Software development process

Name Fangzhengsheng number 13211117
Zhejiang normal university
Software development process

Introduction

In software engineering, a software development methodology (also known as a system development methodology, software development life cycle, software development process, software process) is a splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management. It is often considered a subset of the systems development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.[1]

Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming and various types of agile methodology. Some people consider a life-cycle "model" a more general term for a category of methodologies and a software development "process" a more specific term to refer to a specific process chosen by a specific organization. For example, there are many specific software development processes that fit the spiral life-cycle model.

History

The software development methodology (also known as SDM) framework didn't emerge until the 1960s. According to Elliott (2004) the systems development life cycle (SDLC) can be considered to be the oldest formalized methodology framework for building information systems. 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"[2] within the context of the framework being applied. The main target of this methodology framework in the 1960s was "to develop large scale functional business systems in an age of large scale business conglomerates. Information systems activities revolved around heavy data processing and number crunching routines".[2]

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:

1970s
Structured programming since 1969
Cap Gemini SDM, originally from PANDATA, the first English translation was published in 1974. SDM stands for System Development Methodology
1980s
Structured systems analysis and design method (SSADM) from 1980 onwards
Information Requirement Analysis/Soft systems methodology
1990s
Object-oriented programming (OOP) developed in the early 1960s, and became a dominant programming approach during the mid-1990s
Rapid application development (RAD), since 1991
Dynamic systems development method (DSDM), since 1994
Scrum, since 1995
Team software process, since 1998
Rational Unified Process (RUP), maintained by IBM since 1998
Extreme programming, since 1999
2000s
Agile Unified Process (AUP) maintained since 2005 by Scott Ambler
Disciplined agile delivery (DAD) Superseded of AUP

Overview

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.[1]

Approaches

Several software development approaches have been used since the origin of information technology, in two main categories. Typically an approach or a combination of approaches is chosen by management or a development team.

"Traditional" methodologies such as waterfall that have distinct phases are sometimes known as software development life cycle (SDLC) methodologies, though this term could also be used more generally to refer to any methodology. A "life cycle" approach with distinct phases is in contrast to Agile approaches which define a process of iteration, but where design, construction, and deployment of different pieces can occur simultaneously.

Waterfall development

Main article: Waterfall model

The activities of the software development process represented in the waterfall model. There are several other models to represent this process.
The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases, typically:

Requirements analysis resulting in a software requirements specification
Software design
Implementation

Testing
Integration, if there are multiple subsystems
Deployment (or Installation)
Maintenance
The first formal description of the method is often cited as an article published by Winston W. Royce[3] in 1970 although Royce did not use the term "waterfall" in this article. The basic principles are:[1]

Project is divided into sequential phases, with some overlap and splashback acceptable between phases.
Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
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. Written documentation is an explicit deliverable of each phase.
Iterate:- Create a prototype, then the real system. Repeat each phase using new information gained and the entire process at least once before delivering the live system.
The iteration step has largely been omitted in practice due to how the methodology was taught and the perception at the time that it would be too costly for winning US DoD contracts. The waterfall model with no iteration or prototype is known as the 'traditional' engineering approach applied to software engineering, however it is a direct result of requirements for bidding for US military contracts. As a consequence, unless part of the project plan, a strict waterfall approach discourages revisiting and revising any prior phase once it is complete. This "inflexibility" has been a source of criticism by supporters of other more "flexible" models. It has been widely blamed for several large-scale government projects running over budget, over time and sometimes failing to deliver on requirements due to the Big Design Up Front approach. Except when contractually required, the waterfall model has been largely superseded by more flexible and versatile methodologies developed specifically for software development. See Criticism of Waterfall model.

The waterfall model is also commonly taught with the mnemonic A Dance in the Dark Every Monday, representing Analysis, Design, Implementation, Testing, Documentation and Execution, and Maintenance.[citation needed]

Formal methods

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.

References

1. Geoffrey Elliott (2004) Global Business Information Technology: an integrated systems approach. Pearson Education. p.87.
2.Wasserfallmodell > Entstehungskontext, Markus Rerych, Institut für Gestaltungs- und Wirkungsforschung, TU-Wien. Accessed on line November 28, 2007.
3. ieeecomputersociety.org
4.Barry Boehm (1996., "A Spiral Model of Software Development and Enhancement". In: ACM SIGSOFT Software Engineering Notes (ACM) 11(4):14-24, August 1986
5.J Richard H. Thayer, Barry W. Boehm (1986). Tutorial: software engineering project management. Computer Society Press of the IEEE. p.130
6. Barry W. Boehm (2000). Software cost estimation with Cocomo II: Volume
7. Whitten, Jeffrey L.; Lonnie D. Bentley, Kevin C. Dittman. (2003). Systems Analysis and Design Methods. 6th edition. ISBN 0-256-19906-X.