Software Development Models

Classical Waterfall Model

Merits

- Economic in later stages of the project due to greater time spent in earlier stages.
- Problems encountered in each stage is easier and cheaper to resolve.
- Major decisions related to product design and requirements are taken at the earlier stages which reduces chances of any flaw discovery at later phase.
- Every phase is 100% complete before transition to next phase.
- This approach is simple to follow and is more disciplined.
- Projects with unchanging requirements are more appropriate for this model.

Demerits

- According to agile methodologies, it is believed to be impossible to have each phase completed 100% before transitioning to next phase.
- Clients may not know requirements completely in earlier stages.
- Designers may not be aware of future implementation difficulties when writing a design for an unimplemented software product.
- The idea behind the waterfall model may be "measure twice; cut once," and those opposed to the waterfall model argue that this idea tends to fall apart when the problem constantly changes due to requirement modifications and new realizations about the problem itself.

Iterative Waterfall Model

Merits

- Waterfall model is simple to implement and also the amount of resources required for it are minimal.
- In this model, output is generated after each stage (as seen before), therefore it has high visibility. The client and project manager gets a feel that there is considerable progress. Here it is important to note that in any project psychological factors also play an important role.
- Project management, both at internal level and client's level, is easy again because of visible outputs after each phase. Deadlines can be set for the completion of each phase and evaluation can be done from time to time, to check if project is going as per milestones.
- This methodology is significantly better than the haphazard approach to develop software. It provides a template into which methods of analysis, design, coding, testing and maintenance can be placed.
- This methodology is preferred in projects where quality is more important as compared to schedule or cost.

Demerits

- Real projects rarely follow the sequential flow and iterations in this model are handled indirectly. These changes can cause confusion as the project proceeds.
- It is often difficult to get customer requirements explicitly. Thus specifications can't be freezed. If that case arises baseline approach is followed, wherein output of one phase is carried forward to next phase. For example, even if SRS is not well defined and requirements can't be freezed, still design starts. Now if any changes are made in SRS then formal procedure is followed to put those changes in baseline document.
- In this model we freeze software and hardware. But as technology changes at a rapid pace,such freezing is not advisable especially in long-term projects.
- This method is especially bad in case client is not IT-literate as getting specifications from such a person is tough.
- Even a small change in any previous stage can cause big problem for subsequent phases as all phases are dependent on each-other.
- Going back a phase or two can be a costly affair.

Prototyping Model

Merits

- Reduced cost: Because changes cost exponentially more to implement as they are detected later in development, the early determination of what the user really wants can result in faster and less expensive software.
- This type of approach of developing the software is used for non-IT-literate people. They usually are not good at specifying their requirements, nor can tell properly about what they expect from the software.
- When client is not confident about the developer's capabilities, he asks for a small prototype to be built. Based on this model, he judges capabilities of developer.
- Sometimes it helps to demonstrate the concept to prospective investors to get funding for project.
- It reduces risk of failure.
- Iteration between development team and client provides a very good and conductive environment during project.
- Time required to complete the project after getting final the SRS reduces, since the developer has a better idea about how he should approach the project.
- Improved and increased user involvement: Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications.
- When prototype is shown to the user, he gets a proper clarity and 'feel' of the functionality of the software and he can suggest changes and modifications.

Demerits

- Prototyping is usually done at the cost of the developer. So it should be done using minimal resources. It can be done using Rapid Application Development (RAD) tools. Please note sometimes the start-up cost of building the development team, focused on making prototype, is high.
- Once we get proper requirements from client after showing prototype model, it may be of no use. That is why, sometimes we refer to the prototype as "Throw-away" prototype.
- It is a slow process.
- Too much involvement of client, is not always preferred by the developer.
- Too many changes can disturb the rhythm of the development team.
- Expense of implementing prototyping: the start up costs for building a development team focused on prototyping may be high. Many companies have development methodologies in place, and changing them can mean retraining, retooling, or both. Many companies tend to just jump into the prototyping without bothering to retrain their workers as much as they should.
- Developer attachment to prototype: Developers can also become attached to prototypes they have spent a great deal of effort producing; this can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture.
- Developer misunderstanding of user objectives.

Spiral Model

Merits

- Spiral Life Cycle Model is one of the most flexible SDLC models in place. Development phases can be determined by the project manager, according to the complexity of the project.
- Project monitoring is very easy and effective. Each phase, as well as each loop, requires a review from concerned people. This makes the model more transparent.
- Risk management is one of the in-built features of the model, which makes it extra attractive compared to other models.
- Changes can be introduced later in the life cycle as well. And coping with these changes isn’t a very big headache for the project manager.
- Project estimates in terms of schedule, cost etc. become more and more realistic as the project moves forward and loops in spiral get completed.
- It is suitable for high risk projects, where business needs may be unstable.
- A highly customized product can be developed using this.

Demerits

- Cost involved in this model is usually high.
- It is a complicated approach especially for projects with a clear SRS.
- Skills required, to evaluate and review project from time to time, need expertise.
- Rules and protocols should be followed properly to effectively implement this model. Doing so, through-out the span of project is tough.
- Due to various customizations allowed from the client, using the same prototype in other projects, in future, is difficult.
- It is not suitable for low risk projects.
- Meeting budgetary and scheduling requirements is tough if this development process is followed.
- Amount of documentation required in intermediate stages makes management of project very complex affair.

Evolutionary Model

Merits

- Risk analysis is better.
- Evolutionary model supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During life cycle software is produced early which facilitates customer evaluation and feedback.

Demerits

- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which a risk is.
- Can be costly to use.
- Highly skilled resources are required for risk analysis.
- Project’s progress is highly dependent upon the risk analysis phase.