Higher Computing Science

Development Methodologies –Summary Notes

Software, information systems & web sites are all developed using a series of steps.

Analysis	The requirements of the software are identified, including • **Purpose**: a general description of the purpose of the software. • **Scope**: a list of the deliverables that the project will hand over to the client and/or end-user, e.g. design, completed program, test plan, test results and evaluation report. It can also include any time limits for the project. • **Boundaries**: the limits that help to define what is in the project and what is not. It can also clarify any assumptions made by the software developers regarding the client's requirements. • **Functional requirements**: the features and functions that must be delivered by the system in terms of inputs, processes and outputs. • **User requirements**: the tasks that the user should be able to perform At the analysis stage the software specification is produced, this is legal contract between the client and developers that includes the requirements listed above.
Design	**User interface** is designed using a wireframe. The user interface is the part of the program that the user interacts with. A wireframe is a sketch of the layout of the user interface that shows how the inputs and outputs will be gathered/displayed to the user. It also shows the layout of buttons and media elements such as graphics, video & audio. **Program Structure** is designed using pseudocode, a flowchart or structured diagram. The **top level design** is listed showing the main steps required to create the program, this also shows **data flow** throughout the program. **Data flow** is when the parameters being passed in and out of each subprogram are indicated on the design as IN or OUT. IN parameters are variables passed in to be used by the subprogram, OUT parameters are passed out from the subprogram to be used by other parts of the program. **Stepwise refinement** is then used to break steps from the top level design further until the code can be written.
Implementation	User interface and code are created.
Testing	Testing is carried out by both programmers and independent testers to ensure the requirements have been met. The testing is documented using test tables that use normal, extreme and exceptional data. Screenshots that demonstrate the code working and error messages are gathered at the testing stage.
Documentation	Main documentation is produced – • **User guide** – a guide which describe how to use the main features of the software, usually online. This is used by end users. • **Technical guide** – a document which describes any problems encountered and how they were resolved, installation instructions, hardware requirements, software requirements. This is used by developers.

Evaluation	Solution is compared against a set of criteria –
	• **Robustness** – how does the program cope with exceptional data?
	  ○ Input validation is a feature of robust code
	  ○ Even with input validation the code may still crash if the user inputs a string when a numeric value is expected
	• **Efficient** – does it use unnecessary ram or processor time?
	  ○ Use of fixed loops, conditional loops, arrays, nested if's, , functions and parameter passing are features of efficient code
	  ○ Efficient code uses the least lines of code required to complete a task
	  ○ Efficient code places minimum demands on the processor and RAM
	• **Usability** - how intuitive the software is from a user's perspective?
	  ○ How easily can the user operate the general user interface?
	  ○ How informative are the user prompts?
	  ○ Is the screen layout aesthetically pleasing?
	  ○ Are help screens useful?
	• **Maintainability** - how easy it is to alter the software?
	  ○ readability of the code — made easier by using meaningful variable names, comments, indentation and whitespace
	  ○ amount of modularity – using functions and procedures effectively

Maintenance	Further changes are made to the solution. • Corrective – fixing bugs not detected at testing • Perfective – adding new features • Adaptive – changing the solution to allow it to run on a new platform such as new operating system or different processor

Iterative Method (Waterfall)

This is the traditional method of developing software or information systems. The steps above are carried out in order until a final system is produced. It is a **predictive** method of developing software focusing on analysing and planning the future in detail and catering for known risks.

Iteration often occurs with this method. Iteration is when a previous stage is revisited due to new information or problems occurring.

Teamwork

Teams of analysts, programmers, testers and documenters work independently on each phase of development. Teams mainly work in isolation with some communication required between each phase.

Documentation

A detailed project specification is created at the beginning of a project. Significant time is spent during the project on design, program commentary and test plans.

Measurement of Progress

Follows a strict plan, with progress measured against timescales set at the beginning of the project.

Testing

Testing is carried out when the implementation phase of the project is complete.

Advantages

- Client receives a fully functional system at the end of the process
- Milestones allow progress to be easily tracked

Disadvantages

- Client's is not involved at implementation or testing stages which means that the final solution may not be what they were expecting
- Difficult to incorporate changes if the client's requirements change

Use

Large projects are still often developed using this methodology. Systems for banks, air traffic control or medical systems will be developed using this method as they require all features to be working.

Agile Development

Client is involved throughout the development process. A solution is created with the key features functioning, feedback is then gathered from users to improve the solution. New features and bugs are fixed in each update released. This is an adaptive methodology, focusing on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well.

Teamwork

Teams of developers communicate and collaborate, rather than teams of experts operating in isolation. During a project, fast, face-to-face communication between individuals with different skills is an important factor in progressing the project quickly.

Documentation

While modelling solutions remains important, creating large documents that are never updated or referred to again upon completion of the project are not. Agile focuses on reducing documentation. It spends time on small cycles of coding, testing and adapting to change. Any documentation produced (for example internal commentary in code) should focus purely on progressing the project.

Measurement of Progress

Breaks a project down into a series of short development goals (often called "sprints"). This involves cross-functional teams working on: planning, analysis, design, coding, unit testing, and acceptance testing. Progress is measured by the time it takes to produce prototypes or working components of the software. Agile focuses on delivering software as quickly as possible.

Testing

There is no recognised testing phase, as testing is carried out in conjunction with programming.

Advantages

- Working software is produced very quickly
- Suitable for projects where the requirements may change

Disadvantages

- Client must be prepared to commit time to this method
- Final systems can be badly designed due to new features being added that were not considered initially

Use

Mobile apps & websites are often developed using this methodology

Prototypes

A prototype is model that represents some features of the final solution. Prototypes are used in agile projects to allow the client to have a feel for how the final solution will look and feel. Clients provide feedback, this feedback is then used in the next stage of development.