Software Process

Thomas Fritz

Many thanks to Reid C. Holmes and Elisa Baniassad

Agenda

1. Why process – example of a failure
2. Process overview
   Class activities
3. Agile methods
4. Activity & questions
5. Quiz & next steps

Note: don’t forget to answer the quiz if you haven’t done so already.
10-10:20

Examinable skills

At the end of this lecture, you should be able to…

- Explain why process is important, and why it should be adopted early
- Elaborate on software process risks, explaining how they may affect software development outcomes
- Compare process models (advantages, drawbacks, commonalities)
- Explain key concepts, terms, values and practices of Agile and specifically also Scrum
- Given a case study/project, choose an appropriate process model, explain how one could apply it / structure it and justify it

Sidenote: color coding of slide titles

Class Questions

Answers (in class and on slides after lecture)

Recap (mainly for reference and recap)

I'll go over these ones more briefly or might even skip them

Why Process?

Learning Objectives

Be able to:

- Explain why process is important and why it should be adopted early
- Elaborate on software development risks and their effects

Software Project Risks

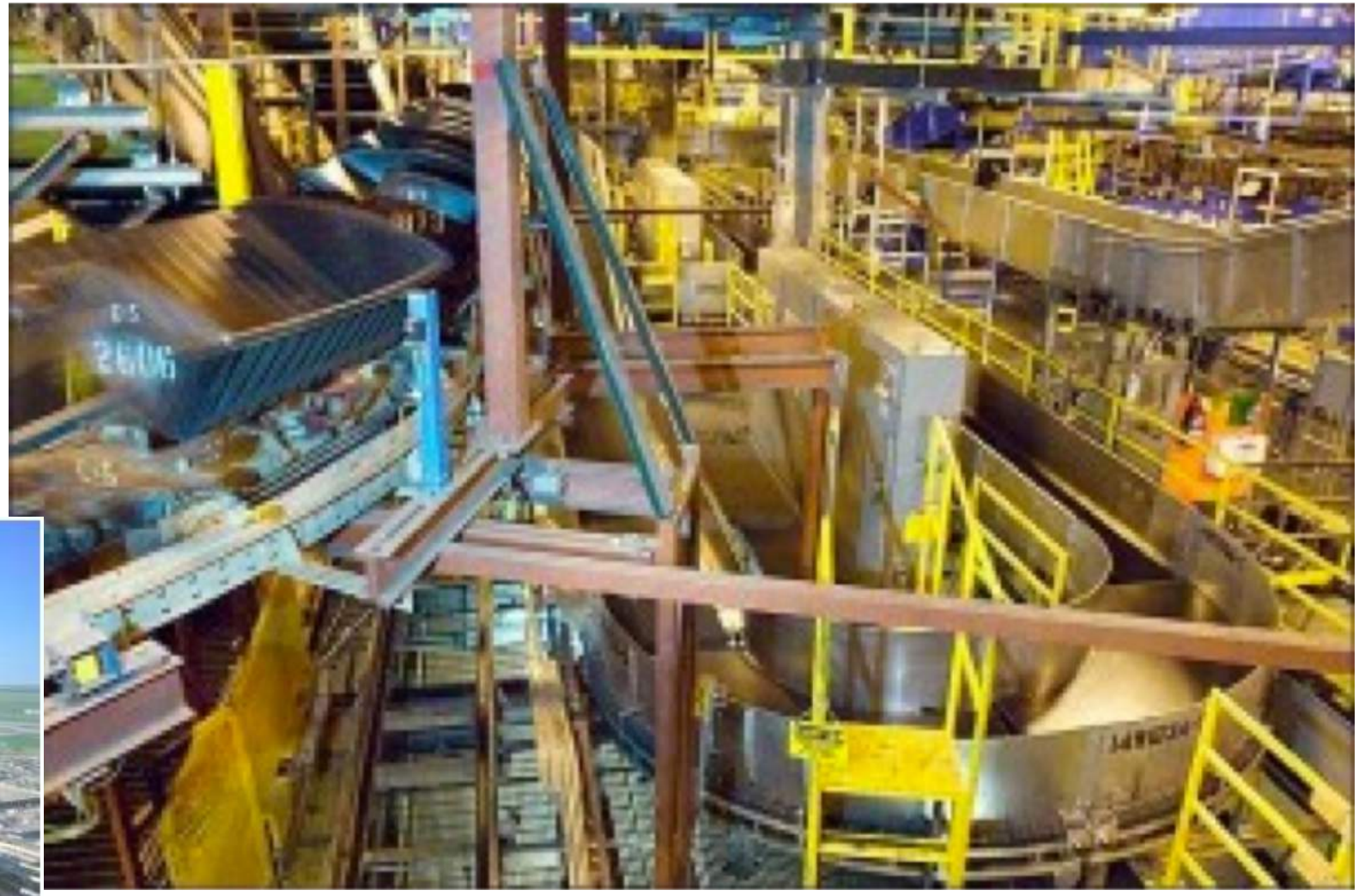
When you take on a project, consider these risks, and whether you can mitigate them!

6 dimensions of risks

- **USER:** resistance to change; conflicts between them; negative attitudes towards the project; lack of commitment; lack of cooperation
- **REQUIREMENTS:** continually changing; inadequately identified; unclear; incorrect
- **PROJECT COMPLEXITY:** new technology; high technical complexity; immature technology; first use of technology
- **PLANNING & CONTROL:** poor process oversight; inadequate estimation resources; poor planning; unclear milestones; inexperienced PM's; ineffective communication
- **TEAM:** lack of experience; lack of training; lack of specialized skill
- **ORGANISATIONAL ENVIRONMENT:** change of management during the project; unstable organization; ongoing restructuring

Why Process?

Denver Baggage (mis)Handling



http://www.nytimes.com/2005/08/26/world/americas/26iht-denver.html?_r=1
http://calleam.com/WTPF/?page_id=2086
http://calleam.com/WTPF/wp-content/uploads/articles/DIABaggage.pdf

The story…part 1

Nov 1989	Work starts on the construction of the airport	
Oct 1990	City of Denver engages Breier Neidle Patrone Associates to analyse feasibility of building an integrated baggage system. Reports advises that complexity makes the proposition unfeasible	risk flagged; ignored.
Feb 1991	Continental Airlines signs on and plans on using Denver as a hub	scale changed
Jun 1991	United Airlines signs on and plans on using Concourse A as a hub	
Jun 1991	United Airlines engages BAE Systems to build an automated baggage system for Concourse A. BAE was a world leader in the supply, installation and operation of baggage handling equipment	
Summer 1991	Airport's Project Management team recognizes that a baggage handling solution for the complete airport was required. Bids for an airport wide solution are requested	
Fall 1991	Of the 16 companies included in the bidding process only 3 respond and review of proposals indicate none could be ready in time for the Oct 1993 opening. The 3 bids are all rejected	risks flagged; ignored
Early 1992	Denver Airport Project Management team approach BAE directly requesting a bid for the project	
Apr 1992	Denver Airport contracts with BAE to expand the United Airlines baggage handling system into an integrated system handling all 3 concourses, all airlines, departing as well as arriving flights. In addition system is to handle transfer baggage automatically. Contract is hammered out in 3 intense working sessions	hasty contract
Aug 1992	United Airlines changes their plans and cuts out plans for the system to transfer bags between aircraft. Resulting changes save $20m, but result in a major redesign of the United Airlines portion of the system. Change requests are raised to add automated handling of oversized baggage and for the creation of a dedicated ski equipment handling area	requirements change
Sep 1992	Continental requests ski equipment handling facilities be added to their concourse as well	requirements change
Oct 1992	Chief Airport Engineer, Walter Singer dies. Mr Singer had been one of the driving forces behind the creation of the automated baggage system	guru dies

The story…part 2

Jan 1993	Change orders raised altering size of ski equipment claim area and adding maintenance tracks so carts could be serviced without having to be removed from the rails
Feb 1993	Target opening date shifted from 31 Oct 93 to 19 Dec 93 and soon thereafter to 9 Mar 94
Sep 1993	Target opening date is shifted again, new target date is 15 May 1994
31 Oct 1993	Original target for opening
19 Dec 1993	Second target for opening
Jan 1994	United Airlines requests further changes to the oversize baggage input area
9 Mar 1994	Third target for opening
Mar 1994	Problems establishing a clean electrical supply results in continual power outages that disrupt testing and development. Solution requires installation of industrial filters into the electrical system. Ordering and installation of the filters takes several months
Apr 1994	Airport authorities arrange a demonstration for the system for the media (without first informing BAE). Demonstration is a disaster as clothes are disgorged from crushed bags
Apr 1994	Denver Mayor cancels 15 May target date and announces an indefinite delay in opening
May 1994	Logplan Consulting engaged to evaluate the project
15 May 1994	Fourth target for opening
May 1994	BAE Systems denies system is malfunctioning. Instead they say many of the issues reported to date had been caused by the airport staff using the system incorrectly
Aug 1994	System testing continues to flounder. Scope of work is radically trimmed back a Logplan's recommendation airport builds a manual tug and trolley system instead
Aug 1994	City of Denver starts fining BAE $12K per day for further delays
28 Feb 1995	Actual opening
Aug 2005	In order to save costs the system is scrapped in favour of a fully manual system. Mainte costs were running at $1M per month at the time.

requirements change

delays

technical challenges

more delays

BAE blames the user

manual system used instead

BAE fined for overtime

done! but badly

scrapped

The system

- 20 miles of track
- 6 miles of conveyor belts
- 56 laser arrays that read bar coded tags
- 3100 standard size baggage “telecars”
- 450 oversize cars
- Some 100 networked computers

Consequences

Airport opening delayed four times

- Overall, 16 months delay

“Revolutionary” new baggage system broadcasted on national television

Engineering firm went bankrupt

New engineering firm

- Split the system in 3 (one per terminal)
- Manual backup system

Overall damage: $1.3 billion

Automatic system abandoned in 2005

- Reported savings of $1M / month

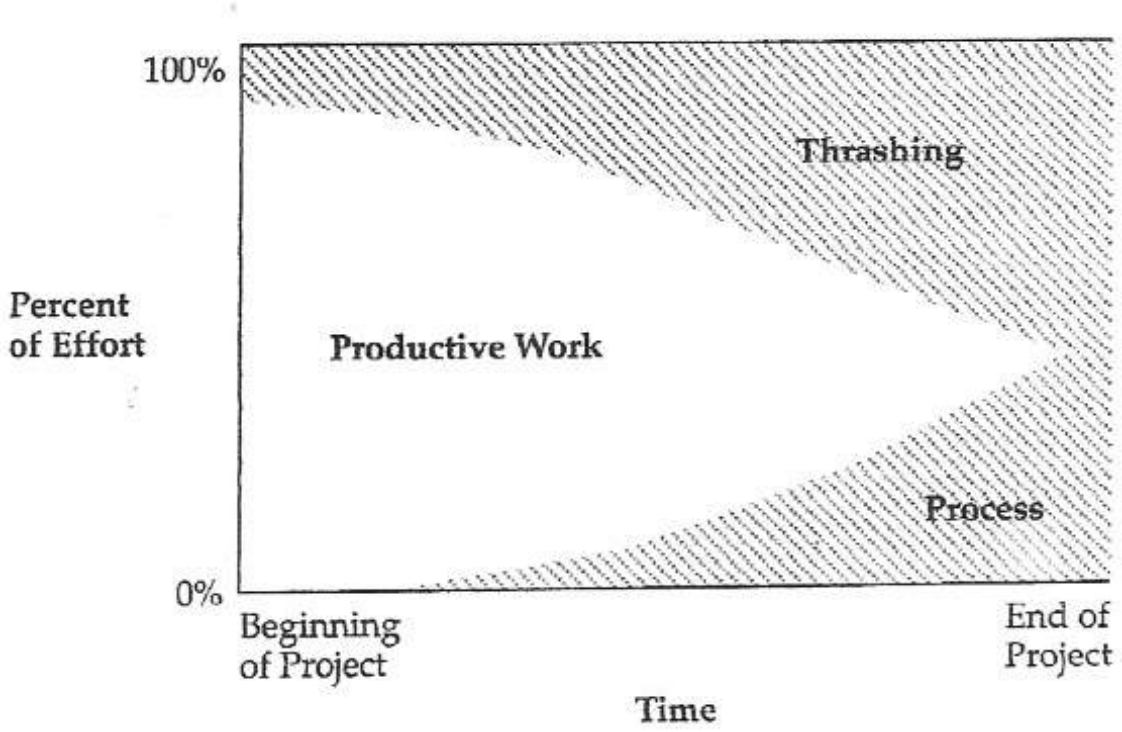
What happened

- **Bad planning:** They left it late! Software production started only 17 months before scheduled opening
  - In Munich, engineers spent **two years testing** a similar but much smaller system. And even that system is not glitch free (nothing ever is - but at least it's 24/7 operational)
- **Physical problems:** Most buildings built before baggage system was designed, meaning the baggage system had to adapt to architecture that wasn't a good fit (sharp turns, narrow corridors)
- **Change of management:** Death of the driving force of the project (you'd think this was atypical, but gurus often leave a company mid-way through a project, leaving the project somewhat stranded if planning isn't good)
- **Accepting changing requirements:** alterations to baggage sizes, types, paths, etc — and the contractor said a firm "yes" to all these changes!
- **Lack of experience:** this was the first time BAE had built a system like this. But for some reason they didn't choose to ask for outside advice from the Munich baggage system engineers who might have provided insights and helped mitigate risk.

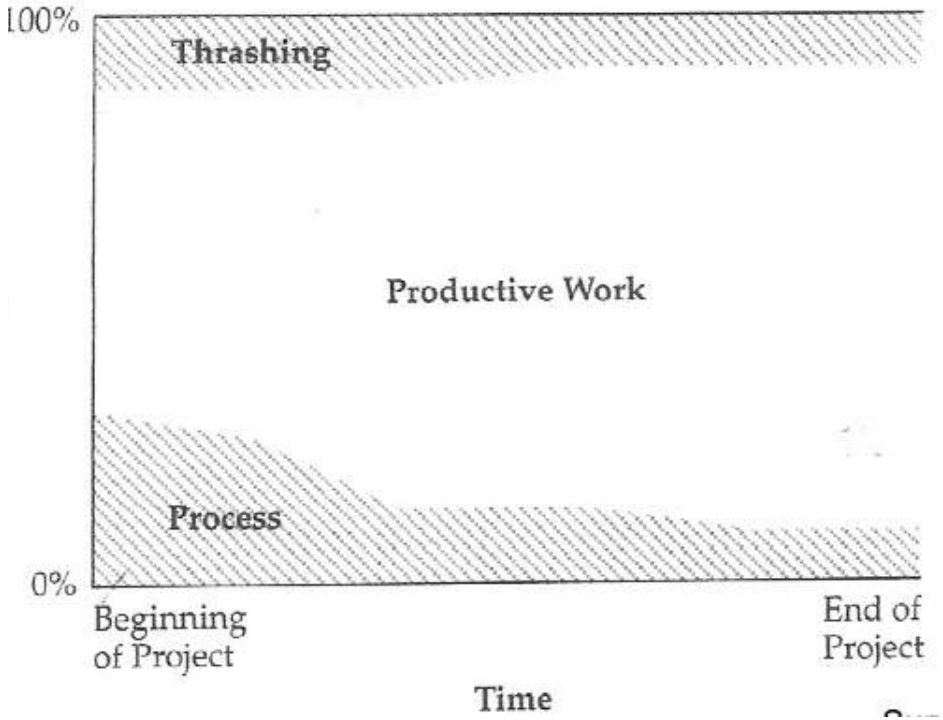
http://www.stevemcconnell.com/articles/art09.htm

Is process worth it?

"thrashing" denotes unproductive work



"When a project has paid too little early attention to the processes it will use, by the end of a project developers feel they are spending all of their time in meetings and correcting defects and little or no time extending the software."



"During the first few weeks of the project, the process-oriented team will seem less productive than the process-phobic team… By the end of the project, the process-oriented team will be operating at a high-speed hum, with little thrashing, and performing its processes with little conscious effort."

Process Overview

Learning Objectives

Be able to:

- Explain what a software process is
- Compare different process models

Software Process

A software process is a structured set of activities to develop a software system

It defines who does what, when, and how, to reach a goal.

Ensures desirable characteristics, e.g. timeliness, efficiency, accountability.

Processes have descriptions that discuss:

Products: the outcome of a process activity

Stakeholders: people who care about the outcome
(managers, developers, customers, testers)

Process phases

Many different software processes.

- Each with their own strengths and weaknesses.

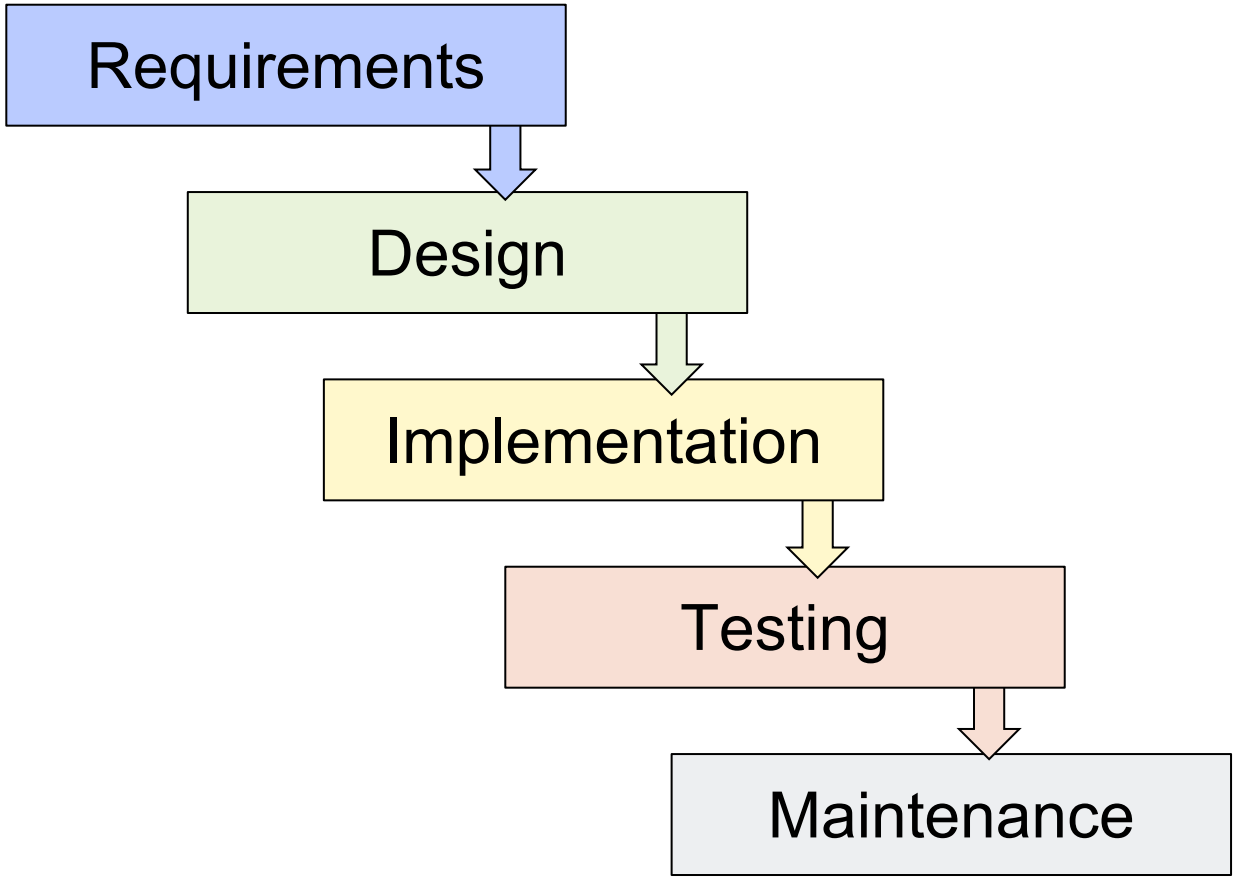
All include:

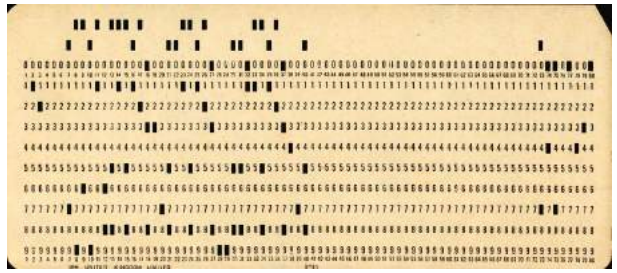
- Requirements elicitation / gathering
- Architectural design
- Detailed design / specification
- Implementation
- Integration
- Testing
- Maintenance

The goal for each of these activities is to:

- Mark out a clear set of steps
- Produce tangible item(s)
- Allow for review of work
- Specify actions to perform next

Waterfall process





Punch card picture by Pete Birkinshaw CC BY 2.0

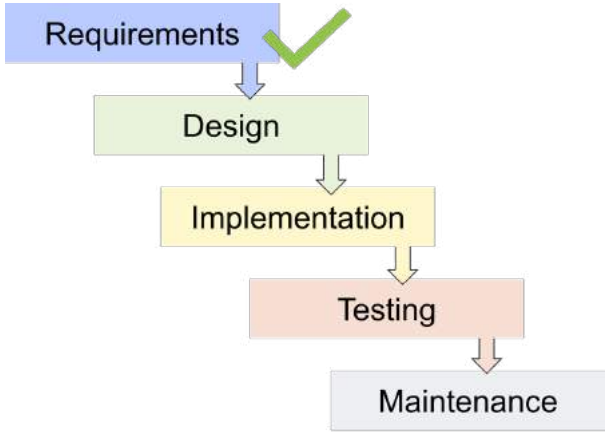
Class Activity – Build a Village

Mandate: Build a Village

The Mayor of a village with many citizens but no infrastructure wants you to build the village infrastructure. **Your job is to design and build a village for the villagers.**

The Business Analyst already identified the following **requirements** talking to the mayor:

- [**REQ1**] The village shall have 2 colorful houses with pitched roofs.
- [**REQ2**] The village shall have a playground with a slide and a swing.



Class Activity – Design & Implement [15 mins]

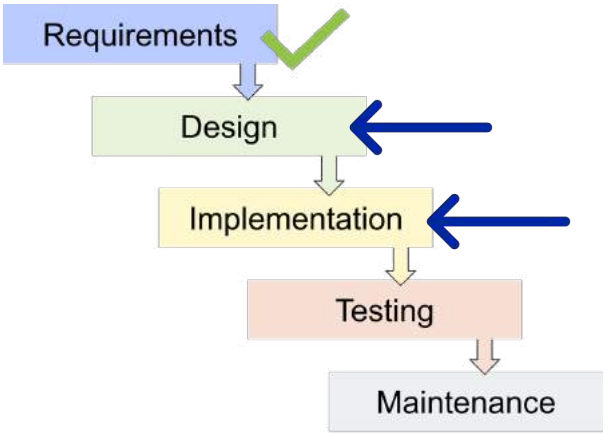
Design & Implement the Village in Parallel

<u>Designers</u> [in teams of 3-4]

Design a village for people according to the requirements defined by the Business Analyst.

<u>Builders</u> [2 teams of 4 volunteers]

Build a village according to the winning design from a previous year.



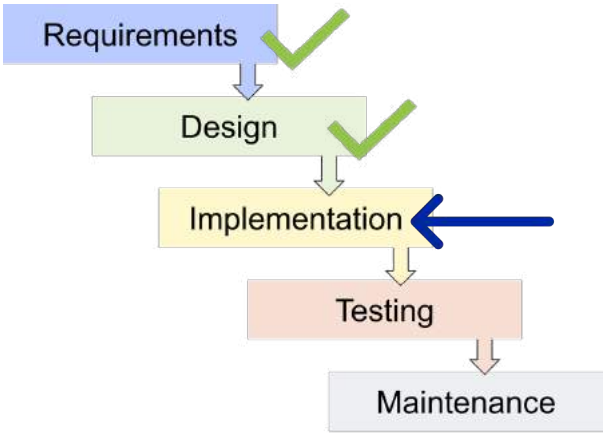
Class Activity – Implementation [in class] (15mins)

Two teams of 4 in-class builders

You are asked to build the village in the next 15 minutes according to the following design.

After that, the mayor will provide feedback and decide on the best village to win.

Team B: each person has to choose ONE color and can only work with that one!

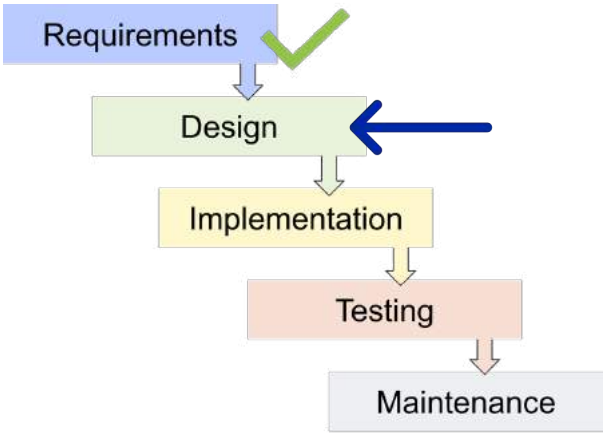


Class Activity – Design [https://bit.ly/49HeYSS] (15mins)

In teams of 3-4 (all that are not building the village)

Mayor of the village wants you to design the village infrastructure. The Business Analyst already identified the following **requirements** talking to the mayor:

- [**REQ1**] The village shall have 2 colorful houses with pitched roofs.
- [**REQ2**] The village shall have a playground with a slide and a swing.



Sketch out the design and paste at

https://bit.ly/49HeYSS

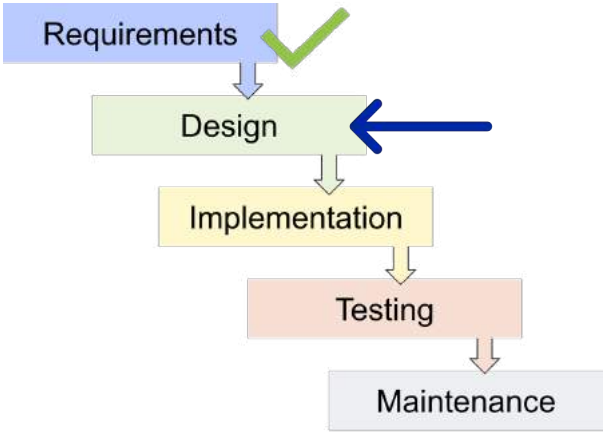
Class Activity – **Design** – Discussion

Designers

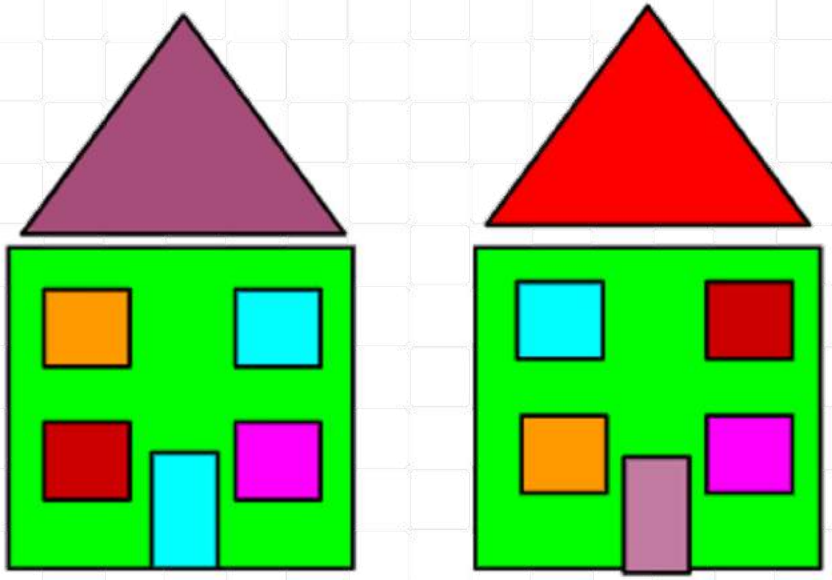
Requirements of the mayor:

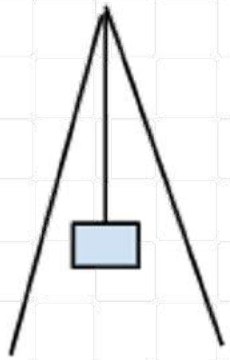
- [**REQ1**] The village shall have 2 colorful houses with pitched roofs.
- [**REQ2**] The village shall have a playground with a slide and a swing.

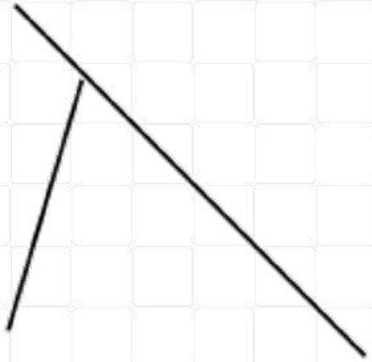
Do you think these are good city designs and will fullfill the customers need? Why or why not?



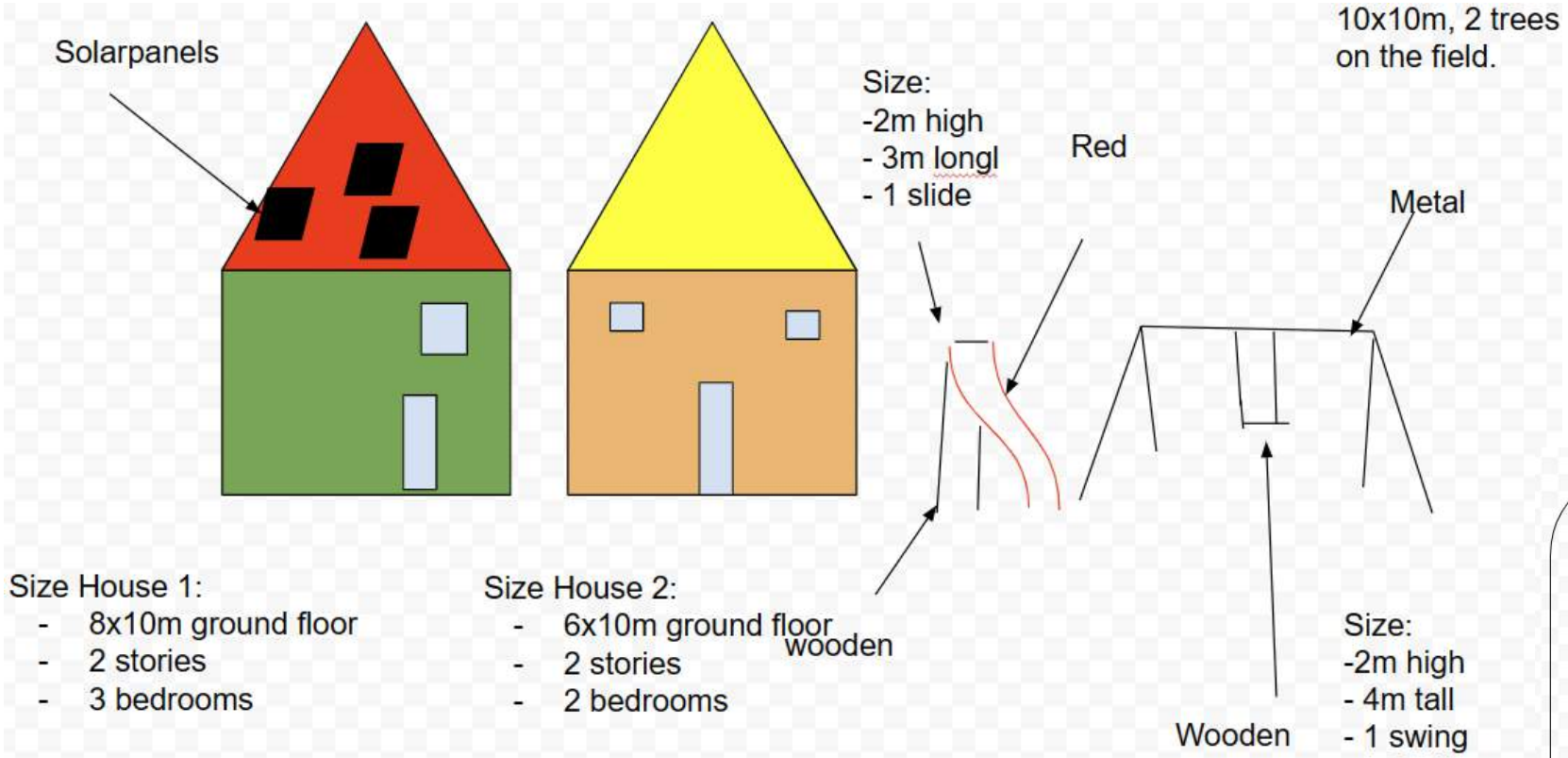
Class Activity – Previous year's village designs





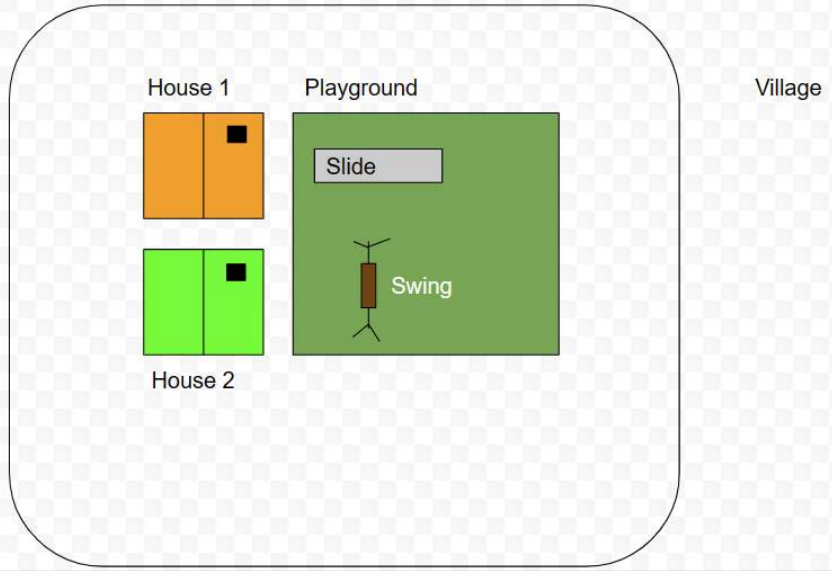


Class Activity – Previous year's village designs

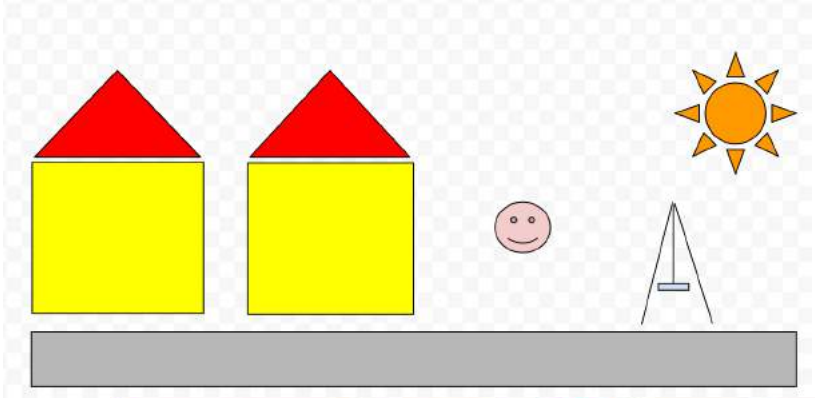


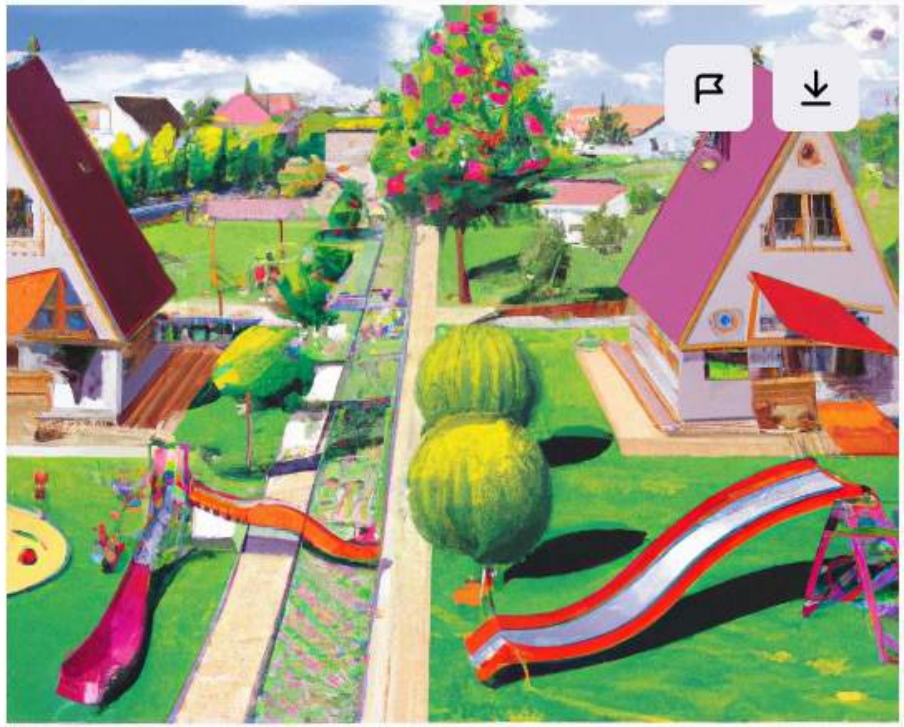
Size House 1:
- 8x10m ground floor
- 2 stories
- 3 bedrooms

Size House 2:
- 6x10m ground floor
- 2 stories
- 2 bedrooms

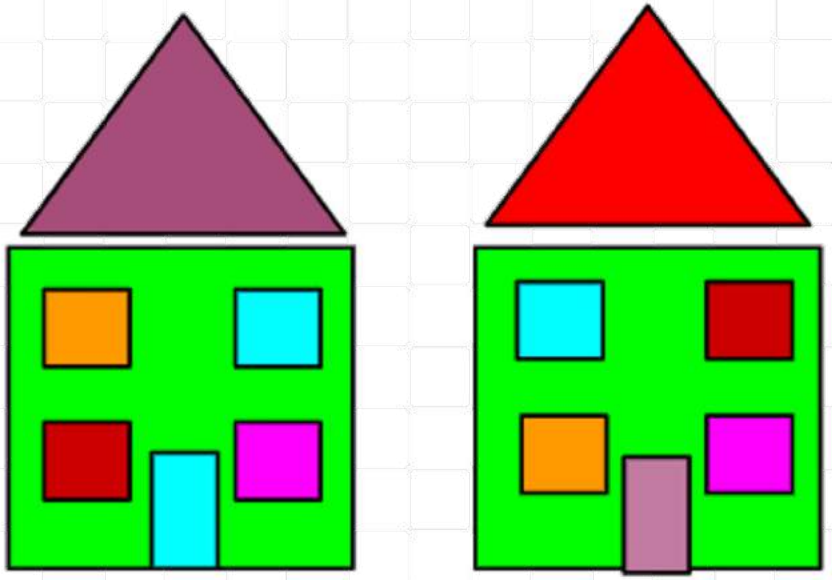


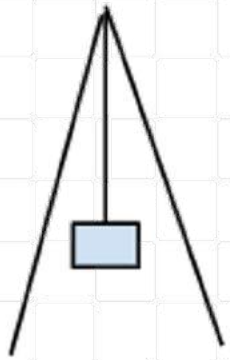
Class Activity – Previous year's village designs

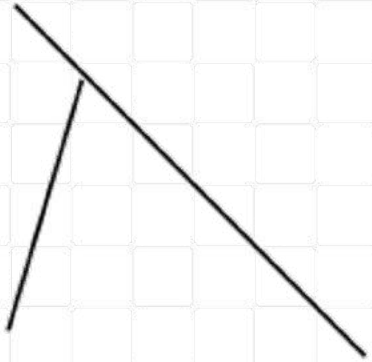




Class Activity – Builders: the design







Building a village exercise with waterfall - discussion

Builders

Requirements of the mayor:

- [**REQ1**] The village shall have 2 colorful houses with pitched roofs.
- [**REQ2**] The village shall have a playground with a slide and a swing.

Each team: please briefly present your village

https://thisiszone.medium.com/using-lego-to-show-the-advantages-of-an-agile-approach-to-software-development-3eda6e5c2114

Building a village exercise with waterfall - feedback

Builders

Requirements of the mayor:

- [**REQ1**] The village shall have 2 colorful houses with pitched roofs.
- [**REQ2**] The village shall have a playground with a slide and a swing.

1. Which team has the better city?
2. Do you think these are good cities and will fullfill the customers need? Why or why not?

https://thisiszone.medium.com/using-lego-to-show-the-advantages-of-an-agile-approach-to-software-development-3eda6e5c2114

Class Activity – Previous year's village implementations

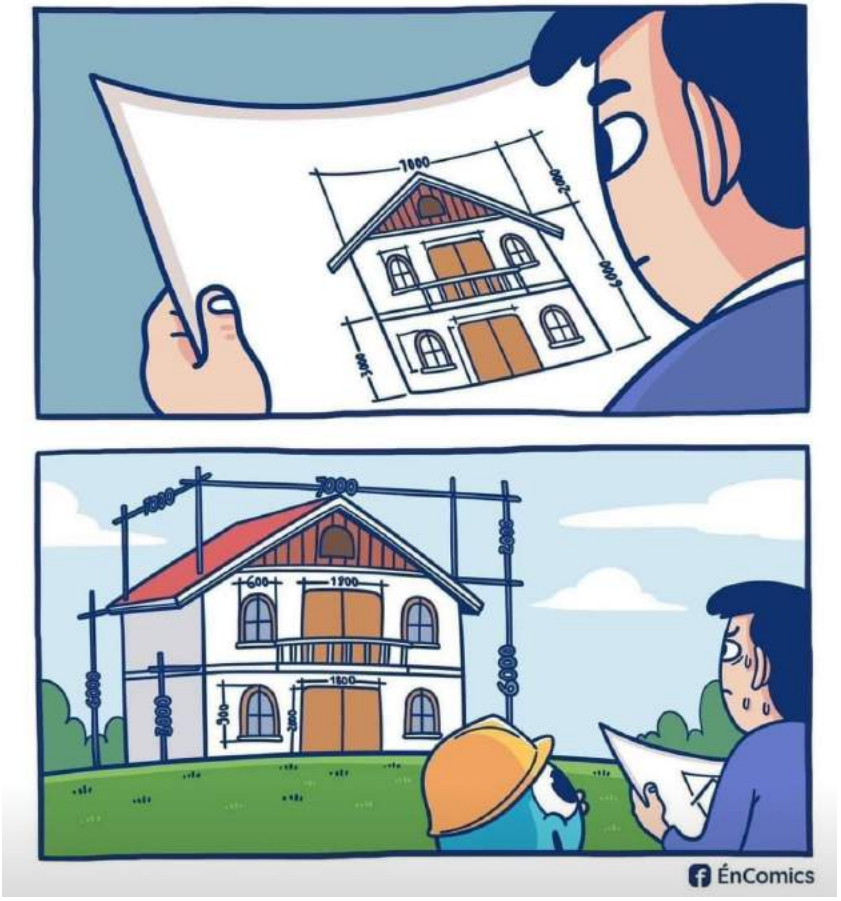
Class Activity – Previous year’s village implementations

City exercise with waterfall - feedback

What was the experience like?
(as observer/designer/builder)

How did it feel to hand over your final product to the mayor/customer?

What did you find challenging about this game?



https://thisiszone.medium.com/using-lego-to-show-the-advantages-of-an-agile-approach-to-software-development-3eda6e5c2114

Some challenges (traditional process)

- Different people doing analysis, design and development can make it less effective (throwing it over the wall)
- Team doing analysis and design might not know anything about the tools that development teams are using
- Requirements prescribtive, hinder creativivty, reduce team autonomy and don’t focus on underlying objective / customer goal
- Even when they are precise, they are often not sufficient to get a shared understanding

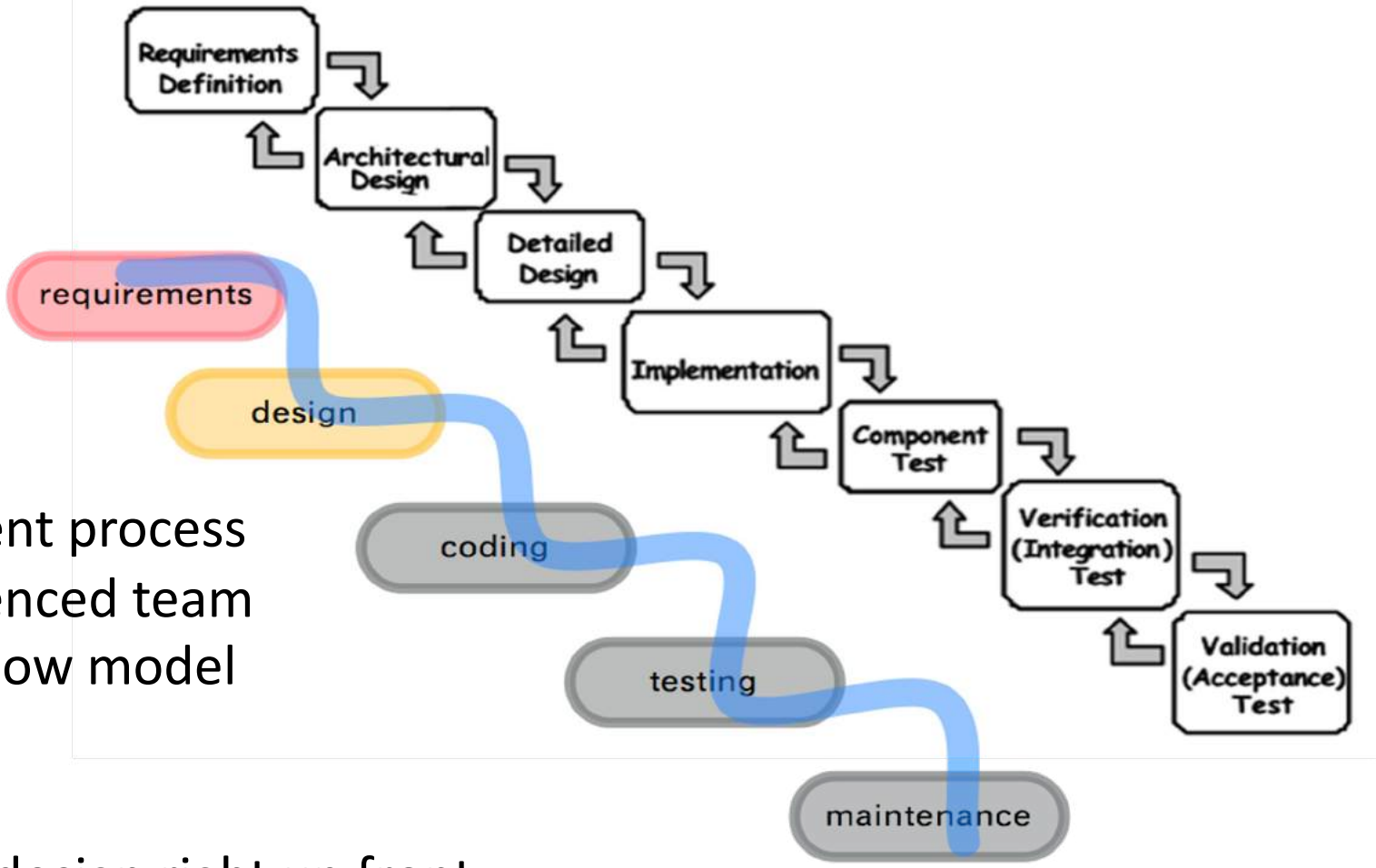
32

Some more challenges

- What customers **want** might not be what they **need**.
- What they want might **not be the most efficient** nor cost-effective solution to meet their need.
- What they want might **not be achievable with the available resources** (time, budget, expertise).

33

Waterfall



+ Good for well-understood but complex projects
    Tackles all planning up front
    No midstream changes = efficient process
+ Provides support for an inexperienced team
    Orderly, sequential, easy-to-follow model
    Reviews at each stage

- Difficult to get requirements and design right up front
- No sense of incremental progress
- Integration only at the end
- Final result not necessarily client-driven

Waterfall scenario (not unusual timeline)

Requirements: 18 months. Soliciting customer feedback, creating high fidelity mockups, validating requirements.

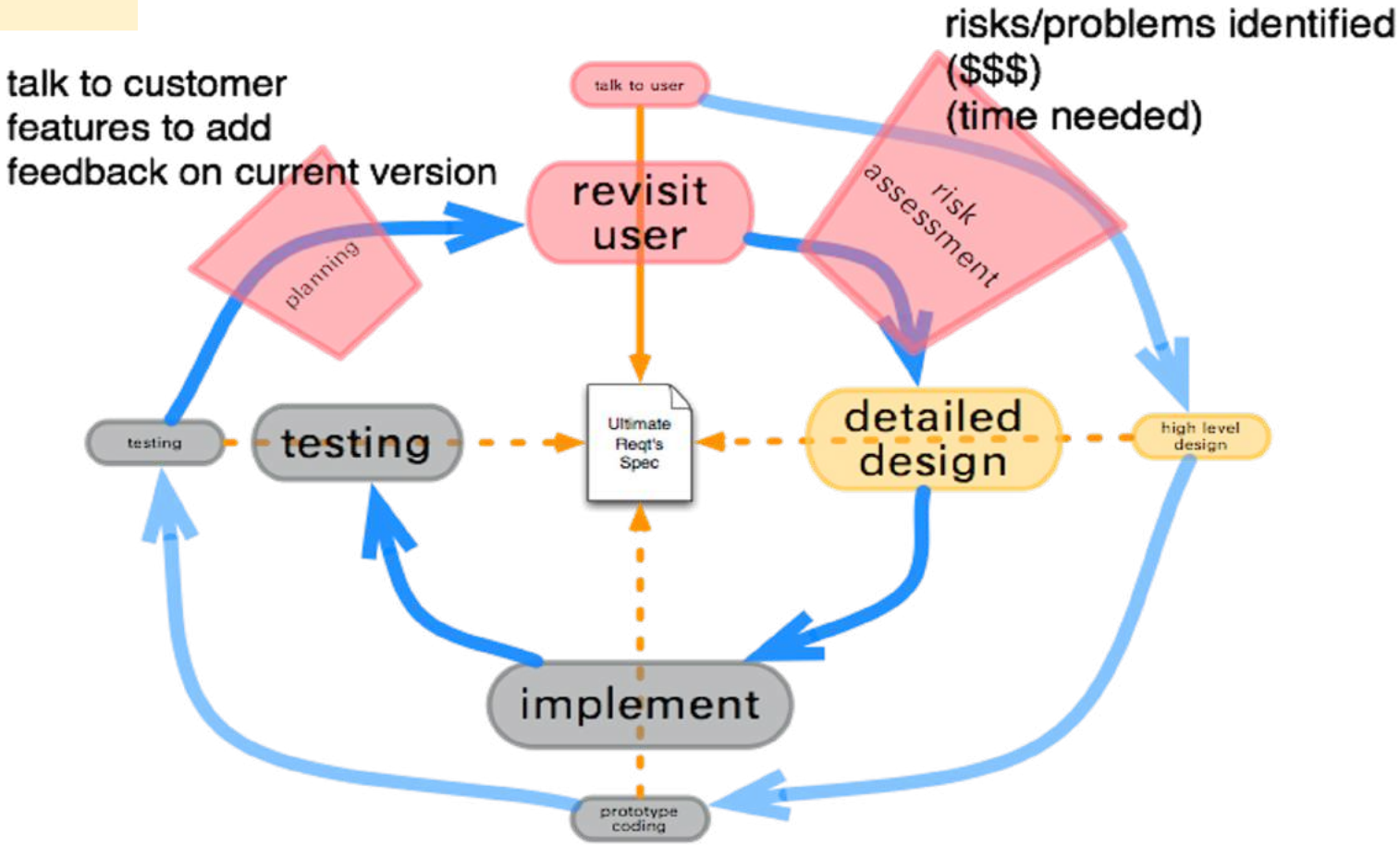
Design: 12 months. Deriving high-level architectural description and all design info and documentation.

Implementation: 18 months. Constructing system components matching design that performs as in the requirements.

Verification: 12 months. Ensure overall system performs as specified in the requirements. Refine implementation as needed.

Maintenance: 15+ years after v1. Keeping the system going amid OS / security / device changes.

Spiral



Risk analysis: identify risks and alternatives, use of prototypes, simulations and analysis software; can require highly specific expertise

Spiral scenario

Planning & Requirements: 3 mo —>

Risk Analysis & Design: 3 mo —>

Build: 3 mo —>

Test & Evaluate: 3 mo

Planning & Requirements: 3 mo —>

Risk Analysis & Design: 3 mo —>

Build: 3 mo —>

Test & Evaluate: 3 mo

Planning & Requirements: 3 mo —>

Risk Analysis & Design: 3 mo —>

Build: 3 mo —>

Test & Evaluate: 3 mo

Planning & Requirements: 3 mo —>

Risk Analysis & Design: 3 mo —>

Build: 3 mo —>

Test & Evaluate: 3 mo

History of agile – coping with change

Change is inevitable in large projects

- Business changes
- New technology or platforms
- Changing requirements
- New management
- Uncertainty and ambiguity at the beginning

Change leads to rework, new functionality or products being outdated the moment they hit the market

→ Have a feedback-loop

Agile Manifesto – 4 values

Individuals and Interactions	over	Processes and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to Change	over	Following a Plan

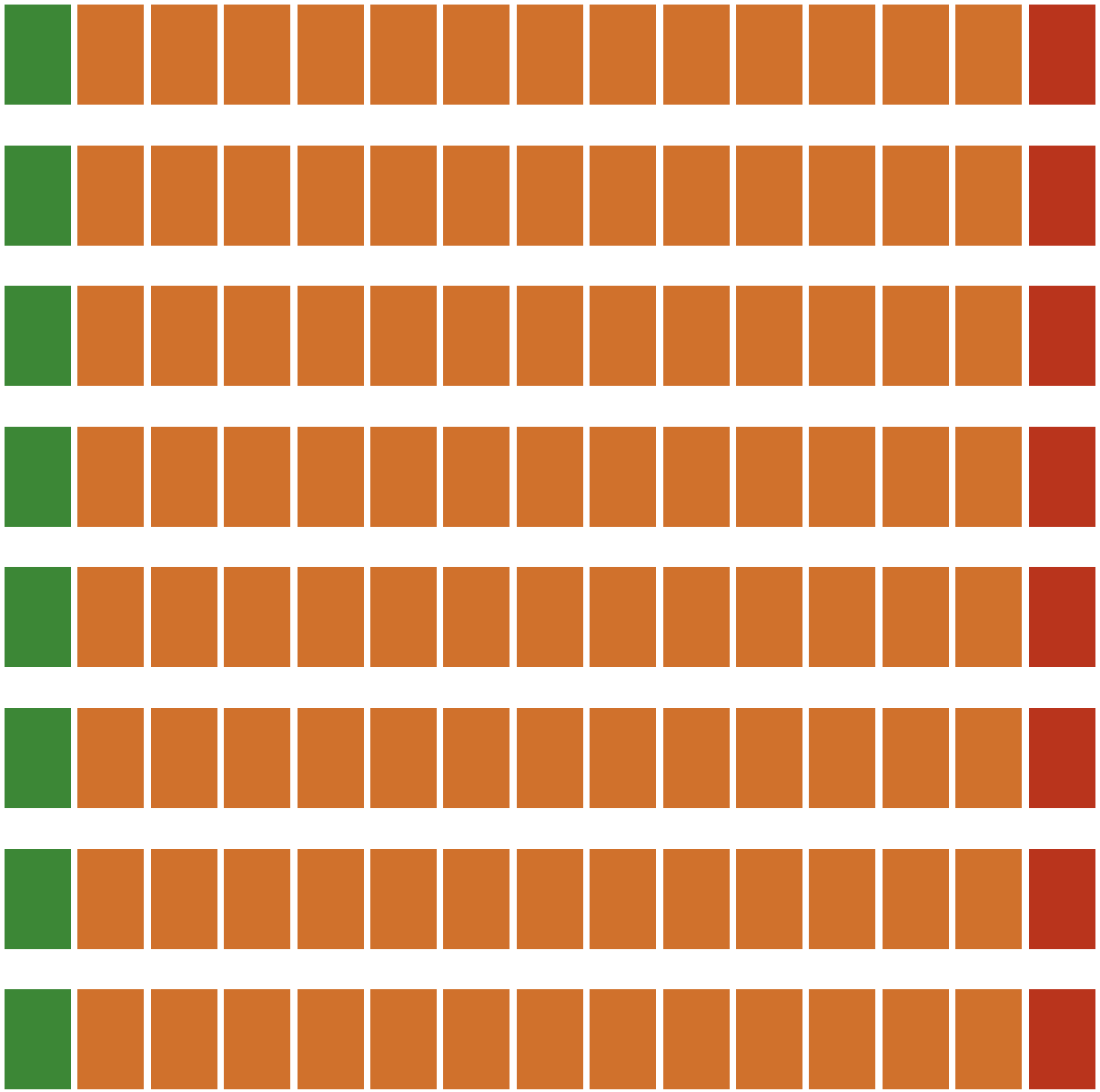
That is, while there is value in the items on the right, we value the items on the left more

agilemanifesto.org

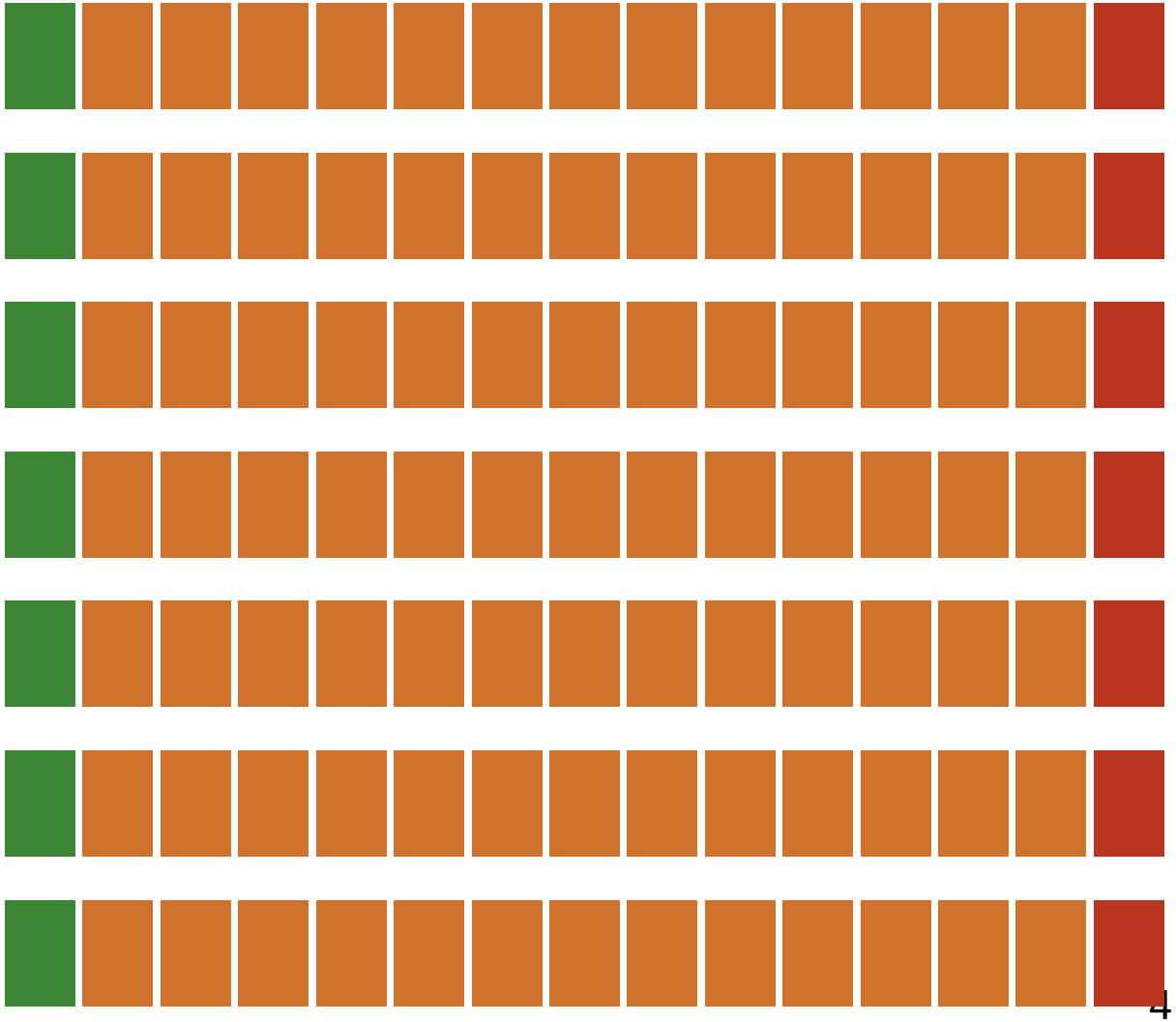
Does “agile” mean NO documentation and/or NO process?

Scrum scenario

Plan | Code & Test | Ceremonies



3 weeks



12 Agile principles

- Satisfy customer through **early and continuous delivery**
  - Heavily iterative & incremental
- Deliver **working software frequently** (short iterations)
  - Primary measure of progress
  - Focuses on small number of prioritized requirements
- **Daily collaboration** between stakeholders (business and developers)
- Welcome **changing requirements**

12 Agile principles cont'd

- Build project around **motivated** individuals and provide supporting environment
- Teams are **self-organizing** (and cross-functional)
- High value on **face-to-face** conversations
- Team **reflects** regularly on how to improve and adjusts
- Teams work at rate that can be maintained for entire project duration (**sustainable pace**)

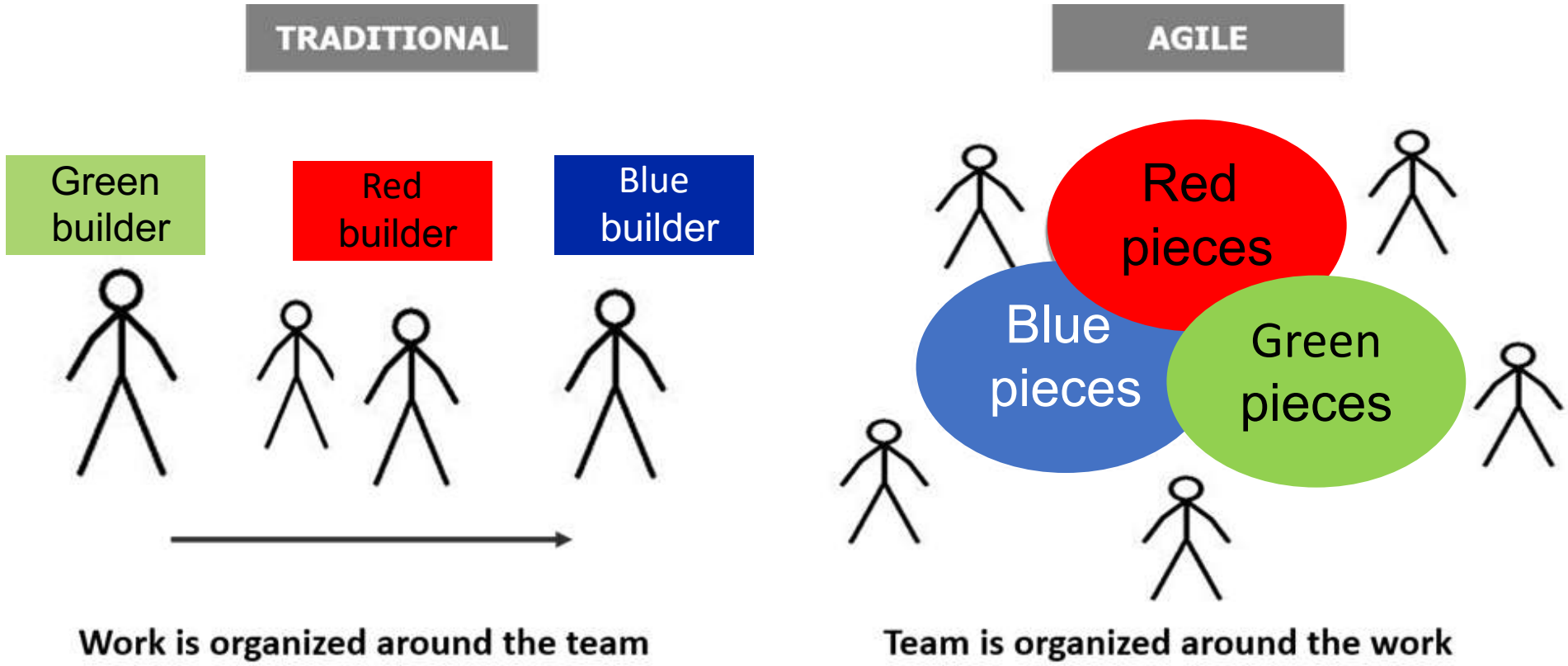
Question on self-organizing teams

In the Lego exercise (implementation/builders):

Were both teams self-organized and cross-functional?

What's the effect of that?

Self-organized cross-functional teams



https://www.tastycupcakes.org/2016/05/5668/

45

12 Agile principles cont'd

- **Simplicity** is essential (art of maximizing the amount of work not done)
- Continuous attention to technical excellence and good design enhances agility
- **Working software** is the primary measure of progress

Class Activity: developing a game (10mins if there's time) [https://bit.ly/3STWA2q]

In teams of 4-5: Imagine you are building an application for a multi-player online game, such as Minecraft (you can pick freely though), address the following questions:

1. What are the major risks and challenges you are facing?
2. What are the main stakeholders for developing such an app?
3. Which process would you use for the development?
4. If you have 24 weeks to develop this app, how would you split up the time?
5. Does the process choice address some of the risks/challenges stated for question 1 and if so which ones?

Agile Methods

Learning Objectives

Be able to:

- Compare different process models
- Explain the terms, practices and values for agile process methods

Agile Methods

Extreme Programming (XP)

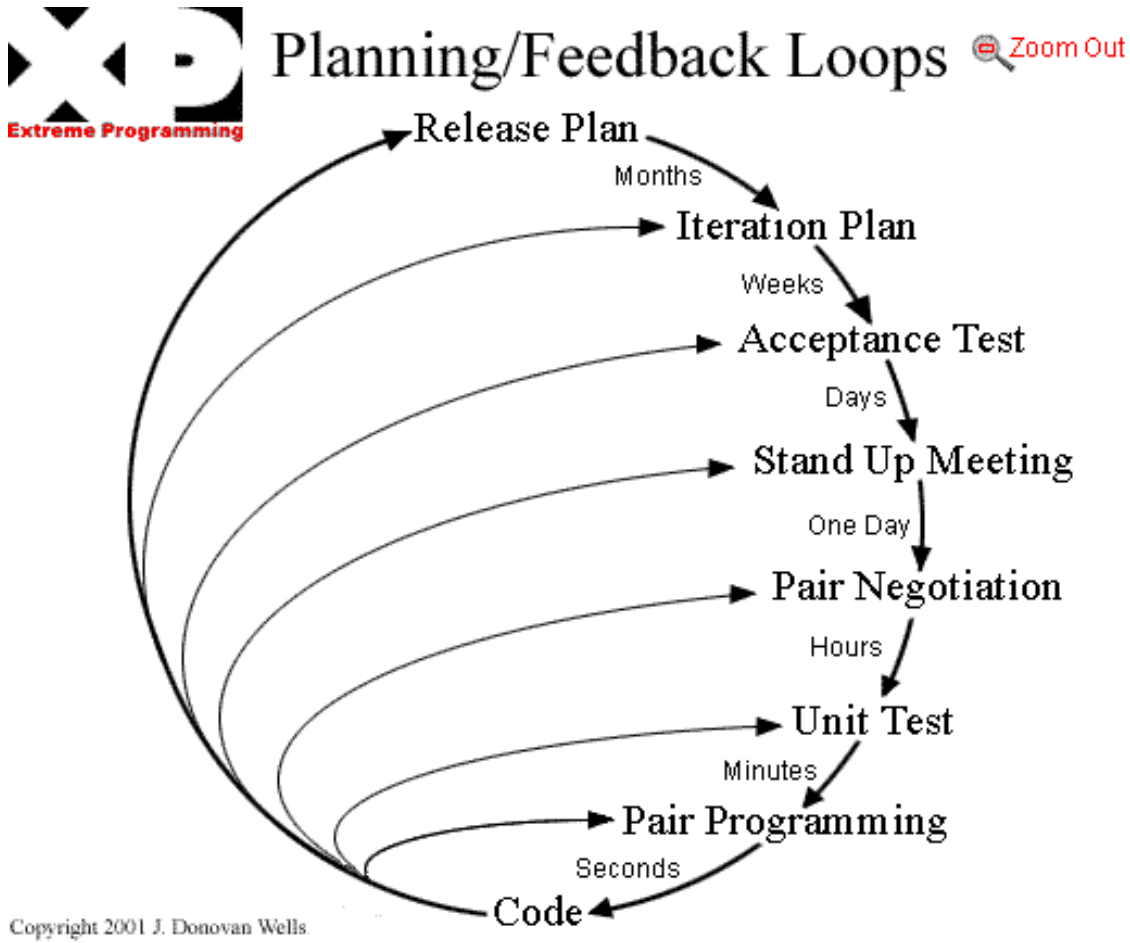
- Specific practices – customer driven development, daily builds, small teams

SCRUM

- Project management approach, relying on self-organizing teams

Several others – Crystal Clear, Lean, FDD, Spotify Model …

What XP looks like



XP

Five principal values

Communication: common metaphors, frequent verbal communication, customer involvement

Simplicity: do the simplest thing that could possibly work, then refactor

Feedback: from the code (unit tests), the customer (co-location), the team (planning game)

Courage: be willing to throw things away

Respect: don't do things that make work for others

XP – 12 core practices

Fine scale feedback

- Pair programming
- Planning game
- Test driven development
- On-site customer / whole team

Continuous process

- Continuous integration
- Refactoring
- Small releases

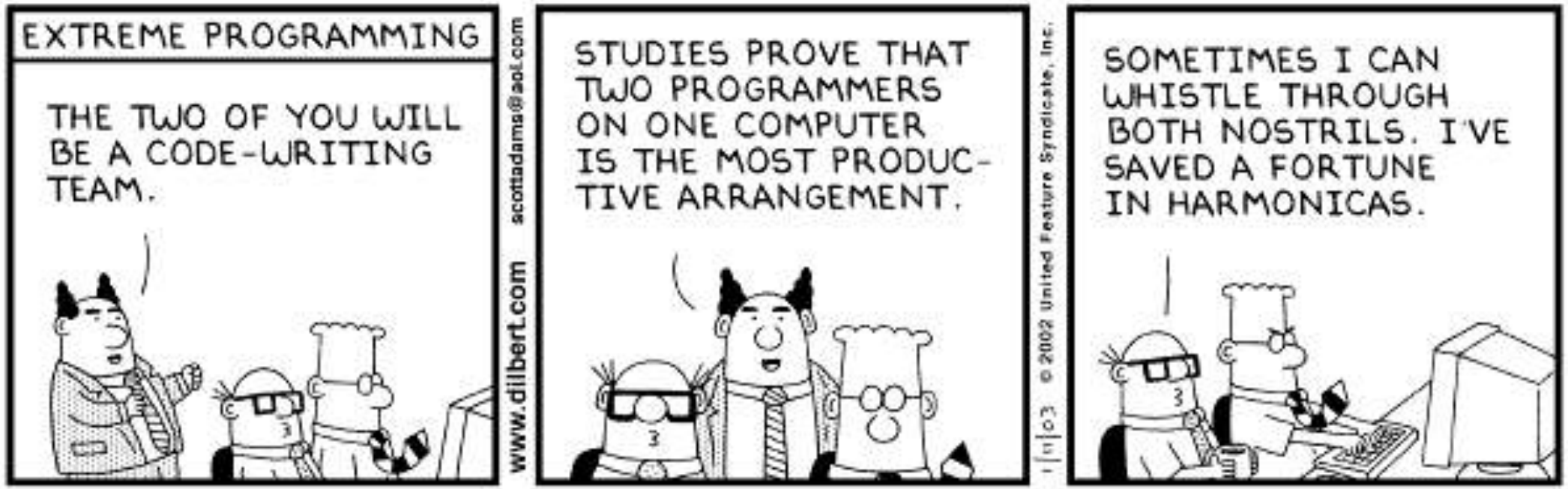
Shared understanding

- Coding standards
- Collective code ownership
- Simple design
- System metaphor

Programmer welfare

- Sustainable pace

XP – Pair Programming



Test-Driven Development (TDD)

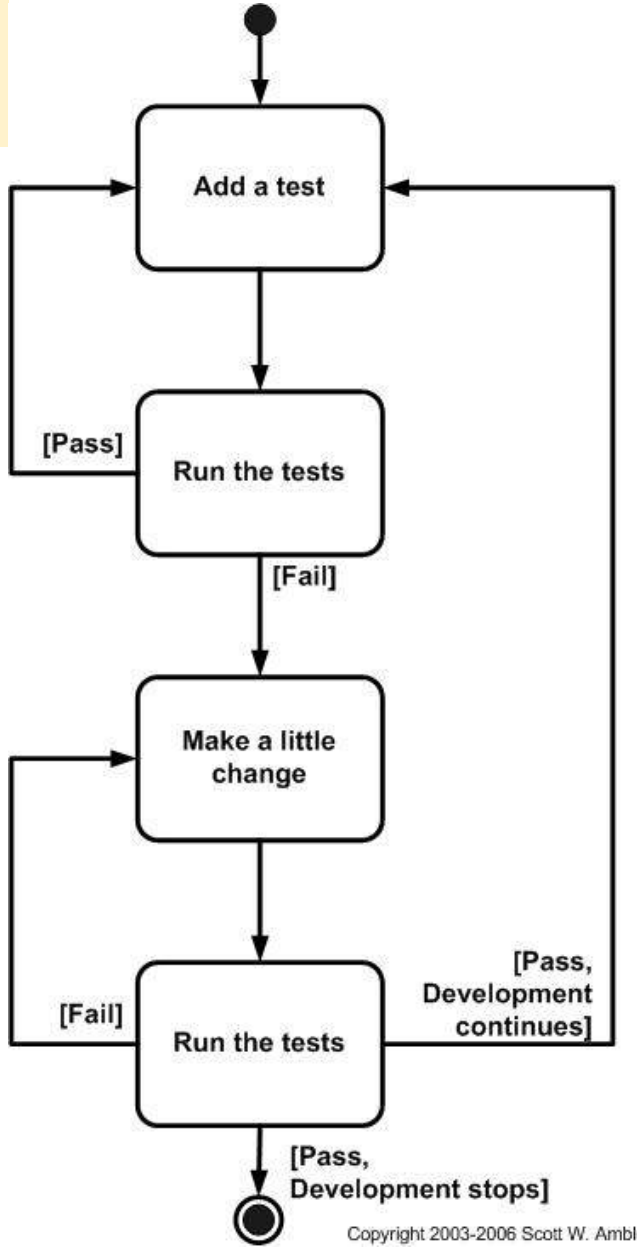
Test cases are written first

- Cover new functionality or improvement

Then the necessary function is implemented

Code is “complete” when all tests pass

Refactor before adding feature if design could be better



SPIKES!

https://www.scaledagileframework.com/spikes/

Extreme programming introduced the concept of “Spikes” -- short, intense, activities that precede all the development iterations (or sometimes fit in between them, but that’s not typical). Here are some common examples of spikes.

Architectural Spike: Right when the product is being devised, decide on the high level and medium level architectures. Is it an iOS app? Is it a web app? What language/framework/etc will we use? And then, what are the major collaborators and subcomponents of each of the major components (client, server).

User Interface Spike: Before beginning to develop anything the user will see, decide on the look and feel, the UI framework, and plan for UI expansion if it is foreseen. These can often involve some initial lightweight prototyping, to help work out the user stories, and set the stage for how the product is designed from a user perspective.

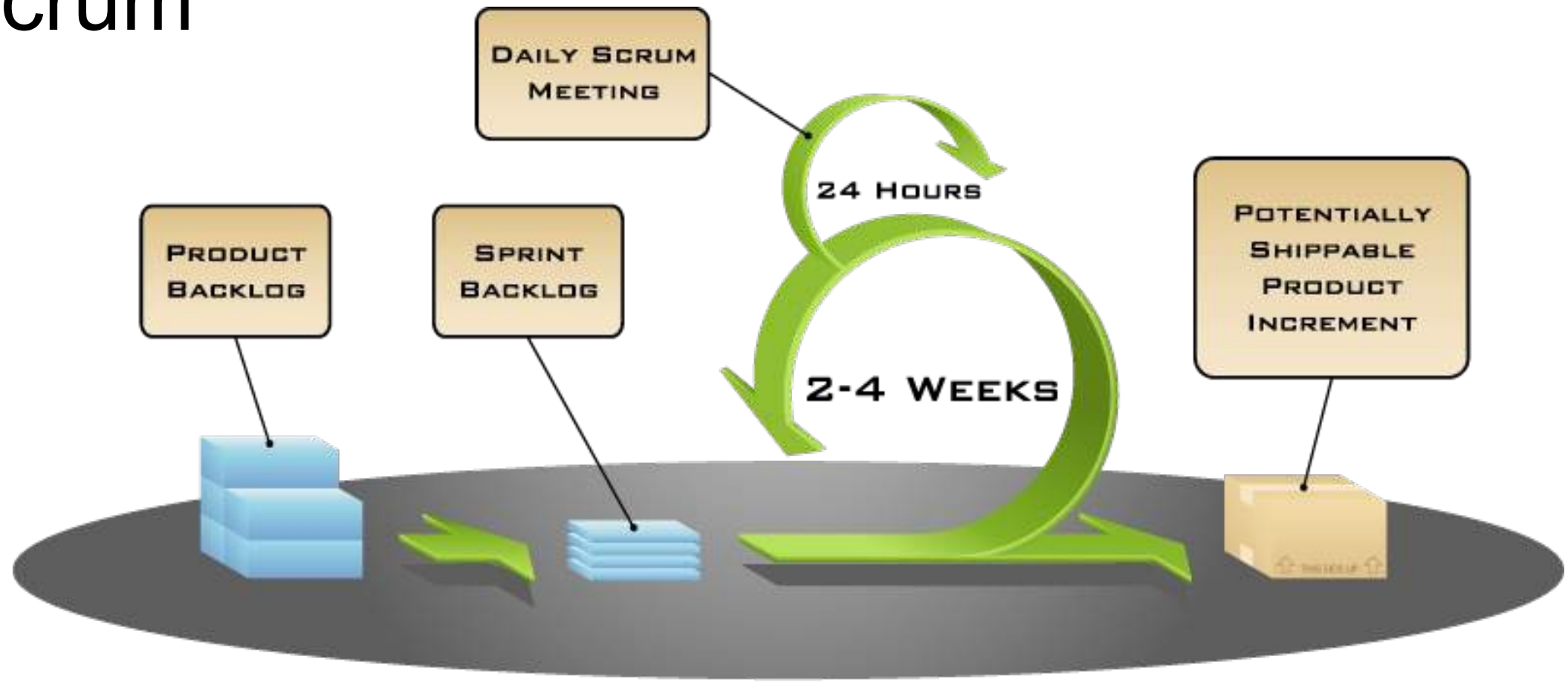
Scrum

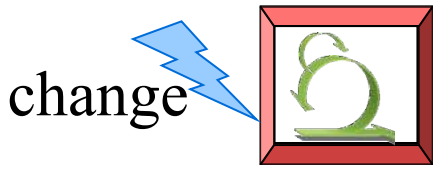
Management framework for incremental product development

- Self-organizing, cross-functional teams
- Product progresses in a serious of two- to four-week (fixed length) iterations: ***sprints***
- Every iteration produces a potentially shippable (properly tested but not complete) product
- Requirements are captured as items in a list: ***product backlog***
- The business sets the priorities. Teams self-organize to determine best way to deliver highest priority features.

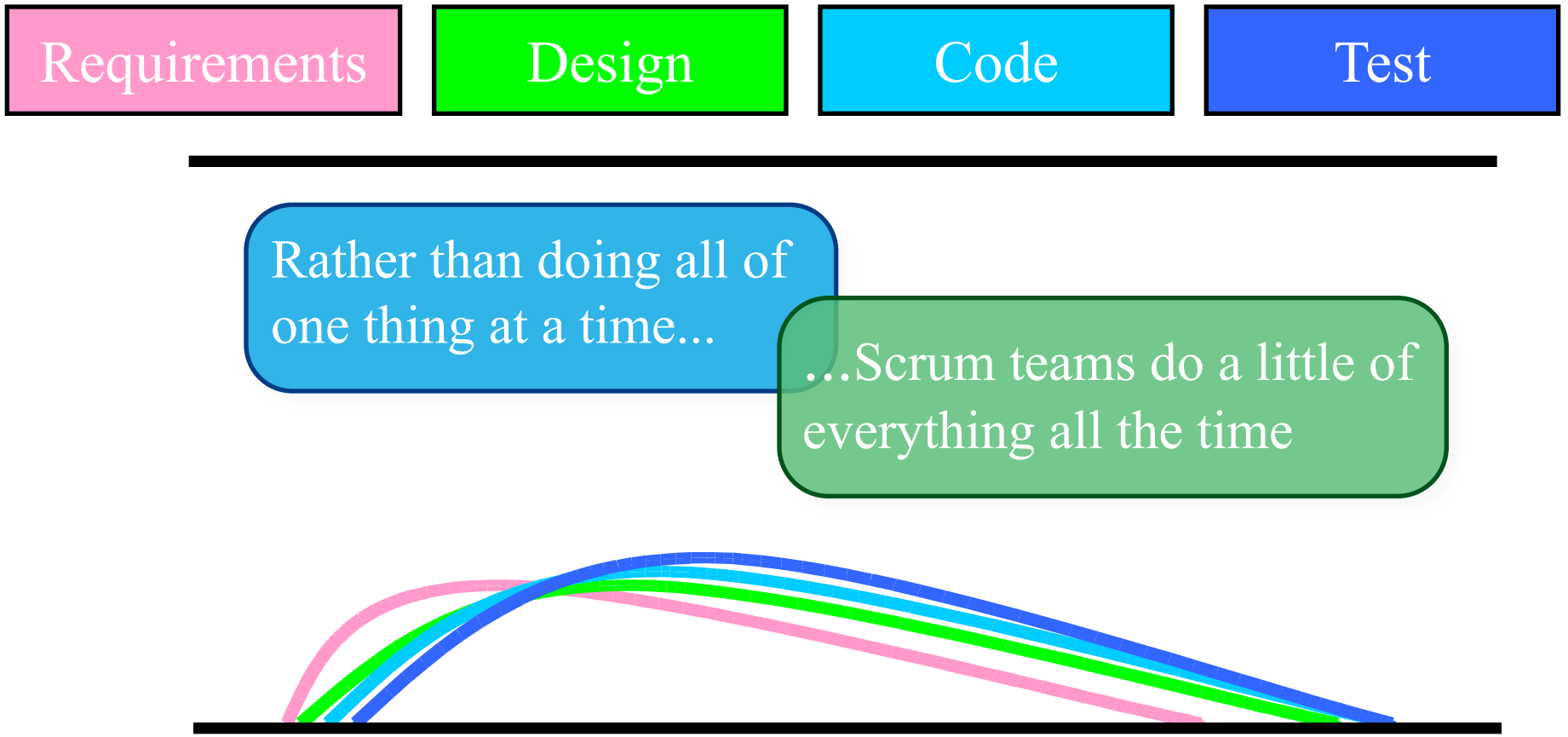
- No specific engineering practices prescribed

Scrum





Sprint (overlapping vs sequential)



Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

Scrum Roles

Product Owner

- Defines features of the product
- Prioritizes features according to market value
- Adjust features and priorities every iteration, as needed

ScrumMaster

- Facilitates Scrum process
- Helps resolve impediments
- Shields team from external interferences
- NOT the manager

Team

- Self-organizing, self-managing, cross-functional
- Developers, designers, testers, …
- 7 (+/- 2) people

Scrum Artifacts

Product Backlog

- prioritized list of requirements/PBIs that describes all desired functionality

Product Backlog Item

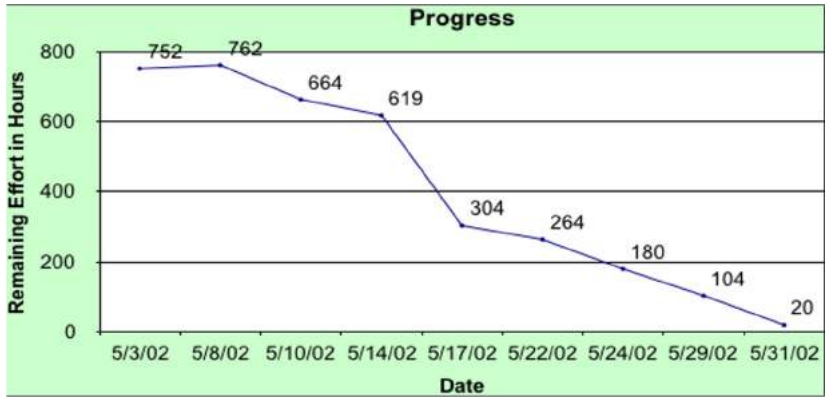
- specifies a customer-centric feature (User Story form) – effort estimated by team, business value and priority estimated by Product Owner

Sprint Backlog

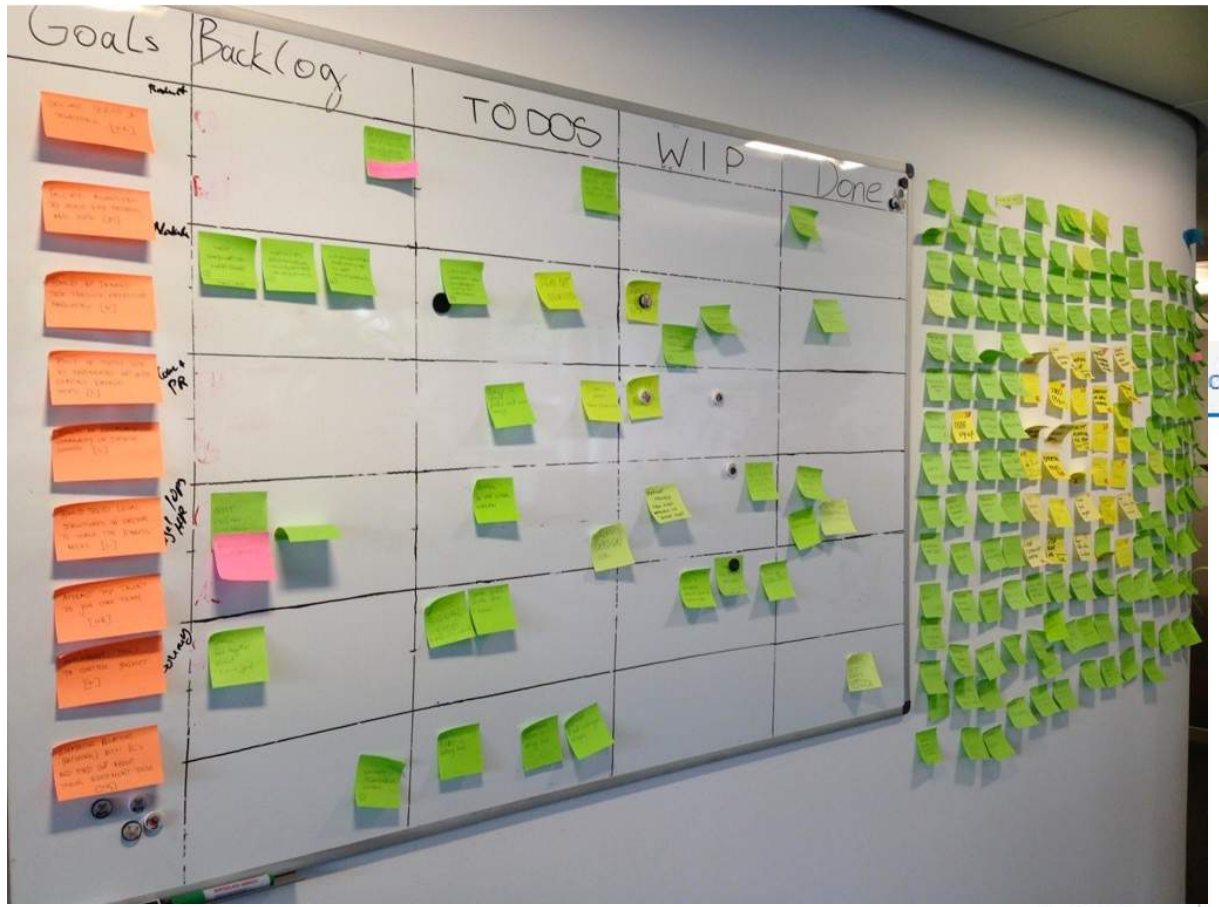
- contains list of sprint items (user stories) that are negotiated by team and product owner from the Product Backlog for the sprint broken down into specific tasks

Burndown Chart

- Total remaining team task hours within one sprint
- (velocity – progress rate: amount of, e.g. story points completed per iteration)



Scrum Task Board



cklogs Queries Plans

Fabrikam Fiber Team Sprint 1

June 11 - June 29
9 work days remaining

Backlog Board Capacity

Group by Backlog items Person All

Collapse all

To do 14 h

In progress 14 h

Phone sign in
Raisa Pokrovskaya 14 h
State Approved

Improve build definitions
Johnnie McLeod 8

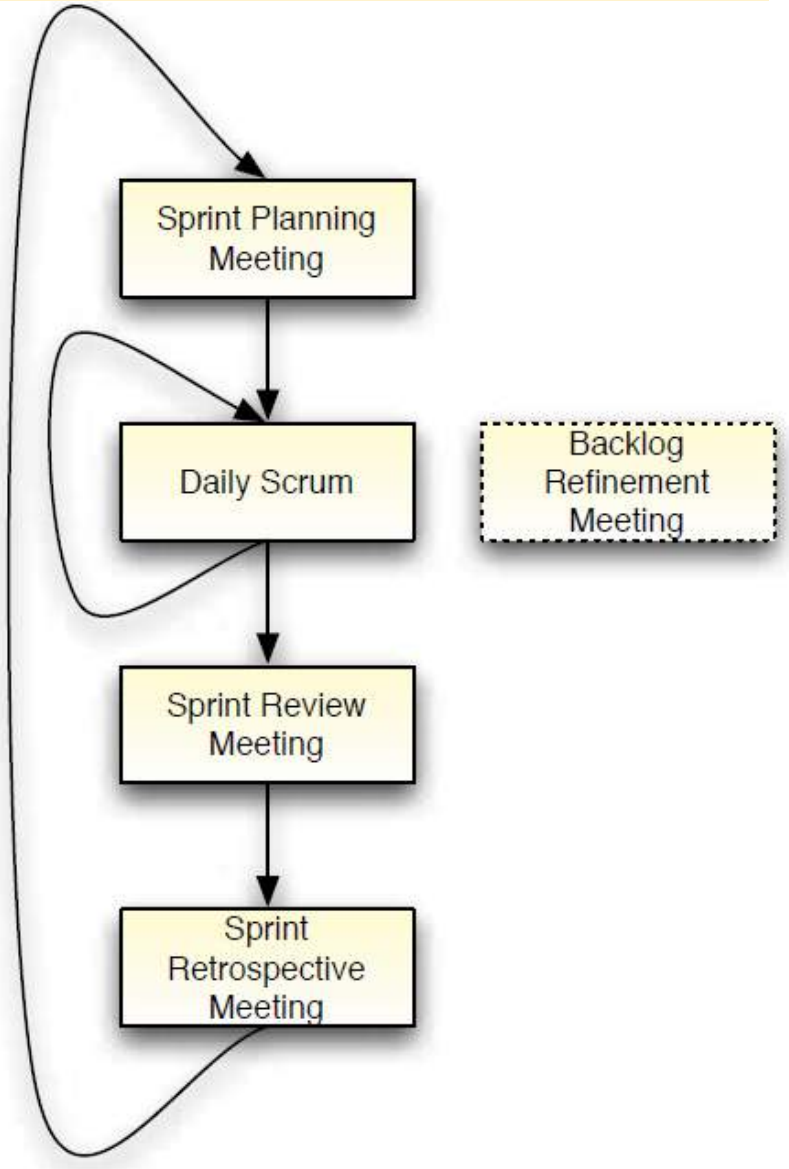
Research slow respo
times
Christie Church

Hello World Web Site
Jamal Hartnett
State Committed

Design welcome screen
Christie Church

Sprint 2
Sprint 6
Sprint 3
Sprint 9

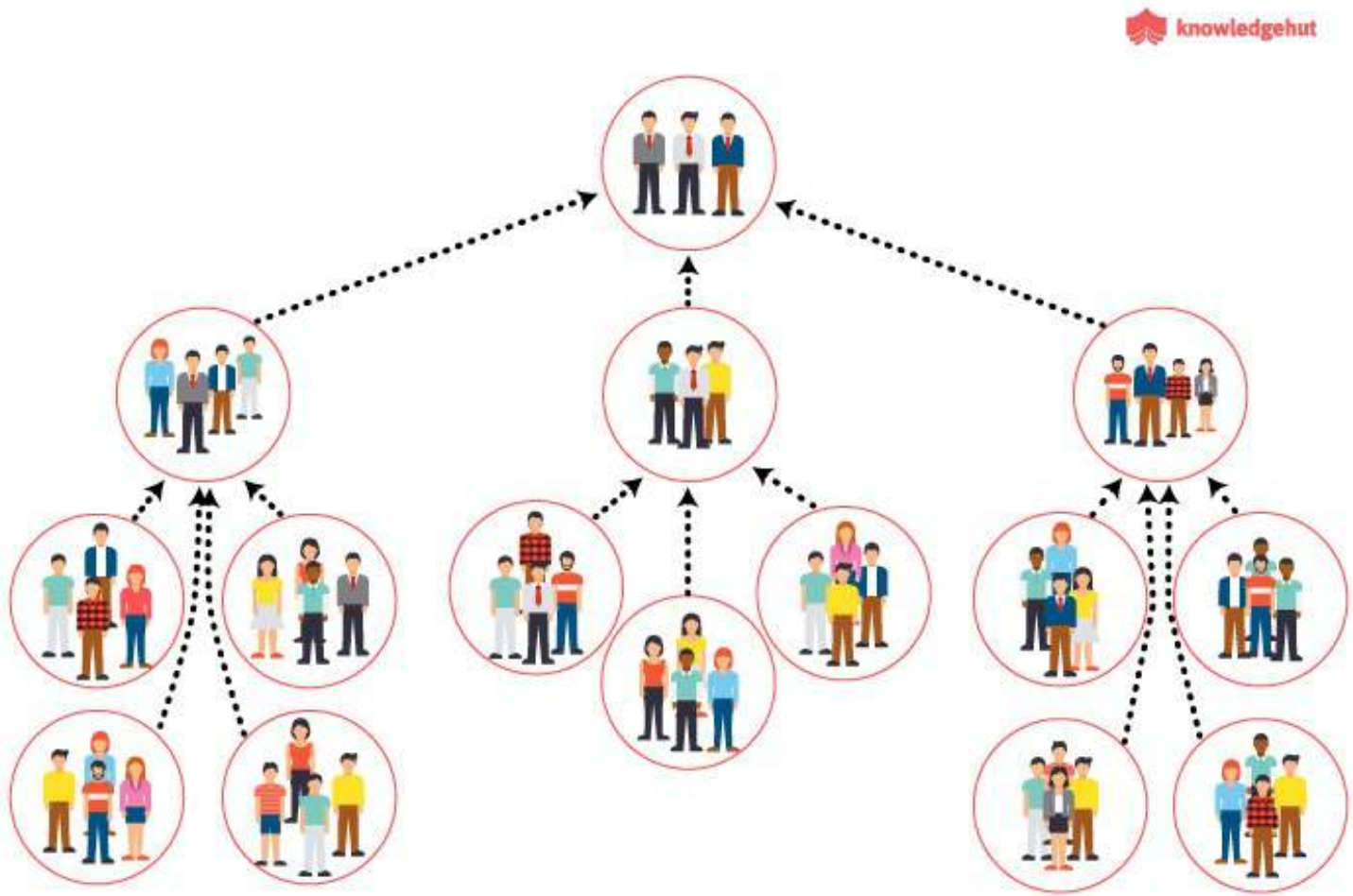
Scrum process



Daily Scrum

- Is **NOT** a problem solving session
- Is **NOT** a way to collect information about who is behind the schedule
- **IS** a meeting in which team members make commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the Team
- 15 minutes long (ish)
- Everyone says:
  - What did you do yesterday?
  - What will you do today?
  - What are the obstacles to progress
- Task estimates may need to be adjusted

Scrum of Scrums (Scrum at scale)



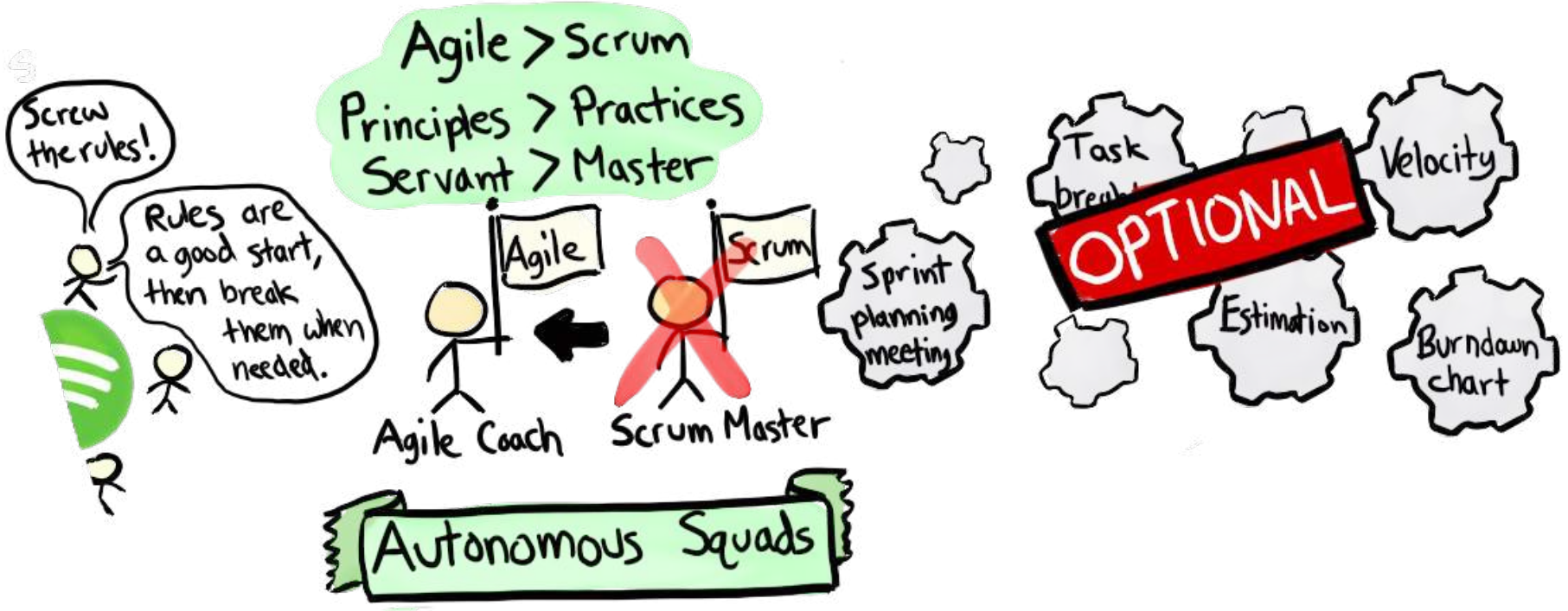
https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-of-scrums

Agile as a mindset – spotify model

Pictures from the 2 parts on spotify's engineering culture

- https://engineering.atspotify.com/2014/03/27/spotify-engineering-culture-part-1/
- https://engineering.atspotify.com/2014/09/20/spotify-engineering-culture-part-2/
- https://www.atlassian.com/agile/agile-at-scale/spotify
- https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

Agile as a mindset – spotify model



Pictures from the 2 parts on spotify's engineering culture

- https://engineering.atspotify.com/2014/03/27/spotify-engineering-culture-part-1/
- https://engineering.atspotify.com/2014/09/20/spotify-engineering-culture-part-2/
- https://www.atlassian.com/agile/agile-at-scale/spotify
- https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

Disclaimer on applying Agile in practice

It’s not about implementing an Agile method ***exactly*** **according to theory**

It’s okay to **cherry pick concepts**

It can **take years** to implement and effectively apply Agile in a company

It’s **about the mindset**
(flexibility, adaption, learning, growing)

66

Activity & Questions

Learning Objectives

Be able to:

- Choose an appropriate process model and justify it given a case study / project

Class survey

Kahoot!

Questions: (from previous courses)

1. When to use which process and why?
2. Can process methods be combined?
3. Who decides on the process?
4. What happens if unexpected events occur during sprints, i.e. how can they be accounted for?
5. What if you can’t finish a sprint item during a sprint?
6. What if a sprint item is too long for a sprint?
7. How to estimate time for projects / sprint items?
8. Can the Scrum master also be the product owner?
9. Scrum: is product owner inside or outside the company or inside the development team?

Quiz Answers

Quiz

1. Imagine you are the CTO of a startup that is about to develop a mobile phone application that is a quiz game in which people compete against each other based on daily quiz questions. You are debating whether you should choose a Waterfall or an agile process for the development that is planned for the next 12 months. Which of the following points are reasons for choosing an agile process over a Waterfall process for the project.
   a) The design for the application is well known due to your prior expertise.
   b) Since you want to ensure that you will have a lot of users, you are most likely going to add more and new features during a planned initial testing phase with some test users.
   c) While you are going to use the newest library for login authentication for your game, this might be updated in 6 or more years, after you have rolled out the game.
   d) You have an inexperienced group of developers to develop the application.

Quiz

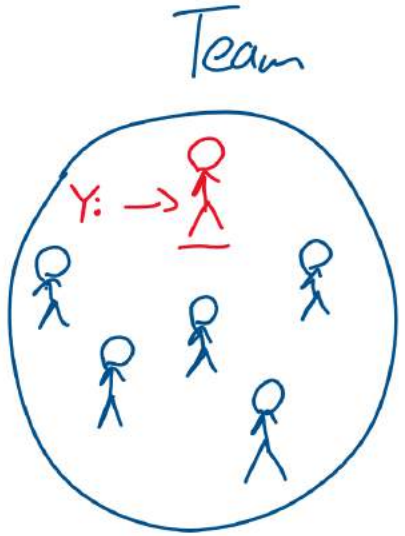
2. Select all that apply to the Spiral Software process but not really to the waterfall process:
    - The ability to add additional features later on in the project
    - Doing maintenance tasks on the project like upgrading a library
    - Getting customer feedback on the project
    - Building a prototype

Quiz

3. Which of the following best describes the primary goal of software processes?
    - To ensure software is developed using only modern methodologies.
    - To structure activities for successful and maintainable future systems.
    - To eliminate the need for documentation in software development.
    - To strictly adhere to traditional development methodologies like Waterfall.

Quiz

4. In Scrum there are defined roles for some people included in the process. Based on the following figure/image, which of the following combinations fits the Scrum process roles the best?
    - X: Sprint Planner, Y: Team Lead
    - X: Team Lead, Y: Sprint Planner
    - X: Product Owner, Y: Scrum Master
    - X: Scum Master, Y: Product Owner



Additional Resources & Next Week: Specification

Things to do THIS WEEK

- **Listen to module videos**
- additionally provided reading also captures most of the video material

- **Fill in Quiz (next Wednesday, 10:00 – 10:20am)**

Additional optional resources for this week:

- Reading on process https://faculty.washington.edu/ajko/books/cooperative-software-development/process

Lego exercises

- https://thisiszone.medium.com/using-lego-to-show-the-advantages-of-an-agile-approach-to-software-development-3eda6e5c2114

- https://www.linkedin.com/pulse/throw-over-wall-lego-game-which-can-shift-waterfall-mindsets-snook/

Misconceptions about Agile

Some ***wrong facts*** about Agile

- Agile = no process
- Agilists don’t document
- Agilists don’t design
- Agile leads to poor quality software
- Agile works only with small teams
- Agile ignores business concerns
- Extreme Programming is the only agile process

Agile

- Embraces change
- Early increments act as prototype to elicit requirements
- Constant customer involvement and validation
- Lower risk of overall project failure

- Can fallback into cowboy coding
- Increases risk of feature creep: adapting is good but you need to draw the line
- Projects can become ever-lasting without a clear end
- Not appropriate when comprehensive requirements spec is required as contract up front
- Requires close customer involvement
- Not everyone works well with constant change

Test-driven development

Advantages

- Tests define units of work
- Tests document expected functionality
- Tests are contracts for methods
- Later addition can be “regression” tested
- Failure indicates breakage
- Small steps

Drawbacks

- Designing good tests is really difficult
- Can require extra time during implementation
- Hard to do for real systems
- Difficult to write tests before implementing system
- GUIs can be hard to test