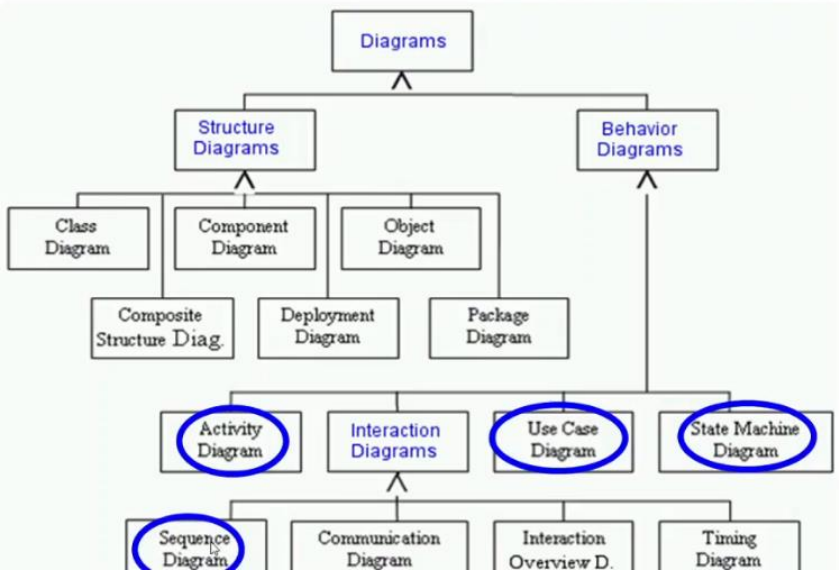
Contenido

UML

Unified Modeling Lenguage, es un conjunto o colección de diagramas y notaciones para modelado, documentación, especificación y visualización de sistemas complejos a diferentes niveles de abstracción y desde diferentes puntos de vista. Es tan versátil que se puede utilizar en cualquier fase del proceso de desarrollo, realizando los diferentes pasos(Requerimientos, Análisis, Diseño) de manera similar.
UML no es un lenguaje de programación de alto nivel, tampoco es un modelo de procesos, solo es una herramienta que te ayuda en el proceso.

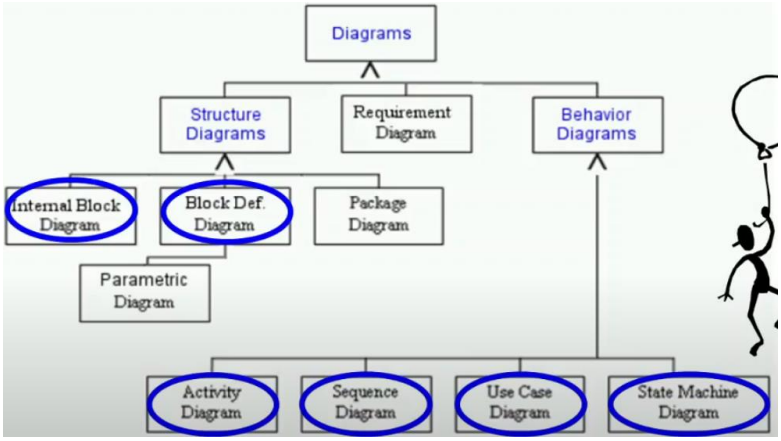
Diagramas de UML



SysML

System Modeling Lenguage, es una extensión de UML para escribir sistemas que consisten en Software y/o hardware o subsistemas, se enfoca en requerimientos e ingeniería de Sistemas.

Diagramas SysML



Use case Diagram

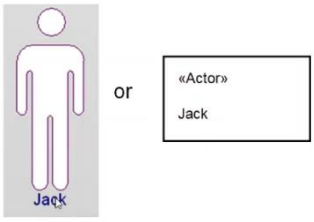
Aplica para SysML y UML, y brinda una vista de alto nivel(entry point) para definir un sistema, porque nos dice para que estamos haciendo un desarrollo, tu diseño, cual es la parte que cubre y que partes no cubre, básicamente ayuda a limitar las fronteras de tu sistema, que es lo que esta adentro, que está afuera, pero influye en el sistema, con quien o que vas a interactuar. Debe de ser lo más simple posible, y si existen varios diagramas de caso de uso, tienen que ser independientes entre sí. Una pregunta que responden los diagramas de caso de uso es ¿Qué funcionalidades(servicios) deberá proveer mi sistema?

Los elementos de este modelo son:

- Caso de uso
- Actor
- Limites(fronteras) del sistema
- Tipos de relaciones (Asociación, Generalización, Dependencias(extend, include))
- Comentarios

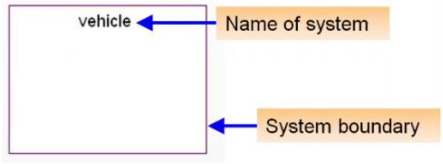
Se representa con un ovalo que tiene un nombre que es el identificador del caso de uso, la descripción se agrega dentro del modelo, un caso de uso describe un conjunto de acciones que se ejecutan una después de otra, llevado a ejecutar un comportamiento específico del sistema, el cual resulta en un valor medible para el actor, la descripción de estos casos siempre está vista desde el punto de vista del usuario del sistema, son independientes entre sí. Se recomienda agregar información adicional que complemente la información del caso de uso, por ejemplo: intención su pre y post condición, descripción de sus pasos, el path principal que ejecuta el caso de uso y paths alternativos como manejo de errores.

Actor



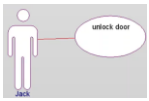
Un actor modela un rol que interactúa con un caso de uso, puede ser cualquier elemento o sistema que interactúe con nuestro sistema o que las acciones o funcionalidades de nuestro sistema impactan sobre él. Por convención se ponen a la izquierda del boundary box los actores usuarios del sistema, del lado derecho aquellos actores que son usados por el sistema.

System Boundary o Boundary Box



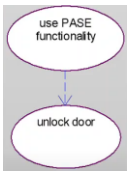
Un rectángulo que tiene el nombre del sistema, el marco representa el marco o borde de nuestro sistema, fuera del marco de nuestro sistema, el sistema interactúa con elementos por medio de las relaciones que describen nuestros diagramas de caso de uso, sirve para encapsular que pertenece al sistema, que esta afuera del sistema y los actores.

Asociación



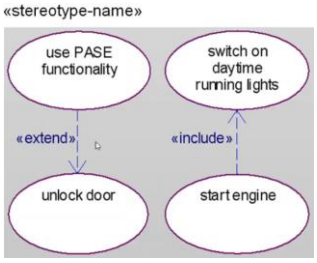
Se representa con una línea continua, sin flecha lo que significa que no indica un flujo, por lo que es una relación binaria o bidireccional.

Dependencia



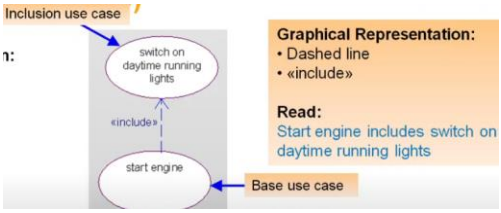
Se representa por una línea entrecortada y tiene una flecha en uno de los extremos, indica una situación en la que un cambio en el elemento objetivo (unlock door), puede requerir de un cambio en el elemento fuente (use PASE functionality). El elemento de la cola de la flecha depende del modelo que se encuentra en la cabeza de la flecha.

Estereotipo



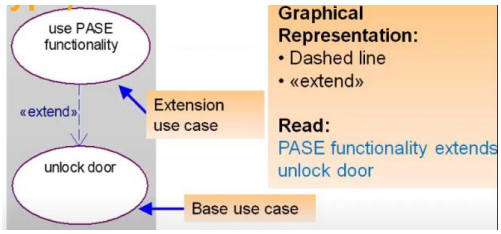
Son extensiones de modelos SysML o UML existentes, que crea un nuevo tipo de elemento basado en el original, y especifica el tipo o propósito del elemento modelado.

Include



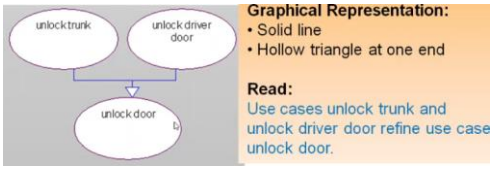
Dependencia(Relación) + Estereotipo(Include)
Un <<include>> de un caso de uso a otro indica que la instancia del primer caso de uso siempre contiene el comportamiento del segundo caso de uso

Extend



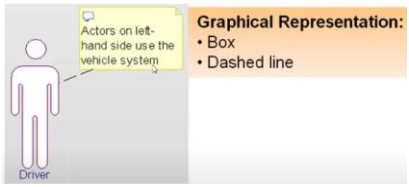
Dependencia(Relación) + Estereotipo(Extend)
Una relación de <<extend>> de un caso de uso a otro indica que el comportamiento del primer caso de uso se agrega de manera opcional al segundo caso de uso.

Generalización



Una relación entre un caso de uso general y caso de uso específicos, donde los casos específicos heredan las propiedades del caso de uso al que se están refiriendo.

Comentarios



Es una oración ligada al elemento modelado

Block definition Diagram/Internal block Diagram

Ambos diagramas son del tipo estructural, lo que se hace en un Internal Block Diagram se puede hacer en un Block Diagram, pero no viceversa. Son de los bloques más utilizados para definir el comportamiento estático dentro de la Arquitectura de Software.

Internal Block Diagram

Se utiliza para definir el comportamiento o la composición interna de un componente, habla de una estructura interna de un bloque, para ver como sus partes internas se comunican entre sí. Responde ¿Cuál es la jerarquía de los bloques y como se comunican o están interconectados?

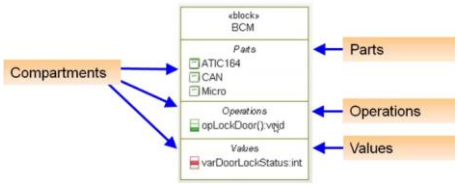
Block Definition Diagram

Abarca la descripción de cualquier tipo de bloque (parte, composición, bloque), es parte de los diagramas de SysML, en UML existe el class diagram, del cual hereda muchas propiedades el Block Definition Diagram. Ayudan a identificar la relación entre bloques, desde la jerarquía hasta la manera en la que se comunican, responde ¿Cuál es la estructura interna del bloque?
Los elementos son:

- Bloques y partes
- Asociaciones (Multiplicidad, dirección)
- Dependencias(<<Usage>>, <<flow>>)
- Rol
- Puerto y contratos, comunicación client-server (lollipop(servidor), Fork(cliente))

- Enlaces(Assembly y Delegation connectors)
- Interfaces(Operaciones, Variables, realization, usage)
- Herencia(<<Generalización>>, <<Agregación>>, <<Composición>>, <<Realización>>)

Bloque(Clase)



Es un rectángulo con diferentes apartados, normalmente lo que aparece son operaciones y valores o partes. Un bloque representa cualquier elemento del subsistema (Software componen, Funciones, actor, elemento externo del sistema). Tiene propiedades estructurales (Valores, partes, puertos).

Partes(Objeto)

Es la instancia de un bloque, es decir, un objeto que se ejecuta. Define la vista en el bloque en un punto especifico en el tiempo (snapshot)

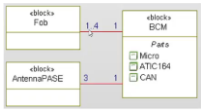
Operaciones

Métodos o funciones que implementan el software componen(Bloque). Una operación especifica una acción que tiene que ser ejecutada por el bloque/parte

Valores

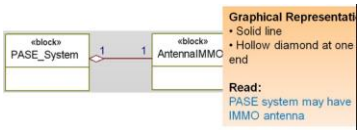
Variables o señales internas que maneja el bloque o software componen

Multiplicidad



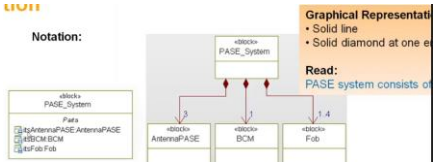
Especifica el rango de cardinalidad permitido que un conjunto puede asumir. Esencialmente es un subconjunto de números no negativos.

Agregación



Es una relación que expresa las composiciones: una parte representa el todo y consiste en ciertas partes. Todas las partes cobran sentido después del movimiento de agregación.

Composición

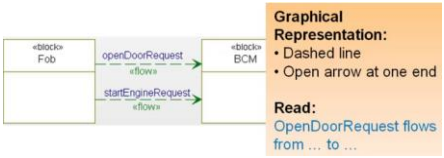


Representa los componentes que se tienen en el sistema, es una forma más fuerte de agregación, que requiere que se incluya una parte en al menos una composición a la vez. Si una composición se elimina, todas sus partes son borradas con ella.

Roles

Indica el tipo de acciones que puede ejecutar el bloque, al cual se le adjunta al final de la asociación. El nombre del rol es opcional.

Flow



Se utiliza para representar el flujo de información de un elemento a otro y tiene un sentido, es un elemento versátil, se puede utilizar para modelar información y objetos reales que son intercambiados entre dos elementos modelados.

Interfaces

Es el contrato que deben satisfacer ambas partes para que la información fluya de una manera correcta. En la interfaz se describe como se provee la información, cualquiera que quiera recibir la información que un bloque publica, debe de utilizar la misma interfaz. Si las partes involucradas no cumplen plenamente la interfaz, se tienen problemas de compatibilidad. Para SymML y UML una interfaz es un tipo de clase, puede tener atributos u operaciones, pero no un comportamiento interno.

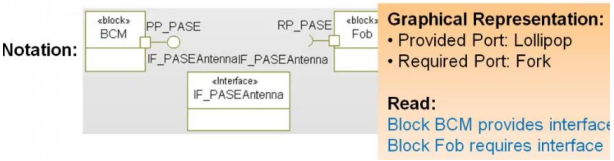
Realization

Este elemento modelado provee la interfaz.

Usage

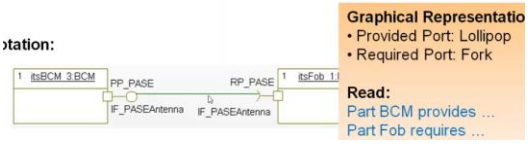
El bloque usado de (Fob), depende de la interfaz utilizada (IF_PASEAntenna), puede acceder al menos a 1 atributo u operación de la interfaz utilizada, puede acceder a todos los atributos u operaciones de la interfaz utilizada, en el sistema final, la funcionalidad de la interfaz es representada por un bloque.

Puertos y contratos



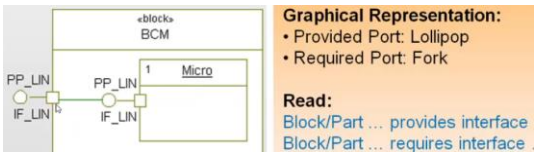
Los puertos describen un punto de interacción el cual es utilizado por las interfaces para proveer y/o solicitar servicios de acceso a la información, se utiliza normalmente en comunicaciones par a par (peer to peer), las cuales pueden ser síncronas(llamadas a funciones) o asíncronas(variables), no se recomienda el uso de variables compartidas. En resumen, el puerto es un punto especifico por el cual pasaremos la información, en un puerto se suscribe una interfaz.

Assembly conector



Entre bloques no puede utilizarse un Assembly conector, solo se puede implementar entre partes

Delegation conector



Cuando tienes un puerto dentro del bloque y quieres comunicarlo con una parte(exterior al bloque), es decir, hacer la comunicación hacia el exterior del bloque.

Activity Diagram

Un diagrama de actividad está basado en un diagrama de flujo, la única diferencia es que de lo que se encarga de describir una actividad de ahí se deriva el nombre.

Pertenece a ambos lenguajes de modelado SysML y UML, no tienen un comportamiento estático, sino más bien dinámico. Se encargan de mostrar como es el control de la información o de la secuencia de la actividad y cuál es el flujo de datos que existen dentro de la actividad, por donde se ejecuta la lógica y la información de los datos de dónde viene y en donde se generó.

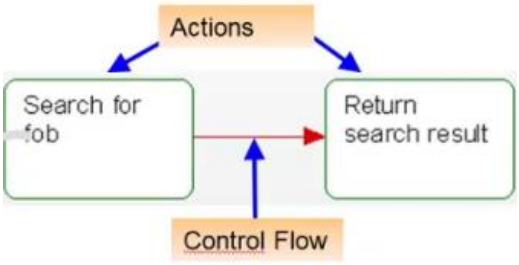
Estos flujos pueden ser secuenciales, paralelos o alternativos, la granularidad de una acción puede ir desde el rango de un sistema completo hasta llamar a una función en el lenguaje código. Ayuda a fragmentar las tareas en unidades más pequeñas de forma que sea más sencillo entender cómo lo vamos a implementar y que secuencias existen en cada una de ellas, responde las preguntas

¿Cómo es que mi sistema hace un comportamiento especial?, y ¿en qué orden ocurren estas actividades?

Elementos del diagrama son:

- Acción
- Estructura (reusó(actividad), responsabilidades y tareas(swimlanes))
- Flujos (inicial, control, final)
- Decisión y condiciones de Guardian
- Paralización(Fork y Join node)
- Sincronización(Sen y Receive signal action)

Action y Control Flow

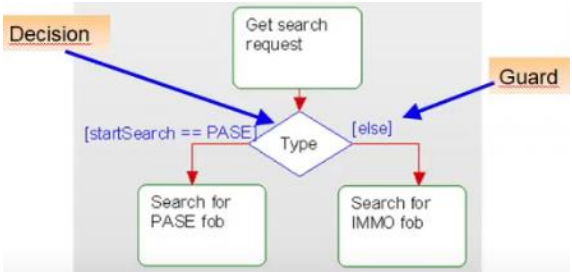


Acciones atómicas que se están ejecutando dentro de nuestra actividad, por atómico se entiende que no se puede dividir. Para enlazarlas tenemos el control flow que indica el flujo de la secuencia lógica.

Initial and Final States

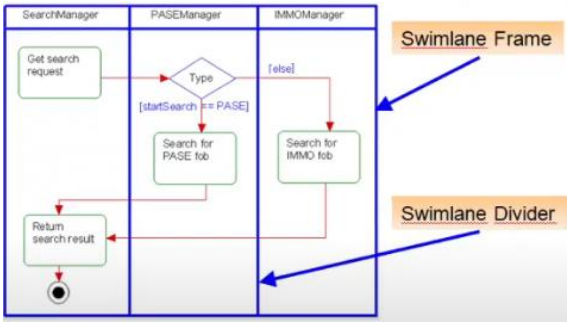
El estado inicial indica cual es el entry point de nuestra actividad, el estado final nos indica en donde termina nuestra actividad, solo se puede tener un estado inicial, pueden existir varios estados finales, pero convergen a uno solo en algún momento.

Decision and Guard Conditions



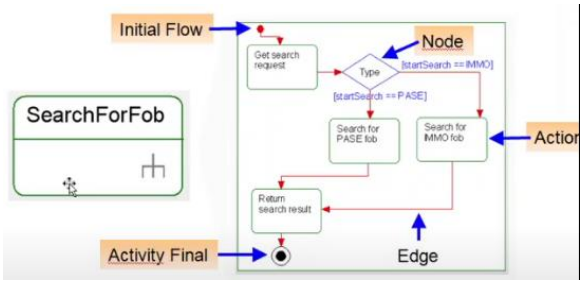
Las condiciones son los rombos o nodos, que no contiene la condición para tomar una ruta especifica. El Guard es el enunciado lógico que está descrito entre corchetes que nos indica si el camino a tomar dependiendo si se cumple o no una condición.

Swimlanes (carriles)



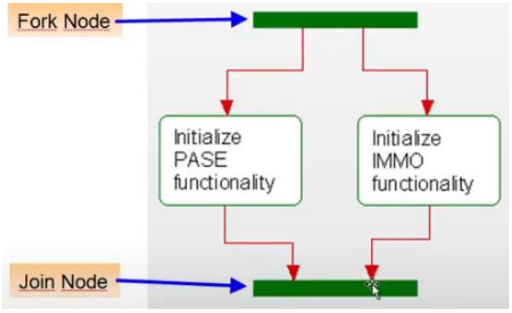
Los carriles asignan las tareas o las acciones que están ocurriendo dentro de nuestra actividad a un actor o a un elemento en especifico.

Activity



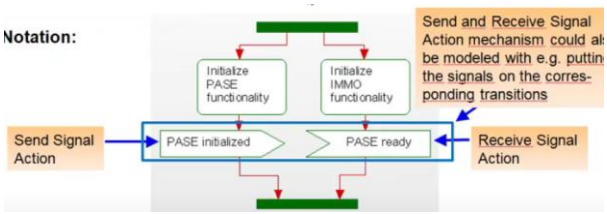
Es una conglomeración de acciones nodos y transiciones qué ocurre dentro de ella, tiene un flujo inicial y al menos 1 actividad final.

Fork Node and Join Node



Se utilizan para indicar procesos en paralelo o alternativas un fork node divide un flujo de control en flujos paralelos, a join node combina múltiples flujos de control en uno solo

Send and receive signal Action



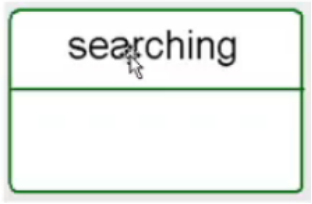
Son elementos de sincronización, send signal crea una señal y la trasmite a un objeto objetivo específico, el que manda la señal continua la ejecución inmediatamente sin esperar por ninguna respuesta.

State Machine Diagram

Son de los diagramas más utilizados cuando se diseñan tareas periódicas del sistema operativo, por que nos ayuda a saber la reacción de nuestro sistema a determinados eventos. Los diagramas de maquina de estado son parte de SysML y UML, y son del tipo diagramas de comportamiento(dinámico), que ayuda a describir la respuesta del sistema hacia un evento (síncrono, asíncrono, etc) que estimula la lógica de algún componente o tarea que se esté diseñando. Su función principal es describir la secuencia de estados y las transiciones que un objeto puede adquirir durante su tiempo de vida o parte de su tiempo de vida. Se utilizan para describir estados internos y la transición entre estos estados. Responden a la pregunta ¿Cómo un objeto reacciona a un evento? Los elementos diagrama son:

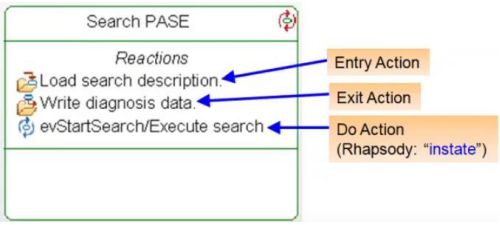
- Estados (Actions on Entry / Do / Exit)
- Transiciones (Default, Null)
- Tipos de evento / Guardianes y Efectos
- Ramificación (Conditions, Fork/Join Sync Bars)
- Procesamiento Paralelo (Fork/Join Sync Bars, Regions)
- Jerarquía (Region, Composite State)
- Estados Finales (Final, Terminación)
- Historia de Estados

States



Modela una situación con condiciones bien definidas en el ciclo de vida del objeto, una maquina de estado describe todos los posibles estados y las transiciones entre ellos. Un estado puede ser una situación estática esperando por inputs o por un evento que estimule el flujo de información, o dinámica donde el sistema ejecuta un proceso o tarea. El estado activo es el estado actual de la maquina de estados.

Actions



Entry

Esta acción opcional se ejecutará al principio del estado y termina antes de hacer cualquier otra cosa.

Do

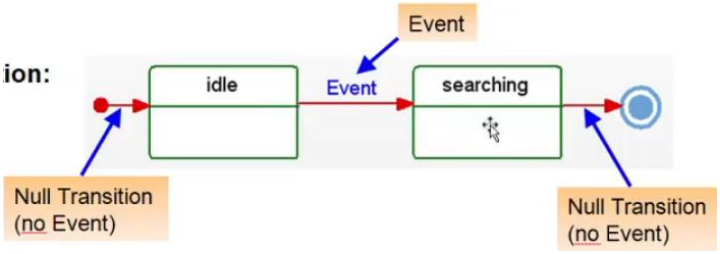
Esta acción opcional comienza después de que el entry action finaliza y sigue ejecutándose hasta que se termine por si misma o el estado cambie

Exit

Después de terminar cualquier acción *entry* o *do* esta acción opcional será ejecutada antes de dejar o salir del estado.

Transiciones

Es una asociación directa entre 2 estados, si un evento ocurre el estado activo es dejado y una transición a otro estado será ejecutada, cualquier evento que no pueda ser manejado, es decir, que no sea esperado, no se almacena, reenvía ni invoca ninguna acción.



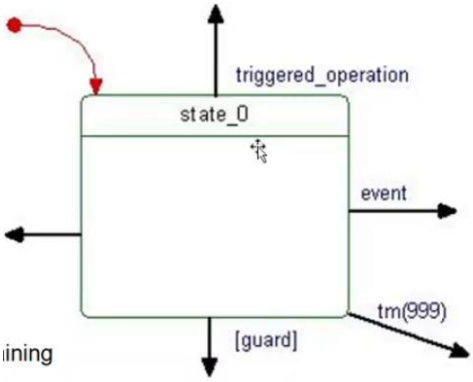
Default

Asigna un punto de partida o inicial de la máquina de estados.

Termination

Es el final de una maquina de estados, esta transición puede detener regiones de estados, subestados y máquinas de estados.

Event types



Triggered Operation (call event)

Una llamada a una operación

Event (Signal Event)

Recibimiento de una señal

Tm (xy, time event)

Un tiempo dado

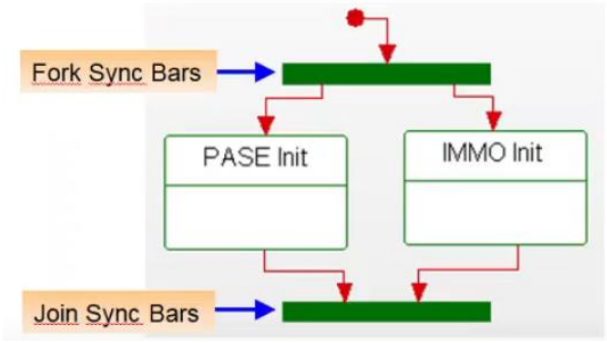
Guard (change event)

Contiene una expresión booleana

Effects

El efecto define una acción opcional que es ejecutada durante una transición, y será invocada directamente en el objeto que tiene la maquina de estados como resultado de la transición.

Branching (either or)



Los conectores de condición son usados para distribuir transiciones entrantes a múltiples estados. Los conectores JOIN son usados para unir múltiples transiciones entrantes.

Fork

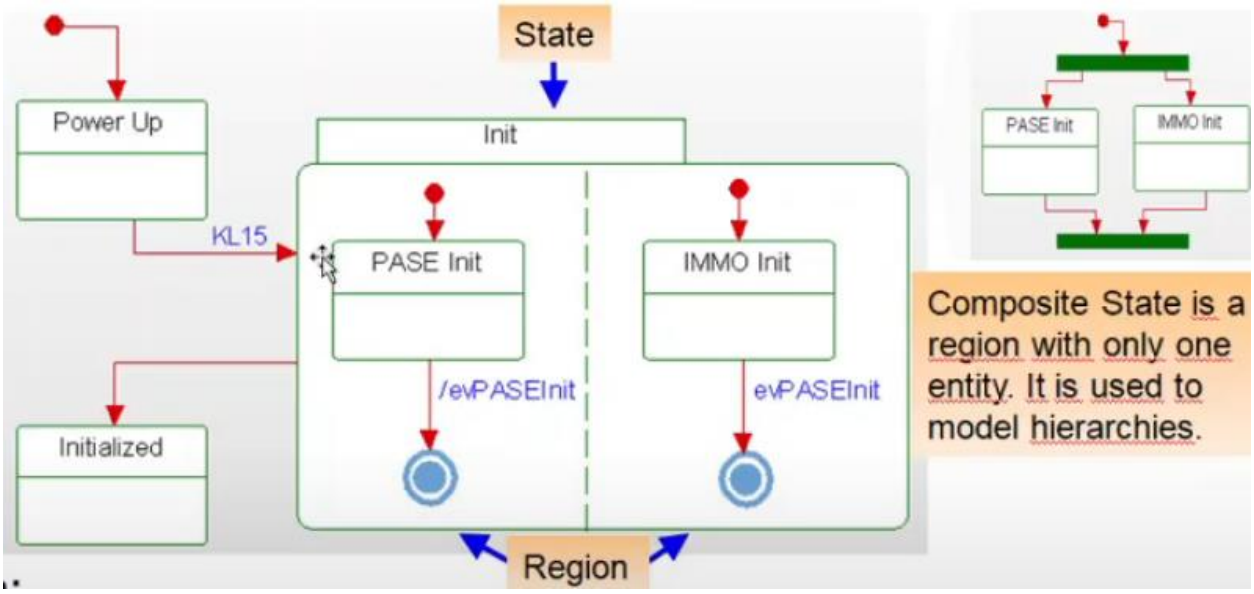
Fork sync bars son usadas para dividir transiciones entrantes en múltiples estados.

Join

Join sync bars son usadas para unir transiciones entrantes.

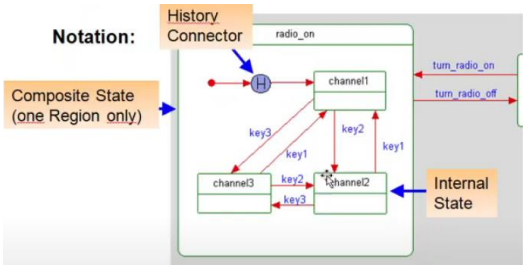
Parallel processing

Regions



Divide un estado en 2 o más entidades disyuntivas procesadas paralelamente, no puede existir una transición entre dos regiones.

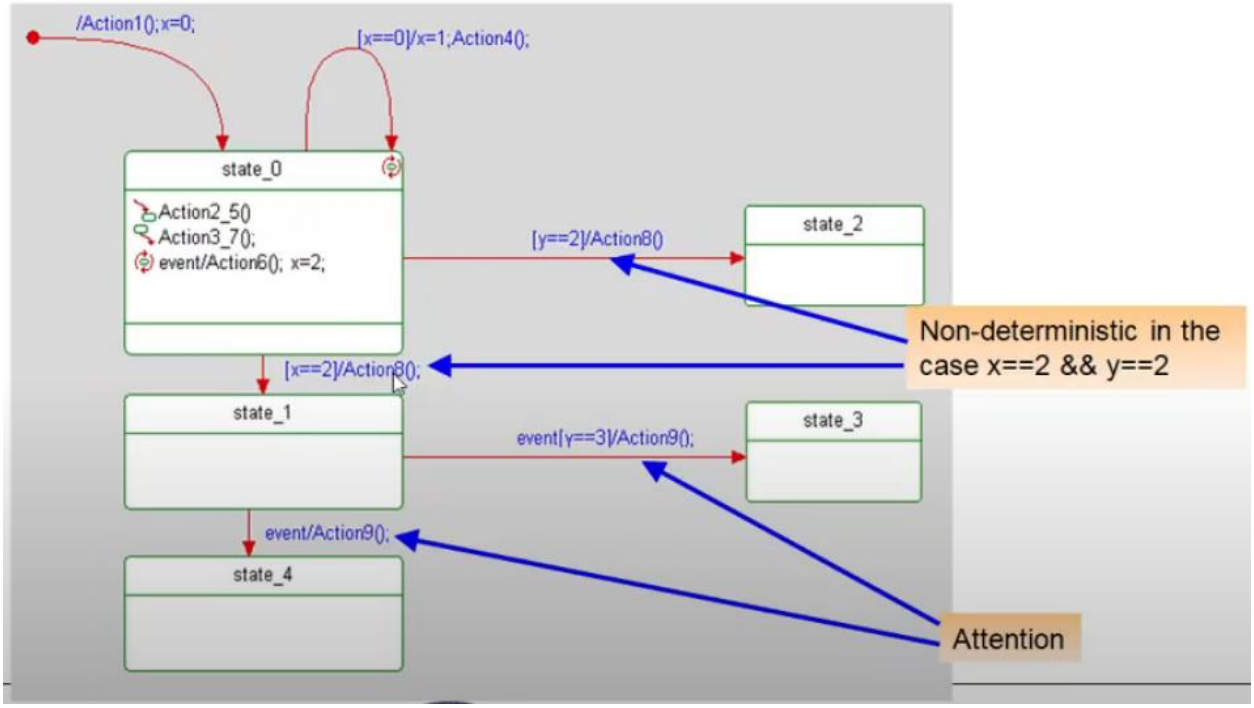
History Connector



Las señales del history connector son reestablecidas del ultimo estado interno cuando entra el estado de composición la próxima vez. Ayuda a tener "memoria", es decir, cuando vuelvas a entrar el default transition te mandara a donde te quedaste la ultima vez.

Action order

La ejecución se basa en las condiciones, el evento y el Guard.



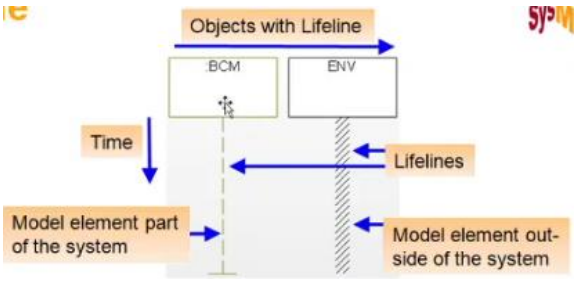
Sequence Diagram

Son parte de SysML y UML, es del tipo de diagrama que describe comportamientos (dinámicos), donde se prioriza su secuencia, es decir, el orden de ejecución de los eventos y como fluye la información entre elementos. El tiempo no es lo primordial en este tipo de diagramas, responden la pregunta ¿Cuándo y cómo quién llama a quién?

Los elementos de este diagrama son:

- Objetos (Lifeline, Destruction Event)
- Estado Invariante
- Activación y llamadas de Operaciones
- Time out / Time interval
- Referencias
- Fragmentos combinados (Alternative, Option, Loop, Break, Parallel Section, Critical, Section)

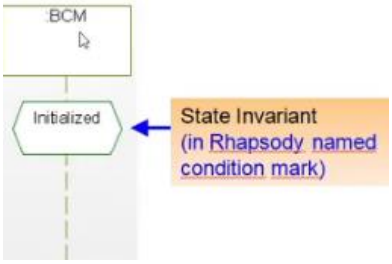
Objets with lifeline



La línea con franja y líneas diagonales representa elementos que están fuera de nuestro sistema, no pertenece al contexto que describimos pero que participa en el sistema. La línea punteada se utiliza para describir los elementos que si pertenecen al sistema, el tiempo transcurre de arriba hacia abajo.

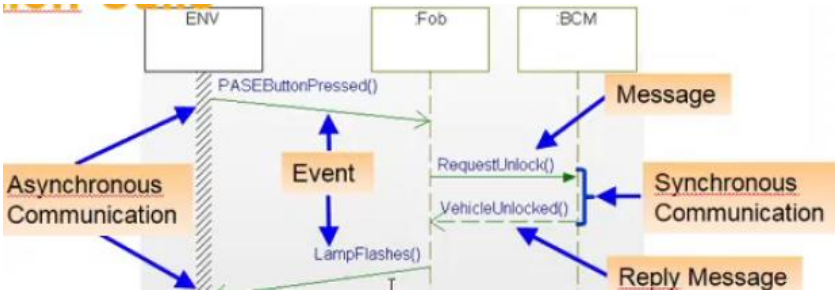
Un objeto denota una pareja de interacción, su línea de vida denota su existencia o duración, el nombre del objeto es opcional, el tipo de objeto es obligatorio. La línea de vida de un objeto representa su existencia, si el objeto es destruido o creado durante el periodo mostrado por el diagrama, la línea de vida para y continua en los puntos apropiados.

State invariant



Especifica la condición requerida del objeto antes de las siguientes interacciones.

Activation and Operation Calls

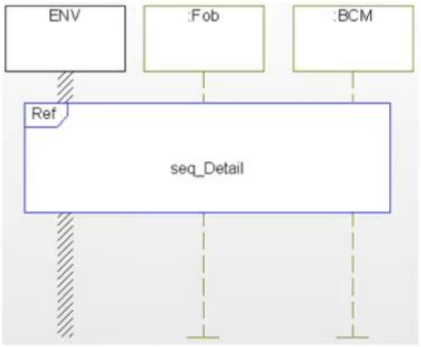


Una operación de llamada define una interacción o comunicación síncrona.

Time out / Time Interval

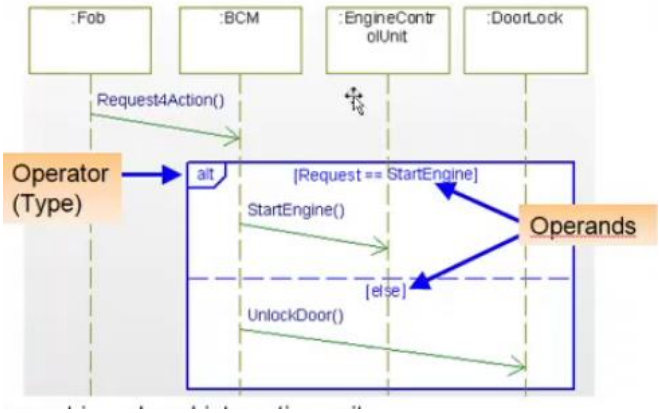
El time out define un mensaje que es utilizado después de que el haya transcurrido, la notación es "Tm (time in milliseconds)", un intervalo de tiempo es una notación vertical que muestra cuanto tiempo (real) ha pasado entre 2 puntos en la línea de vida. Time out y time Interval son siempre mensajes para el propio objeto emisor.

Reference



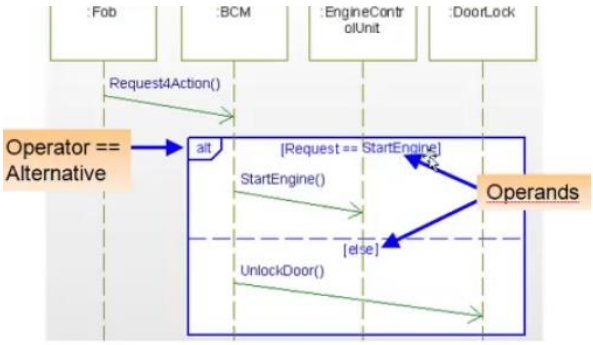
Es un marco de interacción que referencia otro diagrama de interacción

Combined Fragments



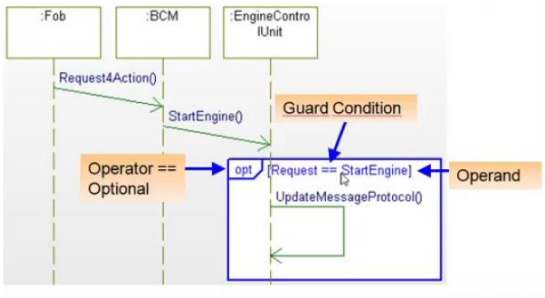
Pueden hacer diferentes tipos de operaciones, es una unidad de interacción cerrada, existen varios tipos (operadores de interacción) que tienen 1 o más operandos de interacción.

Alternative



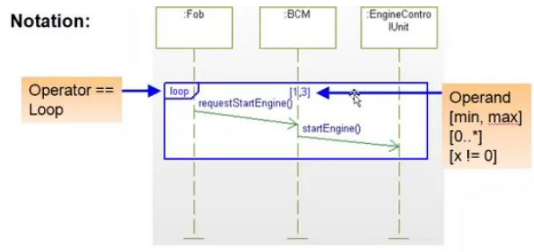
Ofrece múltiples operandos, pero solo 1 será ejecutado. Se tiene una línea divisora y se indica la condición para que la secuencia ocurra, si no se cumple se ejecuta la otra secuencia. Se puede hacer la analogía con un If-Else

Option



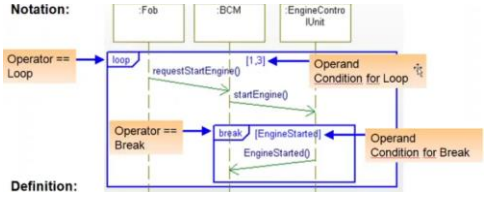
A diferencia de Alternative, Option solo tiene un operando, se puede hacer la analogía con un If. Todos los operandos se ejecutaran solo si la condición Guard es verdadera.

Loop



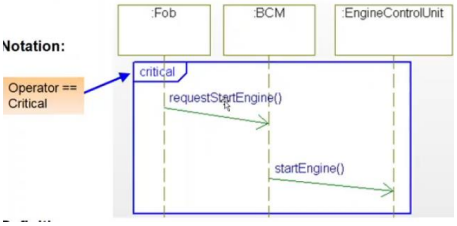
Especifica un numero de ejecuciones repetidas del operando. Se puede hacer la analogía con un for o while.

Break



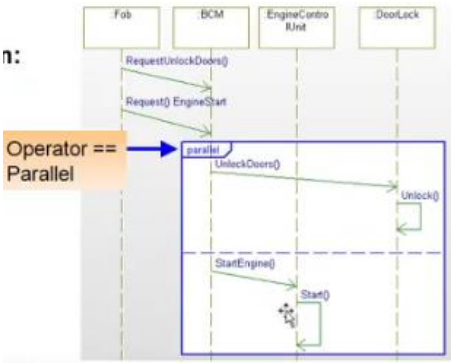
Se utiliza para romper un loop, ya que define una interrupción a una condición para la interacción cerrada.

Critical Section



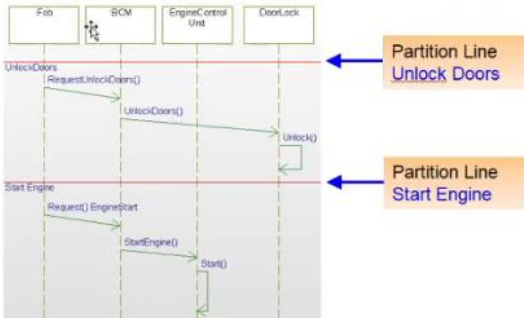
Define una región que no debe de ser interrumpida, se refiere a la ejecución de operación atómica (que no se divide).

Parallel Section



Todos los operandos de la interacción paralela serán ejecutados en paralelo. Se indican secuencias que ocurren al mismo tiempo.

Partition line



Es un elemento grafico y no tiene representación en el modelo.