Comparison of different life cycle models

Classical Waterfall Model: The Classical Waterfall model can be considered as the basic model and all other life cycle models are based on this model. It is an ideal model. However, the Classical Waterfall model cannot be used in practical project development, since this model does not support any mechanism to correct the errors that are committed during any of the phases but detected at a later phase. This problem is overcome by the Iterative Waterfall model through the inclusion of feedback paths.

Iterative Waterfall Model: The Iterative Waterfall model is probably the most used software development model. This model is simple to use and understand. But this model is suitable only for well-understood problems and is not suitable for the development of very large projects and projects that suffer from a large number of risks.

Prototyping Model: The Prototyping model is suitable for projects, which either the customer requirements or the technical solutions are not well understood. This risks must be identified before the project starts. This model is especially popular for the development of the user interface part of the project.

Evolutionary Model: The Evolutionary model is suitable for large projects which can be decomposed into a set of modules for incremental development and delivery. This model is widely used in object-oriented development projects. This model is only used if incremental delivery of the system is acceptable to the customer.

Spiral Model: The Spiral model is considered as a meta-model as it includes all other life cycle models. Flexibility and risk handling are the main characteristics of this model. The spiral model is suitable for the development of technically challenging and large software that is prone to various risks that are difficult to anticipate at the start of the project. But this model is more complex than the other models.

Agile Model: The Agile model was designed to incorporate change requests quickly. In this model, requirements are decomposed into small parts that can be incrementally developed. But the main principle of the Agile model is to deliver an increment to the customer after each Time-box. The end date of an iteration is fixed, it can’t be extended. This agility is achieved by removing unnecessary activities that waste time and effort.

Selection of appropriate life cycle model for a project: Selection of proper lifecycle model to complete a project is the most important task. It can be selected by keeping the advantages and disadvantages of various models in mind. The different issues that are analyzed before selecting a suitable life cycle model are given below:

- **Characteristics of the software to be developed:** The choice of the life cycle model largely depends on the type of the software that is being developed. For small services projects, the agile model is favored. On the other hand, for product and embedded development, the Iterative Waterfall model can be preferred. The evolutionary model is suitable to develop an object-oriented project. User interface part of the project is mainly developed through prototyping model.
- **Characteristics of the development team:** Team member's skill level is an important factor to deciding the life cycle model to use. If the development team is experienced in developing similar software, then even an embedded software can be developed using the Iterative Waterfall model. If the development team is entirely novice, then even a simple data processing application may require a prototyping model.

- **Risk associated with the project:** If the risks are few and can be anticipated at the start of the project, then prototyping model is useful. If the risks are difficult to determine at the beginning of the project but are likely to increase as the development proceeds, then the spiral model is the best model to use.

- **Characteristics of the customer:** If the customer is not quite familiar with computers, then the requirements are likely to change frequently as it would be difficult to form complete, consistent and unambiguous requirements. Thus, a prototyping model may be necessary to reduce later change requests from the customers. Initially, the customer's confidence is high on the development team. During the lengthy development process, customer confidence normally drops off as no working software is yet visible. So, the evolutionary model is useful as the customer can experience a partially working software much earlier than whole complete software. Another advantage of the evolutionary model is that it reduces the customer's trauma of getting used to an entirely new system.