Discussion 1: How is a software development process different from software modeling

Discussion Topic:

How is a software development process different from software modeling?
Explain how one helps the other in creating quality software.
Feel free to include a UML model in your discussion.

My Post:

Hello Class,

Software Engineering (SE) is the art of engineering high-quality software solutions. The Object Oriented (OO) approach, Software Development (SD) process, and Software Modeling (SM) are components of SE. This post explores these components, more specifically the relationship between SD and SM. How through this relationship software developer teams build systems that are efficient, robust, and provide high-quality software solutions to users.

Software Engineering

First, let's define Software Engineering. "The goal of SE is to produce robust, high-quality software solutions that provide value to users. Achieving this goal requires the precision of engineering combined with the subtlety of art" (Unhelkar, 2018, p.1). SE involves a wide range of functions, activities, and tasks such as:

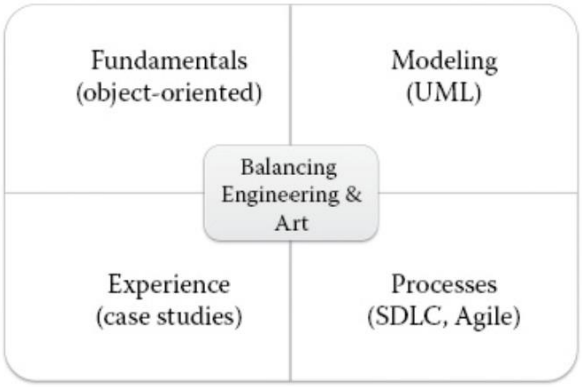
- Project management, business analysis, financial management, regulatory and compliance management**,** risk management, and service management functions.
  Functions are SE teams' responsibilities or disciplines that often span the entire software development lifecycle.
- Development processes, requirements modeling, usability design, operational performance, security, quality assurance, quality control, and release management activities.
  Activities are SE actions taken during certain stages of the software development, they are often performed repeatedly in functions or processes.
- Tasks are small SE's actions taken during certain functions or processes. They are often individual steps necessary to perform a specific activity.

(Unhelkar, 2018)

As shown above, SE is a complex process that can be decomposed into four components which are essential to learn and to adopt SE. These components are fundamentals or object-oriented, modeling (UML standard), process (SDLC, Agile), and experience (case studies and team-based project work).

Figure 1

The Four Essential Components to Adopt Software Engineering



Note: From "Software Engineering Fundamentals with Object Orientation. Software Engineering with UML" by Unhelkar (2018, p.2).

Below is a brief definition of each component:

Object-oriented (OO)

Object-oriented is the concept of object orientation based on Object Oriented Programming (OOP) languages such as Java and Python.
OO is composed of six fundamentals (encapsulation, inheritance, polymorphism, abstraction, composition, and association) that help in creating classes and programs that process and manipulate data and objects.

These OO fundamentals are as follows:

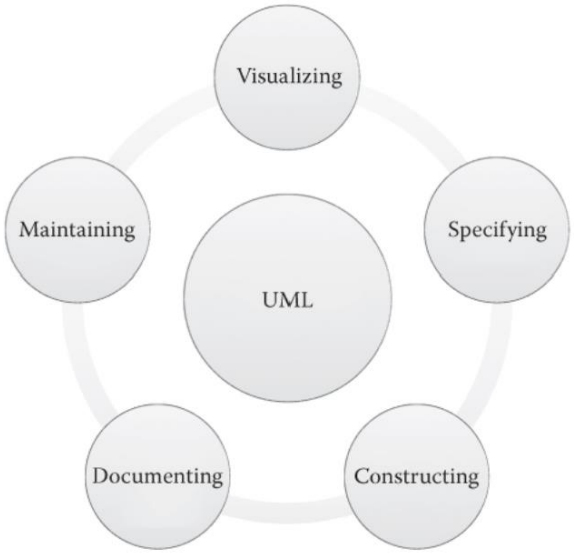
- Classification (grouping)
- Abstraction (representing)
- Encapsulation (modularizing)
- Association (relating)
- Inheritance (generalizing)
- Polymorphism (executing)

(Unhelkar, 2018, p.5)

Software Modeling (SM)

Software Modeling is a project modeling standard based on the Unified Modeling Language (UML) that is used the create diagrams to improve communication and participation from all project stakeholders. This also improves the quality of the software, reduces errors, and encourages easy acceptance of the solution by users. UML's purpose in SE is modeling, developing, and maintaining software.

Figure 2
Purpose of Unified Modeling Language in Software Engineering



Note: From "Software Engineering Fundamentals with Object Orientation. Software Engineering with UML" by Unhelkar (2018, p.13).

UML in modeling purpose modeling, developing, and maintaining software can be listed as follows:

- **Visualizing:** The primary purpose of the UML is to visualize the software requirements, processes, solution design, and architecture.
- **Specifying:** UML is used to facilitate the specification of modeling artifacts. For example, a UML class diagram can specify/describe the attributes and methods of a class, along with their relationships.
- **Constructing:** UML is used for software construction because it can be easily translated into code (e.g., C++, Java).
- **Documenting:** UML diagrams can be used as detailed documentation for requirements, architecture, design, project plans, tests, and prototypes.
- **Maintaining:** UML diagrams are an ongoing aid for the maintenance of software systems. Additionally, they provide a visual representation of a project's existing system, architecture, and IT design. This allows the developer to identify the correct places to implement changes and understand the effect of their changes on the software functionalities and behaviors.

(Unhelkar, 2018)

Process or Software Development (SD)

Process or Software Development is the process that defines activities and phases, as well as providing directions to the different teams of designers and developers throughout the Software Development Lifecycle (SDLC). Methodologies, such as Waterfall and Agile are used to guide and give structure to the development process and ensure that projects are completed in an efficient manner, meet quality standards, and meet user requirements. The waterfall methodological approach is linear and plan-driven whereas the Agile methodological approach is more flexible and adaptive.

These approaches are usually structured around 5 key components:

- Requirements gathering and analysis
- Design and architecture
- Coding and implementation
- Testing and quality assurance
- Deployment and maintenance

(Institute of Data, 2023, p.2)

Experience

Experience or case studies and team-based project work is the process of learning a project's best approaches and solutions through experimenting with UML and object-oriented fundamentals. “Experience in creating UML models, especially in a team environment, is a must for learning the art of SE” (Unhelkar, 2018, p.2)

How Software Modeling Supports Software Development

As described above, SD and SM are components of SE playing different roles in SDLC. The difference between SD and SM resides in SD being the methodological process that guides the creation and development of software, and SM being the representation of the software's architecture and functionality through diagrams based on UML. SM's primary role is to support SD by providing a visual representation of the project, reducing errors and scope creep, as well as providing documentation:

- **Project visualization**: UML diagrams, especially case diagrams, allow stakeholders to visualize program functionality and behaviors at a high level (Fenn, 2017). This helps teams to focus on where they need more requirements, details, and analysis. It supports the SM phase of requirements gathering and analysis.
- **Reducing errors and scope creep**: Software modeling can provide a clear model of the project by serving as a reference for the project requirements, minimizing errors, misunderstandings, and scope creep, particularly during the early stages of the software development process (Fenn, 2017). Scope creep is expanding or adding to the project requirements or objectives beyond the original scope. It supports the SM phase of design and architecture. This …..
- **Providing documentation**: UML diagrams can serve as living documentation for the project, describing the project functionality and behaviors as it is developed and after deployment. This documentation can help with decision-making for functionality/behavior implementation and maintenance of the software. It supports the SM phases of “coding and implementation” and “deployment and maintenance.”

By applying the concept listed above SM helps the SD process to create efficient, robust, high-quality software solutions that provide value to users.

UML Example

The following is a UML Class diagram of a simple banking manager Java program that utilizes the swing library, a graphical user interface (GUI) library. The program manages bank accounts and checking

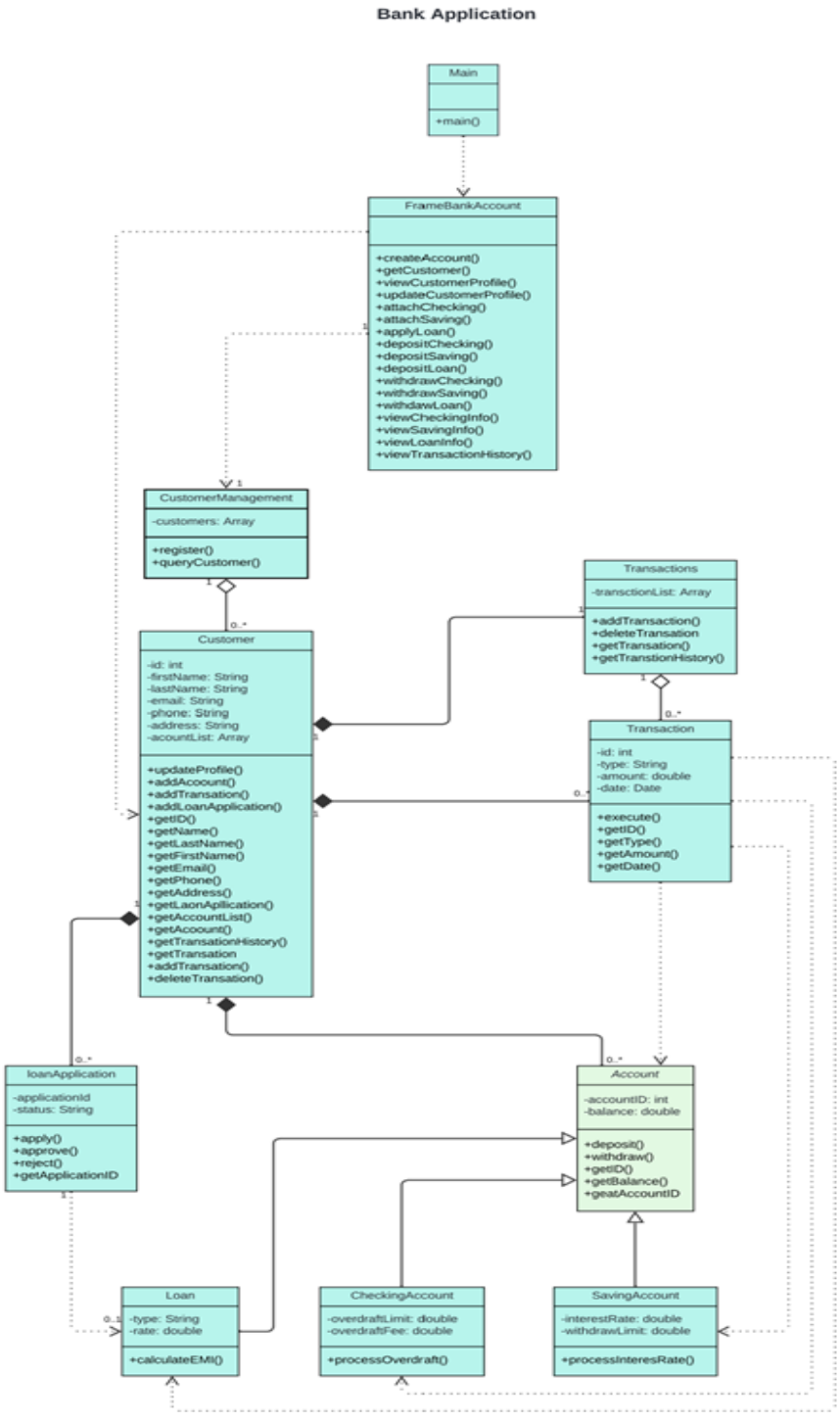
accounts with various functionalities such as creating accounts, attaching checking accounts, depositing and withdrawing funds, and viewing account balances.

> *In UML, class diagrams are one of six types of structural diagram. Class diagrams are fundamental to the object modeling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class diagrams to model the components of a system.*
>
> *Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide.*

(IBM, 2021)

Figure 3

UML Class Diagram Example



Note: From “Module-4: Portfolio Milestone” By Ricciardi (2024, p.5)

To summarize, SE is the art of engineering high-quality software solutions through OO, SD, and SM. SM helps the SD process by providing clear visual representations of system requirements and architecture, reducing errors, minimizing scope creep, improving communication among stakeholders, and serving as living documentation throughout the software development lifecycle.

-Alex

References:

Fenn, B. (2017, October). UML in agile development. *Control Engineering, 64*(10), 48. https://csuglobal.idm.oclc.org/login?qurl=https%3A%2F%2Fwww.proquest.com%2Ftrade-journals%2Fuml-agile-development%2Fdocview%2F2130716718%2Fse-2%3Faccountid%3D38569

IBM (2021, May 5) *Rational Software Modeler 7.5.5*. IBM. https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams

Institute of Data (2023, September 5). *Understanding software process models: What they are and how they work*. Institute of Data. https://www.institutedata.com/us/blog/understand-software-process-models/

Ricciardi, A. (2024, July 7). *Module-4: Portfolio Milestone.* CSC372: Programming 2. Depart of Computer Science. Colorado State University Global.

Unhelkar, B. (2018). Software engineering fundamentals with object orientation. *Software engineering with UML*. CRC Press. ISBN 9781138297432