

System development models for software

The stages of the waterfall model (and other general system development lifecycles)

With this methodology, less customer interaction is involved during development. The product is only shown to the end-user when it is functioning at the end of the cycle.

1. FEASIBILITY/DEFINITION

All activities in this stage are directed toward answering **whether or not, early on, if a program or solution to a problem will work** and be **economically viable**. If the conclusion is no, then we **save time and money** by not going forward with the project.

2. REQUIREMENTS

We **analyse carefully what the user wants** and we **refine the project goals** into a set of requirements. This stage will produce a **detailed requirements specification** document. This outlines **everything that the user wants** the program to do. This document is **checked and signed off** during acceptance testing later on in the testing/implementation phase.

3. ANALYSIS AND DESIGN

The **requirement specification document is given to a design team** who **determine how the solution will be accomplished**. This phase describes **features and operations in detail**; with **mock screen layouts, business rules, process diagrams and pseudocode.....** A test plan may also be produced to ensure all parts of the system will work.

4. IMPLEMENTATION

Only once it is agreed that the team know what the program **needs to do and how it is to do this task**, the documentation of a **design specification is handed to programmers**. **Code** for the program is written.

5. TESTING

A **testing environment** checks all the parts of the **implemented code for errors and bugs**. The program code is also **compared with the system requirements specification to ensure it works as intended**. This stage takes a long time and is **very important**. Numerous **different testing methodologies can be adopted**.

In reality, testing usually occurs **combined with the implementation process** to ease the flow through the waterfall model (reducing time and effort needed to pass code between a design and testing team).

6. EVALUATION

The **program is evaluated against a detailed criteria** based on the **original design specification** given by the **client and analyst**. This stage aims to ensure the program **does what is required** and **does not miss** any key **functionality**.

7. INSTALMENT/ DEPLOYMENT

The product is **only given back to the client once the functional and non-functional testing has been completed** and the **evaluation** concludes the program **functions as intended**. **Acceptance testing** may be used to give the **client** a chance to use the program and confirm it works as intended so the client may **pay the developer**. The software is **installed on user machines**. Evaluated on basis of: **effectiveness, usability** and **maintainability**.

8. MAINTENANCE

Some issues come up in the **client environment** as it is **impossible to test all parts** of a program with **every possible input**, moreover, the users may use to program in a **way not anticipated by the development team** and **identify problems**. To fix such issues: **patches, updates with bug fixes** and **newer versions** are released.

More about the design and analysis phases

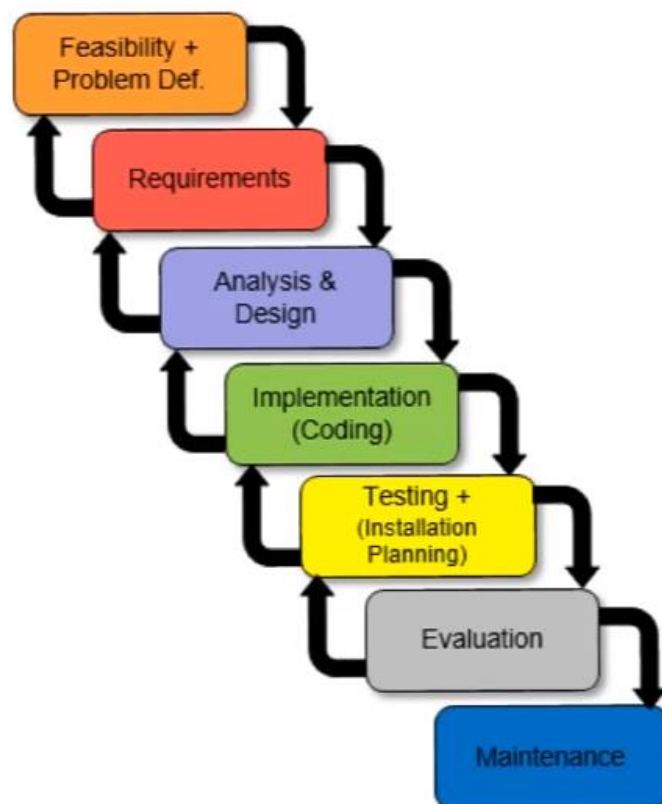
Analysis: Before a problem can be **solved, it must be defined**. The requirements of the system that solve the problem must be established. E.g. for a data processing system in a construction of a website.....

- **The data** – origin, uses, volume, characteristics
- **The procedures**, what is done, where and how, how errors/exceptions are handled.
- **The future** – development plans and expected growth rates.
- **Problems** - with any existing system


Design: some of the following may be considered.....

- **Processing** - The algorithms and appropriate modular structure for the solution, specifying modules with clear documented interfaces.
- **Data structures** – how data will be held and accessed (dynamic structures like trees or queues, or in a file database).
- **Output** – content, format, sequence, frequency, medium (screen or hard copy) etc.
- **Input** – volume, frequency, documents used, input methods.
- **User interface** – screens and dialogues, menus, special-purpose requirements
- **Security** – How the data will be kept secure from accidental corruption or deliberate tampering or hacking.
- **Hardware** - selection of an appropriate configuration.

The waterfall lifecycle model



The Waterfall Model illustrates the methodology described previously:

- 
1. Feasibility/problem definition
 2. Requirements specification
 3. Analysis and design
 4. Implementation (coding)
 5. Testing and installation planning
 6. Evaluation (documentation)
 7. Installation/ deployment
 8. Maintenance

Each step is completed **one step at a time consecutively**. Each step has **specific outputs leading to the next step**. It is possible to **return to a previous step** meaning that they will have to **re-work previous stages** in the **light of experience gained as development progresses**.

The **user/customer is involved** in the **start** of the process, the **Analysis stage**, but then has **little input until the Evaluation stage**.

For this reason, if the **design specification requirements** are not interpreted correctly or if a **coder overlooks the design requirements/misinterpreting desires** then earlier stages must be reworked and this is very expensive.

The model was adopted from the **manufacturing industry**, where **changes** to hardware made **later in the project had high cost** implications to work already completed so it was **important to get each stage right** before moving onto the next stage. This is still popular but has now been superseded by more effective models.

Advantages	Disadvantages
Model is easy to understand and implement	Not suitable for projects with a moderate to high risk factor of changing requirements .
Easy to manage due to the rigidity – each phase specific deliverables and a review process	Once the application is in the testing stage, it is difficult to go back and change something that was not well thought out in the concept stage .
Phases do not overlap and are completed one at a time (may ensure each phase is done correctly to avoid costly errors)	No working software is produced until the end of the cycle so little feedback from the buyer/user and hard to track progress .
Works well with projects where the requirements specification is understood well, is clear and fixed .	Not a good model for complex, object-orientated projects .
With each stage having clear deliverables, it is easy to see the plan timescale if certain stages are taking too long . Progress tracked	End user not involved much in the development

When do we use this methodology then?

- When the requirements are clear, fixed so easily understood
- There is no ambiguity in the design specification.
- Ample resources with required expertise are available freely.
- The project is short

Rapid application development = a general term used to refer to alternatives to the conventional waterfall model.

Less emphasis is put on the **planning of a project** and **more focus is put on the process** (in contrast to the waterfall model that rigorously defines the requirement specification prior to entering the design stage).

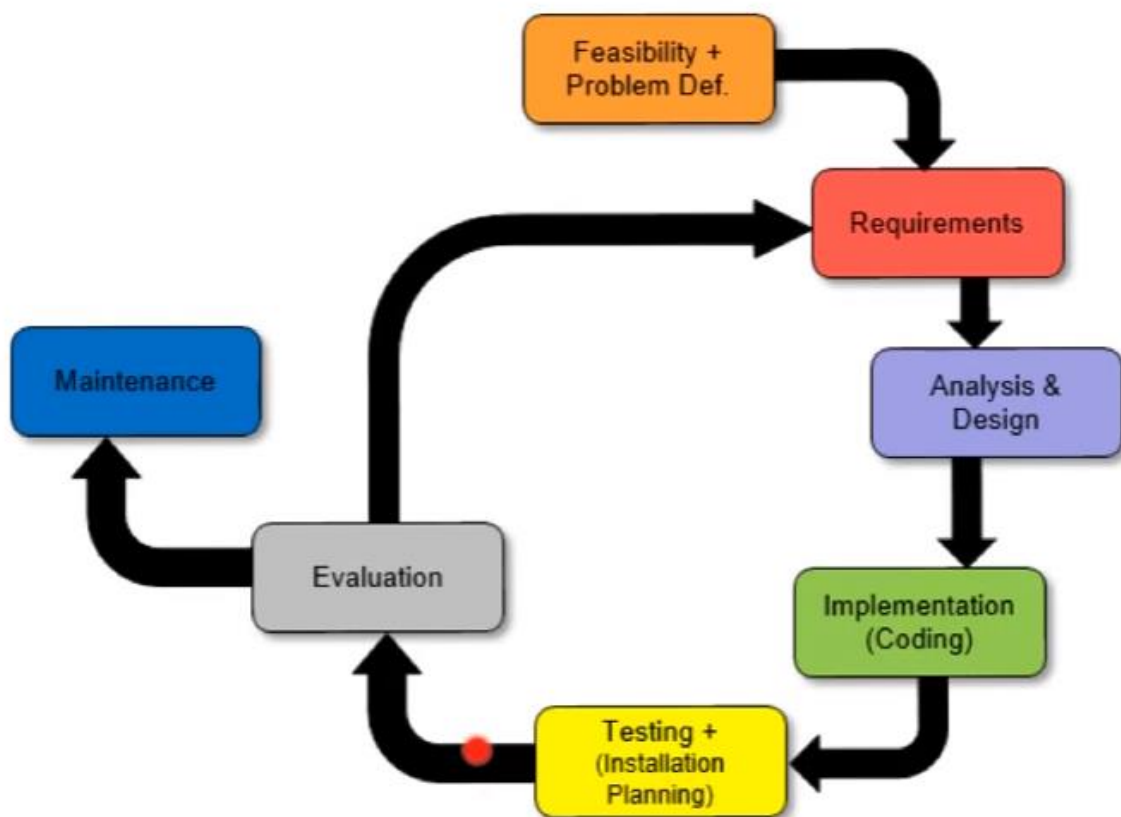
Adaptability is at the forefront of design, requirements are adjusted as knowledge is gained through the development process.

The model works by **successively prototyping the software** until a final version has been produced. This is **far quicker than having to wait until the end of the cycle** like in a waterfall model.

Several increasingly refined prototypes are made.

These are **designed, coded and tested before being evaluated by the user**. The user decides if the program meets all their needs and feedback is given to plan the next iteration of a refined prototype. **Each cycle can last several weeks**.

Rapid Application Development (RAD)



This method is adopted for large projects that made have development over a long period of time; during which technology and user requirements change. In methodologies like the waterfall model this change in user requirements alter on meant cost implications as earlier stages had to be reworked. RAD offers the promise of delivering a product or at least a prototype of it much faster so feedback is obtained.

Idea behind the methodology include:

1. **Workshops and focus groups** (with **scrum masters** ,**leading teams** and supervising their **progress**).
2. **Using prototyping** to continually refine the system **in response to user involvement** and feedback.
3. Producing a prototype in **strict time limits** – each part of the system may not be perfect in prototype but good enough for feedback and insight.
4. **Reusing any software components** that have already been used elsewhere.

Advantages	Disadvantages
The user feels more involved in the process – seeing successive prototypes.	Regular contact needed with the client
The end product is more likely to match the user requirements: saving time, money and effort.	Does not scale well to large projects
The iterative manner makes it easy to account for adjusting user requirements over time.	Not suitable if efficiency of code is priority.

The Spiral Model

The Spiral Model uses the **same structures steps as the waterfall method** but introduces the idea of developing the software in **iterative (repeating) stages**. The main difference is it is a risk driven approach.

At the start of each stage, the **requirements are defined** and the **developers work towards an initial prototype**. **Each successive loop** around the spiral **generates a refined prototype** until the product is finished. User interaction may not occur until the 4th stage (feedback from the user) although this will only be at several week intervals.

The method has 4 stages that each take up one quadrant of the spiral. Each time around the spiral, the following activities are performed.

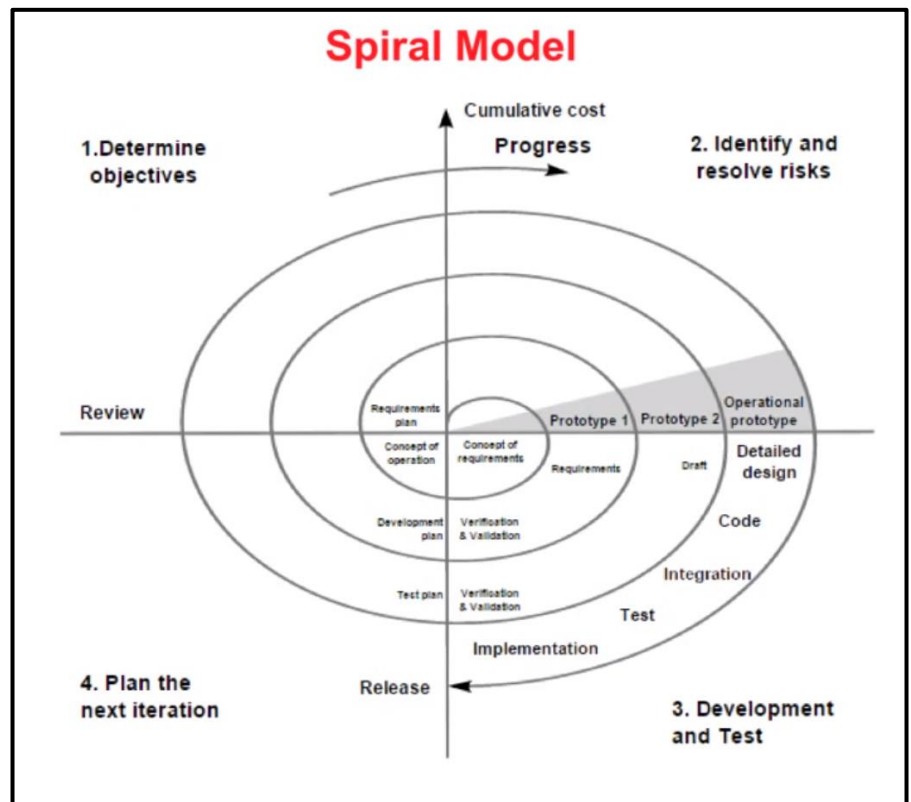
1. **Analyse** the **objectives** and **requirements** for the **next prototype**. **The next prototype is designed**.
2. The **risks and issues** are identified and a prototype is developed (coded by implementation).
3. **Testing** of the **refined prototype**.
4. **Evaluate** the new prototype, which **generates a plan for the next iteration** – **or if the project meets all user requirements it is completed**.

Advantages	Disadvantages
Focusses on risks and issues – considering these in the refined prototype.	Risk management is specialised and costly as professional and competent risk management is needed.
Works well for large projects	May have less contact with the client than agile or extreme programming methods.
Can adapt to changing design requirements as missing functionality is incorporated into the next refined prototype.	
User can give feedback at the end of each prototype iteration/spiral.	

The Spiral Model is mostly used for **large scale projects**, for example Photoshop CC, projects that **take years to deliver**.

Iterative development – each cycle can last several weeks. The spiral model has aspects of waterfall, incremental and RAD.

The difference is that, this is a **risk driven approach** to development. It takes into account that risk is at the heart of many large scale development projects and so is designed to cope with risks and deal with them during the project before they become major problems.



Agile modelling

Small projects use a variation on this methodology called **Agile Model**.

These are a **group of development methodologies** – not a singular one.

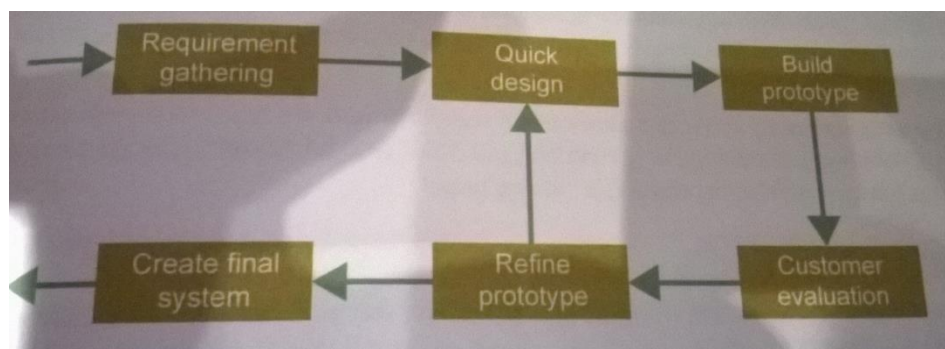
They **focus on the idea that user requirements may change** during development. This can only be dealt with by using an **iterative manner** – with each **iterative prototype having increased requirements** and being **shown to the user**.

At all the stages of analysis, design and implementation, an **agile approach may be adopted**, as the stages of software development may **not be completed in a linear sequence**.

It may well be that some **analysis is done** and then some **parts of the system are designed and implemented** while **other parts are still being analysed**. Then, for example, **implementation and testing may be intermixed**. The developer may **go back to design another aspect of the system**.

Throughout this process, **feedback** may be obtained from **the user**; this is an **iterative process** during which changes made are **incremental** as the **next part of the system is built**.

Typically the software developer's do **just enough modelling** at the start to make sure that the **system is understood by themselves and the users**.



Each stage is built with user participation to ensure that the system is being developed in line with what the user wants. The success of the software development depends on:

1. Keeping the **model simple**- So **not trying to incorporate features** which may **come in useful** at a **later date**.
2. **Rapid feedback** from the user to ensure development is in **line with the user needs**.
3. **Understanding** that the **user requirements may change** during development as they are forced to consider their needs in detail.
4. Being prepared to make **incremental changes** as the model **develops**.

Extreme programming

Extreme programming (XP) is a software development methodology that is **intended to improve software quality** and **responsiveness to changing customer requirements**.

It is a **type of agile software development**, in which **frequent “releases”** of the software are made in **short development cycles**.

This is intended to **improve productivity and introduce checkpoints** at which new customer requirements can be adopted.

It involves the iterative process much like rad but the iterations are very short – **often a single week long**.

A company requesting the software often **embeds a user into the development team**. New requirements are built into successive iterations quickly. The user gives **instant feedback** on the next iteration.

Paired programming may be used with a person who types the code and another who sits and analyses this to give instant feedback.

Extreme programming (XP) and agile methodologies	
Advantages	Disadvantages
Quality of end code is very high (particularly when user is in the development team or paired programmers are used)	Increases development costs
Efficient code with fewer bugs (likewise)	Heavy collaboration with the client is needed
More user interaction so more likely that the end program meets user specifications.	Embedded end user is needed. Strict time limits mean each prototype may not meet the requirements and you have to wait until next iteration for refinement.

When to use each approach

Waterfall system lifecycle is suitable for **very small projects** which need **careful supervision**, such as those undertaken by **students or trainees**. The **absence of user involvement is a serious drawback**.

The spiral model and the agile model are an improvement in that they **acknowledge that users often cannot specify their requirements accurately** because they **don't understand what is possible**. It is much easier to **examine a prototype** and figure out what needs to be done to turn it into a useful system.

Extreme programming and **rapid application development** are good methodologies for **larger projects** where there is a **danger of getting bogged down** or **side-tracked by suggested improvements**, so that developers are constantly chasing a moving target.