1. **Discuss which UML models most apply at different Software Development Life Cycle stages.**

The objects of a software system can be specified, visualised, built, and documented using the Unified Modelling Language (UML), a general-purpose visual modelling language. It includes choices and knowledge of systems that must be made. It is used to comprehend, plan, browse, configure, maintain, and manage data about these systems. It is designed to be applied across all development approaches, lifecycle phases, application domains, and media (Booch et al. 1999). For instance, throughout the entire development lifecycle. From requirements to deployment, the UML is seamless. The same ideas and notation can be applied at various phases of development and even combined within a single model. Moving from one level to another is optional. For iterative, incremental development, this seamlessness is essential.

As Martin (2019) explained, the Software development life cycle (SDLC) is a methodical process that guarantees the quality and accuracy of the software created. The SDLC process aims to develop high-quality software that fulfils client demands. The system development must be finished within the budgeted time range. The SDLC comprises a comprehensive plan that outlines how to organise, create, and maintain specific software. Each stage of the SDLC life cycle has a distinct procedure and outputs that feed into the following stage.

According to O'Reilly (N.D.), requirements gathering, high-level design, low-level design, coding and unit testing, integration testing, and deployment are some of the

traditional software development process stages. These domains are divided into several categories and subcategories using various approaches.

To meet each step of the Software Development Lifecycle (SDLC), UML offers a variety of diagram types. The following UML models are some of the most useful at various Software Development Life Cycle (SDLC) stages:

Use case diagrams to determine the specific groups of activities system users engage in during the high-level design phase. The use case diagrams also describe the participants in each use case. Using use cases might be beneficial when creating test strategies.

Class diagrams specify the domain model for the application as early as the high-level design process, particularly the interactions between and among data objects within the system and the operations that can be carried out on them.

Activity diagrams help define system process flows during requirements gathering and high-level design. Activity diagrams, as opposed to programme flow charts, contain users and activities outside of the code itself and allow for the distinct identification of the responsibilities played by various players.

Sequence and communication diagrams between components in a system are used to model these interactions. They are frequently employed in the SDLC's design and implementation phases.

State machine diagrams represent the various states in which an object may be and the events that may cause it to change conditions. Typically, they are employed during the SDLC's design and implementation phases.

The following table lists the UML models that are best suited for use at various phases of the SDLC:

SDLC Phase	UML Model
Requirements Gathering	Use case diagrams
Design	Class diagrams, sequence diagrams, communication diagrams, activity diagrams, state machine diagrams
Implementation	Class diagrams, sequence diagrams, communication diagrams, activity diagrams, state machine diagrams
Testing	Sequence diagrams, communication diagrams, activity diagrams, state machine diagrams
Deployment	Class diagrams, sequence diagrams, communication diagrams, activity diagrams, state machine diagrams

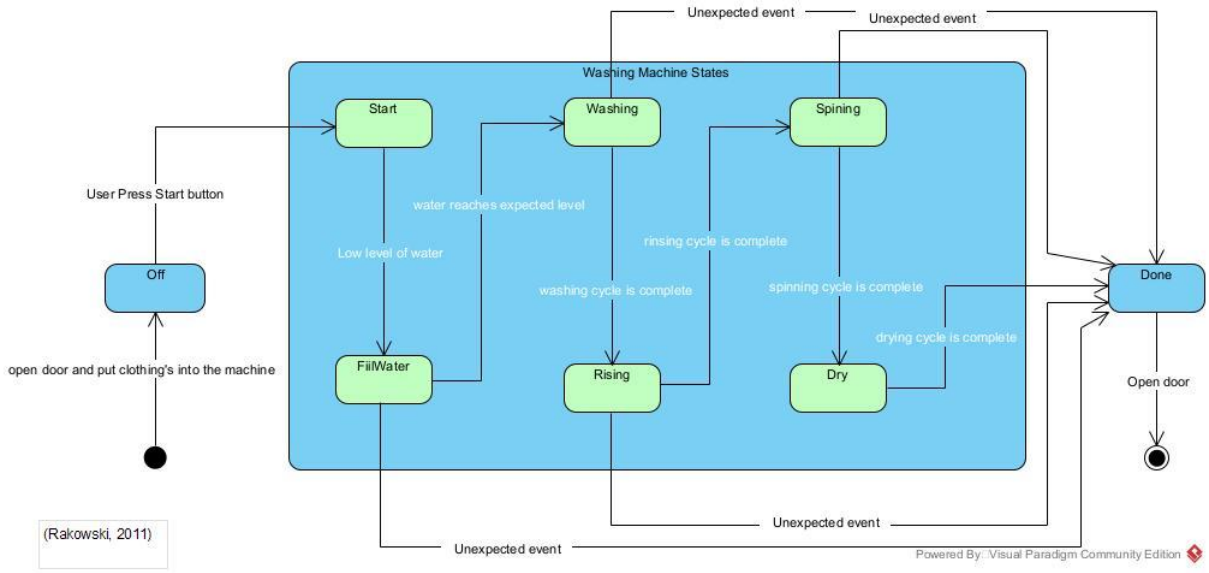
2. **Making reference to 'The Unified Modeling Language Reference Manual Second Edition', use the State Machine Diagram in Figure 3-7 to design a similar model for a washing machine.**

A state machine simulates the potential life histories of a class of items. States in a state machine are connected via transitions. Each state represents an object's lifetime period when it meets specific requirements. When an event occurs, a transition that moves an object to a new state may be fired. An impact (action or activity) associated with a transition may occur when the transition fires. State machine diagrams display state machines (Booch et al. 2004). User interfaces, device controllers, and other reactive subsystems can all be modelled as state machines. They could also be used to define passive objects that, throughout their existence, move through multiple qualitatively diverse stages, each with a particular behaviour.

The following Diagram is a model for a washing machine that illustrates the various states a washing machine might be in and the potential triggers for those states to

change. The diagram, for instance, demonstrates that the washing machine may be in many states, including "Off," “Start”, "Filling," "Washing," "Rinsing," "Spinning," “Dry” and "Done." Unexpected events may also occur to the washing machine; in this case, it will instantly end the current state and move to the Done state.

The washing machine begins in the **Off** position, then switches to the **Start** state when the user clicks the **Start** button and starts **filling** with water; when the water reaches the expected level, it switches to the washing state; now, is washing and cleaning the clothing; when this process finish, moves into the **rinsing** state after the washing cycle is finished; now clean water is used to **rinse** the clothing; when ends this process, switches to **spinning** to remove extra water; then moves into the **Dry** state when the spinning cycle is finished. When it finishes dry, the Machine stops and switches to the **Done** state. The washing machine will also change its state in response to unexpected events.



Additional information for each state:

Off: The washing machine is not in use and has been turned off.

Start: The user has chosen a wash cycle and turned on the washing machine.

Filling: Water is being poured into the washing machine.

Washing: The garments are being cleaned and agitated by the washing machine.

Rinsing: Clean water is used to rinse the clothes in the washing machine.

Spinning: The washing machine spins the garments to get rid of extra water.

Dry: The clothes are dried in the washer.

Done: The washing machine's wash cycle is complete and now ready for unloading.

References:

Rumbaugh, J., Jacobson, I. & Booch, G. (1999). The unified modeling language reference manual

Martin, M. (2019). SDLC (Software Development Life Cycle) Tutorial: What is, Phases, Model. [online] Guru99.com. Available at: https://www.guru99.com/software-development-life-cycle-tutorial.html.

O'Reilly (N.D.). *2.3. UML and Software Development Lifecycles - J2EE Design Patterns [Book]*. [online] Available at: https://learning.oreilly.com/library/view/j2ee-design-patterns/0596004273/ch02s03.html [Accessed 5 Jun. 2023].

Booch, G., Jacobson, I. & Rumbaugh, J. (2004). *Advanced Praise for The Unified Modeling Language Reference Manual, Second Edition. (n.d.).* Available at: https://personal.utdallas.edu/~chung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf.

Shawn Rakowski. (2011). *The State Diagram – Washing Machine*. [online] Available at: https://srakowski.wordpress.com/2011/02/19/the-state-diagram-washing-machine/ [Accessed 10 Jun. 2023].