INCOSE Training Webinar
System Architecture

Developed and presented by
Richard E. (Dick) Fairley

Brief Bio

dickfairley@gmail.com

Dick Fairley is a long-time member of INCOSE. He is an author of the Guide to the Systems Engineering Body of Knowledge (SEBoK V 1.0), an editor of the present SEBoK V 1.9, an INCOSE commissioner for ABET accreditation of systems engineering degree programs, and INCOSE liaison to the IEEE Systems Council.

He is principal associate of Systems and Software Engineering Associates (S2EA), a consulting and training company. Dick is a former professor and associate dean of the Oregon Graduate Institute and past Chair of the IEEE Computer Society Systems and Software Engineering Committee. He was co-editor of the Software Engineering Body of Knowledge (SWEBOK V 1.3) and leader of the teams that developed the Software Engineering Competency Model (SWECOM) and the Software Extension to the PMI Guide to the Project Management Body of Knowledge (SWX).

His Bachelors and Masters degrees are in electrical engineering and his PhD is in computer science and applied math. He worked in industry as an electrical and systems engineer before returning to school to obtain his PhD from UCLA.

Dick and his wife Mary Jane live in the Colorado mountains northwest of Colorado Springs. He enjoys listening to jazz, hiking, and skiing. They enjoy motorcycling together.

Five training webinars

1. Stakeholders' requirements

   November 8, 2018

2. System requirements

   November 15, 2018

3. System architecture

   January 3, 2019

4. System design

   January 10. 2019

5. System implementation

   January 17, 2019

All 5 webinars will be presented at noon EST and recorded for later viewing

Accessing the training webinars

- These and other INCOSE webinars are recorded for listening and downloading of the presentation slides
- First, sign on to:

  https://connect.incose.org/Library/Tutorials/training/SitePages/Home.aspx
- Then scroll to the bottom of the page. Under Systems Engineering Technical Processes you will see the technical process training webinars

For assistance, contact John Clark or Gabriela Coe at john.clark@incose.org or gabriela.coe@incose.org

Primary references for this webinar (1)

Clause 6.4.3 of ISO/EIC/IEEE Standards 15288:2015 and 12207:2017

- **15288:** Systems and Software Engineering - - System life cycle processes

  https://www.iso.org/standard/63711.html
  https://standards.ieee.org/standard/15288-2015.html

- **12207:** Systems and Software Engineering - Software Life Cycle Processes

  https://www.iso.org/standard/63712.html
  https://standards.ieee.org/standard/12207-2017.html

NOTE

- Clause 6.4 of 15288 (Technical Processes) provides the framework for these training webinars
- Clause 6.4 presents 14 technical processes of systems engineering as clauses 6.4.1 through 6.4.14, in sequence
  - Sequential execution of the technical processes is not implied by the sequential listing of them in 15288
- Process models for system development typically incorporate the technical processes in various iterated, overlapped, and repeated ways, and concurrently and recursively as required

Note

- Access to 15288 and 12207 is not necessary for these training webinars
  - However, your organization or library may have access to them if you desire to see them

Another primary reference

Systems Engineering for Software-enabled Physical Systems

e.g., embedded, IoT, cyber-physical, C2 systems, and others

To be published by Wiley in May, 2019

Reference for this webinar:

Chapter 8: Architecture definition and design definition

Other reference materials

1. INCOSE Systems Engineering Handbook
2. INCOSE Systems Engineering Competency Framework
3. The Guide to the Systems Engineering Body of Knowledge

   https://sebokwiki.org

Agenda: System Architecture Definition

- Brief review of training webinars 1 & 2
- System definition
- Purpose of architecture definition
- Notations for architecture definition
- Verifying and validating architecture definitions
- Roles played by systems engineers in architecture definition
- A brief preview of training webinar 4
- Questions and comments

Webinar 1: 15288 Clause 6.4.1
Business or Mission Analysis

Input	A business problem, mission need, or opportunity to be studied.
Process	Define the business problem, mission need, or opportunity. Then describe the solution space and Determine one or more solution classes in the solution space.
Output	A documented definition of a business problem, a mission need, or an opportunity plus a description of the solution space and one or more solution classes. **And** a go, no-go, no-bid, or further-study decision.

Webinar 1: 15288 Clause 6.4.2
Stakeholder Needs and Requirements Definition

Input	A documented statement of a business problem, a mission need, or an opportunity. **And** a decision to proceed.
Process	Identify stakeholders and characterize the operational environment. Elicit, categorize, and prioritize operational requirements. Identify needed system capabilities, constraints, and risk factors. Conduct a feasibility study. Develop a documented agreement between the acquiring organization and the supplying organization.
Outputs	**Assuming** a decision to proceed based on the feasibility study and a risk analysis: Stakeholders' requirements definition, system capabilities, constraints, and identified risk factors. A Statement of Work or Memo of Understanding. A Concept of Operations that includes operational scenarios.

Categories and priorities of stakeholders' requirements

Categories of stakeholders' requirements:

- Features: what the system must do for stakeholders
- Quality attributes: how well it will do it
- Design constraints: must-bes with limited design options
- Will not do: to control expectations
- Must not do: for safety, security, policies, regulations

Priorities of stakeholders' requirements:

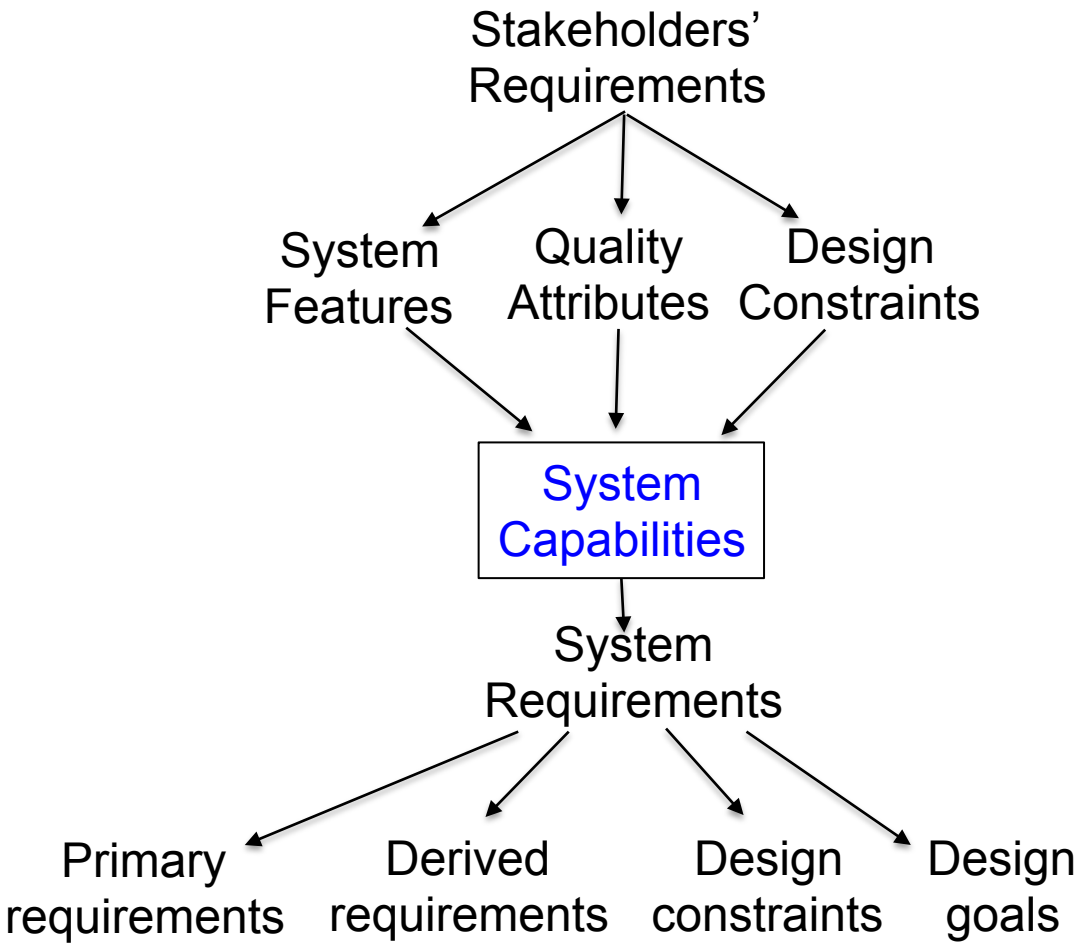
- Essential, Desirable, Optional

Stakeholder features, quality attributes, and design constraints provide the basis for system requirements definition, system design, and system implementation

Webinar 2: 15288 Clause 6.4.3
System Requirements Definition

Input	Business or mission analysis Stakeholders' requirements definition A preliminary SEMP
Process	Develop system capabilities Define, categorize, and prioritize the system requirements iv&v the system requirements Finalize the initial SEMP
Output	Informally verified and validated system requirements Finalized SEMP

A requirements taxonomy



System quality attributes are categorized as system design constraints

Agenda: System Architecture Definition

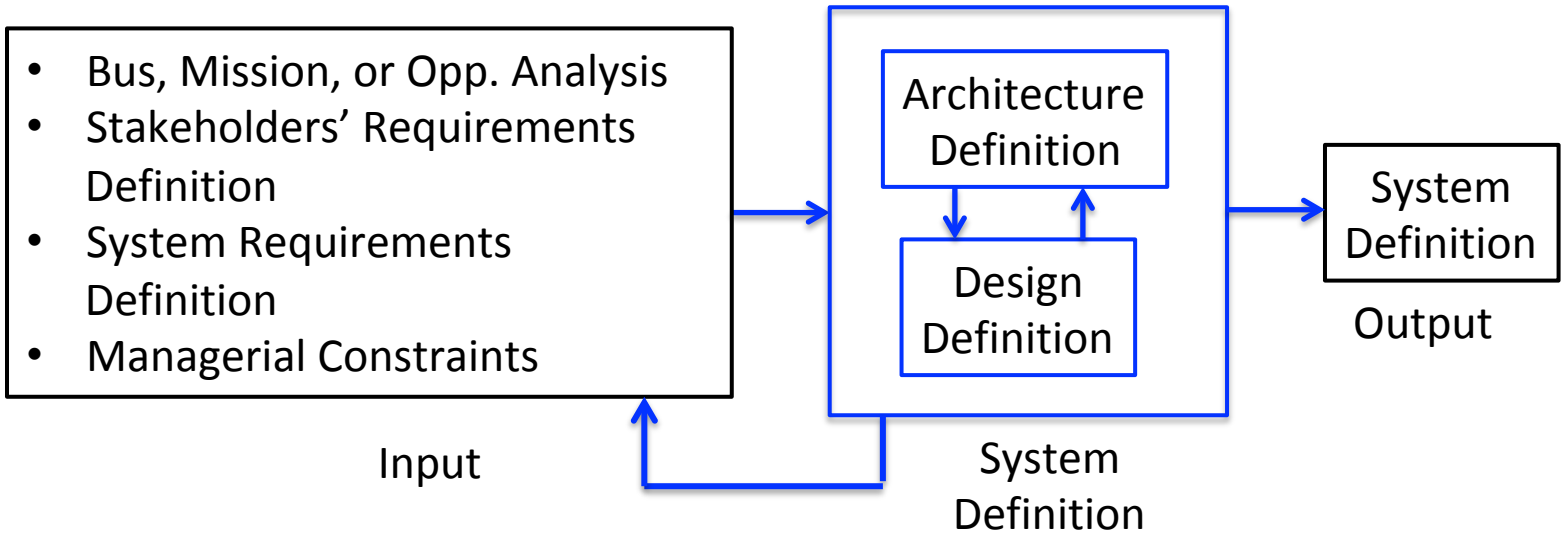
- Brief review of training webinars 1 & 2
- System definition
- Purpose of architecture definition
- Notations for architecture definition
- Verifying and validating architecture definitions
- Roles played by systems engineers in architecture definition
- A brief preview of training webinar 4
- Questions and comments

System definition

- System definition includes architecture definition and design definition
- Webinar 3: Architecture definition is concerned with specifying the structure and behavior of a system that will satisfy the problem statement, mission need, or opportunity; the stakeholders' requirements; the system capabilities; the system requirements; and the managerial constraints
- Webinar 4: Design definition is concerned with defining system elements and interfaces in sufficient detail to provide a basis for system implementation

Architecture definition and design definition are separate but closely related processes in 15288-2015

The system definition process



Initial architecture definition (usually) precedes initial design definition

System definition then becomes an iterative and recursive process of architecture definition and design definition

Quality attributes, other design constraints, and managerial constraints are the primary drivers of system architecture and system design

System Architecture Definition*

"The purpose of the Architecture Definition process is

to generate system architecture alternatives,

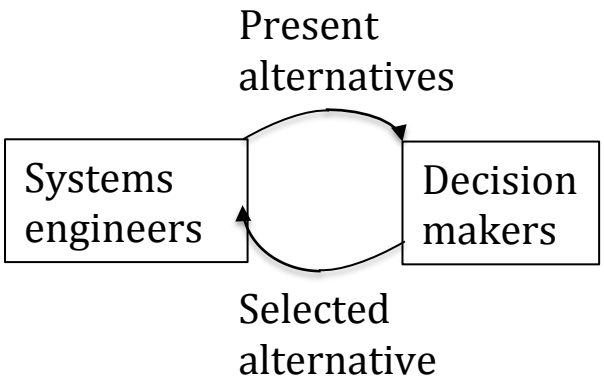
to select one or more alternatives that frame stakeholder concerns and meet system requirements, and

to express this in a set of consistent views."

* ISO/IEC/IEEE Standard 15288, Clause 6.4.4.1

Alternative architectures

- Alternative architectures include different tradeoff options
  - Constrained by the system capabilities and performance parameters to be provided
  - And constrained by schedule, budget, resources, ad risk analysis
  - Potential flexibility in the constraints may be explored
- Objective evaluation criteria are used (or should be used) to select the alternative architectures presented to the decision makers



A case study: the RC-DSS

- An opportunity:
  - To develop, license, and sell a variety of realistic driving system simulators for land based vehicles having four or more wheels.
- The solution space:
  - The solution space ranges from a simple laptop application to a steering wheel and a display screen to full-scale vehicles instrumented with sensors, actuators, controls, and displays.
- A solution class
  - Flight training simulators used to train and certify commercial and military pilots provide a solution class for realistic driving system simulators.

RC-DSS alternative architectures

Alternative architectures for the DSS include:

A. A laptop computer application using the touchpad, display screen, and audio output.

B. An unenclosed driver's seat, steering wheel, and display panel, with minimal controls and indicators. (speedometer, accelerator, brakes).

C. A realistic full-scale vehicle simulator for land-based vehicles having 4 or more wheels.

D. A retrofitted flight-training simulator.

RC-DSS architecture evaluation criteria

Evaluation criteria for RC-DSS architecture options include:

1. A safe and secure environment for all hands-on RC-DSS users including students, instructors, technicians, and others
2. A realistic driving experience for student drivers
3. A system architecture that will support a product line of driving simulators
4. A cost-effective implementation

RC-DSS architecture alternatives and criteria

Criteria / Alternative	1	2	3	4
A	X		X	X
B	X		X	X
C	X	X	X	X
D	X		X	

RC-DSS architecture alternatives and criteria potentially satisfied

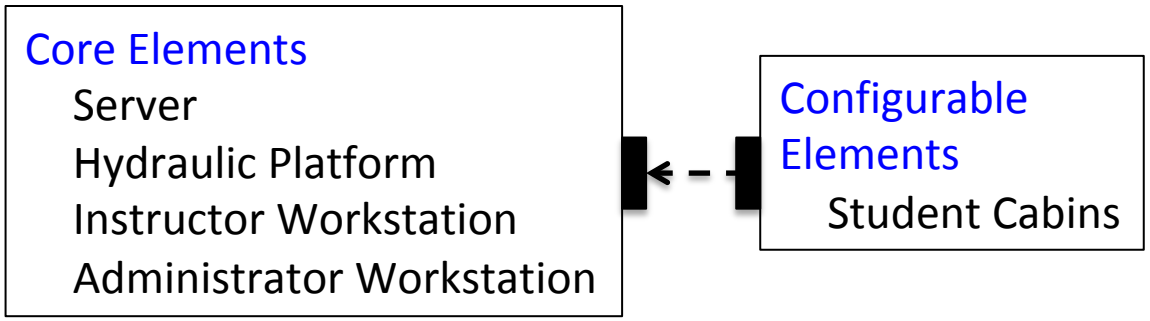
Criteria / Option	1	2	3	4
A	X		X	X
B	X		X	X
C	X	X	X	X
D	X		X	

Q: Did I craft the evaluation criteria to favor alternative C?

Selecting the RC-DSS architecture

- Evaluation criterion 3 (a product line architecture) is a system requirement generated by the Realistic Corporation
- The resulting RC-DSS system architecture will include interchangeable student cabins mounted on a hydraulic platform, similar to a flight training simulator
- Five RC-DSS subsystems:
  - Interchangeable student cabins, hydraulic platform, computer server, instructor workstation, administrator workstation
  - Encased in a lightweight enclosing structure

The RC-DSS product line architecture



A rule of thumb: core elements provide 80% or more of the system functionality, behavior, and quality attributes

System Architecture Definition*

"The purpose of the Architecture Definition process is

to generate system architecture alternatives,

to select one or more alternatives that frame stakeholder concerns and meet system requirements, and

to express this in a set of consistent views."

* ISO/IEC/IEEE Standard 15288, Clause 6.4.4.1

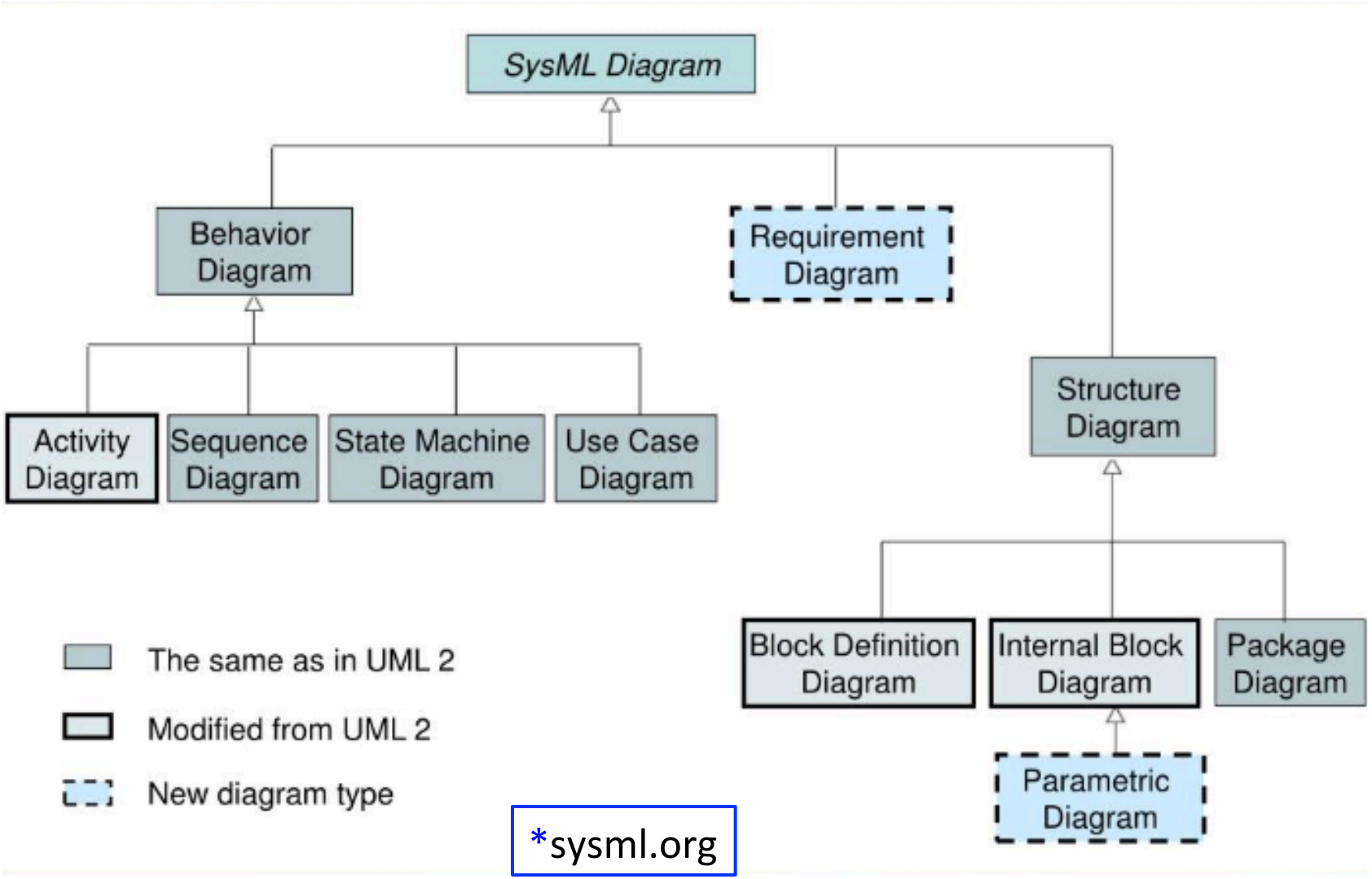
Multiple views

- Multiple views of an architecture are needed because system architectures are too complex to portray in a single view
  - And because separate compatible views of structure and behavior are needed
- Views can be specified using the Systems Modeling Language

  See sysml.org and omgsysml.org

SysML can also used to depict system requirements

SysML diagrams*



*sysml.org

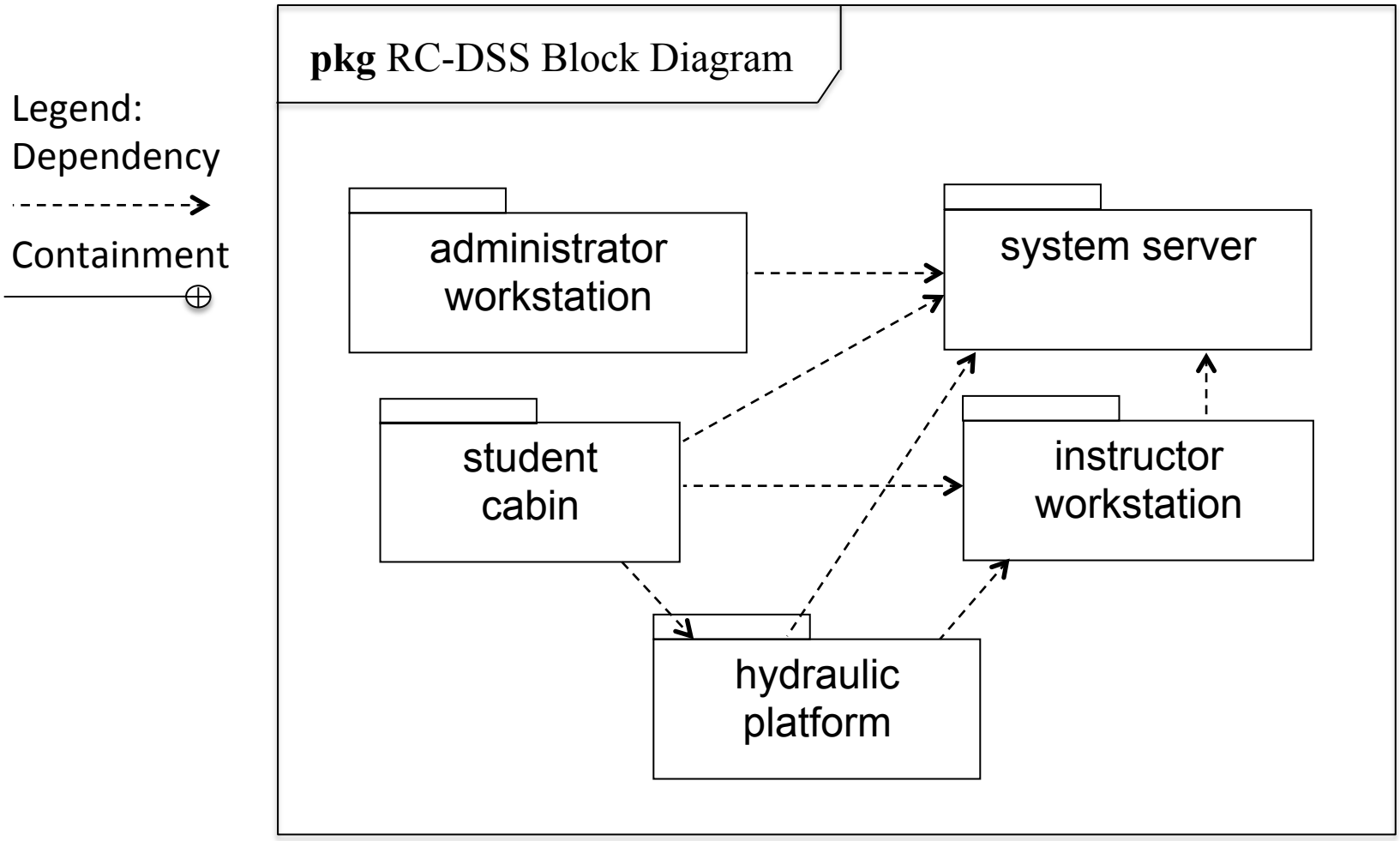
SysML structure diagrams

- Package diagrams (**pkg**)
- Block definition diagrams (**bdd**)
- Internal block diagrams (**ibd**)

SysML packages

- The SysML package diagram (unchanged from UML) is a mechanism for grouping other SysML diagrams, including other packages, in a unique namespace
  - Packages enclose one or more SysML diagrams using the *containment* relationship
  - They provide a mechanism for coping with system complexity

A RC-DSS package diagram

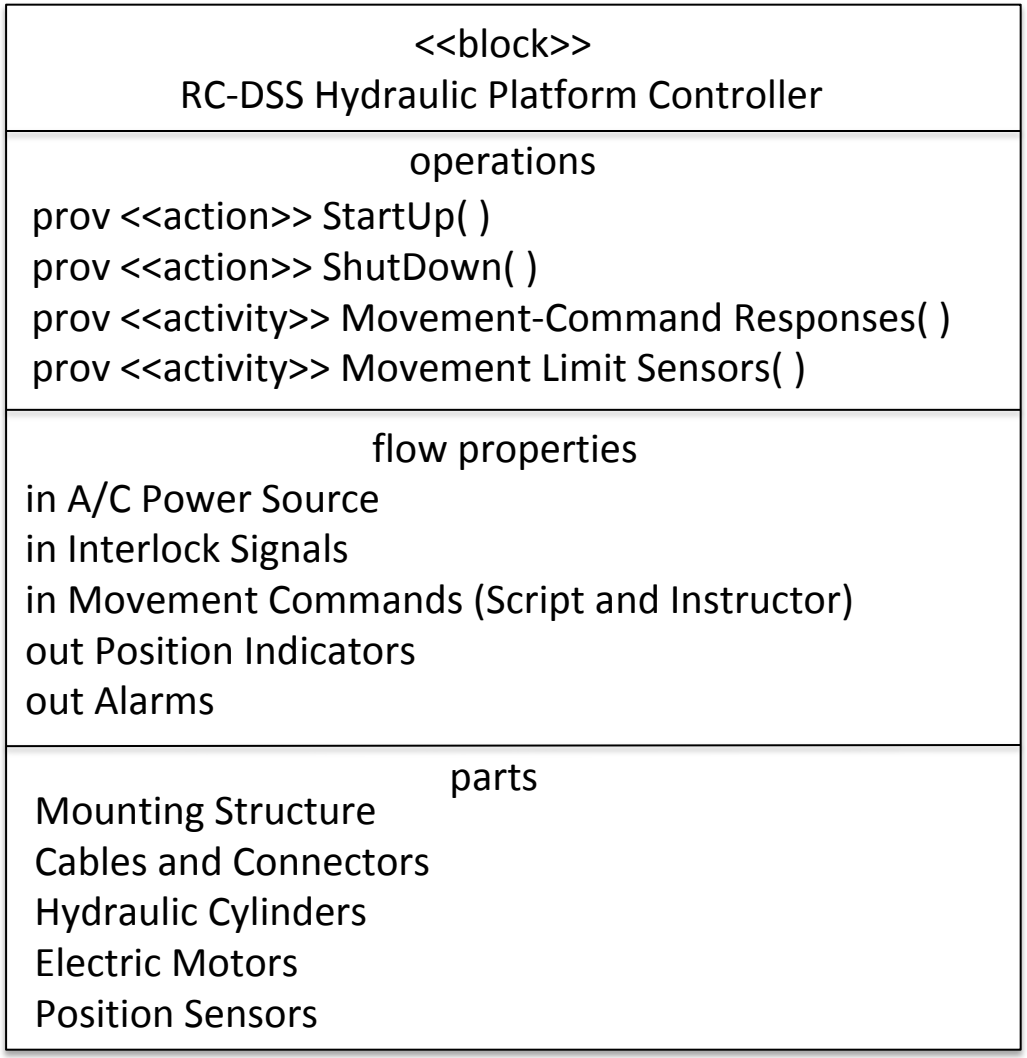


The package diagram is the generic SysML structure diagram for containment

SysML blocks

- Blocks are the SysML mechanism for specifying relationships among systems, subsystems, and elements, including hierarchical decomposition relationships
  - Blocks are customized extensions of UML classes
- Blocks can:
  - recursively depict systems, subsystems, and elements;
  - model logical and physical decomposition of a system;
  - specify hardware, software, and human elements; and
  - specify input/output flows of information and material (including energy)

Blocks can include optional *compartments* that can be used to provide descriptions of block attributes such as operations, flow properties, and parts



Two kinds of SysML block diagrams

There are two kinds of SysML blocks:

1. Block definition diagrams

   bdd blocks are black boxes that specify input and output flows and connections to other blocks

2. Internal block diagrams

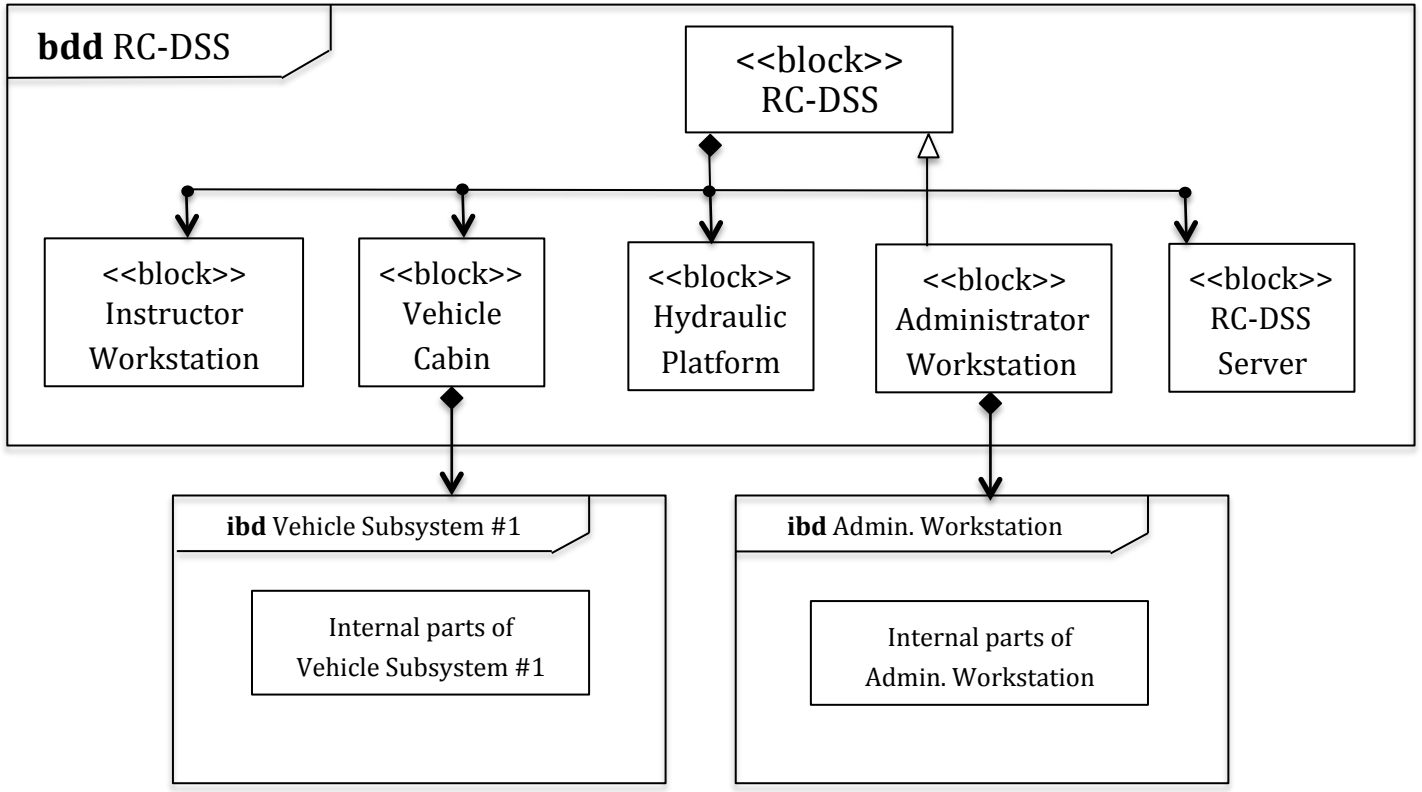
   ibd blocks are white boxes that show the internal structure of other blocks

Composition and aggregation relationships

- Composition: A child block in a hierarchy is owned by the parent block and is specifically designed to be part of the parent block
  - In hardware, the child block may be locally fabricated or "bespoke"
  - In software, the child block does not exist independent of the parent
- Aggregation: The child block is designed to be part of the parent block and part of other blocks, if desired
  - In hardware, the child block is often a commodity item
  - In software, the child block is typically stored in a library for additional use, if desired

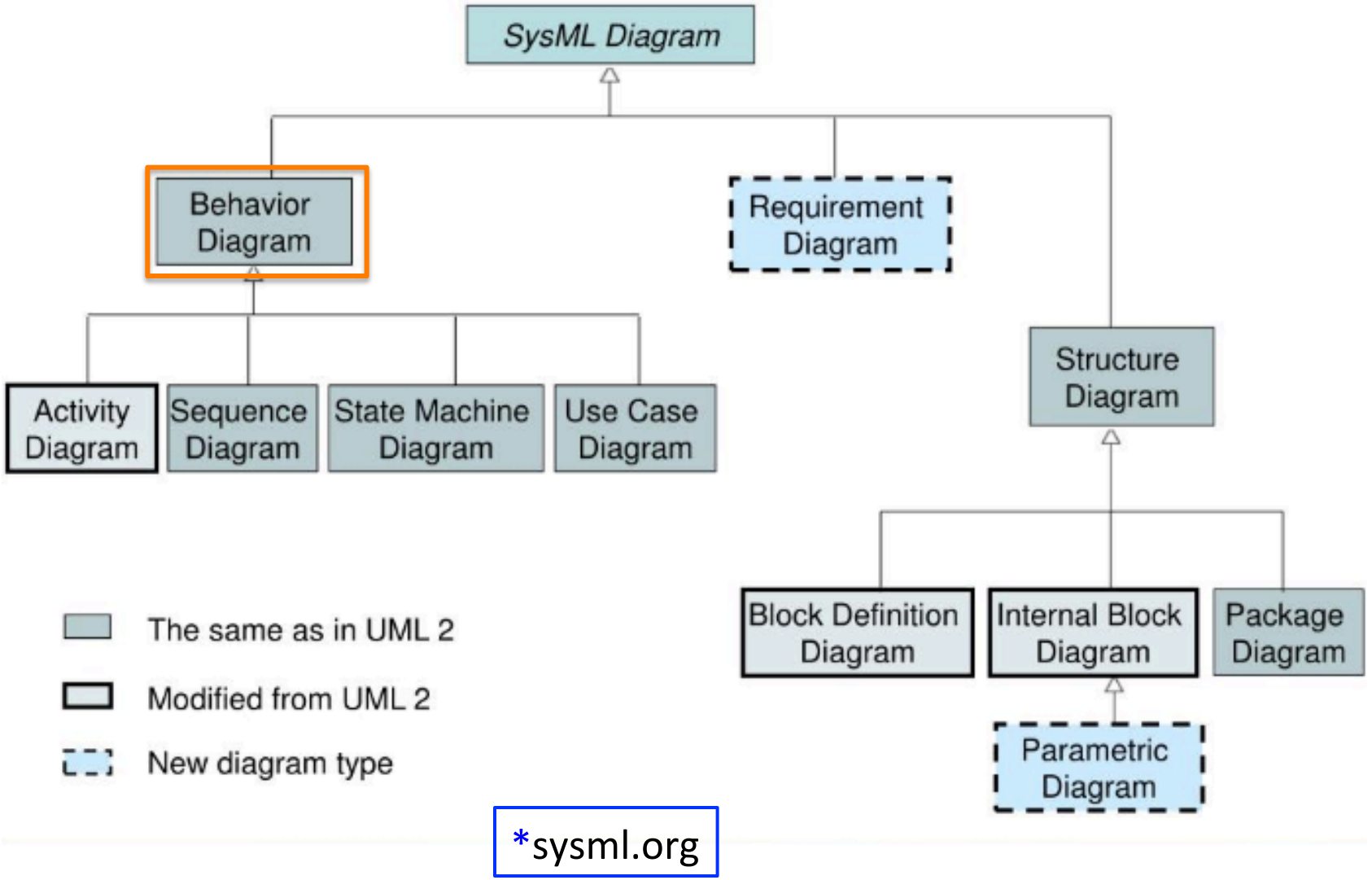
A partially decomposed RC-DSS system structure

Legend:
Composition ◆
aggregation △
connector •



More about blocks in Webinar 4

SysML diagrams*

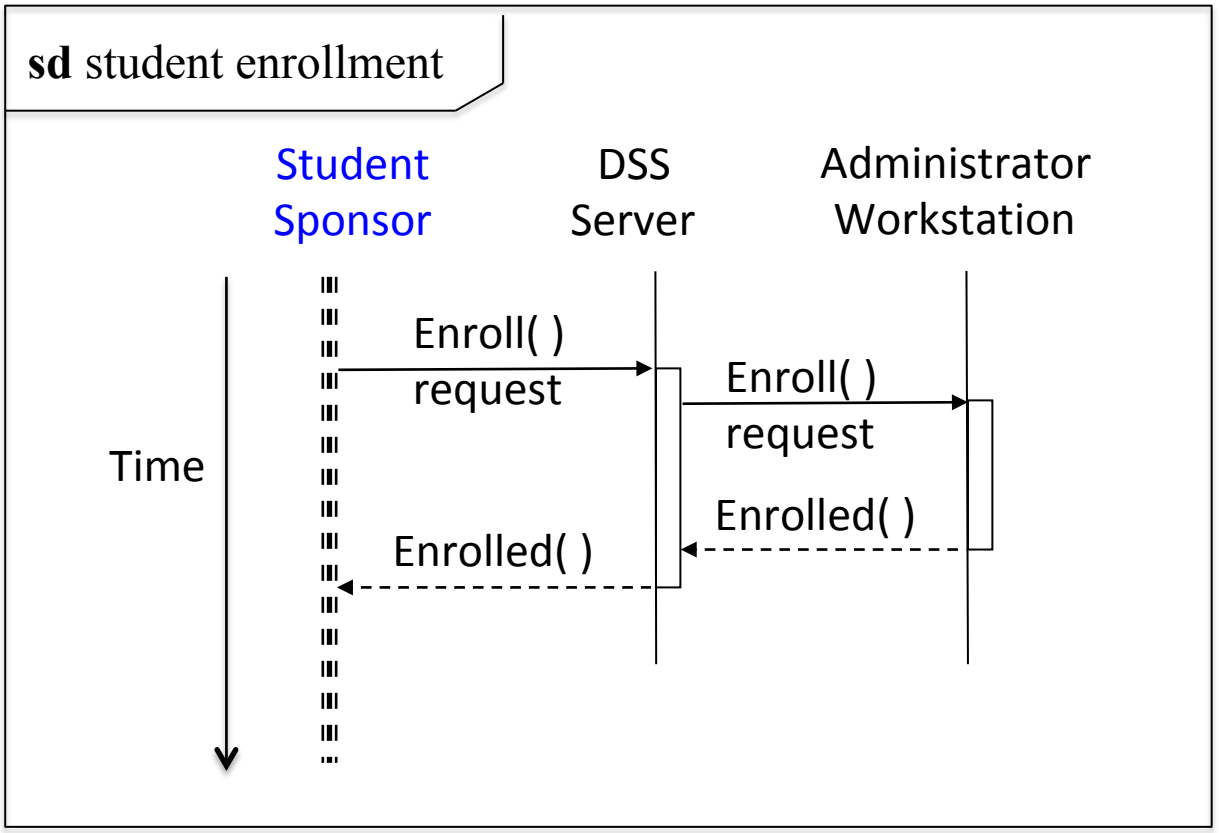


*sysml.org

SysML behavior diagrams

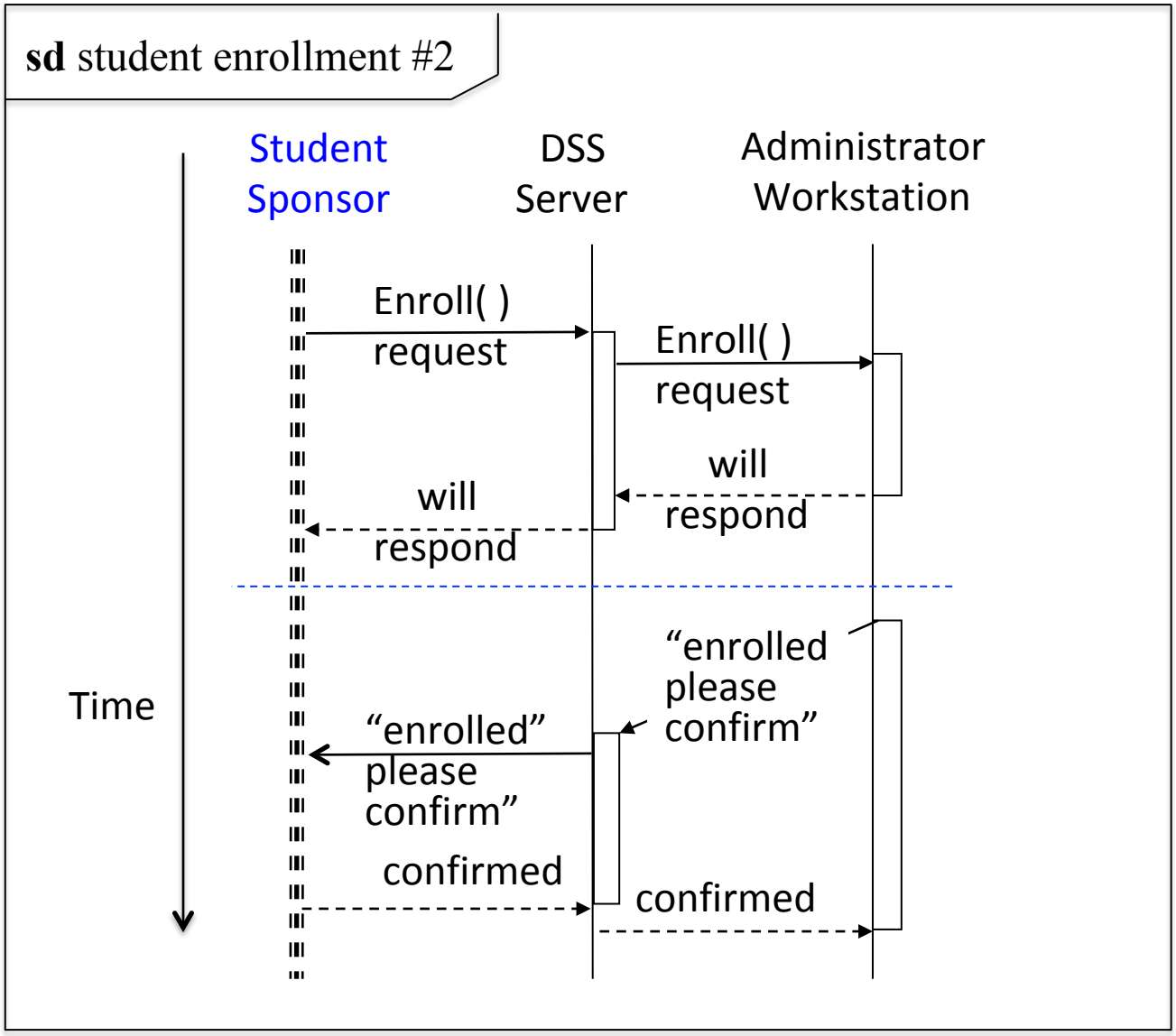
- Sequence diagrams (**sd**)
- Use case diagrams (**uc**)
- Activity diagrams (**act**)
- State machine diagrams (**sm**)

A SysML sequence diagram

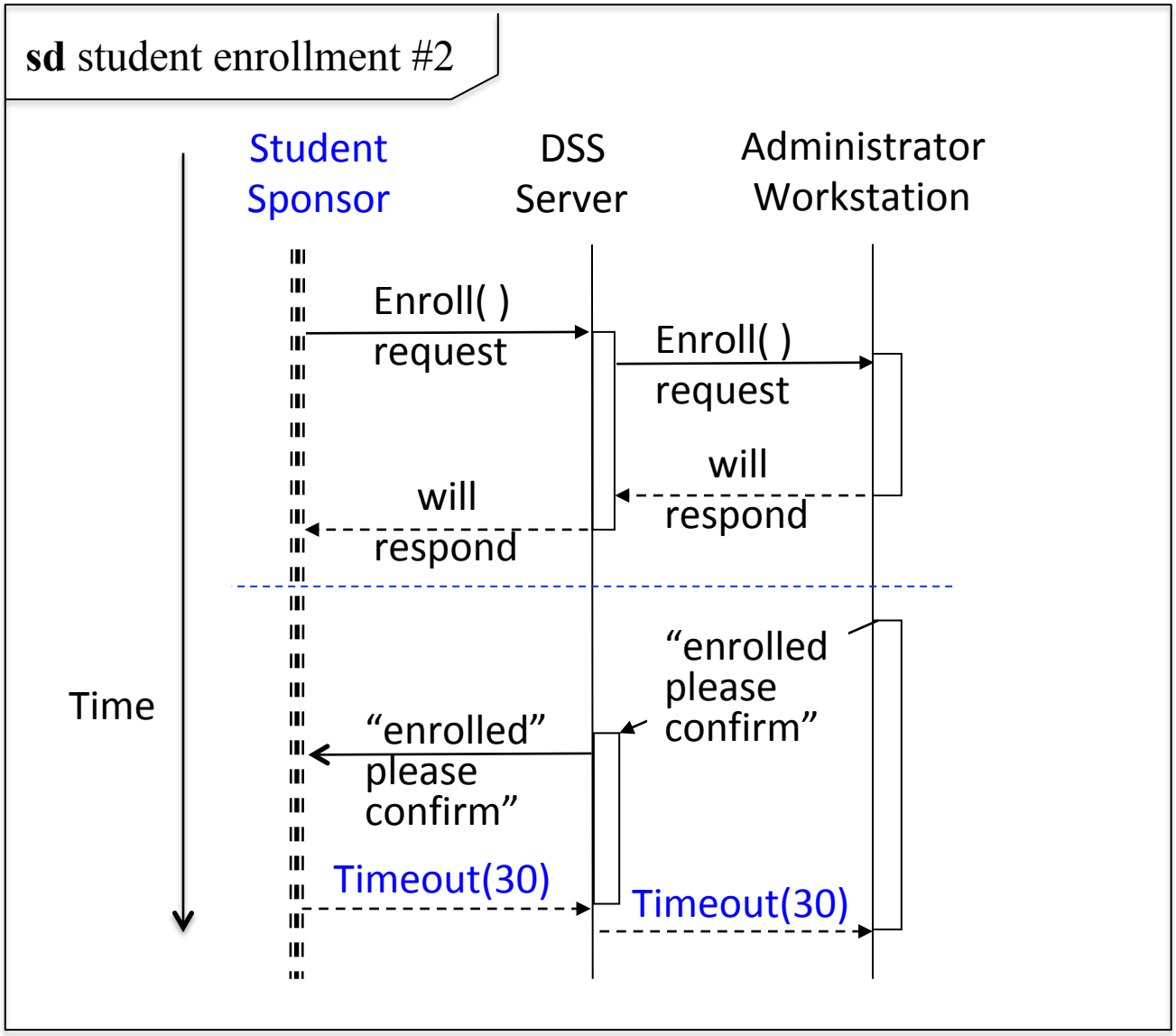


A sequence diagram can depict only one behavioral scenario

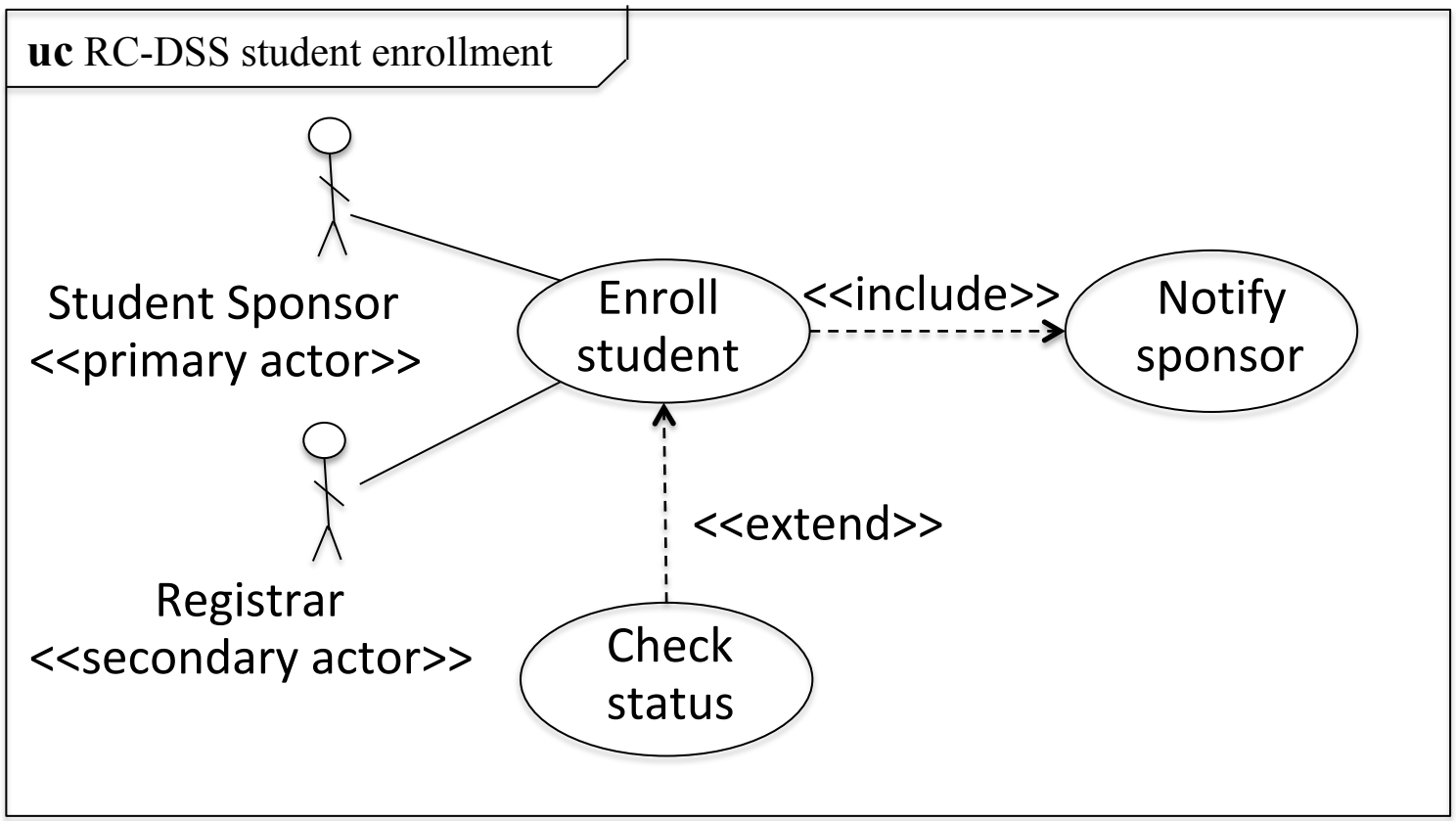
Another SysML sequence diagram



An exception SysML sequence diagram



A RC-DSS use case diagram



<<include>> relates use cases that can be used with more than one other use case
<<extend>> relates conditional use cases that are used as needed with another use case
<<include>> and <<extend>> control complexity by decomposing use cases

A use case template

- Name, id, date
- Author
- Preconditions
- Primary and secondary actors
- Primary scenario
- Names of the secondary scenarios
- Post conditions
- Associated use cases
- Associated stakeholder requirement(s) and business rules
- Comments
  - Can include constraints and quality attributes

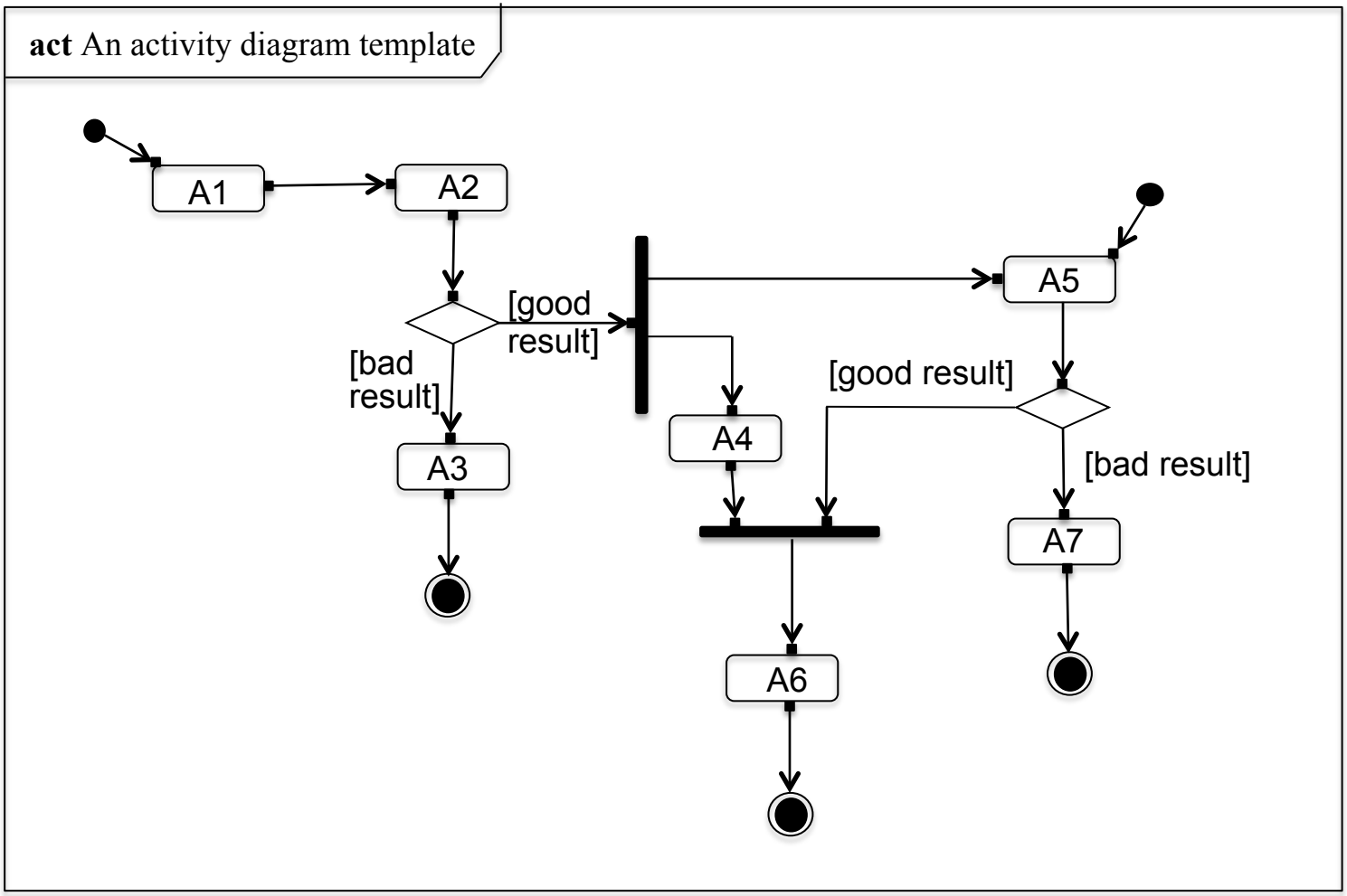
Documenting use case scenarios

Use case scenarios can be documented using:

- Textual sequences of interactions
- Behavioral diagrams
  - Sequence
  - Activity
  - State machine

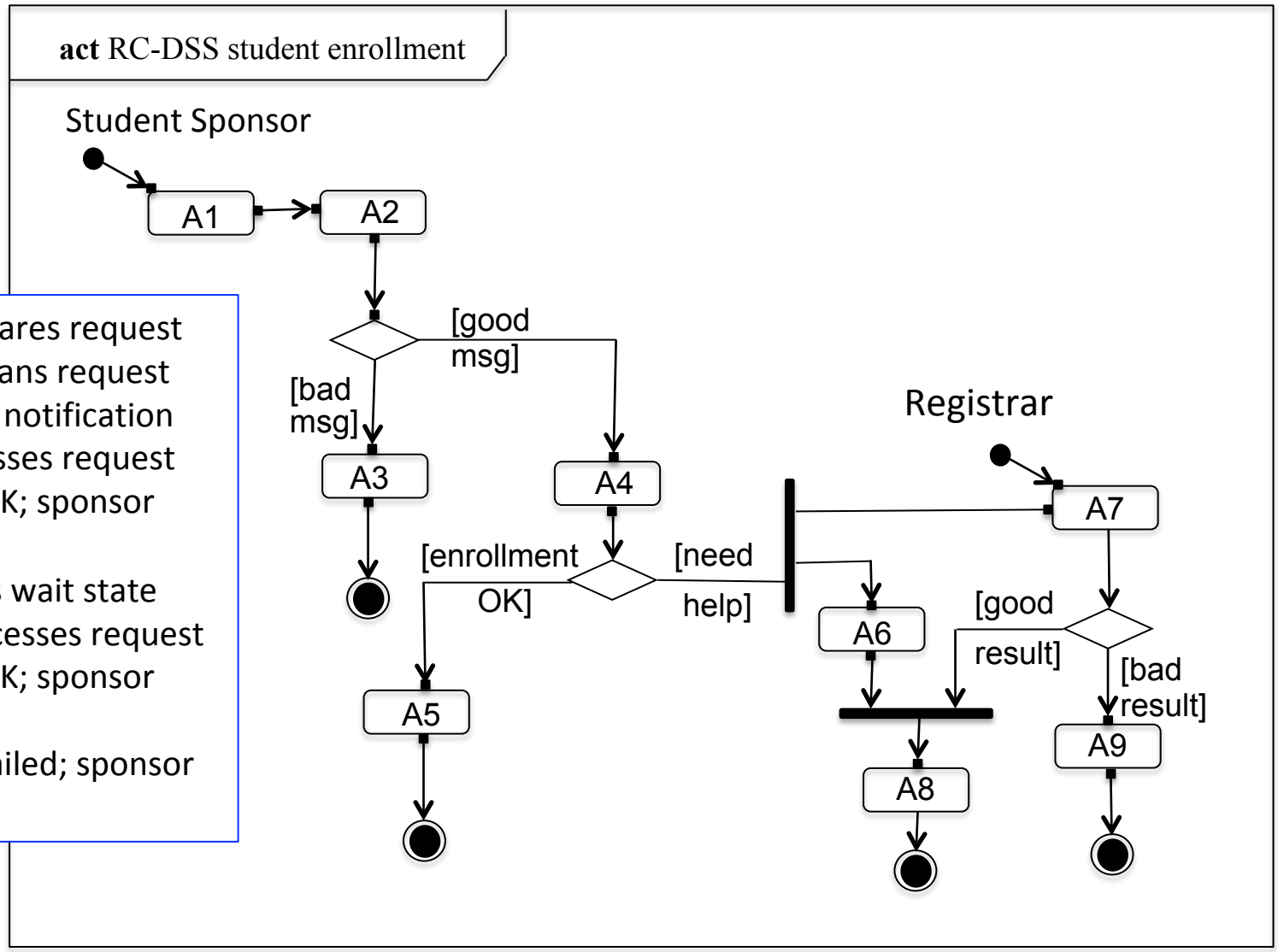
A use case scenario is a back-and-forth dialog between one-or-more actors and a system, a subsystem, or a system element

Template for a SysML activity diagram



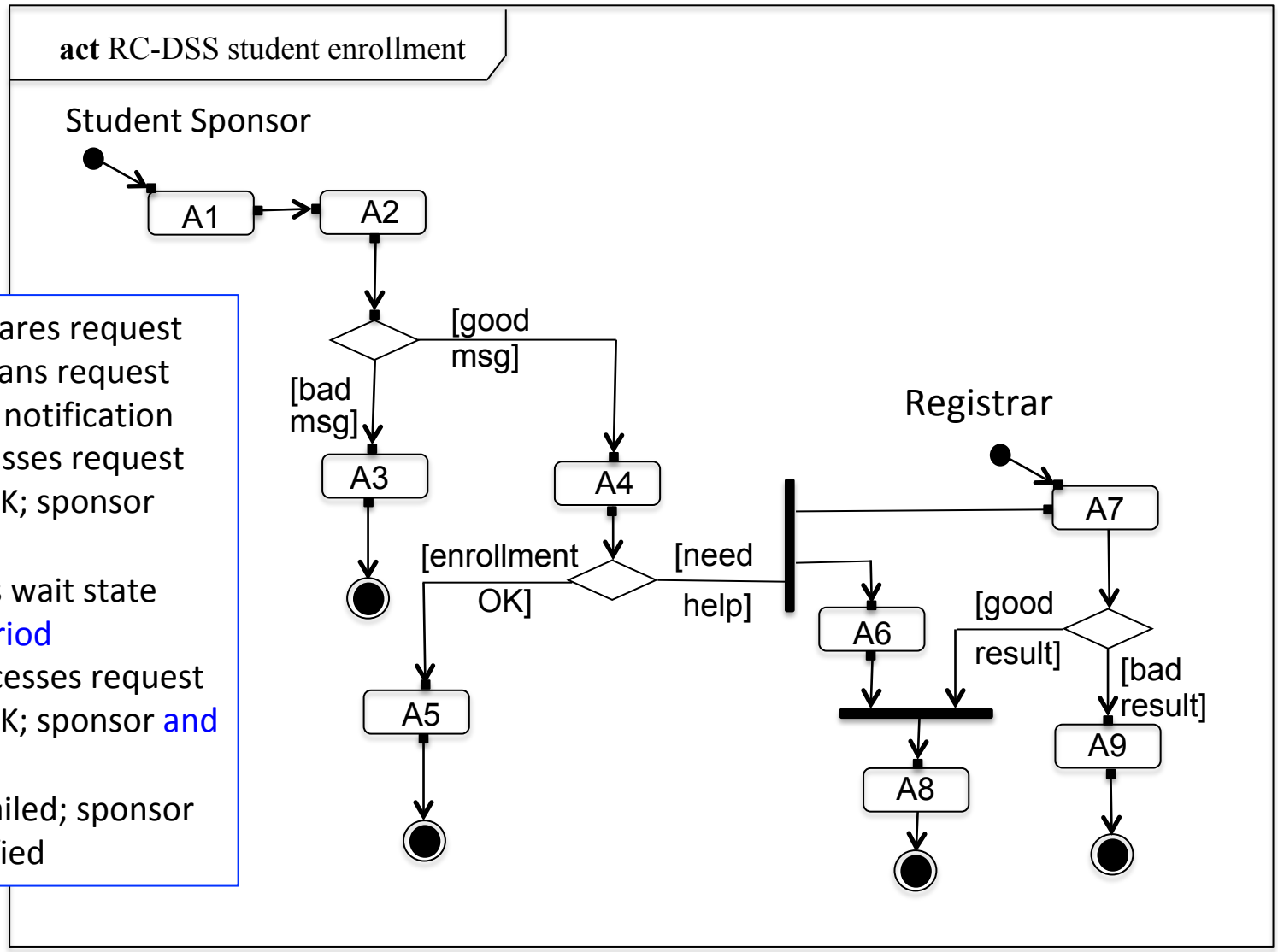
SysML **act** flows can be information or material

RC-DSS student enrollment activity diagram



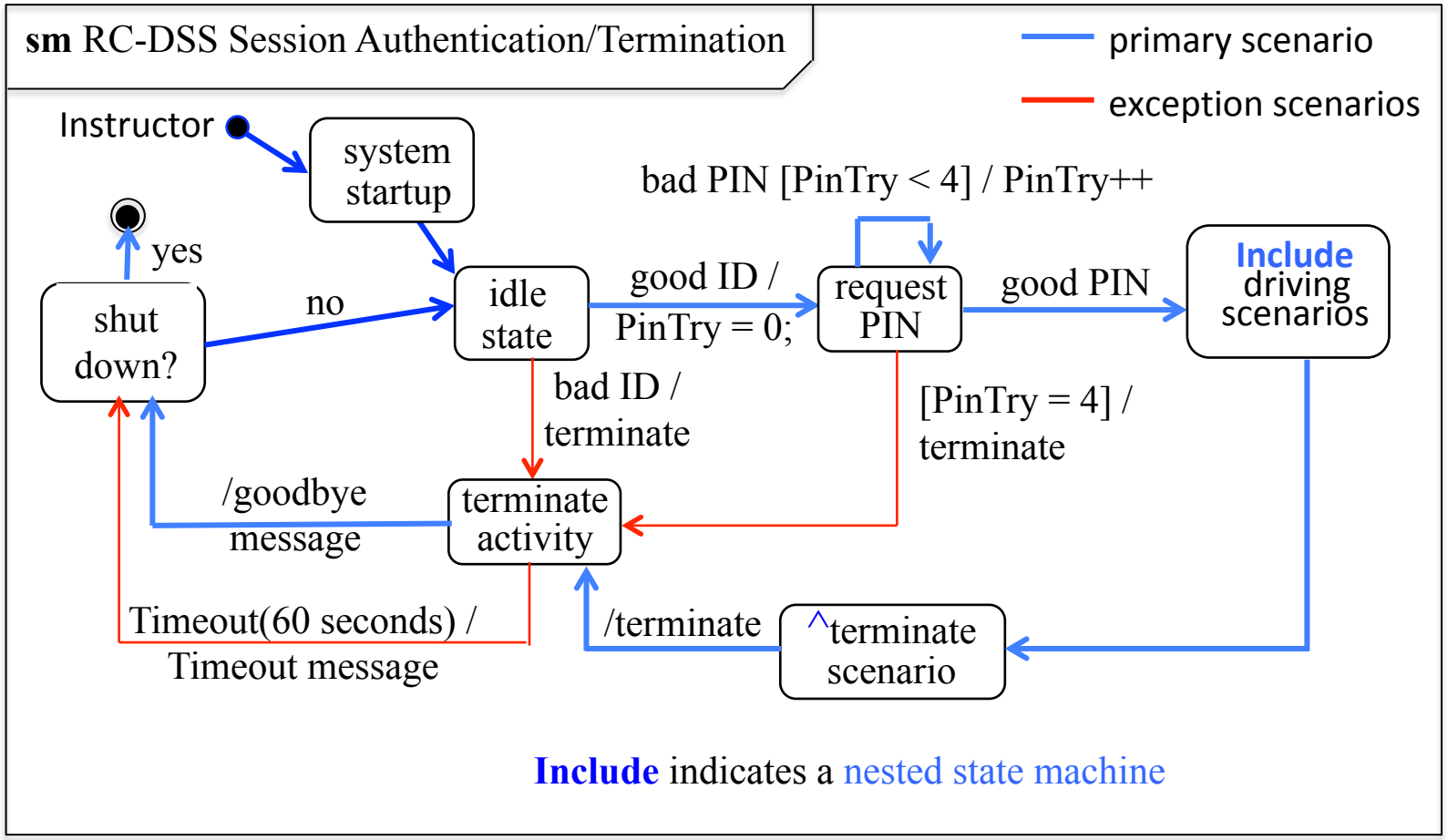
A1 sponsor prepares request
A2 DSS server scans request
A3 bad message notification
A4 Admin processes request
A5 enrollment OK; sponsor notified
A6 Admin enters wait state
A7 registrar processes request
A8 enrollment OK; sponsor notified
A9 enrollment failed; sponsor notified

RC-DSS student enrollment activity diagram



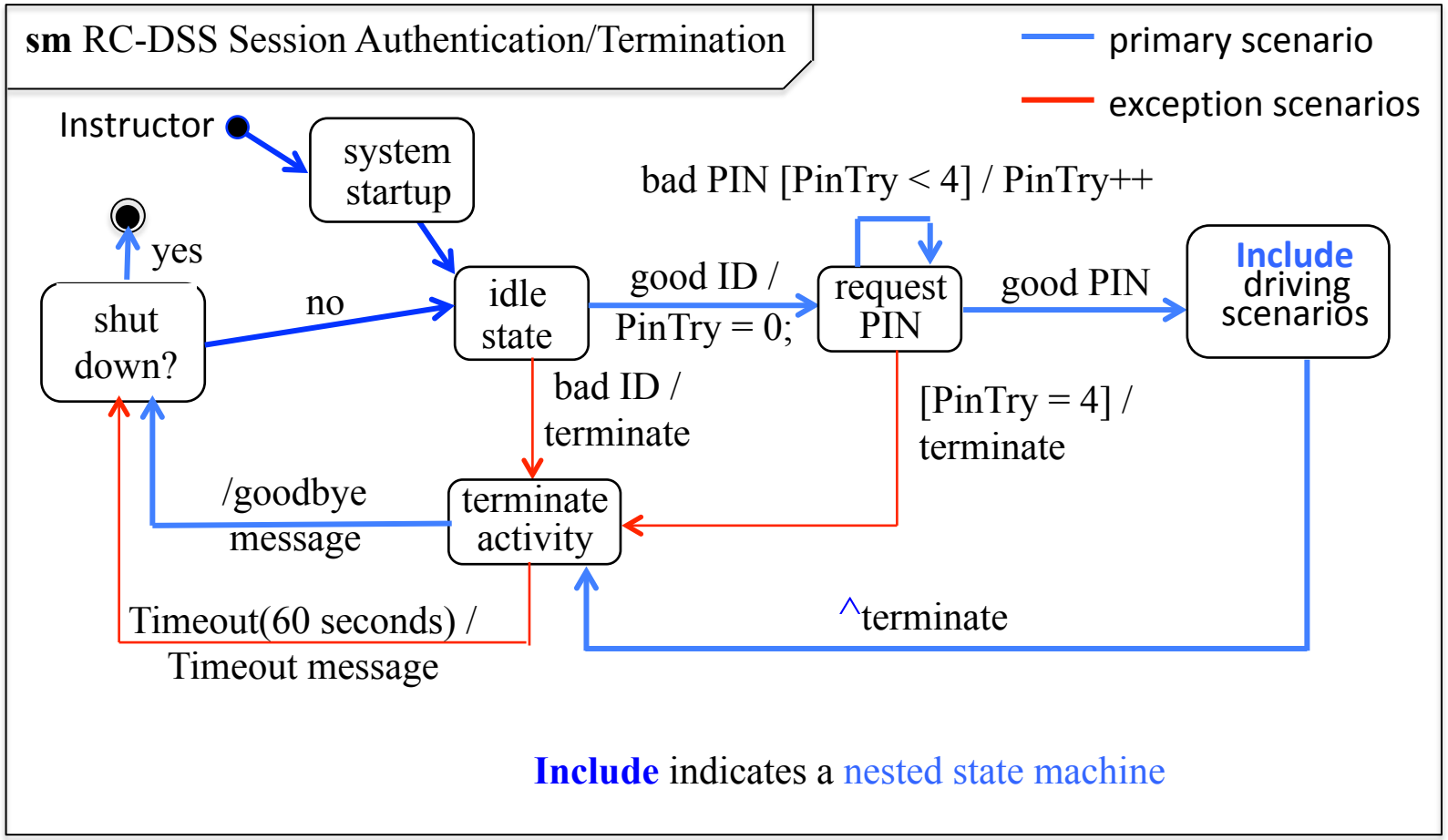
A1 sponsor prepares request
A2 DSS server scans request
A3 bad message notification
A4 Admin. processes request
A5 enrollment OK; sponsor notified
A6 Admin enters wait state with timeout period
A7 registrar processes request
A8 enrollment OK; sponsor and admin. notified
A9 enrollment failed; sponsor and admin. notified

A SysML state machine diagram



Student enters ID; Instructor enters PIN and shutdown response

A SysML state machine diagram



Student enters ID; Instructor enters PIN and shutdown response

Documenting structural and behavioral diagrams

- Diagrammatic views of system architectures must be accompanied with explanatory textual descriptions that are (or should be):

  correct, complete, consistent, concise, and clear

  for the intended audience

Note

- Much of the behavior of modern systems is implemented in software – running on computers with software that provides interfaces to other hardware elements
  - Because of the malleability and flexibility of software as compared to hardware
- However, concurrent development of hardware and software is a continuing issue
  - Because of the different approaches used to develop hardware and software
  - This issue will be addressed in Webinars 4 and 5:
    - ✓System Design and System Implementation

Agenda: System Architecture Definition

- Brief review of training webinars 1 & 2
- System definition
- Purpose of architecture definition
- Notations for architecture definition
- Verifying and validating architecture definitions
- Roles played by systems engineers in architecture definition
- A brief preview of training webinar 4

Formal versus informal V&V

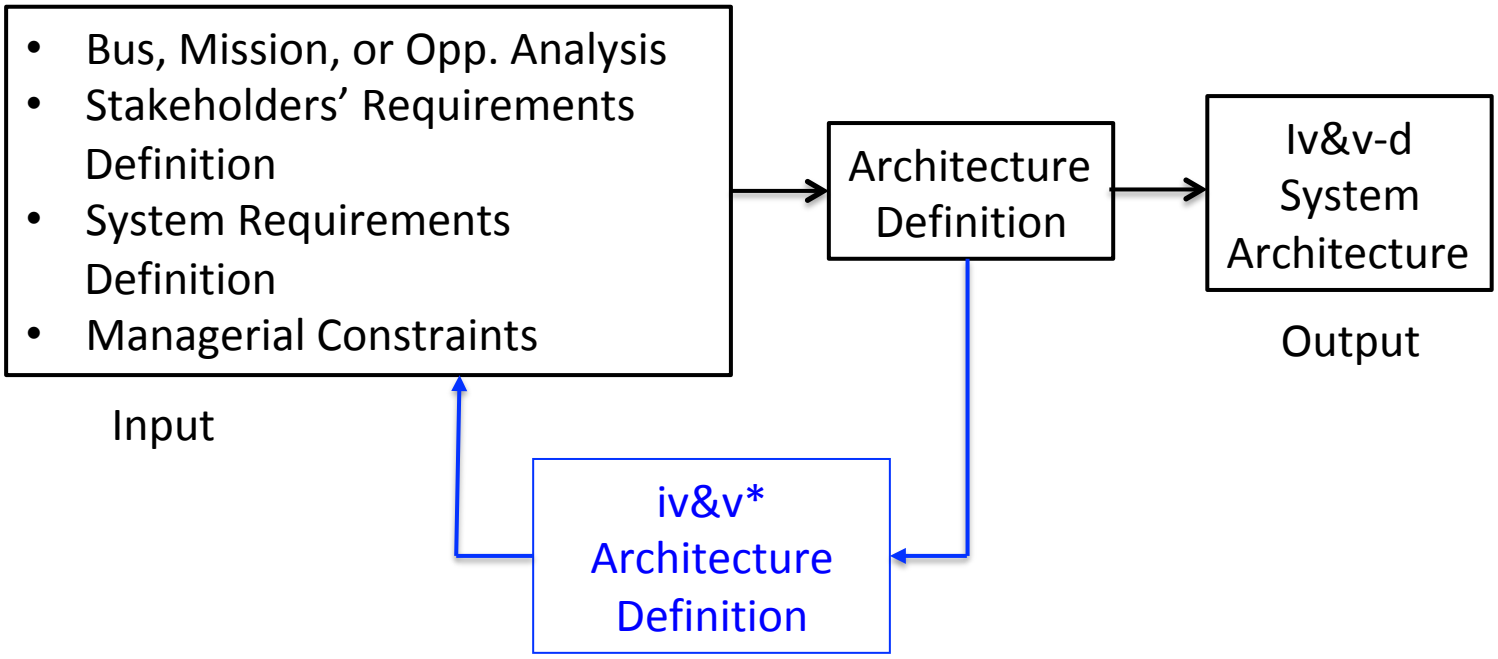
- Formal V&V is accomplished by V&V specialists
  - Typically in a separate department
  - Or in a separate organization for IV&V
  - And is applied to a deliverable system, a subsystem, or a system element
- Informal V&V (iv&v) is accomplished by SEs, disciplinary engineers, specialty engineers, and domain experts
    - Informal does not mean haphazard
    - Techniques include traceability analysis, simulations, prototyping, inspections, scenarios, and reviews

Informal V&V of system architecture (iv&v)

- The purpose of verifying the system architecture definition is *to determine the degree to which** the architecture satisfies the conditions and constraints placed on it by other work products, policies, procedures, and regulations
- The purpose of validating the system architecture definition is to *determine the degree to which** the architecture, as documented, is suitable for use in the intended ways by the intended users of the architecture definition in the intended context

* V&V outcomes are not binary

The iv&v process



*iv&v: informal verification and validation

Roles of systems engineers in architecture definition

- To plan and coordinate development of the architecture definition*
- To participate in and/or facilitate development of alternative architectures and selection of the chosen architecture
- To participate in and/or facilitate architecture definition of system structure and behavior for the selected architecture
- To participate in and/or facilitate informal verification and validation of the system architecture
- To facilitate: to make an activity or action easier
  - A complex system may have multiple subsystems, each with a distinct architecture
  - And perhaps involving different contractors, subcontractors, and vendors

*Technical management processes are addressed in 15288 Clause 6.3

Agenda: System Architecture Definition

- Brief review of training webinars 1 & 2
- System definition
- Purpose of architecture definition
- Notations for architecture definition
- Verifying and validating architecture definitions
- Roles played by systems engineers in architecture definition
- A brief preview of training webinar 4
- Comments and questions

Training webinar 4
Design Definition

- Design definition is concerned with decomposing the system architecture into subsystems (perhaps) and system elements
- Then defining the elements and interfaces in sufficient detail to provide the basis for system implementation
- The system design webinar will be presented and recorded on January 10, 2019

Questions? Comments?

- Contact information:

  dickfairley@gmail.com

  john.clark@incose.org

  gabriela.coe@incose.org