1.2.3 Development Methodologies

Stages of a Development Cycle

Analysis

The problem is investigated and the requirements of the new system are identified.

Considerations may include

- What data is available.
- How can it be used.
- What future plans for the project are.
- What problems may arise.

Design

During the design phase the system designer may consider.

- How the data will be processed, including the algorithms that will be used.
- The data structures and user interface that will be required.
- What the input and output of the system will be.
- The security of the system and the hardware used.

Implementation

The system is developed by breaking the problem into smaller, more manageable components.

During implementation,

- The code will be written in the chosen programming language.
- The code is tested to identify and correct errors before the system is released.
- The system and code are documented.

Evaluation

After the system has been released and used for a period of time, it is reviewed to determine how well it meets its requirements.

Evaluation looks at

- Effectiveness
- Usability
- Maintainability
- Possible improvements to the system are identified.

Software Development Methods

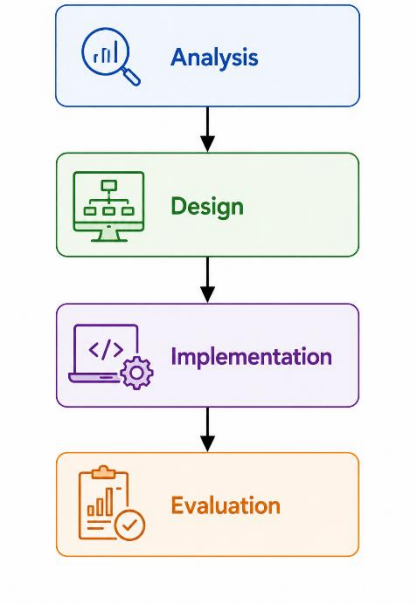
Waterfall

Each stage is completed in a fixed, linear sequence before the next stage begins.

Requirements are established in early stages, and subsequent stages focus on these.

The customer is involved at the start in the analysis phase but then is not involved until the final evaluation phase is reached.

As each stage needs to be finished before moving on it is important to get that stage right as changes to phases later can be costly.



Good for:

- Projects that are likely to have static requirements (not likely to change).
- Low-risk projects.
- Projects requiring little user input.

Pros:

- Straight forward to manage with a clear structure.
- Clearly documented.

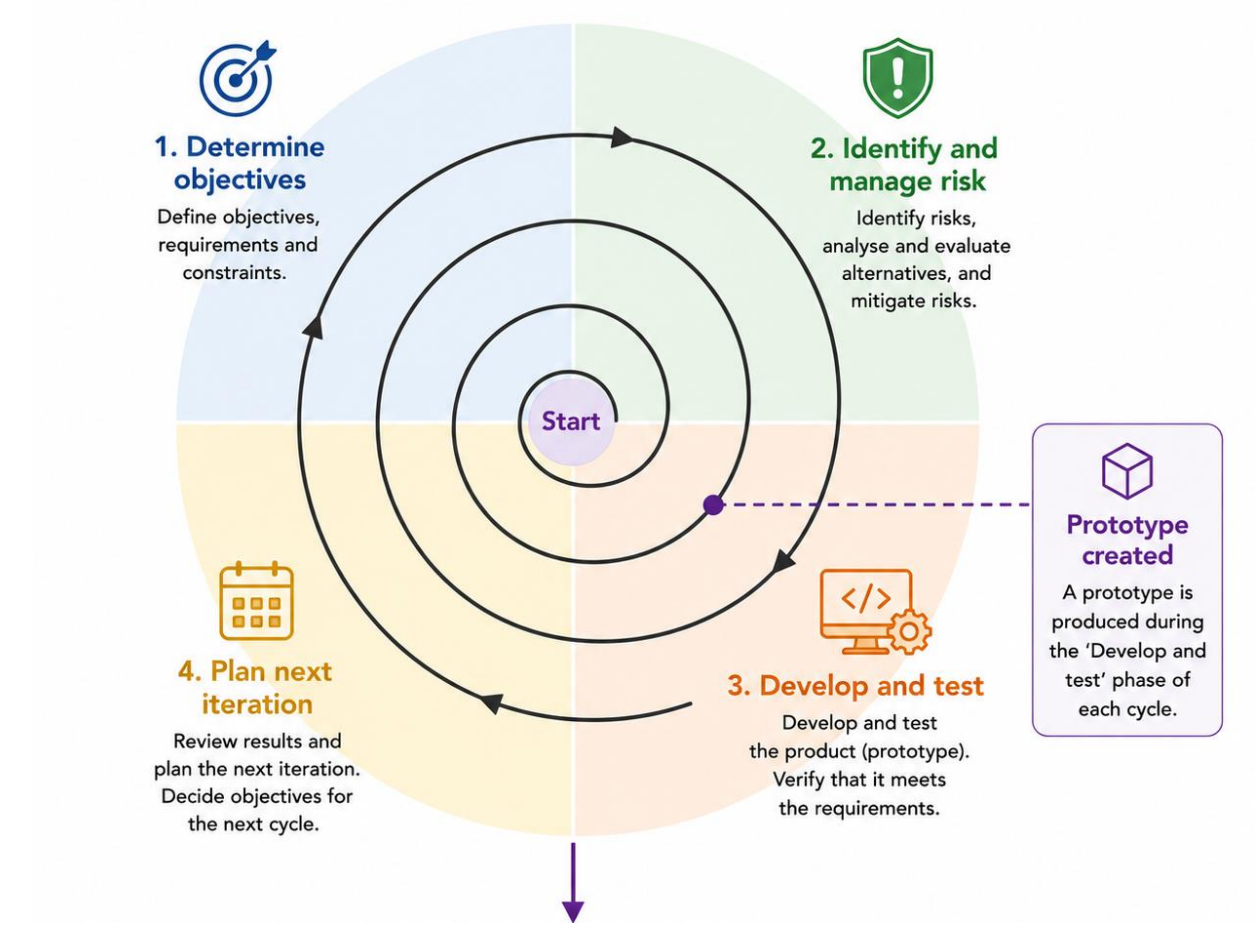
Cons:

- Inflexible and limits changing requirements.
- Excludes client/end user for most of the process.
- Testing is mainly carried out after the implementation stage, so problems may not be discovered until relatively late.

Spiral Model

The development stages are repeated in cycles, with a strong focus on identifying and managing risks.

The stages are followed to build an initial prototype and then, using what has been learned, the stages are repeated over and over, each time resulting in a refined prototype until a finished product is reached.



Good for:

- Large projects that take years to complete.
- Risk – intensive projects with large budgets.

Pros:

- Thorough risk-analysis and mitigation
- Caters to changing user needs
- Produces prototypes throughout

Cons:

- Development can be expensive and time-consuming because of repeated risk analysis and prototyping
- Lack of focus on code efficiency

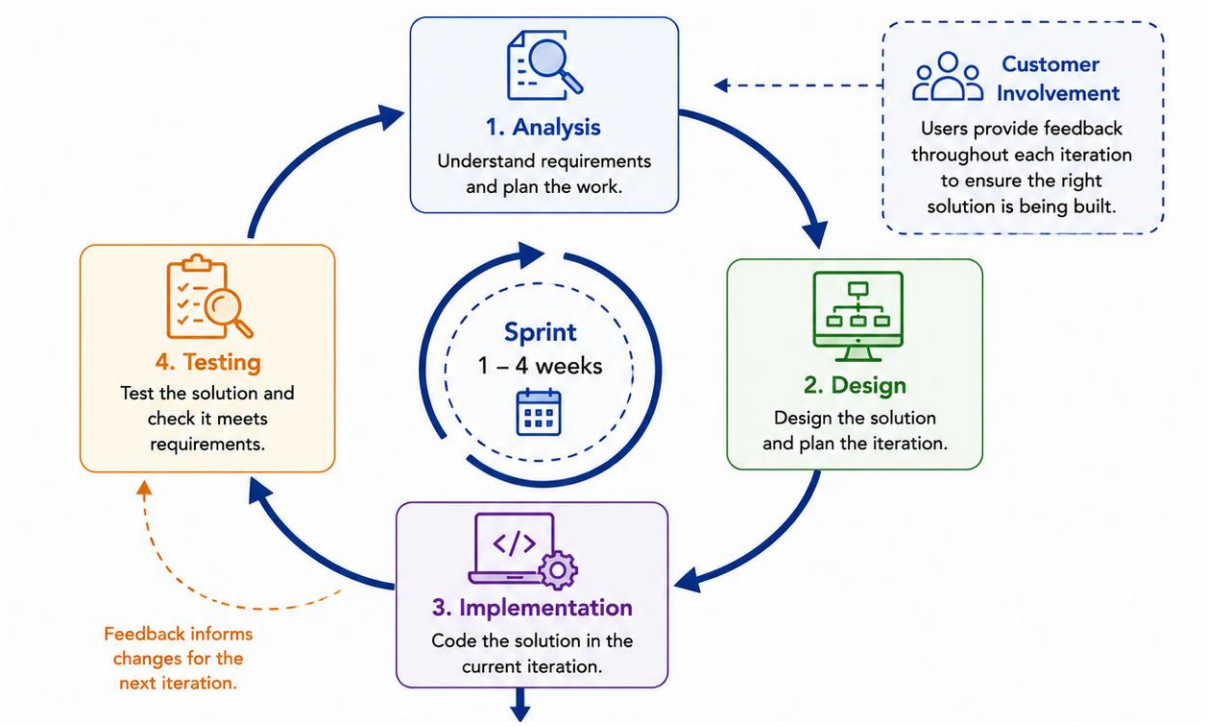
Agile Modelling

Agile modelling is an iterative approach in which development stages can be revisited in a non-linear sequence, with regular testing, feedback and improvements.

For example, parts of the project may finish analysis and be designed and implemented while other parts may still be being analysed.

Development takes place in short iterations called sprints, typically lasting 1–4 weeks, during which a new or improved prototype is produced.

Clients/users can provide feedback each iteration to ensure that requirements are meet and allow any modifications or alteration in the next iteration to be meet.



Good for:

- Small/medium sized projects
- Projects with unclear initial requirements

Pros:

- High quality code produced – less chance of bugs as tested throughout
- Flexible to changing requirements
- Regular user input

Cons:

- Documentation may become outdated as requirements and the system change.
- Requires consistent interaction between user and programmer.
- Room for scope creep – projects getting too big and out of hand.

Extreme Programming

Extreme Programming (XP) is an agile, iterative development methodology that focuses on producing high-quality code while supporting the development team's working practices.

Production cycles are kept short with frequent releases of software. This allows for quick response to changing customer requirements and improved software quality.

Key features:

- Focus on good quality code.
- Fast, agile paradigm.
- Designed to allow development to respond to changing user requirements.
- Program regularly reviewed.

Certain practises are met to mitigate risk and improve developers lives.

- Sustainable working practices are encouraged to reduce excessive working hours. Such as a maximum 40-hour work week.
- Pair programming involves two programmers working together on the same code, allowing code to be reviewed continuously.

Good for:

- Small/medium sized projects.
- Projects with unclear initial requirements.

Pros:

- High quality code produced – less chance of bugs as tested throughout.
- Flexible to changing requirements.
- Regular user input.
- Better quality of life for developers.

Cons:

- Poor documentation due to changing scope – documentation of modules may be rendered obsolete as changes are made so documentation not emphasized upon.
- Requires consistent interaction between user and programmer.
- Room for scope creep – projects getting to big and out of hand.
- Costs of two people working on one module is high.

Rapid Application Development (RAD)

Type of agile model which gives more importance to rapid prototyping and speedy feedback. The focus is on usability and producing a working system quickly.

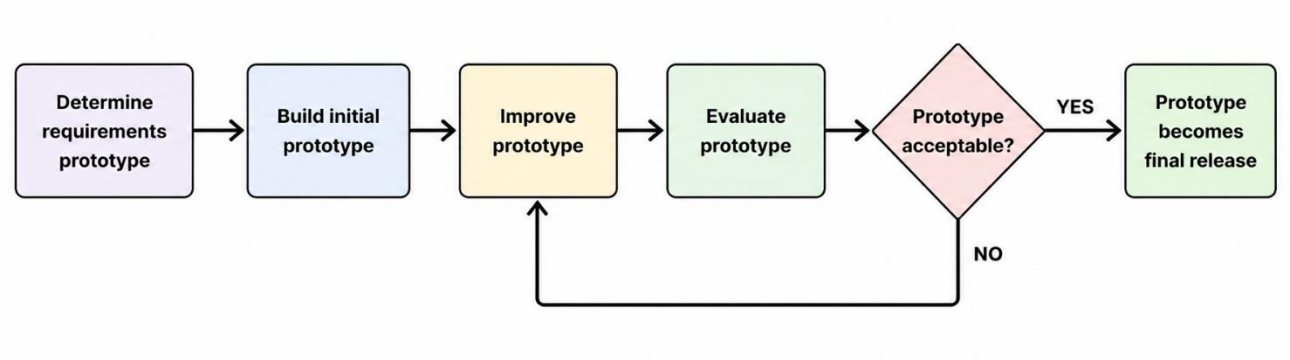
With very large projects that take a considerable length of time, there may be both technologic and user requirements changes with a need for major changes in later stages of development that can lead to high cost or cancellation.

To combat this, RAD aims to accelerate the completion of large projects by:

- Using workshops and focus groups to gather information on requirements of the systems.
- Using prototypes to refine the system in response to feedback.
- Developing components within fixed time limits, even if they are not initially complete or perfect.
- Reusing software components where able.

Process:

- Create a prototype
- Evaluate and get feedback
- Changes made to prototype
- Repeat until the prototype becomes the final product



Good for:

- Small/medium projects

Pros:

- Flexible to changing requirements
- Highly usable finished product
- Focus on core features reducing development time

Cons:

- Poor documentation due to changing scope – documentation of modules may be rendered obsolete as changes are made so documentation not emphasized upon
- Fast pace may reduce code quality

Methodology Comparison Table

Type	Pros	Cons	Uses
Waterfall	Straightforward to manage. Clearly documented.	Lack of flexibility. Limited risk analysis. Limited user involvement.	Static, low-risk project which need little user input, such as a piece of general-purpose software.
Spiral	Through risk-analysis and mitigation. Caters to changing user needs. Produces prototypes	Expensive to hire risk assessors. Lack of focus on code efficiency. High costs due to constant prototyping.	Large, risk-intensive projects with a high budget.
Agile	Produces high quality code. Flexible to changing requirements. Regular user input.	Poor documentation. Requires consistent interaction between user and programmer.	Small to medium projects with unclear initial requirements.
Extreme Programming	Produces high quality code. Constant user involvement means high usability.	High cost of two people working on one project. Teamwork is essential. Requires regular availability of users and developers.	Small to medium projects with unclear initial requirements requiring excellent usability.
Rapid Application Development	Caters to changing user requirements. Highly usable finished product. Focus on core features, reducing development time.	Poorer quality documentation. Faster pace may reduce code quality.	Small to medium, low-budget projects with short time-frames.

Testing Methods

Black box testing	Testing is carried out without examining the internal code. Inputs are provided and the outputs are checked against the expected results.
White box testing	Testing is based on the internal structure of the code, including the different paths that can be taken. However, it may not identify requirements that were never implemented
Alpha testing	Testing is carrying out by a testing team that works within the software development team. It can identify errors and omissions before the software is released to external users.
Beta testing	Potential users test the system before its final release and report problems to the developers. Testing with real users can reveal problems that the developers did not anticipate.