

Software Development Methodologies

By Jagvinder Kang, Director, Technology Law Alliance

Many IT lawyers draft software development agreements for clients without any true understanding of the underlying methodologies which apply to software design and development. Consequently, it is questionable how closely the contracts which they draft reflect their true clients' requirements.

This article therefore outlines some of the key software development paradigms, as well as their main characteristics.

Software Development

It is possible to design and develop software using various methodologies and tools, and depending upon the specific approach which is adopted, will determine:

- What is produced;
- Time for production;
- Resource requirements;
- Cost of production;
- Reliability;
- Ease of maintainability; and
- Extent of fulfilment of customer requirements.

As one may imagine, early software development was undertaken in an ad hoc manner, thus the extent to which the final software matched the customer's requirements varied considerably. It was as a result of this,

that consideration was given to adopting methodologies for software engineering and development, with some of the key models being:

- Waterfall Model/Classical Model;
- Prototyping; and
- Spiral Model.

Waterfall Model

The Waterfall Model consists of various discrete stages, each of which must be completed, before moving on to the next. These stages consist of:

- Identification of system requirements;
- Identification of software requirements;
- Analysis phase;
- Program design;
- Coding phase;
- Testing; and
- Live operation and maintenance.

The Waterfall Model brings the benefit of structure to software engineering and development, but it does suffer from certain shortcomings.

The sequential flow of the Waterfall Model is an ideal rather than an actuality. During software design and development, some form of iteration is almost always a requirement. This of course does not fit in with the strict Waterfall Model.

The Waterfall Model also requires all requirements to be initially specified, something which customers are usually unable to state upfront prior to contract signature.

Furthermore, by adhering to the requirement to complete each stage before moving on to the next stage, also has the disadvantage that the customer is only able to experience the final output at the conclusion of the process. This again increases the risk of mismatch between customer expectations and actual software which is designed and produced. Furthermore, in complex or time intensive projects, there is no mechanism in the Waterfall Model to change the requirements specification to reflect the evolution of the customer's business operations over the project's life.

Prototyping

This process permits certain aspects of a project to be explored in order to factor the findings into the design and development process. This can be advantageous when considering certain interfacing elements of a software project, or for evaluating the efficiency and performance characteristics of a proposed design.

Unlike the Waterfall Model, prototyping adopts an inherent iterative process. The key stages of the model, involve:

- Gathering requirements;
- A limited design process;
- Production of prototype;
- Evaluation of the prototype with customer input;
- Refinement of prototype;

- Production of final software on basis of refinement.

The iteration exists between each stage, and it is possible to move back one or more stages if the findings at any stage suggest that this would be prudent. This is again something which is lacking in the Waterfall Model.

Thus prototyping clearly offers advantages through its iterative nature, which helps with resolving the mismatch between customer requirements and the end product. However, prototyping also suffers for disadvantages, which include:

- The degree of resource initially expended in the prototype, may take the focus off certain other requirements for the complete software project, and thus compromise the quality and reliability of the final output;
- Producing something which to a customer seems to work, may from a: time; cost; and/or customer pressure; perspective, require re-use of the prototype 'as it is' within the software project, without the customer realising that a comprehensive evaluation of the design has not been undertaken, but merely what exists is a representation which allowed the customer to view a certain output. This in itself may result in the future maintainability of the software being adversely affected.

Spiral Model

This model incorporates the benefits of structure offered by the Waterfall Model, together with the advantages of iteration offered by prototyping.

The model commences with consideration of: objectives (which could for example, include performance, functionality and adaptability); alternatives (which would include consideration of alternative designs and the potential for re-use of components); and constraints.

Once this stage has been completed, it then necessary for an evaluation of alternative options, with a focus on identifying and resolving risks. For example, a lack of clarity of specific requirements, may suggest that it is prudent to engage in a limited prototyping exercise to identify and refine such requirements.

The next phase is an engineering phase, which has the inherent flexibility to utilise certain of the stages of the Waterfall Model together with prototyping as required.

Once the engineering phase is completed, there is a joint evaluation of the software with the customer. This evaluation then feeds into the next iteration of the whole process described above, ie there is a continuous 'spiral' of activities, with each iteration or cycle producing a more complete software system.

This model therefore most closely reflects the way in which large scale or high value software developments are carried out.

Conclusion

Each of the methodologies described above has its own distinct characteristics, which may make each model more appropriate than another, for a certain value, size and type of software engineering and development project. It is therefore important when drafting a software development contract, that such features of the methodologies are reflected within the drafting – as after all, software engineering and development methodologies have evolved so that the end result more closely matches the customer's requirements, so it goes without saying that software development contracts also need to match this evolution.

For further information please contact:

Jagvinder Kang

T: 0870 730 5551

E: jagvinder.kang@TLawA.co.uk

Disclaimer:

This briefing is not legal advice, and should not be relied upon as such. Please contact us if you require specific legal advice on the issues discussed.