INCOSE Training Webinar
Design Definition

Developed and presented by

Richard E. (Dick) Fairley

dickfairley@gmail.com

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

Clause 6.4.5 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

and

- **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
- But different process models for system development typically present the technical processes in various iterated, overlapped, and repeated ways

  and concurrently and recursively as required

- The tao of systems engineering: first learn each of the technical processes

  Then combine and integrate them in various ways, as needed

  They are tools in your SE toolbox

Another primary reference

Systems Engineering for Software-enabled Physical Systems

Physical systems: embedded, IoT, cyber-physical, C2 systems, transportation, health care, 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

- The Systems Engineering Handbook, Competency Framework, and other materials are available for free downloading from the INCOSE Store:

  https://connect.incose.org/pages/store.aspx
- SEBoK is universally accessible at no cost

Note

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

Agenda for Design Definition

- Brief review of training webinars 1, 2, and 3
- System definition
- Purpose of design definition
- Some key elements of design definition
- Design definition techniques
- Verifying and validating design definitions
- Roles played by systems engineers in design definition
- How much is enough?
- A brief preview of training webinar 5
- Comments and questions

Webinar 1: 15288 Clause 6.4.1
Business or Mission Analysis

Clause 6.4.1.1: Purpose

define the business or mission problem or opportunity,

characterize the solution space,

and determine potential solution class(es) that could address a problem or take advantage of an opportunity

An Input, Process, Output approach was presented

Webinar 1: 15288 Clause 6.4.2 Stakeholder Needs and Requirements definition

Clause 6.4.2.1: Purpose

define the stakeholder requirements for a system

that can provide the capabilities

needed by users and other stakeholders in a defined environment

An Input, Process, Output approach was presented

Categorizing and prioritizing stakeholders' requirements

Stakeholders' requirements are categorized as:

- 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

Stakeholders' requirements are prioritized as:

- Essential, Desirable, and Optional

Stakeholder features, quality attributes, and design constraints drive system requirements definition, system design, and system implementation

Webinar 2: 15288 Clause 6.4.3
System Requirements Definition

Clause 6.4.3.1: Purpose

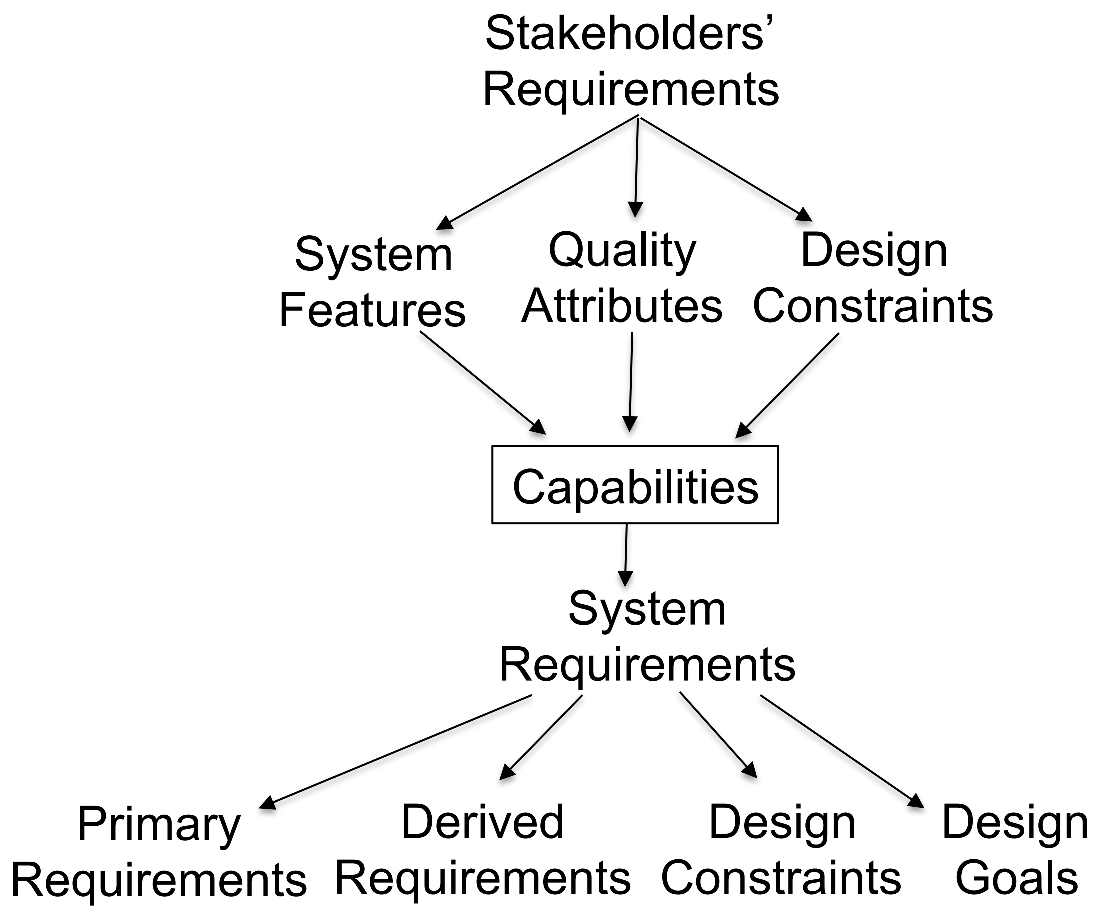
transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution

that meets the operational needs of the user

System requirements can be categorized as primary requirements, derived requirements, design constraints, and design goals

An Input, Process, Output approach is presented

A requirements taxonomy



System-level quality attributes are categorized as design constraints in this taxonomy

A note on terminology

- System requirements are often categorized as functional and non-functional requirements
- The term "behavioral attributes" is sometime used as a synonym for functional requirements
- The term "quality attributes" is sometimes used as a synonym for a category of non-functional requirements
    - Quality attributes are included as a category of design constraints in the system requirements taxonomy
    - Other categories of design constraints might include requirements to include certain legacy elements or to develop an architecture that will support a product line of systems

Webinar 3: 15288 Clause 6.4.4 System Architecture Definition*

Clause 6.4.4.1: Purpose

generate system architecture alternatives

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

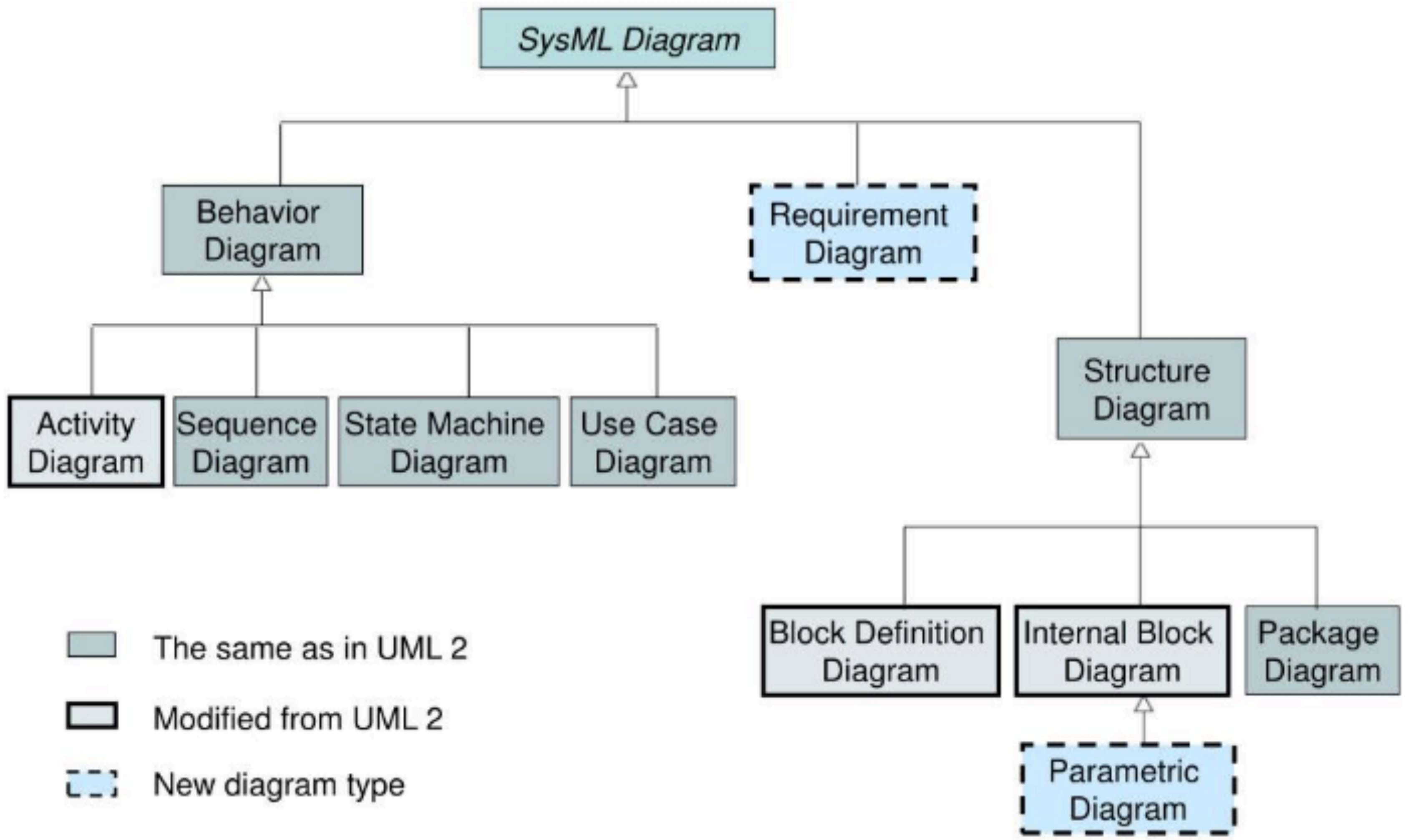
express the selected architecture(s) in a set of consistent views

Multiple views

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

  See sysml.org and omgsysml.org

SysML diagrams



Agenda: Design Definition

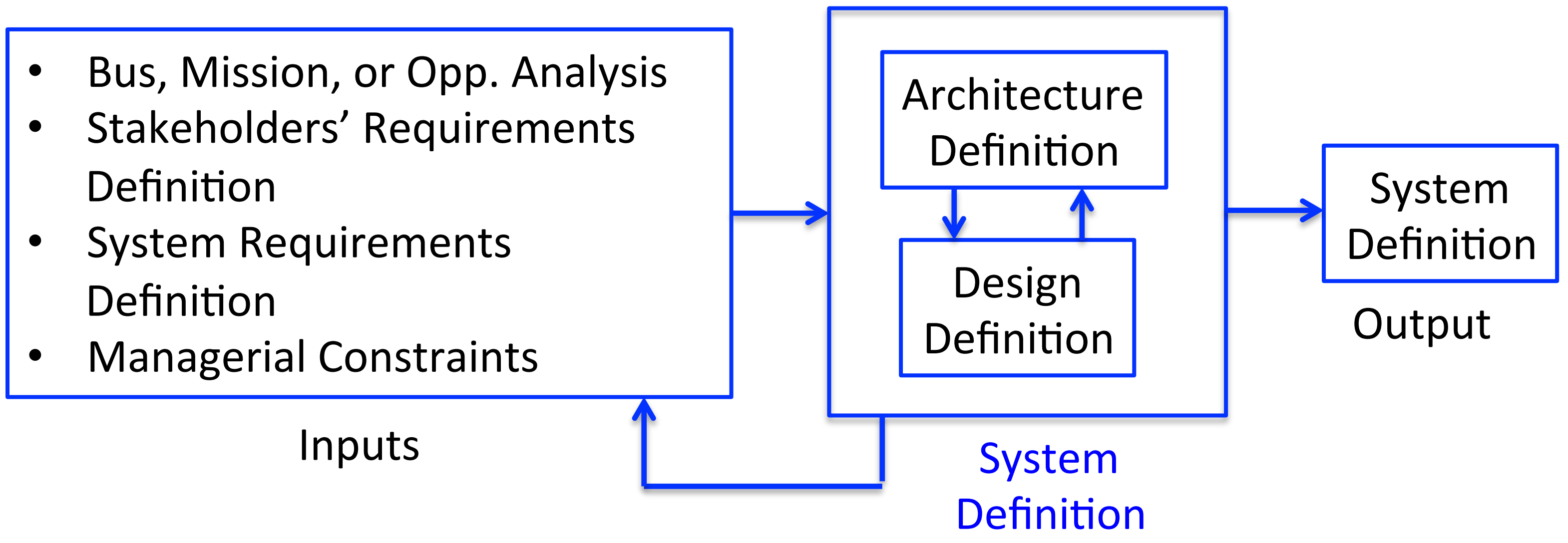
- Brief review of training webinars 1, 2, and 3
- System definition
- Purpose of design definition
- Some key elements of design definition
- Design definition techniques
- Verifying and validating design definitions
- Roles played by systems engineers in design definition
- How much is enough?
- A brief preview of training webinar 5
- Comments and questions

System definition

- System definition includes the system architecture definition and the system design definition
- 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.
- 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 definition

Design Definition: 15288 Clause 6.4.5

Clause 6.4.5.1: Purpose

provide sufficient detailed data and information about the system and its elements

to enable the implementation consistent with architectural entities

as defined in models and views of the system architecture

15288 Clause 6.4.5.1 Notes

- Note 1

  "... design focuses on compatibility with technologies and other design elements and feasibility of construction and integration."

  NOTE 2

  "... Design provides the 'implement-to' level of the definition, such as drawings and detailed design descriptions."

- NOTE 3

  "This process provides feedback to the system architecture to consolidate or confirm the allocation, partitioning and alignment of architectural entities to system elements that compose the system."

Two levels of design definition

- Logical design: concerned with defining the interconnections, interfaces, and flows among system elements plus the interfaces to and from the system environment
- Physical design: concerned with specifying the detailed design of hardware and software elements plus the operations performed by humans (i.e., manual operations)
    - and the interfaces and interactions among hardware, software, and humans

Software design

- Software design is detailed logical design
  - Includes detailed design of algorithms, data structures, interfaces, and exception handling
- Detailed design of software is often integrated with construction and integration of software elements
  - Malleability of software is a mixed blessing
  - Cautious advice: delay binding as long as possible (but no longer)
- Software engineers sometimes distinguish logical design and physical design of software, where physical design specifies the detailed layout of data structures and data repositories

Architectural design of software (structure and behavior) must precede detailed design and construction

Physical design

- Physical design of hardware elements and interfaces may include parameters such as:
  - Mass, volume, and dimensions;
  - power consumption and heat dissipation;
  - ruggedness and EMI resistance;
  - cables and cable specifications;
  - design of hardware and software connectors; and
  - physical housing and mounting details.
- Details must be sufficient for fabrication or procurement by others

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensure the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the *system* definition

Design decomposition criteria

Architectural elements should be recursively decomposed into system elements until:

- Areas of complexity and uncertainty are resolved
- Design details of interfaces provided by and needed by each system element can be specified
- Preconditions and post-conditions for using each element can be specified
- Implementation technologies have been evaluated and confirmed for feasibility
- Buy vs. build decisions for the system elements can be made; elements to be procured are black boxes
- Implementation of each system element will require one or two engineers from one or two different disciplines

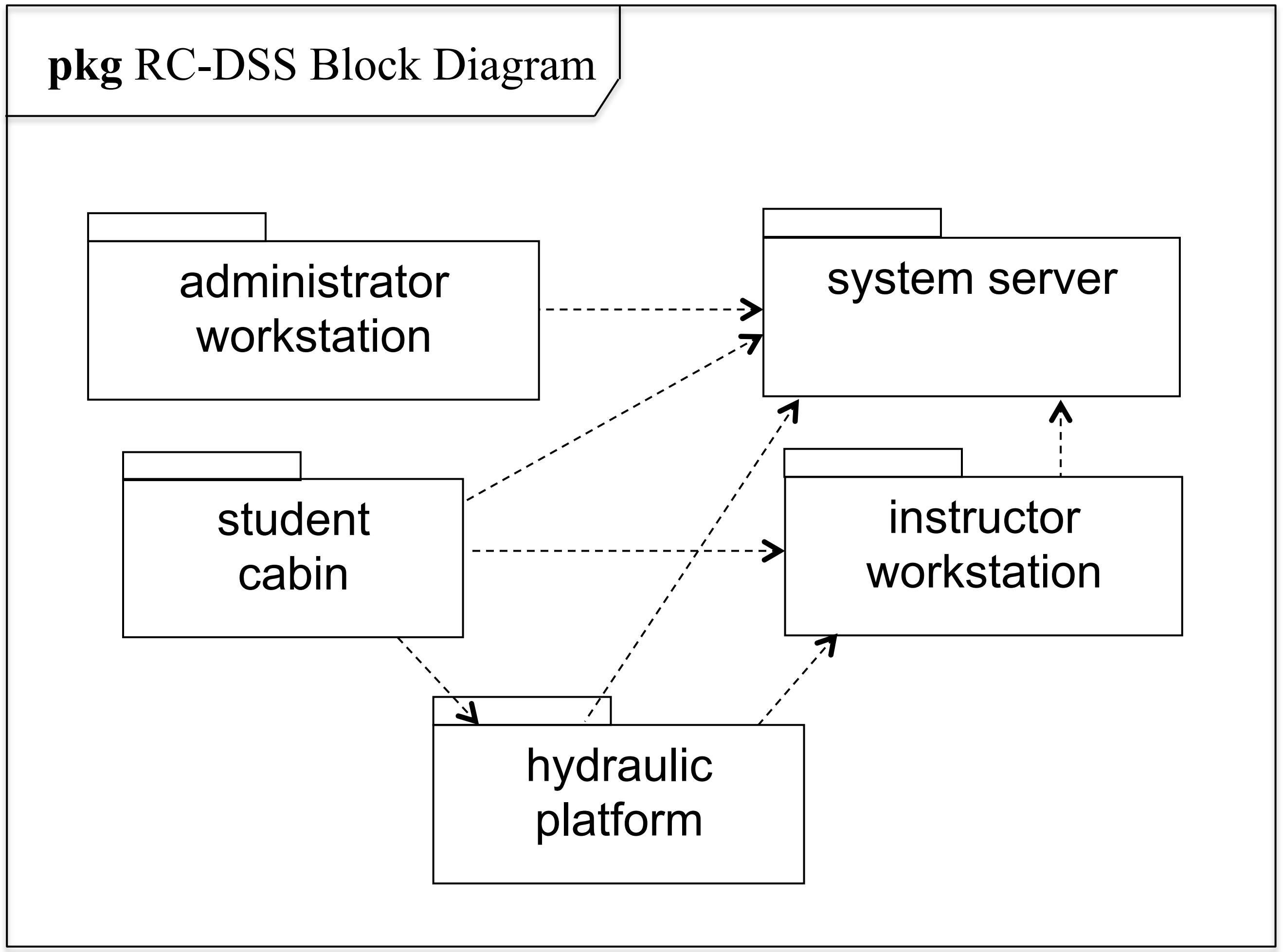
Additional design definition criteria (technical management issues)

- Infrastructure, tools, resources, and technologies needed to implement system elements have been identified
- Kinds and numbers of disciplinary engineers and specialty engineers have been identified as needed to implement, procure, integrate, and verify and validate system elements, subsystems, and system
- Sources and availability of infrastructure, tools, resources, and engineers have been identified
- Cost and schedule estimates for system implementation, verification, validation, transition, and delivery have been updated with confidence, and reconciled as necessary
- The risk management plan has been updated based on the satisfactory/ unsatisfactory results of applying these and the decomposition criteria

A case study: The RC-DSS SysML package diagram

Legend:
Dependency

Containment



The package diagram is the generic SysML structure diagram for containment

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensure the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that will provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the system definition

SysML* block diagrams

Two kinds of SysML blocks are used to specify a hierarchical system architecture

1. Block definition diagrams

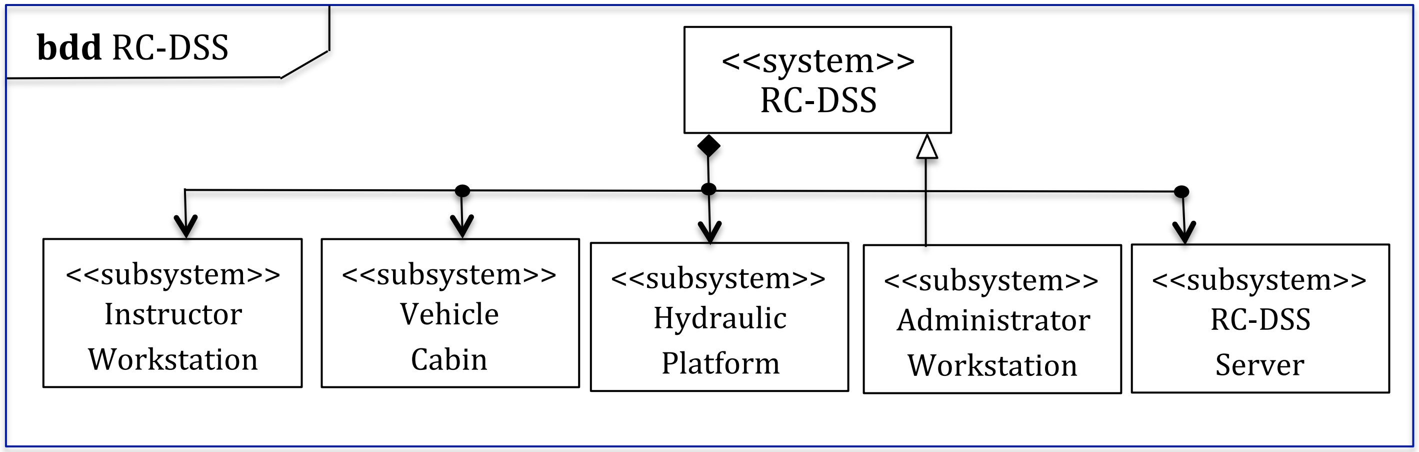
   bdd blocks are black boxes that specify input and output flows and connections to other blocks

2. Internal block diagrams

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

*sysML.org and omgsysML.org

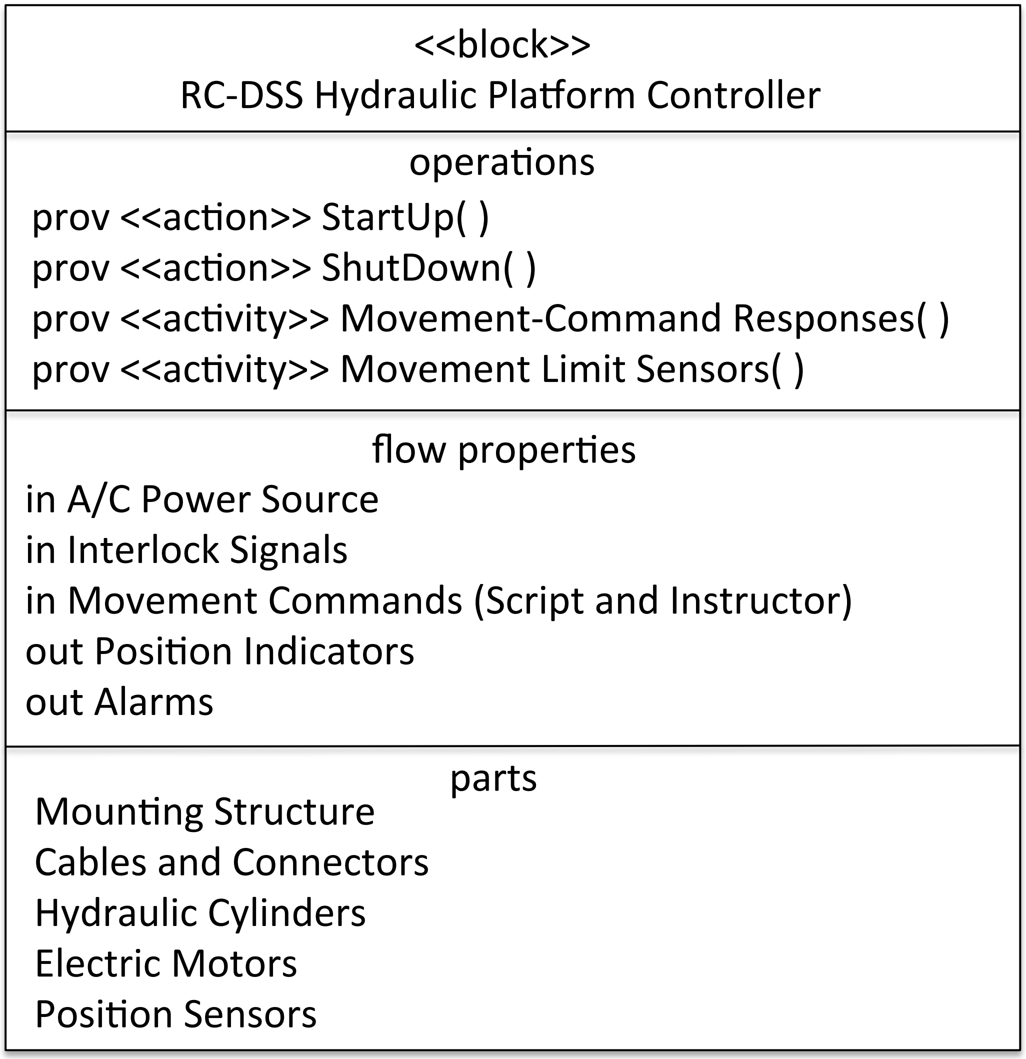
Top level RC-DSS **bdd** system structure


bdd RC-DSS
<<system>> RC-DSS
<<subsystem>> Instructor Workstation
<<subsystem>> Vehicle Cabin
<<subsystem>> Hydraulic Platform
<<subsystem>> Administrator Workstation
<<subsystem>> RC-DSS Server

Legend:

connector	●
composition	◆
aggregation	△

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



<<block>>
RC-DSS Hydraulic Platform Controller

operations

prov <<action>> StartUp( )
prov <<action>> ShutDown( )
prov <<activity>> Movement-Command Responses( )
prov <<activity>> Movement Limit Sensors( )

flow properties

in A/C Power Source
in Interlock Signals
in Movement Commands (Script and Instructor)
out Position Indicators
out Alarms

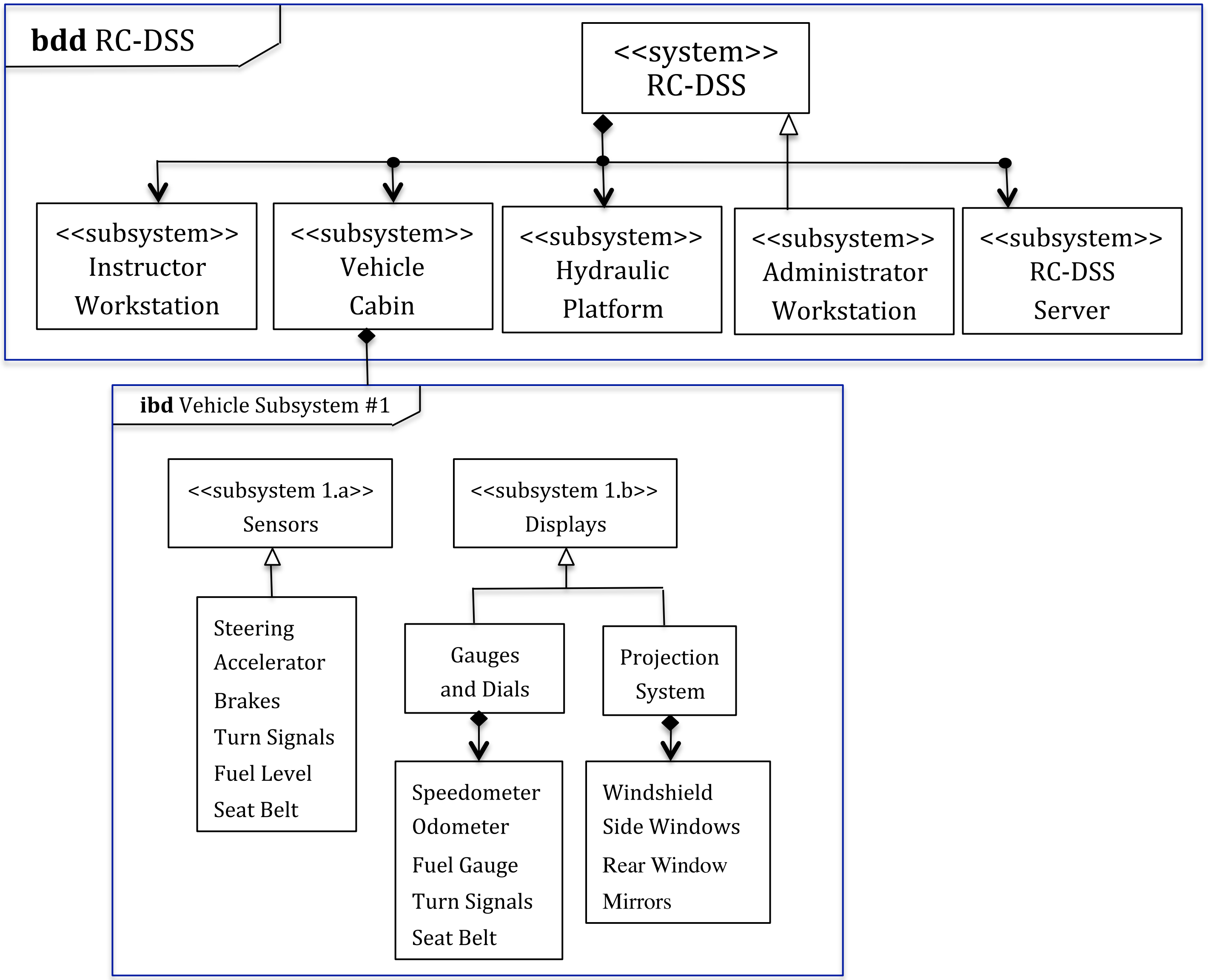
parts

Mounting Structure
Cables and Connectors
Hydraulic Cylinders
Electric Motors
Position Sensors

A partial RC-DSS **bdd-ibd** system structure

Legend:

connector	●
composition	◆
aggregation	△

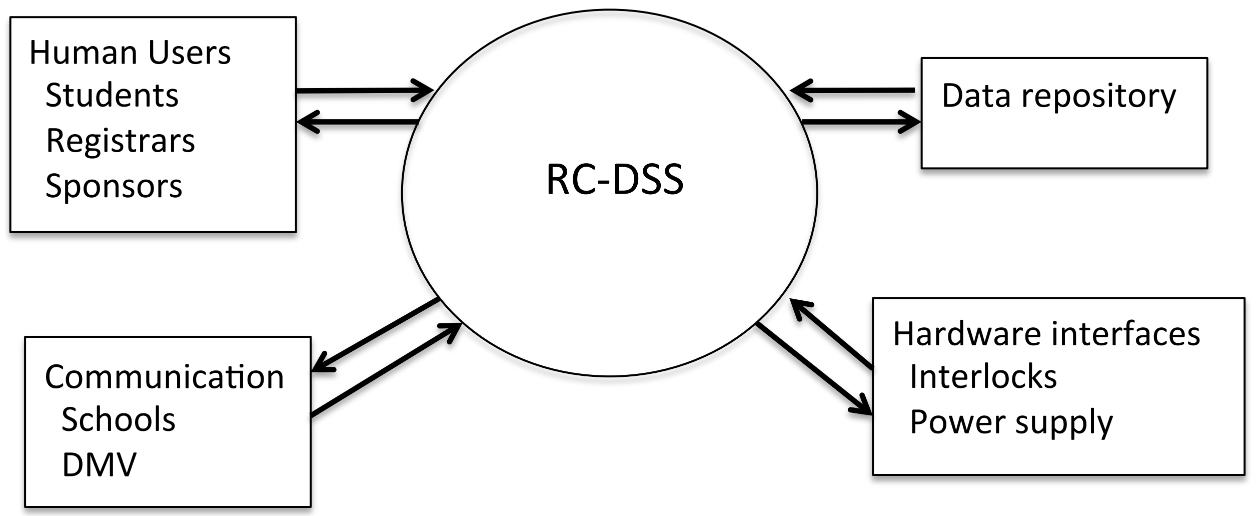


bdd RC-DSS
<<system>> RC-DSS
<<subsystem>> Instructor Workstation
<<subsystem>> Vehicle Cabin
<<subsystem>> Hydraulic Platform
<<subsystem>> Administrator Workstation
<<subsystem>> RC-DSS Server
ibd Vehicle Subsystem #1
<<subsystem 1.a>> Sensors
<<subsystem 1.b>> Displays
Steering Accelerator Brakes Turn Signals Fuel Level Seat Belt
Gauges and Dials
Projection System
Speedometer Odometer Fuel Gauge Turn Signals Seat Belt
Windshield Side Windows Rear Window Mirrors

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensure the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that will provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the system definition

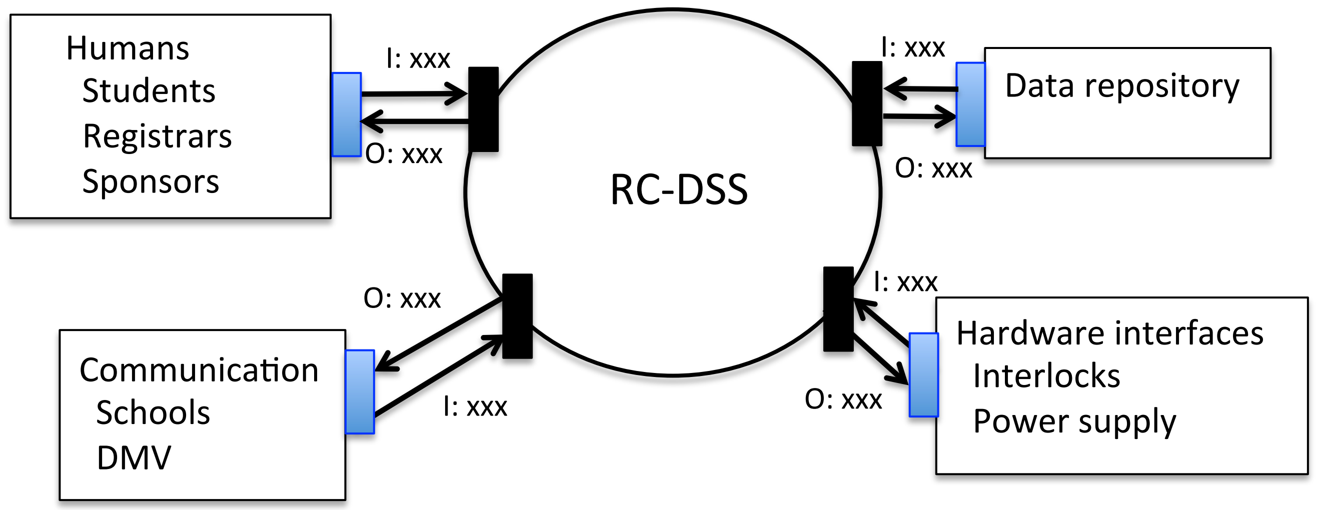
RC-DSS requirements context diagram



Interfaces typically include: hardware, software, (data and communication), and humans

Note: Students, registrars, and sponsors are users
Instructors, technicians, and system administrators are operators

RC-DSS system interface design diagram



- Terminator inputs and outputs are documented in an Interface Design Document (IDD)

  Or an Interface Control Document (ICD) if separate groups are developing the system
- The IDD or ICD is placed under configuration management

Contents of an Interface Design Specification

Design specifications for each interface include:

- An interface diagram
  - each interface depicted and identified by name
  - Details specified using packages and blocks
- Hardware, software, and material interfaces
- Interface coordination
  - concurrent or sequential execution
  - method of synchronization
  - communication protocol to be used
  - priority level of the interface

Some examples of interface design specifications

- For hardware interfaces
  - Mode selectors
  - Monitoring sensors and alarms
  - Voltage and power levels
  - Connector configurations and pin assignments
  - Cable specifications
- For data interfaces:
  - names, units of measure, ranges, limits, accuracy, precision, data rate, pre and post conditions, exception handling methods
- For material interfaces:
  - Flow rate and viscosity sensors
  - Pressure and temperature sensors
  - Heat sinks and heat dissipation rate sensors
  - Alarm interfaces

Note

- The majority of system malfunctions and system failures are caused by interface defects
  - both during implementation and operation
- Two kinds of defects are caused by:
  - Errors of commission
  - Errors of omission
- Some causes of interface defects:
  - Parameter mismatches
  - Emergent behavior
  - Non-deterministic behavior

Some well-known examples

1. The Ariane 5 rocket launch: data passed through the interface from the gyroscopes to the guidance software exceeded the value limit
   - Loss of a 500M USD payload; partial loss of a ten year, one billion dollar program by the European Space Agency
2. The Therac-25 x-ray machine: non-deterministic race condition
   - Faulty software; no hardware interlocks
   - Human injury and loss of several human lives
3. Mars Polar Lander: faulty sensor data
   - Premature engine cutoff caused by faulty sensor data
   - Loss of development time, expense, and Mars data
   - Loss of NASA (JPL) reputation

Why weren't these defect found during simulation and testing?

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensure the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that will provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the system definition

Advantages of buying

“Buying” includes procuring by purchase, by lease, or by license

Advantages of buying:

- Sooner availability (perhaps)
- Known quality attributes (perhaps)

Disadvantages of buying

- Locating and evaluating potential elements may take more time than fabricating
- Cost may be excessive as compared to fabrication
- Candidate elements may provide unwanted capabilities that will consume resources and provide unwanted capabilities that must be masked to prevent unintended use
- Building interfaces to fabricated elements and the system environment plus V&V may be inhibited by lack of sufficient documentation for the procured elements
- Failing to find acceptable candidate elements for procurement may result in delayed fabrication of those elements or descoping the requirements to match a candidate element that is available

Subcontracting

- Subcontracting may be used to obtain needed personnel or domain expertise
- Some issues:
  - Subcontract must be negotiated
  - Subcontractor must be managed
  - Subcontractor may lack needed skills or tools
  - Subcontractor must be provided with detailed requirements, design, interfaces, and other needed information
    - ✓Which may involve disclosure of proprietary information
  - Lack of subcontractor performance can cause project delay or project failure for high risk system elements

Building instead of buying

Advantages of building

- Locating and evaluating available elements is eliminated
- Requirements to be satisfied by the needed elements can be built into the system elements and modified as needed
- Uncertainty about functionality, behavior, and quality attributes may be reduced

The primary disadvantages of building are:

- Uncertainty of resulting quality attributes and performance parameters
- Time and expense to build compared to buying (perhaps)
- Resources, engineers, and schedule need to build the system elements

Additional considerations for procuring software

Sources of procured software include:

- o Private and public libraries
- o Open source
- o Vendors
- o Subcontracting for "bespoke" software

Additional considerations for procuring software:

- The viability of the supplying organization
- The compatibility of updates the supplying organization may make to acquired software elements
- The inability of the acquirer who buys to modify and make updates to the acquired software
- Discontinued support for the software

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensuring the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the system definition

Prepare comprehensive design documentation*

- The purpose of design definition is to provide a detailed road map for system implementation that includes
  - Mechanical drawings, schematics, blue prints
  - block definition and internal definition structural diagrams and behavioral diagrams (sequence, use case, activity, state)
  - Plus descriptive prose that explains and supports the iconic definitions

  And that is correct, complete, consistent, concise, clear, and integrative for the intended audience
- The intended audience includes disciplinary engineers, specialty engineers, verifiers, validators, transition and delivery engineers, and system maintainers

*sometimes referred to as the "build to" specification

Some key elements of design definition

1. Review and revise, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensure the continuity among them*
2. Recursively decompose the architectural elements into logical design elements
3. Specify the design details of the system elements and the interfaces among them
4. Specify the design details of the system elements that provide interfaces to and from the system environment
5. Conduct buy vs. build trade studies and make buy vs. build decisions
6. Prepare comprehensive design documentation
7. Informally verify and validate the *system* definition

Verification and validation (V&V)

- Verification is concerned with *determining the degree to which* a work product satisfies the conditions and constraints placed on it by other work products, policies, procedures, and regulations
- Validation is concerned with *determining the degree to which* a work product, as specified and documented, is suitable for use by the intended users when used in the intended ways

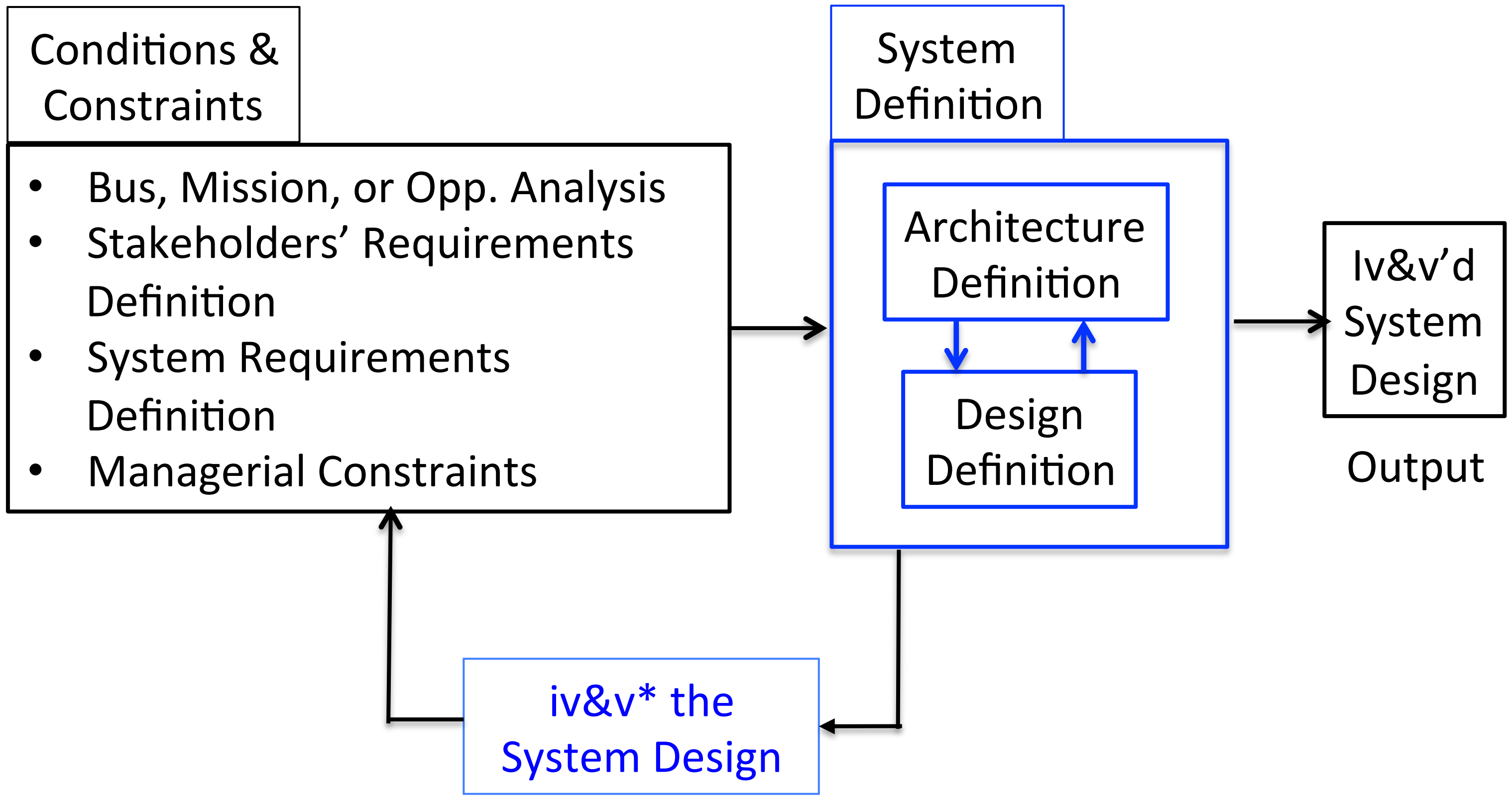
Formal versus informal V&V

- Formal V&V is accomplished by V&V specialists
  - Typically in a separate department
  - Or in a separate organization for IV&V
  - And is typically applied to a deliverable system, subsystem, or 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, technical analysis, simulations, prototyping, inspections, peer reviews, and manual and automated scenarios

iv&v of the system design

- System design includes the architecture definition and the design definition
- The architecture definition has (probably) be previously verified and validated in an informal manner
  - o But has (probably) been revised during design definition
  - o And should be re-evaluated along with iv&v of the design definition

The iv&v process



iv&v: informal verification and validation

Agenda: Design Definition

- Brief review of training webinars 1, 2, and 3
- System definition
- Purpose of design definition
- Some key elements of design definition
- Design definition techniques
- Verifying and validating design definitions
- Roles played by systems engineers in design definition
- How much is enough?
- A brief preview of training webinar 5
- Comments and questions

Roles of systems engineers in design definition

- Planning and coordinating development of the design definition*
- Participating in and/or facilitating:
  - Reviewing and revising, as necessary, the stakeholders' requirements, system requirements, and system architecture to ensure correctness, completeness, consistency, conciseness, and clarity of each, *and ensuring the continuity among them*
  - Recursively decomposing the architectural elements into design elements
  - Specifying the design details of the interfaces among system elements
  - Specifying the design details of the system elements that will provide interfaces to and from the system environment
  - Conducting buy vs. build trade studies and making buy vs. build decisions
  - Preparing comprehensive design documentation
  - Informally verifying and validating the system definition

*See 15288 Clause 6.3 for Technical Management Processes

Note

- Systems engineers coordinate and facilitate decomposition of the system architecture and detailed design of system elements, interconnections, and interfaces
    - Detailed design of system elements is accomplished by disciplinary engineers and specialty engineers
- Systems engineers may also apply their background disciplines to participate as disciplinary or specialty engineers
    - But they are playing different roles if they do so

Agenda: Design Definition

- Brief review of training webinars 1, 2, and 3
- System definition
- Purpose of design definition
- Some key elements of design definition
- Design definition techniques
- Verifying and validating design definitions
- Roles played by systems engineers in design definition
- How much is enough?
- A brief preview of training webinar 5
- Comments and questions

Some questions to consider

1. How much system design, documentation, and informal verification and validation should we do?

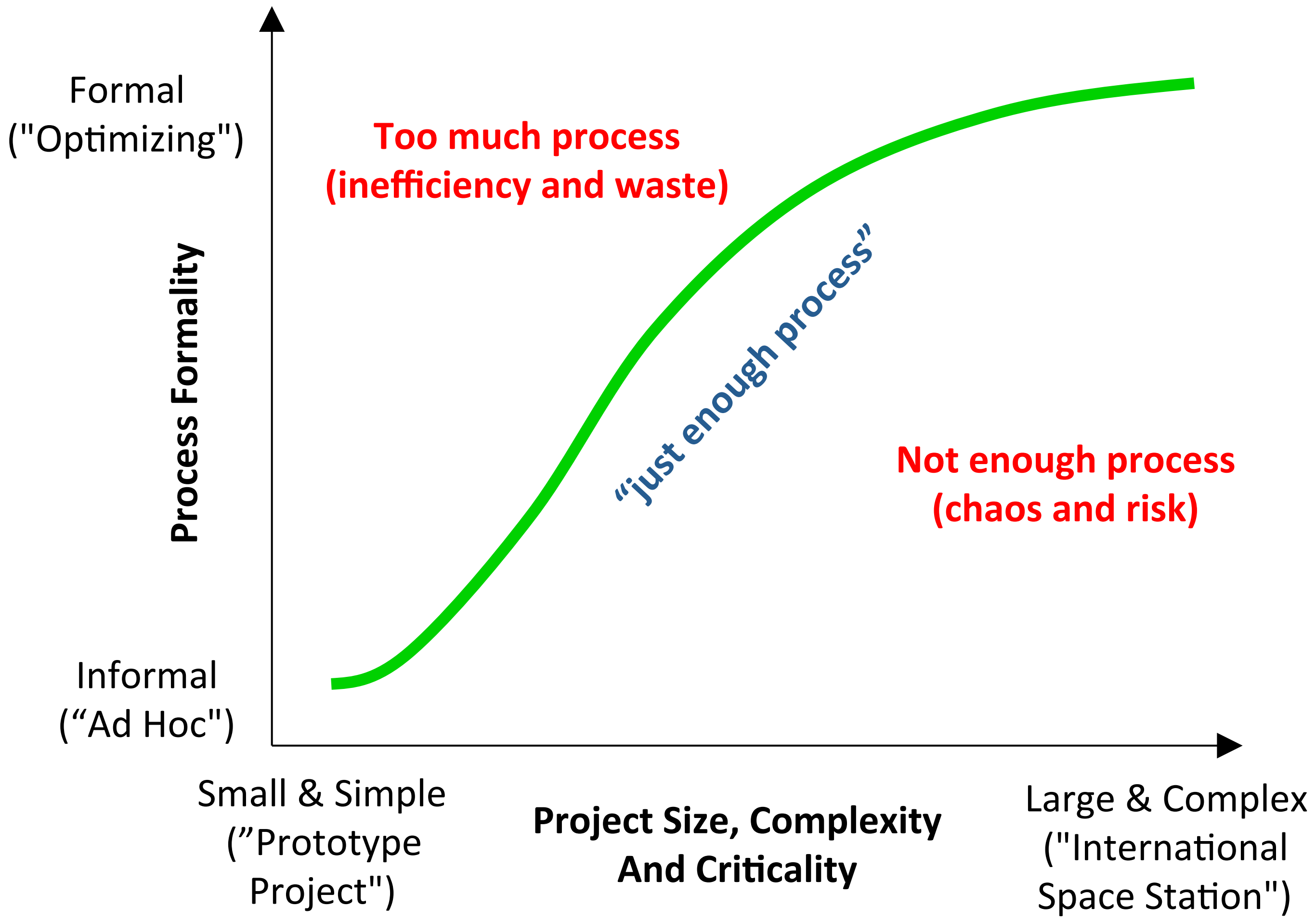
   Factors to be considered:

   - Type, purpose, and criticality of the system
   - Regulatory or legal requirements
   - Schedule, budget, resource, and personnel constraints

2. Is iv&v a one-time process?
3. What risk is incurred if we don't do enough?
4. What risk is incurred if we do too much?

The Goldilocks principle: not to little and not to much but just right

How much process is enough?
(just enough process)



Agenda: Design Definition

- Brief review of training webinars 1, 2, and 3
- System definition
- Purpose of design definition
- Some key elements of design definition
- Design definition techniques
- Verifying and validating design definitions
- Roles played by systems engineers in design definition
- How much is enough?
- A brief preview of training webinar 5
- Comments and questions

Training webinar 5
System Implementation

- System implementation includes:
  - Implementation of system elements, and
  - Integration of system elements
- System implementation requires coordinated development and integration of hardware and software elements
  - An MBSE/M&SBSE approach will be presented
- The system implementation webinar will be presented and recorded on January 17, 2019

Questions? Comments?

- Contact information:
    - dickfairley@gmail.com
    - john.clark@incose.org
    - gabriela.coe@incose.org