How to Be Successful in the Absence of Requirements

Dr. Ron Carson, ESEP, INCOSE Fellow

University of Washington, Seattle Pacific University

"Ron on Requirements" YouTube Channel:

https://www.youtube.com/channel/UC8oARCrOZ7_wv5vgEEeTrnw

LinkedIn – Misperceptions of Systems Engineering:
https://www.linkedin.com/pulse/misperceptions-systems-engineering-1-ron-carson-phd-esep/

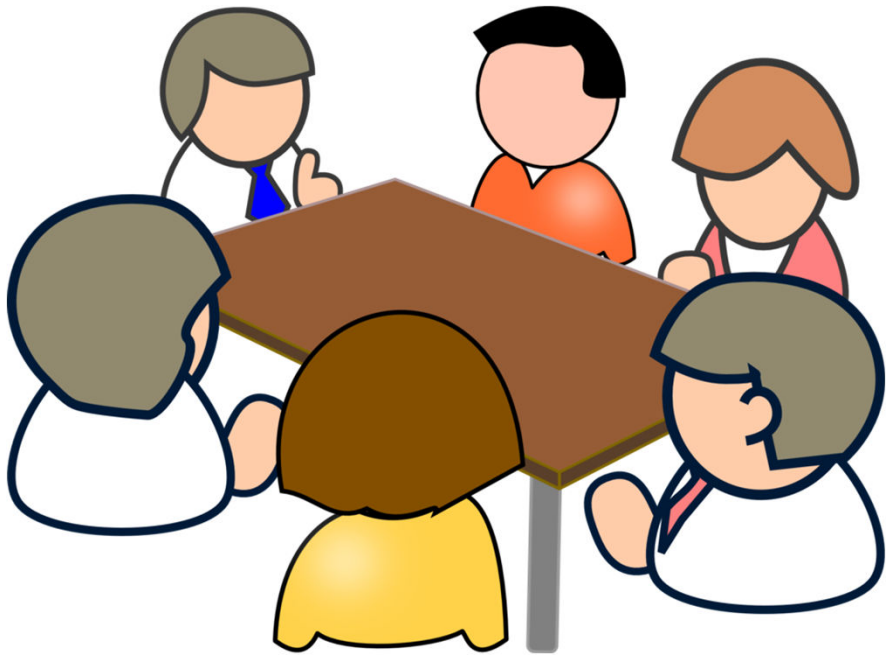
ronald.s.carson@gmail.com

Outline

- Thesis: Optimizing product delivery depends on avoiding and eliminating rework. This requires clarity regarding what is to be done.
- The problem
- Impacts and options
- Methods for addressing ambiguity
- Defining good requirements
- Finding all the requirements
- Validating requirements
- Summary

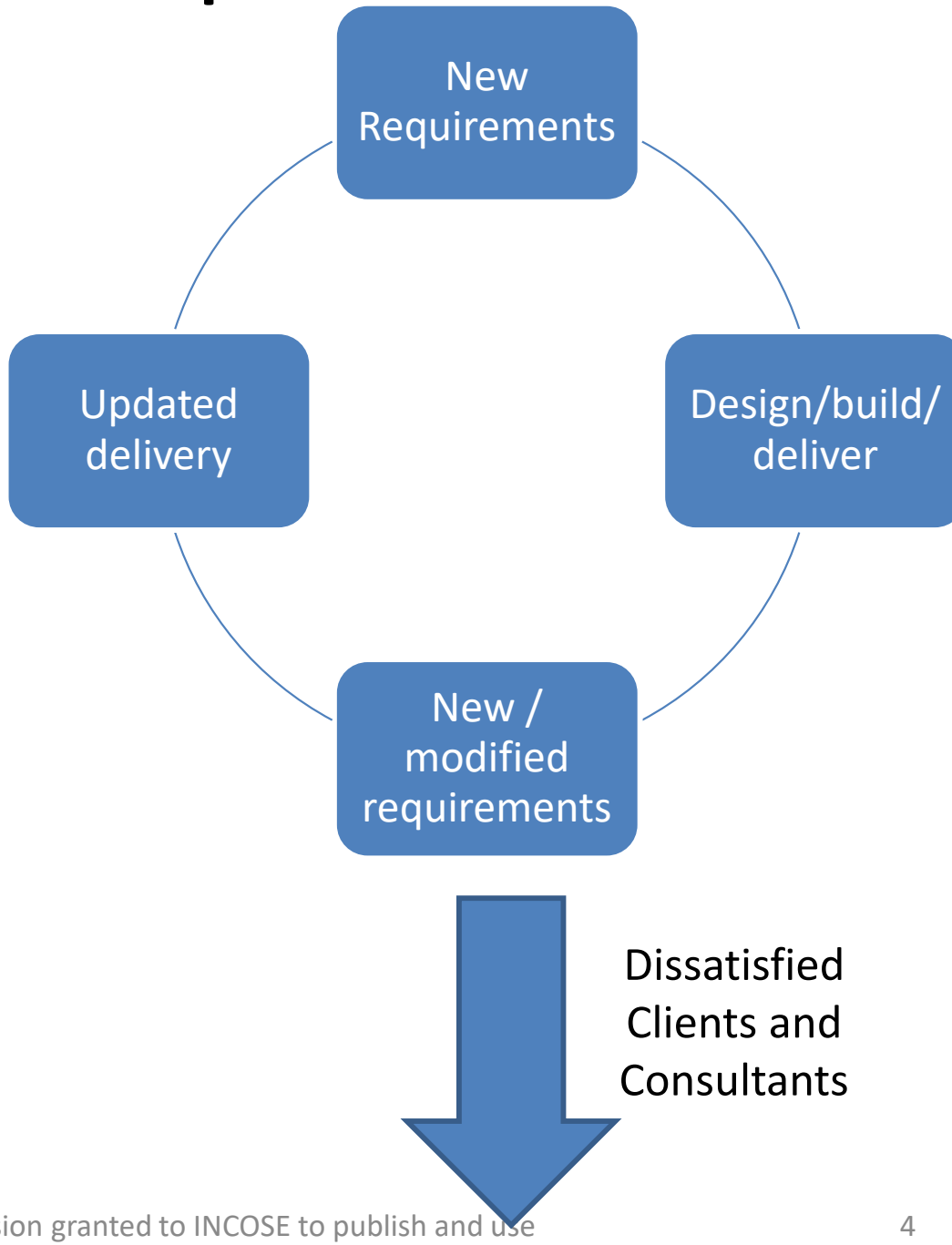
Problems

- Clients think they know what they want
  - This will change over time
- Clients may be unable to describe the problem they want to solve
  - At best they may describe features of a solution
- Clients may think narrowly about a single class of stakeholders (themselves as users)
  - Failure to consider testing, maintenance, other users…

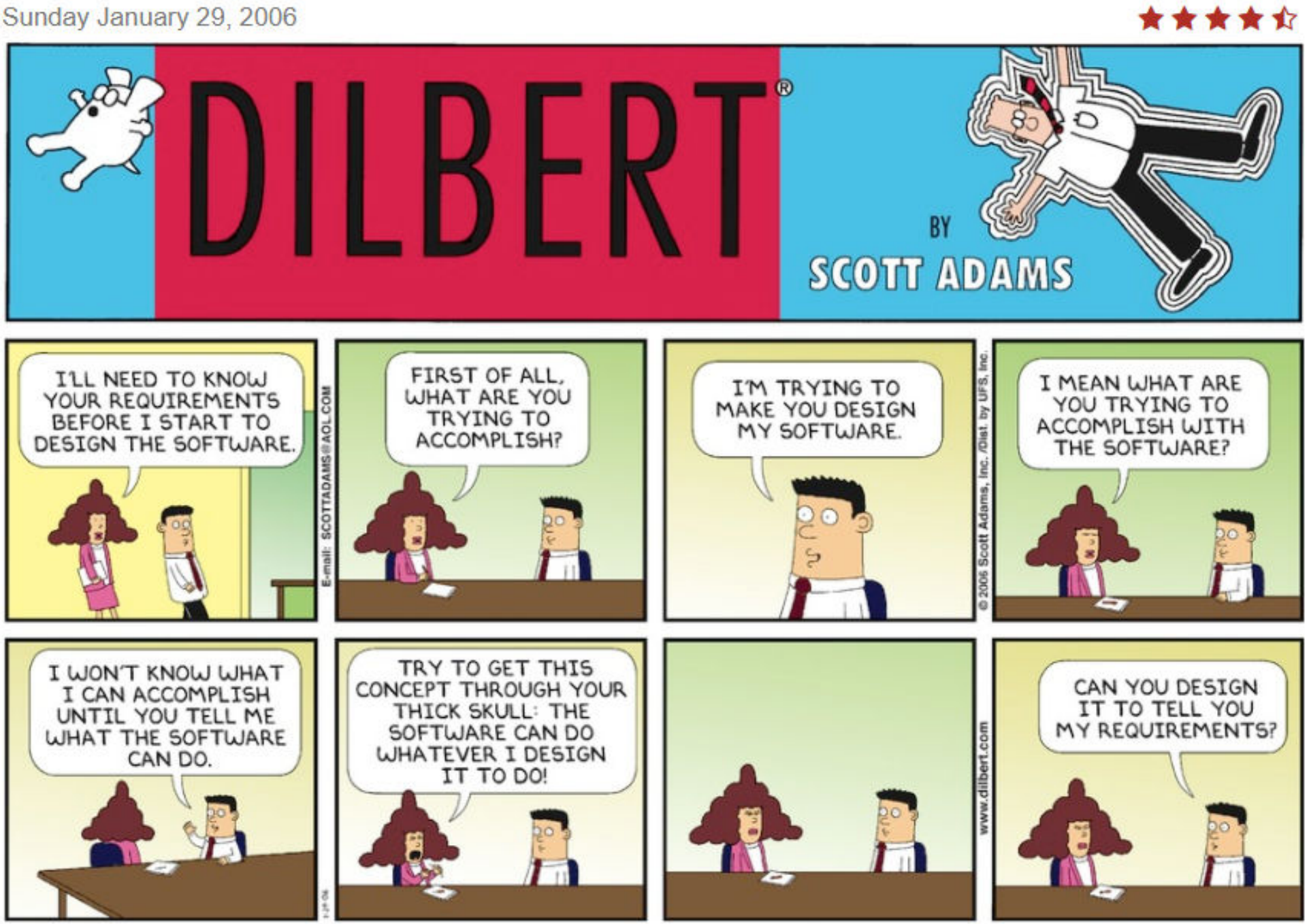


What to Do? Options

- "These are the requirements the client gave me; these are the only things I can work to."
- "We can't do anything until we're sure we have all the customer requirements."
- "The client doesn't have time to give us requirements, we'll just have to guess."
- "We need the client working with us each day to try out our deliveries and give us feedback." ["agile"]
- ➢ "We recognize potential incompleteness in requirements; we will coach the client into disclosing what they know/think/imagine."

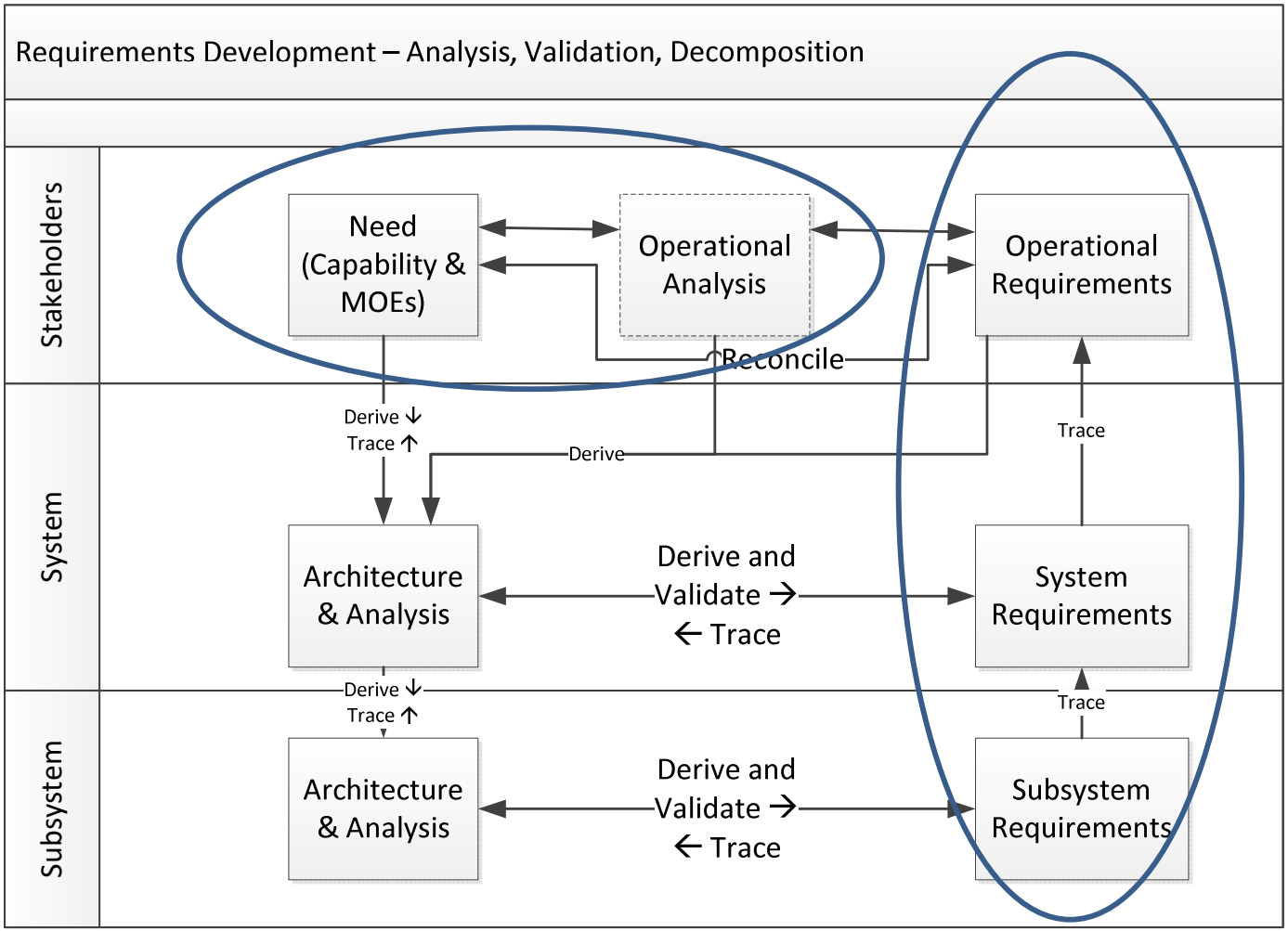


This is not a new problem



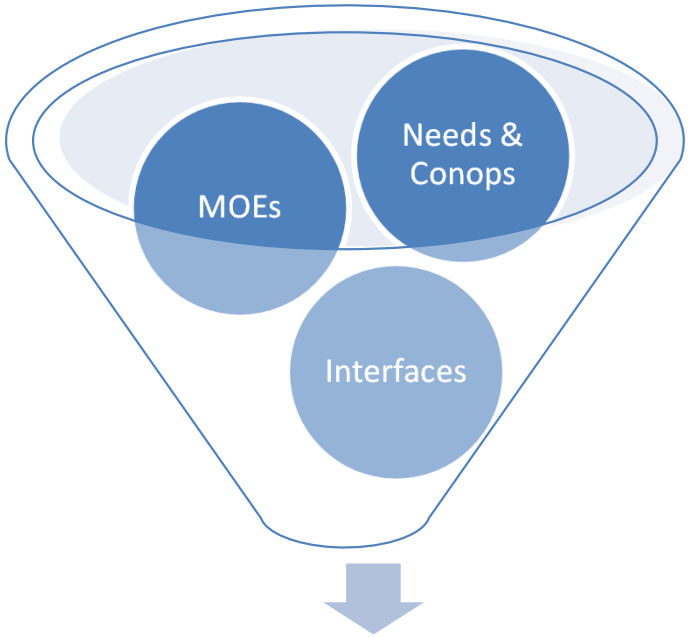
How do we achieve Valid Requirements?

- Analysis of user needs and constraints
- Operational analysis to ensure requirements are consistent with needs, MOEs, and Conops
- Recursive architecture analysis and requirements development
  - Derive and write requirements from analysis
  - Trace requirements to analysis and parent requirements via analysis

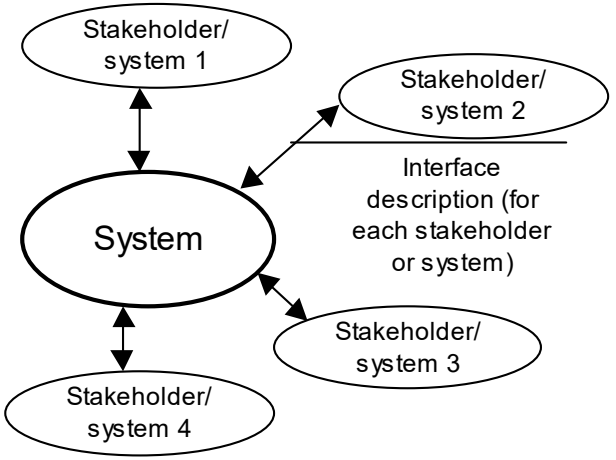


“Begin with the End in Mind” (Covey) by asking Questions

- Questions for clients:
  - **Problem identification**: What problem are we trying to solve?
  - **Measurable results/outputs:** If we solved the problem, how would the universe be different? What would we see/observe/measure (**measures of effectiveness)**?
  - **Concepts of operations**: How will you (and others) use the solution? Maintain? Test? Can you describe a “day in the life”?
  - **External interfaces**: With what other entities/persons must the solution interact?
  - **Constraints**: What characteristics of a solution are absolutely critical for this project?



Requirements



Example Dialog

Client: perceived needs

- I need a new car
- I need to show clients available real estate
- Up to four; it needs to communicate “success”

OR

- I need a new car to get to work.

Consultant: Getting to real need

- How do you want to use it?
- How many clients do you need to hold? What other characteristics are most important to you?

- How far, how fast, how often?

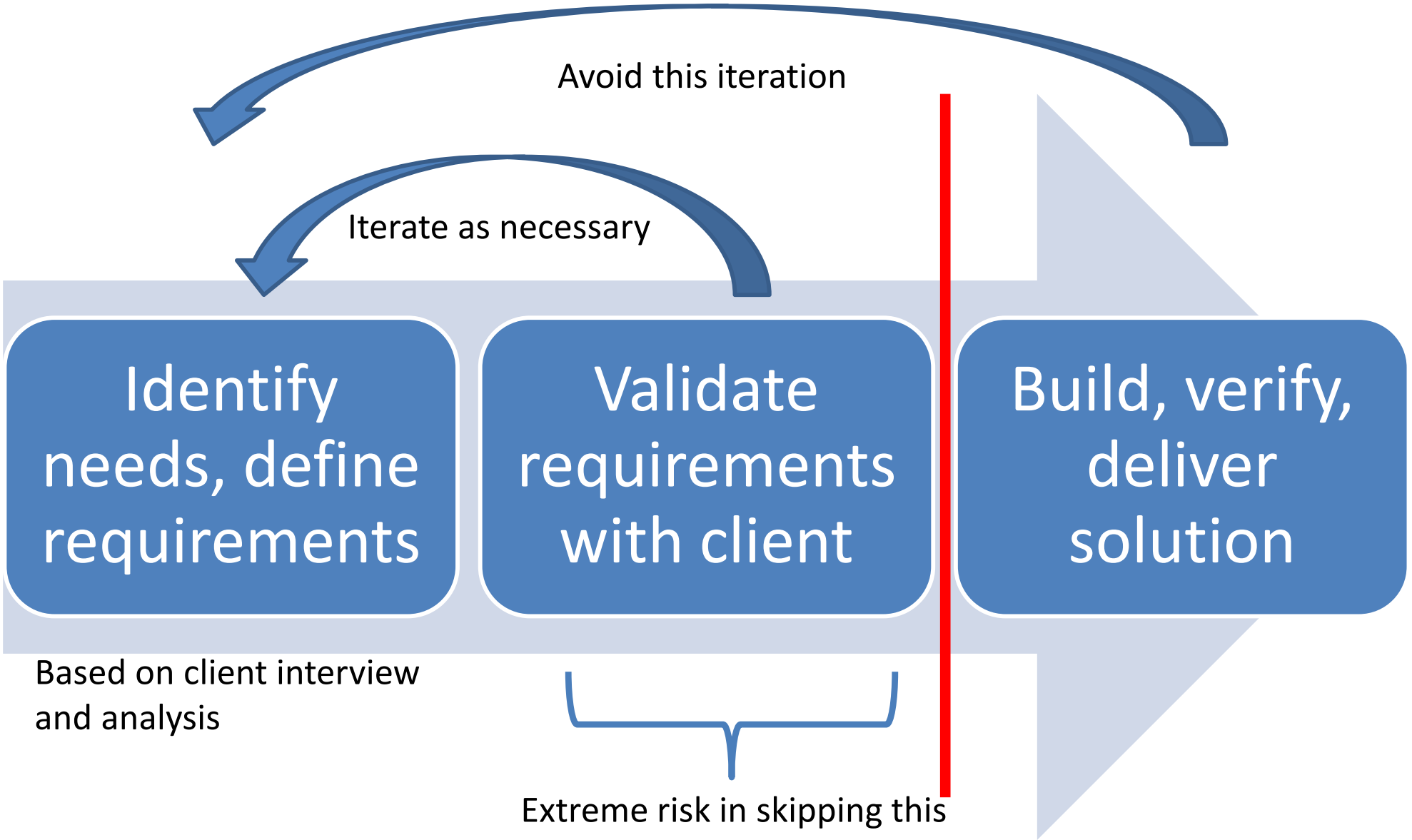
In the Absence of Client Input…

- Consultant must *infer* the problem and solution characteristics:
  - Who are the users and other stakeholders?
  - What do they care about (problem, measures)?
  - How will they use the new system?
  - What are the critical characteristics (acceptance criteria)?
  - What are the constraints on the solution?
- Then the consultant *must* develop and *validate* the requirements with the client before proceeding

Client Meeting Checklist

- **Problem**
  - What is it you don't like about the current situation?
  - What is the problem you think we are trying to solve?
- **Concept of operations**
  - How will you (and other users) use / maintain / test the solution?
  - Can you tell a story about a "day in the life"?
- **Measurable results**
  - If we solved the problem, how will the universe be different?
  - What would we be able to see/observe and measure?
- **External interfaces**
  - With what other entities/persons must the solution interact?
  - Are there existing interfaces, applications or other systems with which the solution must be compatible?
  - Are there any important environments?
- **Design constraints**
  - What characteristics of a solution are absolutely critical for this project?
  - Software
    - What are the host operating system and version and hardware (minimum or existing CPU speed and memory, networks)?
    - Is there a required programming (or modeling) language?
- **General**
  - Why is this constraint or characteristic *absolutely* required? If the solution does not have that feature (or characteristic), would it still be acceptable?
  - Are there any allowable options or variations?

A More Linear, 2-phase Process



What Requirements Require

- Identify which characteristic is being addressed
  - Function/behavior, performance, condition, interface, solution constraint (design)
- Differentiate *mandatory* from *preferred*
  - Question to client: "if the solution does not have that feature (or characteristic), would it still be acceptable?"
  - "Mandatory" is a *requirement*, a basis for accepting/ rejecting a solution. There should be *few* requirements.
  - All else are design descriptions or preferences.
  - "Shall" (or other identified key word) connotes *mandatory*
  - Organize characteristics as either "requirements" or "preferences"
    - Viable solutions MUST satisfy *requirements*
- The Basic Structure*:
  - The *who* shall *what, how well, under what conditions.*

Functional/Performance - The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.

Design - The AGENT shall exhibit DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION].

Environmental - The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].

Suitability - The AGENT shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION].

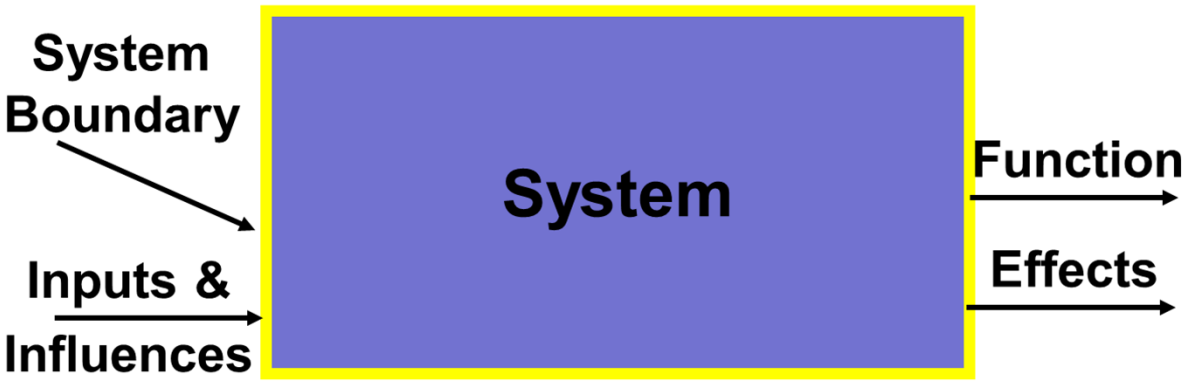
*Ref., Ronald Carson, "Implementing Structured Requirements to Improve Requirements Quality", Proceedings of the International Council on Systems Engineering (INCOSE) 2015.

Examples

- The System shall deliver the user at the selected destination within 10 miles of a starting location with ≥ 99.0% reliability within 1 hour of beginning travel while exposed to any natural environment, from 0500 to 2300 local time.
- The System shall be built in C#, version abc or newer.
- The System shall satisfy all functional requirements during and after exposure to a denial of service attack defined in *abcd*.
- The System shall have availability > 99% during any 1-year period while powered and exposed to environments defined in *efgh*.

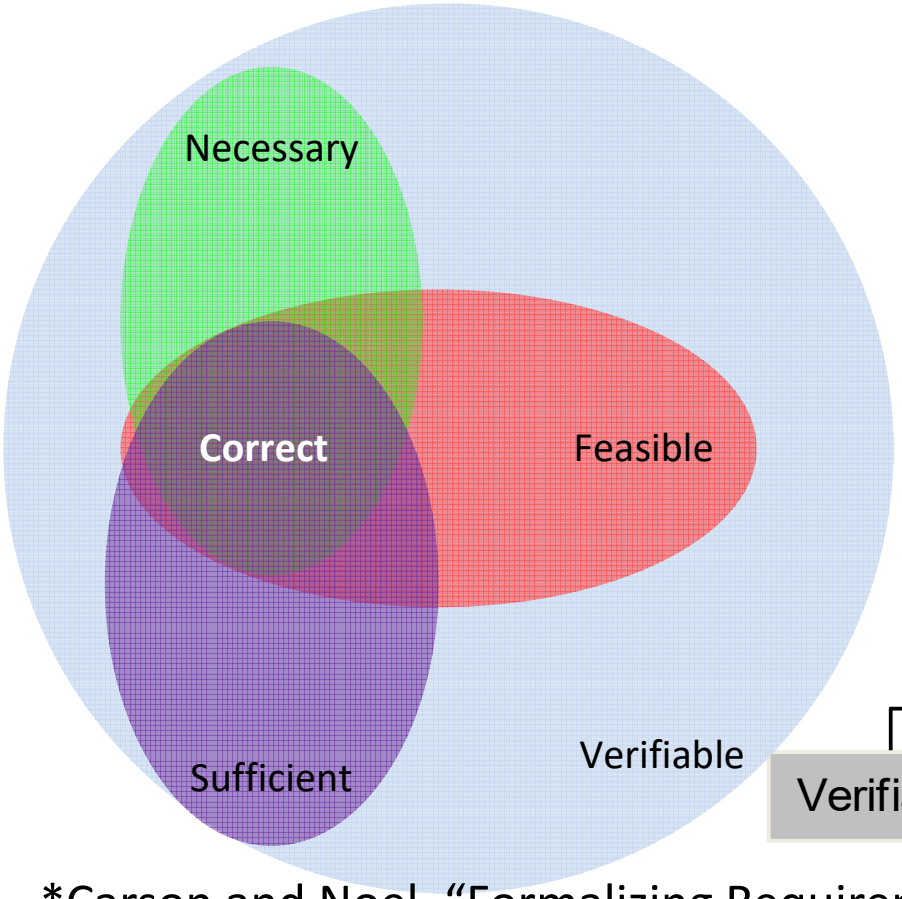
Finding All the Requirements

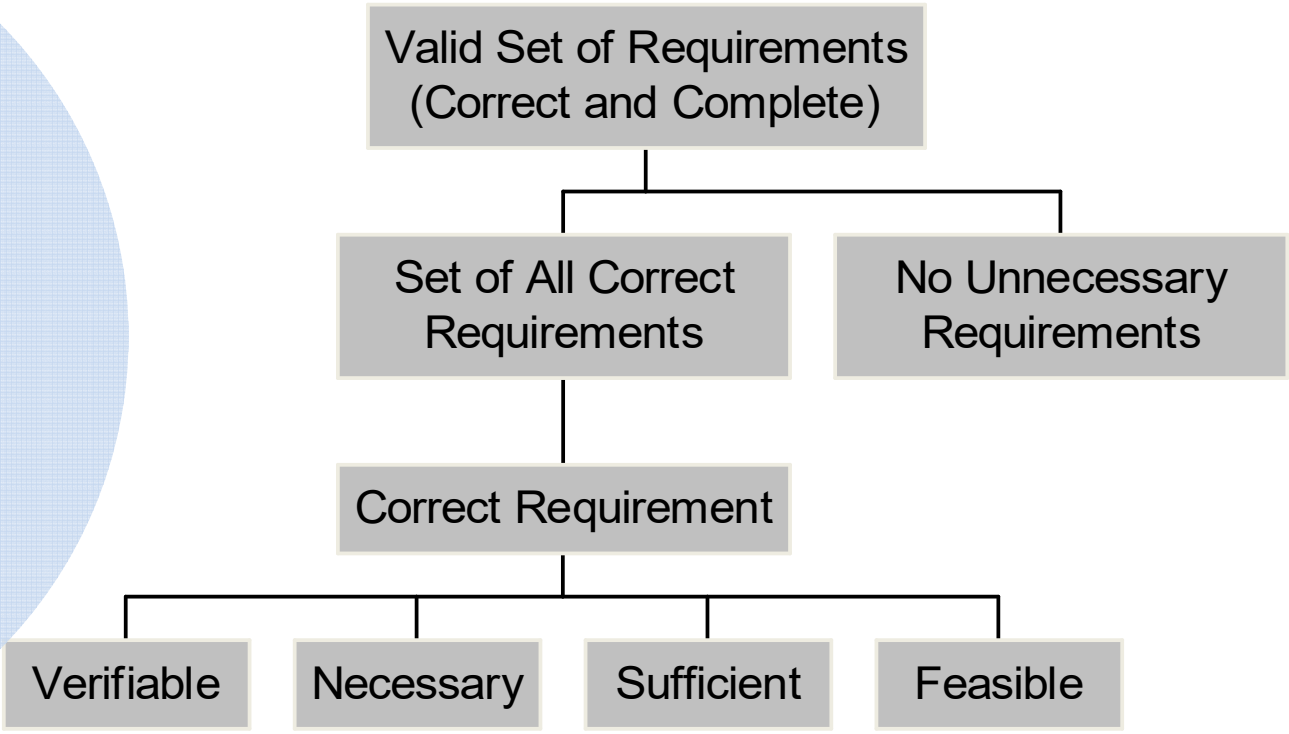
- Ad hoc: Identify "critical to quality" characteristics based on system scope, "user stories" or Conops, measures of success
- Formal: Define required behavior for all interface conditions (Carson 1998, 2001, 2004)



Validating the Requirements

- "Validation" is about ensuring a correct and complete (none missing) set of requirements.





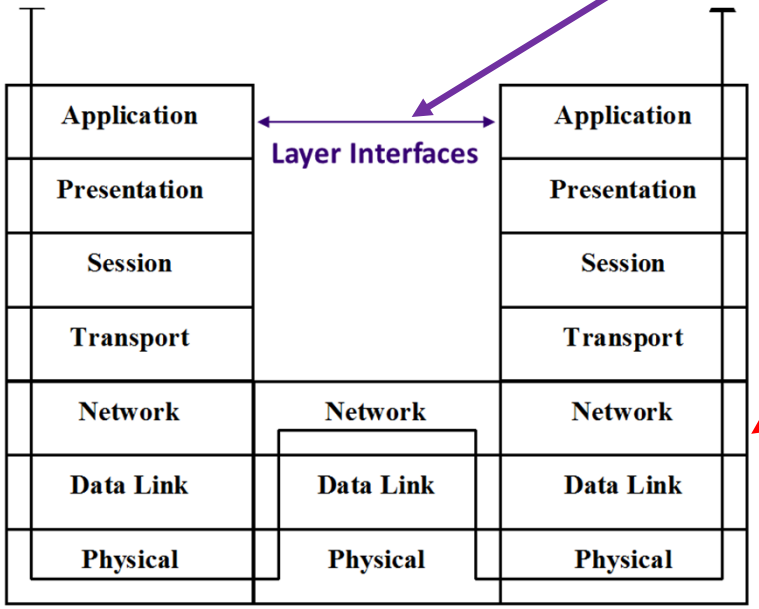
*Carson and Noel, "Formalizing Requirements Verification and Validation", INCOSE 2018

Checklist for Validating Requirements

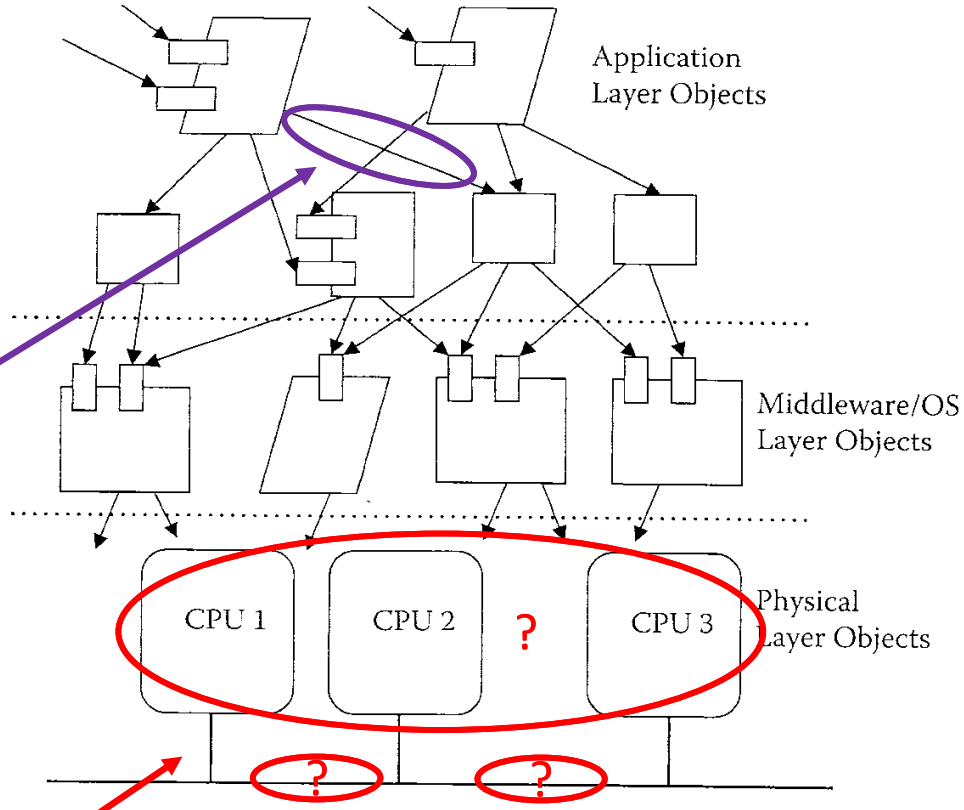
- ❑ Is it a mandatory characteristic? (Yes)
- ❑ Verifiable: is it written such that it could be objectively confirmed or falsified? (Yes)
- ❑ Necessary: Could we satisfy all user needs even if we deleted it? (No)
- ❑ Sufficient: If we satisfy this requirement will we satisfy the measurable stakeholder need? (Yes) Are all stakeholder needs being addressed by the set of requirements? (Yes)
- ❑ Feasible: Can this requirement, in combination with all others, be implemented with acceptable risk given the budget, schedule, and technology? (Yes)

Don’t Ignore the Hardware

- Software-only view misses hardware requirements in analysis of child requirements (e.g., physical layer)
- Resulting SW requirements cannot be determined to be complete (“sufficient”) without concurrent examination of hardware requirements



OSI layer model (from ISO/IEC 7498-1)



Software layered architecture (Maier & Rechtin, “The Art of System Architecting, 3rd Edition, modified Figure 6.4)

Summary

- Clients may not really know what they want until they see something close
- The Consultant's challenge is to elicit and define the requirements based on perceived, inferred, and actual problem statements and solution characteristics
- Requirements must be validated to ensure they are verifiable, necessary, sufficient, and feasible
- Having validated (correct and complete) requirements helps minimize rework and improves client satisfaction

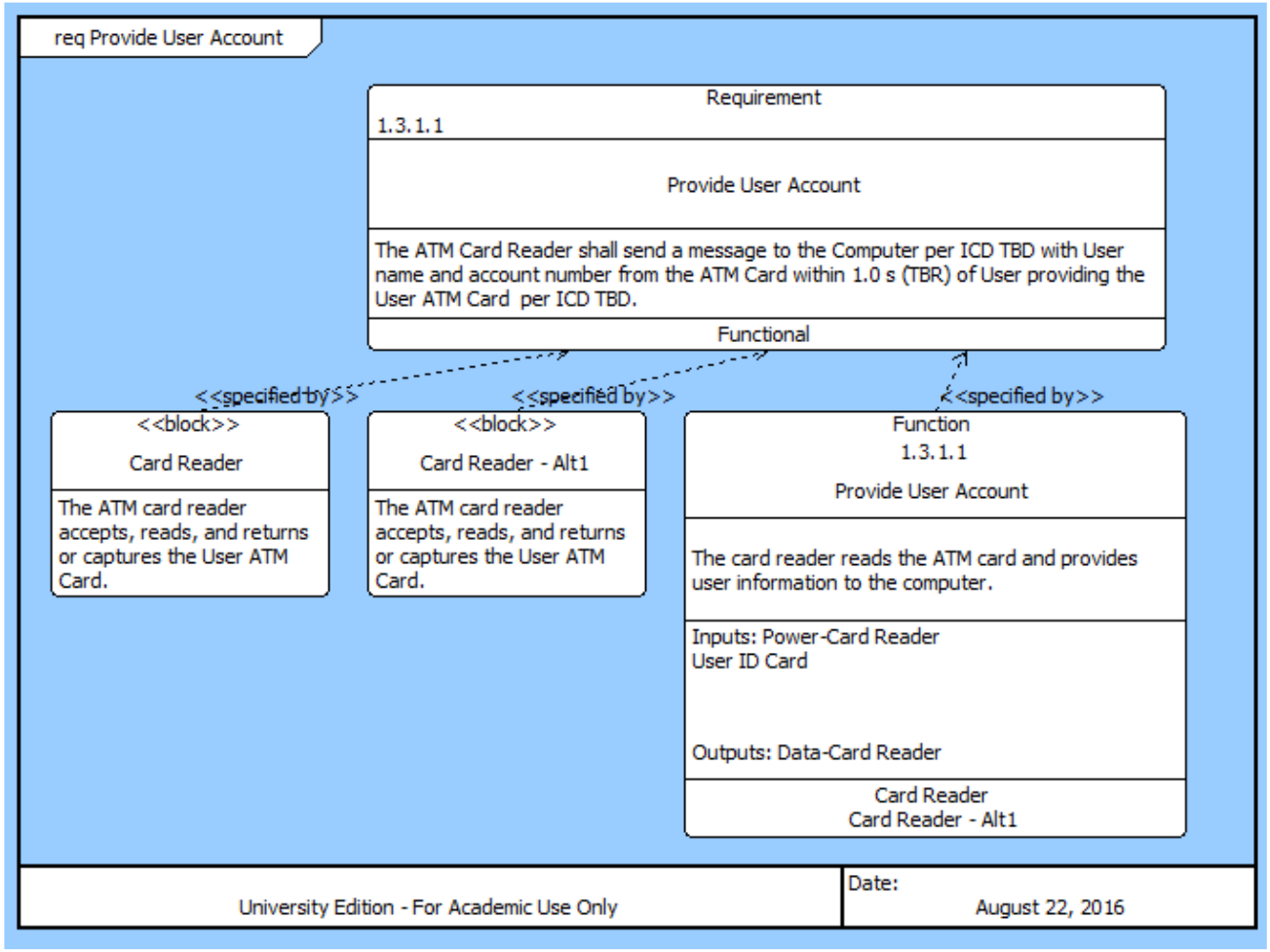
Related Topics and References

- “User experience” – focus on user interface and interaction
- “Business [process] analysis” – focus on business processes
- “System architecting” – figuring out how to satisfy what clients need
- International Council on Systems Engineering (INCOSE, incose.org)

- Some references:
  - Hooks and Farry, “Customer-Centered Products”
  - Robertson and Robertson, “Master the Requirements Process”
  - Maier and Rechtin, “The Art of Systems Architecting”
  - Carson and Noel, “Formalizing Requirements Verification and Validation”, INCOSE 2018
  - Carson, “Implementing Structured Requirements to Improve Requirements Quality”, INCOSE 2015

Information for Structured Requirements* in CORE

- Model-based structured requirements help ensure valid requirements by explicitly exposing the bases of the requirements from the validated analysis (based on the simulation)
  - Function, Input/ Output Interfaces, MOEs, Performance information



*Carson, "Implementing Structured Requirements to Improve Requirements Quality", *Proceedings of INCOSE 2015*.