Software development models essay examples

Business, Management

SDLC acronym stands for Software Development Life Cycle. A SDLC model provides a standard for planning, organizing and executing a software development process (Maheshwari, 2012). It involves phases like project initiation, requirements gathering and analysis, design, system specification, programming and testing, installation and maintenance, and phase-out. Two main approaches to SDLC depend on whether the projects can (predictive approach) or cannot (adaptive approach) be planned out in advance.

- Waterfall Model: Each phase is completed in sequence and then the results of a completed phase flow into the next phase. There is no going back once the phase is completed. The decisions made in each phase are frozen (finalized) and they cannot be changed. The advantage is that the system requirements are captured and decided long before software development begins. The design, development and testing phases are easy to plan and manage as requirements are confirmed. The drawbacks are that it becomes difficult to change requirements and the functional software (no prototyping) is visible only after delivery.

- Spiral Model: The project is divided into multiple parts. The development proceeds from the smallest to the biggest part and all phases are executed in sequence for each part. This is preferred for big mission-critical projects. The project functionality is delivered in chunks and there is high focus on risk mitigation. This is not an economical model as process and management overheads are high. This is not suitable for small projects and frequent functional changes. It is similar to incremental model but with emphasis on risk handling.

- V model: Just like the waterfall model, this involves a sequential execution

of phases. Each phase must be complete before the next phase begins. Testing is emphasized more than the waterfall model. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. This is simple, easy to use and manage, more testing-focussed and works well for small, medium sized projects. There is rigidity in phase execution which allows no flexibility and no early prototypes. Changes in requirements are difficult to incorporate as it impacts all test plans.

- Iterative model: This assumes that the deliverables of first cycle may have to be modified iteratively to deliver the software with acceptable functionality as it presumes that functional analysis may not be complete before the design phase. All iterations are not expected to deliver functional software but allows for prototype display and feedback incorporation (Nabil, 2010). This model allows for change in requirements as in real-world. There are management overheads and more time is required for delivery. Tracking the deliverables of different phases is a big challenge and could involve costly tools.

- Extreme programming (XP) and Agile: In XP, incremental development is supported through small, frequent system releases. There is heavy customer interaction with the team. Teams create quality code and avoid delays through pair programming and collective ownership. The changes to software are supported through regular system releases. This is suitable for small and medium sized projects; involves no overheads; ensures quality; frequent testing and incremental deliverables. It is difficult to scale-up to manage large projects, needs skilled and experienced talent for good quality

deliverables as talented pairs are high cost.

The selection of a SDLC model depends on the type of project and its objectives (Nabil, 2010). Projects generally go for a mixed (hybrid) model where there is enough flexibility to map with the real-world requirements. The waterfall is still the most natural and common method of development followed closely by a mix of iterative, agile and spiral development methodologies. Agile has currently gained prominence because of its practical, qualitative and win-win approach towards customers and developers.

References

Maheshwari Shikha, Jain Dinesh C. (2012). A Comparative Analysis of Different types of Models in Software Development Life Cycle. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 5, 285-290. www. ijarcsse. com

Nabil Mohammed Ali Munassar, A. Govardhan. (2010). A Comparison Between Five models of Software Engineering. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, 94-101. www. IJCSI. org