Software Engineering

Lydie du Bousquet

Lydie.du-bousquet@imag.fr

In collaboration with J.-M. Favre, I. Parissis, Ph. Lalanda, Y. Ledru

Short introduction

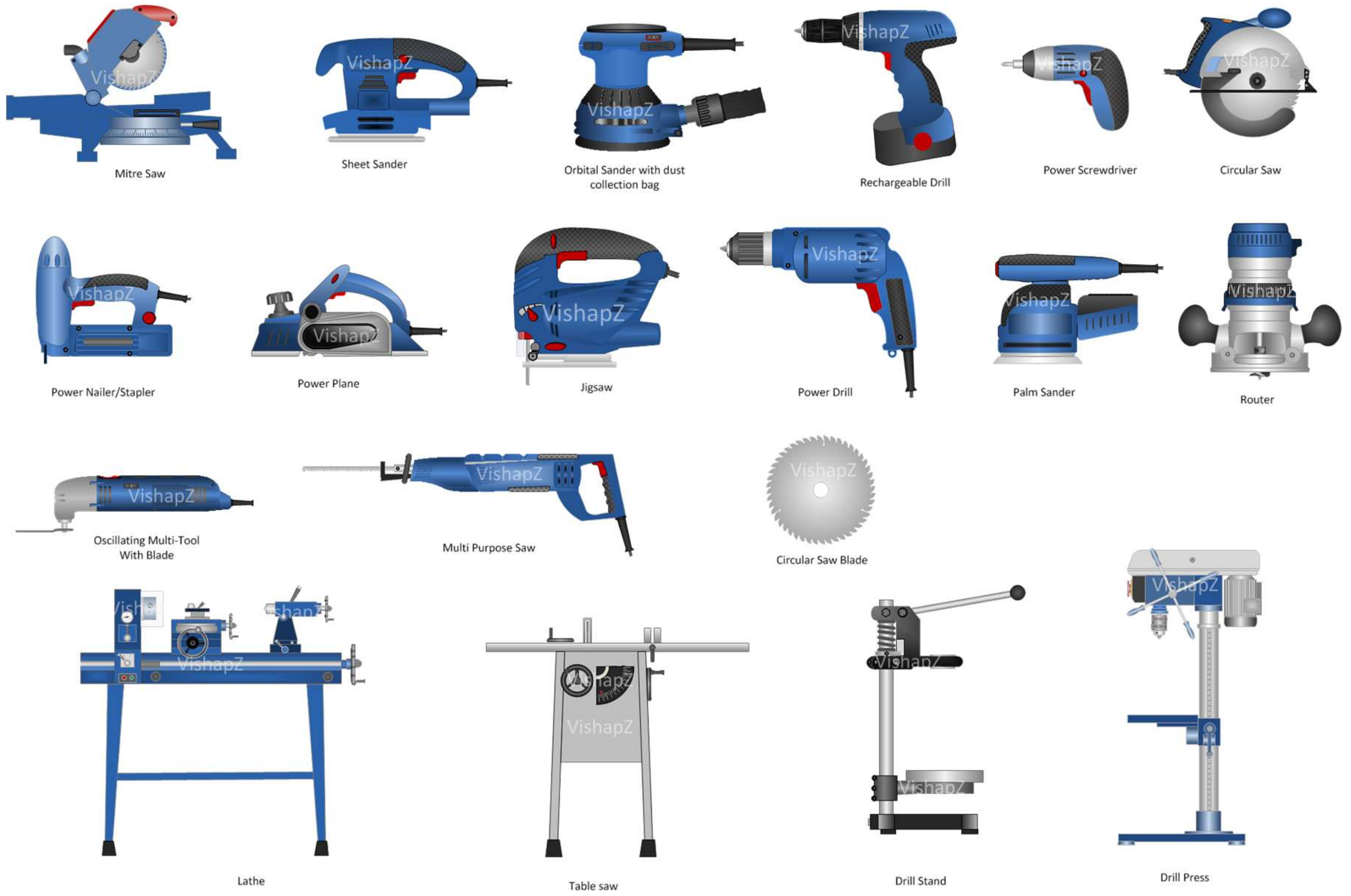
- Lydie du Bousquet
  - Professor at UGA
  - *Software engineering, validation, test*

- Frédéric Lang
  - Researcher at INRIA
  - *Compositional Verification, New Generation Formal Description Techniques*

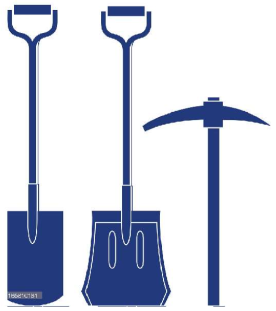
This class

- Introduction to software engineering as a tool box

Choose a tool to drill something



Wait! What should I drill? What size?



Before using a tool, you should :

- Know the **characteristics of the tools**
- Learn to **use** them
- Be able to **choose** the appropriate one(s)

This is the objective of the class!
(for software engineering tools)

Class program during the semestrer

- Software engineering, development process
- Requirement engineering
- Model with UML
- Model with formal methods
- Validation by test

Evaluation

- One **mid-term exam** **(0.3)**
  - Exercises to check that the notions are mastered
  - Basic exercices but not so simple
  - Related to dev. process, req. eng. and UML
- One **final exam** **(0.7)**
  - More complex problems
  - Related to all the chapters

Class program this week

1(a) Software engineering and process

1(b) Requirement engineering

Schedule

- What is Software Engineering?
- What are the activities during the development?
- How are they organized?
- Which process should you choose?

What is software Engineering?

Exercise 0, Q1

What is software Engineering?

- Engineering?
  - derived from the Latin
  - ***ingenium***, meaning "cleverness"
  - ***ingeniare***, meaning "to contrive, devise" (find a solution, build)
- Software Engineering
  - discipline that is concerned with all aspects of software production

Why Software Engineering?

- Provide systematic methods, tools
- To achieve
  - predictability
  - precision
  - mitigated risk
  - professionalism

Schedule

- What is Software Engineering?
- What are the activities during the development?
- How are they organized?
- Which process should you choose?

What are the activities during a software project?

Exercise 0, Q2

Classical activities

- Requirements Analysis
- Specification
- Software architecture
- Design
- Implementation
- Testing
- Documentation
- Installation, deployment
- Training and Support
- Maintenance

Development process

- Also known as
  - development methodology
  - software development life cycle,
  - software process
- Process followed during the development of a software product
  - organization of the tasks or activities that take place during the development
  - several models of development process

What development process do you know ?

How the previous activities are organized?

Classical development processes

- Code and fix
- Waterfall development
- V-shaped Model
- Prototyping
- Incremental/iterative development
- Agile development
- SCRUM, KANBAN, …
- Spiral

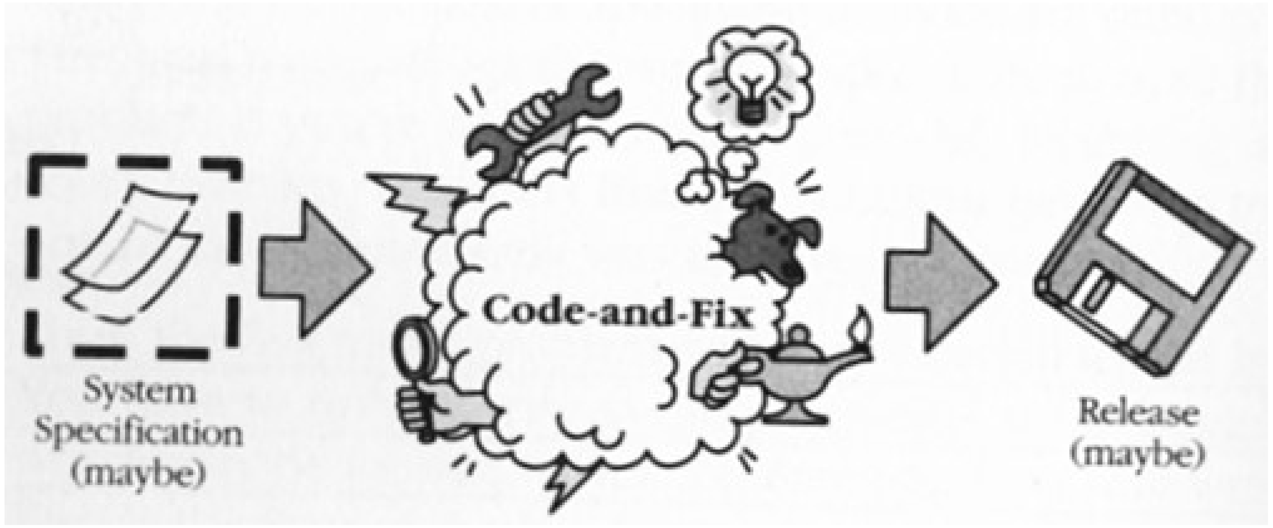
Classical development processes

A software engineer should

- Be able to recognize a process
- Know the step order
- Know advantages and drawbacks
- Choose an appropriate process w.r.t. the situation

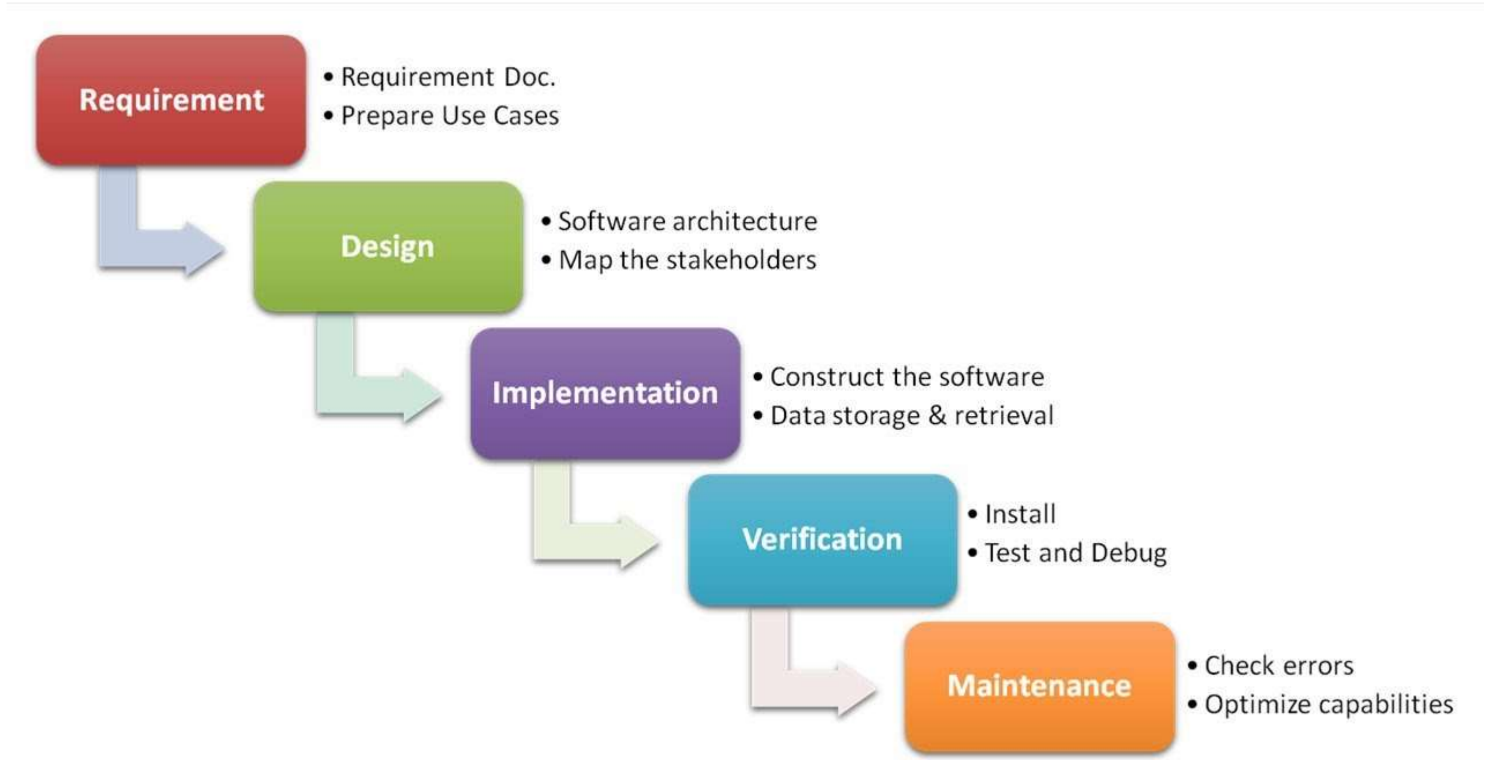
Code and fix

- Without much of a design in the way, programmers immediately begin producing code.
- At some point, testing begins (often late), unavoidable bugs must then be fixed before the product can be shipped.



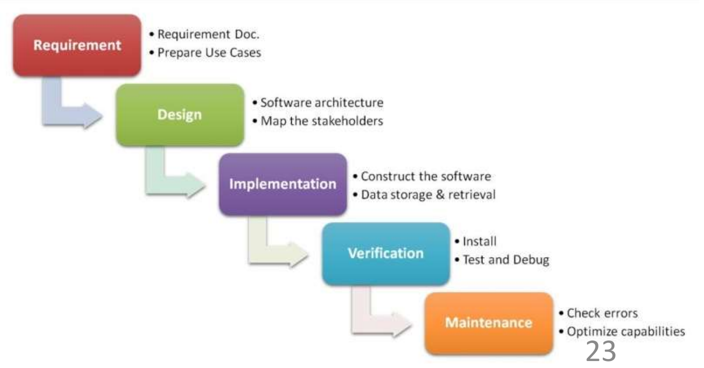
Waterfall development

- sequential development approach,
  in which development is seen as flowing steadily downwards through several phases

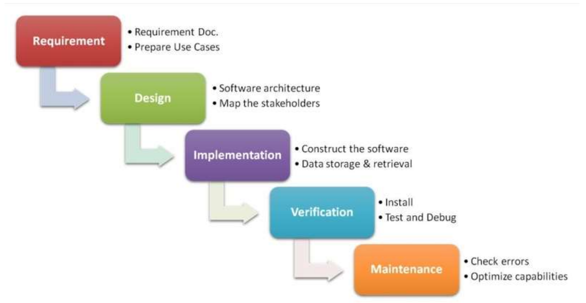


Waterfall development

- The idea behind:
  - **structured approach**:
    the current phase should be finished **before** starting a new one
  - identifiable **milestones**:
    each phase is documented and validated
- Criticisms
  - Many documents
  - Clients may not know exactly what their requirements
  - Requirements may change
- Can be used
  - Projects with
    well-known requirements

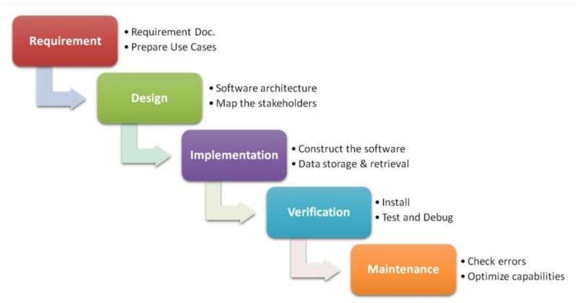


Waterfall development



- The idea behind:
  - **structured approach**:
    the current phase should be finished **before** starting a new one
  - identifiable **milestones**:
    each phase is documented and validated
- **Advantages**
  - Each phase has specific **deliverables**
  - **Verification** at each stage for early detection of errors / misunderstanding
  - Simple
- **Disadvantages**
  - Assumes that the requirements are **frozen**
  - Very **difficult to go back** to any stage after it finished
  - Little flexibility and adjusting scope
  - Executable is only available at the end

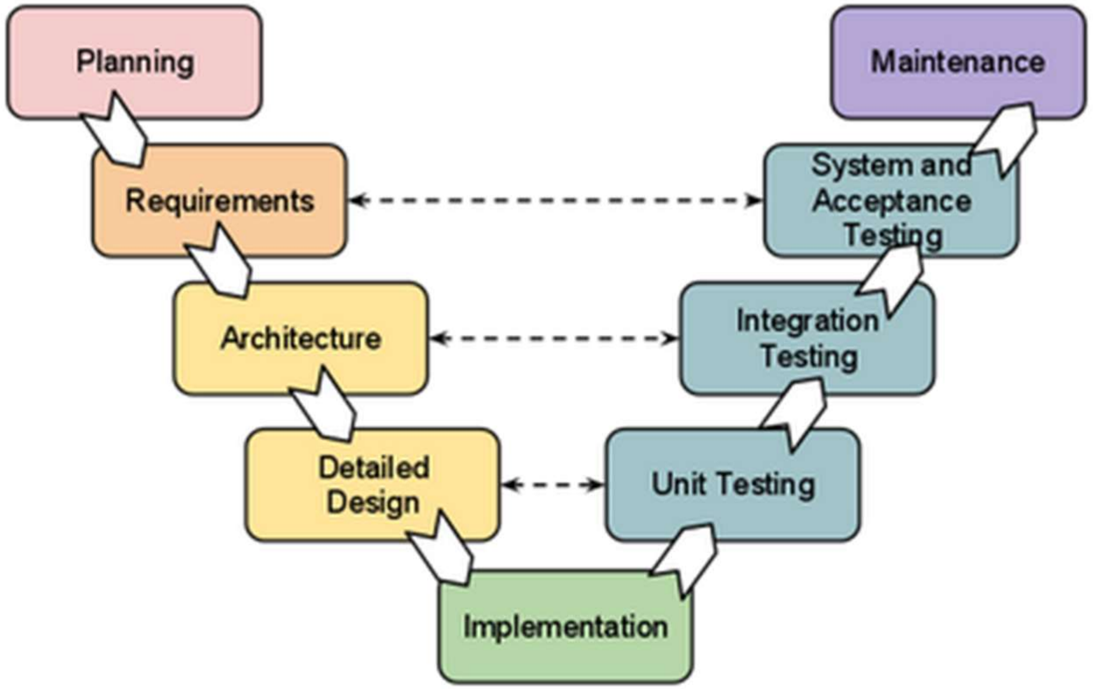
Waterfall development



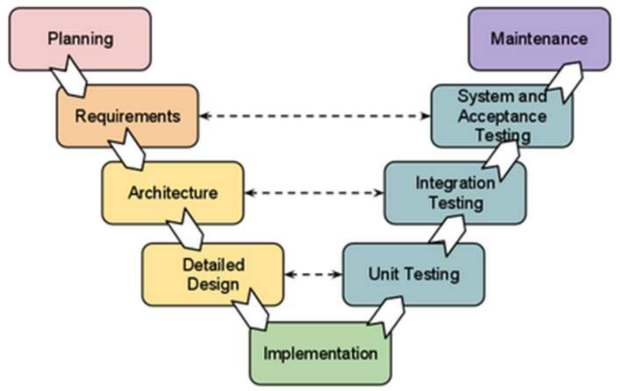
- **Criticisms**
  - (Too) many documents
  - Clients may not know exactly what their requirements
  - Requirements may change
- **Can be used**
  - Projects with well-known requirements
  - Middle-size

V-shaped Model

- Like the waterfall model, it is a **sequential** path of execution of processes
- **Testing** of the product is planned in parallel with a corresponding phase of development



V-shaped process



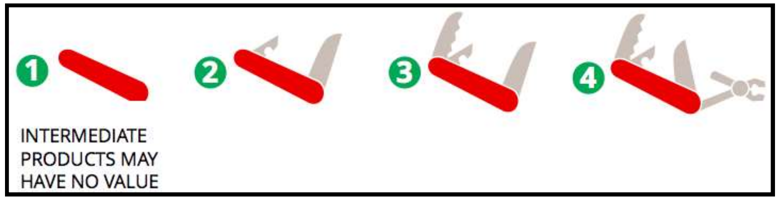
- The **idea** behind
  - Focusing on validation (testing, most of the time)
- **Advantages**
  - Simple and easy to use.
  - Testing happens well before/in parallel of coding.
- **Disadvantages** = same as water-fall process
  - If any changes happen in midway, then the test documents along with requirement documents have to be updated.
- When to use the V-model
  - For projects where requirements are easily understood
  - When validation is a key point,
    when you have an independant validation team

Incremental/iterative development

- Objective:
  - Being able provide an adequate final product
  - reduce risks by breaking a project into smaller parts and providing more ease-of-change during the development
- Different possibilities:
  - A series of mini-Waterfalls are performed.
    All phases of the Waterfall are completed for a small part,
    before proceeding to the next increment,
  - **Overall requirements** are defined **before** proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or
  - Requirements analysis, and design of architecture and system core are defined, followed by developing successively the different parts (= increments).

Incremental development

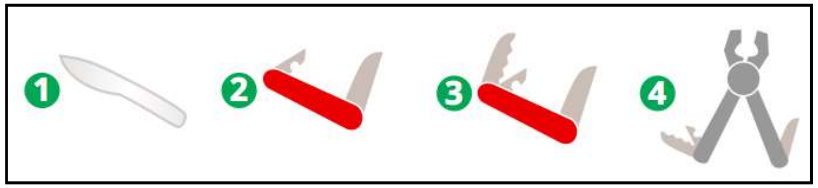
- **Principles**:
  - Requirements are globally collected,
  - Development is sliced into parts
  - Parts are developed at various times and integrated to obtain the releases
  - Each increment adds more software value – e.g. adding package



- **Advantages**
  - First release can be delivered sooner than in a waterfall process
  - Major parts can be developed first, the order of others can be modify
  - Users can provide feedbacks to adjust requirements for each the release
  - Easier to test and easier to manage risks than waterfall
- **Disadvantages**
  - Architecture generally chosen at the beginning => needs a complete definition of the whole system before it can be broken down into parts
  - Total cost is higher than waterfall

Illustration from https://productcoalition.com/product-development-using-agile-methodology-446c01ecd510

Iterative development



- **Principles**
  - Build a very first version,
  - Get some feedback and refine it to make better,
  - Keep doing that until the product is good
- **Advantages**
  - Reduce the rarely used features, maximize the frequently used features
  - Usable product at any time
  - Suited for Agile Organizations
- **Disadvantages**
  - Increased Pressure on User Engagement
  - Each phase of an iteration is rigid with no overlaps
  - Requires higher level of technical excellence (than the other processes)
  - Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

Illustration from https://productcoalition.com/product-development-using-agile-methodology-446c01ecd510

Incremental vs iterative development

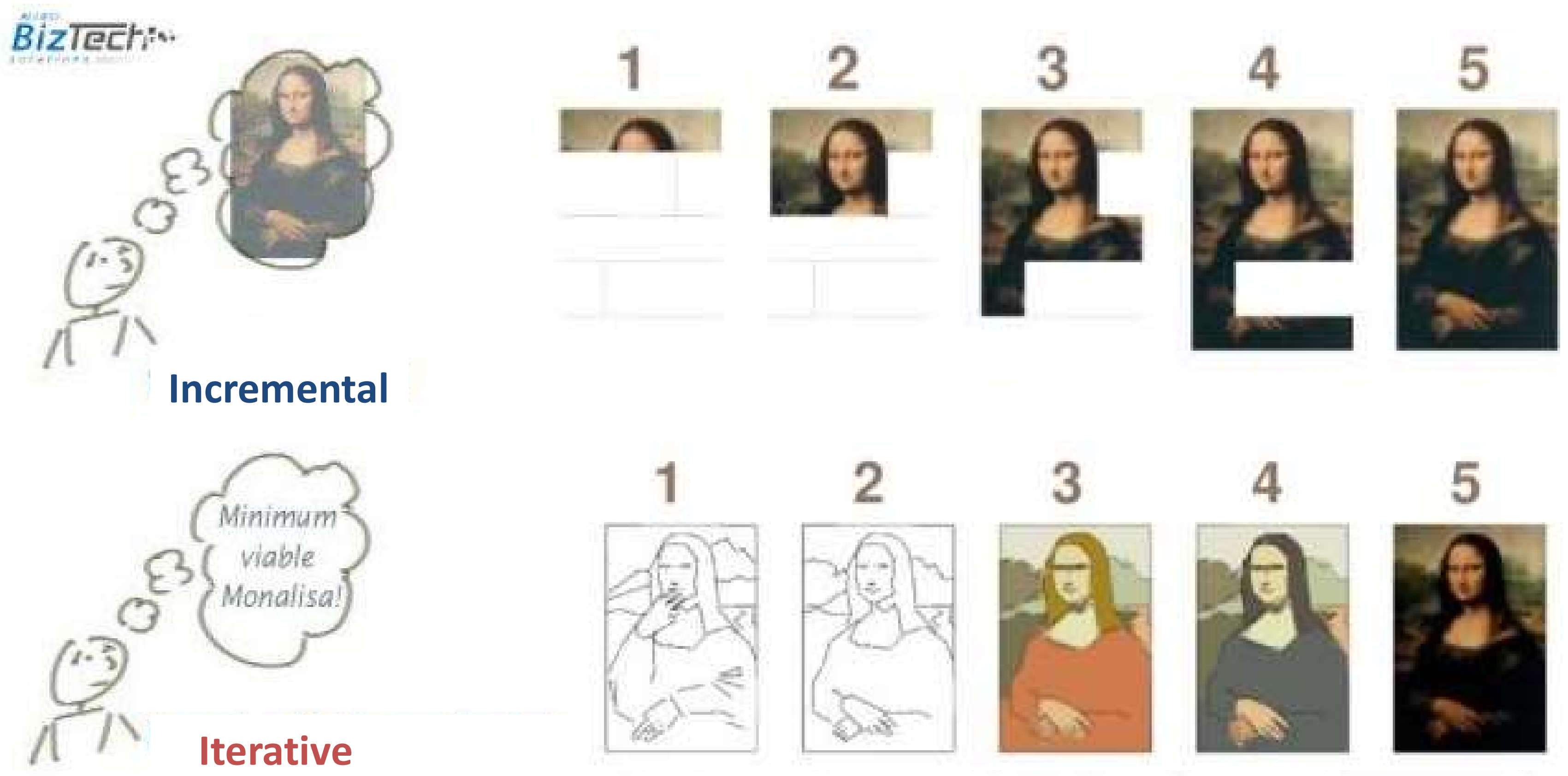
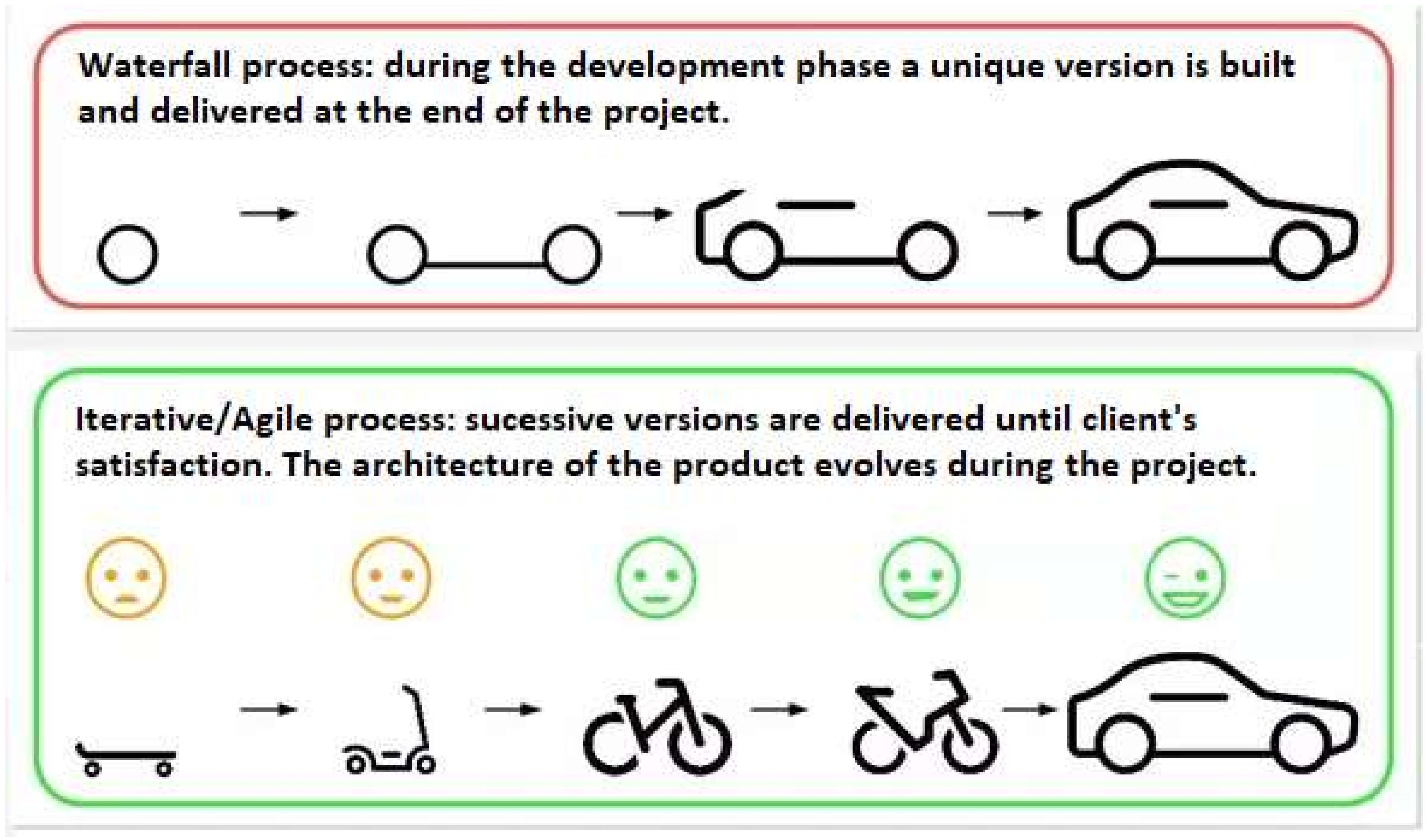


Illustration from http://www.golali.co/waterfall-vs-iterative-model/ and from https://agilenotion.com/agile-categoriesiterative-incremental-evolutionary/

Waterfall vs Iterative



Waterfall process: during the development phase a unique version is built and delivered at the end of the project.

Iterative/Agile process: sucessive versions are delivered until client's satisfaction. The architecture of the product evolves during the project.

Adapted from an image of http://www.golali.co/waterfall-vs-iterative-model/

Incremental vs **iterative** development

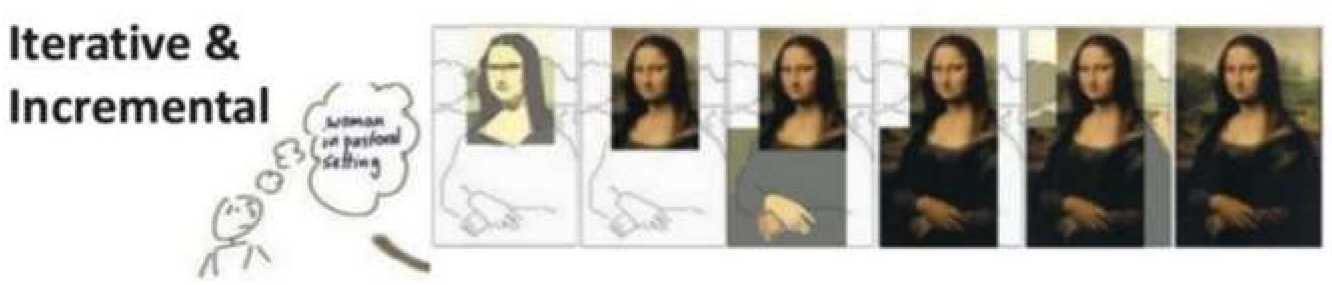
- The word **increment** fundamentally means **add onto**
- The word **iterate** fundamentally means **re-do**

- For **incremental**,
  - you need to have well defined requirements
  - Between each increment, you can adjust part of the requirements
  - Several increments can be produced in parallel

- For **iterative**,
  - You start with a set of requirements in order to produce a first delivery
  - At each iteration, you collect new requirements and you make a new version

Agile development processes

- Mainly an iterative development processes
  - Rapid cycles (1 to 3 weeks)
  - Small release
- Examples of agile processes
  - XP (Extreme Programming)
  - Scrum
  - …



Agile processes: 12 principles

- **Customer satisfaction** by early and continuous delivery of useful software
- Welcome **changing requirements**, even in late development
- Working software is delivered frequently (weeks rather than months)
- Close, **daily cooperation** between business people and developers
- Projects are built around motivated individuals, who should be trusted
- **Face-to-face conversation** is the best form of communication (co-location)
- **Working software** is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to **technical excellence** and good design
- **Simplicity** (maximizing the amount of work not done) is essential
- Self-organizing teams
- Regular **adaptation** to changing circumstance

Agile software development practices

- Backlogs (Product and Sprint)
- Behavior-driven development (BDD)
- Continuous integration (CI)
- Daily stand-up / Daily Scrum
- Iterative and incremental development (IID)
- Pair programming
- Planning poker
- Refactoring
- Retrospective
- Story-driven modeling
- Test-driven development (TDD)

Agile processes:

- **Advantages**
  - **Customer satisfaction** by rapid, continuous delivery of useful software
  - **People and interactions** are emphasized rather than process and tools
  - Customers, developers and testers **constantly interact** with each other
  - Working software is **delivered frequently** (weeks rather than months)
  - Face-to-face conversation is the best form of communication
  - Close, **daily cooperation** between business people and developers
  - Continuous attention to **technical excellence** and good design
  - Regular adaptation to changing circumstances
  - Even late changes in requirements are welcomed

Agile processes

- **Disadvantages**
  - Difficult to **assess the effort** required at the beginning of the dev. life cycle.
  - There is **lack of emphasis** on necessary **designing** and **documentation**.
  - The project can easily get taken off track
- When to use Agile processes:
  - When new changes are needed to be implemented
  - End users' needs are ever changing in a dynamic business and IT world

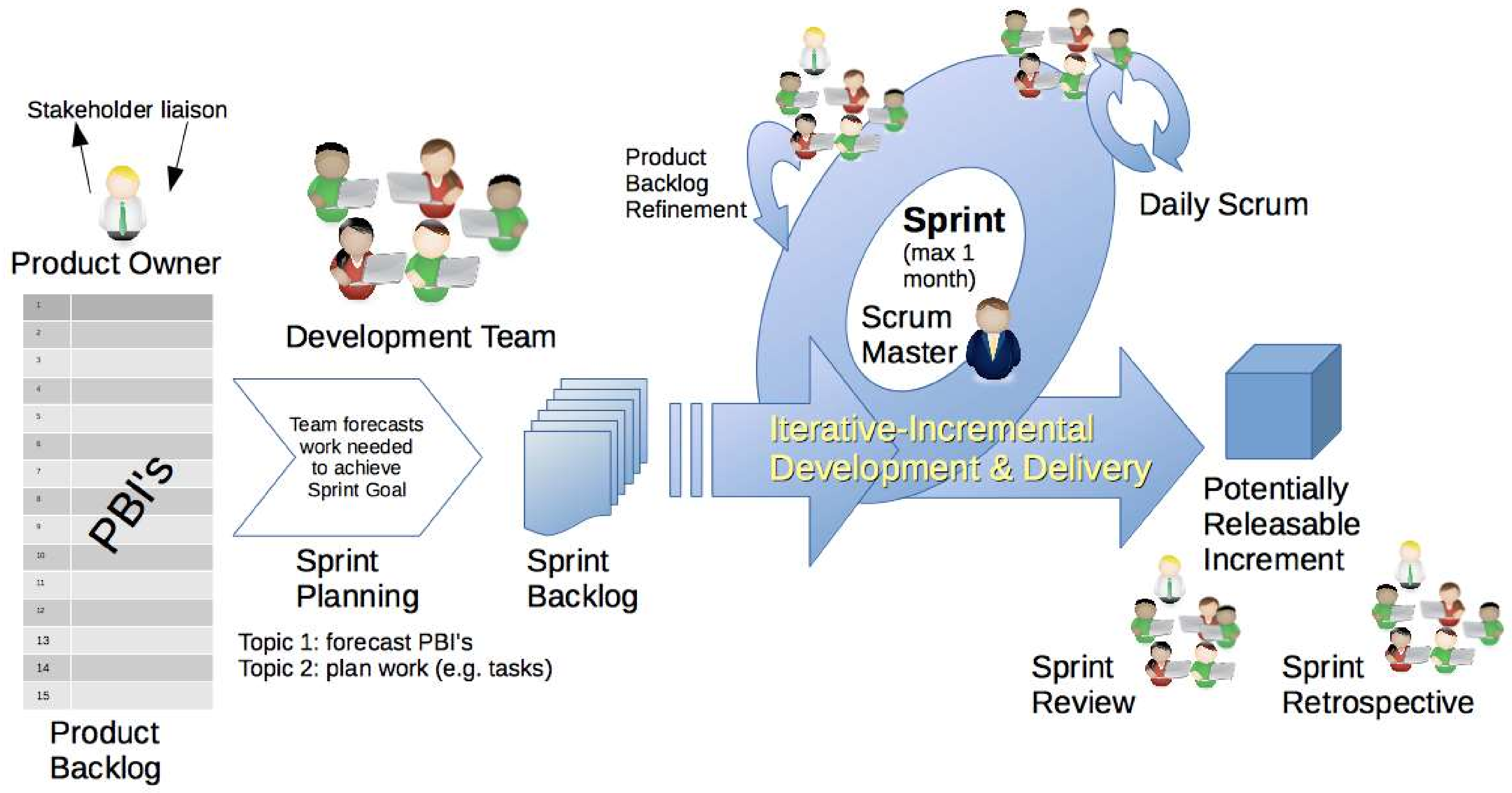
XP

- Agile project management
- Intended to improve
  - software quality and
  - responsiveness to changing requirements
- Based on multiple short development cycles
  to reduce the cost of changes in requirements
- Some key elements
  - programming in pairs or doing extensive code review,
  - unit testing of all code,
  - not programming features until they are really needed,
  - code simplicity and clarity
  - a flat management structure

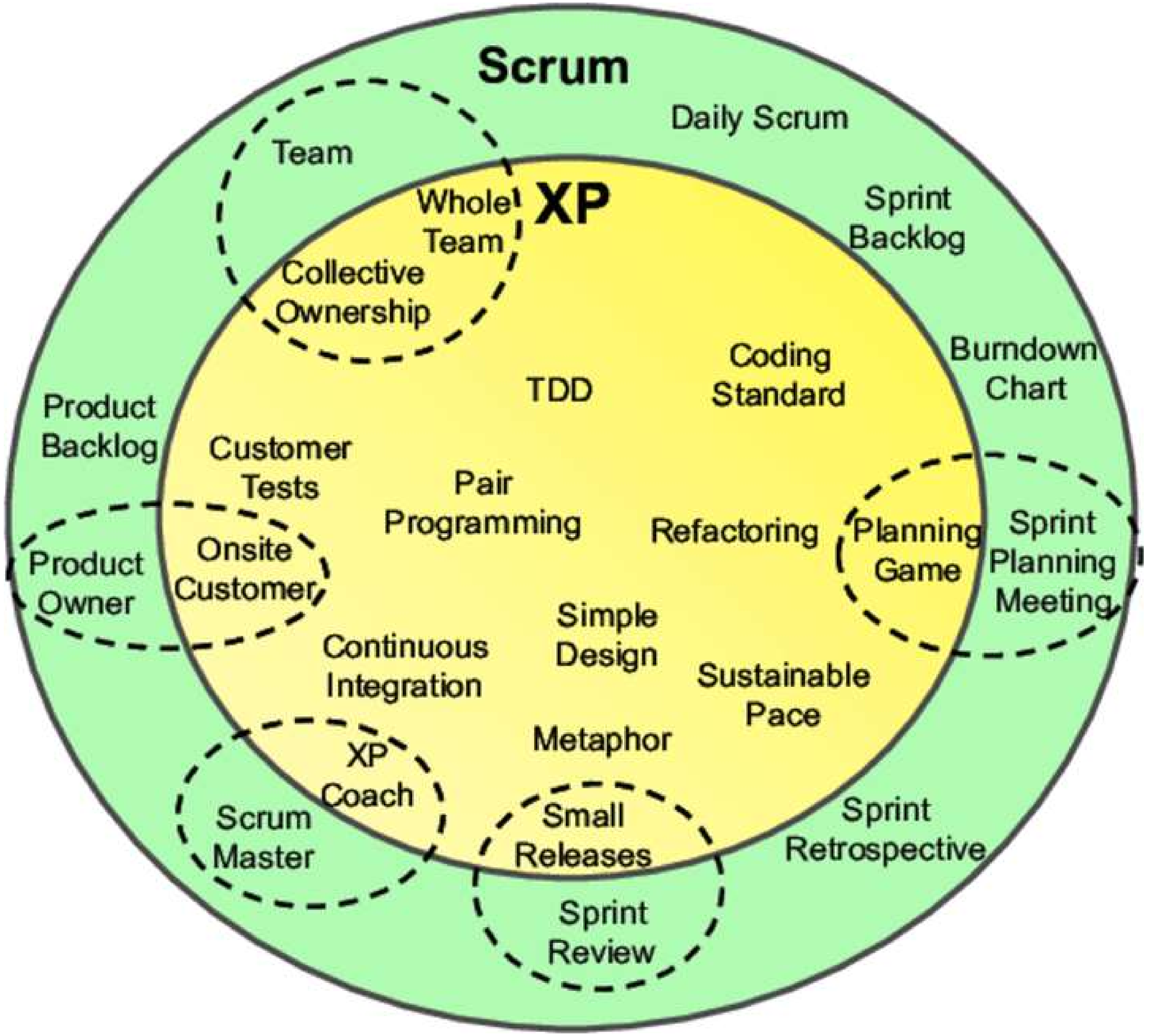
SCRUM

- Agile project management
- Scrum team: product owner, developers, scrum master
- Sprint
  - fixed period, between one week and one month
  - starts with a sprint planning to define sprint goal
  - ends with
    - A *sprint review* = to elicit stakeholders feedback
    - A *sprint retrospective* = to identify lessons and improvements for the next sprints
- Daily scrum meetings
  - are intended to be less than 15 minutes in length,
  - to announce progress and issues that may be hindering the goal

SCRUM

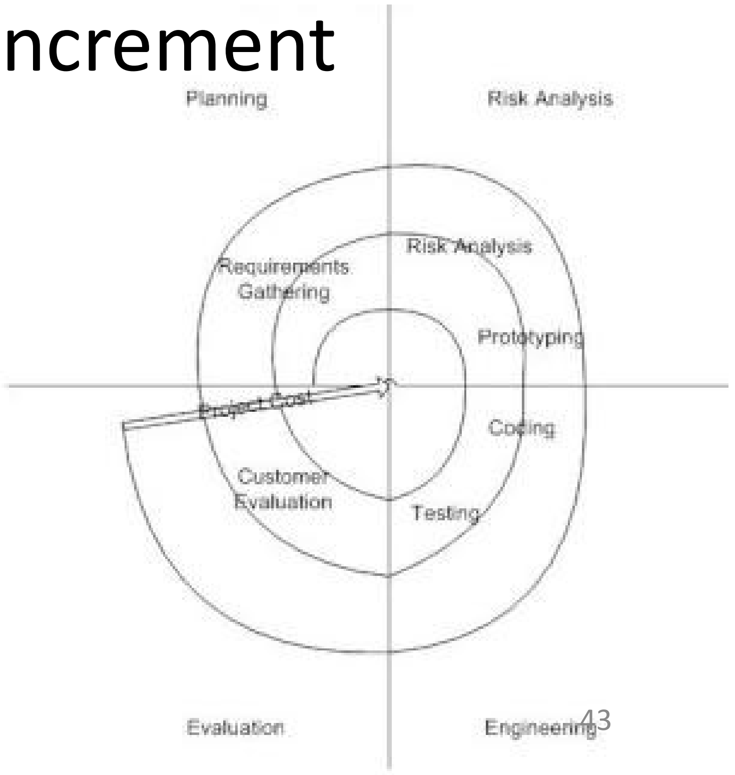


Stakeholder liaison
Product Owner
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
PBI's
Product Backlog
Development Team
Team forecasts work needed to achieve Sprint Goal
Sprint Planning
Topic 1: forecast PBI's
Topic 2: plan work (e.g. tasks)
Sprint Backlog
Product Backlog Refinement
Sprint (max 1 month)
Scrum Master
Daily Scrum
Iterative-Incremental Development & Delivery
Potentially Releasable Increment
Sprint Review
Sprint Retrospective


Scrum
Daily Scrum
Team
Whole Team
XP
Sprint Backlog
Collective Ownership
Burndown Chart
Coding Standard
TDD
Product Backlog
Customer Tests
Pair Programming
Refactoring
Planning Game
Sprint Planning Meeting
Product Owner
Onsite Customer
Simple Design
Continuous Integration
Sustainable Pace
Metaphor
XP Coach
Scrum Master
Small Releases
Sprint Retrospective
Sprint Review

Spiral development model

- Risk-driven process model
  - Similar to the incremental model,
  - with more emphasis placed on risk analysis.
- Each spiral corresponds to a increment
- Each spiral has four phases:
  - Planning,
  - Risk Analysis,
  - Engineering and
  - Evaluation



Spiral development model

- **Advantages** of Spiral model
  - High amount of risk analysis hence, avoidance of Risk is enhanced
  - Strong approval and documentation control
  - Additional Functionality can be added at a later date
- **Disadvantages** of Spiral model
  - Can be a costly model to use.
  - Risk analysis requires highly specific expertise.
  - Project’s success is highly dependent on the risk analysis
  - Doesn’t work well for smaller projects.

Spiral development model

- **When** to use Spiral model?
  - When costs and risk evaluation is important
  - For medium to high-risk projects
  - Users are unsure of their needs
  - Requirements are complex
  - Significant changes are expected (research and exploration)

Prototyping (activity)

- Activity of creating **prototypes** of the application, i.e., **incomplete** versions of the product
  - simulates only a few aspects of the final product
- **Outline**
  - Identify basic requirements
  - Develop an Initial Prototype
  - Review of the customers, including end-users, to examine the prototype and to provide feedback on additions or changes.
  - Revise and Enhance the specification / prototype using the feedback

Prototyping (activity)

- The **idea** behind
  - **Not** a standalone, complete development methodology
  - Attempts to reduce inherent project risk
  - User is involved throughout the development process
- Example of usages
  - Understanding of user requirements
  - Checking some technology

Prototyping (activity)

- **Advantages** of prototyping
  - Reduced time and costs
  - Improved and increased user involvement
- **Risks** using of prototyping
  - Insufficient analysis
  - Prototype vs finished system
  - Excessive development time of the prototype

Choice of a development process

- Learn the about the development processes
- Assess the needs of Stakeholders
- Use the criteria

Factors	Waterfall	V-shaped	Proto-type	Iter & increm	Agile
Unclear User Requirement	Poor	Poor	Good	Good	Excellent
Complex System	Good	Good	Excellent	Good	Poor
Reliable system	Good	Good	-	Good	Good
Documentations	Excellent	Excellent	-	Depends	Poor
Component reusability	Good	Good	Poor	Depends	Poor

Parameter	**Process Model→**	**Waterfall Model**	**Incremental Model**	**Prototype Model**	**Rad Model**	**Spiral Model**	**Agile Model**	**Xp programming**
Clear Requirement Specifications		Initial level	Initial level	At medium level	Initial level	Initial level	Change incrementally	Initial level
Feedback from user		No	No	Yes	No	No	No	Yes
Speed to change		Low	High	Medium	No	High	High	High
Predictability		Low	Low	High	Low	Medium	High	High
Risk identification		At initial level	No	No	No	Yes	Yes	Yes
Practically implementation		No	Low	Medium	No	Medium	High	High
Loom		Systematic sequence	Iterative sequence	Priority on customer feedback	Use readymade component	Identification of risk at each stage	Highly customer satisfaction and incremental development[09]	Customer satisfaction and incremental development
Any variation done		Yes-v model	No	No	No	Yes-win win spiral[6]	No	No
Understandability		Simple	Intermediate	Intermediate	Intermediate	Hard	Much complex	Intermediate
Precondition		Requirement clearly defined	Core product should clearly define	Clear idea of Quick Design	Clean idea of Reuse component	No	No	No
Usability		Basic	Medium	High	Medium	Medium	Most use now a days	medium
Customer priority		Nil	Nil	Intermediate	Nil	Intermediate	High	Intermediate
Industry approach		Basic	Basic	Medium	Medium	Medium	High	Medium
Cost		Low	Low	High	very high	Expensive	Much Expensive	High
Resource organization		Yes	Yes	Yes	Yes	No	No	Yes
Elasticity		No	No	Yes	Yes	No	Very high	Medium

Exercise 3 for
Which development process?

- System for student management in a university (this system replace an existing system without any functional evolutions)
- An new interactive system for travelers to have schedules on their smartphones
- A system to control subway without drivers
- A very large system for Air traffic management
- A very new 3D-system for software maintenance
- Infrastructure and services for city that wants to become a “smart-city”