Development Process Models

Software lifecycle models are representations of the sequence and interrelationship of broad phases within the software lifecycle. Their principal purpose is to provide a high-level plan for software lifecycle activities. They are therefore essentially management tools. The use of a software lifecycle model on a software project is important. Without the plan it provides, it can be difficult to effectively manage the project.

Within the field of Computer Science, many software lifecycle models have been proposed. Each model has its own strengths and weaknesses, and each is more appropriate in certain project circumstances than others. It is generally recognised that no single software lifecycle model is appropriate in all circumstances. Because of this, for a particular software project, it is necessary to select a software lifecycle model that suits the project's characteristics. This is an important decision. The use of an inappropriate software lifecycle model can increase project costs and timescales and reduce software quality.

Professional system developers and the customers they serve share a common goal of building information systems that effectively support business process objectives. To ensure that cost-effective, quality systems are developed which address an organization's business needs, developers employ system development Process Model to direct the project's life cycle.

Typical activities performed include the following:

- System conceptualization
- System requirements and benefits analysis
- Project adoption and project scoping
- System design
- Specification of software requirements
- Architectural design
- Detailed design
- Unit development
- Software integration & testing
- System integration & testing
- Installation at site
- Site testing and acceptance
- Training and documentation
- Implementation
- Maintenance

Most system development Process Models in use today have evolved from three primary approaches: Ad-hoc Development, Waterfall Model, and the Iterative process.

The Idea of the Iterative Process:

1) STRENGTH

- Develop high-risk or major functions first.
- Each release delivers an operational product.
- Customer can respond to each build.
- Uses “divide and conquer” breakdown of tasks.
- Lowers initial delivery cost.
- Initial product delivery is faster.
- Customers get important functionality early.
- Risk of changing requirements is reduced.
- More flexible than waterfall.

2) WEAKNESS

- Requires good planning and design.
- Requires early definition of a complete and fully functional system to allow for the definition of increments.
- Well-defined module interfaces are required (some will be developed long before others)
- Total cost of the complete system is not lower.

3) OPPORTUNITIES

- Risk, funding, schedule, program complexity, or need for early realization of benefits.
- Most of the requirements are known up-front but are expected to evolve over time.
- A need to get basic functionality to the market early.
- On projects which have lengthy development schedules.
- On a project with new technology.

4) THREATS

- Social, Legal, and Ethical issues.
- Vendor specific solution.
- Loss of key staff.