Software Development Process

Li Qixiang
ZheJiang normal university

Software Development Process

Abstract

1.Preliminary understanding related systems analyst to user needs, and then use the relevant tools software list the major functional modules in the system, to develop each function module what small function module, the demand for some more specifically related to the interface, in this step can be preliminarily defined inside a small amount of interface.

2.The system analyst thorough understanding and analysis of demand, according to their experience and demand with a WORD or related tools to make a document system functional requirements document. This document will clearly outline system of big function module, function module what small function module, and also lists the relevant interface and interface functions.

3. The systems analyst to user confirm the demand again.

Keywords :requriement analysis , preliminary design ,detail design,coding

Introduction

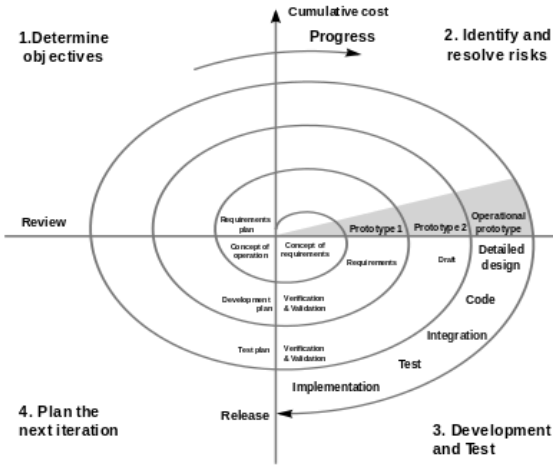
A software developments process is a set of activies that leads to the production of a software products .These acativities may involve the development of software from Scratch in standard programming language like Java or C . Increasingly, however ,new software is developed by extending and modifying existing systems and by condiguring and intergrating off-the-shelf software or system components.

History of Software Project Management

The software development methodology (also known as SDM) framework didn't emerge until the 1960s. According to Elliott (2004) the (SDLC) can be considered to be the oldest formalized methodology framework for building information system. The main idea of the SDLC has been "to pursue the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle—from inception of the idea to delivery of the final system—to be carried out rigidly and sequentially" within the context of the framework being applied. The main target of this methodology framework in the 1960s was "to develop large scale functional businessin an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines"

Methodologies, processes, and frameworks range from specific proscriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a "sponsor" or "maintenance" organization distributes an official set of documents that describe the process. Specific examples include:

Overview



In 1988, Barry published a formal software system development "spiral model," which combines some key aspect of the water model and rapid prototyping methodologies, in an effort to combine advantages of top-down and bottom-up concepts. It provided emphasis in a key area many felt had been neglected by other methodologies: deliberate iterative risk analysis, particularly suited to large-scale complex systems.

The basic principles are 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.

- "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."
- 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.
- Begin each cycle with an identification of stakeholders and their "win conditions", and end each cycle with review and commitment.

Method/techniques

Formal methods are mathematical approaches to solving software (and hardware) problems at the requirements, specification, and design levels. Formal

methods are most likely to be applied to safety-critical or security-critical software and systems, such as avionics software. Software safety assurance standards, such as DO-178B, DO-178C, and Common Criteria demand formal methods at the highest levels of categorization.

For sequential software, examples of formal methods include the B-Method, the specification languages used in automated theorem proving, RAISE, and the Z notation.

Formalization of software development is creeping in, in other places, with the application of Object Constraint Language (and specializations such as Java Modeling Language) and especially with model-driven architecture allowing execution of designs, if not specifications.

For concurrent software and systems, Petri nets, process algebra, and finite state machines (which are based on automata theory - see also virtual finite state machine or event driven finite state machine) allow executable software specification and can be used to build up and validate application behavior.

Another emerging trend in software development is to write a specification in some form of logic—usually a variation of first-order logic (FOL)—and then to directly execute the logic as though it were a program. The OWL language, based on Description Logic (DL), is an example. There is also work on mapping some version of English (or another natural language) automatically to and from logic, and executing the logic directly. Examples are Attempto Controlled English, and Internet Business Logic, which do not seek to control the vocabulary or syntax. A feature of systems that support bidirectional English-logic mapping and direct execution of the logic is that they can be made to explain their results, in English, at the business or scientific level.

Conclusion

A variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. One software development methodology framework is not necessarily suitable for use by all projects. Each of the available methodology frameworks are best suited to specific kinds of projects, based on various technical, organizational, project and team considerations