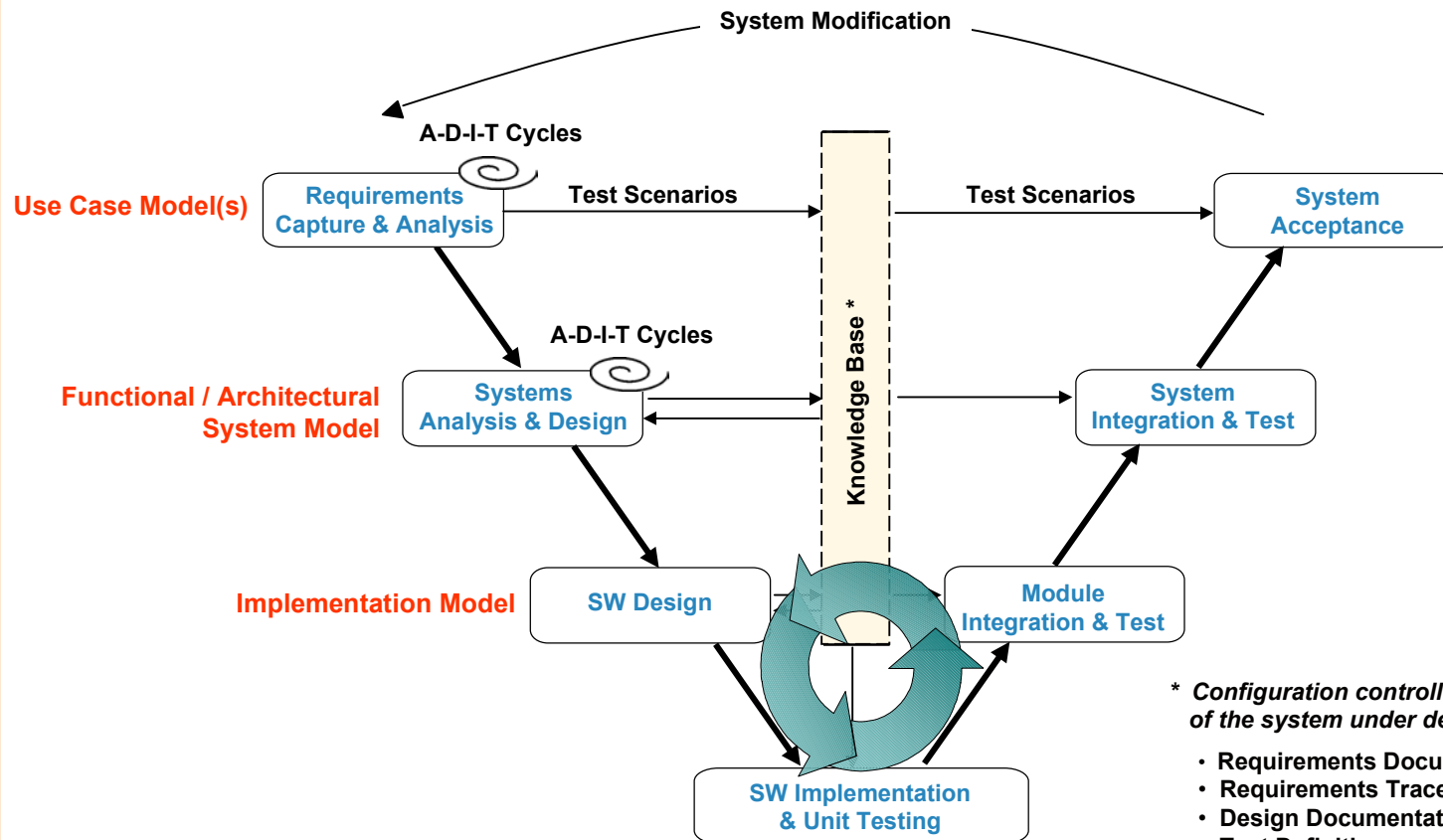
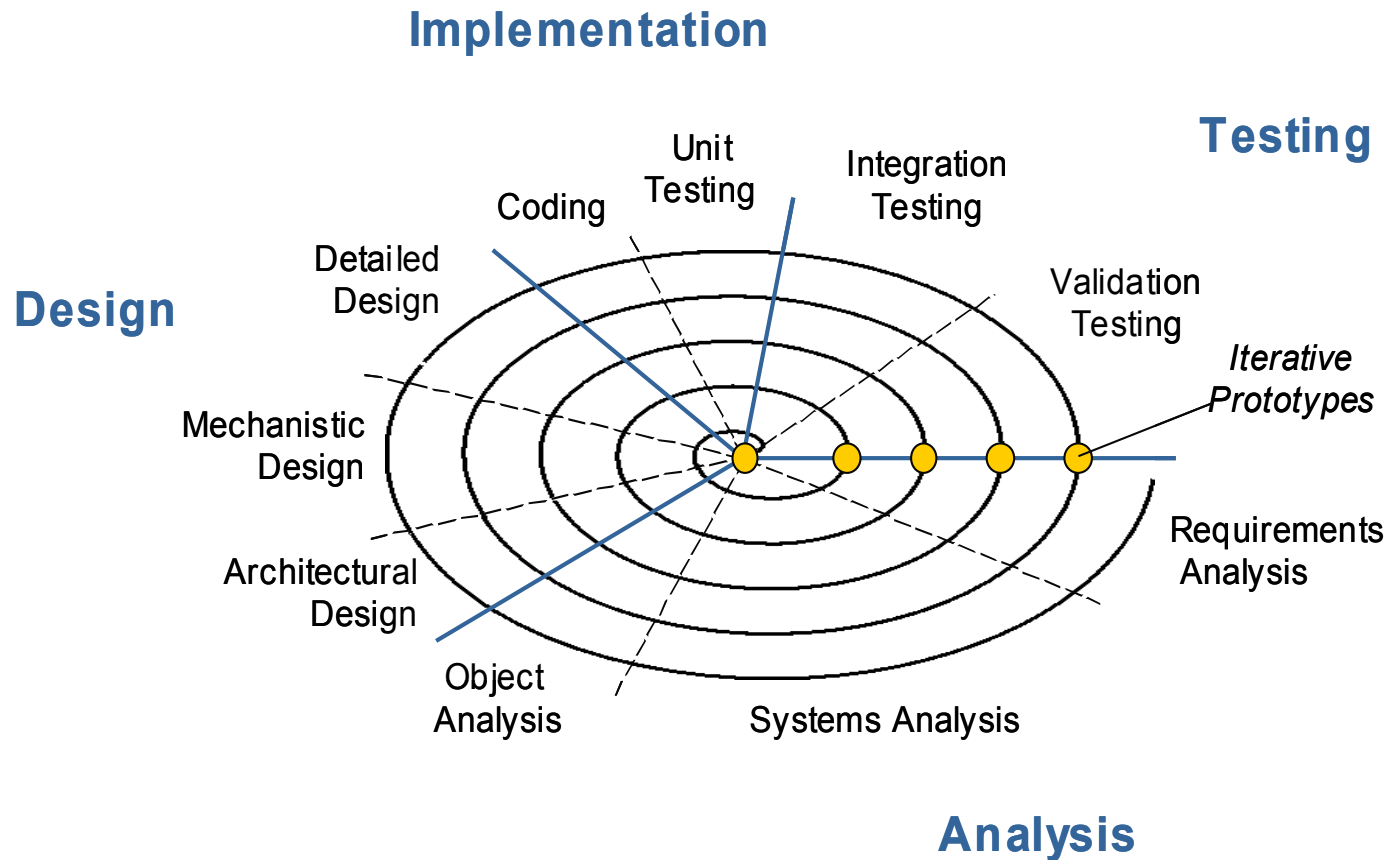


Classical V-Diagram of the Engineering Lifecycle in a Model based Development

(A)-D-I-T Cycles = (Analysis)-Design-Implementation-Test cycles including iterative prototypes

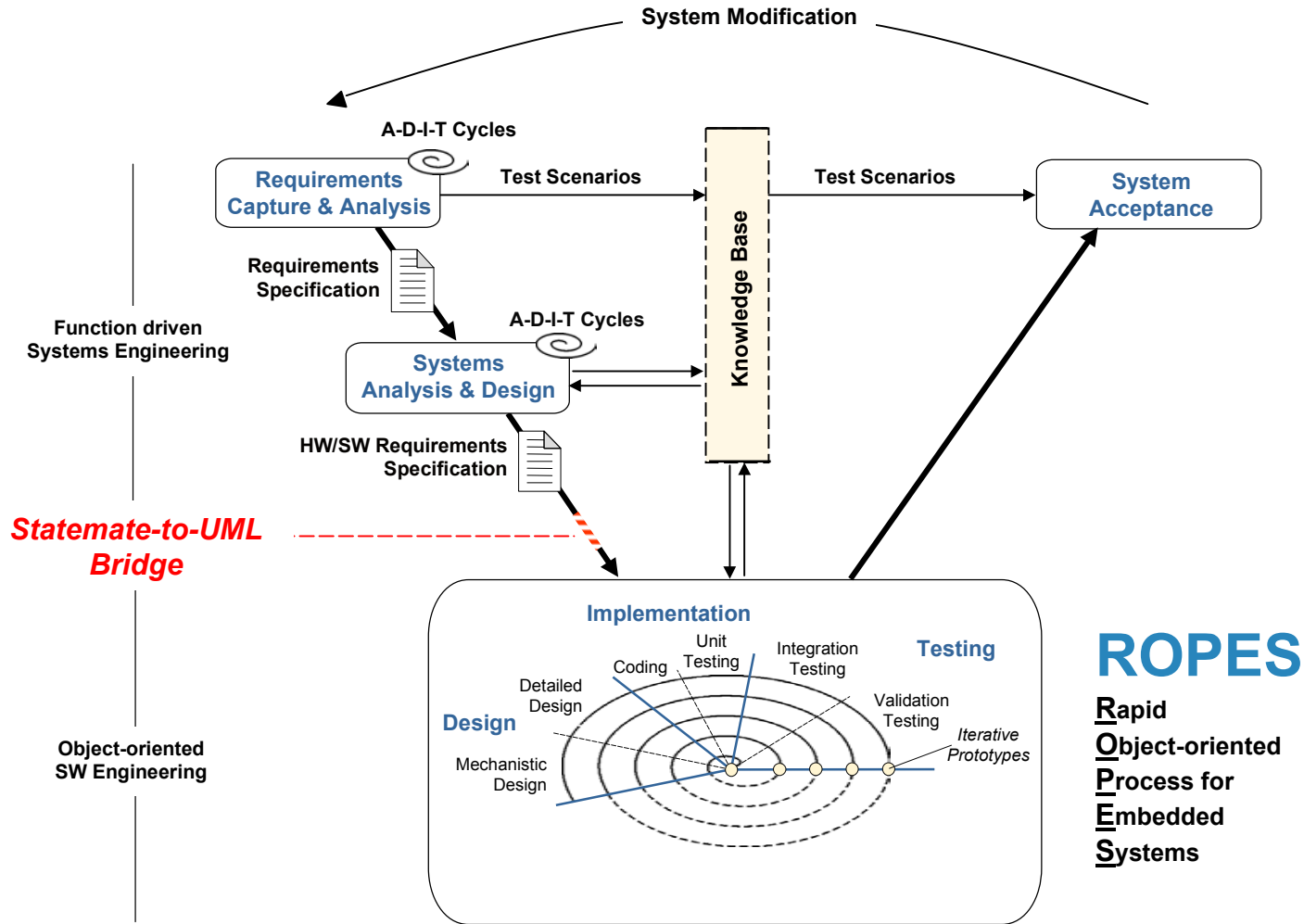


Rapid Object-oriented Process for EMBEDDED Systems (ROPES) [Bruce Douglass]

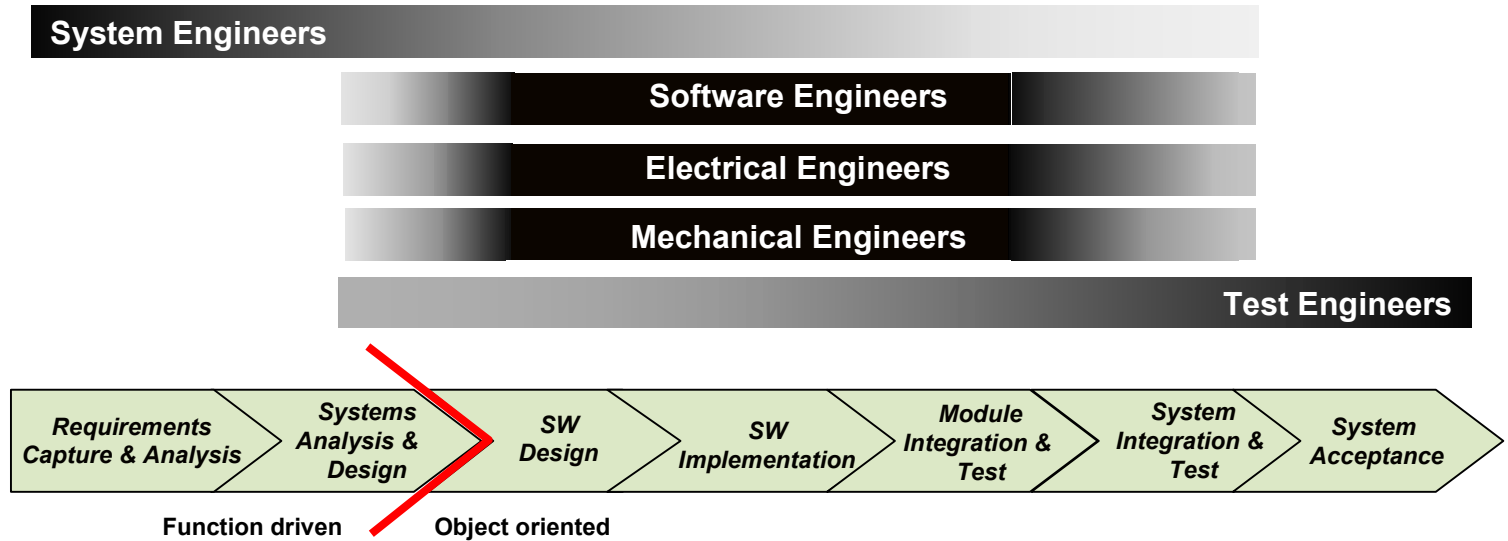


The I-Logix "Integrated Process"

From function driven Systems Engineering to object-oriented SW Engineering



Concurrent Engineering in the Integrated Process



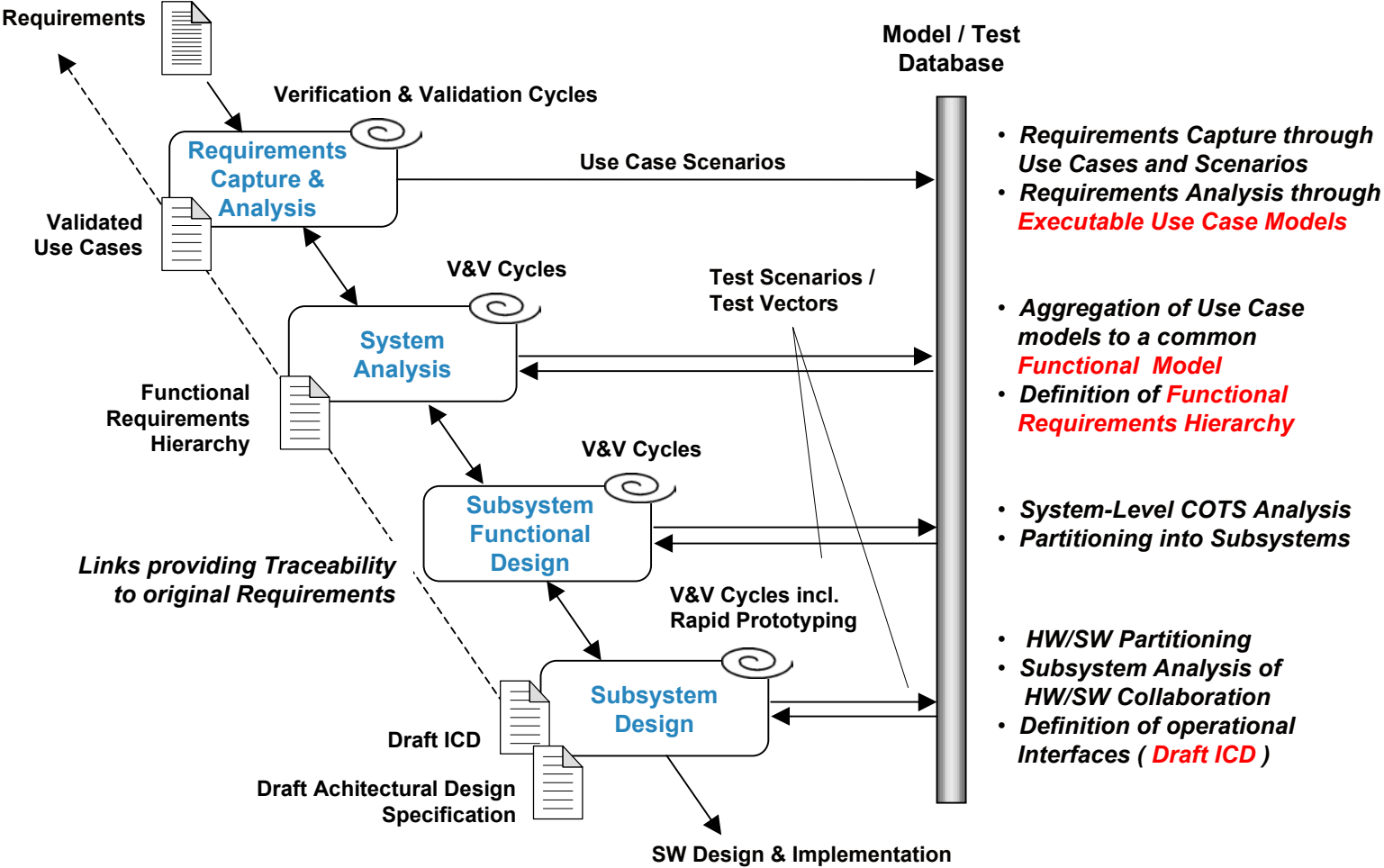
Note:

System architectural design is a *concurrent engineering* task.

Especially in regards to the SW part, the subsystem and subsystem-component structure of the architectural model needs to be approved by SW developers.

The agreed upon structure should then become the mandate for any further development.

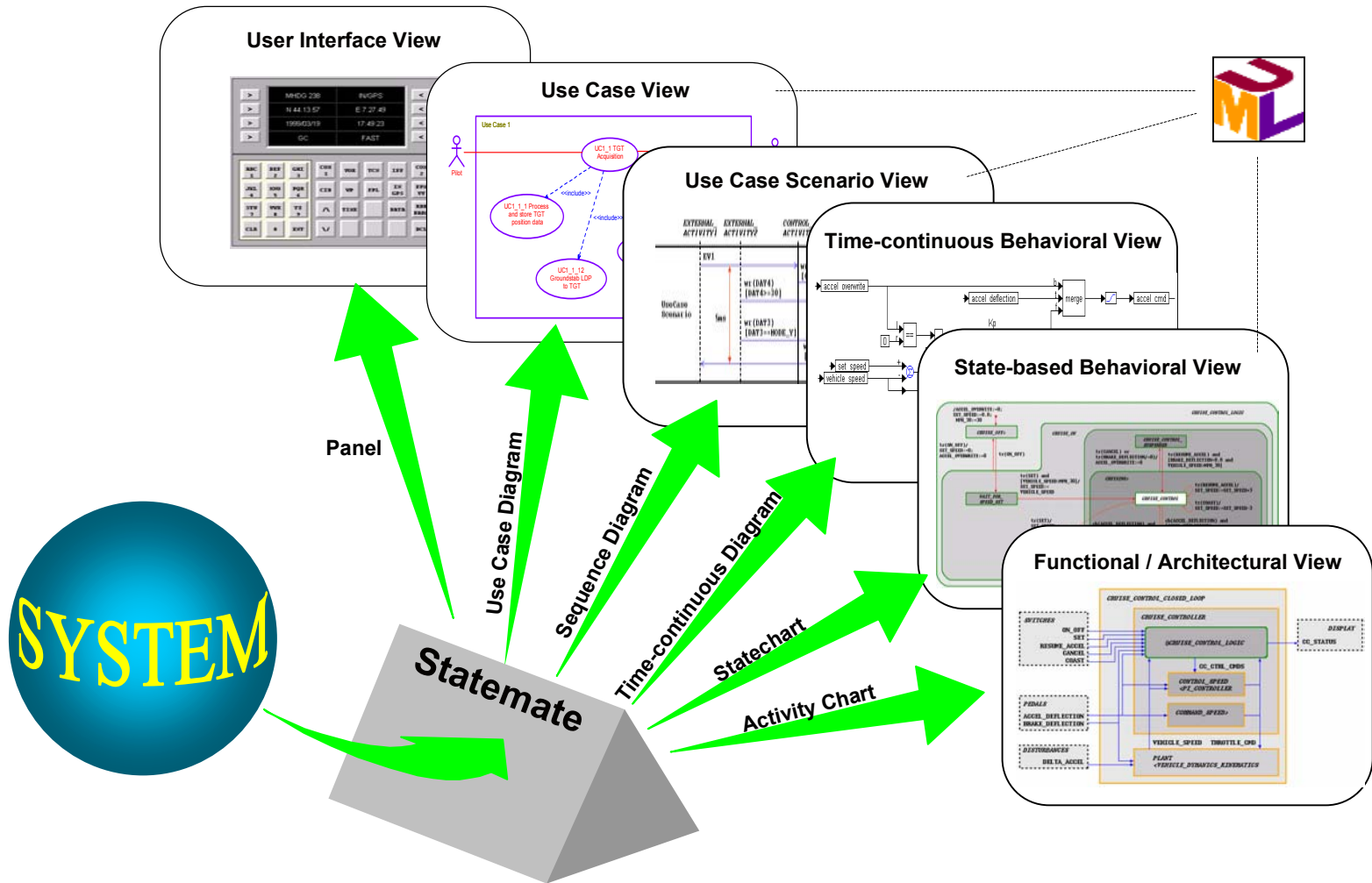
Function driven Systems Engineering



- Requirements Capture through Use Cases and Scenarios
- Requirements Analysis through Executable Use Case Models
- Aggregation of Use Case models to a common Functional Model
- Definition of Functional Requirements Hierarchy
- System-Level COTS Analysis
- Partitioning into Subsystems
- HW/SW Partitioning
- Subsystem Analysis of HW/SW Collaboration
- Definition of operational Interfaces (Draft ICD)

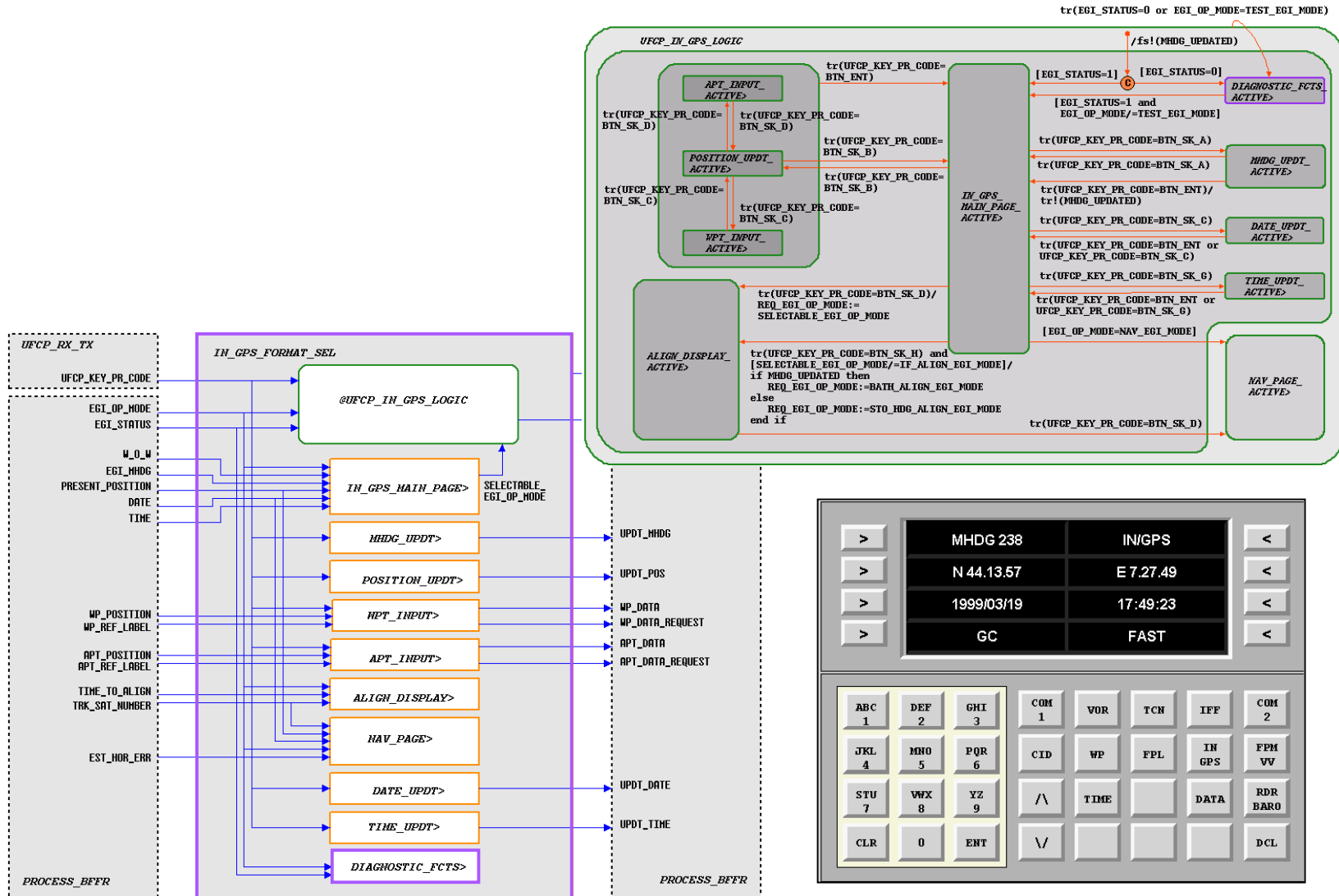
Modeling Languages for Systems Engineering (*Statestate*TM)

The different "Views" to a System



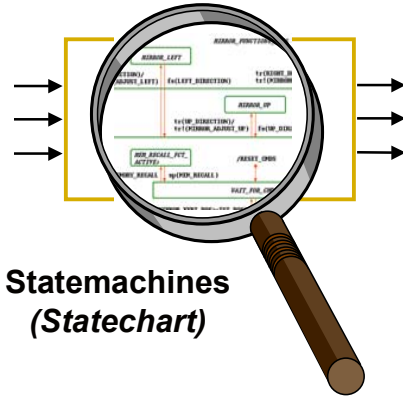
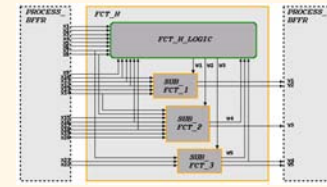
Languages of *StateMate*™

Describing structure and state-based behavior through Activity Chart and Statechart

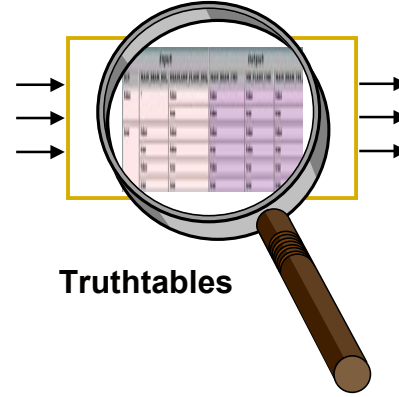


State^{mate}™ Activity Charts

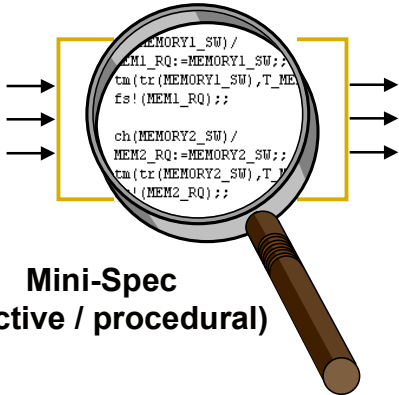
Describing Basic Function-Blocks (“Basic Activities”)



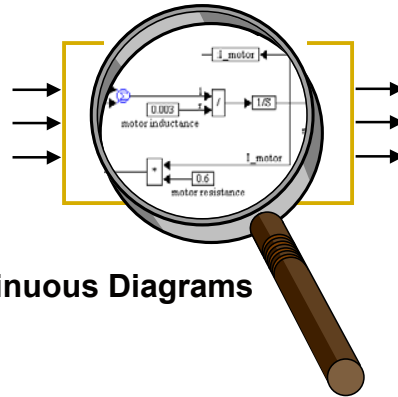
Statemachines
(Statechart)



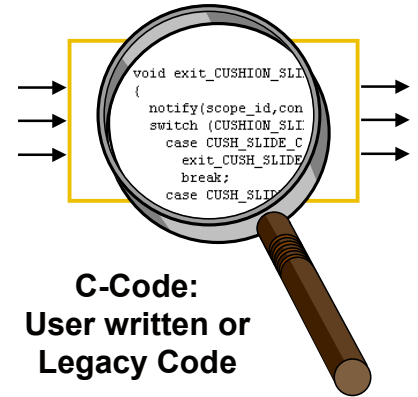
Truthtables



Mini-Spec
(reactive / procedural)



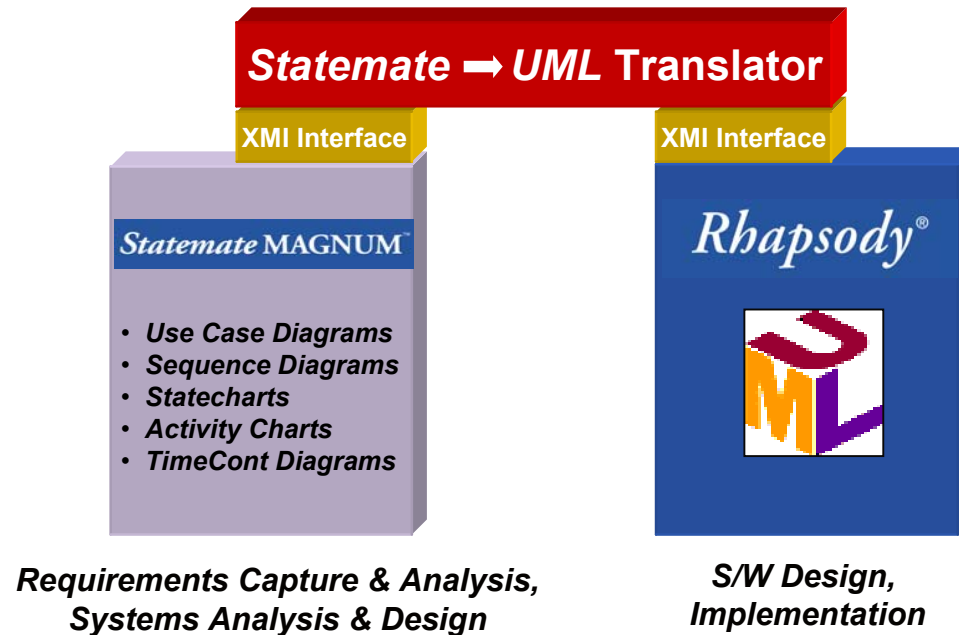
Continuous Diagrams



C-Code:
User written or
Legacy Code

Transition from function driven SE to object oriented SWE

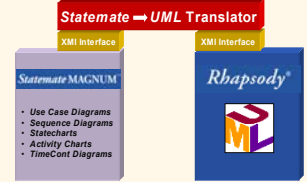
The "Statestate-to-UML Bridge"



The primary goal of the translation is to carry as much information as possible from the Systems Analysis and Design phase over into the UML.

Regardless of how much information is finally transferred, the translated UML model will still require some manual modifications before target code can be generated from it.

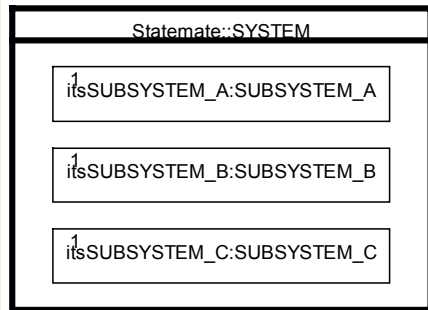
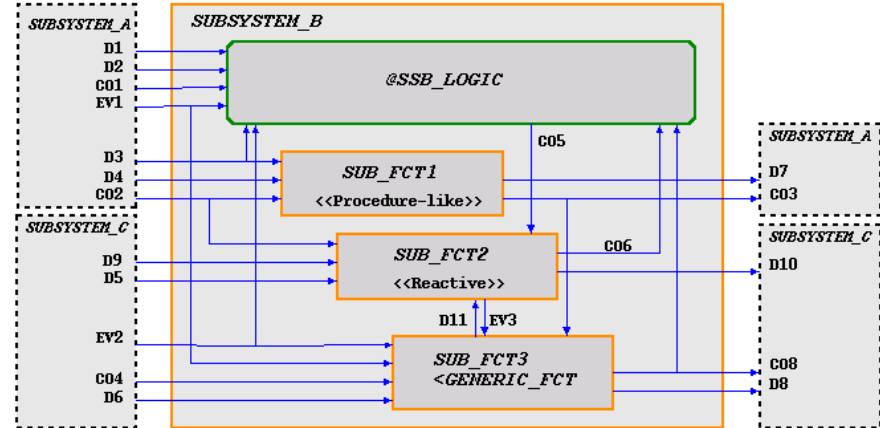
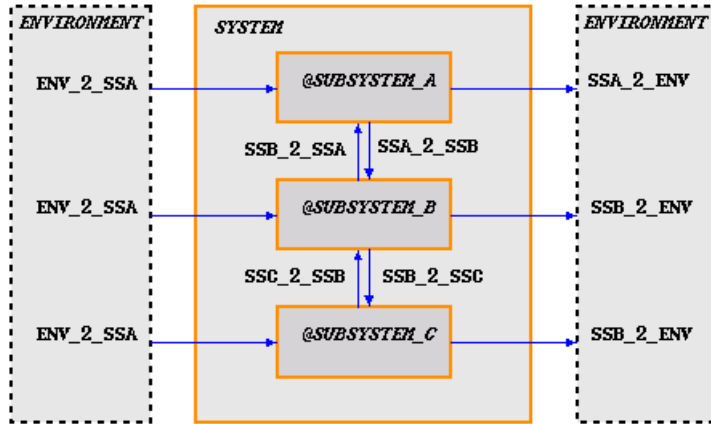
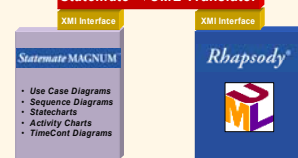
Translation Rules



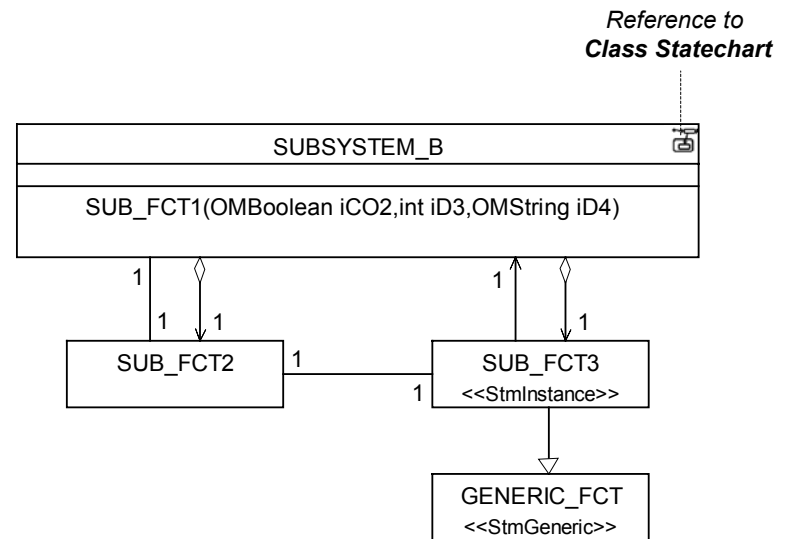
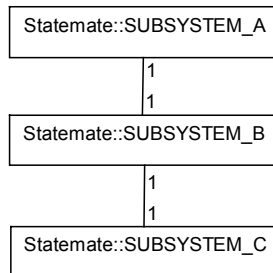
StateMate Artifacts	UML Artifacts
Activities: <ul style="list-style-type: none"> - Environmental - Compound - Procedure-like - Reactive - Time-continuous - Generics - Instances of Generics 	Actors (Composite) Classes Operations Reactive Classes¹ Classes, stereotyped <<Continuous>> Classes, stereotyped <<StmGeneric>> Instances of the SuperClass, stereotyped <<StmInstance>>
Legacy Code	Classes, stereotyped <<Legacy>>
Statecharts	Class Statecharts
Sequence Diagrams	Sequence Diagrams

¹ UML: Static reaction in one-state statemachine
 Rh: Reactive behavior copied into the description field of the class

Translation of the *Statestate*TM Activity Chart Structure to the *UML*

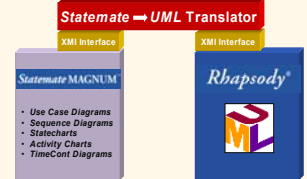


Composite Class

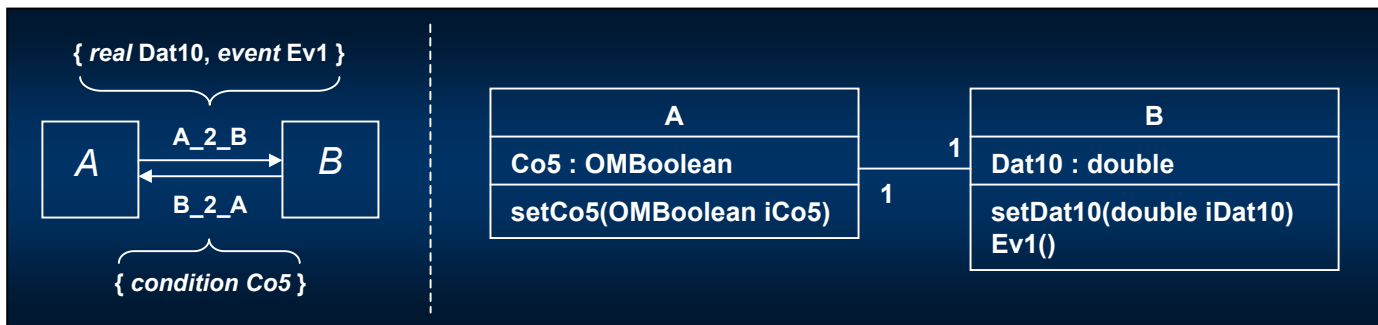


Reference to Class Statechart

Translation Rules cont'd

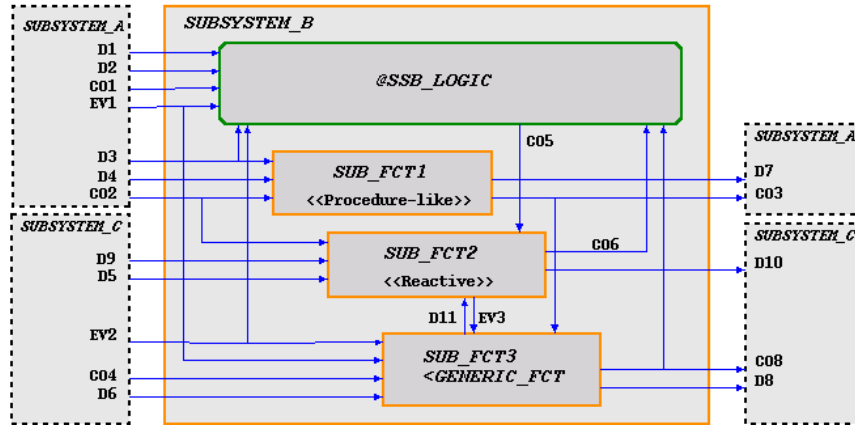
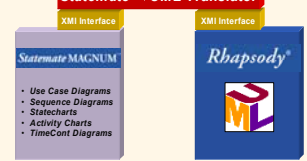


StateMate Artifacts	UML Artifacts
Information Flows	Associations between objects
InfoFlow Elements <ul style="list-style-type: none"> - Data - Conditions - Events 	Messages setDataX(iDataX:typeX) with matching Operations and Attributes on the receiving object Messages setCondY(iCondY:OMBoolean) with matching Operations and Attributes on the receiving object Events on the receiving object
Non-InfoFlow Elements	Global Data
Global Definition Sets	Packages

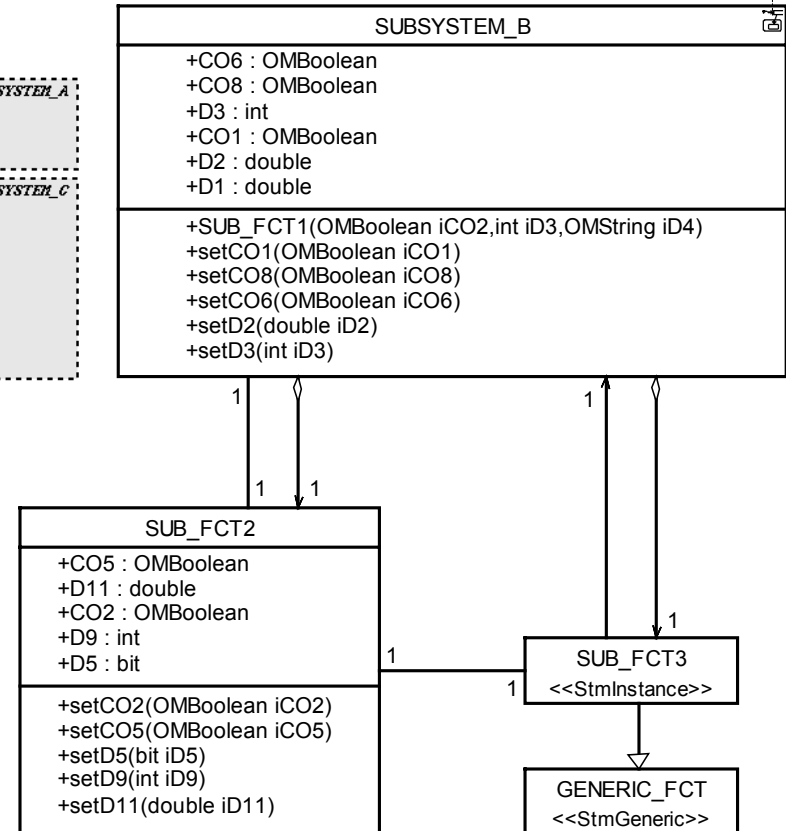


Note: Only InfoFlows into activities are considered . Outgoing InfoFlows are captured at the target activity.

Translation of Statestate™ Information Flows to the UML

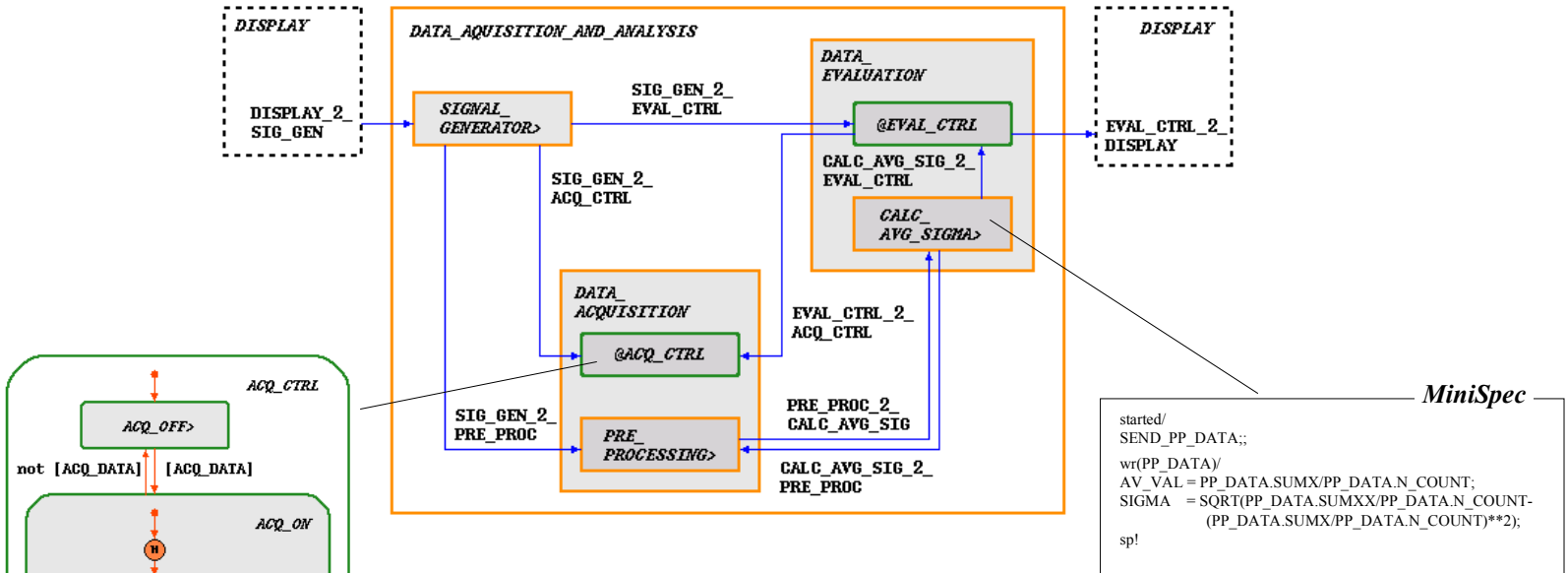


Reference to Class Statechart



Actors neglected

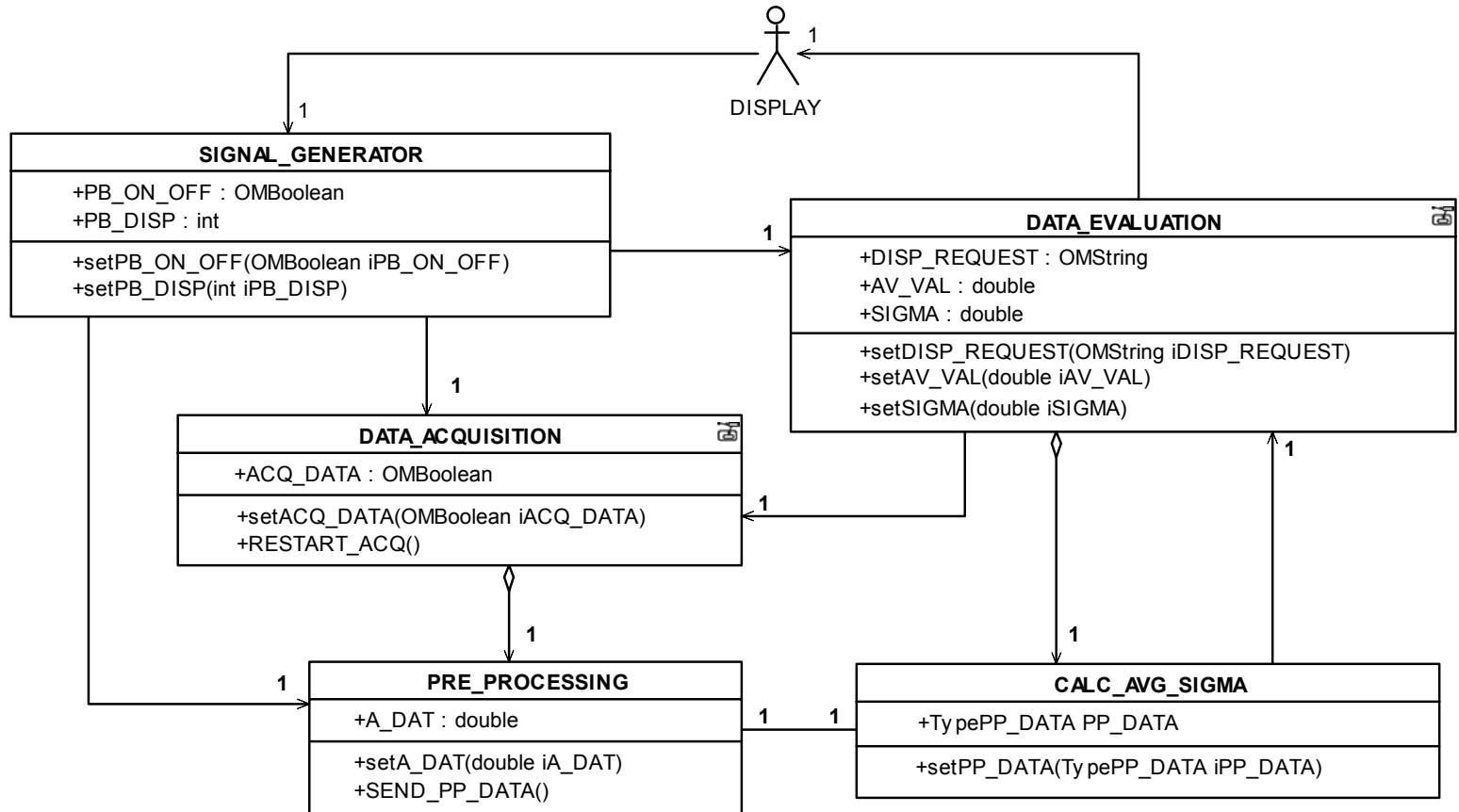
Statemate™ Model of a Data Acquisition and Analysis System



Data Dictionary:

DISPLAY_2_SIG_GEN:	{integer PB_DISP, condition PB_ON_OFF}
EVAL_CTRL_2_DISPLAY:	{real DISP_DATA}
SIG_GEN_2_ACQ_CTRL:	{condition ACQ_DATA}
SIG_GEN_2_PRE_PROC:	{real A_DAT}
SIG_GEN_2_EVAL_CTRL:	string DISP_REQUEST
PRE_PROC_2_CALC_AVG_SIG:	{record PP_DATA { integer N_COUNT, real SUMX, real SUMXX}}
CALC_AVG_SIG_2_PRE_PROC:	{event SEND_PP_DATA}
CALC_AVG_SIG_2_EVAL_CTRL:	{real AV_VAL, real SIGMA}
EVAL_CTRL_2_ACQ_CTRL:	{event RESTART_ACQ}

Translated UML Model of the Data Acquisition and Analysis System specified in *Statemate*™



Benefits of the Hybrid Approach

- The Hybrid Approach facilitates the communication and the design interchange between function-driven systems engineering teams and object-oriented software development teams.
- Using the Function-driven Approach, systems engineers can continue to work in the methodology they are comfortable with.
- The Function driven Approach enables the reuse of legacy information (e.g. *TEAMWORK™* Models).
- The Hybrid Approach provides a transition to the evolving standard for object-oriented software development.