Software Development Methodology

a.k.a. System Development Life Cycle

Methodologies: What and Why?

Software engineering is the practice of using selected process techniques to improve the quality of a software development effort. This is based on the assumption, subject to endless debate and supported by patient experience, that a methodical approach to software development results in fewer defects and, therefore, ultimately provides shorter delivery times and better value. The documented collection of policies, processes and procedures used by a development team or organization to practice software engineering is called its software development methodology (SDM) or system development life cycle (SDLC).

Methodology as Risk Management

The challenge in selecting and following a methodology is to do it wisely -- to provide sufficient process disciplines to deliver the quality required for business success, while avoiding steps that waste time, squander productivity, demoralize developers, and create useless administrivia. The best approach for applying a methodology is to consider it as a means to manage risk. You can identify risks by looking at past projects.

If your organization has been plagued by problems resulting from poor requirements management, then a robust requirements management methodology would be well advised. Once this problem has been solved, through a repeatable process, the organization might then streamline its process, while ensuring that quality is maintained.

Every step along the system development life cycle has its own risks and a number of available techniques to improve process discipline and resulting output quality. Moving through the development life cycle, you might encounter the following major steps:

- Project charter and business case
- Definition of the business process and business requirements
- Documentation of user, functional and system requirements
- Top level architecture, technical approach, and system design
- System decomposition into component and unit specifications and design
- Coding, unit test planning, and unit test
- Generation of test data for unit testing and system testing
- System integration and testing
- Implementation, delivery and cut-over
- Training and user support
- System upgrades and routine software maintenance

In addition, you might have support activities throughout the development effort such as:

- Configuration management (version identification, baseline management and change control)
- Requirements management and tracability
- Quality management (quality assurance, quality reviews, defect tracking)
- System engineering reviews (requirements review, prelim. and critical design reviews, etc.)
- Support environment (development tools, libraries, files management, data management)

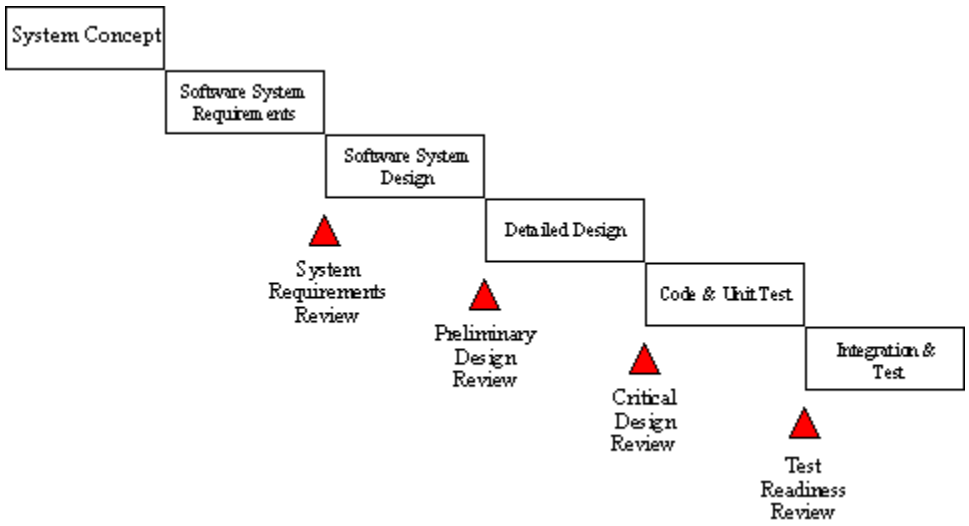
Written guidance for all these steps would constitute the core of your methodology. You can see how it wouldn't take long to fill a number of big binders with development processes and procedures. Hence, the importance of selecting processes wisely - to address known risks - keeping the methodology streamlined, and allowing for some discretion on the part of the project team.

Question: How is a methodologist different from a terrorist?

Answer: You can negotiate with a terrorist. :)

Waterfall Methodology

All projects can be managed better when segmented into a hierarchy of chunks such as phases, stages, activities, tasks and steps. In system development projects, the simplist rendition of this is called the "waterfall" methodology, as shown in the following figure:



In looking at this graphic, which was for major defense systems developments, please note this presumes that the system requirement have already been defined and scrubbed exhaustively, which is probably the most important step towards project success. Nevertheless, the graphic illustrates a few critical principles of a good methodology:

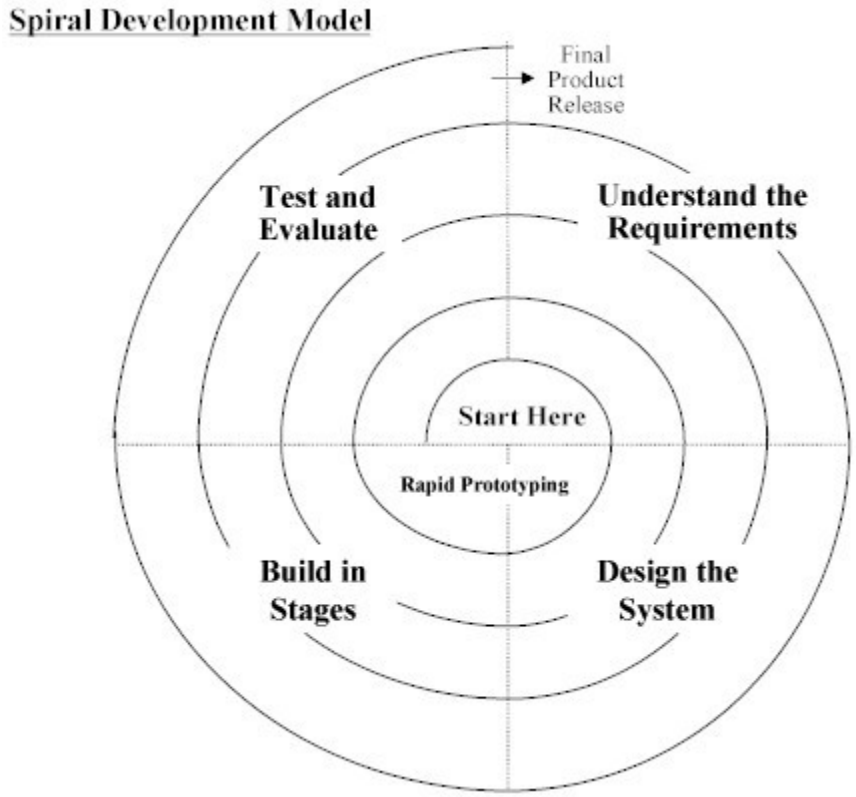
- Work is done in stages,
- Content reviews are conducted between stages, and
- Reviews represent quality gates and decision points for continuing.

The waterfall provides an orderly sequence of development steps and helps ensure the adequacy of documentation and design reviews to ensure the quality, reliability, and maintainability of the developed software. While almost everyone these days disparages the "waterfall methodology" as being needlessly slow and cumbersome, it does illustrate a few sound principles of life cycle development.

Spiral Methodology:

While the waterfall methodology offers an orderly structure for software development, demands for reduced time-to-market make its series steps inappropriate. The next evolutionary step from the waterfall is where the various steps are staged for multiple deliveries or handoffs. The ultimate evolution from the water fall is the spiral, taking advantage of the fact that development projects work best when they are both incremental and iterative, where the team is able to start small and benefit from enlightened trial and error along the way.

The spiral methodology reflects the relationship of tasks with rapid prototyping, increased parallelism, and concurrency in design and build activities. The spiral method should still be planned methodically, with tasks and deliverables identified for each step in the spiral.

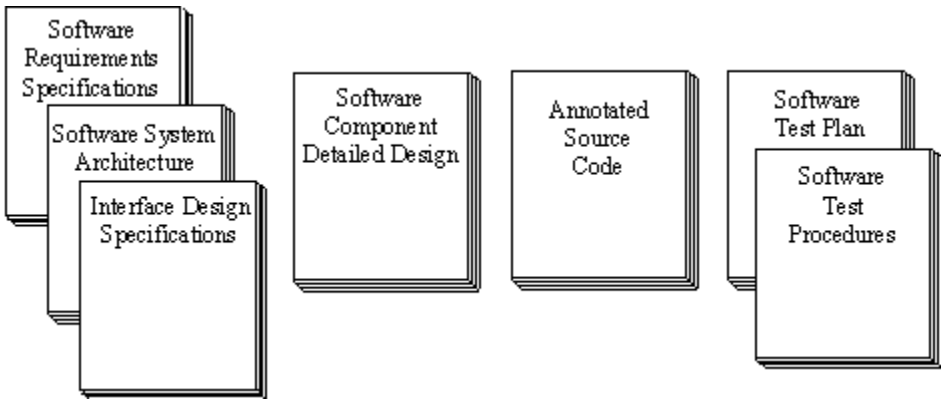


Documentation:

The reality is that increased processes usually result in increased documentation. An improved process produces intermediate work products that represent the elaboration of the product design at each step in the development life cycle. Where possible, documentation should be generated using automated tools, so outputs can contribute to generation of code structures or help generate the code itself.

The difference between hacking and software engineering is professional discipline applied with common sense. Software quality, reliability, and maintainability are enhanced by having good documentation for requirements, architecture, interfaces, detailed design, well-commented code, and good test procedures. Requirements documentation practices should facilitate your customer's understanding and review of the real requirements. Software project planning should include estimating the time and resources to produce, review, approve, and manage such documentation products.

Sample Software Documentation Work Products



Capability Maturity Model® Integration (CMMI)

Some years ago the Software Engineering Institute at Carnegie Mellon Institute in Pittsburgh established standards and guidance for developing software engineering disciplines and management. This was known as the Capability Maturity Model (CMM), and its use has become widespread among mature software development organizations, especially for those developing large scale software in a competitive procurement environment. Government and corporate software customers have increasingly required that proposals include information about a software development organization's certified level of maturity. The CMM had recognized five steps towards organizational software maturity:

- **Level 1 (Initial)** - Processes are *ad hoc* and occasionally chaotic. Few processes are defined, and success depends on individual effort and heroics. *(A street-person with a laptop would be at Level 1.)*
- **Level 2 (Repeatable)** - Basic project management processes are established to track cost, schedule and functionality. A process discipline is in place to repeat earlier successes on projects with similar applications.
- **Level 3 (Defined)** - Management and engineering processes are documented and integrated into a standard software process. Projects use an approved, tailored version of the organization's standard software process.
- **Level 4 (Managed)** - Detailed measures of the software process and product quality are collected. Processes and products are quantitatively understood and controlled.
- **Level 5 (Optimizing)** - Continuous process improvement is aided by quantitative feedback from the process and from piloting innovative ideas and technologies.

Around year 2000, the SEI introduced the next major evolution along this path - calledCapability Maturity Model® Integration (CMMI). This builds on the previous CMM, which is now regarded as "legacy" and no longer supported. You can find introductions to the CMMI and sample models at this Web link: SEI-CMMI models.

Go to: Project Management Training Index

Return to: Project Home