SysML

The Systems Modeling Language (SysML) is a standardized modeling language designed to support the modeling of complex systems that integrate hardware, SW, information, processes, and more. It was developed as a modification of UML to suit the needs of systems engineering.

SysML is useful for designing and analyzing multidisciplinary systems, providing a common language for engineers from different domains. It allows to manage complexity, improve communication between teams and ensure that all aspects of the system are considered.

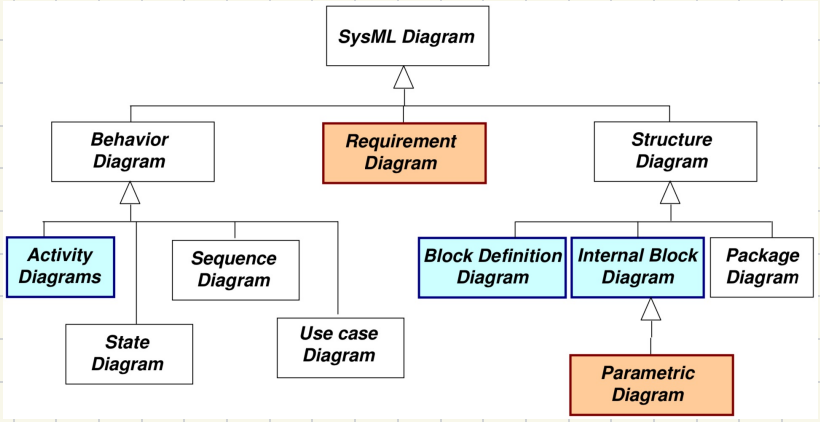
SysML vs UML

While UML focuses primarily on SW dev, SysML is intended for modeling general purpose systems, including hardware and other domains.

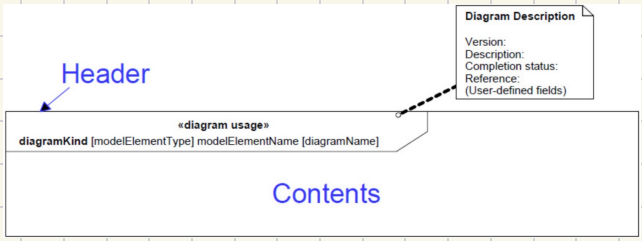
SysML introduces new diagrams and concepts compared to UML, such as requirements diagrams, to support systems engineering.

SysML doesn't include all the details of UML that are specific to the SW, making it more streamlined and geared towards complex systems.

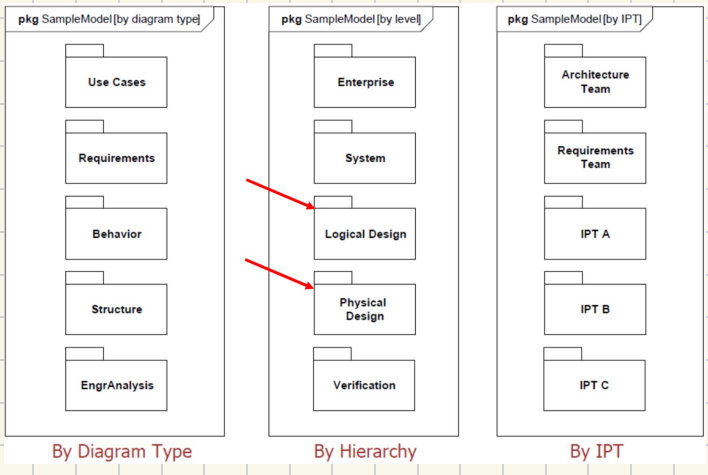
Available Diagrams

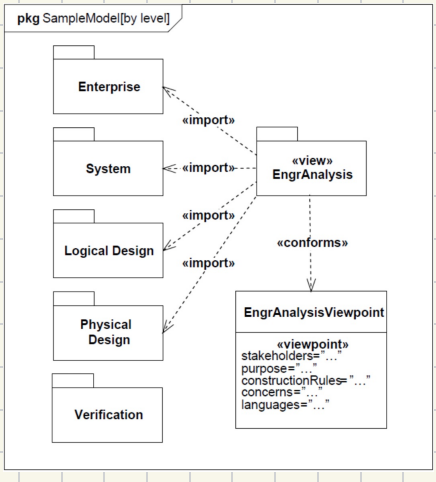


Each SysML diagram must have a diagram frame

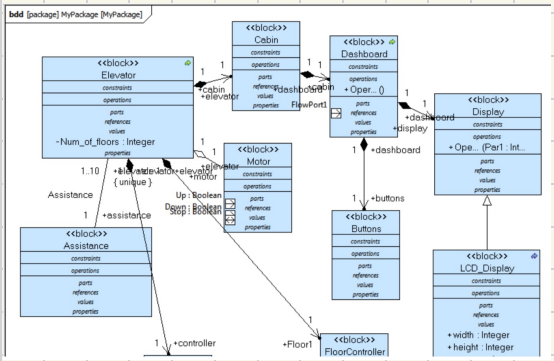


- **Structure Diagrams:**
  - **Package Diagram**: organize templates into packages for easy management.



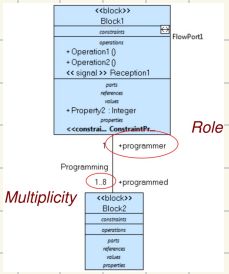


- **BLOCK DEFINITION DIAGRAM (BDD)**: IS USED TO DEFINE BLOCKS AND DESCRIBES THE MAIN COMPONENTS OF A SYSTEM AND THEIR RELATIONSHIPS. IT IS SIMILIAR TO THE CLASS DIAGRAM IN UML.

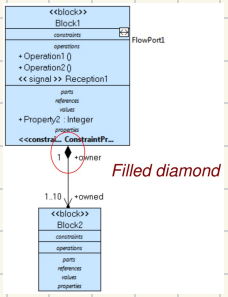


RELATIONSHIPS.

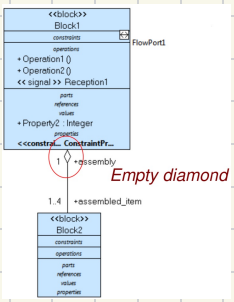
GENERIL ASSOCIATION: THE TWO BLOCKS COOPERATE IN SOME WAY.



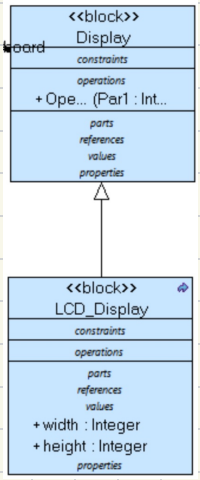
COMPOSITION: THE COMPONENT BLOCKS CAN ONLY EXIST IN THE CONTEXT OF THE OWNER COMPOSITE BLOCK.



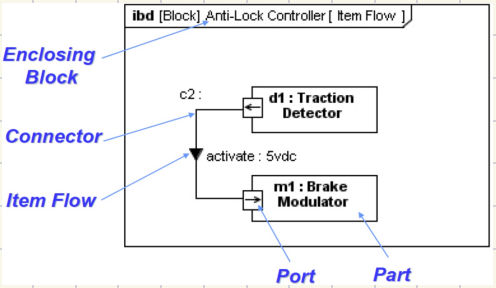
AGGREGATION THE COMPOSITE CONTAINS THE COMPONENTS BUT THE COMPONENTS CAN EXIST OUTSIDE THE COMPOSITE.



GENERALIZATION: THE SPECIALIZED BLOCK HAS ALL THE PROPERTIES/OPERATIONS/... OF THE GENERIC OBJECT BUT CAN ADD SOME OF ITS OWN.



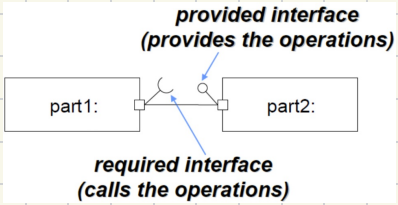
- **INTERNAL BLOCK DIAGRAM (IBD):** DEFINES THE USE OF BLOCKS IN A COMPOSITION AND SHOWS HOW THE SYSTEM COMPONENTS INTERACT INTERNALLY.



SYSML PORT: SPECIFY INTERACTION POINTS ON BLOCKS AND PARTS

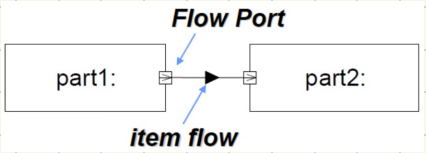
STANDARD (UML) PORT: SPECIFIES A SET OF REQUIRED OR PROVIDED OPERATIONS AND/OR SIGNALS.

FOR SW COMPONENTS

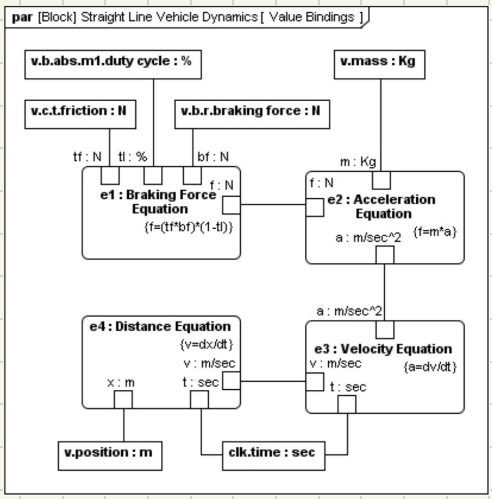


FLOW PORT: SPECIFIES WHAT CAN FLOW IN OR OUT OF BLOCK/PART.

FOR SIGNALS AND PHYSICAL FLOWS

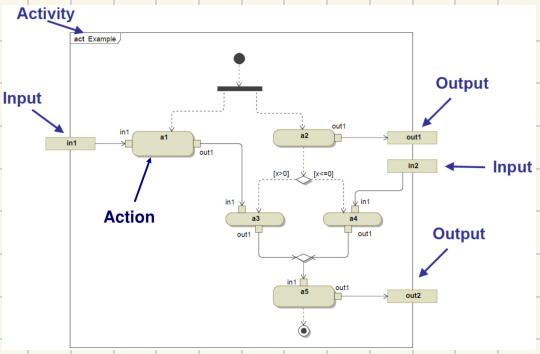


- **PARAMETRIC DIAGRAMS:** USED TO MODEL MATHEMATICAL RELATIONSHIPS OR CONSTRAINTS THAT GOVERN SYSTEM BEHAVIOR. HELPS CALCULATE VALUES BASED ON DEFINED PARAMETERS.

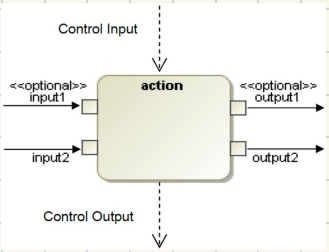


- **BEHAVIOR DIAGRAMS:**

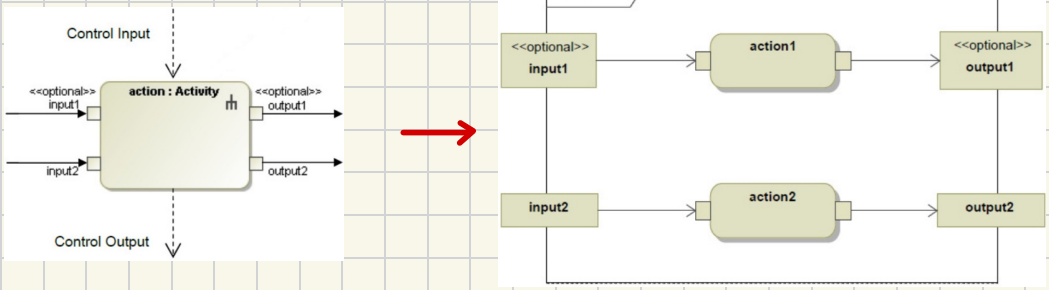
  - **ACTIVITY DIAGRAMS:** ACTIVITY SPECIFIES TRANSFORMATION OF INPUTS TO OUTPUTS THROUGH A CONTROLLED SEQUENCE OF ACTIONS.



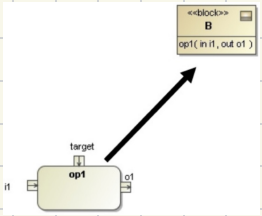
ACTIONS: UNIT OF FLOW IS CALLED TOKEN (CONSUMED AND PRODUCED BY ACTIONS). ACTIONS EXECUTION BEGINS WHEN TOKENS ARE AVAILABLE ON ALL CONTROL INPUTS AND REQUIRED INPUTS (UNLESS LABELED AS "OPTIONAL").



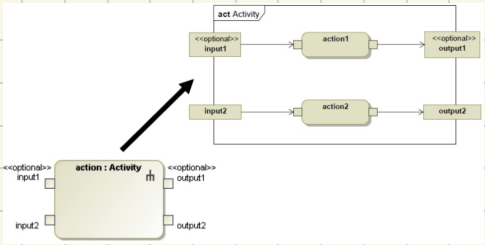
AN ENTIRE ACTIVITY IS USED TO SPECIFY THE ACTION BEHAVIOR AND IT'S INVOKED WHEN THE PARENT ACTION BEGINS EXECUTION.



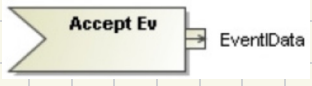
COMMON ACTION



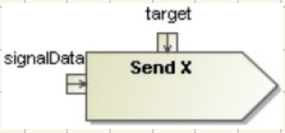
CALL OPERATION ACTION



CALL BEHAVIOR ACTION



ACCEPT EVENT ACTION



SEND SIGNAL ACTION

ACTION TYPES AND BEHAVIOR:

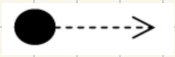
i **STARTING AN ACTION.** AN ACTION STARTS WHEN A TOKEN IS PLACED ON ALL OF ITS CONTROL INPUTS AND REQUIRED INPUTS AND THE PREVIOUS ACTIVITY HAS COMPLETED.

ii **DURING AN EXECUTION:** AN ACTION CONTINUES TO ACCEPT STREAMING INPUTS AND PRODUCE STREAMING OUTPUTS.

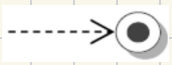
iii **TERMINATING AN ACTION:** AN ACTION TERMINATES WHEN ITS INVOKED ACTIVITY REACHES AN ACTIVITY FINAL, OR WHEN THE ACTION RECEIVES A CONTROL DISABLE. THE TOKENS ON THE OUTPUT PARAMETER NODES OF THE ACTIVITY ARE PLACED ON THE OUTPUT PINS OF THE ACTION AND A CONTROL TOKEN IS PLACED ON EACH OF THE CONTROL OUTPUTS OF THE ACTION.

iv **FOLLOWING ACTION TERMINATION:** THE TOKENS ON THE OUTPUT PINS AND CONTROL OUTPUTS OF THE ACTION ARE MOVED TO THE INPUT PINS OF THE NEXT ACTIONS WHEN THEY ARE READY TO START PER ABOVE.

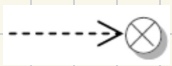
NOTATION:



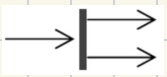
INITIAL NODE: ON EXECUTION OF PARENT CONTROL TOKEN PLACED ON OUTGOING CONTROL FLOWS.



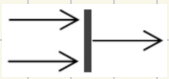
ACTIVITY FINAL NODE: RECEIPT OF A CONTROL TOKEN TERMINATES PARENT.



FLOW FINAL NODE: SINK FOR CONTROL TOKEN.

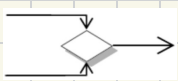


FORK NODE: DUPLICATES INPUT TOKENS FROM ITS INPUT FLOW ONTO ALL OUTGOING FLOWS.



JOIN NODE: WAITS FOR AN INPUT TOKEN ON ALL INPUT FLOWS AND THEN PLACES THEM ALL ON THE OUTGOING FLOW.

DECISION NODE: WAITS FOR AN INPUT TOKEN ON ITS INPUT FLOW AND PLACES IT ON ONE OUTGOING FLOW BASED ON GUARDS



MERGE NODE: WAITS FOR AN INPUT TOKEN ON ANY INPUT FLOWS AND THEN PLACES IT ON THE OUTGOING FLOW.

- **SEQUENCE DIAGRAM**: IT SHOWS THE TEMPORAL INTERACTION BETWEEN PARTS OF THE SYSTEM, HIGHLIGHTING THE ORDER OF MESSAGES EXCHANGED.

FRAGMENT TYPES:

SEQ: EACH LIFELINE MAY SEE DIFFERENT ORDERS FOR THE EXCHANGE.

STRICT: THE MESSAGE EXCHANGE OCCURS IN THE ORDER DESCRIBED.

CRITICAL: THE SEQUENCE DIAGRAM FRAGMENT IS A CRITICAL REGION.

NEG: THE SEQUENCE DIAGRAM FRAGMENT IS FORBIDDEN.

ASSERT: THE SEQUENCE DIAGRAM FRAGMENT IS THE ONLY ONE POSSIBLE.

REF: NAME REFERENCE TO A SEQUENCE DIAGRAM FRAGMENT.

OPT: HAS 1 PART THAT MAY BE EXECUTED BASED ON A CONDITION.

ALT: HAS 2 OR MORE PARTS, BUT ONLY 1 EXECUTES BASED ON A CONDITION

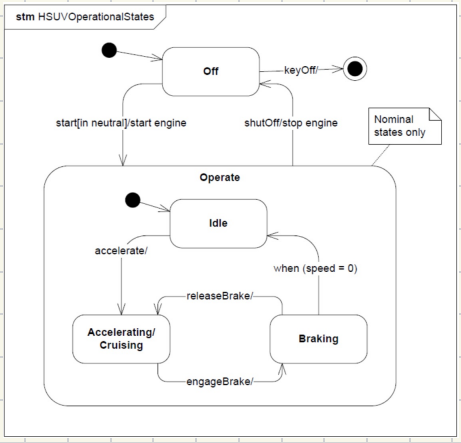
PAR: HAS 2 OR MORE PARTS THAT EXECUTE CONCURRENTLY.

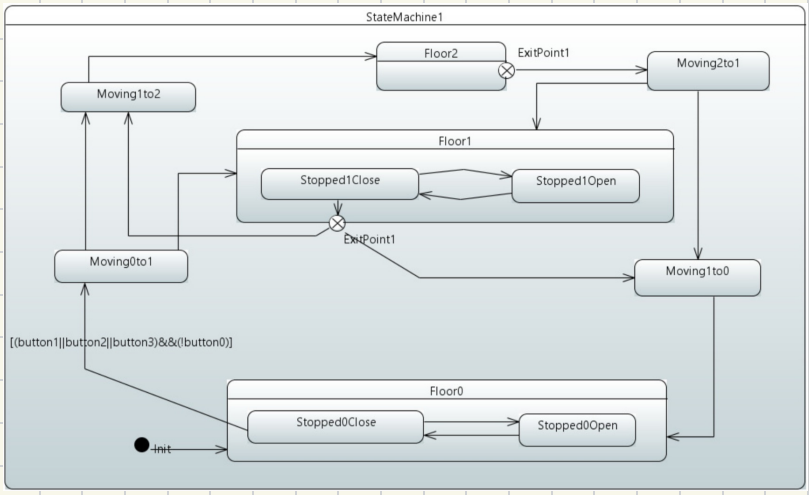
LOOP: HAS A MINIMUM # OF EXECUTIONS, AND OPTIMAL MAXIMUM # OF EXECUTIONS, AND OPTIMAL ESCAPE CONDITION.

CONSIDER: MESSAGES THAT ARE RELEVANT IN THIS SEQUENCE FRAGMENT.

IGNORE: MESSAGES THAT MAY ARRIVE, BUT ARE NOT INTERESTING HERE

- **STATE DIAGRAM**: USED TO REPRESENT THE LIFE CYCLE OF A BLOCK.





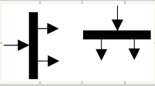
ELEMENTS:

FINAL STATE: A PSEUDO-STATE SIGNIFYING EITHER THE LEAVING STATE FOR AN OBJECT OR THE TERMINATION OF THE ENCLOSING REGION.

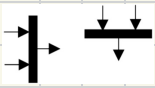
ENTRY POINT: A REFERENCE FOR THE TARGET OF A TRANSITION.

EXIT POINT. A REFERENCE AS THE SOURCE OF A TRANSITION.

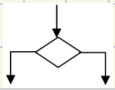
CONNECTION POINT REFERENCE: RAPRESENT ENTRIES INTO OR EXITS OUT OF THE SUBMACHINE REFERENCED BY THE SUPERSTATE.



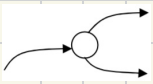
FORK. SPLITS AN INCOMING TRANSITION INTO TWO OR MORE TRANSITIONS TERMINATING ON ORTHOGONAL TARGET VERTICES.



JOIN: MERGE TRANSITIONS EMANATING FROM SOURCE VERTICES IN DIFFERENT ORTHOGONAL REGIONS.



CHOICE POINT: SPLIT TRANSITIONS PATHS.



JUNCTION: CHAIN TOGETHER MULTIPLE TRANSITIONS.

- **USE CASE DIAGRAM:** IDENTIFY USE CASES OF THE SYSTEM, SHOWING HOW ACTORS INTERACT WITH IT.

- **REQUIREMENT DIAGRAM.** DEFINES, TRACKS AND VERIFIES SYSTEM REQUIREMENTS. IT ALLOWS TO CONNECT REQUIREMENTS WITH THE COMPONENTS THAT SATISFY THEM.