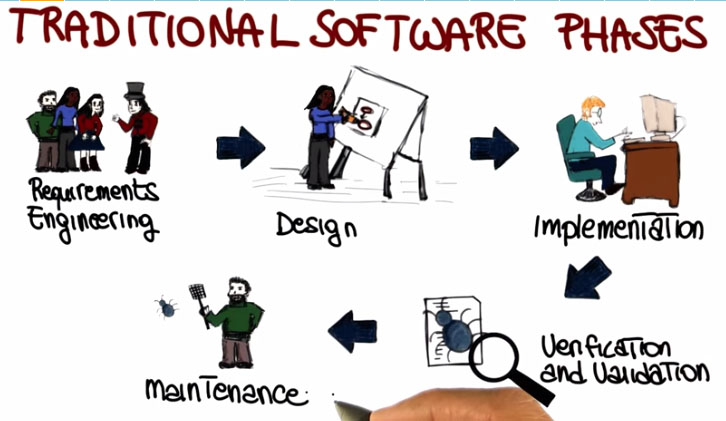
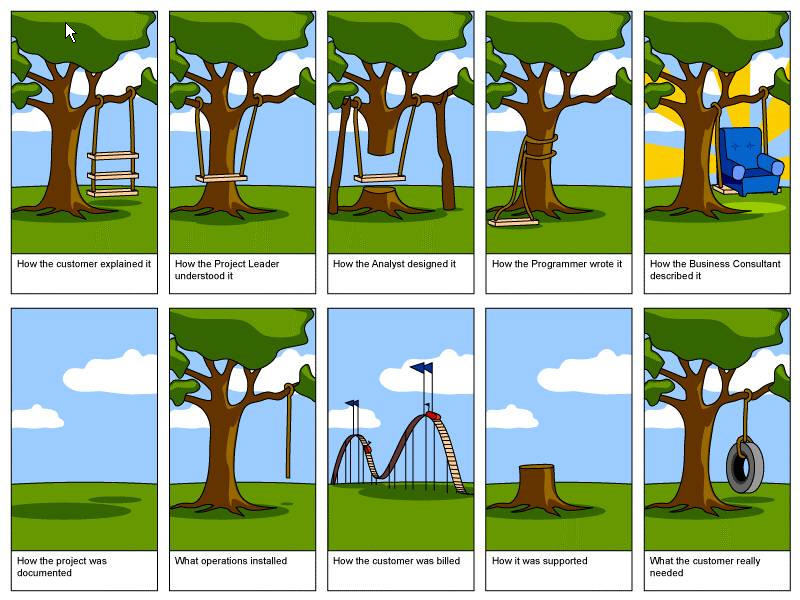
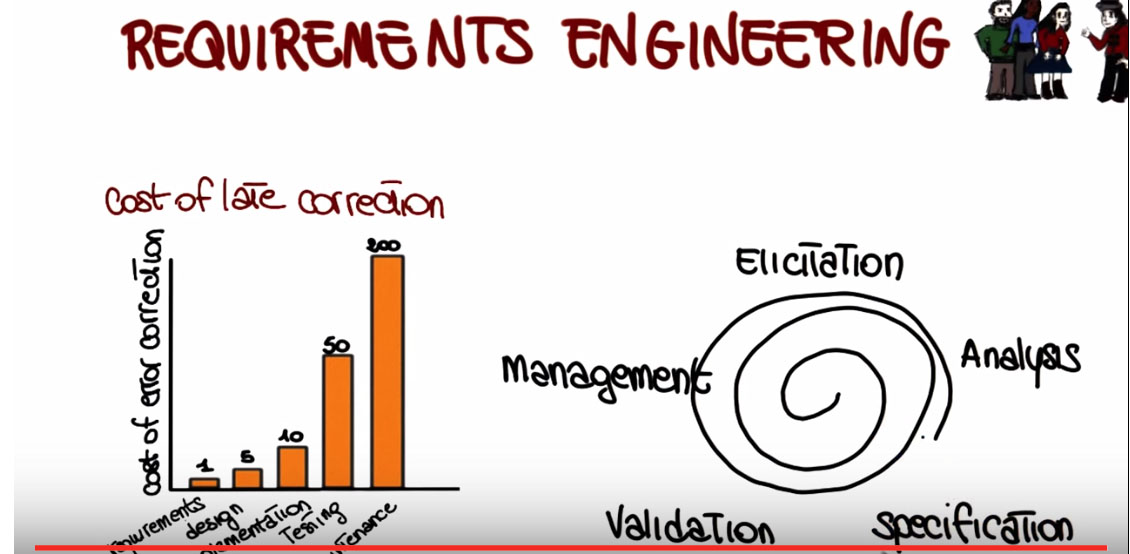
**Lesson 2: Life Cycle Models**



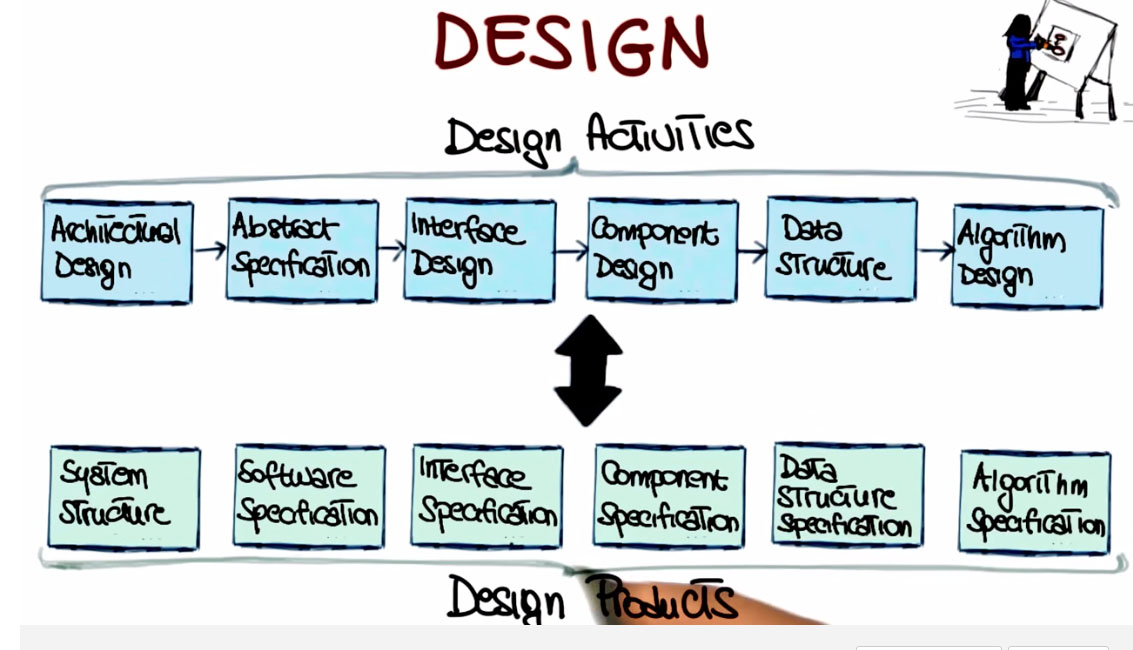
1. **Requirements Engineering**

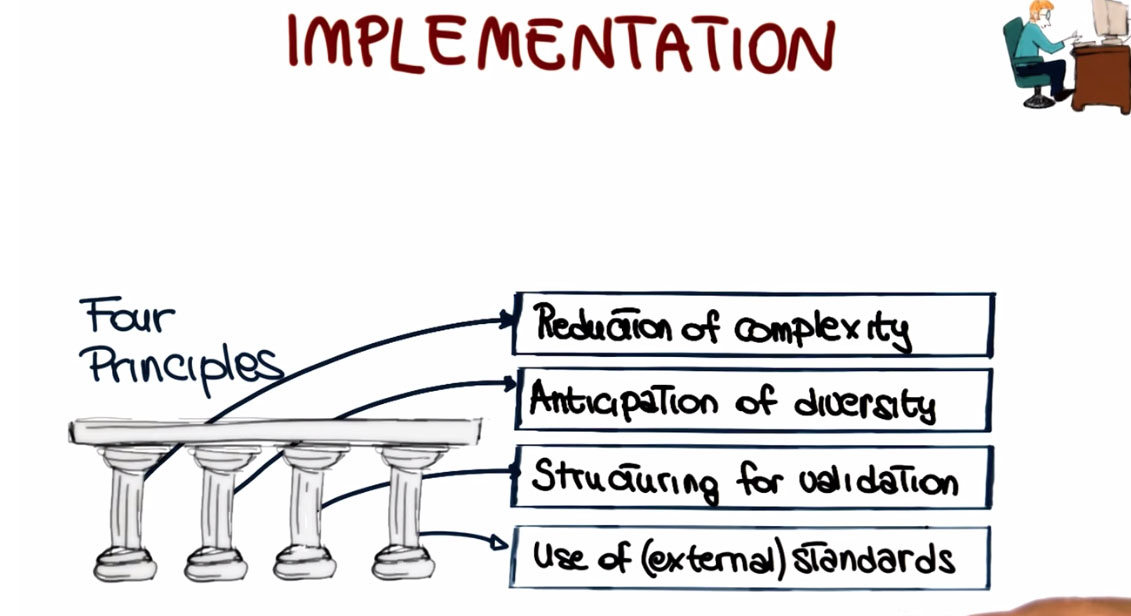


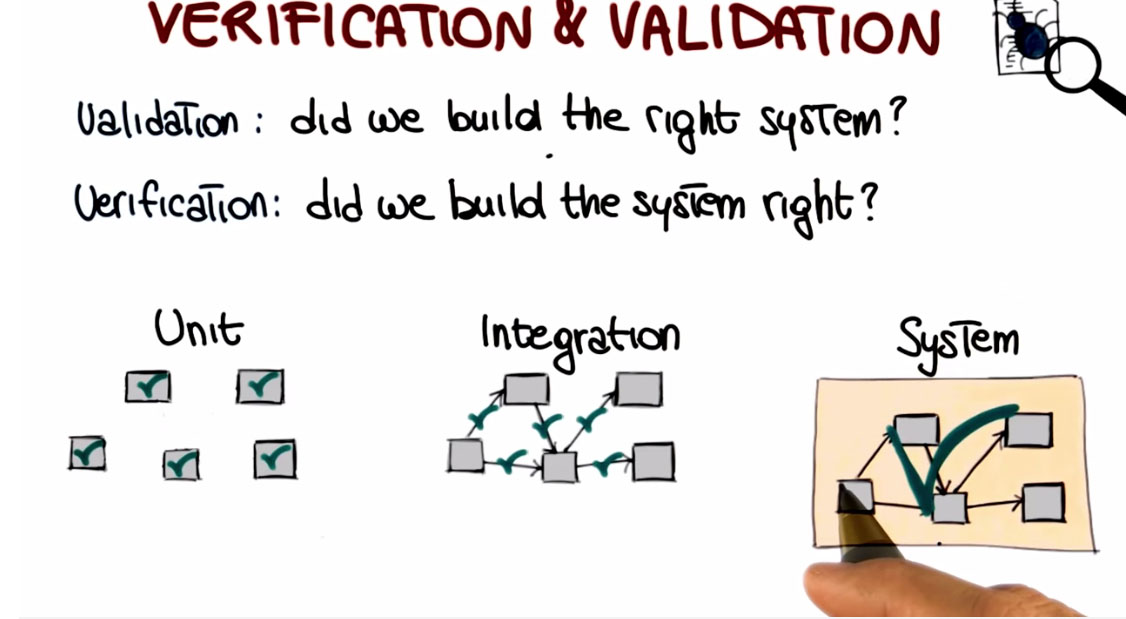


1. **Design**

* Software design is the phase where software requirements are analyzed in order to produce a description of the internal structure and organization of the system.
* The important point is that we go from sort of a high-level view of the system, which is the architectural design, to a low-level view, which is the algorithm design. And these activities result in a set of design products, which describe various characteristics of the system.

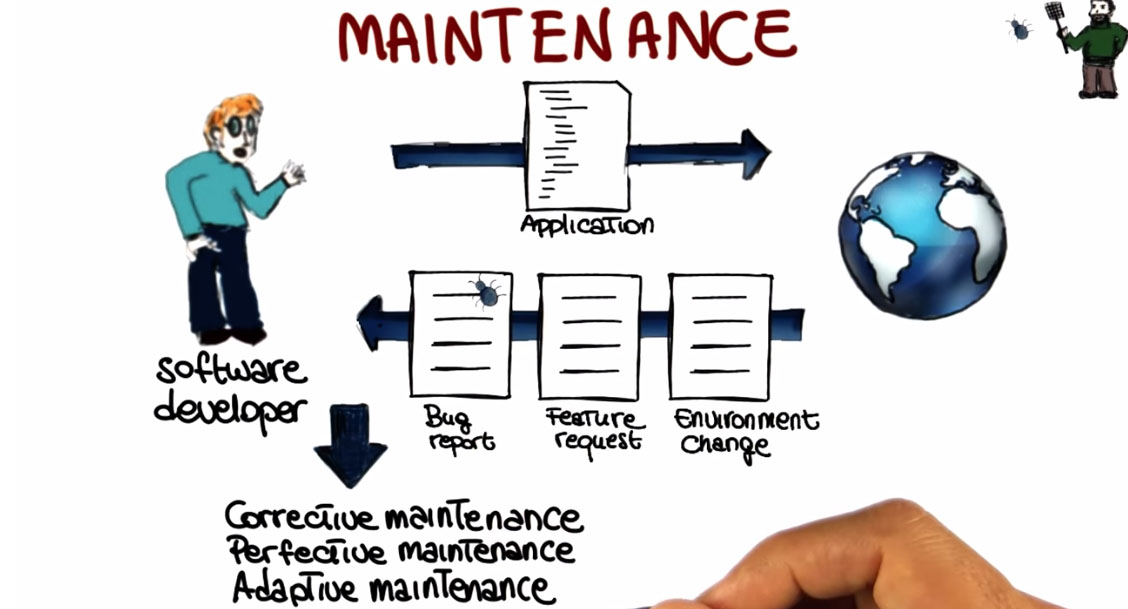






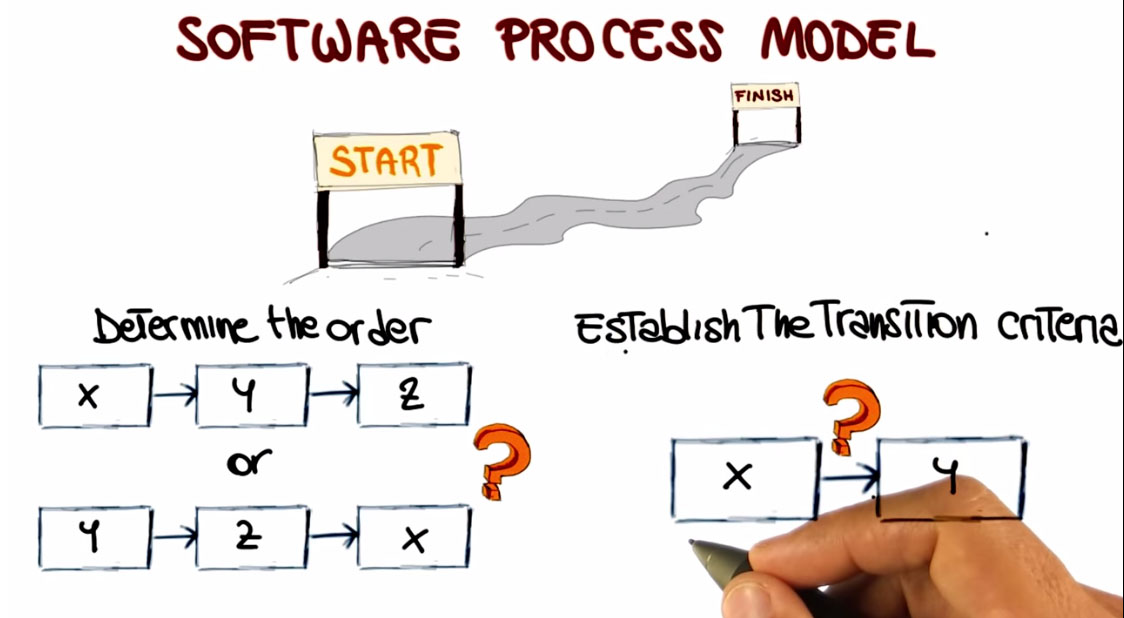
1. **Maintenance**

* During maintenance every time you modify your application you have to regression test the application, where regression testing is the activity of retesting software after it has been modified to make sure that the changes you perform to the software work as expected, and that your changes did not introduce any unforseen effect.



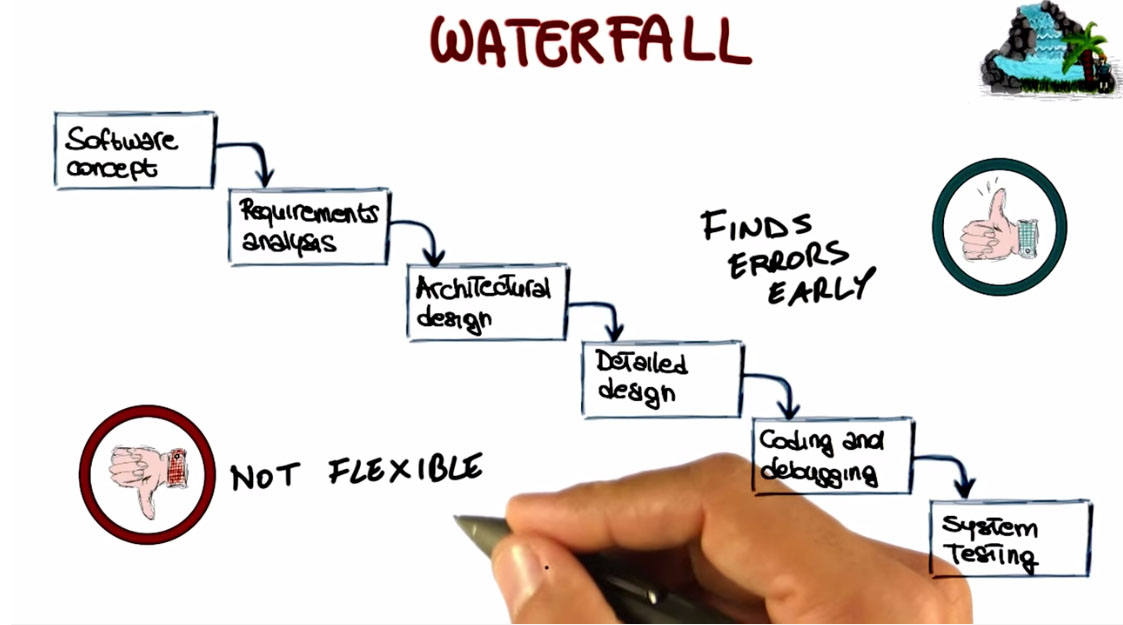
1. **Software Process Model Introduction**

* How should we put these activities together to develop software? And this all comes down to the concept of software process model. Also called software lifecycle model. And what this is, is a prescriptive model of what should happen from the very beginning to the very end of a software development process. The main function of the life cycle model is to determine the order of the different activities so that we know which activities should come first and which ones should follow. Another important function of the life cycle model is to determine the transition criteria between activities



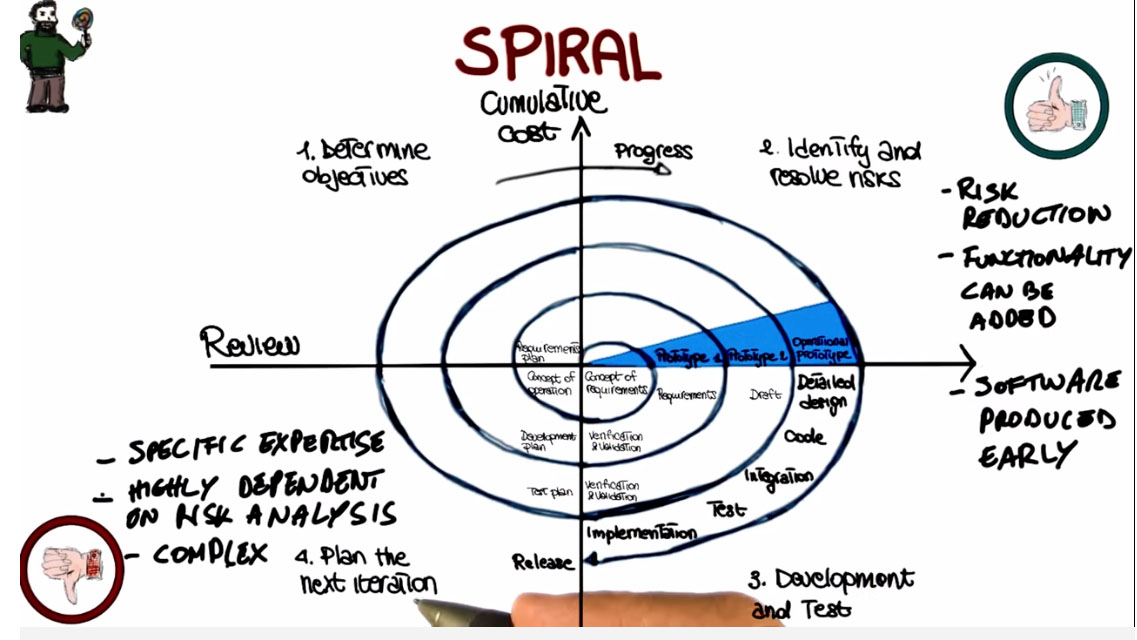
1. **Waterfall Process**

* The pure waterfall model performs well for software products in which there is a stable product definition, the domain is well known and the technologies involved are well understood. In these kind of domains, the waterfall model helps you to find errors in the early, local stages of the projects.
* The main disadvantages of the waterfall model arise from the fact that it is not flexible. Normally, it is difficult to fully specify requirements at the beginning of a project. And this lack of flexibility is far from ideal when dealing with project in which requirements change, the developers are not domain experts or the technology used are new and evolving, that is it is less than ideal for most real world projects.



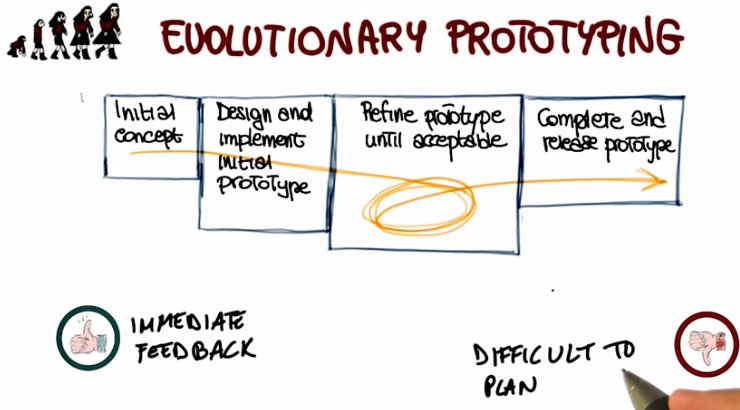
1. **Spiral Process**

* The spiral model is an incremental risk-oriented lifecycle model that has four main phases listed here: determine objectives, identify and resolve risks, development and tests, and plan the next iteration.
* There are several advantages of using a spiral model.
* The first one is that the extensive risk analysis does reduce the chances of the project to fail. So there is a risk reduction advantage.
* The second advantage is that functionality can be added at a later phase because of the iterative nature of the process.
* And finally, software is produced early in the software lifecycle. So, at any iteration, we have something to show for our development. We don't wait until the end before producing something. And then of course there's also the advantage that we can get early feedback from the customer about what we produced.
* The main disadvantages of the spiral model,
* are that the risk analysis requires a highly specific expertise. And unfortunately, the whole success of the process is highly dependent on risk analysis. So risk analysis has to be done right.
* And finally the spiral model is way more complex than other models, like for example, the water fall model. And therefore it can be costly to implement.

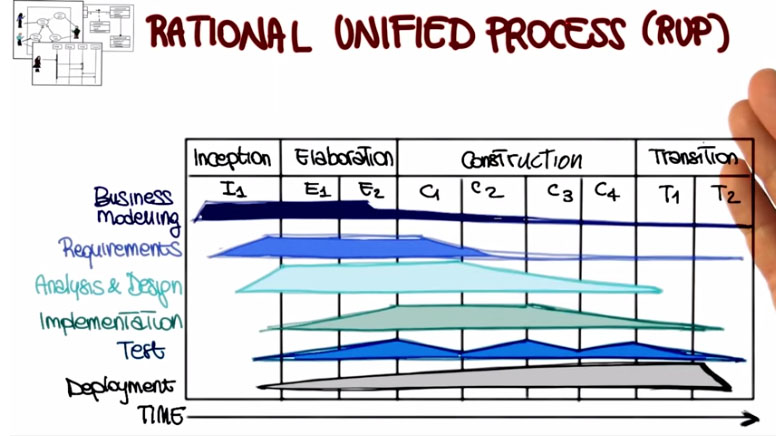


1. **Evolutionary Prototyping Process**

* Throwaway prototyping is another kind of prototyping in which the prototype is just used to gather requirements, but is thrown away at the end of the requirements gathering.
* Extreme Prototyping is used especially for developing web applications. It breaks down web development into three phases, each one based on the preceding one.
* The first phase is a static prototype that consists mainly of HTML pages.
* In the second phase, the screens are programmed and fully functional using a simulated services layer.
* In the third phase, the services are implemented.
* It is an ideal process when not all requirements are well understood
* The main advantage is the immediate feedback. Developers get feedback immediately as soon as they produce a prototype and they show it to the customer and therefore, the risk of implementing the wrong system is minimized.
* The disadvantages:
* It is difficult to plan. When using evolutionary prototype it is difficult to plan in advance how long the development is going to take, because we don't know how many iterations will be needed.
* And another drawback is that it can easily become an excuse to do kind of do cut and fix kind of approaches in which we hack something together, fix the main issues when the customer gives us feedback, and then continue this way, until the final product is something that is kind of working, but it's not really a product of high quality.

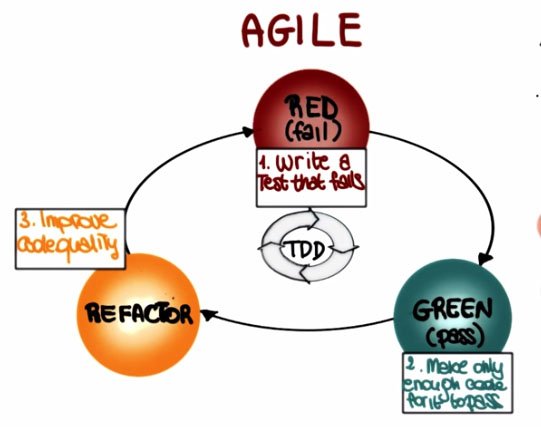


1. **Rational Unified Process**

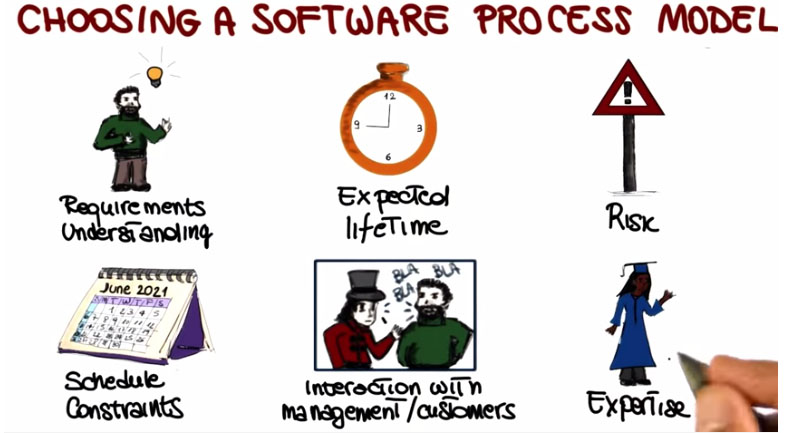


1. **Agile Process**

* Agile Software Development Processes – A group of software development methods that are based on highly iterative and incremental development.
* On Agile technique is Test Driven Development or TDD. 3 phases in each iteration:.
* We write test cases that encode our requirements, and for which we haven't written code yet.
* We write the just enough code to make the test cases pass. When we do this over time though, what happens is that the structure of the code deteriorates, because we keep adding pieces. So that's why we have the third step,
* Refactoring. In this step, we modify the code to make it more readable, more maintainable. In general, we modify to improve the design of the code.
* Extreme programming, or XP, and Scrum



1. **Choosing a Model**



17 -

1. **Lifecycle Documents**

* Documenting the activities carried out during the different phases of the software lifecycle, is a very important task.
* There are standardized document that are provided by IEEE that you can use for this purpose.

