**Waterfall Method**

Introduce the Topic

* a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance
* originated in the manufacturing and construction industries: highly structured physical environments in which after the fact changes are prohibitively costly, if not impossible
* Winston W. Royce wrote an article that first formally describe the waterfall method
  + he did not use the term waterfall in the article
  + presented this model as an example of a flawed, non-working model
  + Royce was more of the opinion that a successful model should have the allowance of repetition or to go back and forth between phases which the waterfall model doesn’t do
* illustrates the software development process in a linear sequential flow - referred to as a linear-sequential life cycle model
  + any phase in the development process begins only if the previous phase is complete
  + phases don’t overlap
  + one phase acts as the input for the next phase sequentially

Present details about it (including code and non code based examples)

* Sequential Phases
  + **Requirement Gathering and Analysis**: all possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc
    - research is being conducted which includes brainstorming about the software, what it is going to be and what purpose is it going to fulfill
  + **System Design**: the requirement specifications from the first phase are studied in this phase and system design is prepared
    - System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture
    - functions of each of the part are decided and the engineering units are placed for example modules, programs, etc.
  + **Implementation**: with inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase
    - each unit is developed and tested for its functionality which is referred to as Unit Testing
    - the source code of the program is written
  + **Integration and Testing**: all the units developed in the implementation phase are integrated into a system after testing of each unit
    - Post integration the entire system is tested for any faults and failures
    - whole design and its construction is put under a test to check its functionality
      * any errors will surface at this point of the process
  + **Deployment of System**: once the functional and nonfunctional testing is done, the product is deployed in the customer environment or released into the market
  + **Maintenance**: there are some issues which come up in the client environment
    - to fix those issues patches are released
    - to enhance the product some better version are released
    - maintenance is done to deliver these changes in the customer environment
* all these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases
* the next phase is started only after the defined set of goals are achieved for previous phase and it is signed off
* phases don’t overlap
* Situations where the model is most appropriate
  + requirements are very well documented, clear and fixed
  + product definition is stable
  + technology is understood and is not dynamic
  + there are no ambiguous requirements
  + ample resources with required expertise are available to support the product
  + the project is short
* Pros
  + it allows for departmentalization and control
  + a schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one
  + development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at the operation and maintenance
  + each phase of development proceeds in strict order
* Cons
  + doesn’t allow for much reflection or revision
  + once the application is in the testing stage, it is very difficult to go back and change something that was not well documented or thought upon the concept stage
* More Pros and Cons (list)

|  |  |
| --- | --- |
| Pros | Cons |
| simple and easy to understand and use | no working software is produced until late during the life cycle |
| easy to manage due to the rigidity of the model  each phase has specific deliverables and review process | high amounts of risk and uncertainty |
| phases are processed and completed one at a time | not a good model for complex and object oriented projects |
| works well for smaller projects where requirements are very well understood | poor model for long ongoing projects |
| clearly defined stages | not suitable for the projects where requirements are at a moderate to high risk of changing  so risk  and uncertainty is high with this process model |
| well understood milestone | it is difficult to measure progress within stages |
| easy to arrange tasks | can’t accommodate changing requirements |
| process and results are well documented | no working software is produced until late in the lift cycle |
|  | adjusting scope during the life cycle can end a project |
|  | integration is done as a big bang at the very end which doesn’t allow identifying any technological or business bottleneck or challenges early |

Provide Pointers to additional material on the topic for interested readers

* Dr. Winston Royce original paper on Waterfall
  + <http://agileconsortium.pbworks.com/w/file/fetch/52184636/waterfall.pdf>

Why should software developers care about this topic?

* Offers numerous advantages for software developers
  + the staged development cycle enforces discipline
    - every phase has a defined start and end point and progress can be conclusively identified by both vendor and client
  + emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage or of customer expectations not being met
  + getting the requirements and design out of the way first also improves quality
    - much easier to catch and correct possible flaws at the design stage than at the testing stage after all the components have been integrated and tracking down specific errors is more complex
  + can aid efficient knowledge transfer when team members are dispersed in different locations
* project management team spends a lot of time before the project begins to try to understand, document and plan everything that needs to be included
  + client meetings, interviews, surveys and all types of information gathering goes into the front end so sketches or mocks up of workflow can be produced and approved by the customer

Other information

* Modified Waterfall Models
  + Royce Model
    - original definition, the model consisted of the following steps: Requirements specification, Design, Construction (coding), integration, testing and debugging, installation, and maintenance
  + Sashimi Model
    - phases are somewhat the same as in the Waterfall Model; only this time the phases are overlapping each other which present many advantages
      * one can return to the previous step if desired
      * Ex: the time won’t be wasted because before Phase 1 would be completed, Phase 2 would already be underway
  + Aorta Lifecycle Model
    - rely a lot on the feedback which comes from other phases before progressing onto the next
  + V Waterfall Model
    - relies on a linear software developmental program which stresses  on balanced development more than anything else

References

* <https://en.wikipedia.org/wiki/Waterfall_model>
* <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>
* <http://www.waterfall-model.com>
* <http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/>
* <http://cabforward.com/waterfallmethodofsoftwaredevelopment/>