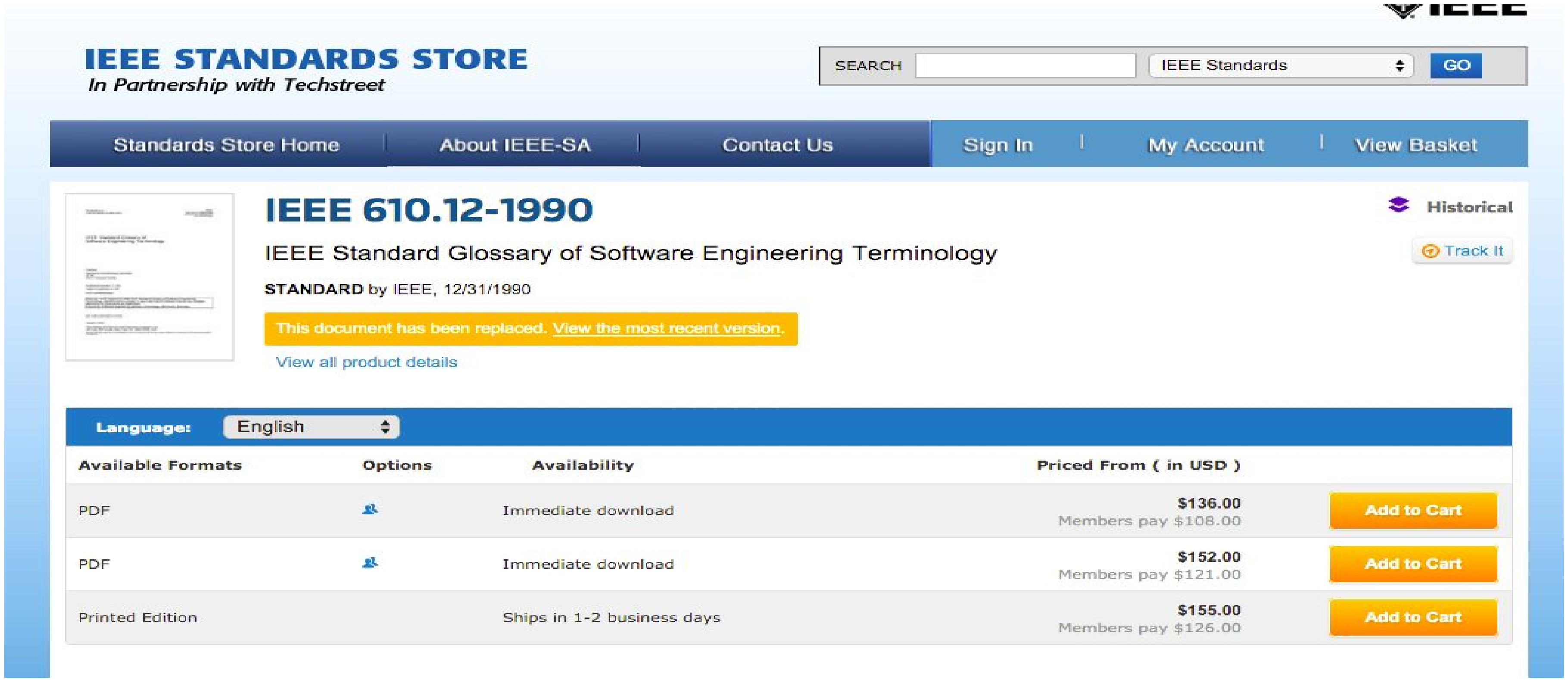
Software Engineering
Lecture 4: Requirements

Gregory S. DeLozier, Ph.D.

gdelozie@kent.edu

System Requirements

Where do definitions come from?



IEEE STANDARDS STORE
In Partnership with Techstreet

IEEE 610.12-1990

IEEE Standard Glossary of Software Engineering Terminology

STANDARD by IEEE, 12/31/1990

This document has been replaced. View the most recent version.

View all product details

Language: English

Available Formats	Options	Availability	Priced From (in USD)	
PDF		Immediate download	$136.00 Members pay $108.00	Add to Cart
PDF		Immediate download	$152.00 Members pay $121.00	Add to Cart
Printed Edition		Ships in 1-2 business days	$155.00 Members pay $126.00	Add to Cart

Software Requirement (via Wikipedia)

The IEEE Standard Glossary of Software Engineering Technology defines a software requirement as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

Requirements Engineering

- Process that creates, documents, and maintain requirements
- Waterfall: Do this process once, for the whole problem.
- Agile: Do this process repeatedly, over increments of the problem.
- Major activities:
  - Elicitation - what are we doing?
  - Specification - what, *exactly*, are we doing?
  - Validation - are we sure we are specifying the right thing?
  - Verification - did we do what we specified we would do?
- Each of these has methods and tools

Requirements Elicitation

- This process asks: what should we do?
- Problems:
    - Scope - too little discussion, or too much detail
    - Mistakes - wrong or incomplete information
    - Volatility - drift in correct information
- Systems:
    - Interviews
    - Committees
    - Prototypes
- Reading:
    - "Issues in Requirements Elicitation", Christel and Kang, in the repository under .../papers

Requirements Elicitation - Interviews

- Talking to people who have an interest in the system ("stakeholders")
  - People who use the system
  - People who pay for the system
  - People who approve the system
  - People who accept output or provide input
  - People who will provide an environment or context
- Free discussion ("brainstorming", focus groups)
- Structured interviews - e.g. Miller has > 2000 standard questions...
- Document and artifact reviews
- Shadowing or participation

Requirements Elicitation - Committee

- Working with developers and stakeholders simultaneously
- Early versions of this: Joint Application Development
  - Fairly constant feedback
  - Quick observation of problems
  - Relatively painless correction
- Later version in agile: Customer on the Team
  - Continuous participation
  - Creative input (vs just offering feedback)
- Joint ownership of requirements

Requirements Elicitation - Prototyping

- Creation of an exploration of a possible solution
- Low-fidelity, broad
  - Paper
  - Wire framing
  - View-only
- High-fidelity, narrow
  - Single feature or story
  - Complete functionality per specification
- Prototypes can accumulate into basis for production code
- Prototypes can _be_ production code in agile
  - (We can always refactor, right?)

Requirements Elicitation - Committee

- This process asks: what should we do?
- Problems:
  - Scope - too little discussion, or too much detail
  - Mistakes - wrong or incomplete information
  - Volatility - drift in correct information
- Systems:
  - Interviews
  - Committees
  - Prototypes

Requirements Elicitation - Committee

- This process asks: what should we do?
- Problems:
  - Scope - too little discussion, or too much detail
  - Mistakes - wrong or incomplete information
  - Volatility - drift in correct information
- Systems:
  - Interviews
  - Committees
  - Prototypes

Requirements Specification

- Wikipedia says: *A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.*
- Terms:
  - *Functional requirements*
  - *Non-functional requirements*
  - *Use cases*
- *This is a _document_.*

Functional Requirements

- Requirement: A condition or capability needed …
- Functional Requirement: A condition or capability needed for the system to serve the purpose for which it was designed.
  - Edit documents
  - Measure performance
  - Archive photographs
- If it did not do these things, it would not serve its purpose, or would not serve it completely.

Non-Functional (Ancillary) Requirements

- Requirement: A condition or capability needed …
- Non-Functional Requirement: A condition or capability needed for the system to work in the environment in which it was designed, at suitable levels of performance and quality.
  - Performance
  - Reliability
  - Portability
  - Legal compliance
- If it did not do these things, it would serve its purpose, but would not serve it as well or would not work in this environment.
- These are often things that must not happen, or rules that must be followed.

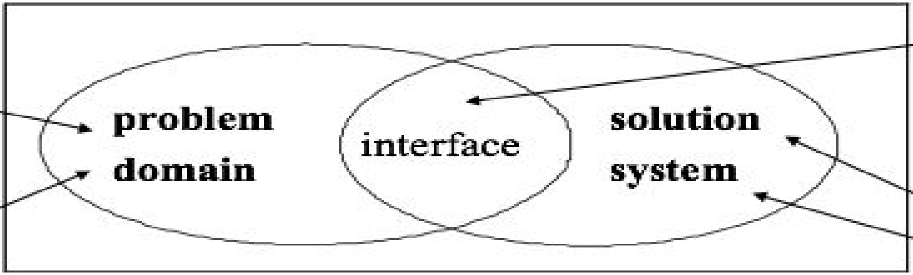
Requirements Validation

- Are we building the right product?
- Perform at every stage:
  - So, are you saying… ?
  - Does this work correctly?
  - How will we know this is correct?
- This last one is really important!
  - Testable evidence requires clear understanding
  - Testable evidence enables that verification stage
- Tests involve domains, requirements, and specifications
  - Domains - things understood to be true
  - Requirements - things that must happen
  - Specifications - descriptions of system state or behavior

The World and the Machine[1]

(or the problem domain and the system) These 6 slides are taken from Introduction to Analysis

- Domain properties (D)
  these are assumptions about the environment of the system-to-be
- Requirements (R)



Specification (S)

Hardware (C)
Software (P)

- **Validation question** (do we build the right system?) : if the domain-to-be (excluding the system-to-be) has the properties D, and the system-to-be has the properties S, then the requirements R will be satisfied.

  D and S $\Rightarrow$ R

- **Verification question** (do we build the system right?) : if the hardware has the properties H, and the software has the properties P, then the system requirements S will be satisfied.

  C and P $\Rightarrow$ S

- Conclusion:

  D and C and P $\Rightarrow$ R

[1] M. Jackson, 1995

Example

- Requirement
  - (R) Reverse thrust shall only be enabled when the aircraft is moving on runway.
- Domain Properties
  - (D1) Deploying reverse thrust in mid-flight has catastrophic effects.
  - (D2) Wheel pulses are on if and only if wheels are turning.
  - (D3) Wheels are turning if and only if the plane is moving on the runway.
- System specification
  - (S) The system shall allow reverse thrust to be enabled if and only if wheel pulses are on.
- Does D1 and D2 and D3 and S ⇒ R?
  - Are the domain assumptions (D) right? Are the requirement (R) or specification (S) what is really needed?

The assumption D3 is false because the plane may hydroplane on wet runway.

based on P. Heymans, 2005