Requirements Engineering, Part 2

Developed and presented by
Richard E. (Dick) Fairley
for INCOSE

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

Primary references for this webinar (1)

Clauses 6.4.3 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

Primary references (2)

Systems Engineering for Software-enabled Physical Systems

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

To be published by Wiley in xx, 2019

Reference for this webinar:

Section 5.2: Capabilities-based system development

Chapter 7: System requirements definition

Other reference materials

1. INCOSE Systems Engineering Handbook
2. INCOSE Systems Engineering Competency Framework
3. INCOSE Guide for Writing Requirements

   https://connect.incose.org/Pages/Store.aspx

3. The Guide to the Systems Engineering Body of Knowledge

   https://sebokwiki.org

4. ISO/IEC/IEEE 29148:2011 Systems and software engineering Life cycle processes -- Requirements engineering

   https://www.iso.org/standard/45171.html

   https://standards.ieee.org/standard/29148-2011.html

Agenda: Requirements Engineering Part 2

- Brief review of Part 1
- Traditional approaches to developing system requirements
  - Define, categorize, and prioritize system requirements
  - Verify and validate system requirements
  - Role of systems engineers in traditional requirements engineering
- Capabilities-based requirements definition
  - Role of systems engineers in capabilities-based requirements engineering

Requirements Engineering
Review of Part 1

- Some terminology
- Business or mission analysis
- Stakeholder needs and requirements definition
- Brief intro to capabilities-based RE

Recording of and slides for Part 1 are on the INCOSE training Web site

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.

Part 1: Stakeholders' 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.

Categorizing 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

Prioritizing stakeholders' requirements

Stakeholders' requirements (features, quality attributes, and design constraints) are categorized as:

- Essential: must be included
- Desirable: should be included
- Optional: could be included
- Will nots: to control expectations
- Must nots: for safety, security, policies, regulations

Priorities of stakeholders' requirements influence the priorities of system requirements

Agenda: Requirements Engineering Part 2

- Brief review of Part 1
- Traditional approaches to defining system requirements
  - Define, categorize, and prioritize system requirements
  - Verify and validate system requirements
  - The role of systems engineers in traditional requirements engineering
- Capabilities-based requirements engineering
  - The role of systems engineers in capabilities-based requirements engineering

System requirements definition

System requirements definition*

"The purpose of the System Requirements Definition process is to transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user."

* ISO/IEC/IEEE Standard 15288, clause 6.4.3.1

defining system requirements is typically a recursive and iterative process that may include clarifying and revising the stakeholders' requirements (with stakeholder's consent), and perhaps revising the problem, mission, or opportunity statement

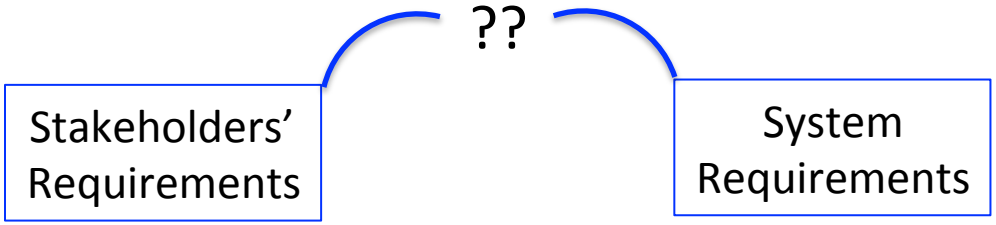
RE Part 2: system requirements definition

Input	Business or mission analysis, stakeholders' requirements definition, preliminary SEMP
Process	• Define, categorize, prioritize, and V&V system requirements • Bi-directionally trace system requirements to stakeholders' requirements definition
Output	• Verified and validated system requirements • Finalized SEMP

The challenge

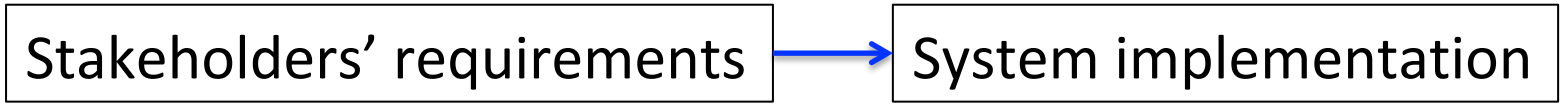
How to transform stakeholders' desired capabilities into a technical view of the system that will provide those capabilities?

how to bridge the gap from what stakeholders want and need, to defining what will be built??

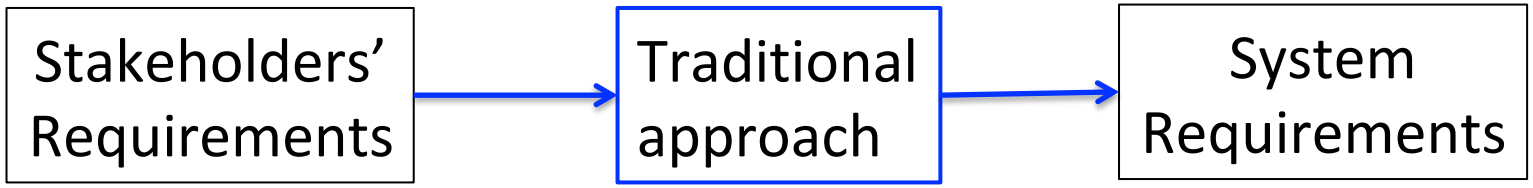


Two approaches

1. The “shortcut” approach:

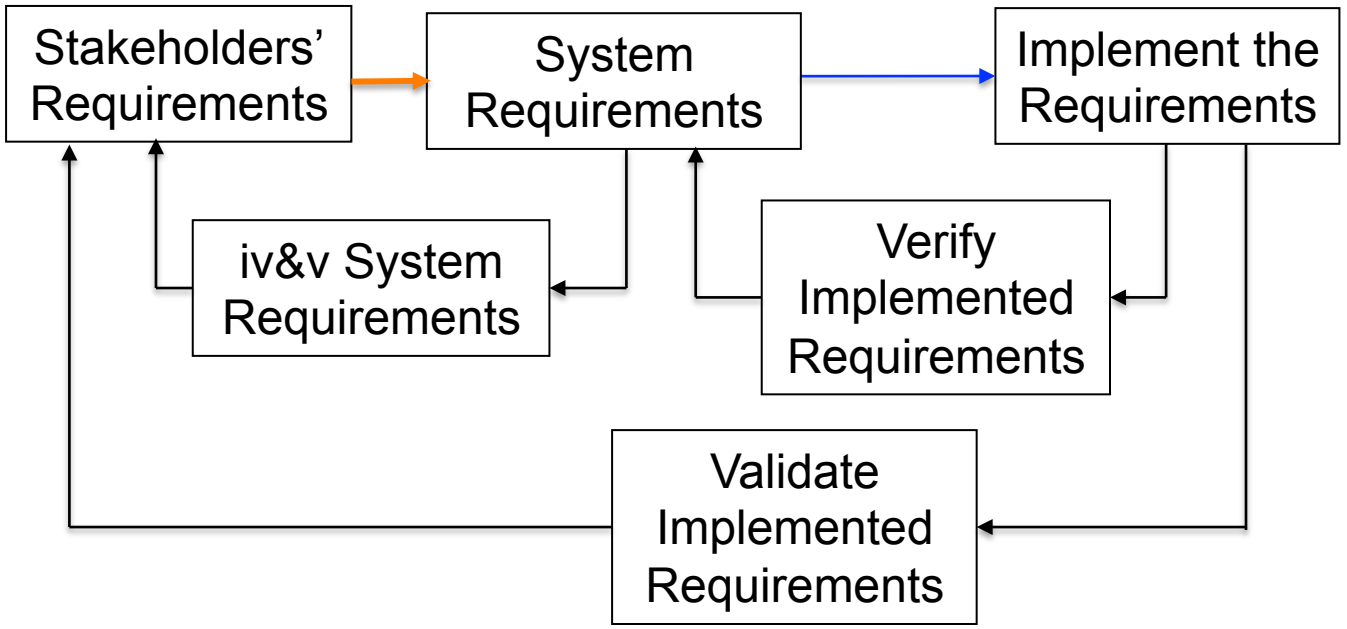


2. The traditional approach



The traditional approach is enabled by domain expertise and technical competence

Traditional system development

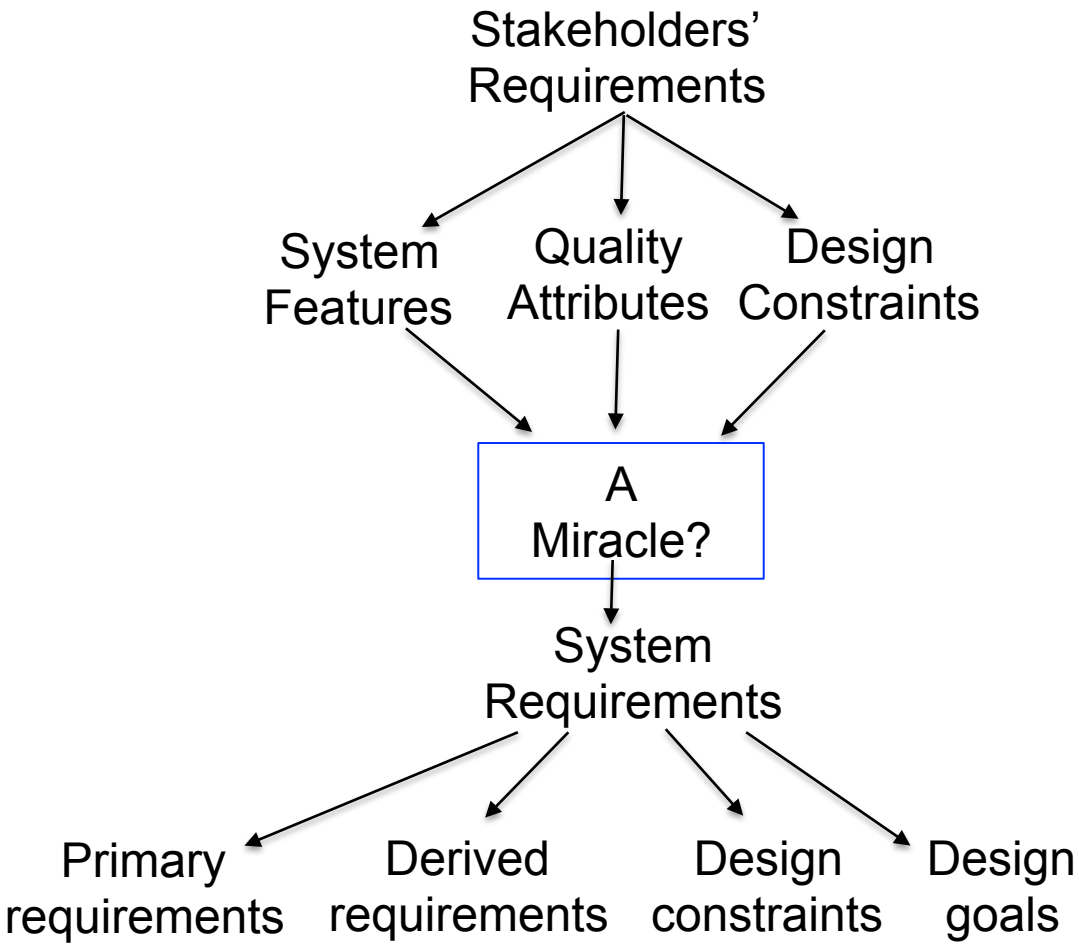


Architecture and design definition are omitted for simplicity of presentation

The role of systems engineers in traditional system requirements definition

- Define system requirements that satisfy the stakeholders' requirements
- Categorize and prioritize the system requirements
- Facilitate V&V of the system requirements
- Complete plans for the technical work to be accomplished by the disciplinary and specialty engineers
  - Initiated during stakeholders' requirements definition
  - Clause 6.3 of 15288 and 12207 covers technical management processes
  - Clause 6.3.1 covers the project planning process
    - ✓Technical planning is typically documented in a SEMP

A requirements taxonomy



Primary system requirements

- Primary system requirements are generated directly from stakeholders' requirements
- Example stakeholders' requirement:

  "The system will exhibit acceptable response time"

  Might become:

  "Response to Type 1 signals shall be not more than 100 milliseconds when the embedded computer in subsystem B is running at a load capacity of 80%"

  "Response to Type 2 signals shall be not more than 300 milliseconds when the embedded computer in subsystem B is running at a load capacity of 80%"

The load capacity of the embedded computer in subsystem B shall not exceed 80 percent under normal and exception conditions

Derived system requirements

- Derived system requirements are added to support primary requirements by augmenting and clarifying them
- Example
  - Stakeholders' requirement:

    The system must maintain records of transactions
  - A primary requirement:

    "The system shall have the ability to record and time stamp transactions"
  - Derived requirement:

    "Time stamping of transactions shall be provided by a clock function having specified accuracy, precision, and drift (TBD)"

Derived requirements are traced to parent requirements and verified to determine the degree to which they support their parent requirements in the intended ways

System design constraints

- Requirements state what a system should do without stating how the system should do it, mostly:

  Design constraints are design “must bes” stated in the system requirements

Requirements should be “design agnostic” to the extent possible

System design constraints (2)

- Some system design constraints are inherited from the stakeholders' design constraints
  - And some may be added by the system developer
- An example based on a stakeholders' design constraint:

  "The system shall provide an interface to the stakeholders' cloud-based Oracle data repository of operational parameters"
- An example added by the system developer:

  "the system requirements shall be stated in a manner that will not inhibit definition of a product line architecture"

Design constraints are bi-directionally traced to and from their sources

System quality requirements

- System quality requirements are categorized as design constraints or design goals
- Design constraints are objectively stated; design goals are subjectively stated
- Quality requirements are the primary determinants of system architecture
  - They are must-be design constraints and design goals

There may be multiple ways to realize system functionality, behavior, and performance
But fewer ways to realize quality attributes

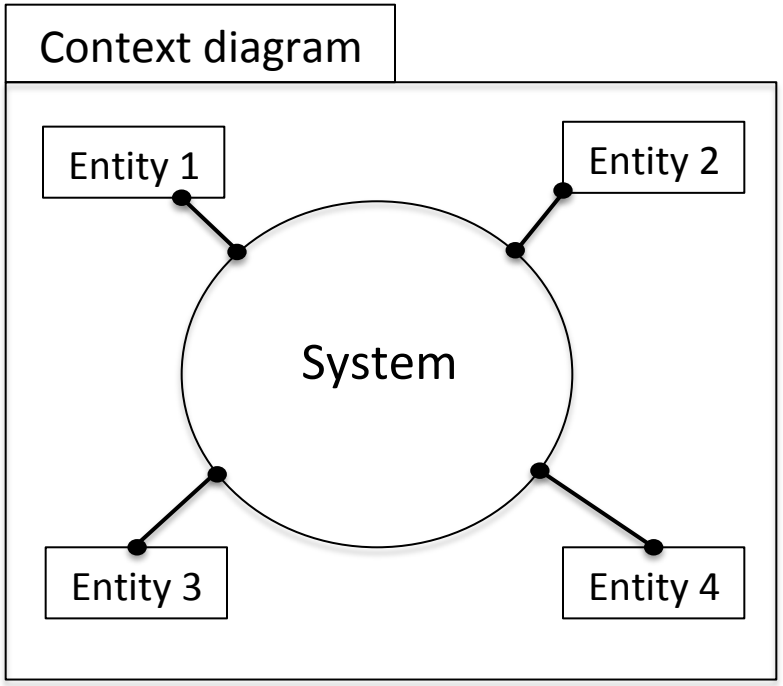
Some quality attributes

Visible to system users	Not visible to system users
Safety	Testability
Security	Modifiability
Reliability	Reusability
Availability	Configurability
Performance	Serviceability
Ease of use	Installability

- System quality attributes must be quantified and prioritized, to the extent possible
- They are often initially stated as design goals

System interface requirements

- Interface connections to the system environment are categorized as design constraints
  - The system environment usually has interface design constraints that cannot be altered



System design goals

- Primary requirements, derived requirements, and design constraints are stated in an objective manner
  - To permit verification of the implemented system requirements
- Design goals are subjectively stated requirements
  - Some will be inherited stakeholder's requirements
  - Some may be self-imposed design goals

Design goals are bi-directionally traced to the corresponding stakeholders' requirements

Examples of system design goals

Examples:

1. "the system must be easy to learn and easy to use"

   Qs: For whom? How much learning time? Easy to use for what tasks?

2. We (the system supplier) desire this system to be the safest system for use in the health care domain

   Qs: What kinds of health care systems? How will safety be measured? What are the competitors' safety metrics?

Design goals are bi-directionally traced to and from their sources

System design goals (2)

- Over time, some design goals may stated objectively after discussions, analysis, simulation, and prototyping
  - They are then move to one of the objective category
  - Especially the quality attributes
- Some design goals may be forever design goals
  - But they should not be ignored
  - Important design goals will influence design decisions
  - e.g., a subjectively stated safety goal may be more important than objectively stated requirements that might create safety threats

An example of converting subjectively stated design goals to objectively measurable requirements

- A new point of sale system for a large organization must be “easy to learn” and “easy to use” for the sales clerks
- A two-day training course and a half-day exercise were developed; 30 sales clerks were randomly selected
  - the system was deemed “easy to learn” and “easy to use” if 27 of 30 sales clerks successfully completed the training and exercise (90% success)
- One week later the 28 sales clerks, without pre-warning, were given a half-day refresher class and exercise to complete
  - The system was deemed to be easy to re-learn and easy to use if 26 of the 28 sales clerks successfully completed the training and exercise (~ 93% success)

Training and exercises were conducted remotely with local facilitators

Prioritizing system requirements

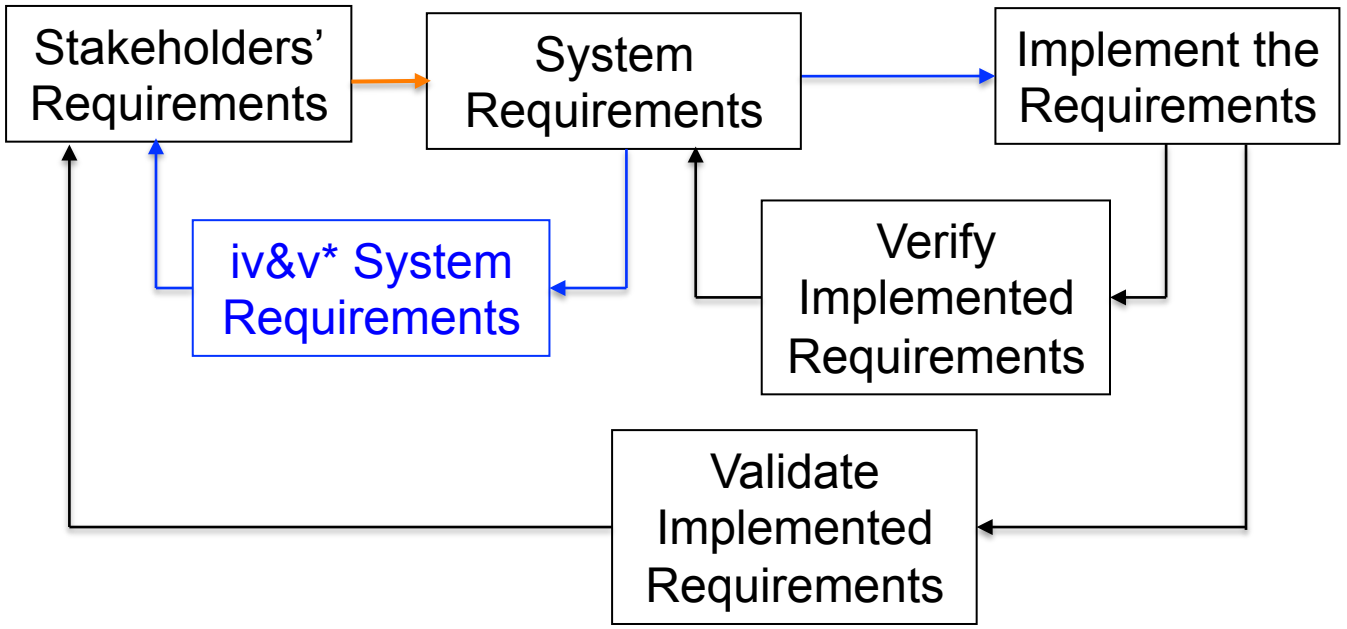
- Priorities among system requirements prioritizes are based on the priorities of stakeholders' requirements:
  - Essential
  - Desirable
  - Optional
  - Won't Do
  - Must Not Do

The role of systems engineers in traditional system requirements definition

- Define system requirements that satisfy the stakeholders' requirements
- Categorize and prioritize the system requirements
- Verifying and validating the system requirements
- Complete plans for the technical work to be accomplished by the disciplinary and specialty engineers
  - Initiated during stakeholders' requirements definition
  - Clause 6.3 of 15288 and 12207 covers eight technical management processes
  - Clause 6.3.1 covers the project planning process

Technical planning is documented in a SEMP

Traditional system development



Iv&v: informal verification and validation

Architecture and design definition are omitted for simplicity of presentation

Informal V&V of work products

- All intermediate work products of system development should be informally verified and validated
  - V&V should not be limited to the deliverable system, a subsystem, or a system element
- In general, to the purpose of verification is *to determine the degree to which** a work product satisfies the conditions and constraints placed on it by other work products, policies, procedures, and regulations.
- In general, the purpose of validation is to *determine the degree to which** a work product, as documented, is suitable for use by the intended users of the work product in the intended ways in the intended context.

* V&V outcomes are not binary

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, simulation, prototyping, inspections, and reviews

The V&V Annex

Some slides on formal and Informal V&V are in the annex to these slides

What traditionally happens next?

Typically, hardware and software requirements are separately allocated to hardware disciplinary engineers and software engineers

- Hardware-based system development:

  results in delayed software availability

- Software-based system development

  results in delayed hardware availability

Hardware-based and software-based system development are primary reasons systems are deliver late and have poor quality

Agenda: Requirements Engineering Part 2

- Brief review of Part 1
- Traditional approaches to defining system requirements
  - Defining, categorization, prioritization, and prioritizing system requirements
  - Verifying and validating system requirements
  - Role of systems engineers in traditional requirements engineering
- Capabilities-based requirements engineering
  - Role of systems engineers in capabilities-based requirements engineering

System requirements definition

System requirements definition*

"The purpose of the System Requirements Definition process is to transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user."

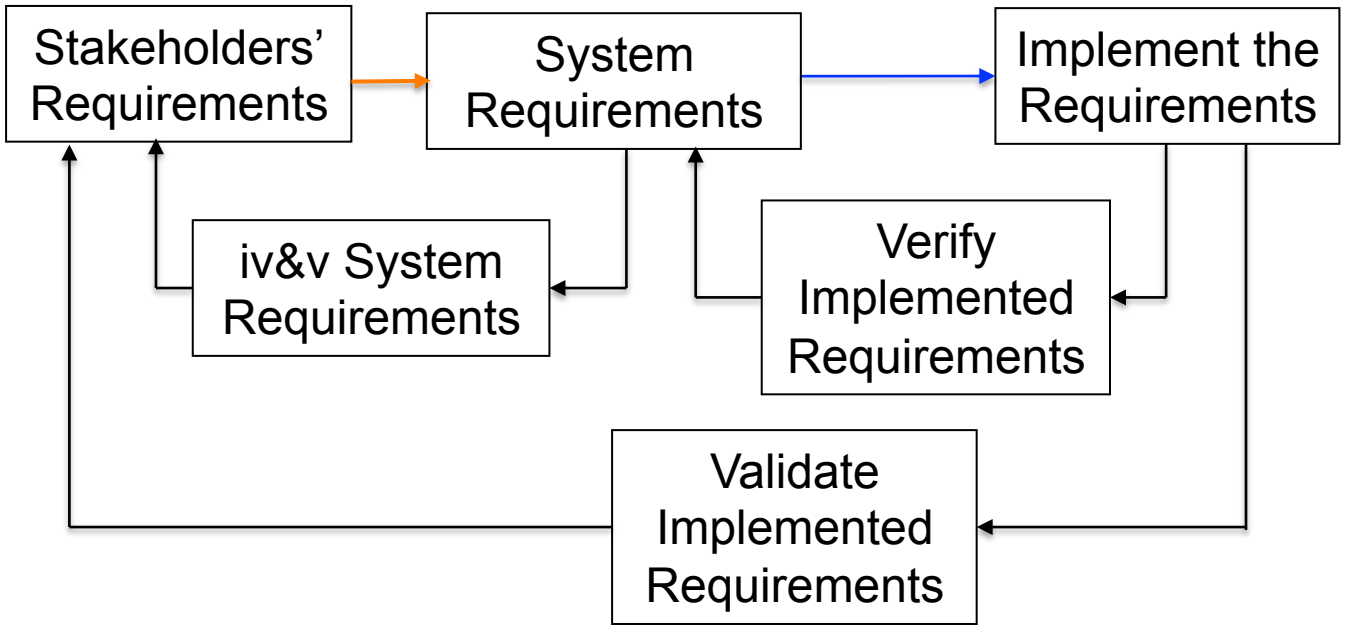
* ISO/IEC/IEEE Standard 15288, clause 6.4.3.1

A third approach

1. The “shortcut” approach
2. The traditional approach
3. The capabilities approach

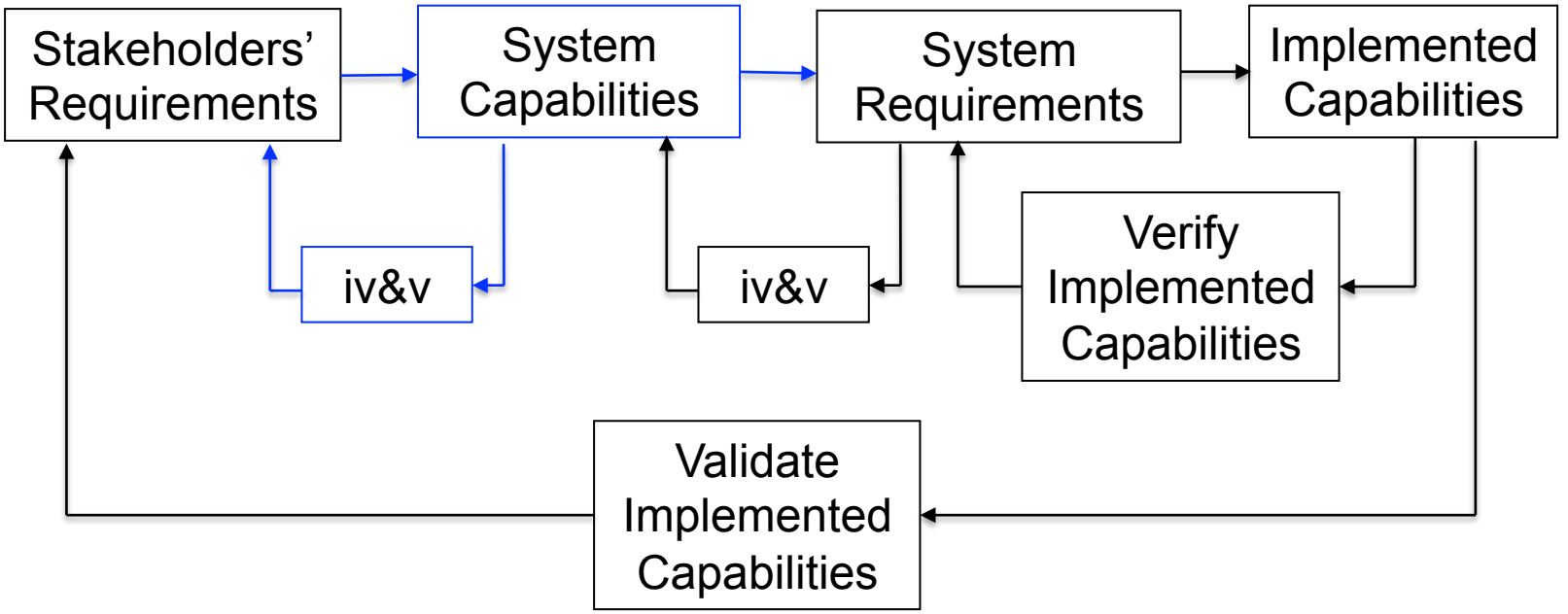
The capabilities approach is enabled by domain expertise and technical competence

Traditional system development



Architecture and design definition are omitted for simplicity of presentation

Capabilities-based system development



System capabilities bridge the gap from stakeholders' requirements to system requirements

System capabilities

- A system capability is the ability of a system to execute a ~~specified~~ course of action or to exhibit a ~~specified~~ state of being
  - Specified and otherwise

    otherwise:

    e.g., emergent or non-deterministic behavior

  Emergent behavior: unanticipated behavior when a system element or subsystem is modified, a new element or subsystem is added, or two systems are combined

  Non-deterministic behavior: different behaviors occur for the same system state and stimulus

A small familiar example

- A stakeholders' requirement:

  The customers of our financial institution should be able conduct financial transactions at convenient times and in convenient locations; safely and securely

- Some ATM system capabilities:

  C1: ability to provide secure customer authentication

  C2: ability to provide secure session termination

  C3: ability to check account balances

  C4: ability to withdraw funds

  C5: ability to deposit funds into accounts

  C6: ability to transfer funds between accounts

  C7: ability to handle exception conditions that may arise

Physical security of customers will be addressed in the operational requirements

The capabilities-based approach

Capabilities	**ATM Hardware, Software, and Interfaces**						
	Card Reader	Display Screen	Keypad	Cash Safe	Cash Dispenser	Funds Depository	Printer
Customer authentication	X	X	X				
Session termination		X	X				X
Balance query		X	X				
Cash withdrawal		X	X	X	X		
Funds deposit		X	X			X	
Funds transfer		X	X				

Note: designing and implementing the card reader, display screen, and keypad includes designing and implementing the user interface

NOTE

- System requirements for hardware, software, and interfaces are defined together for each system capability
- System capabilities are then realized by concurrently developing hardware, software, and interfaces for each capability
- The capabilities-based approach mitigates two major problems of system development
  1. The hardware-first, software-first approaches to system development
  2. And, the associated hardware-software interface issues

Capabilities implementation priorities

- Priority 1: Secure Customer Authentication capability
- Priority 2: Secure Session Termination capability
- Priority 3: Balance Query capability
- Priority 4: Cash Withdrawal capability
- Priority 5: Funds Deposit capability
- Priority 6: Funds Transfer capability

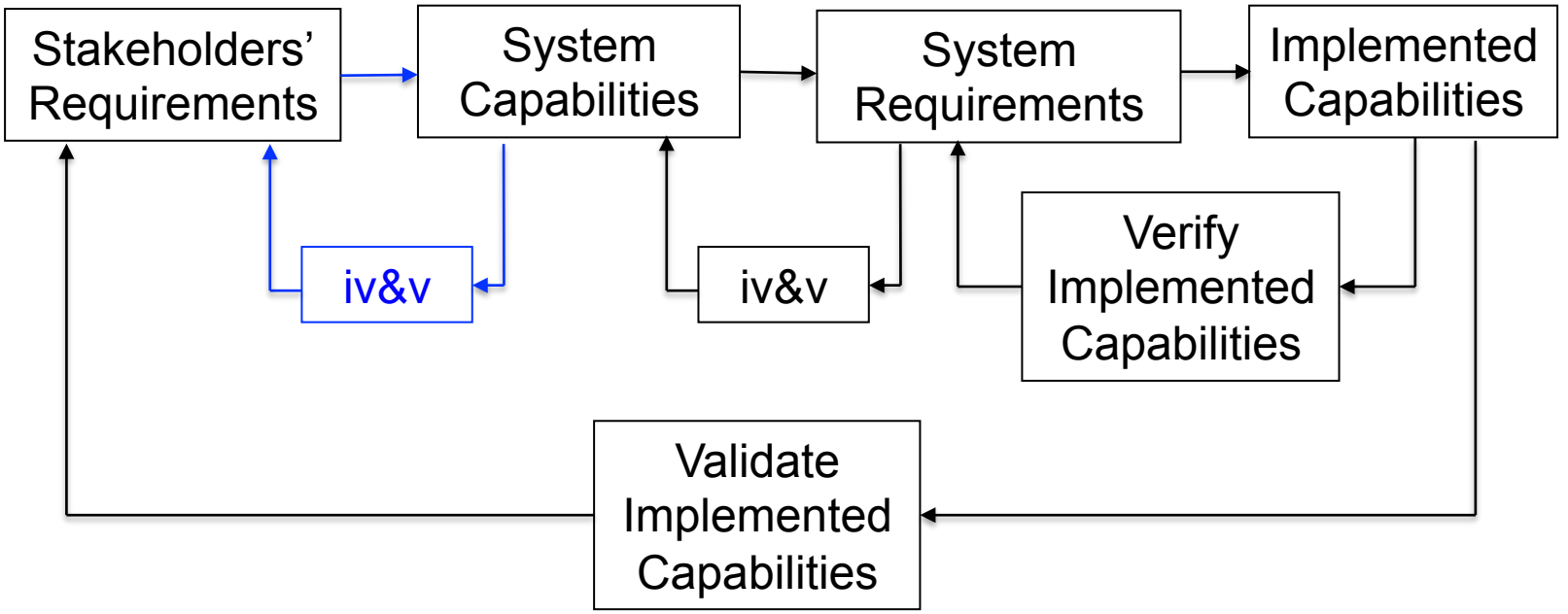
Note 1: capabilities developed first are tested most

Note 2: capabilities developed first are used by lower priority capabilities

Some capabilities prioritization criteria

Criterion	Rationale
Easiest-to-realize capabilities first	To gain familiarity with the capabilities-based approach and build team cohesion
Capabilities need by other capabilities realized first	To provide a platform for realizing capabilities that are added later
Safety and security capabilities realized first	Capabilities developed first are tested and demonstrated most frequently in combination with capabilities that are added later
Deliverable subset capabilities realized first	To be evaluated by potential users and/or placed into operational use
Highest risk capabilities realized first	To determine that known uncertainties and complexities can be accommodated and to expose previously unknown risks and complexities
External connections realized first (real and simulated)	To establish connections to the real or a simulated operational environment
Legacy-element capabilities realized	To determine that legacy elements can be satisfactorily

Capabilities-based system development



Architecture and design definition are omitted for simplicity of presentation

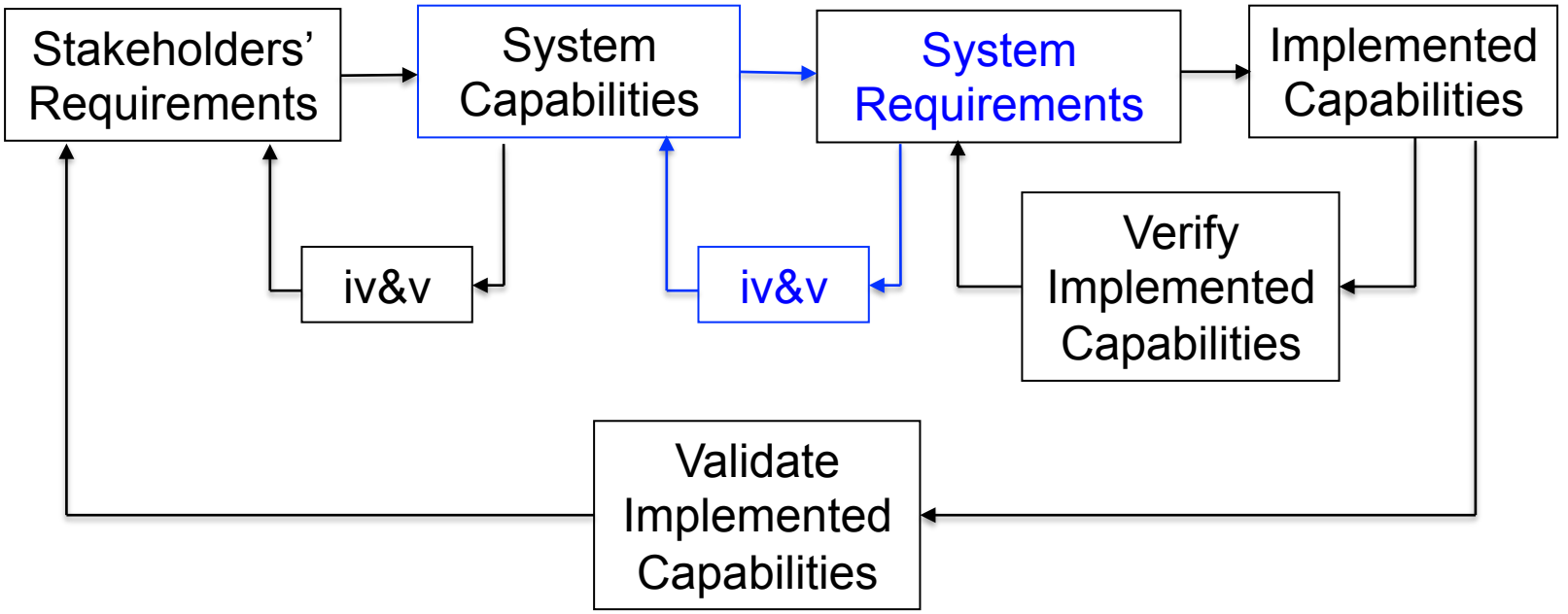
Verifying and validating system capabilities

Verifying and validating system capabilities involves

- Informally verifying *the degree to which*
  - the system capabilities cover the stakeholders' requirements
- Informally validating *the degree to which*
- the system capabilities, as documented, will provide the information needed by those who will use the capabilities to develop their work products (or not)
  - Requirements engineers, designers, implementers, testers, and others

V&V techniques for system capabilities are those listed above and in the V&V annex

Capabilities-based system development



- These processes may be recursively decomposed iterated, interleaved, and accomplished incrementally
- Architecture and design definition are omitted for simplicity of presentation

Capabilities-based requirements definition

- System requirements derived from capabilities are:
  - Categorized;
  - Prioritized; and
  - Informally verified and validated wrt the system capabilities

Agenda: Requirements Engineering Part 2

- Brief review of Part 1
- Traditional approaches to defining system requirements
  - Role of systems engineers in traditional requirements engineering
- Defining, categorization, prioritization, and prioritizing system requirements
- Verifying and validating system requirements
- Capabilities-based requirements engineering
  - Role of systems engineers in capabilities-based requirements engineering

The role of systems engineers in *capabilities-based* RE

- Identify system stakeholders
  - Hands-on users and others who affect or will be affected by the proposed system
- Elicit stakeholders' needs, wants, and desires
  - Don't ignore subjective statements (design goals)
- Analyze stakeholders' need, wants, and desires
- Determine technical feasibility
- Document the stakeholders' requirements definition
  - Develop a SOW or MOU and a Concept of Operations
- Manage the stakeholders' requirements

Role of systems engineers in *capabilities-based* RE (2)

- *Derive, categorize, and prioritize the system capabilities needed to satisfy the stakeholders' requirements*
- *Informally verify that the system capabilities adequately cover the stakeholders' requirements*
- *Informally validate that the system capabilities, as documented, provide adequate documentation for those who will use the capabilities to develop their work products*

Role of systems engineers in *capabilities-based* RE (3)

- *Define and categorize system requirements needed to provide the system capabilities*
  - *Primary requirements, derived requirements, design constraints, and design goals*
- *Establish bi-directional requirements traceability to and from the stakeholders' requirement, the system capabilities, and the system requirements*
  - *And to and from related test plans, test scenarios, and test cases*
    - ✓*Perhaps developed by testing specialist engineers*
- Develop plans for the technical work to be accomplished by the disciplinary and specialist engineers

Two concerns about the capabilities approach

Experience with capabilities-based requirements engineering has revealed the following concerns:

1. The “overhead” of adding another process to requirements engineering
2. Defining contradictory and inconsistent capabilities-based system requirements

The “overhead” of capabilities-based RE

1. Long experience has shown that more attention paid to upstream processes yields dividends by savings to:
   - Schedule, resources, and budget
   - More is returned than is invested
   - Confusion, thrashing, and rework are reduced
2. Focusing on system capabilities results in more effective conversations with stakeholders than discussing system requirements
3. Capabilities can be recursively decomposed to control complexity of the system requirements
4. Defining capability-based system requirements improves the “5 Cs” of requirements:
   - correct, complete, consist, concise, and clear

The “overhead” of capabilities-based RE (2)

- System requirements for hardware, software, and interfaces are defined together for each system capability
- System capabilities are then realized by concurrently developing hardware, software, and interfaces for each capability
- The capabilities-based approach mitigates two major problems of system development
  1. The hardware-first, software-first approaches to system development
  2. And, the associated hardware-software interface issues

Contradictory and inconsistent system requirements

- System requirements are reused for different system capabilities and new requirements are generated as needed
  - This may result in contradictory and inconsistent requirements
  - This commonly occurs for both the traditional and capabilities approaches to defining system requirements
- For example:

  Capability A generates a system requirement for degraded performance on exception condition X

  Capability B generates a system requirement for a system reboot on exception condition X

  Capability C generates a system requirement for a rollover to hot backup computer H in subsystem S on exception condition X

Contradictory and inconsistent system requirements (2)

- Association matrices that shows couplings and dependencies among capabilities can illustrate the impacts of each capability on other capabilities
  - With bi-directional traceability of capabilities to the associated system requirements

Capabilities can be recursively decomposed to control the complexity of system requirements

Agenda: Requirements Engineering Part 2

- Brief review of Part 1
- Traditional approaches to defining system requirements
  - Role of systems engineers in traditional requirements engineering
- Defining, categorization, prioritization, and prioritizing system requirements
- Verifying and validating system requirements
- Capabilities-based requirements engineering
  - Role of systems engineers in capabilities-based requirements engineering

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

Questions and comments?

V&V Annex

- ISO/IEC/IEEE Standard 15288 includes V&V in Clauses 6.4.9 and 6.4.11
- 6.4.9 Verification process
- 6.4.10 Transition process
- 6.4.11 Validation process

These are the processes of “formal verification”

6.4.9 Verification

6.4.9.1 Purpose

The purpose of the Verification process is to provide objective evidence that a system or system element fulfills its specified requirements and characteristics.

This process . . . installs a verified system, together with relevant enabling systems, e.g., planning system, support system, operator training system, user training system, as defined in agreements.

NOTE Verification methods or techniques include: inspection (including peer review), analysis (including modeling, simulation, and analogy/similarity), demonstration, or testing.

6.4.9 Verification (2)

- **6.4.10.2 Outcomes**
- a) . . .d)
- e) Operators, users and other stakeholders necessary to the system utilization and support are trained.
- f) . . . h)

6.4.10 Transition process

6.4.10.1 Purpose

The purpose of the Transition process is to establish a capability for a system to provide services specified by stakeholder requirements in the operational environment.

This process . . . installs a verified system, together with relevant enabling systems, e.g., planning system, support system, operator training system, user training system, as defined in agreements.

. . .

6.4.11 Validation process

6.4.11.1 Purpose

The purpose of the Validation process is to provide objective evidence that the system, when in use, fulfills its business or mission objectives and stakeholder requirements, achieving its intended use in its intended operational environment.

- Simulations, demonstrations, and testing are the primary mechanisms of informal validation
- Stating objective criteria and providing objective evidence is an ongoing challenge for formal validation

6.4.11.3 Activities and tasks

a) **Prepare for validation.** This activity consists of the following tasks:

- 1) Identify the validation scope and corresponding validation actions.
- NOTE 1 . . . The scope [of validation] depends on what is appropriate for the systems life cycle stage; it can be the system-of-interest or any system element or engineering artifact, such as a concept description or document, an operational scenario, a model, a mock-up, or prototype.
- The scope [of validation] also includes evaluating that the product or service is predictable in its intended environment and does not enable any unintended uses that can negatively impact the intended use of the system.

V&V Annex

- All major work products of system development should be informally verified and validated
  - V&V should not be limited to the deliverable system
- In general, to the purpose of verification is *to determine the degree to which** a work product satisfies the conditions and constraints placed on it by other work products, policies, procedures, and regulations.
- In general, the purpose of validation is to *determine the degree to which** a work product is suitable for use by the intended users of the work product in the intended ways in the intended context.

* V&V outcomes are not binary results

Formal versus informal V&V

- Formal V&V of deliverable systems is typically accomplished by specialists who are in a separate department
  - or in another organization in the case of IV&V
- Informal V&V of intermediate work products is accomplished by SEs, disciplinary engineers, specialty engineers, and domain experts

Another note of V&V

- V&V specialists from another department may be reluctant and confused if asked to V&V work products other than a deliverable system, subsystem, or system element
  - Including intermediate increments of an evolving system

Systems engineers may want to use different terms for informal V&V

e.g., “evaluating the adequacy of work products”

or, “applying checklists and reviews”

The role of systems engineers in traditional system requirements definition

- Verify: determine *the degree to which* the system requirements cover the stakeholders' requirements and other conditions and constraints
- Validate: determine *the degree to which* the system requirements, as documented, will provide the information needed by those who will use the system requirements for their intended roles
- Ensure that the system requirements are reworked to satisfy the V&V criteria

Determining *the degree to which*: the outcomes of V&V are not binary

V&V techniques for system requirements

- Bi-directional traceability analysis
  - Are all stakeholders' requirements adequately covered by the system requirements?
  - Are there any extraneous system requirements that don't trace to any stakeholders' requirements, directly or indirectly?
- Manually "executing" the operational scenarios in the ConOps
  - Which system requirements support which scenarios?
- Simulation and prototyping
- Reviews and Inspections
- Technical analysis

These techniques are useful for informal V&V of other work products

The role of systems engineers for informal V&V

- SEs facilitate informal V&V of the system requirements
  - Facilitate: to help bring about
  - Systems engineers may accomplish informal V&V
  - Or, engage disciplinary engineers with V&V expertise
- Systems engineers may provide domain expertise
  - Additional domain expertise may be needed