1997, Paper 2, Question 8, Answer

Matthew Johnson

February 22, 2002

1 Describe The Waterfall Method of Software Development

1.1 The Waterfall Method

The waterfall method consists of a linear sequence of steps, from the original requirements to the finished product and maintainance thereof. Each step is a well defined unit, and the breaks between them make convenient milestones, often contractually specified.

- *Requirements* - Done by 2 different groups of people who speack different languages, from different disciplines. This is written in the user's language, in close cooperation with the client.
- *Specification* - Done by a groups of single discipline professionals who can usually communicate. Written in some sort of explicit system language, gives a complete specification of the system.
- *Implementation & unit testing* - Here you check back all the units to the individual unit specification made in the previous step.
- *Integration* - This is where all the units are integrated together and you check that the complete product works as was specified in the requirements
- *Installation* - Done by people who don't undrestand the issues or problems

- *Operation* - left to people with no guarenteed understanding
- *Maintainance* - Often the original implementers have left, and possibly done by relatively inexperienced people.

1.2 Feedback

None of the steps in the model will be trouble free, so there must be feedback, and verification and validation. Validation checks that we are building the system that we are meant to be. Verification make sure what we write, does work. The waterfall method doesn't allow feedback for validation, which can leave the possibility that the wrong spec gets implemented. The waterfall method does allow for easier management, and deadlines. It also means that you can more easily charge a cient for specification changes. It is ideal where appropriate - the condition is how easily the requirements can be defined beforehand.

1.3 Pros & Cons of Waterfall Model

The waterfall method makes it a lot easier to give the project discrete milestones, which you can work towards, and can give you a more structured development. It also enables you to more easily charge for changes to the speification after it has initially been agreed, because you have had a (often contractual) milestone where you finalised the specification. There are a lot of design tools that work well with the waterfall method, and since you can clarify the goals and interfaces together it is condusive to a good programming style.

The waterfall model is ideal when it is appropriate. The critical factor that makes it appropriate is whether or not you can define the requirements to a sufficient level of detail in advance. There are some systems this works well for, but others, like HCI development, the set of requirements is not well defined in advance of proto-typing.

There are many changing factors that make the waterfall model inappropriate. In the time between specification and release a lot of things may have changed and the waterfall model allows for very little change in the specifications during the project. It is often important in software development to persue a more 'iterative' method, where you can constantly revise and improve the specifications & requirements, particularly in critical software and packaged software

2 When is the waterfall model suitable?

an incremental Java compiler	suitable	This can easily have the specifications completely drawn up in advance, and so would work well with the waterfall method.
a clinical record keeping system for dentists	suitable	The specifications of this might change, but over a long period of time. The waterfall model could be used, and the system re-implemented if the requirements change
a word processing package	not suitable	Packaged software is more suited to a model of development where the product can be shown to users continually during the development, and the program updated as appropriate
a guidence system for an interplanetary probe	suitable	Projects like this generally are very well specified in advance, since nothing can be changed during the deployment of the device. Hence there is a good argument for this type of development.