Software methodologies

Traditional methodologies

As developing software products is not an easy task, there are some different methodologies that we can follow for this development. A **methodology** is a set of techniques and methods that help us face each stage of a lifecycle. This way, we can:

- Optimize the process and the final product
- Use guided protocols for the planning and the development
- Set what to do, how to do it and when, along all the project

1. Elements of a methodology

A methodology is composed of:

- **Stages**: a set of activities that need to be done in order to get an intermediate product, or finish a step of the process
- **Products**: the set of inputs and outputs required/produced by a given stage
- **Procedures and tools**: elements that help us do each task. Examples: code editors, planning software...
- **Evaluation criteria** for the process and the final product, in order to determine if all the targets have been reached

2. Traditional methodologies

Some methodologies are **traditional**, they focus on controlling every part of the process at any time. They set the activities to be done, the deliverables that must be produced in every stage, and the tools and standards that will be employed. For certain projects, these methodologies can be too rigid, and hard to adapt.

Mosf of these methodologies are based on some of the lifecycle models seen before, specially the iterative and spiral models. Both of them apply several iterations over the software development stages, generating some intermediate products along the process.

Let's see some of the most popular, traditional methodologies.

2.1. Example: RUP

Rational Unified Process (RUP) is an iterative framework created by Rational Software Corporation, which is part of the IBM company. But, instead of being what a traditional methodology is supposed to be, RUP is not a rigid framework. It can be adapted to different projects, choosing the most suitable elements for each one.

RUP methodology is based on spiral lifecycle model, and it tries to improve the drawbacks of the cascade model. Besides, it also tries to include the object oriented paradigm through UML (Unified Modeling Language). We will talk about it in later units.

Modules

RUP is structured in some elements called **modules**. These modules are:

- **Roles**: they define a set of competences, abilities and responsibilities, so that people with a specific role must have all of them.
- **Work products**: they are the result of a task, including documents and secondary models that may be produced
- **Tasks**: work units, which are assigned to a specific role, and produce a given work product.

These three modules define who does the job (role), what it is about (work product) and how he does it (task).

Stages

According to RUP methodology, the lifecycle of a product consists of four stages

- **Inception**, where we define the scope of the project. We create a model for the business (gathering some documentation about the company, finding out its weaknesses and strenghts, its capacity of producing benefits...), and then a requirements analysis for the project to be developed.
- **Elaboration**, where the project is analyzed more in depth, so its main architecture is defined. We create an initial design an implementation, which are considered a basis. This stage, along with the inception, are targeted on understanding the problem to be solved, skipping the most dangerous risks.
- **Building**, where the application is finally designed and implemented. We can need many iterations of the spiral model, with many intermediate prototypes, to reach our target.
- **Transition**, where the final product is prepared for its final delivery to the customer.

We can think that this methodology is very similar to the cascade model, but one of the strenghts of RUP relies on the iterations that can be performed at any stage. Besides, each stage has a final target to reach.

> **Exercise 1:**
>
> Search on the Internet about all the possible diagrams that we can use in UML (*Unified Modeling Language*), which is an essential part of the RUP methodology. These diagrams can be of two types: structural diagrams or behavioral diagrams. Identify at least three diagrams of each type.

2.2. Example: MSF

Microsoft Solutions Framework (MSF) is a customizable methodology that applies a traditional approach to develop a software product. As you can figure out, it was created by Microsoft, and it can be applied not only to software development, but also to other computer issues, such as network planning, infrastructures...

Stages

As we have seen for RUP, MSF is also based on the spiral model. The process consists of 5 iterative stages. At the end of each one we must have reached a given target, and completed a set of deliverables.

- **Vision**: we evaluate the business model, the benefits that we can have, restrictions, scopes... We must also get a requirements specification, an initial risk assessment, and a general overview about the company for which we are going to develop the software product. It may be equivalent to the *Inception* stage of RUP.
- **Planning**: the project development and its architecture are planned in this stage, so that we must follow a schedule. This way, we generate a list of tasks to be completed, people involved in each task, responsibilities, costs... trying to avoid the most dangerous, potential risks.
- **Development**: we begin to implement the application from the very primary functionalities. We deliver some prototypes to be tested and evaluated by the customer
- **Stabilization**: the product is tuned so that the customer can test it completely. The set of tests is also documented before the customer agrees with the final product.
- **Implantation and support**: the application is installed in the customer's company, and an additional support may also be offered depending on what was initially specified.

The main benefits of this methodology are its adaptability to different projects (as RUP does), and the cooperation with the customer. Their main drawbacks are the excessive documentation that needs to be created, and the dependence on Microsoft products.