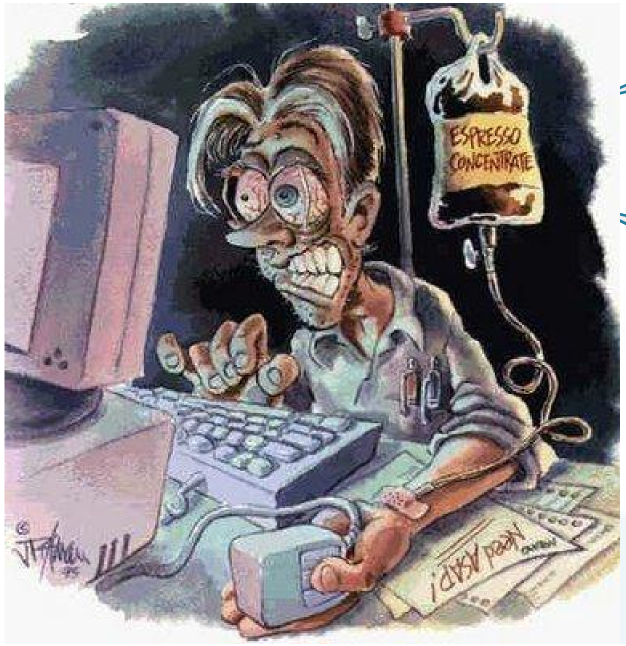


DEFINITION OF DONE

- CODE CHECKED IN?

BACKLOG = ? POINTS

VELOCITY = ? POINTS/SPRINT



What if we don't make the release?

You HAVE to make the release! We've promised!

Let's create a backlog & change the definition of done & start measuring velocity.

DEFINITION OF DONE

- TESTED & INTEGRATED
- RELEASABLE

Reality hurts

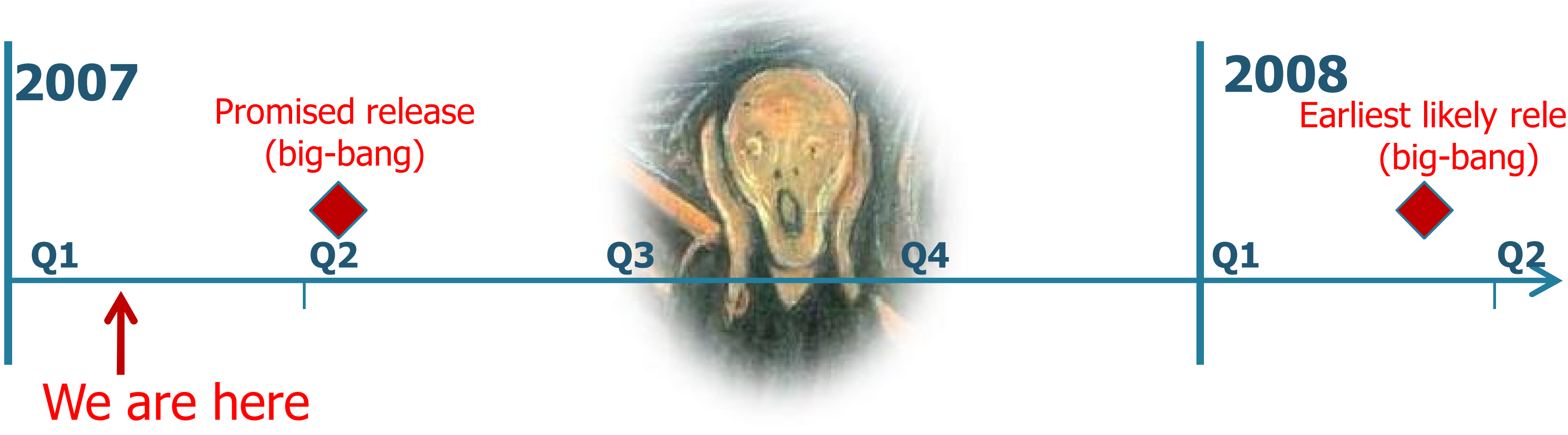
DEFINITION OF DONE

- TESTED & INTEGRATED
- RELEASABLE

BACKLOG = 250 POINTS

VELOCITY = 10 POINTS/SPRINT

⇨ 25 SPRINTS ⇨ > 1 YEAR UNTIL RELEASE!



2007

Promised release (big-bang)

Q1 Q2 Q3 Q4

2008

Earliest likely release (big-bang)

Q1 Q2

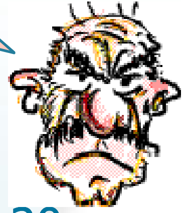
We are here

This plan is dead, let's make a new one!

There is no such thing as "the plan must be right because we promised"

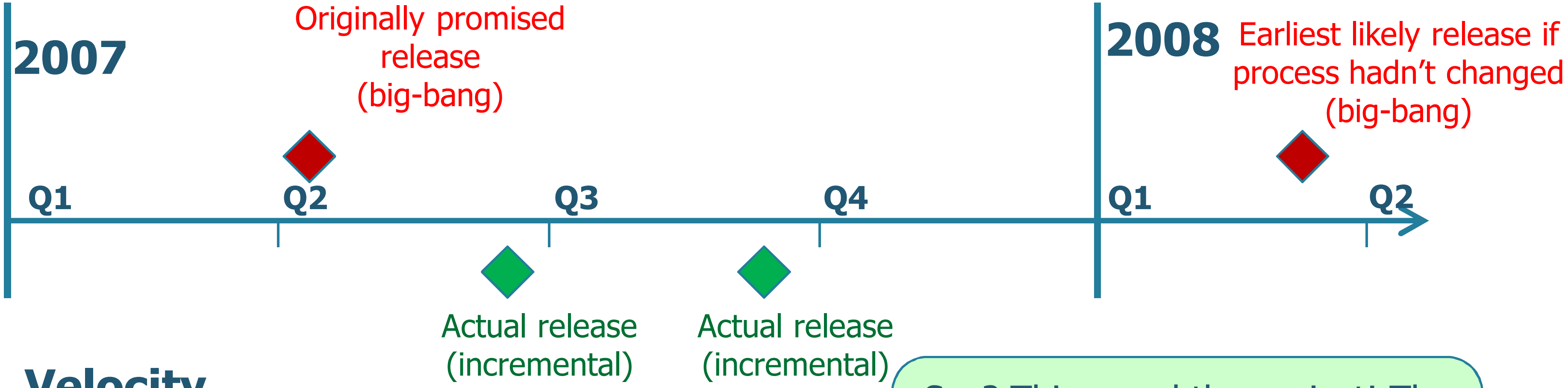
Honesty is the best way to build trust, even when it hurts

Better to bring the bad news now than later

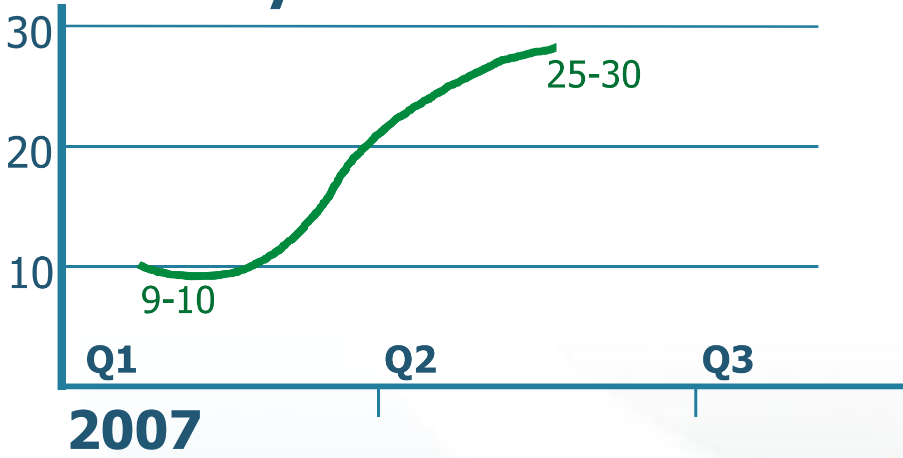


Look what you did!
You caused an uproar!
Now we're all gonna get fired!

Getting back on track


2007
Originally promised release (big-bang)
2008
Earliest likely release if process hadn't changed (big-bang)
Q1
Q2
Q3
Q4
Q1
Q2
Actual release (incremental)
Actual release (incremental)

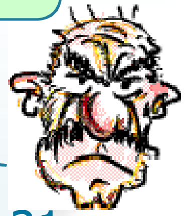
Velocity



See? This saved the project! The customer reduced scope, and everyone helped remove impediments from the team.

No more death marches! No more unrealistic deadlines!

You still caused an uproar.



Being an agile developer can be hard because you have to:

Work as a team

1: Individuals and interactions over processes and tools

Pair programming

Shared responsibility
Collective code ownership

I get to teach & learn!

If I get stuck someone will help me!

Team flow!

Better quality code = Less debugging!

I don’t want to help those other dimwits

Stay away from me. I just want to work in peace.

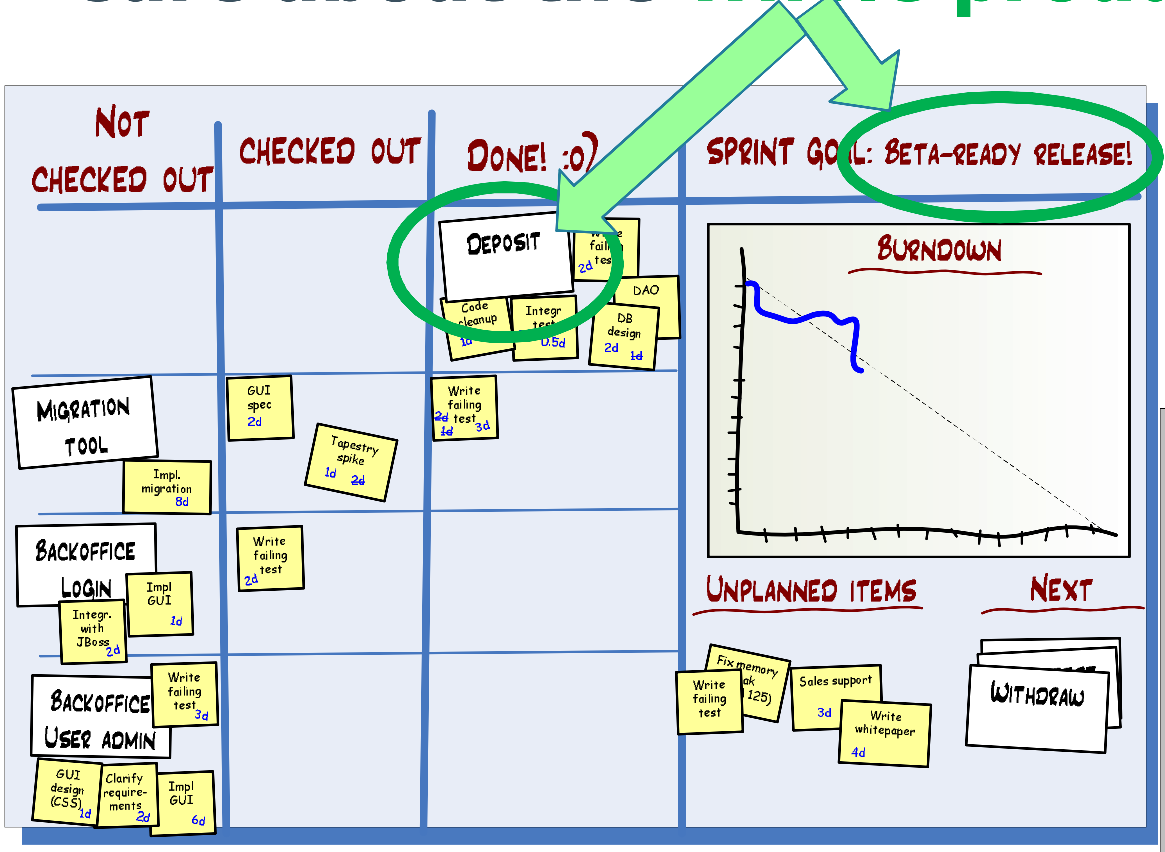
Besides, you smell bad

Don’t you DARE touch my code

Henrik Kniberg

Being an agile developer can be hard because you have to:

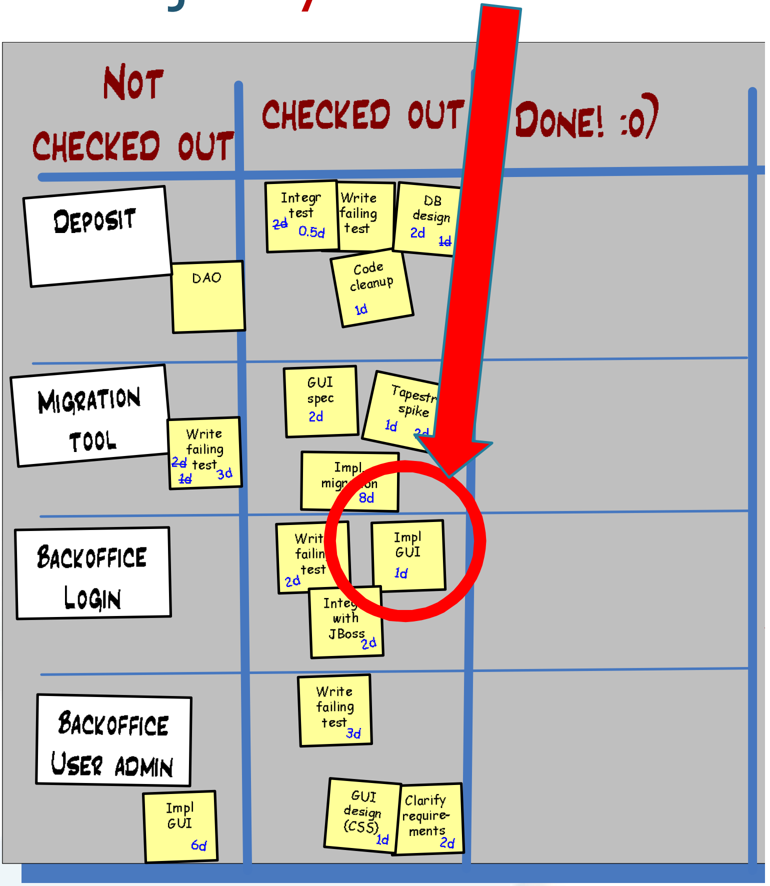
Care about the whole product

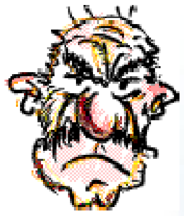


This product rocks!

Boy are we effective as a team!

Not just your little task



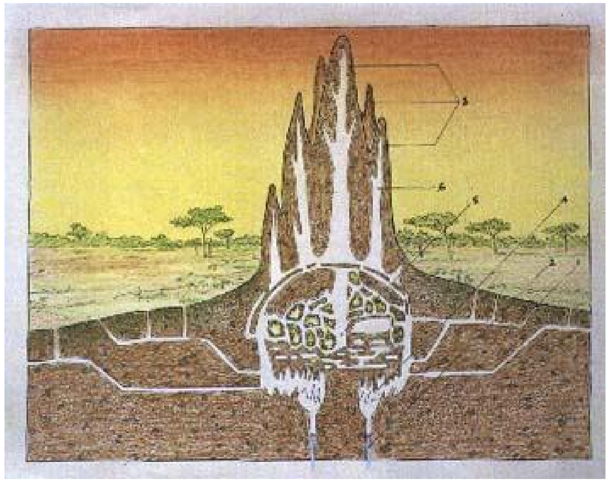


I'm more efficient if I just do my tasks

Henrik Kniberg

crisp

Self-organization



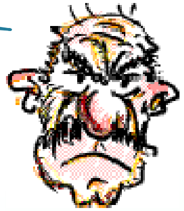
No micromanagement, nobody bossing me around!

I don’t know what I’m supposed to do today.

Less overtime!

You get to figure that our for yourself in collaboration with your team!

I’m not paid to think. I’m paid to code!



Summary

Being an agile developer can be hard because you have to:

- **Think about how you work**
- **Work with the customer**
- **Deliver incrementally**
- **Plan continuously**
- **Write good code**
- **Be brave**
- **Work as a team**
- **Care about the whole product**
- ... among other things

Take-away points

Agile is simple but hard

... like chess

... and piano playing

If you are agile already

- Respect the beginner's initial discomfort and gently help them get over it.
- Don't be dogmatic.

If you are new to agile

- Be prepared for some initial pain & discomfort.
- Be patient. Once you get used to being agile you'll probably never want to go back.