Beginning Software Engineering

Software engineering

maintain software systems. Beginning in the 1960s, software engineering was recognized as a separate field of engineering. The development of software engineering

Software engineering is a branch of both computer science and engineering focused on designing, developing, testing, and maintaining software applications. It involves applying engineering principles and computer programming expertise to develop software systems that meet user needs.

A software engineer applies a software development process to define, implement, test, manage, and maintain software systems.

== History ==

Software development process

to it, software development process often refers to the high-level process that governs the development of a software system from its beginning to its

A software development process prescribes a process for developing software. It typically divides an overall effort into smaller steps or sub-processes that are intended to ensure high-quality results. The process may describe specific deliverables – artifacts to be created and completed.

Although not strictly limited to it, software development process often refers to the high-level process that governs the development of a software system from its beginning to its end of life – known as a methodology, model or framework. The system development life cycle (SDLC) describes the typical phases that a development effort goes through from the beginning to the end of life for a system – including a software system. A methodology prescribes how engineers go about their work in order to move the system through its life cycle. A methodology is a classification of processes or a blueprint for a process that is devised for the SDLC. For example, many processes can be classified as a spiral model.

Software process and software quality are closely interrelated; some unexpected facets and effects have been observed in practice.

== Methodology ==

The SDLC drives the definition of a methodology in...

Reverse engineering

electronic engineering, civil engineering, nuclear engineering, aerospace engineering, software engineering, chemical engineering, systems biology and more

Reverse engineering (also known as backwards engineering or back engineering) is a process or method through which one attempts to understand through deductive reasoning how a previously made device, process, system, or piece of software accomplishes a task with very little (if any) insight into exactly how it does so. Depending on the system under consideration and the technologies employed, the knowledge gained during reverse engineering can help with repurposing obsolete objects, doing security analysis, or learning how something works.

Although the process is specific to the object on which it is being performed, all reverse engineering processes consist of three basic steps: information extraction, modeling, and review. Information extraction is the practice of gathering all relevant information for performing the operation. Modeling is the practice of combining the gathered information into an abstract model, which can be used as a guide for designing the new object or system. Review is the testing of the model to ensure the validity of the chosen abstract. Reverse engineering is applicable in the fields of computer engineering, mechanical engineering, design, electrical and electronic...

Software architecture

into software architecture knowledge management. There is no sharp distinction between software architecture versus design and requirements engineering (see

Software architecture is the set of structures needed to reason about a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.

The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as the blueprints for the system and the development project, which project management can later use to extrapolate the tasks necessary to be executed by the teams and people involved.

Software architecture is about making fundamental structural choices that are costly to change once implemented. Software architecture choices include specific structural options from possibilities in the design of the software. There are two fundamental

laws in software architecture:

Everything is a trade-off

"Why is more important than how"

"Architectural Kata" is a teamwork which can be used to produce an architectural solution that fits the needs. Each team extracts and prioritizes architectural characteristics (aka non functional requirements) then models the components accordingly. The team can use C4 Model which is a flexible method...

Systems engineering

control engineering, software engineering, electrical engineering, cybernetics, aerospace engineering, organizational studies, civil engineering and project

Systems engineering is an interdisciplinary field of engineering and engineering management that focuses on how to design, integrate, and manage complex systems over their life cycles. At its core, systems engineering utilizes systems thinking principles to organize the systems engineering body of knowledge. The individual outcome of such efforts, an engineered system, can be defined as a combination of components that work in synergy to collectively perform a useful function.

Issues such as requirements engineering, reliability, logistics, coordination of different teams, testing and evaluation, maintainability, and many other disciplines, aka "ilities", necessary for successful system design, development, implementation, and ultimate decommission become more difficult when dealing with large or complex projects. Systems engineering deals with work processes, optimization methods, and risk management tools in such projects. It overlaps technical and human-centered disciplines such as industrial engineering, production systems engineering, process systems engineering, mechanical engineering, manufacturing engineering, production engineering, control engineering, software engineering...

Rational Software

bought by IBM. It sought to advance its customers' practice of Modern Software Engineering through process, tools, and service. Rational Machines was founded

Rational Software was an independent software development company until 2003, when it was bought by IBM. It sought to advance its customers' practice of Modern Software Engineering through process, tools, and service.

Rational Machines was founded by Paul Levy and Mike Devlin in 1981 to provide tools to expand the use of modern software engineering practices, particularly explicit modular architecture and iterative development. Its initial product -- the Rational Environment -- was an integrated development environment for the Ada programming language, which provided good support for abstraction through strong typing. The Rational Environment was organized around a persistent intermediate representation (DIANA), providing users with syntactic and semantic completion, incremental compilation, and integrated configuration management and version control. The Rational Environment ran on custom hardware, the Rational R1000, which implemented a high-level architecture optimized for execution of Ada programs in general and the Rational Environment in particular. Rational later added code generators targeted to then-popular instruction set architectures such as the VAX, 68K, and X86; much...

Software requirements

the constraints on its operation. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as: A condition or capability needed

Software requirements for a system are the description of what the system should do, the service or services that it provides and the constraints on its operation. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

A condition or capability needed by a user to solve a problem or achieve an objective

A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document

A documented representation of a condition or capability as in 1 or 2

The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

Note that the wording Software requirements is additionally used in software release notes to explain, which depending on software packages are required for a certain software to be built/installed/used.

Software quality

In the context of software engineering, software quality refers to two related but distinct notions: Software's functional quality reflects how well it

In the context of software engineering, software quality refers to two related but distinct notions:

Software's functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for the purpose of a piece of software or how it compares to

competitors in the marketplace as a worthwhile product. It is the degree to which the correct software was produced.

Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability. It has a lot more to do with the degree to which the software works as needed.

Many aspects of structural quality can be evaluated only statically through the analysis of the software's inner structure, its source code (see Software metrics), at the unit level, and at the system level (sometimes referred to as end-to-end testing), which is in effect how its architecture adheres to sound principles of software architecture outlined in a paper on the topic by Object Management Group (OMG).

Some structural qualities, such as usability, can be...

SIGSOFT

Association for Computing Machinery (ACM)'s Special Interest Group on Software Engineering (SIGSOFT) provides a forum for computing professionals from industry

The Association for Computing Machinery (ACM)'s Special Interest Group on Software Engineering (SIGSOFT) provides a forum for computing professionals from industry, government, and academia to examine principles, practices, and new research results in software engineering.

SIGSOFT was officially formed in 1976 as the Special Interest Committee on Software Engineering (SICSOFT) by Tony Wasserman and R. Stockton Gaines, and converted to SIGSOFT in 1977.

SIGSOFT focuses on issues related to all aspects of software development and maintenance, with emphasis on requirements, specification and design, software architecture, validation, verification, debugging, software safety, mining software repositories, software processes, software management, measurement, user interfaces, configuration management, software engineering environments, AI for software engineering, and CASE tools.

SIGSOFT (co-)sponsors conferences and symposia including the International Conference on Software Engineering (ICSE), the ACM International Conference on the Foundations of Software Engineering (FSE) and other events.

SIGSOFT publishes the informal bimonthly newsletter Software Engineering Notes (SEN) newsletter...

Software verification

Software verification is a discipline of software engineering, programming languages, and theory of computation whose goal is to assure that software

Software verification is a discipline of software engineering, programming languages, and theory of computation whose goal is to assure that software satisfies the expected requirements.

== Broad scope and classification ==

A broad definition of verification makes it related to software testing. In that case, there are two fundamental approaches to verification: