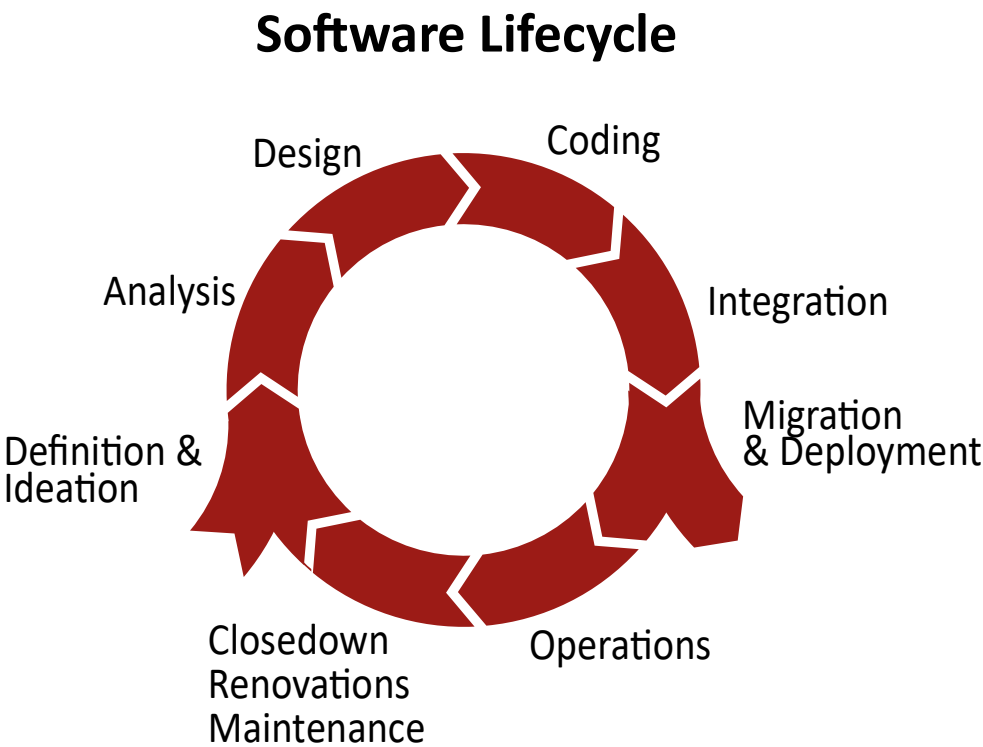




Process Paradigms for Software Development Projects

Lifecycle

The software development lifecycle is the general conceptual framework for software processes at all scales. It depicts the logical dependencies between phases without specifying their sizes, detail activities, or products. For very small programming tasks, we may not demarcate explicitly an “Analysis Phase”, say, but there always is one, even if it is only a minute activity in the head of the programmer.

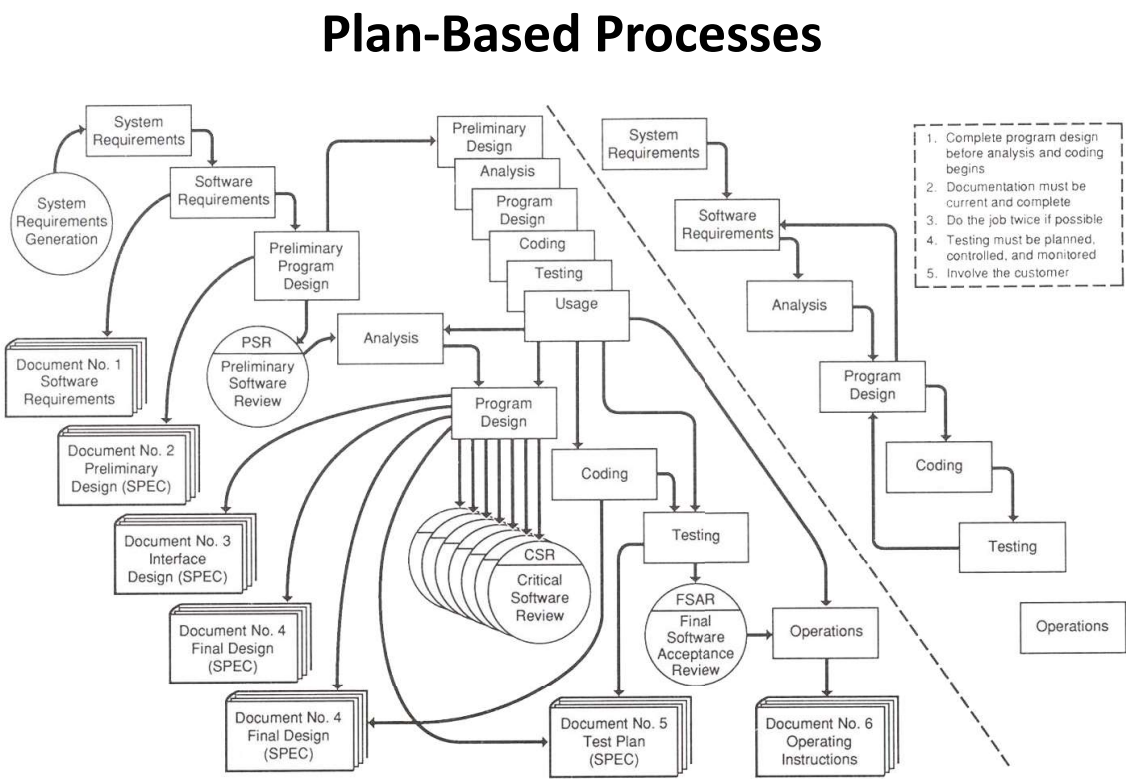


Benefits and Issues

Understanding the software lifecycle is paramount for understanding software development. It defines the essential concepts and terminology, and reminds us of the logical dependencies. Frequently, though, the lifecycle is misunderstood as being a *prescriptive* model, i.e., a set of instructions of how to run a software development project.

Sequential

The sequential model (often called “Waterfall”) has been developed in one of the first (and extremely successful) large scale development projects, the IBM S/360 in the 1960s. “Waterfall” is often used as a derogatory term implying strict sequence, rigid structure, and stifling regulation. However, even the original model shown on the right obviously does not advocate a strict, unidirectional top-down flow of artifacts. Instead, it specifically defines feedback loops.



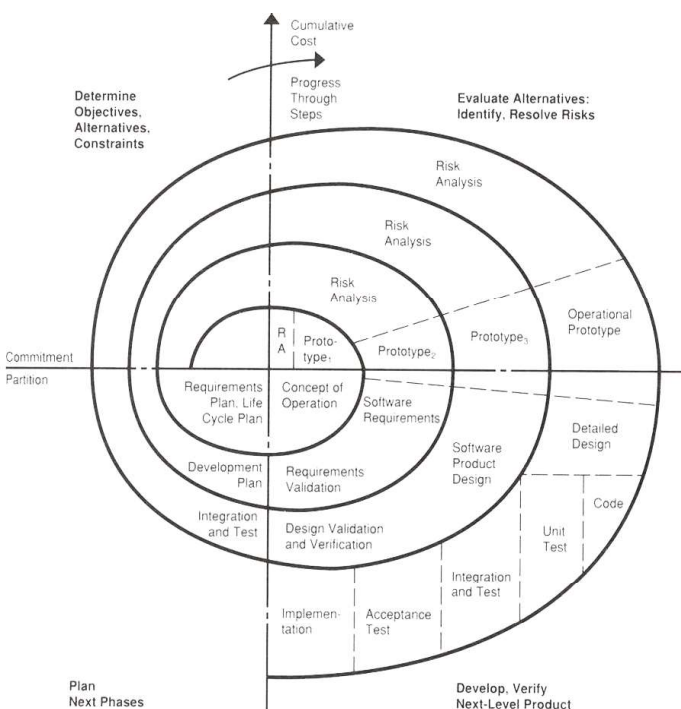
Benefits and Issues

The obvious benefits of a rigid, plan-based process are plannability and managerial control: it is always clear what is supposed to be happening next and who is responsible. This is essential for very large projects and highly regulated industries (e.g., aerospace, medical). However, reality does not always conform to our plans, so that predictions and estimations are very difficult. Sticking to a poor plan may result in catastrophic failures.

Spiral

Soon after the sequential process was proposed, the Spiral model followed. Where the sequential model focuses on planning and control, the spiral model suggests to create a sequence of (more or less complete) prototypes and revisit the whole life cycle iteratively. In each iteration, the product so far is evaluated and the plan is adapted. Thus, we trade effort for risk, and we rely on greater individual and organisational capabilities to ensure quality.

Product-Based Processes



Benefits and Issues

The obvious benefits of a flexible, result-based approach is that each iteration is shorter than the overall project, so less time expires between planning and delivery. Thus, prediction is easier, deviations from plans are smaller, and adapting the overall project is easier. The obvious drawback is that it is more difficult to stay on a pre-meditated plan, that each iteration incurs additional fixed cost, and that parts may have to be re-done after a plan-change.