Prof. Philip Koopman

Carnegie Mellon University

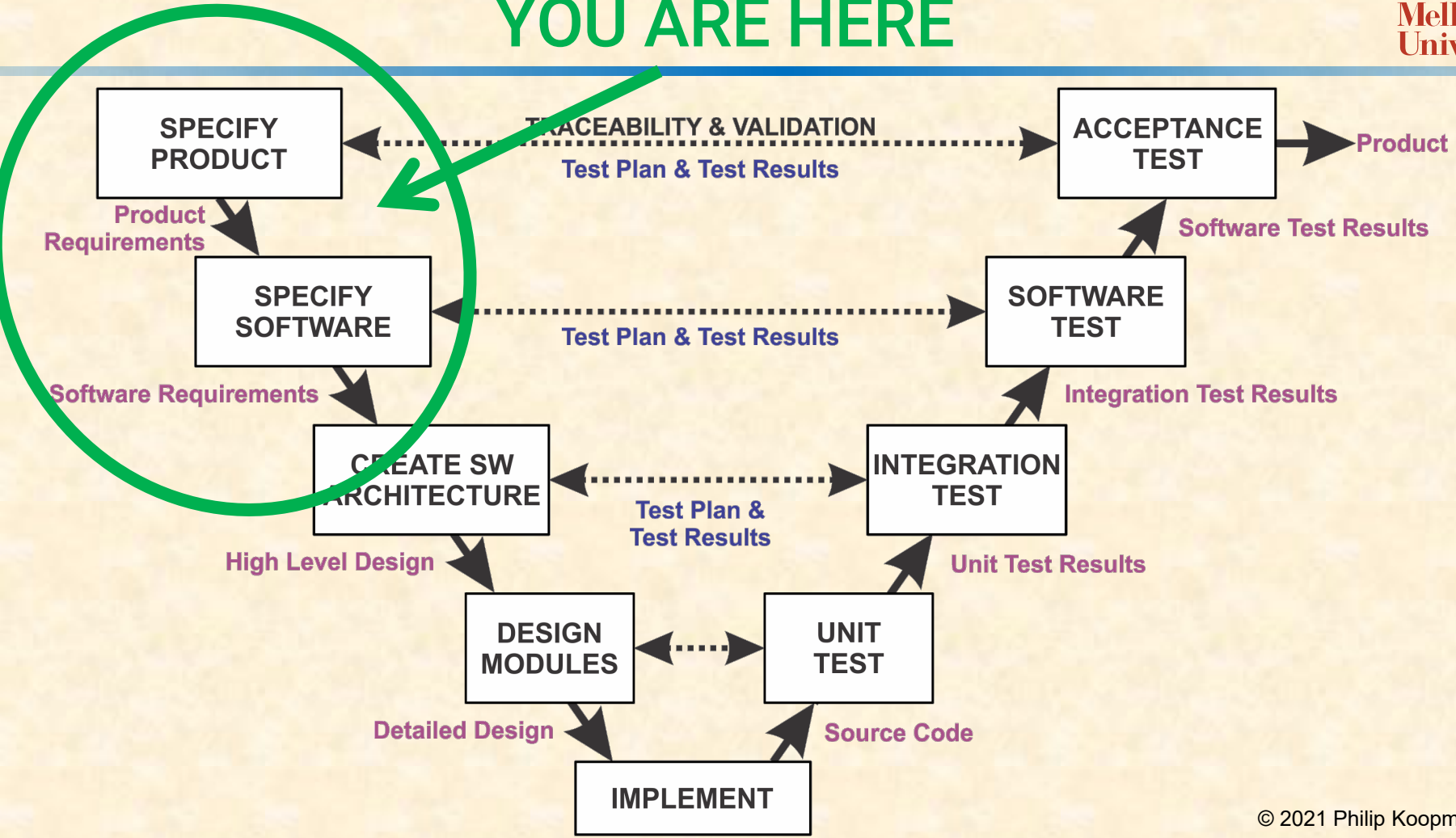
Electrical & Computer ENGINEERING

Embedded Software Requirements

"In spite of appearances, people seldom know what they want until you give them what they ask for. "

– Donald Gause and Gerald Weinberg, Are Your Lights On?

YOU ARE HERE



Requirements Overview

Carnegie Mellon University

- Anti-Patterns:
  - Requirements aren't written down
  - Requirements incomplete, imprecise
  - "Be like last version, except…"

- Requirements
  - Requirements faults can defeat a design before it is even built
  - Describe what system does
    - Also what it's not supposed to do
  - Precise, testable language
    - Each requirement traces to system test

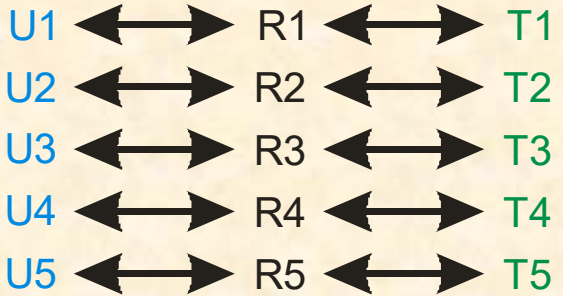
- 2005:
  $170M
  FBI Virtual Case File project terminated

- Requirements issues:
  - Requirements not defined when development contract signed
  - "We will know it when we see it"
  - Repeated requirements changes
  - Scope creep (new requirements added) of 80%

Characteristics of Good Requirements

Carnegie Mellon University

- Precise and minimally constrained
  - Describes what system should do, not how it does it
  - Uses "shall" to require an action; "should" to state a goal
  - If possible has a numeric target instead of qualitative term
    - Has tolerance (e.g., 500 msec +/- 10%, "less than X")
- Traceable & testable
  - Each requirement has a unique label (e.g., "R-7.3")
  - Each requirement cleanly traces to an acceptance test
  - Requirement satisfaction has a feasible yes/no test
- Supported within context of system
  - Supported by rationale or commentary
  - Uses consistent terminology
  - Any conflicting requirements resolved or prioritized

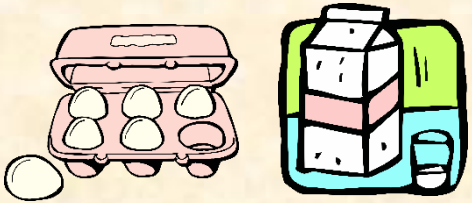


Problematic Requirements

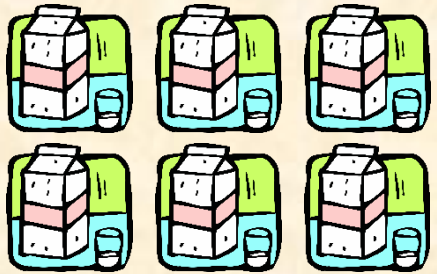
- Untraceable (no label)
  - System shall shut down when E-STOP is activated.
- Untestable
  - R-1.1: System shall never crash
- Imprecise
  - R-1.7: The system provides quick feedback to the user.
- No measurement tolerance
  - R-2.3: LED shall flash with a period of 500 msec
- Overly complex
  - R-7.3: Pressing the red button shall activate Widget X, while pressing the blue button should cause LED Z to blink instead of LED Y illuminating steadily, which would be accomplished via the yellow button.
- Describes implementation
  - R-8.3: Pressing button W shall cause two 16-bit integer values to be added, then …

Requirements Ambiguity

- A requirements engineer gets a text message:
  "On the way home, please pick up one carton of milk.
  And if they have eggs, get six."



- The requirements engineer comes home with:
  6 cartons of milk and no eggs.



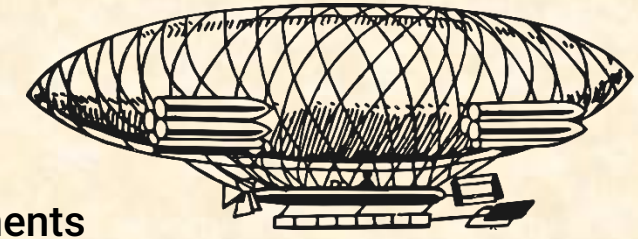
- Spouse: "Why did you buy six cartons of milk?!"
- Requirements Engineer: "They had eggs."

Adapted from: www.ganssle.com/jokes.htm.

Extra-Functional Requirements

Carnegie Mellon University

- Emergent properties (things hard to attribute to one component)
  - Performance, real-time deadlines
  - Security, Safety, Dependability in general
  - Size, Weight and Power consumption ("SWaP")
    - Often handled with an allocation budget across components
  - Forbidden behaviors ("shall not do X")
    - Often in context of safety requirements
    - "Safety function" is a way to ensure a negative behavior, but some behaviors are emergent



- Design constraints
  - Must meet a particular set of standards
  - Must use a particular technology
  - System cost, project deadline, project staffing

https://goo.gl/hT3nDU

Product vs. Engineering Requirements

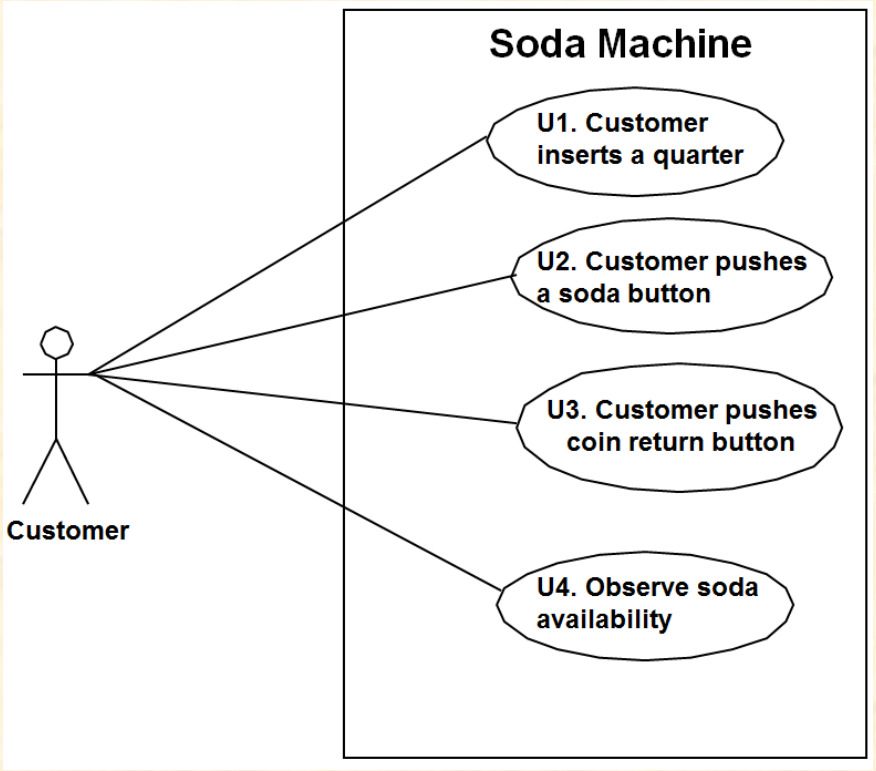
Carnegie Mellon University

- Product level requirements:
  what the product does
  - Example:
    "PR6. The clock shall support a user-settable audible alarm."
  - Gives a feature list of what the product actually does
  - Can be the interface between marketing and engineering

- Detailed functional/engineering requirements:
  how the product actually works
  - Example: "R5. Time set buttons shall change the alarm set time."
  - Embedded systems often have detailed requirements tied to operational modes
    - "R5. In Alarm Set Mode the time set buttons shall change the alarm set time."
    - "R6. Pressing the "+" time set button shall increase time value by one minute per button press according to the current set mode."

Requirements Approaches

- Text document with list of requirements
  - Works best if domain experts already know reqts.
  - Over time, this can converge to OK reqts.
- UML Use Cases
  - Different activities performed by actors
  - Requirements are scenarios attached to each use case
- Agile User Stories
  - Each story describes a system interaction
- Functional decomposition
  - Start with primary system functions
  - Make more and more detailed lists of sub-functions (creates a "functional architecture")
- Prototyping to elicit requirements
  - Customers know it when they see it
  - Sometimes a paper mock-up is enough



UML Use Case Diagram

Requirements Templates (EARS)

Carnegie Mellon University

- Easy Approach to Requirements Syntax (Mavin et al.) e.g.: https://bit.ly/2CQSF37
- [While/Where <precondition>] [when/if <trigger> then]
  <system> shall <response>
  - Ubiquitous: The touch screen shall have a response time of less than 250 msec.
  - State-driven: WHILE an external speaker is connected, the internal speaker shall mute.
  - Event-driven: WHEN a card is inserted, the card reader shall verify credentials.
  - Optional feature: WHERE a convertible roof is installed, a park/roof motion interlock function shall be provided.
  - Unwanted: IF an invalid value is entered THEN an error message shall be displayed.
  - Complex: combinations of the above
- Requirements issues to avoid:
  - Ambiguous, vague, complex, omitted, duplicated, wordy, implementation, untestable

Example Software Requirements

Carnegie Mellon University

National Aeronautics and Space Administration

Communications, Navigation, and Networking reConfigurable Testbed (CoNNeCT) Project		
Title: **Software Requirements Specification**	Document No.: **GRC-CONN-REQ-0084**	Revision: –
	Effective Date: **02/23/2010**	Page **18** of 61

Parent Req	ReqID	Requirement Text and Rationale	Prior-ity	Allocated To
FSRD-3714	SRS-3.2.6.12	The Software **shall** send data at a user data rate from zero up to and including 100 Mbps. *Rationale: The maximum data rate the Payload Avionics Software must send is 100 Mbps. Lower rates must also be handled.*	P2	PAS
FSRD-3133	SRS-3.2.6.13	The software **shall** send and receive data on two SpaceWire channels simultaneously at up to the maximum SDR interface data rate (full duplex) that can be sustained by both SDRs. *Rationale: When communicating with multiple radios, the Software will need to sustain an achievable data rate. In this requirement, it is defined as the minimum data rate of the two (or three, if possible) SDRs involved in the experiment. For instance, this data could be provided in the routing table. If two other radios are involved, then the data rate may change, based on the capability of those two radios (i.e. a new minimum interface data rate). This value should not be hard coded, but should have the capability for change, once on-orbit.*	P2	PAS

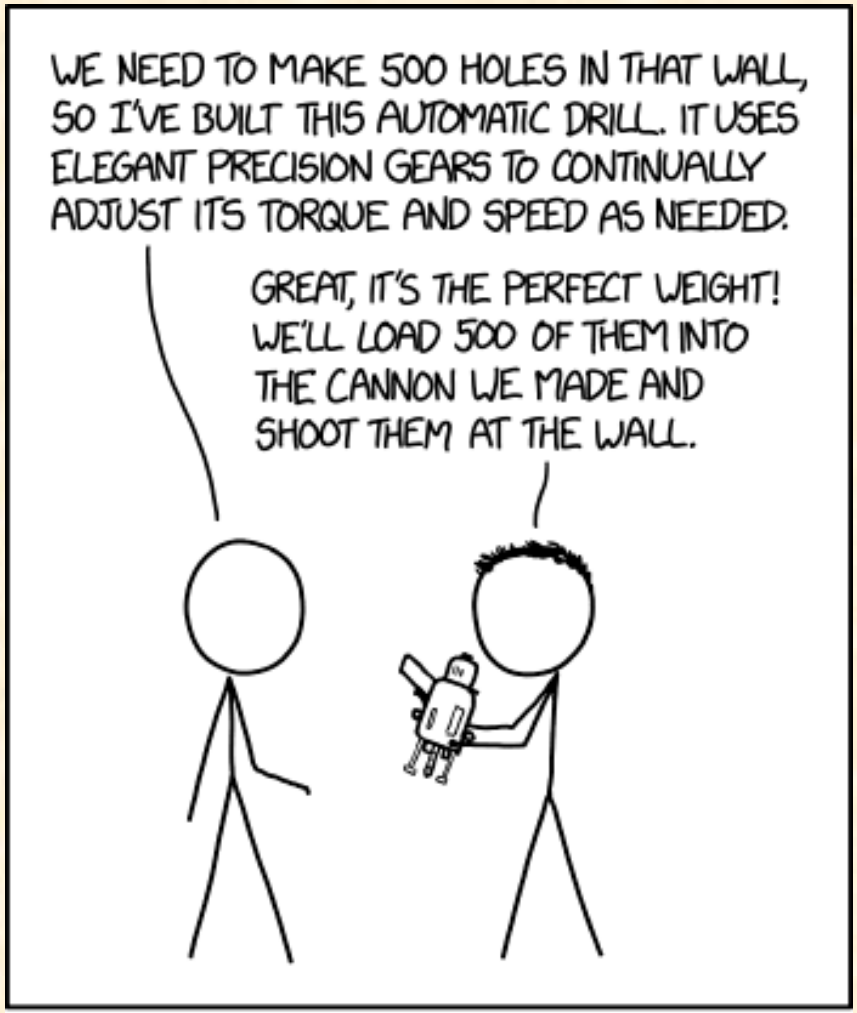
GRC-CONN-REQ-0084
EFFECTIVE DATE: 02/23/2010

https://goo.gl/qct5tL

Best Practices for Requirements

- Six C-terms for Good Requirements
  - Clear, Concise, Correct, Coherent, Complete and Confirmable
- Also:
  - Deal with extra-functional issues
  - Relate requirements to design flow
    - Associate with user stories or use cases
    - Trace to corresponding test
- Requirements pitfalls
  - Avoid unnecessary details and implementation
  - If it's missing from requirements, it won't get done
  - If it's not testable, you won't know if it got done

https://goo.gl/6H3dxi



HOW SOFTWARE DEVELOPMENT WORKS

https://xkcd.com/2021/