**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 (1970)
  + 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
* Why still used
  + Science often works
    - Look at the scientific articles on software engineering that discuss the waterfall
      * All cite Royce’s article
      * Are saying something like “The waterfall is a proven method (Royce 1970)”
    - Based their claims on an article that actually says the opposite
    - Researchers just cite something because everyone else does so as well and don’t really read the publications that they refer to
    - Eventually an often cited claim becomes a fact
  + The waterfall model provides a structured approach; the model itself progresses linearly through discrete, easily understandable and explainable phases and thus is easy to understand
    - Also provides easily identifiable milestones in the development process.
  + Straightforward to Measure
  + Possible to move between stages when the need occurs
  + Experience applying steps in past projects can be used in estimating duration of steps in future projects
  + Produces software artifacts that can be re-used in other projects
* Cons
  + This made the process inflexible and monolithic
  + Making estimates about how long the process would take was difficult
  + Did not deal well with changing requirements
  + Maintenance phase not handled well

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
    - Gathering information about what the customer needs and defining, in the clearest possible terms, the problem that the product is expected to solve
    - Techniques used to obtain this understanding include customer interviews, use cases, and "shopping lists" of software features.
    - The results of the analysis are typically captured in a formal requirements specification, which serves as input to the next step.
  + **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
      * Defining the hardware and software architecture, specifying performance and security parameters, designing data storage containers and constraints, choosing the IDE and programming language, and indicating strategies to deal with issues such as exception handling, resource management and interface connectivity
    - Functions of each of the part are decided and the engineering units are placed for example modules, programs, etc.
    - User interface design is addressed, including issues relating to navigation and accessibility.
    - The output of this stage is one or more design specifications, which are used in the next stage of implementation.
  + **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
    - The output of this step is one or more product components, built according to a pre-defined coding standard and debugged, tested and integrated to satisfy the system architecture requirements.
  + **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
    - Three types of testing are done:
      * Unit testing of individual code modules;
      * System testing of the integrated product;
      * Acceptance testing, formally conducted by or on behalf of the customer
    - Defects, if found, are logged and feedback provided to the implementation team to enable correction.
    - User manual and product documentation are created
  + **Deployment of System**: once the functional and nonfunctional testing is done, the product is deployed in the customer environment or released into the market
    - Delivery may take place via the Internet or physical media, and the deliverable is typically tagged with a formal revision number to facilitate updates at a later date.
  + **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>

Example of Waterfall Method

* Product Development – developing a customer address book
  + Product Requirements
    - Product manager creates requirements documents that include the following requirements (in order of priority)
      * User should be able to create new contacts
      * User should be able to view their contacts
      * User should be able to import contacts from other programs
      * User should be able to email their contacts from the address book
      * User should be able to add pictures to represent their contacts
    - These requirement documents will include details requirements, user scenarios and potential layouts for the functionality
    - Timeframe: 2 weeks
  + Analysis
    - Engineering team takes these requirements and analyzes them, asking questions as needed. Product manager updates documents as questions are resolved.
    - Timeframe: 1 week
  + Design
    - Engineering team creates a design for functionality, including database design, mock-ups and workflows
    - Timeframe: 3 weeks
  + Implementation
    - Engineering team develops functionality and prepares it for testing
    - Timeframe: 1 week
  + Software Product Testing
    - Product team tests entire functionality
    - Timeframe: 2 weeks
  + Release
    - The product functionality is released
  + Note: if any changes to the design occur during this workflow, the project would have to return to the second or third phase and restart the process

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
* models are important to software engineering
  + a program is a model
  + even at the assembly level, programming is constructing a model
* modeling is the way humans deal with complexity
* modeling helps us reduce and understand
* domain capturing is perhaps one of the hardest part of the engineering process and at its heart it is domain understanding
  + for that you need models
* software engineering is about models

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
    - Peter DeGrace
    - Named for the sushi model that has overlapping layers of fish
    - 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

<http://www.thomasalspaugh.org/pub/fnd/img/softwareProcess-sashimi.png>

<http://www.thomasalspaugh.org/pub/fnd/img/softwareProcess-waterfallFeedback.png>

* Originated by Peter DeGrace
* Called Sashimi because it features overlapping phases, like the overlapping fish of Japanese sashimi
* Sometimes referred to as the “waterfall model with overlapping phases” or “the waterfall model with feedback”
* Since phases overlap, information of problem spots can be acted upon during phases of the waterfall model that would typically “precede” others in the pure waterfall model
* Ex: design and implementation phase overlap – implementation problems may be discovered during the “design and implementation” phases of the development process
  + Helps alleviate many of the problems associated with the Big Design Up Front philosophy of the waterfall model
* <http://iamjeab9.blogspot.com/2007/12/sashimi-model.html>
* Similar to the waterfall, except that the phases overlap to show that requirements can’t be completed until architecture is at least partially explored
* Architecture can’t be completed until module design is at least partially explored and so on
* Most appropriate for medium-sized projects for which the communication between phases can be handled in an improvised manner
* <http://www.thomasalspaugh.org/pub/fnd/softwareProcess.html>
* Key feature is the possibility of overlapping development phases
  + Introducing feedback into the classical waterfall model
* Idea – identification of errors made on time, while the development phase is still in progress
* Ex: errors made in the design phase are identified during implementation, while design is still in progress
* Another important feature – different treatment of documentation
  + Documentation in the classical waterfall model is exchanged by teams in charge of completing individual phases
  + Sashimi model treats the documentation as unified document
    - This results in a significant reduction in the documentation volume
* <http://www.iim.ftn.uns.ac.rs/casopis/volume1/ijiem_vol1_no4_4.pdf>
  + Aorta Lifecycle Model
    - Relies a lot on the feedback which comes from other phases before progressing onto the next

https://complextester.files.wordpress.com/2012/08/aorta1.png

* Development method according to the waterfall method, but after each cycle gets back to the customer location
* Includes 9 steps
  + Orientation phase
    - Globally mapped to the wishes of the user of application is
  + Planning
    - Project plan is drawn up for the rest of the project
  + Concrete work comprising of 5 essential phases
    - Analyse
    - Design
    - Realisation
    - Test
    - Acceptation
    - Note: Aorta
  + Evaluation
    - Looks back on the project to bring the experience card
  + Maintenance
    - Maintenance on the delivered application
* <https://nl.wikipedia.org/wiki/Aorta_lifecycle-model>
  + V Waterfall Model
    - Relies on a linear software developmental program which stresses on balanced development more than anything else

<http://1.bp.blogspot.com/-puUJFe5lFHk/T4iml5pOc9I/AAAAAAAACb8/yKMKNb_ff8g/s1600/V-Shaped.png>

https://melsatar.files.wordpress.com/2012/03/vmodel.jpg

* Means Verification and Validation model
* Just like waterfall, life cycle is a sequential path of execution of processes
* Each phase must be completed before the next phase begins
* Testing of the product is planned in parallel with a corresponding phase of development
* Requirements – before development is started, a system test plan is created
  + Test plan focuses on meeting the functionality specified in the requirements gathering
* High level design – focuses on system architecture and design
  + Provide overview of solution, platform, system, product and service/process
  + Integration test plan is created as well in order to test the pieces of the software system ability to work together
* Low level design – the actual software components are designed
  + Defines the actual logic for each and every component of the system
  + Class diagram with all the methods and relation between classes comes under this phase
  + Components test are created as well
* Implementation – all coding takes place
  + Once coding is complete, the path of execution continues up the right side of the V where test plans developed earlier are now put to use
* Coding – module design is converted into code by developers
* Advantages
  + Testing activities like planning, test designing happens well before coding
    - Saves a lot of time
    - High chance of success over the waterfall model
  + Proactive defect tracking – defects are found at early stage
  + Avoids the downward flow of the defects
  + Works well for small projects where requirements are easily understood
* Disadvantages
  + Very rigid and least flexible
  + Software is developed during the implementation phase
    - No early prototypes of the software are produced
  + Any changes happen midway, the test documents along with requirement documents has to be updated
* <http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>

Agile vs Waterfall

* There are several ways to develop software and two of the most prominent methods are waterfall and Agile
* Debate rages about which is better
  + Does it matter really
  + Doesn’t either way give you a product
* Both have their strength and weakness
* Key to deciding which is right for you comes down to the context of the project
* Is it going to be changing rapidly?
  + Yes – choose Agile
* Do you know exactly what you need?
  + Yes – choose waterfall
* <http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/>

|  |  |
| --- | --- |
| Agile | Waterfall |
| Agile | Waterfall |
| Flexible | Structured |
| Many small projects | One big project |
| Highly collaborative | A sequential process |
| Best for those who want continuous improvements | Suited for situations where change is uncommon |
| Involved customers | Internal |
| A process in which requirements are expected to evolve and change | A process that requires clearly defined requirements upfront |
| Best for projects that deal with services-oriented and non-physical deliverables like code, copywriting and design projects | Best for projects that deal with physical objects – from a construction project to a hardware installation project |
| Allows for quick course correction based on stakeholder feedback | Best for projects with defined tasks and phases that must be completed in a specific sequence |
| Empowers project teams to work creatively and efficiency | Projects plans are repeatable for identical or similar projects in the future |
| Includes engagement and collaboration from all team members |  |

* <http://www.onedesk.com/2013/01/waterfall-vs-agile/>
* http://www.liquidplanner.com/blog/agile-v-waterfall-which-project-management-style-is-right-for-you/

References

* <https://en.wikipedia.org/wiki/Waterfall_model>
* <http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm>
* <http://www.waterfall-model.com> - Account Suspended
* <http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/>
* <http://cabforward.com/waterfallmethodofsoftwaredevelopment/>
* <http://www.marsdd.com/mars-library/product-development-the-waterfall-methodology-model-in-software-development/>
* <http://tarmo.fi/blog/2005/09/dont-draw-diagrams-of-wrong-practices-or-why-people-still-believe-in-the-waterfall-model/>
* http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/
* <http://www.slideshare.net/BHARGAV_VISANI/waterfall-model>

MLA Format

* "Waterfall Model." *Wikipedia*. Wikimedia Foundation. Web. 24 Sept. 2015. <https://en.wikipedia.org/wiki/Waterfall\_model>.
* "SDLC - Waterfall Model." *Www.tutorialspoint.com*. Web. 24 Sept. 2015. <http://www.tutorialspoint.com/sdlc/sdlc\_waterfall\_model.htm>.
* "Reflections on the Waterfall Method of Software Development." *CabForward*. 5 June 2014. Web. 24 Sept. 2015. <http://cabforward.com/waterfallmethodofsoftwaredevelopment/>.
* Melonfire, Contributor. "Understanding the Pros and Cons of the Waterfall Model of Software Development." *TechRepublic*. Web. 24 Sept. 2015. <http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/>.
* "Product Development: The Waterfall Methodology (model) in Software Development." *MaRS*. 6 Dec. 2009. Web. 24 Sept. 2015. <http://www.marsdd.com/mars-library/product-development-the-waterfall-methodology-model-in-software-development/>.
* Tikkanen, Tarmo. "Don't Draw Diagrams of Wrong Practices – Or: Why People Still Believe in the Waterfall Model." *Tarmofi*. 9 Sept. 2005. Web. 24 Sept. 2015. <http://tarmo.fi/blog/2005/09/dont-draw-diagrams-of-wrong-practices-or-why-people-still-believe-in-the-waterfall-model/>.
* Melonfire, Contributor. "Understanding the Pros and Cons of the Waterfall Model of Software Development." *TechRepublic*. 22 Sept. 2006. Web. 24 Sept. 2015. <http://www.techrepublic.com/article/understanding-the-pros-and-cons-of-the-waterfall-model-of-software-development/>.
* "The "sashimi" Model." *: The "sashimi" Model*. Web. 25 Sept. 2015. <http://iamjeab9.blogspot.com/2007/12/sashimi-model.html>.
* "Aorta Lifecycle-model." *- Wikipedia*. 27 May 2015. Web. 25 Sept. 2015. <https://nl.wikipedia.org/wiki/Aorta\_lifecycle-model>.
* "What Is V-model- Advantages, Disadvantages and When to Use It?" *ISTQB Exam Certification*. Web. 25 Sept. 2015. <http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/>.
* Sussex, Tatyana. "Infographic: Agile vs. Waterfall–Which Project Management Style Is Right for You?" *LiquidPlanner*. 2 Oct. 2013. Web. 28 Sept. 2015. <http://www.liquidplanner.com/blog/agile-v-waterfall-which-project-management-style-is-right-for-you/>.
* "Agile & Waterfall Methodologies – A Side-By-Side Comparison." *Base36*. Web. 28 Sept. 2015. <http://www.base36.com/2012/12/agile-waterfall-methodologies-a-side-by-side-comparison/>.
* Chan, Kimberley. "Waterfall vs. Agile." *Waterfall vs. Agile*. 31 Jan. 2013. Web. 28 Sept. 2015. <http://www.onedesk.com/2013/01/waterfall-vs-agile/>.