**Module 6 Critical Thinking**

Michael Maye

Colorado State University Global

CSC470-1: Software Engineering

Dr. Shaher Daoud

06/25/2023

The software development and engineering process requires a clear and accurate understanding of the client or business's needs, wants, vision, and user interactions to deliver a system that fully captures the clients' purposes and functions regarding business operations. The system's users play a vital role in its creation, as a system that requires but is without users will inevitably become a failed system. Depending on the client's business model, users can be the clients' customers, employees, third-party actors, or any combination. With this wide range of possible users to consider during this process, there is a challenge in capturing, modeling, and translating user needs into an effective software solution. By analyzing these requirements, engineers can develop a deep understanding of the overall structure and function of the system they are asked to create.

This understanding alone is not enough to lead the project to a successful outcome; however, with this analysis in mind, engineers and stakeholders could design and analyze the system through modeling, creating a bridge between the analysis and the physical construction of the system by developers. Modeling these system aspects, structures, functions, and components before development allows engineers to develop and fine-tune many facets of the system without moving out of the planning phase instead of attempting to change system components or structures midway through the development process.

Through design, verification, testing, and re-design, the development teams engage in communication and deep analysis of the system. They can identify shortfalls, inefficiencies, information/functionality gaps, and technology incompatibles to ensure the system meets user and stakeholder expectations. In addition, modeling also gives a detailed and accurate view of the system's requirements throughout its lifecycle, which assists in planning and budgeting for both the development team and the client when the system is eventually delivered and deployed. This paper will discuss how engineering teams of all sizes employ the Universal Modeling Language, or UML, system on many projects and its importance during the Software Development Life Cycle.

**Project Types and UML:**

The New Software Development Project is a type of project that involves building a system from scratch, often starting with limited or no existing information. In these projects, meticulous planning and detailed design are essential for a successful system. UML models are critical in guiding the development and design process, ensuring the resulting system meets diverse requirements.

Different UML models facilitate success depending on the project's needs. Designers and developers typically utilize Case Use and Activity diagrams for projects that facilitate and support business operations. These diagrams allow them to model user needs and visualize the flow of business operations throughout a typical day. By identifying key interactions and dependencies, designers can create a system that aligns with the users' requirements and seamlessly integrates into their operations.

In contrast, projects focusing on systems architectural design solutions rely on Use Class, Sequence, and Solution State diagrams. These diagrams depict the system's various states and requirements during its use, visually representing the actions required to achieve the desired solution.

Using these UML models empowers teams entering a project with limited background information to develop a comprehensive understanding of the required solutions and the requirements. It enables them to gain the insights needed to design a new system that effectively integrates into, supports, and efficiently enhances the client's operations. Development teams can capture, model, and translate user needs into a well-designed and robust software solution.

**Integrating Applications and Services to Existing Projects:**

Integrating Applications and Services into Existing Projects is a common project undertaken by a Software Engineering Team. These projects aim to integrate a new system or application into an existing or legacy system. Successful integration of a new system requires careful investigation and consideration of the areas in which the two systems interface, communicate, and interact. Typical UML models for this project include the Class and Sequence diagram. These models define and detail the system's different classes and how its interactions are executed, managed, and eventually resolved, giving designers and engineers a detailed map of how the new system will integrate with the legacy system.

Using UML models in integrating applications and services enables teams to gain a practical and complete understanding of available resources, potential requirements, and system incompatibilities. This understanding allows for proactive planning and addressing and assists in overcoming issues during the planning phase, ultimately saving time and resources during development.

**Package Implementation Projects:**

In contrast to Integrating Applications and Services projects, the main focus of a Package Implementation Project is to take existing systems, generally off-the-shelf software systems, and configure them to integrate into an existing environment. These packages can include enterprise software suites such as Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) software, a Learning Management System (LMS), Human resources management (HRM), security or antivirus systems, and countless other systems designed to support operations. A project of this nature involves understanding the organization's existing processes, workflows, and data requirements, then mapping them to the functionalities and features offered by the Package.

To this end, these projects apply architectural diagrams such as Component and Deployment UML diagrams. Component diagrams give a visual understanding of the different software components and their relationships within the system, helping developers visualize and reference the Package's structure and how its different components interact. Deployment diagrams describe the distribution of software components across the environment's available hardware nodes or network assets. They give a visual of the physical infrastructure and required configuration to deploy the Package within the existing infrastructure successfully.

These modeling techniques and UML architectural diagrams allow project teams to communicate and document requirements, design, and deployment aspects of the Package. Modeling ensures a clear understanding of the problem, helps align the package solution with the organization's needs, and ensures seamless integration with the existing enterprise architecture.

**Mobile App Development Projects:**

Projects focused on mobile app development involve the development of software applications designed to run on mobile devices. Most of the planning process entails identifying the application's purpose and target audience. With countless types, sizes, operating systems, and intended functions of mobile technology, teams must carefully look at what the intended user would most likely require for the application to run successfully and meet expectations. These variations in technology can include screen sizes, operating systems and versions, hardware available on the device, and the demographic information of the typical user. In planning processes like these, developers utilize Use Case diagrams heavily to ensure they meet the users' expectations and consider how different users may need and use this application in various ways. By visualizing the interactions between the user and different functionalities of the applications in many different ways, the developers can understand what requirements may be needed for different user demographics or technology types, ensuring the application is successful across the spectrum of users. UML diagrams are handy for bridging the divide between storyboards and mockups. While mobile application projects also rely on systems other than UML diagrams, they are instrumental in creating storyboards, relating them to mockups, and modeling algorithms for analytics and processing. Using a UML Class or Activity, a developer or designer can easily take the components of the storyboard and map them to a mockup of the system, creating a visual depiction of the application and allowing for further analysis and tuning of the activities and components. Further in the process, the Activity and Sequence diagrams can help developers understand the algorithms and processes behind the app's different functionalities.

**Business Process Modeling Projects:**

Business Process Modeling Projects involve extensive modeling of current business systems, procedures, and processes within an organization to understand and improve them overall. These projects rely heavily on UML to effectively model and describe the organization. Activity Diagrams and Business Process Model and Notation (BPMN) are effective tools to accomplish this; they help capture the structure, flow, and behaviors of business processes and make them easier to understand and analyze. Activity diagrams depict the flow of different actions and processes, the operation of decision points, and interactions between different business processes by giving a detailed view of a given action and the steps involved, identifying decision points, and visualizing the control flow. The use of BPMN is integral to the success of these projects. It allows teams to use a standardized graphical notation designed to model different business processes and actions in an understandable and digestible way, ensuring developers, stakeholders, and any other party can analyze the systems involved in the project. It offers a comprehensive set of symbols and elements to represent different aspects of a process, such as tasks, events, gateways, and flows, which allow for detailed and precise documentation of the model's components and helps the model convey the intended information in a way that is easily understood by all.

**Cloud-Based Service Development Projects:**

A Cloud-Based Service Development Project involves the development of software and application systems designed to run in a cloud environment. These systems are typically accessed, managed, and used remotely by developers and users alike. In planning these projects, developers rely on many modeling systems, including Class, Component, Sequence, and Deployment diagrams. Class and Component diagrams allow developers to visualize the components within the system and how they will act or interact with each other, including input/outputs, message flow, data structures, and the requirements needed to implement these. Deployment diagrams map the services and components across the given hardware structure, nodes, virtual machines, and/or containers and help budget and design an efficient infrastructure to deploy the service or application. These UML diagrams provide a visual representation that aids in understanding the service's architecture, interactions, and deployment infrastructure. This understanding facilitates collaboration among team members, helps identify potential issues or bottlenecks, and guides the implementation and deployment of cloud-based services.

**Project Sizes**

Projects of any type can vary in size. Small projects generally involve a team of 5 to 15 people, can take 3-6 months, and usually have a budget of under two million dollars. Medium projects have teams of 15 to 20 people, last for around 6-12 months, and have budgets between three and ten million dollars. Large, whether isolated or collaborative, projects usually have teams of over 50 people, last longer than a year, and have budgets of over 10 million dollars.

**Small Projects:**

Small projects generally use UML in the solution space, emphasizing the design and implementation of software solutions rather than extensive documentation. In such projects, the primary focus is on delivering a practical and efficient software solution within the given time and budget constraints. Due to the project's size and duration, they are extremely agile. They can move quickly to a solution, unlike large projects requiring constant communication and coordination between departments. Projects of this nature emphasize less extensive documentation and more hands-on development, implementing solutions and systems quickly and efficiently. UML is integral to maintaining the speed and efficiency of these teams. With any given problem, a team of this size, if presented with properly planning, mapping, designing, and analyzing the system prior to actual production, saves both time and resources during to development and testing phases which can very easily convert to increased revenue for a business, user retention for user-related systems, and lives saved for safety-related programs.

Projects of this size utilize Class, Sequence, and State Machine diagrams to plan systems efficiently and quickly. Class diagrams model the system's structure, including the classes, their attributes, and the relationships and connections between them. They give an overview of the software system's structure and help developers visualize the organization and dependencies of its components. Sequence diagrams visualize the interactions, communication, and messages between components during a particular operation in runtime. These help developers plan and coordinate actions and messages of the running system coherently and efficiently and ensure the system's actions and behaviors align with user expectations and client requirements. State Machine diagrams give a detailed look at the behavior and actions of individual components. It visually represents the events and actions that occur during various system states and the conditions that caused that state, offering an excellent perspective to help developers design and fine-tune the logic and control flow of the system.

**Medium Projects:**

Due to their size and potential complexity, Medium Projects require a more formality or a structured development process with defined ceremonies and practices than smaller projects. As there is a need for a more formal project management environment, it may involve regular meetings, updates, constrained workflows, and consistent communication between teams, project managers, and stakeholders, all three of which have increased in size and responsibility. Due to the increased size and complexity of this project's scope, there are also higher expectations from users and stakeholders. In order to maintain this required level of quality, there is an increased need for thorough analysis and documentation of all systems involved in the project.

UML models help these projects map, keep track of, and analyze their systems to maintain high levels of quality and efficiency while developing. Use Case diagrams provide a way to capture user requirements and depict the users' interactions within the system, allowing developers to analyze efficiency and experience for the user and the system they are using and then share this with stakeholders and project managers to ensure it meets any requirements or constraints placed on the system. Activity diagrams help identify the key steps and potential decision points within a process, allowing for a detailed analysis and understanding of the system's behavior and, in the same vein as Use Cases, allow stakeholders and project managers to verify requirements, standards, and constraints along with verifying any security or vulnerability concerns.

**Large Projects:**

Large Projects work under different constraints than small or medium projects. Given that they are of such incredible scale and complexity, there is not only a practical need to create models for capturing and understanding the project's requirements and designing the solutions, but there may also be a legal or regulatory requirement to do so. Many industries currently have internal or external regulations and requirements regarding documenting the system's architecture, functionality, security measures, data handling procedures, or compliance with industry-specific standards, especially regarding users' personal information.

UML provides these projects with a standardized and accredited way to visualize and document the various aspects of the system. Use Case diagrams can give an extensive understanding of user requirements from many perspectives, ensuring that most scenarios are considered even with a vast user base. Activity diagrams help to visualize and analyze the sequence of cations, decision points, and branching possibilities in the increasingly complex environments within the system, allowing teams, stakeholders, project managers, and owners, as well as reviewers for regulatory compliance, to verify the operation and quality of the system. Class diagrams offer a useful way for multiple teams to work on the project and ensure that the different aspects of the system meet the requirements of and will integrate seamlessly with the system as a whole when in operation.

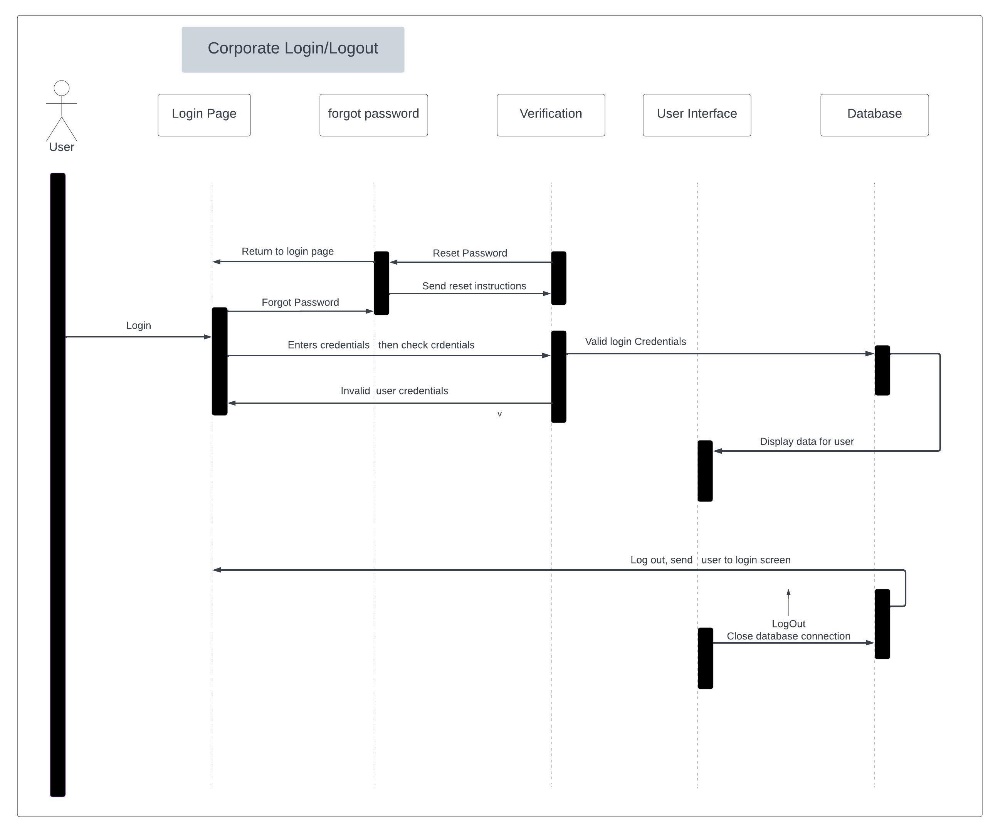
Collaborative projects are a special case, like large projects in size and scope, but have the increased complexity of being outsourced or having outside teams work alongside internal teams. The addition of outside collaboration adds an extra level of requirement for the documentation of systems within the project. UML models assist with the thorough documentation of a project's different systems and aspects, allowing outside teams collaborating on the project to easily work within the constraints and requirements of the system, as well as allow stakeholders and project managers to verify that work done by collaborators meets any requirements placed on the project.

In addition, UML also assists in the tracking, testing, and verification of deliverables throughout the development process. UML allows teams to monitor progress, identify dependencies, and ensure the quality of any milestone delivery to meet the timeline, quality, and regulatory restrictions.

**Visual Comparison:**

A screenshot of a computer

Description automatically generated with low confidence



To give a visual understanding of the types of expected differences between project sizes, above is a UML Sequence diagram of a log in action to a corporate website. The first is a large project that focuses on data and personal information security, the second is a small project that has a simple login/logout system. The large project has an authentication action that is called upon when the user logs in. Prior to the database giving access to the user, it verifies the authenticity of the approval by the authentication system. This adds levels of security to the database and the information contained in it. The small projects does not have this authentication step, and simply verifies the users credentials and allows access to the database through the user interface. While these two systems are different in terms of complexity and scale, they are both able to be properly modeled through the Sequence Diagram and offer the same insight into the steps taken within the system.

Resources:

Bashir, R. S., Lee, S. P., Khan, S. U., Chang, V., & Farid, S. (2016). UML models consistency management: Guidelines for Software Quality manager. *International Journal of Information Management*, *36*(6), 883–899. https://doi.org/10.1016/j.ijinfomgt.2016.05.024

IBM. (2021, March 5). *UML models and diagrams*. IBM. https://www.ibm.com/docs/en/rsas/7.5.0?topic=models-uml-diagrams

Unhelkar, B. (2020). 1, 2, 3. In *Software engineering with UML* (1sts ed., pp. 1–52). essay, CRC PRESS.