# Software Development

Building large pieces of software can be costly and time consuming.

## Elements of Software Development

**Feasibility Study**

- Software development is costly and large projects are not worth starting if they are going to fail. A feasibility study justifies whether a project is worth starting.

- There are a number of reasons a project might fail:  
 - Budget may not be enough to cover the costs of the project.

- It might not be legally feasible.

- It might not be technically feasible.

**Requirements Specification**

- Setting out easily measurable criteria for success of your program.

- The determining of requirements is traditionally done in the requirements elicitation phase which normally culminates in a document called the **requirements specification** which lists each and ever requirement of the final product.

- When the project is signed off it is tested against the requirements specification in what is known as **acceptance testing.**

- This assures the user that the program will be okay for them.

**Testing**

- Should take place continuously through coding process.

- Every time a module of code is written it should be tested to be certain it works. Theoretically if you know all modules work on their own then you only need to test if they work together.

- Testing should include **destructive testing** where testers try to cause a program to crash or behave unexpectedly.

- Testing can be used to show the presence of bugs but never their absence.

- Once the code is free of obvious bugs then the next phase is **alpha testing.** This is where product is used within the company by people who haven’t programmed it.

- The next stage is **beta testing** this is where the software is tested by a small group of users outside the company.

- The final stage is **acceptance testing** when the user tests the program against every requirement in the user specification.

**Documentation**

- Written documents are produced during the software development process.

- Requirements Specification is produced.

- The system design may be documented to allow the programmers to understand what they are doing.

- As the system is built, it may be documented to allow software engineers to be able to understand and maintain it in future.

- As the system is built, it may be documented to allow software engineers to be able to understand and maintain it in future. This is referred to as the **technical documentation.**

- Another important type of documentation is **user documentation.** This is effectively the manual that tells the user how to use the software.

## Methodologies

- To ensure software is delivered on time and budget, different budgets have been developed. These methodologies have been developed.

- These methodologies will have the stages above but take different approaches as to when extent.

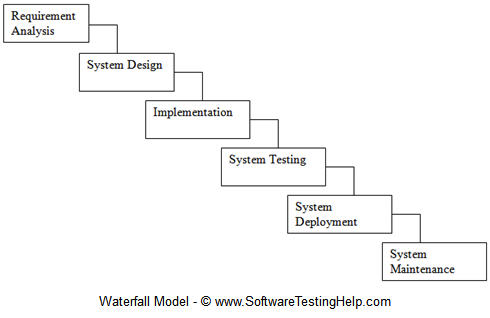
**Waterfall Life-Cycle**

- The waterfall cycle is a well-known (and often criticised) development model.

- It consists of a sequence of stages. In its more basic form, each stage is started only if the previous has been completed.

- This only works if each stage is completed perfectly the first time.

- It has been proposed that the system can be improved if you allow iteration between adjacent stages.



**Advantages and Disadvantages**

- Simple and easy to manage.

- Clear to see if a project is running to schedule as there is an expected output on the end of each stage.

- Carries a lot of risk. It isn’t until the testing stage that the user gets to see something tangible.

- If the requirements have been misunderstood it may be very difficult to rectify mistakes.

- For this reason the waterfall life-cycle is more suited to less complex problems where the requirements are fully understood.

**Rapid Application Development (RAD)**

- Involves the use of prototypes - a version of the system that lacks full functionality.

- This could be anything from screen mock-ups to final versions of the system.

- This means there is something to show the user at the end of every stage.

- So changes can be made according to what the user wants at each stage.

- This means that errors can be amended before the developers move on to the next stage of development.

- The cycle of prototyping and evaluation continues until the program has all the functionality the user wants and they can approve it.

- At this stage it becomes the final product.

**Advantages and Disadvantages**

- Well suited to projects where requirements aren’t clear from the outset.

- With continuous feedback from the user the final product is likely to have excellent usability.

- As the focus is usability there is often not a focus on how the product works and therefore RAD is not appropriate for projects where efficiency of code is important.

- It is important to have continual contact with the client for RAD so its unsuitable in situations where the user cannot make this commitment.

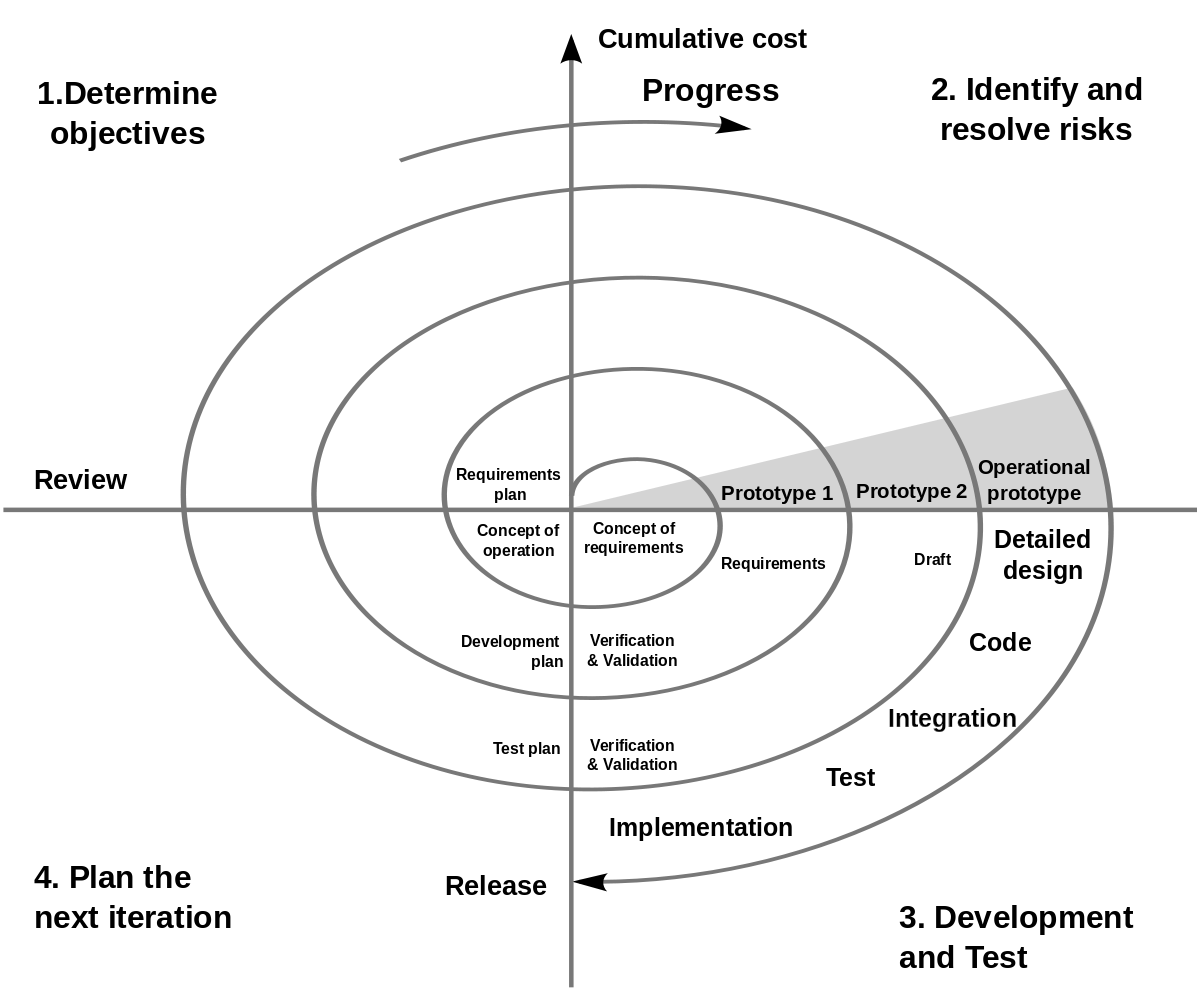
- RAD doesn’t scale well and thus is less suited to large projects with big teams.

**Spiral Model**

- Software development can involve high amounts of risk. Projects can run out of time, requirements can change and competitors can change and competitors can come out with better alternatives.

- The spiral model is designed to take into account risks within the project.

- By focusing on managing risks this can be dealt with before they become issues.



- The model consists of four stages, each forming a quadrant of the spiral. The first stage is to determine the objectives of that rotation of the spiral.

- In the first instance this may be determining the requirements of the project.

- In the next stage the possible risks are identified and alternative options considered, this may involve building a prototype of the system.

- If risks are considered too high at this stage the project may be stopped.

- The third stage allows the project to be worked on to be made and tested.

- After this stage there is a stage to decide what will happen in the next iteration of the spiral.

- There will be a product at the end of each stage of the spiral which can be shown to a user.

**Advantages and Disadvantages**

- The fact that risk is at the heart of the spiral model is its biggest advantage and means it is ideal for projects that are likely to be high risk.

- Large projects in particular tend to be high risk and are thus suitable to this model.

- Risk analysis is in itself a very specialised skill which means the model is only as good as the risk analyst working on it.

- Good risk analysts are expensive which adds cost to the project.

**Agile Programming**

- Agile programming isn’t a single methodology but a group of methods. These methods are designed to cope with changing requirements through producing software in an iterative manner.

- The idea is that in each iteration more functionality is added and the user can choose throughout the project to add more functionality in a future iteration.

**Extreme Programming**

- An example of an agile programming methodology.

- A representative user becomes part of the team. They help decide the **user stories** - essentially requirements and decide what tests will be used to show they are properly implemented.

- Iterations are much shorter in XP than in RAD - typically a week.

- While RAD uses prototyping, XP produces a version of the system every time just with limited functionality. The code needs to be good enough in quality to be used in the final product.

- At the start of each iteration the user goes through the **planning game**. This involves deciding what the user stories will be and how the team will divide the work.

- One of the key features is pair programming.

- One programmer **driver** the other is the **navigator.** The driver will use the keyboard while the navigator analyses what has been written.

- The two switch roles regularly ensuring that code works.

- One of the principles of XP is that no programmer should work more than a 40 hour week as code quality will start to decrease.

- Testing is carried out continuously every module is tested as soon as it is made in what is known as **unit testing.**

- Once a module is known to work it is integrated into the main code version so that everyone has access to it.

**Advantages and Disadvantages**

- With such emphasis on programming the quality of the code is likely to be very high.

- A project done using XP is carried out using a team of programmers who are able to collaborate well in the same building. (XP doesn’t work well if the team is distributed across the globe.)

- The client needs to be able to commit to having a representative working within the team.