Skoltech
Skolkovo Institute of Science and Technology

Software Development Models

Foundations of Software Engineering

FSE v2020.1, Block 1 Module 2

Alexey Artemov, Fall 2020

Lecture Outline

Skoltech
Skolkovo Institute of Science and Technology

§1. S/w development models overview [10 min]

1.1. Why models for software development?

1.2. Model classification: predictive vs. adaptive, incremental and iterative

§2. Predictive models: Waterfall [15 min]

2.1. Waterfall, V-model, Sashimi

2.2. Incremental waterfall models

Lecture Outline

Skoltech
Skolkovo Institute of Science and Technology

§3. Iterative models [15 min]

3.1. The spiral model

3.2. Unified process and RUP

§4. Agile models [15 min]

4.1. Challenges of traditional approaches

4.2. Agile manifesto: values and principles

4.3. Scrum Agile framework

4.4. Kanban Agile framework

Skoltech
Skolkovo Institute of Science and Technology

The Goal: Think Before You Code

Skoltech
Skolkovo Institute of Science and Technology

§1. Software development models overview

Skoltech
Skolkovo Institute of Science and Technology

Why models for software development?

Skoltech

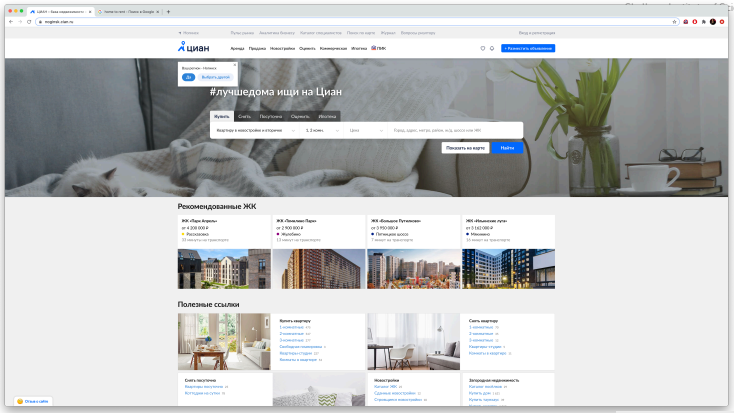


Image Credit: Alexey Artemov

Skoltech

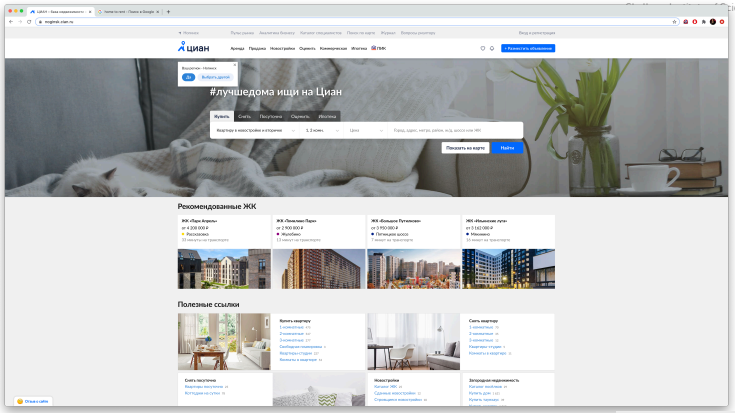


Image Credit: Alexey Artemov

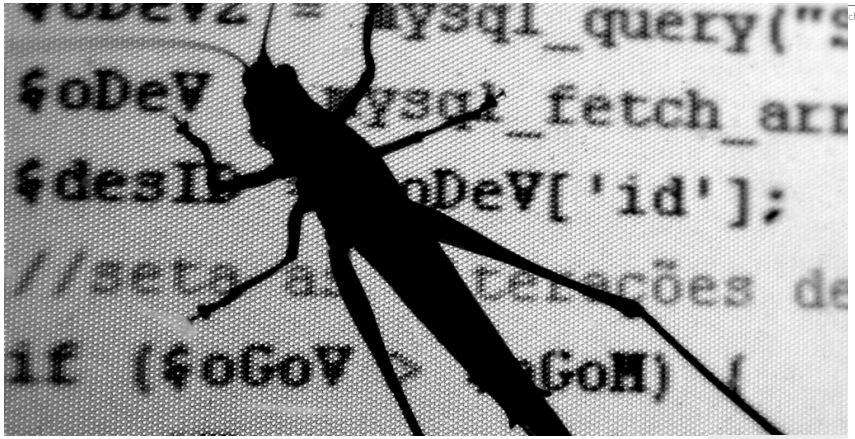


Image Credit: Guilherme Tavares

Image Credit: wikimedia commons

§1. S/w development models overview

1.1. Why models for software development?

- Address s/w engineering problems by adopting a methodology
- Goal for software development process models: create **a workflow that a team can use**
- Increase chances of produce high-quality software reasonably close to on time and within your budget

- Why many models?
  - Emphasize one part of development or another
  - Organisations are different
  - Many models so you can select the right one

How to classify software development process models?

§1. S/w development models overview

1.3. Model classification: predictive vs. adaptive, incremental and iterative

- How to classify models? Predictive vs. Adaptive
- **Predictive:**
  - You have a good understanding of the requirements (customers know exactly what they want)
  - You go through all phases, e.g. development and testing, in one shot

Image Credit: torange.biz

§1. S/w development models overview

1.3. Model classification: predictive vs. adaptive, incremental and iterative

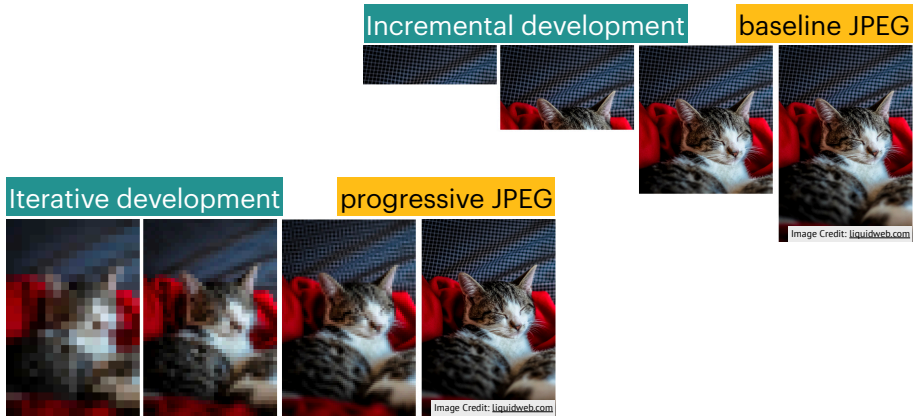
- How to classify models? Predictive vs. Adaptive
- **Predictive:**
  - You have a good understanding of the requirements (customers know exactly what they want)
  - You go through all phases, e.g. development and testing, in one shot
- **Adaptive**
  - Customers have an idea
  - Developers build something to start
  - Customers provide feedback

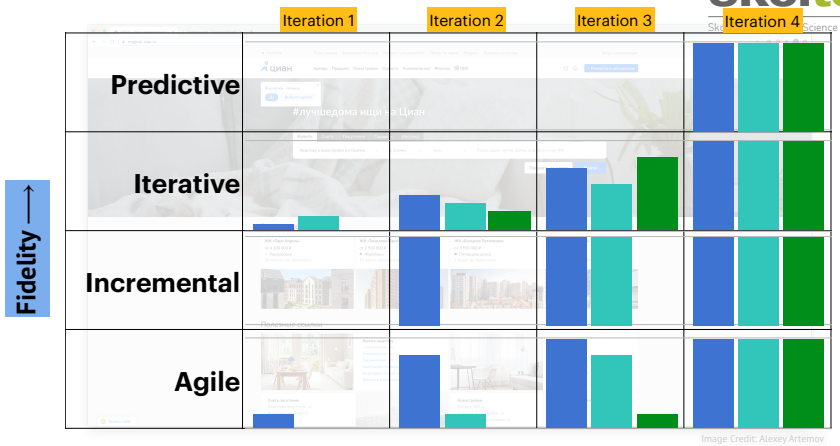
Image Credit: JISC

§1. S/w development models overview

1.3. Model classification: predictive vs. adaptive, incremental and iterative

- How to classify models? Predictive vs. Adaptive or Incremental vs. Iterative
- **Incremental:**
  - You have a good idea of what you need
  - Build in increments
- **Iterative:**
  - Build on top of simpler existing products
- Iterative is not incremental





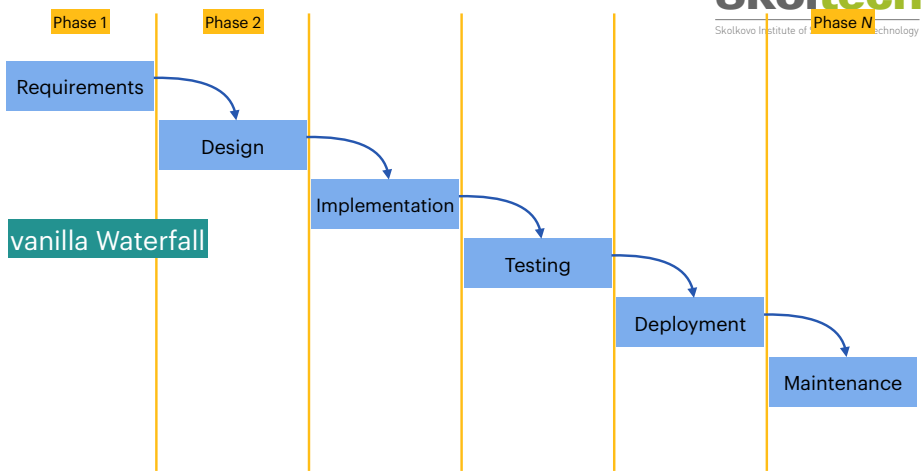
§1. Summary: s/w development models

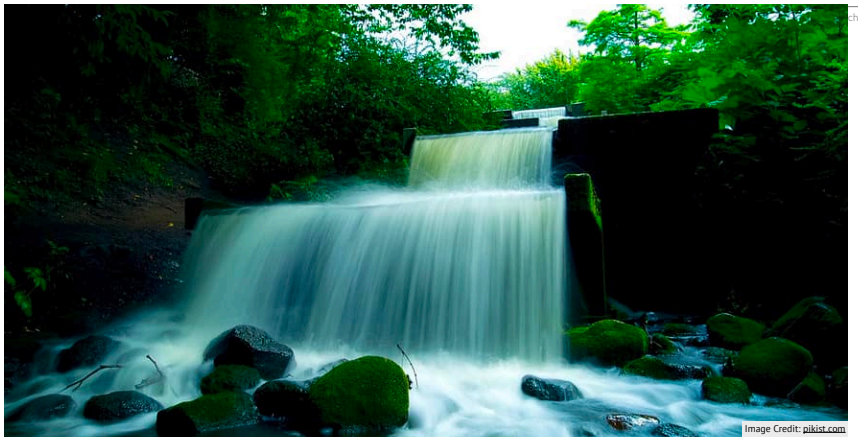
- Process models: ways to organise workflow of your team
- Increase chances of produce high-quality software reasonably close to on time and within your budget

- Predictive vs. adaptive
- Incremental vs. iterative
- Agile

§2. Predictive models: Waterfall

Waterfall, V-model, and Sashimi





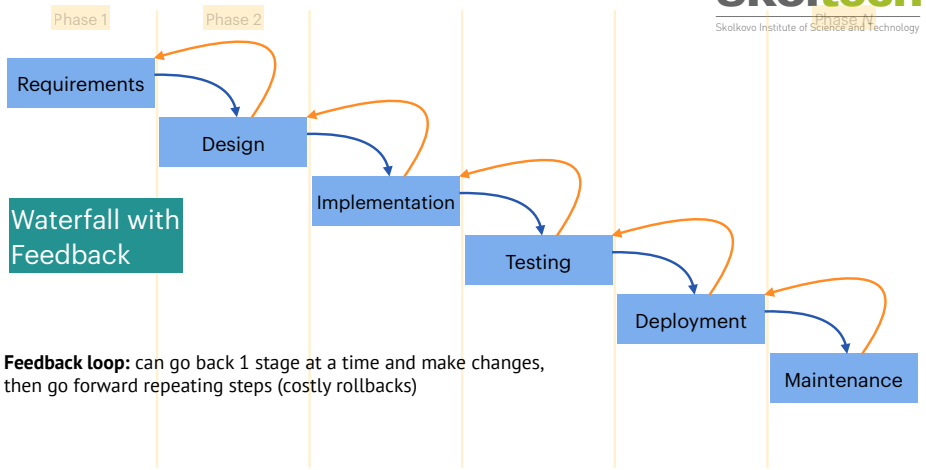
§2. Waterfall models

Skoltech
Skolkovo Institute of Science and Technology

2.1. Waterfall

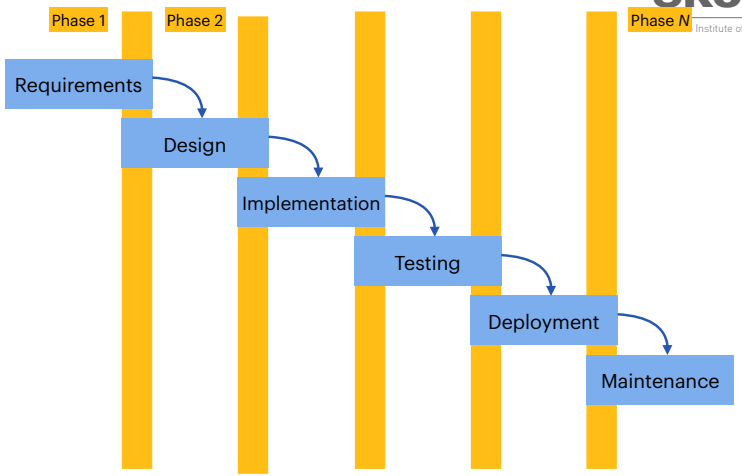
- **Waterfall** [works reasonably well when]:
  - Requirements are **known** very well, include **no unresolved high-risk** items, and **won't change**
  - Team has experience building something similar
  - Transition requirements -> product
- Pros:
  - Very **predictive** model (requirements known upfront)
  - Very efficient process (simple and robust)
- Cons:
  - Not flexible
  - No early release / value during development process

Skoltech
Skolkovo Institute of Science and Technology

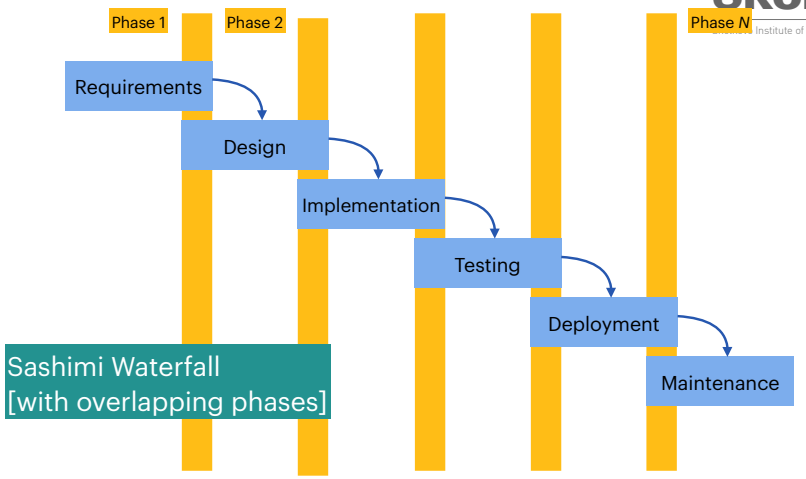


Feedback loop: can go back 1 stage at a time and make changes, then go forward repeating steps (costly rollbacks)

Skoltech
Skolkovo Institute of Science and Technology



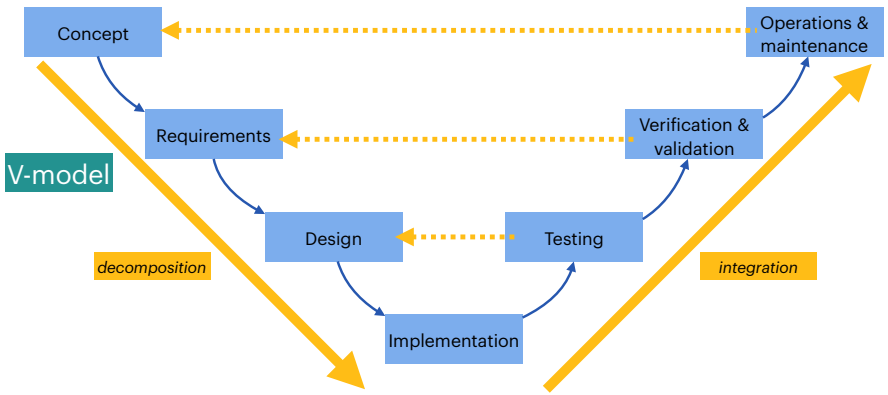
Skoltech



§2. Waterfall models

2.1. Sashimi

- **Sashimi** [works reasonably well when]:
  - Allow **overlap** between different phases during the lifecycle
    - Design starts before the requirements are fully finished
    - Architects start working on the completed subset of requirements, not waiting for the other parts
- Pros:
  - Mostly **predictive** model (requirements known upfront)
  - Shorten development time
  - People with different skills can start doing work simultaneously
- Cons:
  - May result in rework in later phases when previous phases get complete

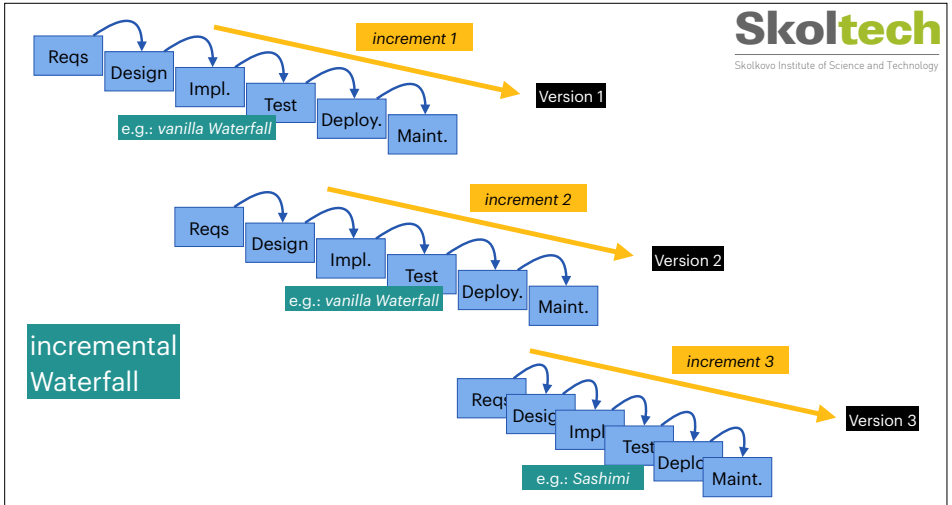


§2. Waterfall models

2.1. V-model

- **V-model** [works reasonably well when]:
  - Address shortage of time for testing by cascading the validation of software on multiple levels
  - Emphasize validation earlier in the process
- Pros:
  - Mostly **predictive** model
  - Early detection on potential problems
- Cons:
  - May result in rework in later phases when previous phases get complete

Incremental Waterfall



§2. Waterfall models

2.2. Incremental Waterfall models

- **Incremental Waterfall:**
  - Build an **increment** (all waterfall steps apply), then switch to another one
  - Increments may overlap
  - Can use different models during each increment
- Pros:
  - Deliver value early
  - Get feedback early
- Cons
  - Don't know about some of the requirements early in the phase
- If your organisation may benefit from getting partial work early, use it

§3. Iterative models

Skoltech
Skolkovo Institute of Science and Technology

Why iterative software development?

good predictive design

bad predictive design

Skoltech
Skolkovo Institute of Science and Technology

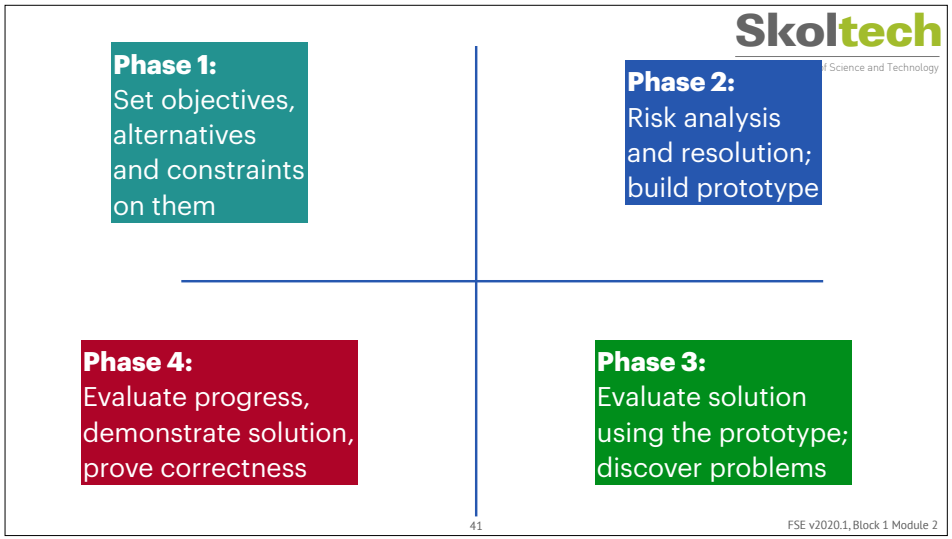
§3. Iterative models

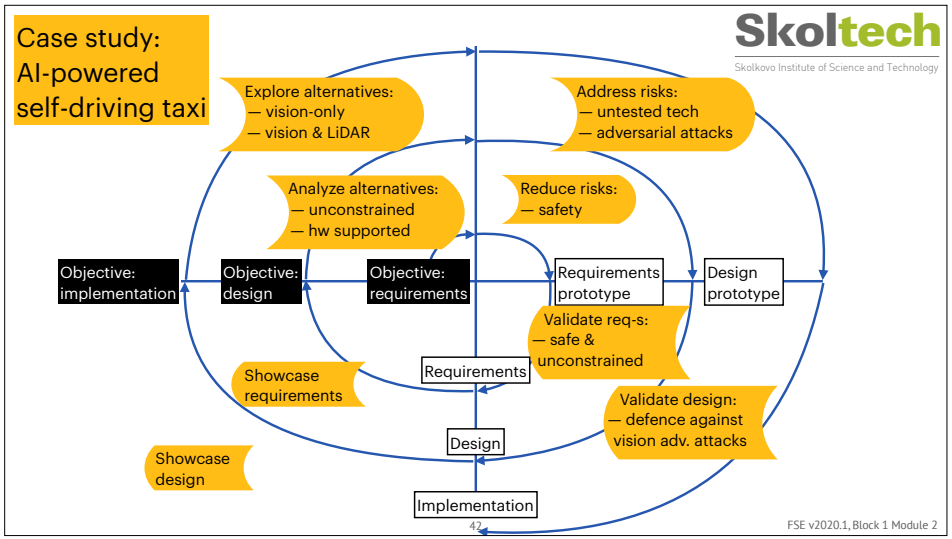
Why iterative software development?

- **Predictive models:**
  - Ill-suited to handle unexpected change
  - Don't handle fuzzy requirements well
  - Spend a lot of effort at the beginning figuring out exactly what they will do
- **Iterative models:**
  - Build the application incrementally, **start with the smallest reasonably useful program**
  - Add more features in a series of increments
  - Increments cheaper (smaller amount of work), including cheaper to cancel
  - Handle fuzzy requirements by building the known parts now, figuring the rest later

Skoltech
Skolkovo Institute of Science and Technology

The spiral model





§3. Iterative models

3.1. The spiral model

- **Risk-driven** cyclic approach: do only as much as needed to lower the risk at a particular stage
- **Process generator:** use other models within
- Determine objectives ⟶ identify and resolve risks ⟶ develop and test ⟶ planning for next iteration
- Milestones:
  - Life cycle objectives (LCO): consensus on technical and management approach
  - Life cycle architecture (LCA): project's approach can satisfy goals + eliminated risks
  - Initial operational capability (IOC): sufficient preparation of the software and all participants for release

§3. Iterative models

3.1. The spiral model

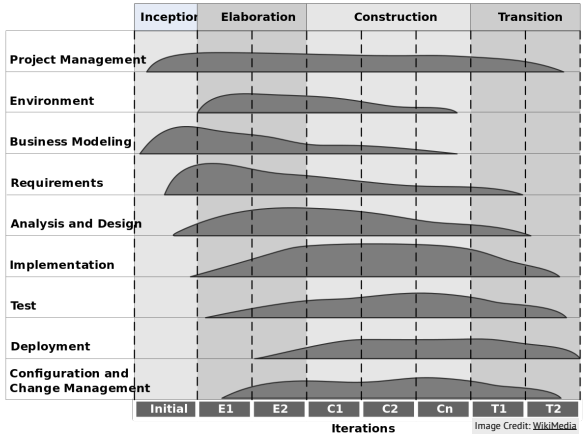
+ Supports go/no-go decisions
+ Risk focused
+ Easy to accommodate change
+ Iterative: cost and effort estimates become more accurate over time
− Complicated, not always worth the effort
− Costs more
− Need stakeholder engagement and skills for project review
− Risk analysis costs much time: costly for small projects
- **Use:** very large high risk projects

The Unified Process

§3. Iterative models

3.2. Unified process

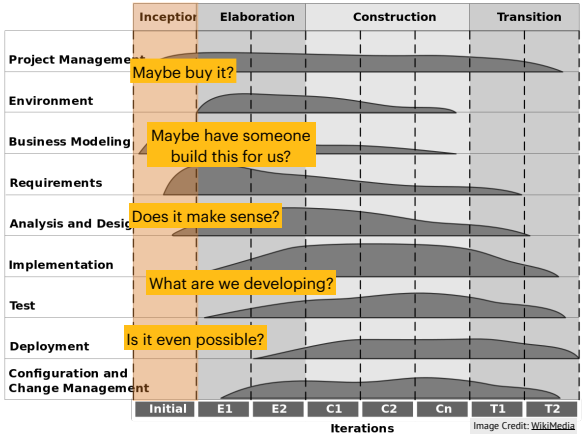
- Iterative and incremental
- Inception ⟶ Elaboration ⟶ Construction ⟶ Transition
- Gray area: share of effort into process step at each phase



§3. Iterative models

3.2. Unified process

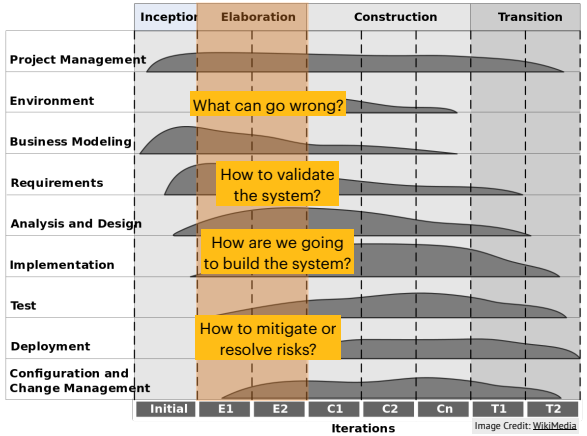
- **Inception:**
  - The shortest phase
  - Focus on business cases, scope and requirements
  - Feasibility study
  - Build vs. buy
  - Preliminary schedule and cost
  - Milestone: lifecycle objective (LCO)



§3. Iterative models

3.2. Unified process

- **Elaboration:**
  - Capture requirements
  - Address known risks
  - Validate architecture
  - Build core components of the system to demonstrate that it will work
  - Milestone: lifecycle architecture (LCA)



§3. Iterative models

Skoltech
Skolkovo Institute of Science and Technology

3.2. Unified process

- **Construction:**
  - The largest phase
  - Build software
  - Many iterations ⟶ each time release the software
  - Iterative and incremental
  - Milestone: initial operation capability (IOC)

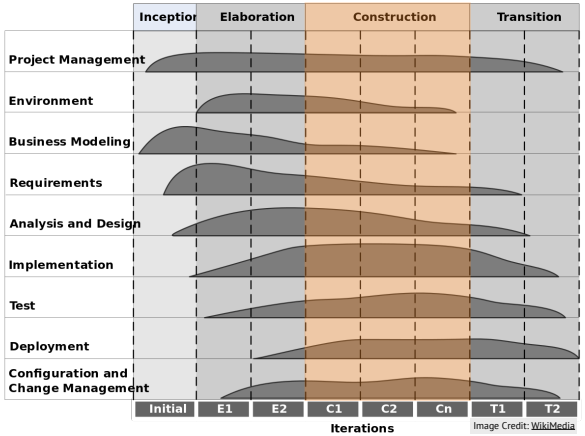


Image Credit: WikiMedia

§3. Iterative models

Skoltech
Skolkovo Institute of Science and Technology

3.2. Unified process

- **Transition:**
  - Deployment
  - Get feedback from users
  - Refine software based on feedback

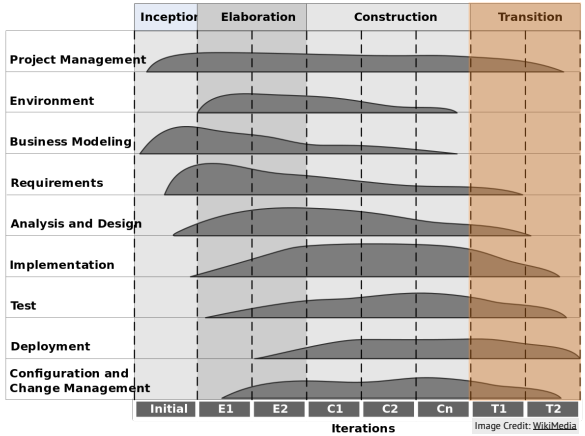


Image Credit: WikiMedia

§3. Iterative models

Skoltech
Skolkovo Institute of Science and Technology

3.2. Unified process

- **A framework, not a process:** can use any model during any phase
- Any step requires **all phases** in process, but emphasis may be on different steps
- Architecture-centric, use-case-centric
- Focus on risk mitigation
- **Pros:**
  - Quality and reuse by thinking about architecture ahead of time
  - Risk mitigation: more chances for success
  - Flexibly incorporates other s/w development models
- **Cons:** complicated, needs more resources, too heavy for small projects
- **Use:** bigger and riskier projects, not all requirements known early, desire to deliver value earlier

§4. Agile models

Skoltech
Skolkovo Institute of Science and Technology

Skoltech
Skolkovo Institute of Science and Technology

Why Agile?

Skoltech
Skolkovo Institute of Science and Technology

Image Credit: kickvick.com

Skoltech
wo Institute of Science and Technology

Image Credit: kickvick.com

Skoltech
Skolkovo Institute of Science and Technology

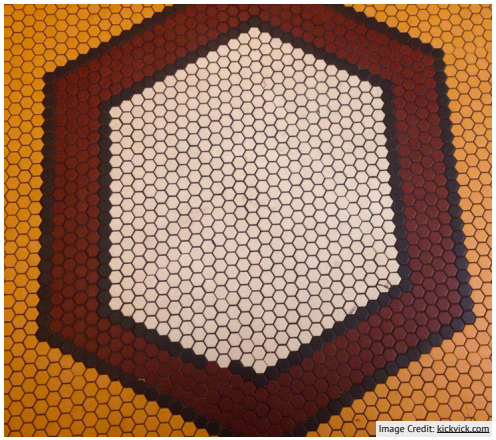


Image Credit: kickvick.com

Image Credit: The Fortune Teller (1617), WikiMedia

Image Credit: "Blind monks examining an elephant" by Itcho Hanabusa (1888), WikiMedia

Image Credit: pixabay.com

Image Credit: wallpaperflare.com

Image Credit: Billy Brown

The Agile manifesto

§4. Agile models

4.2. Agile manifesto: values and principles

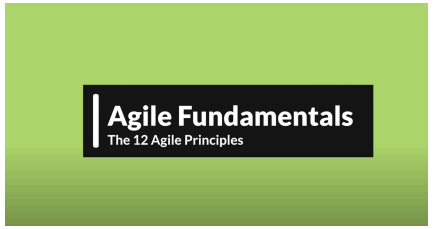
- Individuals and interactions over processes and tools ⟶ informal chats are better
- Working software over comprehensive docs
- Customer collaboration over contract negotiation ⟶ understand user needs
- Responding to change over following a plan ⟶ make adjustments as situation changes

YouTube video:
The Agile Manifesto - 4 Agile Values Explained

§4. Agile models

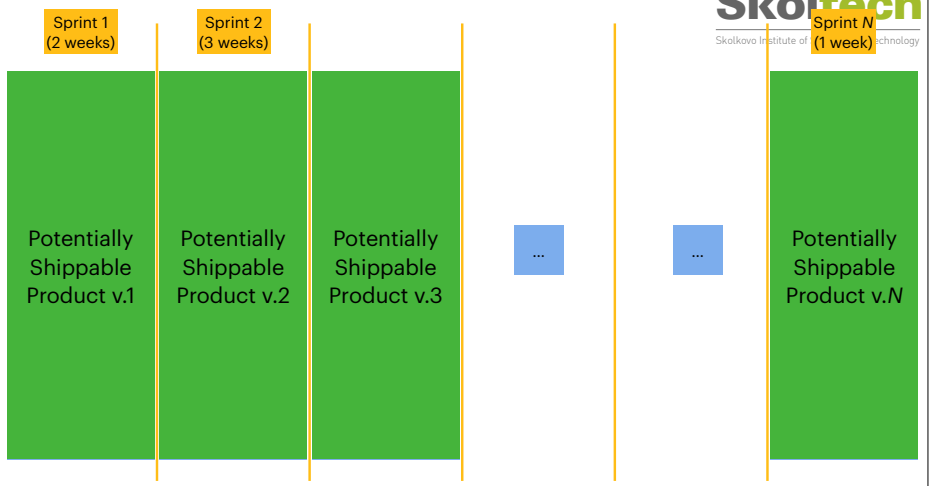
4.2. Agile manifesto: values and principles

- Focus in customer values
- Embrace change
- Preference towards a shorter time scale
- People with different skills work together
- Motivate individuals
- Face-to-face communication
- Working software is the first priority
- Maintain a constant pace indefinitely
- Pay attention to your design ⟶ make changes easily
- Simplicity ⟶ maximize amount of work not done
- Self-organisation in teams
- Reflect on how to become more effective



YouTube video:
Agile Fundamentals: The 12 Agile Principles

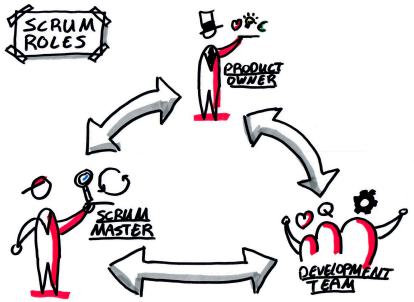
Agile frameworks: Scrum

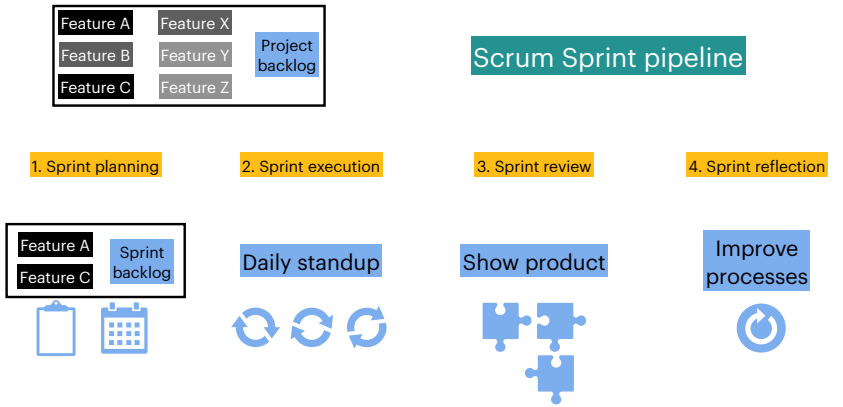


§4. Agile models

4.3. Scrum Agile framework

- Roles:
  - Product Owner: defines what needs to be done
  - Scrum Master: facilitates meetings
  - Team: builds software





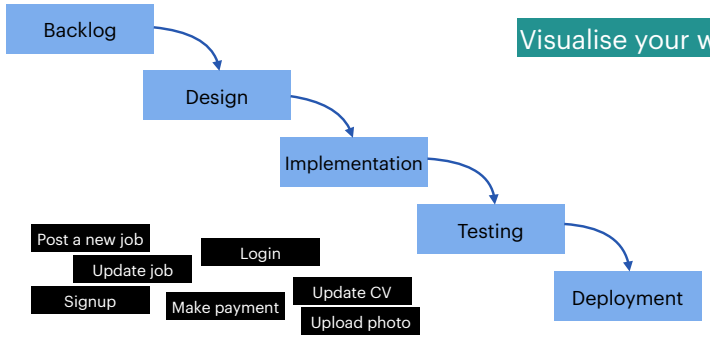
§4. Agile models

4.3. Scrum Agile framework

- Agile principles:
  - Build iteratively ⟶ address changes
  - Lots of meetings ⟶ collaboration
  - Continuous improvement with retrospectives
  - ...

Agile frameworks: Kanban

Skoltech
Skolkovo Institute of Science and Technology



Skoltech
Skolkovo Institute of Science and Technology

Visualise your workflow

Backlog	Design	Implementation	Testing	Deployment
Post a new job				
Update job				
Login				
Signup				
Make payment				
Update CV				
Upload photo				

Skoltech
Skolkovo Institute of Science and Technology

Visualise your workflow

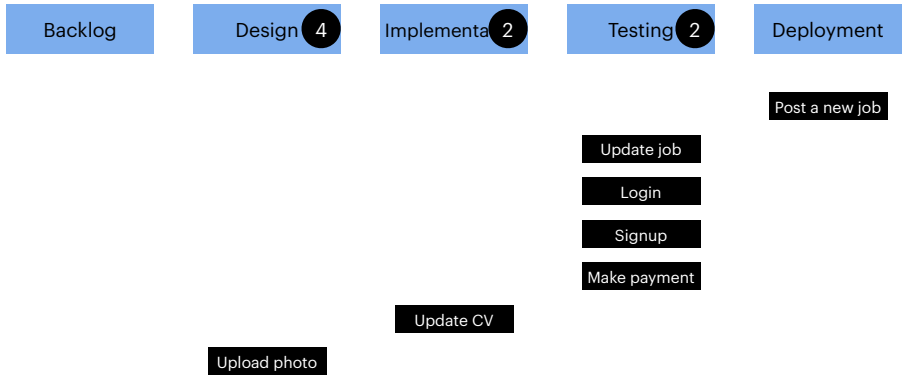
Backlog	Design	Implementation	Testing	Deployment
	Post a new job			
	Update job			
Login				
Signup				
Make payment				
Update CV				
Upload photo				

Skoltech
Skolkovo Institute of Science and Technology

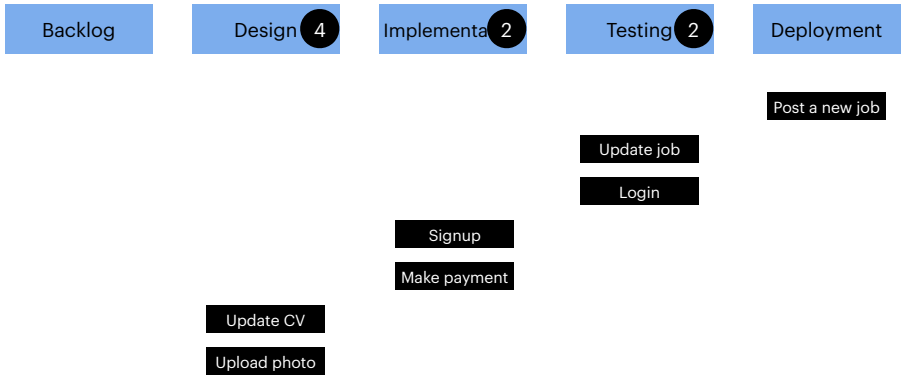
Visualise your workflow

Backlog	Design	Implementation	Testing	Deployment
				Post a new job
			Update job	
			Login	
			Signup	
			Make payment	
		Update CV		
	Upload photo			

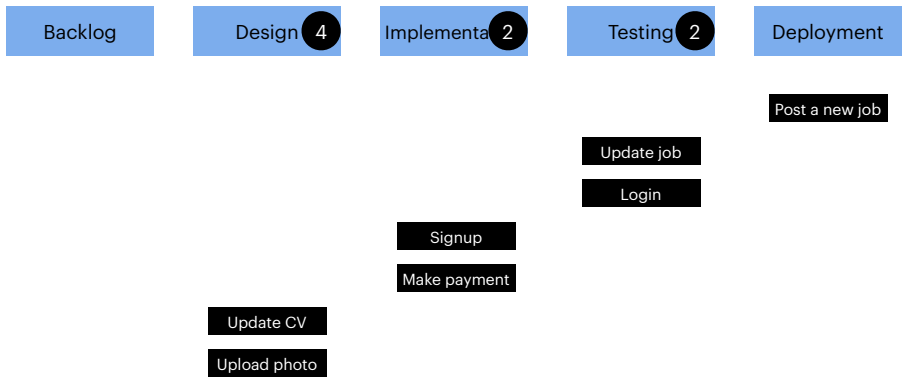
Limit work in progress



Manage the flow



Make process policies explicit



§4. Agile models

4.4. Kanban Agile framework

- Agile principles:
  - Visualise your workflow ⟶ Simplicity
  - Limit work in progress ⟶ Maintain a constant pace indefinitely
  - Manage the flow ⟶ Maintain a constant pace indefinitely
  - Make process policies explicit ⟶ Self-organisation in teams / Motivate individuals / Face-to-face communication

§4. Agile models

Skoltech
Skolkovo Institute of Science and Technology

- **Pros** from adaptive nature of software development
  - Detect process problems early
  - Validate user needs early
  - Detect integration issues early
  - People and interaction ⟶ detect translation issues quickly
- **Cons** from iterative development
  - Constant changes ⟶ lack of control and unpredictability
  - People need to be more involved ⟶ spent more time on the system
- **Use:** when the requirements may change or the technology is unknown

References

Skoltech
Skolkovo Institute of Science and Technology

1. Stephens, R. (2015). Beginning software engineering. John Wiley & Sons.
2. Cockburn, A. (2006). Agile software development: the cooperative game. Pearson Education.
3. *Coursera course:* Software Development Lifecycle. University of Minnesota