**Name**

**Institution**

**Course**

**Professor**

**Date**

**Flow-based behavior modeling**

Undertakings give a way of illustrating flow-based performance that is denoted in a diagrammatic form. An activity characterizes a well-ordered pattern of action that converts inputs to output. Parameters refer to outputs and inputs of activity. An activity consists of arrangements that signify the bottom level of a hierarchy of its behavior. An activity uses inputs token that in turn produces output tokens by the use of its pins. Flows usually connect actions that are of two kinds. The flowing tokens require aligning or keeping future processing. They are stored in specialized nodes referred to as the central buffer nodes. Datastores can also be used in storing tokes. Flows are categorized as streaming or continuous, depending on their domain. The control flow is the other type of flow that transports flows from one activity to the other through control tokens that include the; merge, join, fork and decision. Control flow permits flow to divide and merge in different ways. There also exist some specific nodes that define what happens at the start and the end of action; they include; the initial nodule, final activity nodule, and the final flow-nodule

Actions are also grouped as the call action or the send signal action. The call to action is crucial since one activity invokes the other. Call actions contain pins similar to the parameters of a known entity. Recalls behavior activity permits an activity to comprise accomplishment of a different activity. They, however, use a transmitting device of the SysML blocks to decouple the initiator using the awareness of invoked conduct.

On the other hand, the send signal action allows communication through signals. The conduct of activities and actions can be limited in some means, like totting up pressure on the duration of action. Activity appeal offers the ability to give responsibility to actions within an activity to connect through signals rather than parents. When executing an activity in the setting of a region, the doings can allow indications transfer either to the region or straight to the activity; Designed activities permit the modelers to categorize engagements that require to be executed together like the conditional execution. Regional define diagrams define the hierarchical connection among activities and the connection goings-on with their outputs and inputs. Using a blocked definition drawing for these reasons is the same as the traditional practical hierarchy figure.

**Modeling message-based behavior**

Sequence figures described interactions used to apprehend systems situations as actual occurrences across numerous system blocks represented by the lifeline. An interaction is definite using incidence specifications grouped in chains and ordered by interaction users. If an interaction is successful, it assesses a group of event occurrences produced by its lifeline and defines whether they are legal. The most crucial basis of occurrences is distributing information among lifelines activating executions.

The critical aspect of interactions. Lifeline signifies the block that maintains interactions. During interactions, a lifeline merely indicates one occurrence; hence, if the block has a superior bound higher than one, an extra sector appearance is needed to represent precisely one of the occurrences represented by the block. Lifeline runs across a sequence diagram, showing that the blocks they represented existed previously and after the execution of the interaction. Comprehension of the messages cam be passed assynchronously as communication is passed and a response validation is queued. With the semantical identfication and ordering, an operator and operands can be evaluated based on the specific program section. Internal descriptions can also be described by the various interrelations of components.

.Interaction can enhance scalability by using other interactions. It can also describe the internal interaction of one of its lifelines by using other interactions. An interaction can use other related interactions to define some region of its entire behavior, including the number of its lifelines. This disintegration is done either to minimize the scope of the system diagram or to recycle some major pattern of the interaction. Interaction settings can feature linking points on their parameter known as gates. To allow information to pass through the interaction borders.

**Modeling events based behavior with state machines**

A state machine is used to define a block's behavior concerning its state and the transition amongst them. State machines are poised hierarchically as other SysML interactive constructs, allowing for state-based deeds at random compound representation. A state machine denotes a hypothetically recycled definition for the state-dependent activities of a block. Every state machine figure defines a single block of a state machine. They also contain a single sector with some state and pseudo-states and a transition. The entire region exhibits an active state that determines the current viability when executing a state machine. A part contains an initial and a final pseudo-state corresponding to its start and end, correspondingly.

A state is a perception of some necessary coding in the span of a block. It identifies the influence of movement and ways the block responds to the conditions using the behaviors'. Transition defines a valid state reform and the circumstance that leads to change. Transition consists of a guard, an effect, and triggers. A trigger is connected to an event corresponding to the call event or a signal event by its block. A guard shows an additional barrier that must be contented for the changeover to be triggered. In a lawful event, the guard is executed, and if it is correct, the change is activated.

Transition includes a transitional impact defined by behavior; hence, an effect is executed when the transition has triggered. In some circumstances, simple transitions are insufficient to stipulate the requisite behavior. The choice and junction pseudo-states allow various changes to combine and form compound transitions. Even though compound transition only has one chance with prompts, it can have several changeovers with effects and guards. Junction and choice transitions can have several movements, forming compound transitions with more than one transition path. On the other hand, history, pseudo-state allows for the distraction of a region.

A state machine can be recycled through submachine states. The interactions with the recycled devices occur through transitions within the periphery of the conforming submachine state either by a direct entry or exit. Change machines are propelled by principles of variables of the state machines or the features of its block. Therefore a behavior constrains some variables that must be factual with the relevant state.

**Modeling functionality with used cases**

The user case captures the functionality of a structure required to attain operator goals. An operator case usually describes the functionality necessary for a design. It can enhance SysML necessities to improve the meaning of a functional text-based requirement. The courses which use case operations are highly reliant on the method used. A use case defines a specific use of a system to attain the desired operator goal. Use aces relationship, extension, clarification, and inclusion are beneficial for future identification of the standard functionalities in a use case that other use cases can recycle. A use case is typically done as part of the base used case. Exemptions usually carry out a use case that encompasses the base used case, and general sand does not directly support the base use case's goals.

This topic provides the functionality needed by actors expressed as use cases. Actors define a role played by an entity outside the system and can signify individuals, organizations, or external systems. Generalization is used to classify associations among different performers—association-related performers to the use case they participate in.

Works Cited

Friedenthal, Sanford, et al. A Practical Guide to SysML. 3rd ed., 2014.