IDC SDLC Methodology

LC 101

**Author:** Bhavani Prasad M



Plot No. 520, Road No. 27

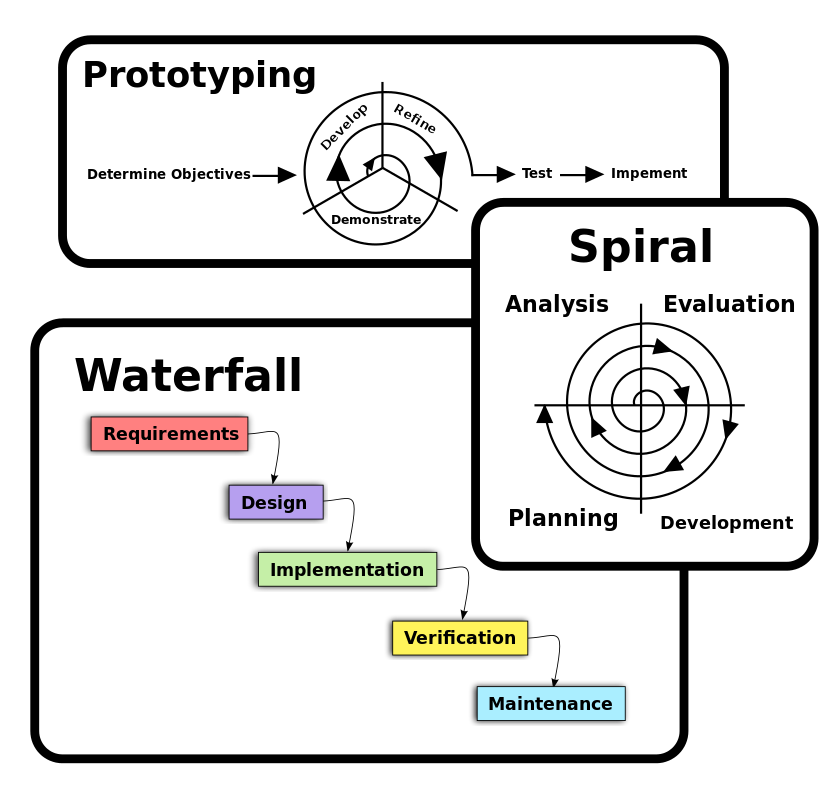
Jubilee Hills, Hyderabad - 33

A.P, India

www.indussoftware.com

A software development methodology is a framework that is used to structure, plan, and control the process of developing an information system - this includes the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.

Being a software product development company and being a company that constantly keeps adding value to its products at a rapid pace, Indus (IDC) does not find any one specific standard methodology suitable for its operations. But it needs the best traits from the proven models like waterfall, prototyping, iterative, spiral and the rapid application development. That creates IDC specific methodology, with the following features:

[](http://en.wikipedia.org/wiki/File:Three_software_development_patterns_mashed_together.svg)

**Diagrams of the three basic approaches applied to software development methodology frameworks** (given for easy understanding of the fundamental concepts)

**From Waterfall:**

1. Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
2. Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
3. Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

**From Prototyping:**

1. Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users’ requirements.
2. A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem.

**From Iterative:**

1. The initial software concept, requirements analysis, and design of architecture and system core are defined via Waterfall, followed by iterative Prototyping, which culminates in installing the final prototype, a working system.

**From Spiral:**

1. Focus is on risk assessment and on minimizing project risk by breaking a project into smaller segments and providing more ease-of-change during the development process, as well as providing the opportunity to evaluate risks and weigh consideration of project continuation throughout the life cycle.
2. "Each cycle involves a progression through the same sequence of steps, for each part of the product and for each of its levels of elaboration, from an overall concept-of-operation document down to the coding of each individual program."
3. Each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration.
4. Begin each cycle with an identification of stakeholders and their win conditions, and end each cycle with review and commitment.

**From Rapid Application Development:**

1. Key objective is for fast development and delivery of a high quality system at a relatively low investment cost.
2. Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
3. Aims to produce high quality systems quickly, primarily via iterative Prototyping (at any stage of development), active user involvement, and computerized development tools. Key emphasis is on fulfilling the business need, while technological or engineering excellence is of lesser importance.
4. Project control involves prioritizing development and defining delivery deadlines or “time boxes”. If the project starts to slip, emphasis is on reducing requirements to fit the time box, not in increasing the deadline.
5. Active user involvement is imperative.
6. Iteratively produces production software, as opposed to a throwaway prototype.
7. Produces documentation necessary to facilitate future development and maintenance.
8. Standard systems analysis and design methods can be fitted into this framework.

**Software Development Process:**

A software development process is a framework imposed on the development of a software product.

A decades-long goal has been to find repeatable, predictable processes that improve productivity and quality. Some try to systematize or formalize the seemingly unruly task of writing software. Others apply project management methods to writing software. Without project management, software projects can easily be delivered late or over budget. With large numbers of software projects not meeting their expectations in terms of functionality, cost, or delivery schedule, effective project management appears to be lacking.

As any software development methodology is comparable with a systematic software development process guided by certain guidelines, rules, paradigms and similar methodical activities, such a process qualifies as being described as a method. The term *methodology*, which is stigmatized as a buzzword, consequently breaks the logical link between the term and its intended content and therefore creating confusion. The creation and use of methods which incorporate other methods to produce an abstracter or farther scoped method (method of methods), which may even be guided by paradigms, does not qualify as methodology. A real software development methodology would, for example, require the analysis of different software development methods to identify similarities or differences in structure on a meta level, whereas the actual use of the content of the methods in a broader context is not within the scope of an methodology.

**Milestone Based Software Development Process Model:**

Overcoming these drawbacks requires a more flexible, iterative, process-oriented development model. The milestone-based process model presented here is derived from the product life cycle model proven so successful for small to large software product development companies. It encourages thinking about work in terms of processes rather than tasks. Milestones mark the self-regulation points of these processes.

The four characteristics of the milestone-driven model are:

Milestone-based approach: The application development process is driven by external and internal milestones, which are checkpoints to guide the development process.

Clear ownership and accountability: The process model connects responsibility for each milestone to the project team roles.

Risk-driven scheduling: High risk components of a project are completed as early as possible.

Versioned releases: The concept of versioned releases is an important one throughout the systems development life cycle because it impacts how expectations are set and how the entire project is planned and managed.

Each of these characteristics is discussed in the following sections.

Milestone-Based Approach

This process model consists of four high-level milestones as represented here:

**Process Milestones**

The milestones are depicted as checkpoints in the process flow — to emphasize that the process stage is could be completed only after the designed checks and approvals. But Milestones need not be freeze points, like in the linear model. Rather, they are "baseline" points at which the deliverables described by the milestone are placed under change control. This facilitates flexibility and successive refinement.

Briefly, the four major milestones are described in the following list, though there can be few other smaller milestones associated to these milestones.

**Vision/Scope Approved Milestone**

Once a requirement or set of requirements driven by the market developments or customer enquiries or internal ideas of the teams gains interest and approval, a project team is assembled to define the product or product version needs. A vision statement establishes scope and provides direction. The Vision/Scope Approved milestone is an opportunity for both end users and the development team to agree upon the project scope and vision.

**Functional Specification Approved Milestone**

The next visible milestone is availability of a functional specification baseline. This specification provides enough detail about the application so that the project team can begin identifying resource requirements and determining commitments. At this milestone, users and team members agree on what is to be delivered and establish priorities and expectations. This is an opportunity to reassess risk and to revalidate earlier estimates for schedules and resources.

**Code Complete Milestone**

An approved functional specification provides the baseline for focused development to begin. The development team sets a number of interim delivery milestones, but the critical milestone is, of course, Code Complete. The Code Complete milestone is an opportunity for users and team members to make a final assessment of the release and to verify that rollout and support plans and procedures are in place. At this milestone all new development ceases, and any deferred functionality is noted for the next release.

**Release of Product Milestone**

After the code has been baselined, beta programs, testing, and quality assurance activities are performed concurrently with further code refinement, driving the system to the Release milestone. The Release milestone is the point when the application (or "product") is formally turned over from the project team to the operations and support groups.

These four milestones are major progress points that embody all major development concepts. They are review milestones. Again, deliverables for the milestones are not frozen; instead, they are placed under change control.

Note The application under development is often referred to as a "product", while the strategies defined here certainly apply to developing commercially marketed software, they are by no means limited to it. Rather, developing a mindset in which the application is a "product" has significant advantages, including rapid application development, willingness to postpone new or risky features for future releases, and adherence to fixed milestones and ship