

DAT257

Lecture 5: Technical agile

Jonas Petré

Welcome! We will begin in a few minutes

Jonas Petrén

10 years in software testing, development and as test manager. **6+ years of Scrum Master experience.** Full-time Scrum Master since beginning of 2018. Employed by HiQ since 2012.

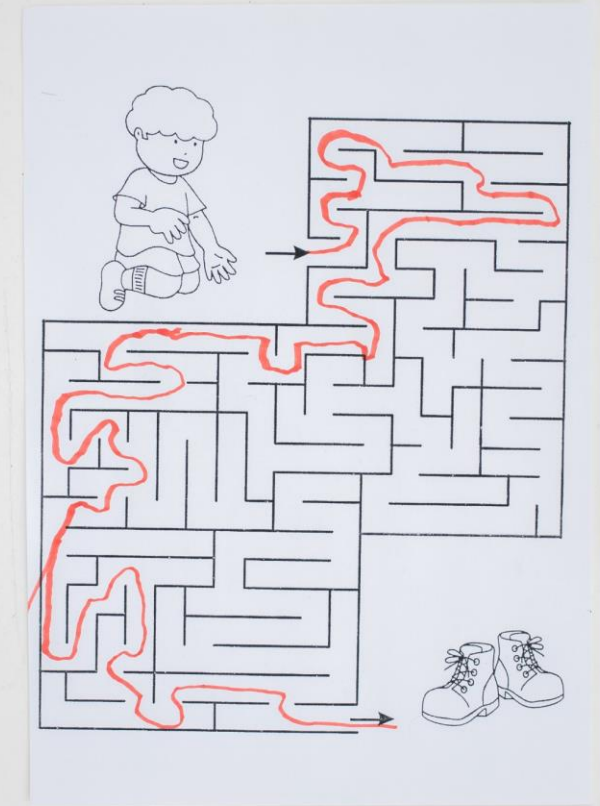
Civilingenjör Informationsteknologi, Linköping University, 2003-2008

Today: Senior Scrum Master/Agile coach
Certified **Professional Scrum Master III**

The logo for HiQ, featuring the letters 'HiQ' in a stylized, handwritten black font.

The goal for today


To get a little bit of understanding of the **technical skills** needed to build a product using agile thinking



Agenda

- Good code
- CI/CD
- DevOps
- Technical debt
- Ending



The background of the slide is a blurred photograph of several people in a meeting or collaborative work environment. They appear to be looking at a screen or document together. The image is tilted slightly to the right.

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.

Continuous attention to technical excellence
and good design enhances agility.

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

GOOD CODE

Easy to read, understand and work with




Good code

When you are working with software – 90 % of your time is reading code

Write your code so that others can understand it

Non-existing code doesn't crash. Write only the code you need

The background of the slide features a blurred image of a computer screen displaying Python code. The code appears to be related to Blender's bpy module, specifically dealing with mirror modifiers and object selection. Visible snippets include setting mirror_mod.mirror_object, defining mirror_mod.use_x, use_y, and use_z for different mirror operations (MIRROR_X, MIRROR_Y, MIRROR_Z), and setting context.scene.objects.active.

```
mirror_mod = modifier_ob.  
Set mirror object to mirror  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
selection at the end -add  
obj.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.name))  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select
```

Continuous attention to technical excellence and good design enhances agility.

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

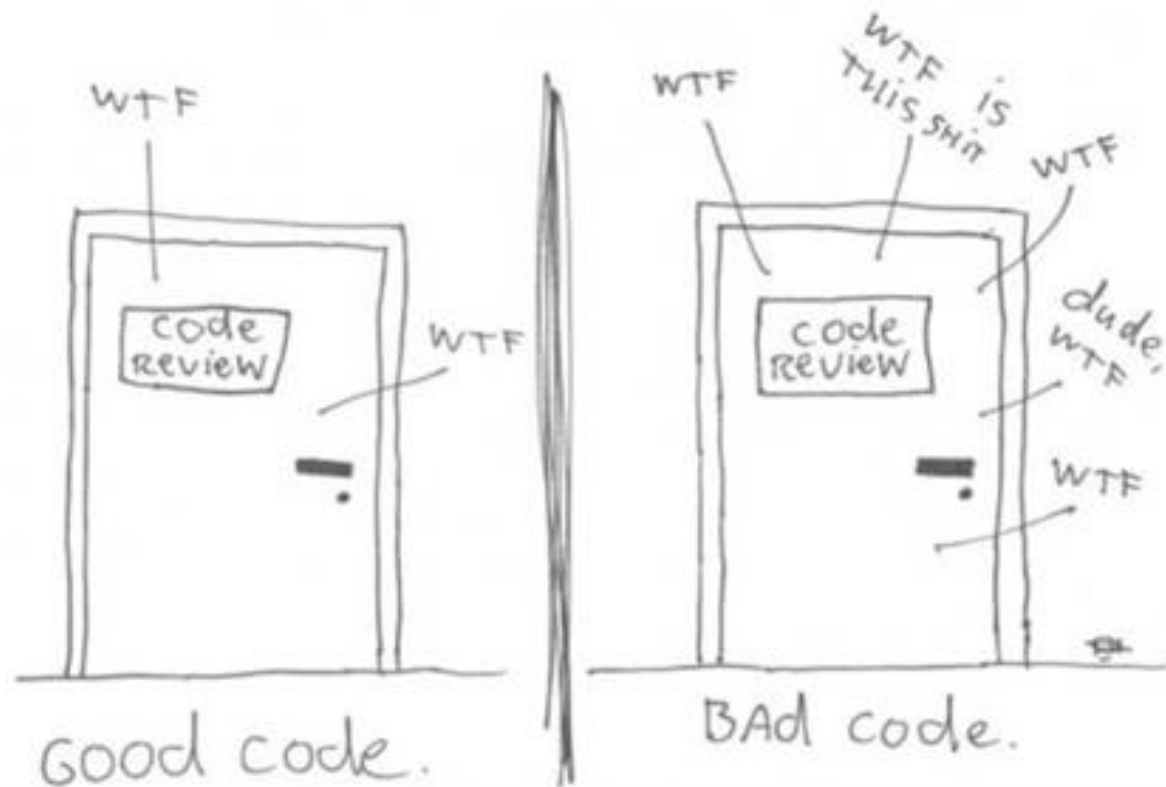
```
context):  
context.active_object is not None
```




**It looks like
someone has
programmed with
their feet**

Takeaway: Care for your code

The ONLY valid measurement
of code quality: WTFs/minute



Discussion 2 minutes

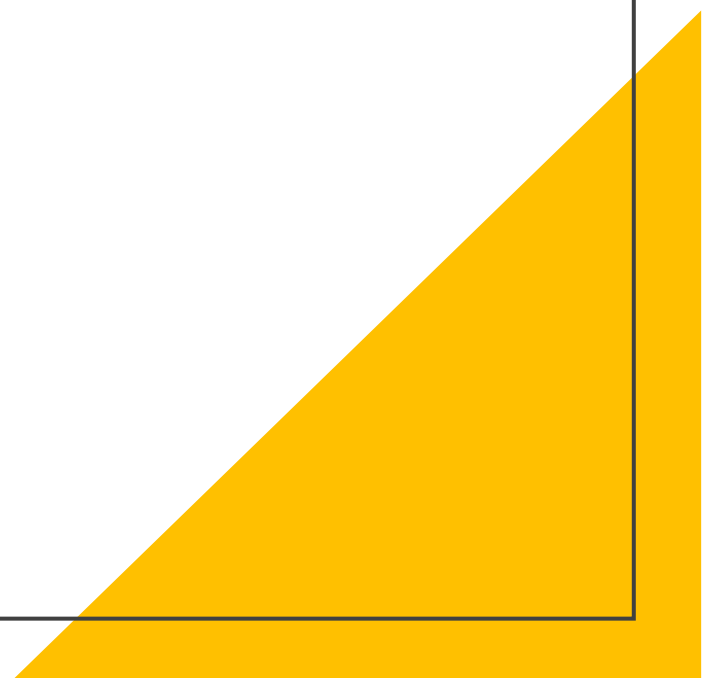
Have you seen good code? If so, what made it good? How can you try to write good code in your project?

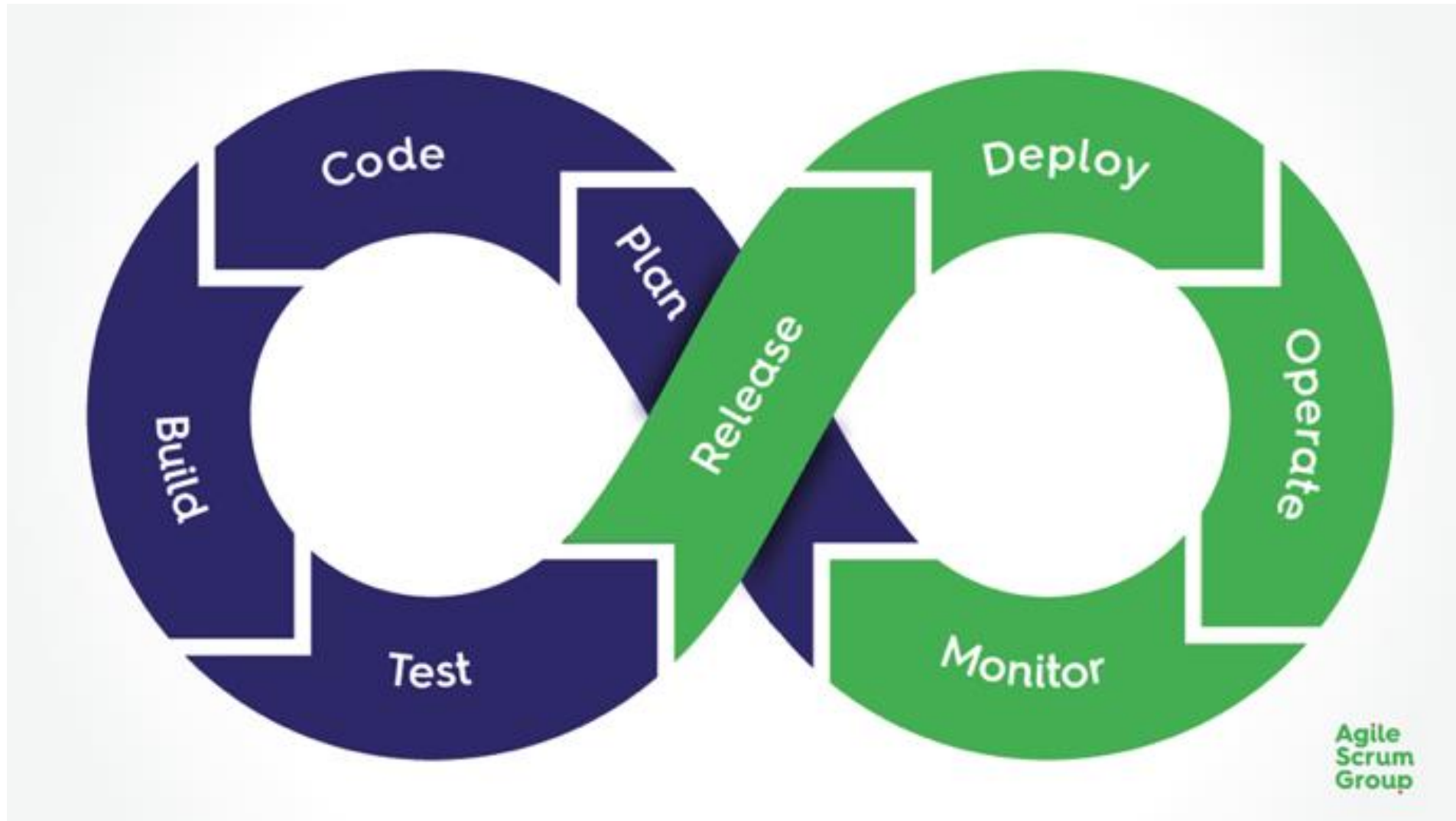


CI/CD

Continuous integration

Continuous delivery





Why CI/CD?

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.



Early Detection of Bugs



Enables Continuous Deployment



Emphasize Test Automation



Helps in determining Code Quality / Code Breaks



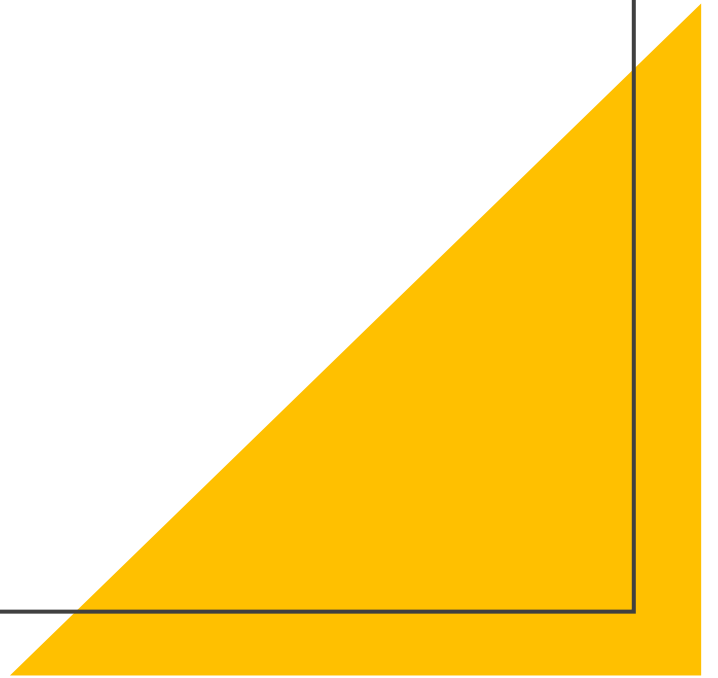
Faster Development Cycle



Improved Quality of the Product

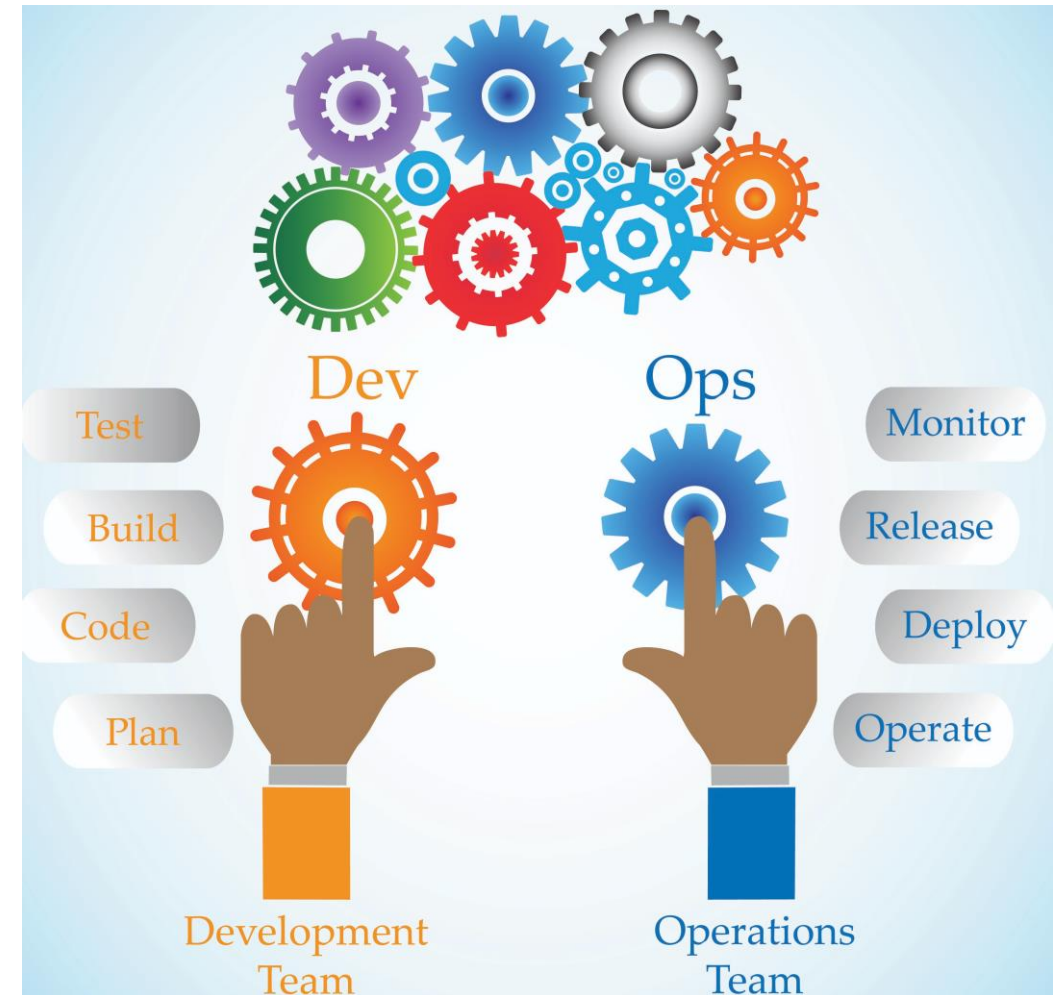
DEVOPS

Integrating development and
operations



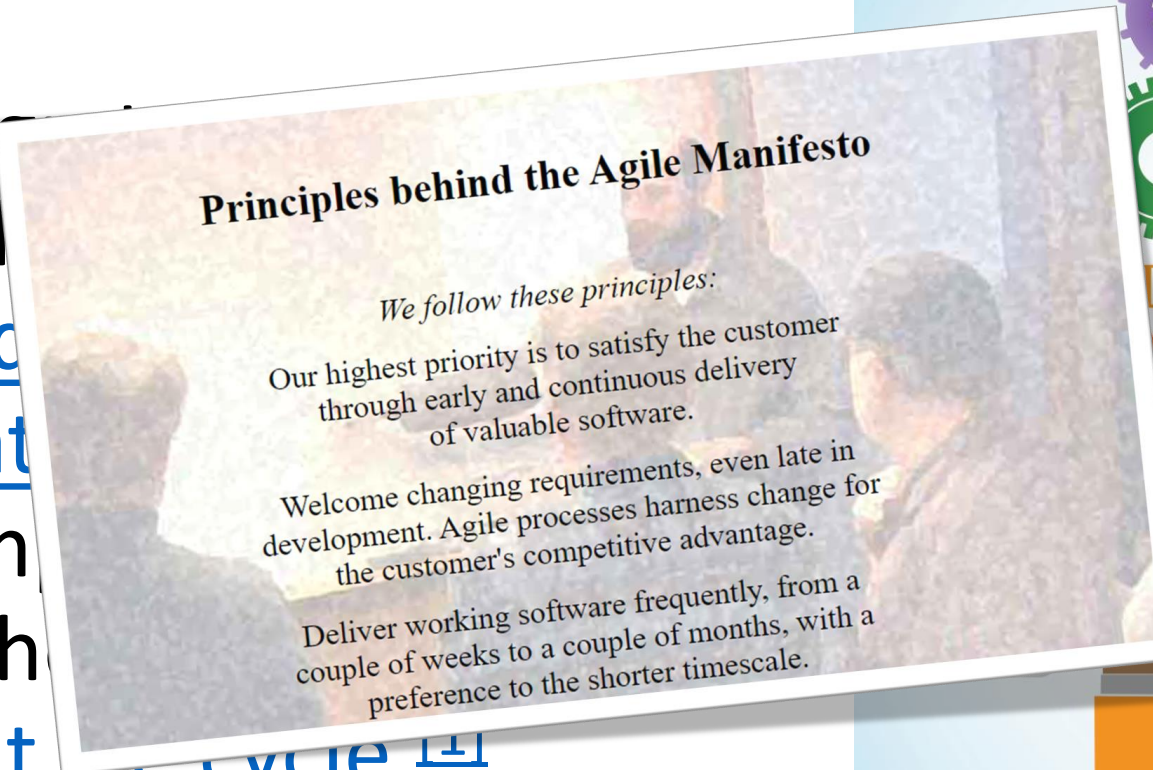
What is DevOps?

DevOps integrates and automates the work of software development (*Dev*) and IT operations (*Ops*) as a means for improving and shortening the systems development life cycle.^[1]



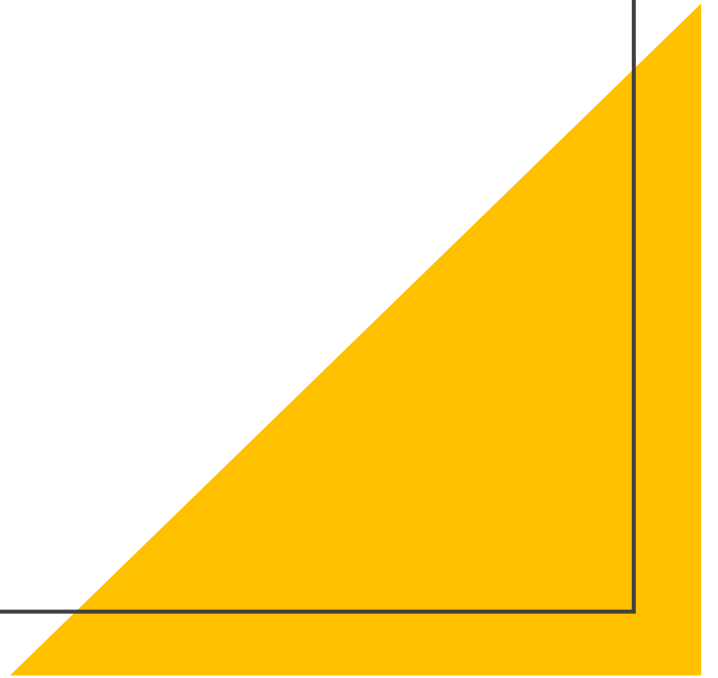
What is DevOps?

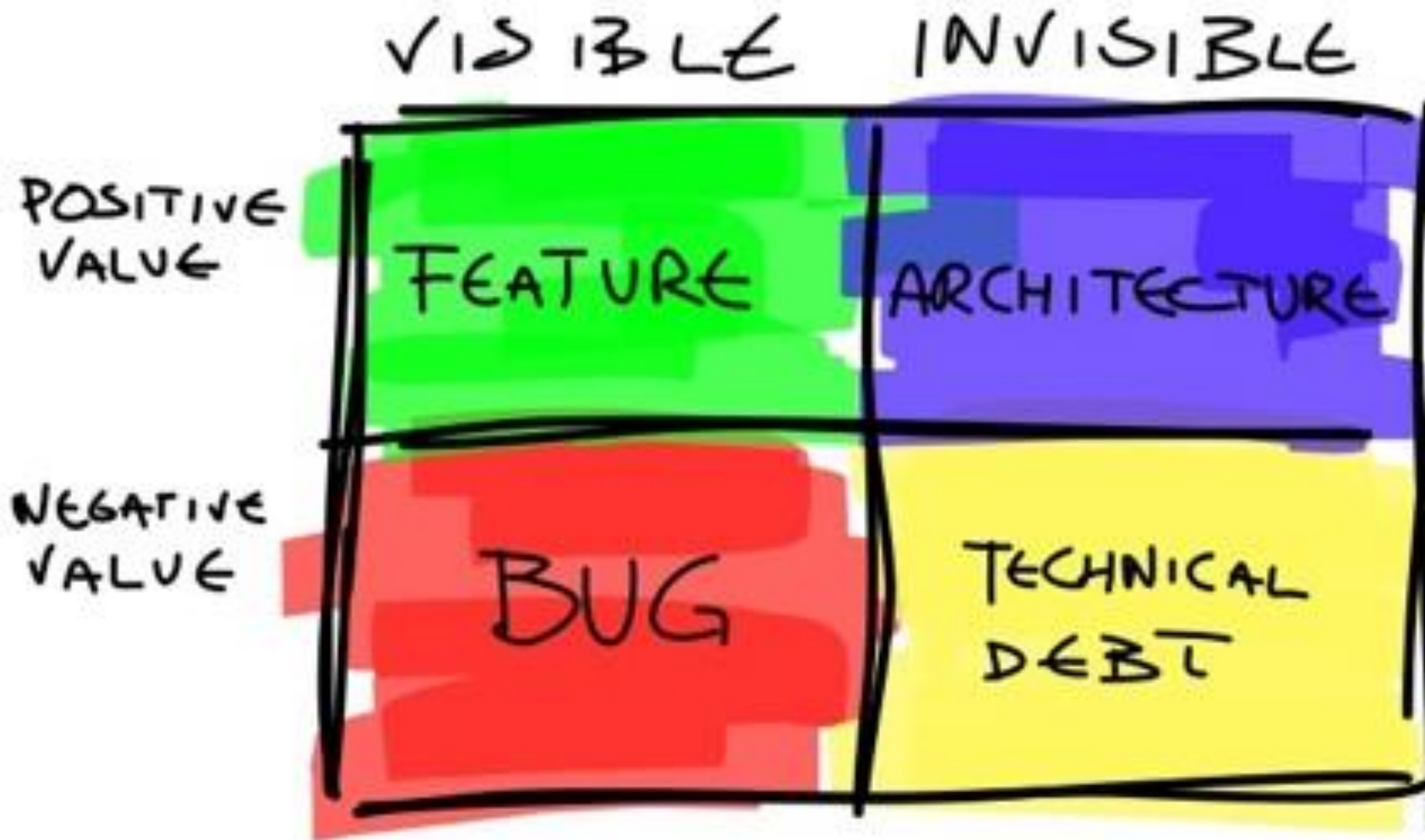
DevOps integrates development and operations, automates the workflow, and enables the delivery of software and IT operations. DevOps means for improving collaboration, automating the workflow, and shortening the development and deployment cycle.



TECHNICAL DEBT

Try to keep it under control!







Causes and effect

Technical debt may stem from bad design, bad architecture, time pressure, bad programming etc.

It leads to **false assumptions** about the current state of the system.

Causes and effect

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.

system
d
pressure,
c.

assumptions about the
current state of the system.

How to improve the situation

Technical debt is like the credit on your house.
To reduce your debts, you must

- 1) **Stop taking new loans** – make no more quick-fixes or take other short cuts in the code
- 2) **Pay your rent** – this is the time you spend on fighting the bad code base
- 3) **Pay off your loan** – pay back your debts by refactoring the code



Key takeaways



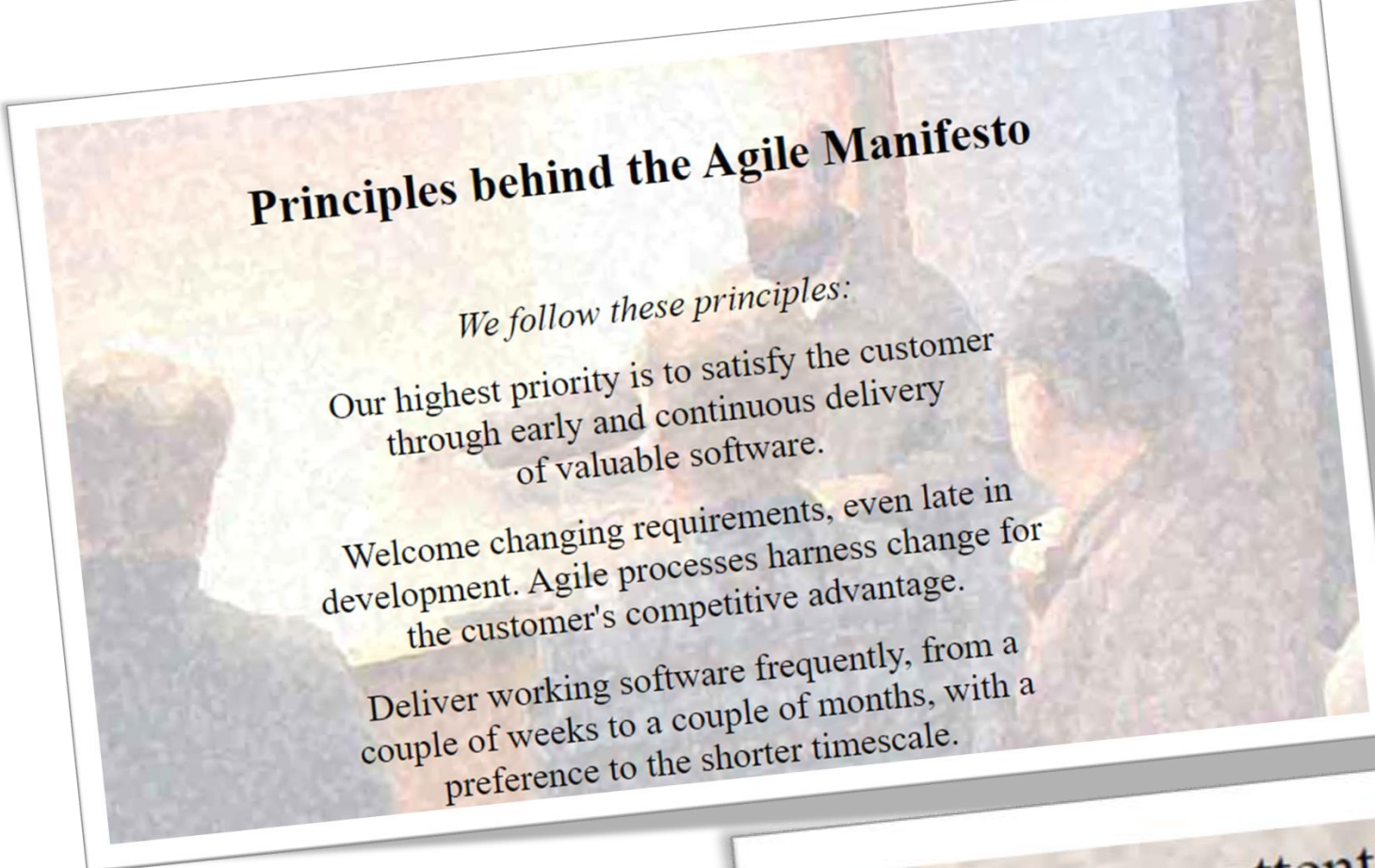


Care for your code



Keep your
technical debt
under control





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.

Continuous attention to technical excellence and good design enhances agility.

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

Technical skills are keys to a successful, long-lived product





Thea Schukken
www.beeldinwerking.nl