The software development process

Most software development projects use one of two development methodologies: the 'waterfall' method or the 'agile' method, (and occasionally a combination of the two).

Waterfall Method

Overview

The 'waterfall' method is the traditional ''linear'' approach to software development. The client knows broadly what it wants from the software at the outset, and the development process flows on from there in the stages below (supposedly like a waterfall):

1. Planning and formulating requirements for the software, (see our note managing the procurement process);
2. Design
3. Coding
4. Testing
5. Integration
6. Deployment
7. Support and maintenance

Appropriate for: projects which have fixed/guaranteed requirements from the outset or involve complex integration of multiple components and/or with external systems.

Benefits

- certainty of the requirements from the outset means it should be easier to plan for;
- delivering the ''whole product'' in one go means it should all work together;
- easier to measure progress and success against at the end of the process; and
- less client time/input required.

Disadvantages

- less flexibility in the process; making changes after the initial planning stage may be hard;
- the client may not know exactly what they want or need at the outset of the project;
- problems may only become apparent at the implementation stage, at which point resolving them may be a much bigger undertaking – being more difficult, costly and cause substantial delays; and

- less client involvement increases the risk that the client will not be happy with the end product.

Agile Method

Overview

Agile methodology focusses on the rapid delivery of software by using an ''evolutionary'', phased approach. Over the years, different agile methodologies have evolved including: Scrum, Kanban, Lean and Dynamic System Development Model.

Although each agile methodology has its own structure and terminology, the typical process involves:

- development happening in phases called 'sprints', which each has a defined duration and a specific set of deliverables;
- each sprint should produce a stand-alone piece of code;
- in depth and continued client involvement; the project team and the client work together in cross-functional teams which are usually located in the same physical space, to promote communication and short feedback loops.

Appropriate for: where the client is not sure of the finished product they want or need at the outset or in ''phased'' developments, where a basic software product is created first, then further functionality subsequently added to address feedback from end-users.

Advantages

- increased client involvement in design and decision-making should lead to more client buy-in and in turn to greater client satisfaction;
- end-product should be more user-focused;
- speed of development, if managed correctly;
- each phase should produce a discrete element and so evidence progress earlier in, and continuing throughout, the project; and
- flexibility which allows for additional functionality to be added and priorities changed during development process.

Disadvantages

- clients can find project costs far exceeding their initial budgets, fixing overall project budgets and timescales can be difficult;
- generally works better for tech-savvy clients due to the requirements for their involvement;
- increased client involvement can lead to additional features being requested throughout the project which may be costly and time consuming;
- sometimes the deliverables are not completed within the phase timeframe, requiring additional sprints and increasing delays and project cost; and
- as focus is on producing working product quickly, documentation is often neglected.

Need further information?

Read:

- Our overview of the ''waterfall'' software development process and the key legal and commercial issues for a software development agreement
- Our guide to 'software development projects – The client perspective'.