Software Life-Cycle models
February 2020

Waterfall model

Waterfall model, proposed by Winston Royce in 1970, was the first well-defined model for software development process and it gained huge popularity in its time (Royce 1970). The model contains a strict sequence of requirements, specifications, design, implementation, integration and maintenance. In the beginning of a project, the customer has to provide detailed requirements to the project manager. After the requirements are verified, a design for the software is generated and then the implementation begins. Thus, the model contains different phases and only after verifying and approving one phase, can we start the next phase.

Strengths

This model of software development process produces detailed documentation of every phase because of the approval step at the end of every phase. Thus it's possible to extensively analyze the safety and security of the software. This property makes it useful for the development of critical systems.
This is also the simplest model for small and not-so-complex software projects, especially when the exact requirements are already known.

Weaknesses

In reality, it’s not possible to write down exact requirements for a complex software. And because of its sequential nature, change at the implementation or maintenance stage is very expensive. So this model is not suitable for real-life complex software.

Incremental model

Incremental model is like an interactive model, in which an initial version is built, followed by feedback from the customer and then changing it again until the requirements are finally met (Sommerville 2016). In this model, a basic skeleton is built first and later each increment adds some more functionality. Each increment can be verified before merging it to the main build, thus it's easy to have a bug free build in the end. After each increment, the user can interact with the product and provide feedback and requirement changes, if any.

Strengths

The cost of incorporating changes in requirements is much easier compared to the waterfall model because we just need to change the recent increments.
Since there is a product at the end of each increment, it gives two benefits. First, it's easier to get user feedback about the developing product. Second, job satisfaction is increased as manufacturers can see results of their work earlier.

Weaknesses

Since there are so many intermediate builds before final build, it's is not easy to document all the versions of the software. This might create problems in extensively analyzing the software after the final build.
With each increment, the complexity of the software increases as new functionalities are added. This leads to difficulty in maintenance and evolution of the product.

Applications of Incremental model

Incremental development is nowadays quite approach for software development. It has led to successful development of many huge projects. For example, in 1972, IBM FSD used iterative and incremental development (IID) to develop the command and control system for the first US Trident submarine. The project manager was awarded IBM Outstanding Contribution Award for his work (Larman & Basili, 2003). Another example of using this method is by TRW for a $100 million project to develop Army Site Defense software. In fact, this approach should be used anywhere where evolution of the product is required.

RUP model

Rational Unified Process (RUP) is a hybrid of waterfall and incremental model. It has four different phases, namely inception, elaboration, construction and transition. In inception phase, a business case for the system is established; requirements and architecture are developed in elaboration phase; the implementation (writing code and so) is done in construction part; and the system is finally deployed in transition phase. (Sommerville 2016)

This model minimizes the risk due to evolving requirements of the end-user. Since it tries to use the existing components again and again, it takes less time. But it might not be that effective in developing new and complex software.

References

Royce, W. W. 1970. Managing the Development of Large Software Systems: Concepts and Techniques. In IEEE WESTCON, 1–9. Los Angeles, CA.

SOMMERVILLE , I., 2016. Software Engineering. 10th ed. Boston: Pearson.

LARMAN, C., BASILI, V. R., 2003. Iterative and Incremental Development: A Brief History. In IEEE Computer Society.