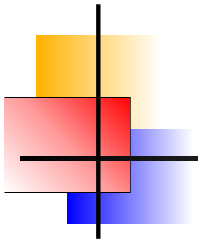


Programming Project: Agile Development and eXtreme Programming

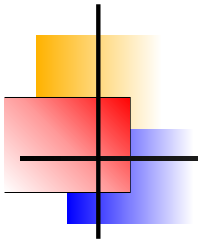
Guillermo Román Díez

groman@fi.upm.es

E.T.S. Ingenieros en Informática

Universidad Politécnica de Madrid

2019-2020

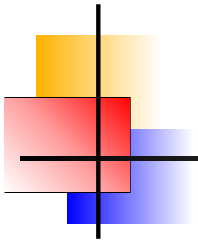


Introduction

- **Agile** development is gaining popularity in the recent years
  - It is not a specific recipe with concrete tasks that you only have to follow
  - It is more *a way of thinking* about software development
- The *Agile Manifesto* gives the basic ideas of the Agile philosophy

  `https://www.agilealliance.org/agile101/the-agile-manifesto/`

  - **Individuals and interactions** over processes and tools
  - **Working software** over comprehensive documentation
  - **Customer collaboration** over contract negotiation
  - **Responding to change** over following a plan

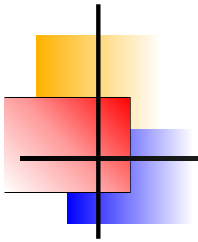


Introduction

Definition

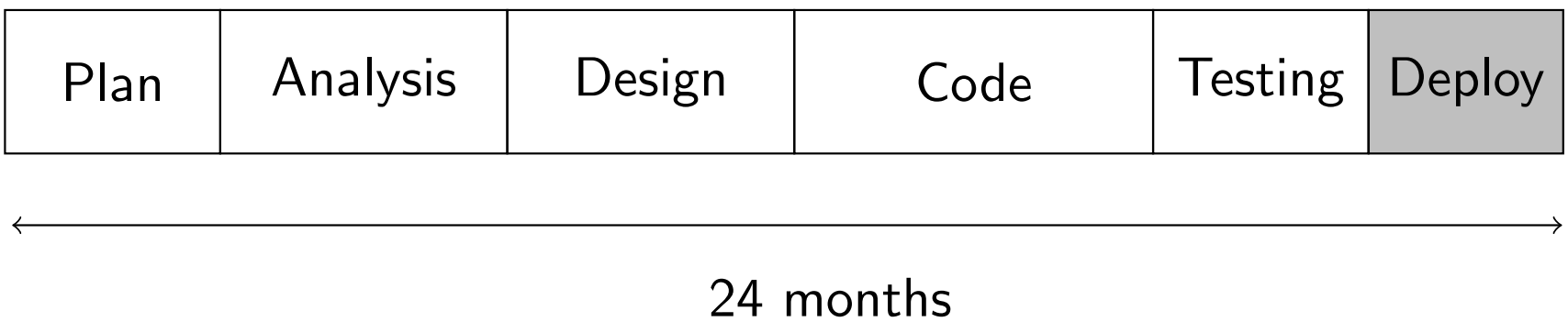
Agile project management is an iterative development methodology that values human communication and feedback, adapting to change, and producing working results

- Agile methods consist of a set of individual and team practices
- Involves the customer in the project as much as possible
- Some practices and tools are crucial to follow the Agile ideas:
  - Use version control systems
  - Use build tools
  - Follow coding standards
  - Refactoring
  - Continuous Integration
  - Test-driven-development
  - ...



Traditionally Lifecycles: Waterfall

- The classical **waterfall** model is divided in several phases performed sequentially
  - Plan, Analysis, Design, Code, Testing, Deploy
- Waterfall model presents some problems
  - Do not respond efficiently changes in the requirements
  - Phases has to be complete to pass to the next phase
  - Releases are not available until the end of the project

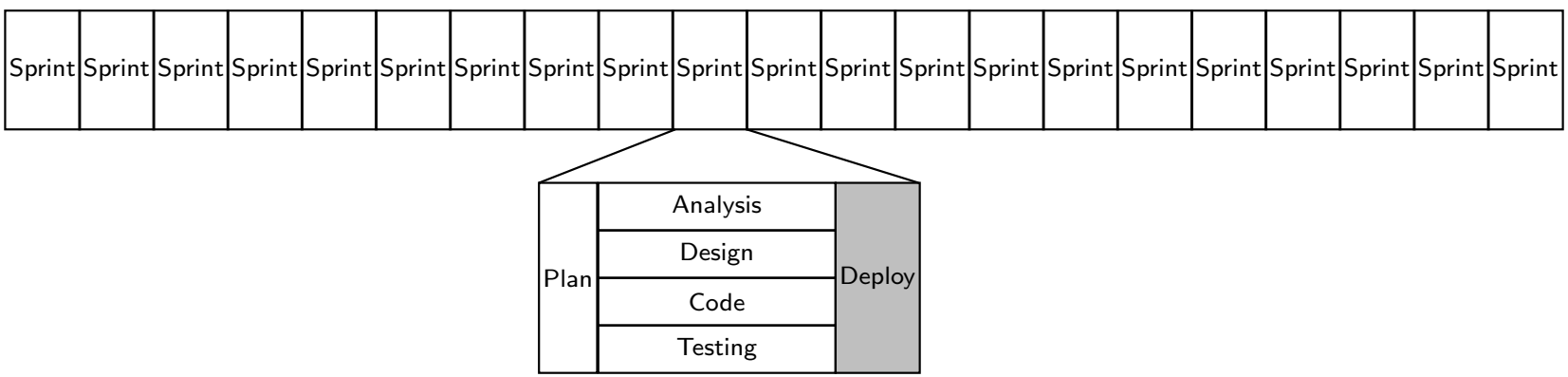


Traditionally Lifecycles: Iterative

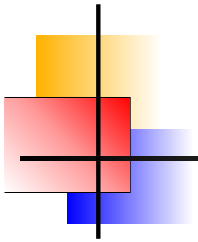
- **Iterative** lifecycle divides the project in smaller periods (1-4 months) with the same phases
  - Plan, analysis, etc... are performed several times along the project
- It reponses better to changes and subsequent periods can fix problems found in earlier periods
- As the project is deployed several times, the team can get feedback more frequently
- The phases are still clearly distinguished

The eXtreme Programming (XP) lifecycle

- The idea of XP is that the team works on all phases every day
- Releasing periods are very short (1-2 weeks)
  - Sprints start with a planification meeting
  - The team works on analysis, design, code and testing every day
  - At the end of each sprint the team must deploy working software with a subset of features

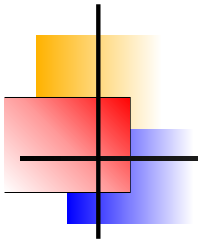


- It does not mean that the team are more productive
  - It means that the team has feedback more frequently
  - It is easier to refine an idea iterating on it multiple times
  - Any information learned can be applied in suqsequent periods



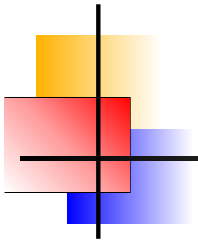
The eXtreme Programming (XP) sprints

- XP teams works on **sprints** of 1-2 weeks
- In every sprint, the team does a bit of planning, a bit of analysis, a bit of design, ...
- The basic notion are the **stories**: small features, or part of features, that have customer value
- In every iteration the team works on some stories (4-10 ideally)
- The stories selected for the sprint should be finished and deployed at the end of the sprint
  - At least, for interal review
  - In some cases, the stories are deployed to actual customers



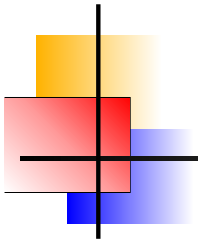
The eXtreme Programming (XP) phases

- Planning phase
  - Should include the customer (or the product owner), who is the responsible for making business decissions
  - Some stories must be selected to be done in the current iteration
  - The planning phase is more intense in the beginnig of the projects and will be reduced for the next phases
- Analysis
  - It is recommended that the customer is also involved in this phase
  - Programmers figure out the concrete requirements for a story
  - A list of tests should be produced at this phase for the considered stories



The eXtreme Programming (XP) phases

- Design and coding
  - **Test-driven-development** is the heart of this phase as it weaves toghether testing, coding and design (by refactoring)
  - Programmers uses **version control systems** for code managing
  - Uses **build tools** for automated build
  - Programmers uses **continous-integration** techniques to ensure that every integration is correct and deployable
  - The quality of the code can be evaluated and guaranteed by means of **software analysis tools**
- Testing
  - TDD is the first line of defense as it ensures that the project has an automated unit and integration test suite
  - With this test suite, regression testing is not needed
  - Customers can test the features added in the stories included in the sprint

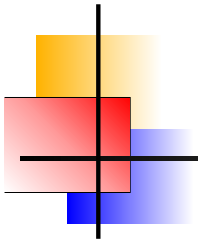


Agile concepts: Backlog

Definition

The **Product Backlog** is a prioritized list of all product requirements and goals and represents the customer expectations with respect to the project releases

- The elements of the backlog of the project could be:
  - **Features/epics**: A functionaltity to be develop in the project
  - **User-stories**: Fine-grained pieces of funcionalities that comes from a feature
  - **Bugs**: Problems to be fixed in the project
  - **Work Items**: Other things to be done (e.g. write the README, refactor something, ...)

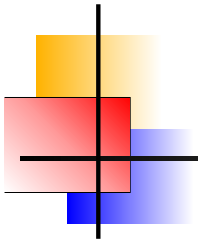


User Stories

Definition

A **User Story** is a small (actually, the smallest) piece of work that represents some value to an end user and can be delivered during a sprint

- A user story is a sentence in simple language that outline the desired outcome
- A user story describes a feature or part of a feature, or requirement, to be implemented
- Stories should be **I.N.V.E.S.T**:
  - **I**ndependent: stories can be delivered in any order
  - **N**egotiable: the team can decide how to implement it
  - **V**aluable: it has some value for the end user
  - **E**stimable: it is easy to estimate the time it will take
  - **S**mall: it can be developed within a sprint
  - **T**estable: it should have a clear acceptance criteria



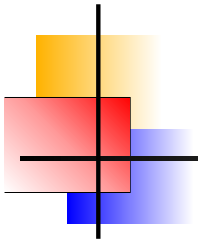
User Stories

- User stories should be described by the customer (or by te product owner)
- A user story usually follows the pattern:

  As [role], I want [whatever] so that [why?]

  > *As player, I want to move the warehouse man inside the level so that I can be closer to pass the level*
  > *As card owner, I want to add money to my card so that I can buy more things*

- Stories are generally *done* when the user can complete the outlined task
- A user story should include a clear acceptance criteria

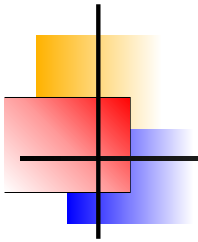


Agile concepts: Milestones

Definition

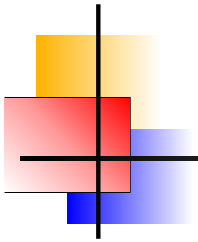
A **milestone** is a checkpoint at a concrete date at which the team checks the progress

- Releases always corresponds to a milestone
- However, the project could have more milestones, e.g. the sprint deadlines
- All elements in the backlog must be assigned to a milestone and should be finished in the milestone deadline



XP Methodology in your projects

- We are going to use `GitLab` to support the Backlog
- Your projects must have, at least, **two sprints** and, thus, two milestones
  - The date of the first meeting with the lecturer (December)
  - The deadline of the project (January)
- You have to use `GitLab` issues and labels to register the **backlog elements**
  - Features must be added with label **feature**
  - Stories must be added with label **user-story**
  - Bugs must be added with label **bug**
  - Work-items must be added with label **work-item**
- All backlog elements must also have a **priority label** (**high**, **medium**, **low**)
- **Planning and review** must be registered as an issue (closed) with the label **meeting**



Further Reading

James Shore and Shane Warden, **The art of agile development**, first ed., O'Reilly, 2007.