The Evolution of Software Engineering: Best Practices and Emerging Trends

Eric Vara

The University of Arizona Global Campus

CST 301: Software Technology & Design

Professor John Howerton

February 06, 2023

**Diagram

Description automatically generated**

**The Evolution of Software Engineering: Best Practices and Emerging Trends**

This paper is about contemporary software engineering and its key features. It covers software development, software engineering, software process, process models, and system modeling fundamentals. It also touches on software engineering development techniques such as Agile methodology, plan-driven methodology, and requirements engineering. System modeling is an important aspect of software engineering and object-oriented software development. It helps define the structure, behavior, and interactions of a software system and its components. The architectural design of a software system is defined using system modeling and it is used to guide the implementation of the software system. The paper concludes that system modeling basics help create well-designed, scalable, and maintainable software systems by improving our understanding of the software system.

**Features of Contemporary Software Engineering**

Contemporary Software Engineering encompasses several key features. Software Development which includes the development of software systems using various methodologies, tools, and techniques. Software Engineering which is the application of engineering principles to the design, development, and maintenance of software systems. Software Process which refers to the set of activities involved in creating a software system, including requirements analysis, design, implementation, testing, and maintenance. Process Models which is a framework that provides guidelines and standards for software development, such as Agile, Waterfall, Spiral, and others.

Process Activities where there are the individual steps involved, including planning, design, coding, testing, and deployment. Software Design and Implementation that includes the creation of a detailed plan for software development, including the use of design patterns, data structures, and algorithms to implement software systems. Software Validation which refers to the process of ensuring that software meets its specified requirements and is fit for its intended use. This involves activities such as testing, inspection, and verification. Software evolution refers to the process of changing and updating software systems over time to address changing requirements and technology. This includes activities such as maintenance, refactoring, and updating.

**Software Engineering Development Techniques**

Software engineering is the process of designing, building, and maintaining software systems. There are different approaches and methodologies used in software development, including:

Agile methodology: An iterative and adaptive approach to software development that emphasizes collaboration, flexibility, and customer satisfaction. The Agile process is often divided into sprints, during which a small, cross-functional team develops and delivers a functional portion of the software.

Plan-driven methodology: A traditional, sequential approach to software development that follows a set of strict plans and specifications. The process is typically divided into distinct phases, including requirements gathering, design, implementation, testing, and deployment.

Requirements engineering: The process of eliciting, analyzing, and documenting the requirements of a software system. This includes defining the problem to be solved, identifying the stakeholders, and determining the functional and non-functional requirements.

Specifications: Detailed description of how a software system is expected to behave and perform. Specifications can include functional requirements, performance requirements, and design constraints.

Use cases: A description of how a software system will be used to solve a particular problem for a specific user. Use cases are often used to define requirements and to help design the software.

Validations: The process of checking that a software system meets the specified requirements and works as intended. This includes both functional and non-functional testing, such as unit testing, integration testing, and acceptance testing.

Change management: The process of managing changes to a software system, including changes to requirements, design, and code. Change management involves documenting and tracking changes, assessing the impact of changes, and controlling the release of new software versions.

**Various System Modeling Fundamentals**

System modeling is an important aspect of software engineering that helps to describe and understand the behavior and structure of a software system. Context-models represent the system's operating environment as well as the interactions between the system's stakeholders. The limits of the system and the actors who interact with it can both be determined using context models. Interaction-models describe the stakeholder and system communication channels. They support the definition of the system's inputs, outputs, and user-system interactions.

The system's constituent parts, and their interactions are represented by structural models. The architecture of the system and each of its parts, including the data structures and algorithms employed, are described using structural models. The dynamic behavior of the system is described by behavioral models, along with the events that cause state changes and the responses of the system to these events. The most important interactions between the system and its stakeholders can be found using behavioral models.

These system modeling fundamentals, which include the specification of architectural perspectives, architectural patterns, and application architectures, contribute to the architectural design of a software system. The architectural views of a software system represent several perspectives, including the data perspective, the process perspective, and the interface perspective. The behavior and structure of the system can be understood from many angles with the use of architectural views. Reusable solutions to persistent architectural issues are known as architectural patterns. They offer a guide for resolving particular issues, such scalability or performance optimization. An application architecture explains the general organization and operation of a software system.

They offer a high-level overview of the system, including its elements and their connections, the data structures, and algorithms in use, as well as interactions between the system and its users. In conclusion, the numerous system modeling basics help create well-designed, scalable, and maintainable application architectures by improving our understanding of the software system.

**System Modeling and Object-Oriented Software**

System modeling is an important aspect of object-oriented software development, as it helps to define the structure, behavior, and interactions of the system and its components. Object-oriented software development focuses on modeling real-world objects and their relationships, and system modeling provides a way to represent these objects and relationships in a software system. In the process of developing object-oriented software, system modeling is used to define the architectural design, including the components and their relationships, the interactions between the components, and the behavior of the system as a whole.

For example, a context model can be used to define the environment in which the software system operates and the interactions between the system and its stakeholders. An interaction model can be used to define the communication between the system and its stakeholders, and a structural model can be used to define the components of the system and their relationships. Once the architectural design has been defined, system modeling is used to guide the implementation of the software system. The structural model can be used to guide the design and implementation of the components of the system, and the interaction model can be used to guide the design and implementation of the communication between the components.

In conclusion, system modeling is a valuable tool in the development of object-oriented software, as it helps to define the architectural design and guide the implementation of the software system. By representing the structure, behavior, and interactions of the system and its components, system modeling helps to ensure that the software system is well-designed, scalable, and maintainable.

**Software Testing Plans**

Software testing is an important aspect of software development, as it helps to ensure that the software is functioning as intended and meets the requirements of its stakeholders. There are various software testing plans, including:

* Development Testing
* Release Testing
* User Testing

The mechanisms to control software changes or evolution processes include change management and version control. Change management is the process of controlling changes to the software, including the review and approval of changes, the tracking of changes, and the rollback of changes if necessary. Version control is the management of different versions of the software, including the tracking of changes and the ability to revert to a previous version if necessary.

Software maintenance is the process of maintaining and updating the software after its release. This includes fixing bugs, adding new features, and updating the software to meet changing requirements. Software maintenance is an ongoing process, and the software testing plan should include ongoing testing to ensure that the software remains functioning as intended. In conclusion, the software testing plan is an important aspect of software development, as it helps to ensure that the software is of high quality and meets the needs of its stakeholders. The software testing plan should include a variety of testing phases, including development testing, release testing, and user testing, as well as mechanisms to control software changes and evolution processes, and a plan for software maintenance.

**Software Engineering Applied to Real Life**

The concepts of software engineering, such as system modeling, software testing plans, and software development methodologies, are fundamental skills that can be applied in many areas of the software development process. These concepts can be useful in future courses and jobs in a variety of ways:

In future courses: The concepts of software engineering can be applied in advanced software development courses, where students can build on their knowledge of these concepts and apply them to real-world software development projects.

In future jobs: The concepts of software engineering can be applied in a variety of roles within the software development industry, including software development, software testing, software architecture, and project management. By understanding the concepts of software engineering, individuals can work effectively in these roles and contribute to the development of high-quality software systems.

In conclusion, the concepts of software engineering are valuable skills that can be applied in a variety of ways in future courses and jobs. Understanding these concepts and how to apply them can help individuals to contribute to the development of high-quality software systems and be successful in a variety of roles within the software development industry.

**Work Cited**

Gomez, J., &amp; 20, U. J. (2022, June 20). The four types of software maintenance. Koombea. Retrieved February 2, 2023, from <https://www.koombea.com/blog/types-of-software-maintenance/>

Full Scale. (2022, June 24). What is software reengineering. Full Scale. Retrieved February 2, 2023, from <https://fullscale.io/blog/software-reengineering/>

Software engineering - quick guide. Tutorials Point. (n.d.). Retrieved January 09, 2023, from <https://www.tutorialspoint.com/software_engineering/software_engineering_quick_guide.htm>

8 best software development methodologies. app development company. (n.d.). Retrieved January 9, 2023, from <https://www.uptech.team/blog/software-development-methodologies>

Kay, R. M. (2021, April 29). How to learn the fundamentals of software engineering – in a more interesting and less painful way. freeCodeCamp.org. Retrieved January 9, 2023, from <https://www.freecodecamp.org/news/learn-the-fundamentals-of-software-engineering/>

Koushik, B. (2022). What is the generic process model in software development? Educative. Retrieved January 19, 2023, from <https://www.educative.io/answers/what-is-the-generic-process-model-in-software-development>

Cherry, K. (2022, March 6). How is behavioral analysis used as a treatment? Verywell Mind. Retrieved January 19, 2023, from <https://www.verywellmind.com/what-is-behavior-analysis-2794865#:~:text=1%20Behaviorism%20is%20a%20branch,used%20to%20shape%20human%20behavior>.

Hernandez, R. D. (2021, April 20). The model view controller pattern – MVC architecture and Frameworks explained. freeCodeCamp.org. Retrieved January 19, 2023, from <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>

Osis, J., &amp; Donins, U. (2017). Unified modeling language. Unified Modeling Language - an overview | ScienceDirect Topics. Retrieved January 19, 2023, from <https://www.sciencedirect.com/topics/computer-science/unified-modeling-language>