doug murray consulting

Managing Requirements

Getting the Right Requirements Right

doug murray
system architect

february 9, 2017

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

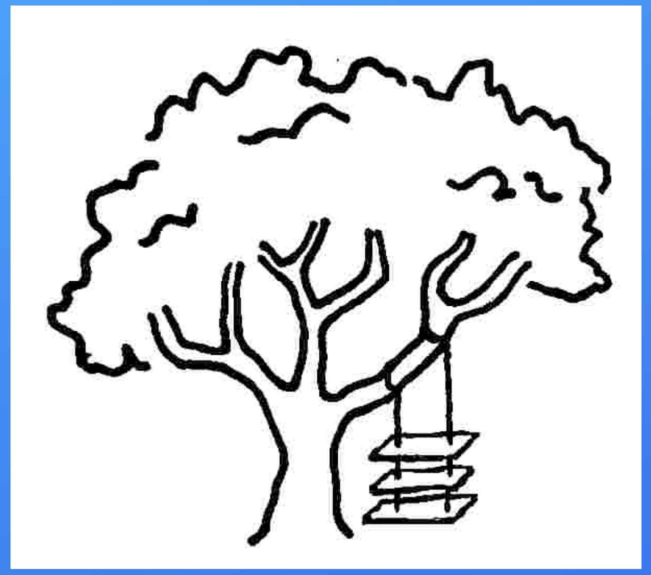
Where Do We Find Them?

Organizing Them

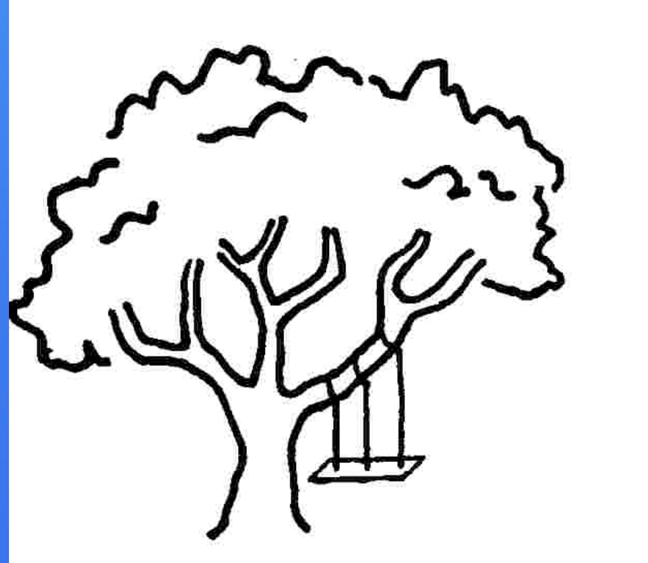
Why Do We Need Requirements?

- Gathering and understanding requirements make projects more predictable
- Bad requirements historically account for most of the rework done later in the project
- Reducing the number of defects caused by poor requirements can yield a 7 to 1 return on investment

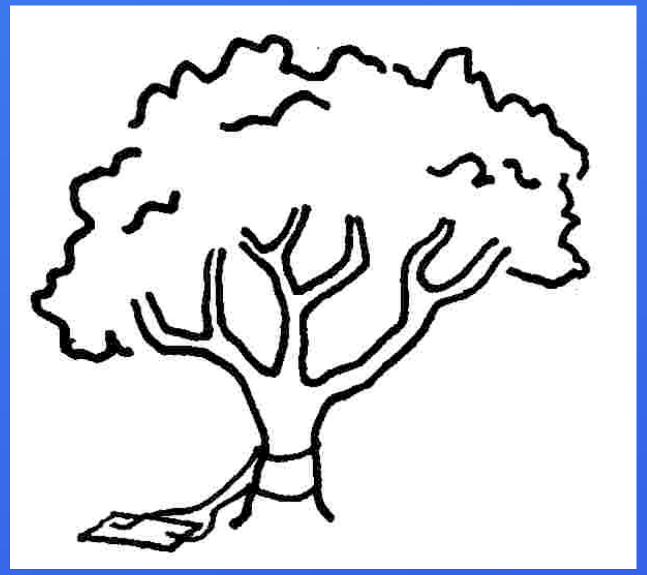
Why Do We Need Requirements?



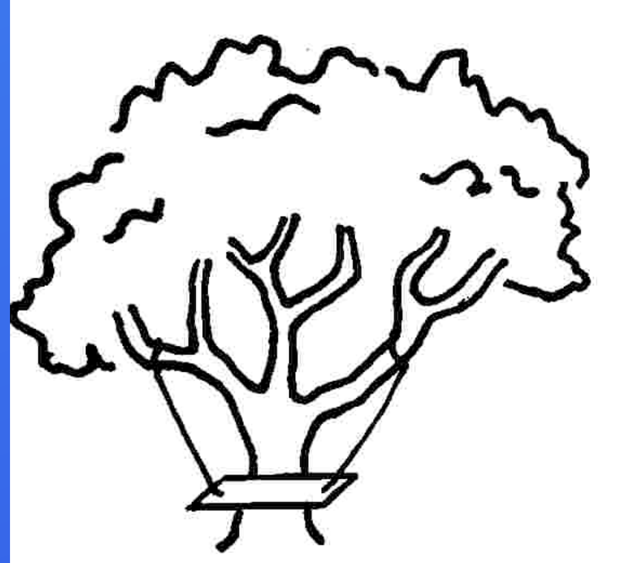
Marketing Specified



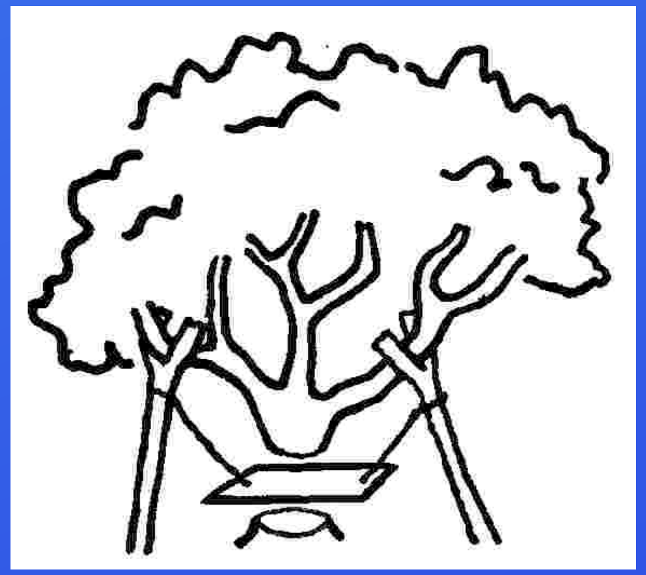
Management Approved



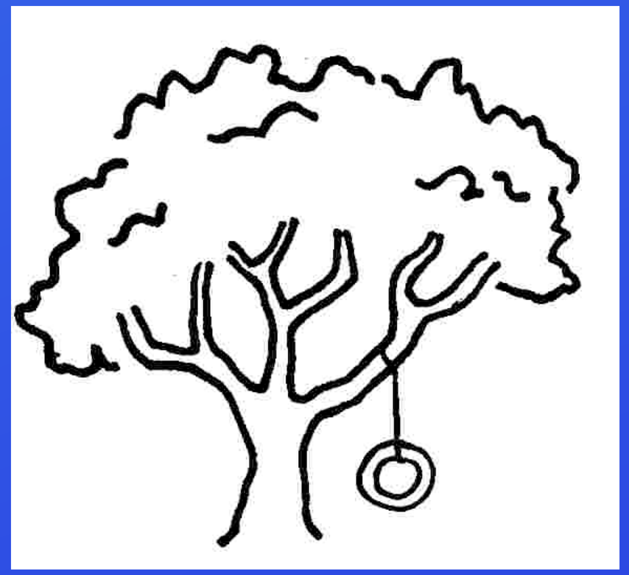
Engineering Designed



Manufacturing Built



Service Installed



Customer Wanted

Why Are Requirements So Important?

Most Project Work is Best
Driven By Requirements

- The product should provide only what is required of it, no more and no less
- Architecture and Design come directly from requirements
- All system-level tests relate directly to requirements

Why Are Requirements So Important?

- Customer Acceptance Tests, Product Validation and System Verification are all based on meeting requirements

- This presentation deals with a more intensive, rigorous environment intended to develop large medical devices
  - Smaller projects or those for an unregulated industry will still benefit from many of these ideas

Good Management Practices, Good Engineering Practices

Definitions

Some Basics for Our Purposes

Project the management of people, material and procedures to produce a product

Product something our business or institution needs to produce

System the technical part of the product that we will design, build and test

They Each Have Requirements

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Types of Requirements

Let's consider two broad categories of
Requirements for the Product and the System

Functional	•	What the System will do
Non-Functional	•	How well the System does it

Formal requirements are expressed with the word "shall"

Examples

Functional

- The automobile shall provide a mechanism to stop the motion of the vehicle

Non-Functional

- The automobile shall be capable of coming to a complete stop in less than three seconds when moving at a speed of 100 km/h

Examples

Functional	• The System shall provide a self-destruct mechanism
Non-Functional	• The self-destruct mechanism shall provide a delay of three seconds between the time of its activation and the time the system self-destructs

Examples

Functional

- The System shall start the process of producing electrons when the “Beam On” button is pressed

Non-Functional

- The System shall provide a visual indication that the “Beam On” button has been pressed within 200 milliseconds of the button being pressed

Common Pitfalls

- Anyone who has experience with or exposure to a certain technology tends to see requirements in terms of that technology

- It is common to place constraints on a solution for inconsistent reasons

Common Pitfalls

Requirements need to specify what is required, not how it should be implemented

✗ `The trend analysis system shall have a user interface running on Windows Vista using a standard Dell desktop computer`

Backwards compatibility with existing systems should be treated as Design Constraints, not testable requirements

Common Pitfalls

Requirements should be able to stand on their own, independent of the context in which they appear

✗ That software shall have an operating mode that complies with the following requirements

Common Pitfalls

The System must be the subject
of the requirement, not the user

✗ `The User shall be able to view beam diagnostics from a portable computer`

✓ `The System shall provide beam diagnostics viewing software for use on a portable computer`

Examples

✗ The User Interface shall be easy to use

✓ The RFQ shall produce 3.5 MeV protons

✓ The beam pipe flange in the research area shall have a kapton window capable of supporting a pressure gradient of $4x10^{-3}$ Pa

Examples

✗ The User Interface shall be easy to use

✓ The System shall be considered easy to use by at least 4 out of 5 trained dental assistants

Examples

✗ The data access software shall support JSON as the data exchange format for web access

✗ The clinical couch shall be able to support extremely obese patients

Words and Phrases to Avoid

- among others
- any
- efficient
- not limited to
- rapid
- simple
- user-friendly
- and so on
- as well as
- etc.
- optimal
- same as
- state of the art
- various
- and / or
- easy
- improved
- or
- several
- sufficient

Words and Phrases to Avoid

acceptable
and / or
average
improved
optimum
rapid
safe
simple
timely

adequate
any
easy
normal
or
reasonable
same as
state of the art
typical

among others
appropriate
efficient
not limited to
possible
reliable
secure
sufficient
user-friendly

and so on
as well as
etc.
optimal
proper
robust
several
suitable
various

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

What Makes a Good Requirement?

- **Understandable**; It must be communicated in a formal way
- **Measurable**; It must be testable to ensure it has been implemented
- **Feasible**; It must be possible for someone to implement

What Makes a Good Requirement?

Consistency

Requirements must not conflict with any other requirements at any level

Inconsistencies between them must be resolved before development can proceed

What Makes a Good Requirement?

Controlled

Requirements must be uniquely identified for the lifetime of the project

A history of changes made to each requirement should be maintained

Requirements are more usable and maintainable when related ones are kept together

What Makes a Good Requirement?

There are several characteristics of good requirements

Traceable

Unambiguous

Verifiable

Prioritized

Correct

Focused

Necessary

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

Good Requirements Will Exhibit These Characteristics

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

The best way to ensure correctness is to have experts review the requirement

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

Is the requirement really required? Determine where the it came from and ensure it was from a source of authority. Reduce "gold-plating" by repeatedly asking *Why* until you find the source

Requirements

of Good Requirements

Correct
Necessary
Focused
Verifiable
Traceable
Unambiguous
Prioritized

A requirement should address a single, testable need

Break compound requirements into separate statements

Requirements

of Good Requirements

- Correct
- Necessary
- Focused
- **Verifiable**
- Traceable
- Unambiguous
- Prioritized

Each requirement has to be testable

We need to know if the requirement has been met

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

Each requirement needs a reference to its source. Also, the tests which verify that a requirement has been met and the designs to implement the requirement need to reference that requirement

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

Someone reading the requirement must be able to draw only one conclusion from it.

Different stakeholders must arrive at the same interpretation

Requirements

of Good Requirements

Correct

Necessary

Focused

Verifiable

Traceable

Unambiguous

Prioritized

Each requirement should have an indication of priority, ideally with only a few (3) levels

What Makes a Good Requirement?

Traceable

Requirements shall provide a way to reference a more general source, such as a higher-level (product) requirement or a Use Case

Links (references, dependencies) can be made to requirements from design elements, test cases and other artifacts coming later in the development process

What Makes a Good Requirement?

Prioritized

Critical, high-priority requirements are visible and can be met within given cost and schedule constraints

Lower priority requirements can be postponed if necessary.

Requirements

need to include Attributes

Traceability, Priority and most qualities imply that one or more attributes are saved with each Requirement

We need to keep this information with each Requirement for all stakeholders to see

Some attributes are not really optional

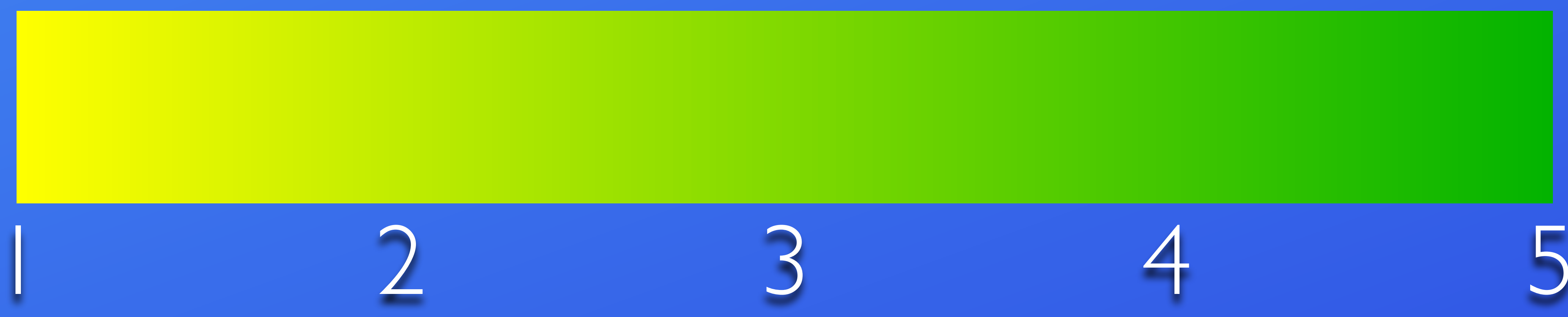
Requirements

need to include Attributes

- Requirement Text
- Unique ID
- Requirement Type
- Summary Phrase
- Rationale (justification)
- Originator
- Fit Criterion
- Priority
- Customer Satisfaction Index
- Customer Dissatisfaction Index
- History
- Conflicts

Customer Satisfaction

The degree of happiness if this requirement has been successfully implemented in the product



1 2 3 4 5

Not Interested

Very Pleased

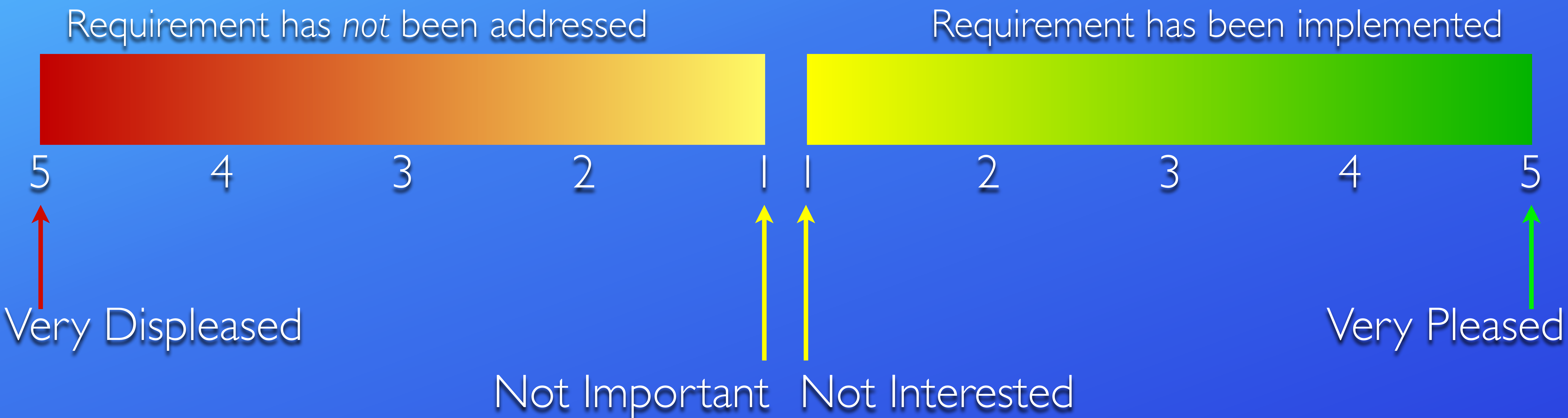
Customer Dissatisfaction

Measure of unhappiness if this requirement
is missing from the final product

5 4 3 2 1

Very Displeased ... Not Important

The Customer Perspective



Requirement has *not* been addressed

Requirement has been implemented

5 4 3 2 1

1 2 3 4 5

Very Displeased

Not Important

Not Interested

Very Pleased

Non-Functional Requirements

Qualities of Good Requirements
and the attributes associated with them
are used for both Functional
and Non-Functional Requirements

Non-Functional Requirements

Non-Functional Requirements address the qualities that the System must possess, essentially describing how well the System will perform its Functional Requirements

Non-Functional Requirements

Non-Functional Requirements are critical to the success of the product

Not just performance, but usability and the product's look and feel

Non-Functional Requirements

- Access
- Appearance
- Extensibility
- Internationalization
- Maintainability
- Privacy
- Reliability and Availability
- Scalability
- Style
- Accessibility
- Capacity
- Fault Tolerance
- Learning (Training)
- Personalization
- Productization
- Robustness
- Security
- Supportability
- Adaptability
- Ease of Use
- Integrity
- Longevity
- Precision or Accuracy
- Release
- Safety-Critical
- Speed and Latency
- Understandability

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Requirements

Where do they come from?

Project requirements come from cost, schedule and “performance” goals

Product requirements come from marketing research and customer needs

System requirements are derived from product and project requirements

Requirements

some examples

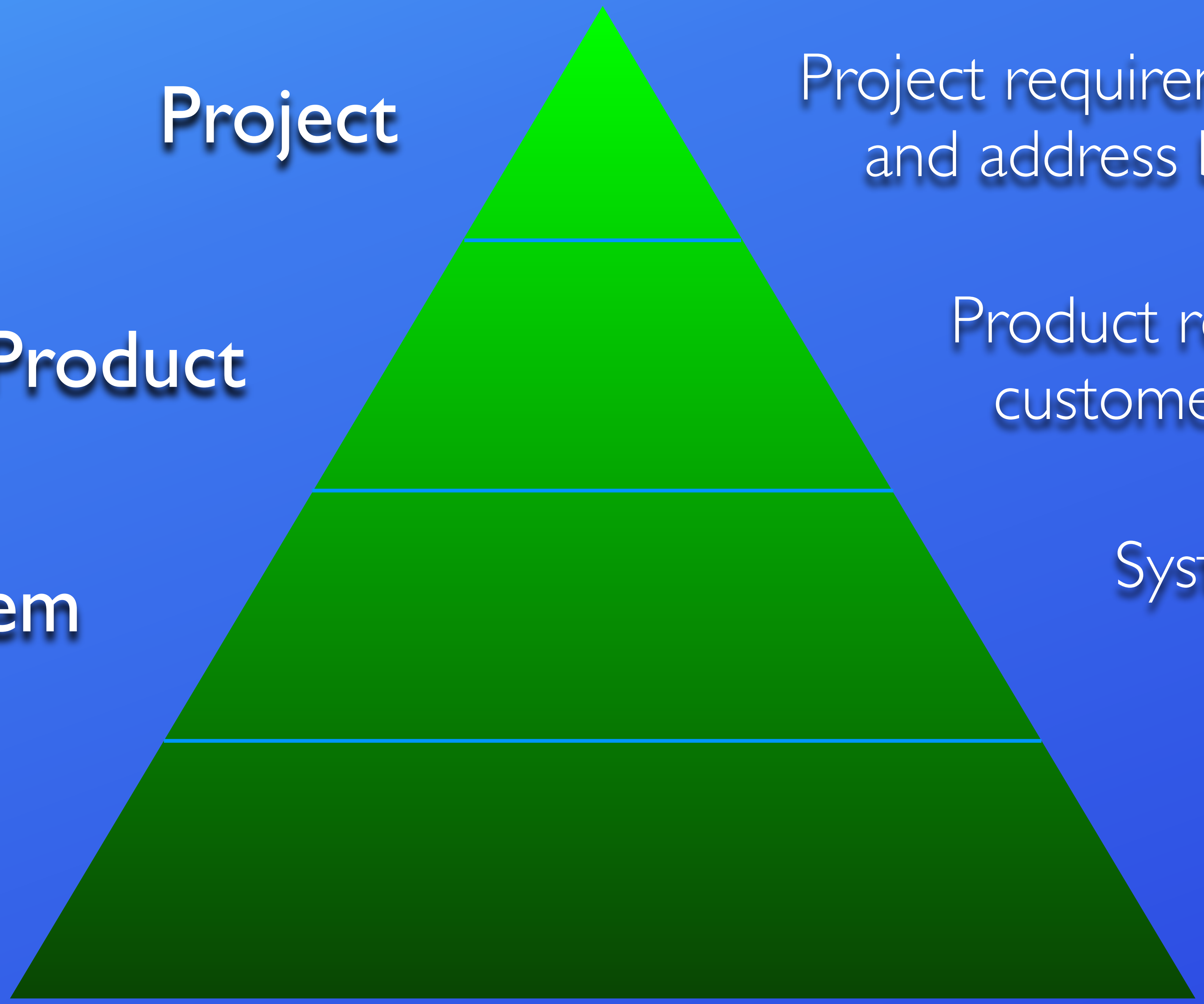
Project The Super Duper CT product will sell for $300,000 and accommodate larger patients

Product The SDCT product can image patients weighing 380 lbs

System The System shall provide a patient imaging table capable of supporting patients in a supine position weighing 380 pounds or less

Requirements

from general to specific needs



Project

Project requirements are few in number and address business or team goals

Product

Product requirements are written for the customer with terms they understand

System

System requirements are written for engineer's designs and tests

Subsystem

Large systems might have subsystems needing more technical detail

Where Do Requirements Come From?

Typically from Marketing - "the Voice of the Customer"

- From previous personal experience
- From interactive workshops with experts
- From direct observation in a daily work setting
- From insight, interpolation, understanding, abstraction

Where Do Requirements Come From?

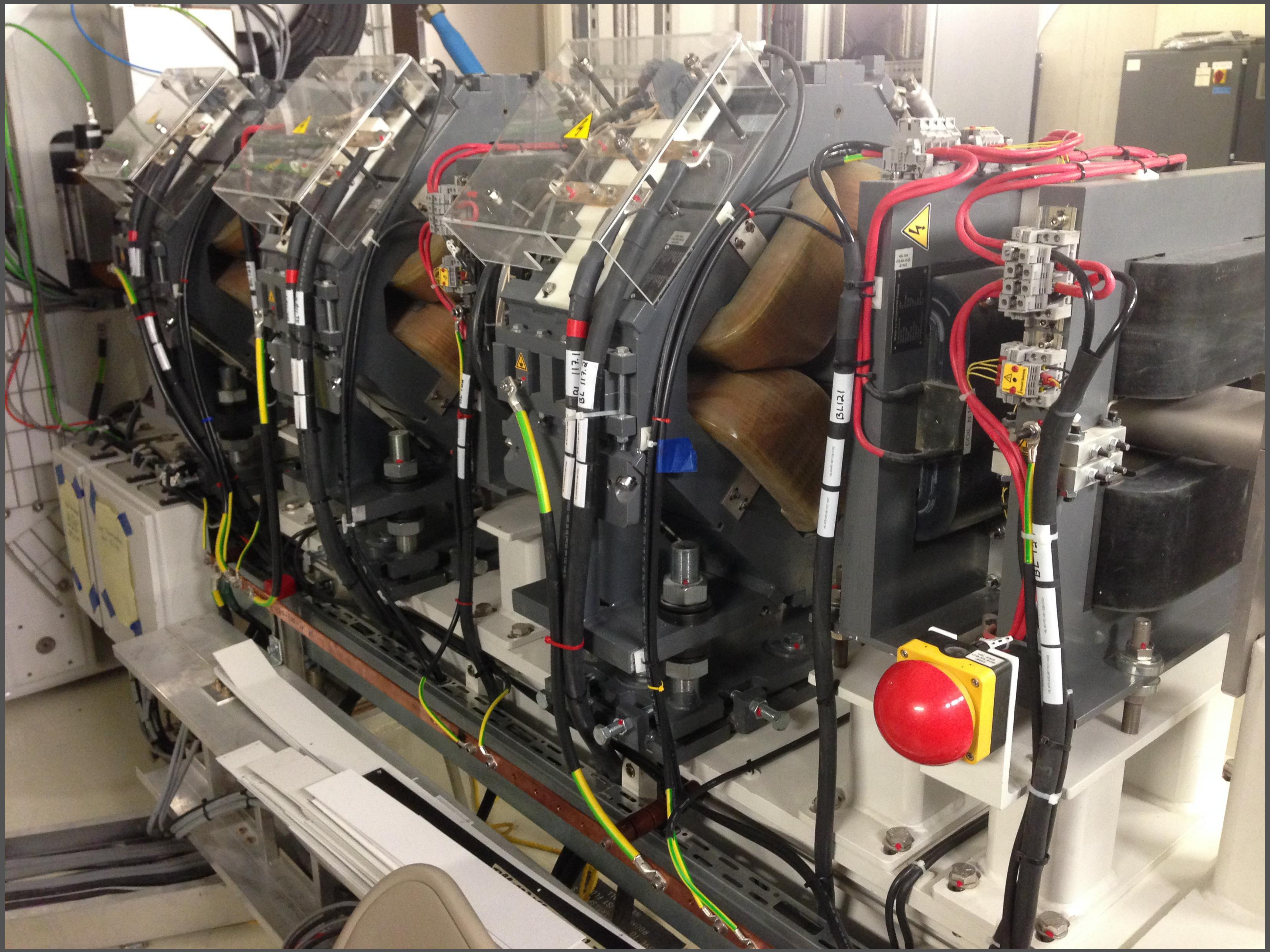
- Use Case Analysis
- Brainstorming
- Customer Feedback and Specific Requests
- Corrective and Preventative Action Reports (CAPA)
- Competitive Reviews

Where Do Requirements Come From?

- High-level Requirements come from business strategy, market research and "the voice of the customer"

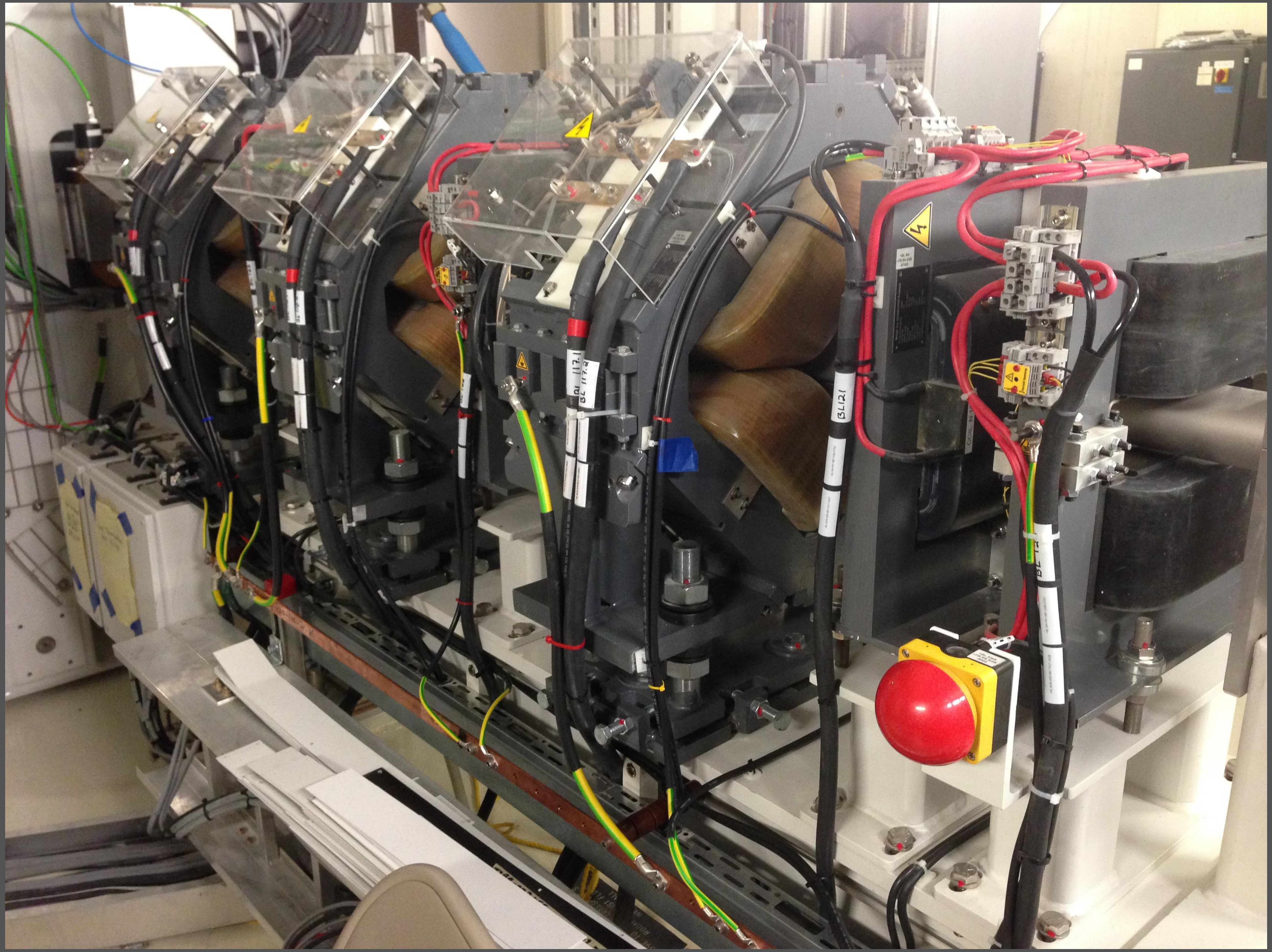
Where Do Requirements Come From?

- High-level Requirements come from business strategy, market research and "the voice of the customer"
- Lower-level Requirements can be based on customer input only if they understand specific technologies or specifications that are important to *their* business



Where Do Requirements Come From?

- Lower-level Requirements come from technical experts, based on the Product or Customer high-level requirements
- These experts will translate informal requirements into formal ones that are unambiguous and testable, and able to be realized by way of a buildable system



Where Do Requirements Come From?

Management will suggest that once in place, requirements need never change

"…we've already built this, why look for new requirements for something we already understand?"

Why re-invent the wheel?

To Improve! We want steel-belted radial tires

How Do We Gather Requirements?

first steps, gather the most general ones

Actor ✓ Identify the stakeholders of the product, the roles they play

Use Case ✓ Discover the actions the system must perform for them

Persona ✓ Think of specific individuals in each of those roles

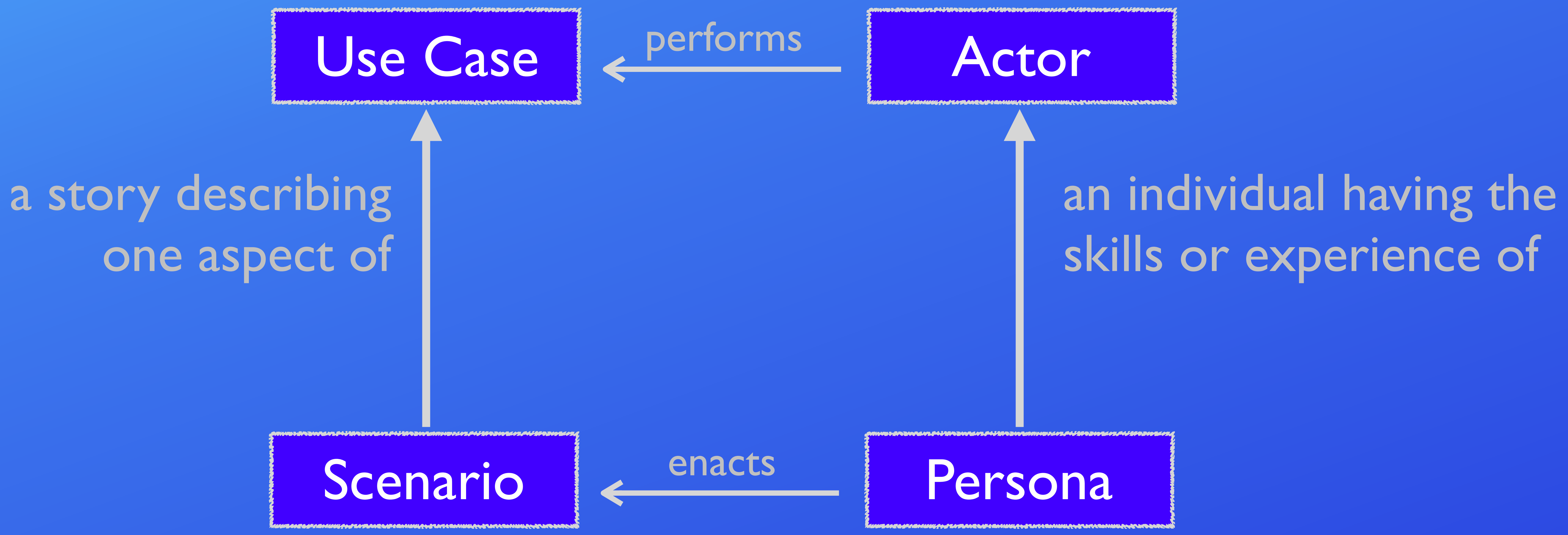
Scenario ✓ Consider how the system operates as those individuals work

How Do We Gather Requirements?

examples

Actor	Radiotherapist
Use Case	Position patient on treatment table
Persona	Jennifer, 32 year old female Oncologist, working in RT role this day
Scenario	She brings an obese patient in a wheelchair to the couch, lowers it to its lowest point. She decides to request help from Jim, her associate to transfer the patient to the couch.

How Do We Gather Requirements?


Use Case
performs
Actor
a story describing one aspect of
an individual having the skills or experience of
Scenario
enacts
Persona

How Do We Gather Requirements?

Actor	The *types* of users involved with the Use Case
Use Case	Describes activities the product needs to perform
Persona	Specific but fictitious individuals involved with the scenario’s story
Scenario	Detailed account of how that individual makes use of the system, told as a story using natural language

How Do We Gather Requirements?

Actor

Requirements exist to satisfy *all* stakeholders. Identify the roles played by people that will interact directly with the System, or be affected by it. That could include managers expecting reports from it, or patients expecting treatment from it.

Use Case

A good set of Use Cases helps ensure that we're not missing any requirements. Without Use Cases, the missing requirements are hard to find because they have no source or basis in necessity

How Do We Gather Requirements?

Requirements should be traceable back to the
Use Cases and Scenarios that inspired them

Functional Requirements are often gleaned from Actors and Use Cases
Non-Functional Requirements are often gleaned from Personas and Scenarios

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

Agenda

Why Requirements?

Writing Requirements

Requirements of Good Requirements

Where Do We Find Them?

Organizing Them

How Do We Organize?

Functional Requirements are usually determined first, but requirements gathering is an iterative process

Architecture, Design or Development work can start when only a few functional requirements are understood

Beware - Requirements and the development work that they trigger can change, especially at the project's start!

Organizing Requirements

- Requirements will ultimately be recorded and stored with their attributes in a repository

- Requirements should be organized in a way that helps engineers, testers, quality assurance experts and other stakeholders do their work

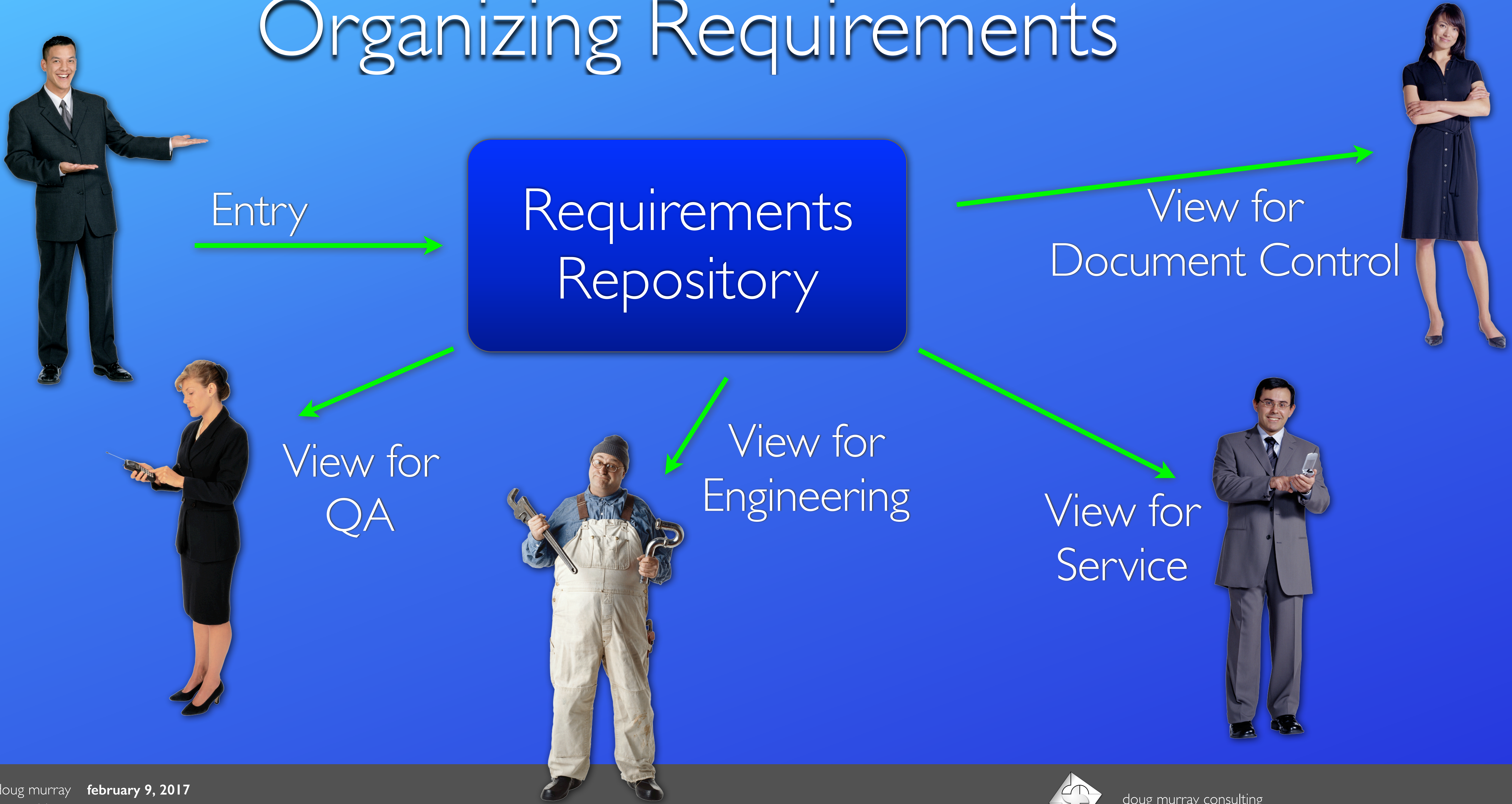
Organizing Requirements

- Modern software tools will generate documents (artifacts) based on the repository content

- They will also provide version control for each requirement

- These tools can also provide stakeholders with custom views of the requirements

Organizing Requirements



Requirements Repository

Entry

View for Document Control

View for QA

View for Engineering

View for Service

Organizing Requirements

Some projects need to maintain other requirements-related information which is not specific to any single requirement

- Project Drivers
- Project Constraints
- Project Issues
- Design Constraints

By keeping additional sections in the repository, more complete and useful documents can be generated

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

These categories ensure the requirements are grouped logically so engineers can easily find them and documents are easily maintained

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

They also make it easier to develop complete architectures and designs, development plans, good test plans and protocols. All but the first two are optional.

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

The fundamental, essential subject matter of the product. They describe what the product has to do or the processing actions it must take

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

The properties that the functions must have, such as performance, usability or security. These are often referred to as qualities of the product.

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

These remaining categories don't provide true requirements, but can be used to better communicate those things that affect the requirements

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

These are restrictions on the product such as the budget or time available to build it, market assumptions, naming conventions and more

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

These are technical constraints upon the design, often from legacy issues; use a 4-20 mA signal, using TTL level logic or being backwards compatible with an existing API

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

The business related forces driving the project forward. Trade shows, market changes, supplier schedules

Organizing Requirements

Functional

Non-Functional

Project Constraints

Design Constraints

Project Drivers

Project Issues

The business related forces holding the project back, or impacting its success. Project risks, postponed requirements, upgrade paths

Organizing Requirements

At the System level or lower, Functional Requirements are kept separate from Non-functional ones

Non-functional requirements will probably change more often than Functional ones during the development process.

Keeping them separate can make documents easier to maintain and tests easier to repeat

How Do We Organize?

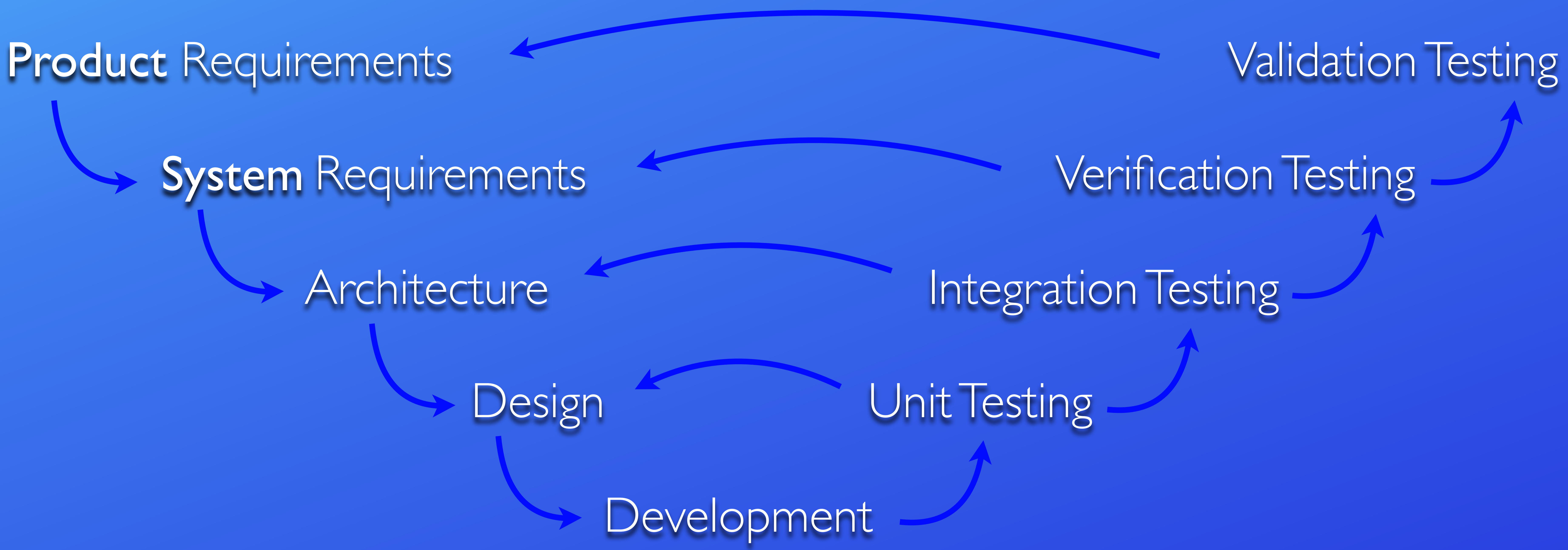
Project requirements are informal and often communicated through a concept document

Product or customer requirements are a bit more specific, communicated through a requirements document

System requirements are quite formal and require thorough review, especially in regulated environments

Project Model

Requirements play a crucial role in the "V" process



Product Requirements

System Requirements

Architecture

Design

Development

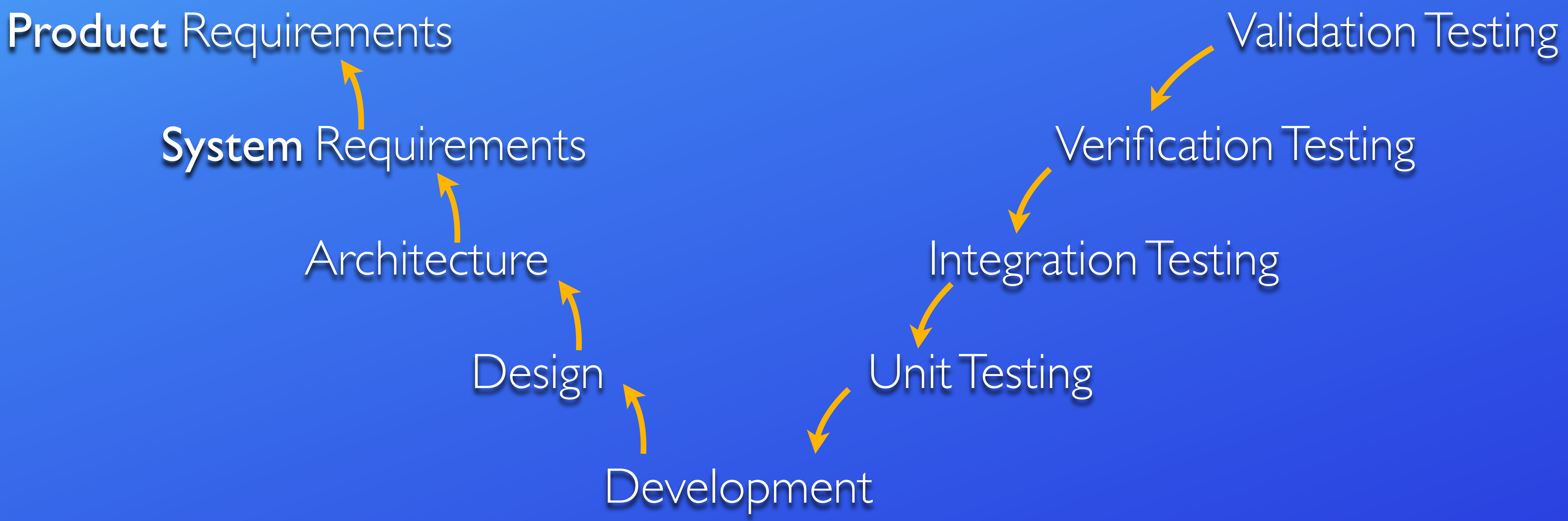
Unit Testing

Integration Testing

Verification Testing

Validation Testing

Dependencies



Product Requirements

Validation Testing

System Requirements

Verification Testing

Architecture

Integration Testing

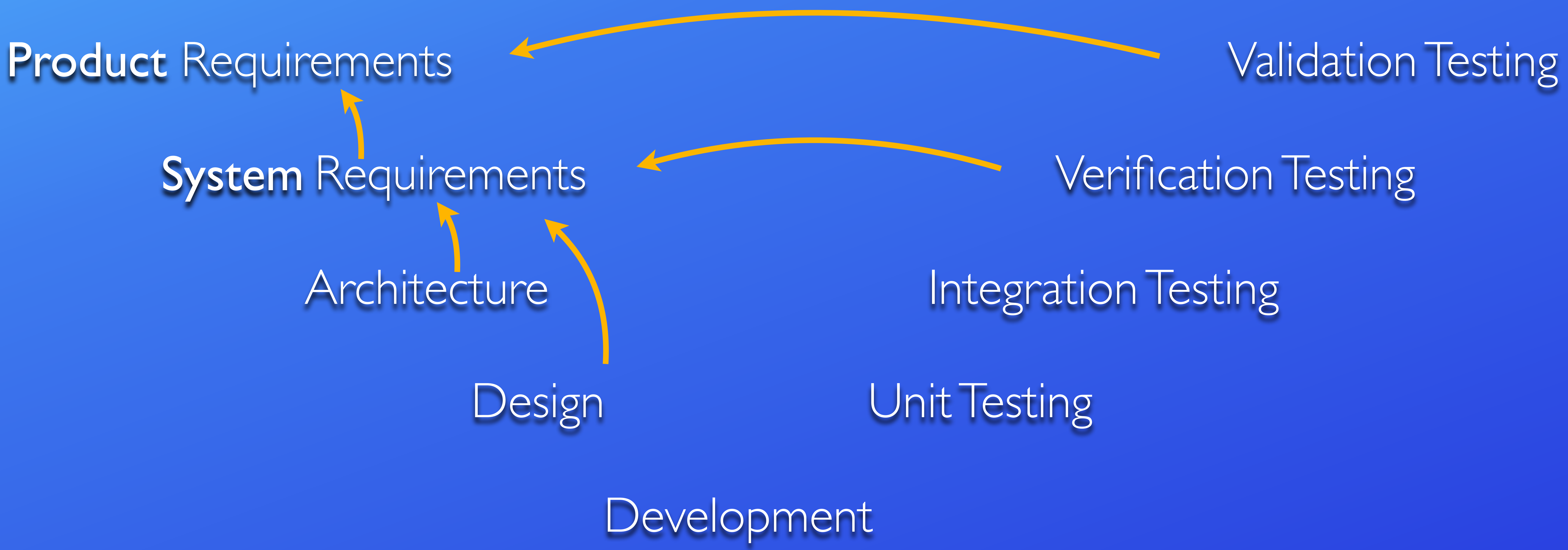
Design

Unit Testing

Development

Traceability

Tests and possibly some Design Elements will Reference Requirements



Product Requirements

Validation Testing

System Requirements

Verification Testing

Architecture

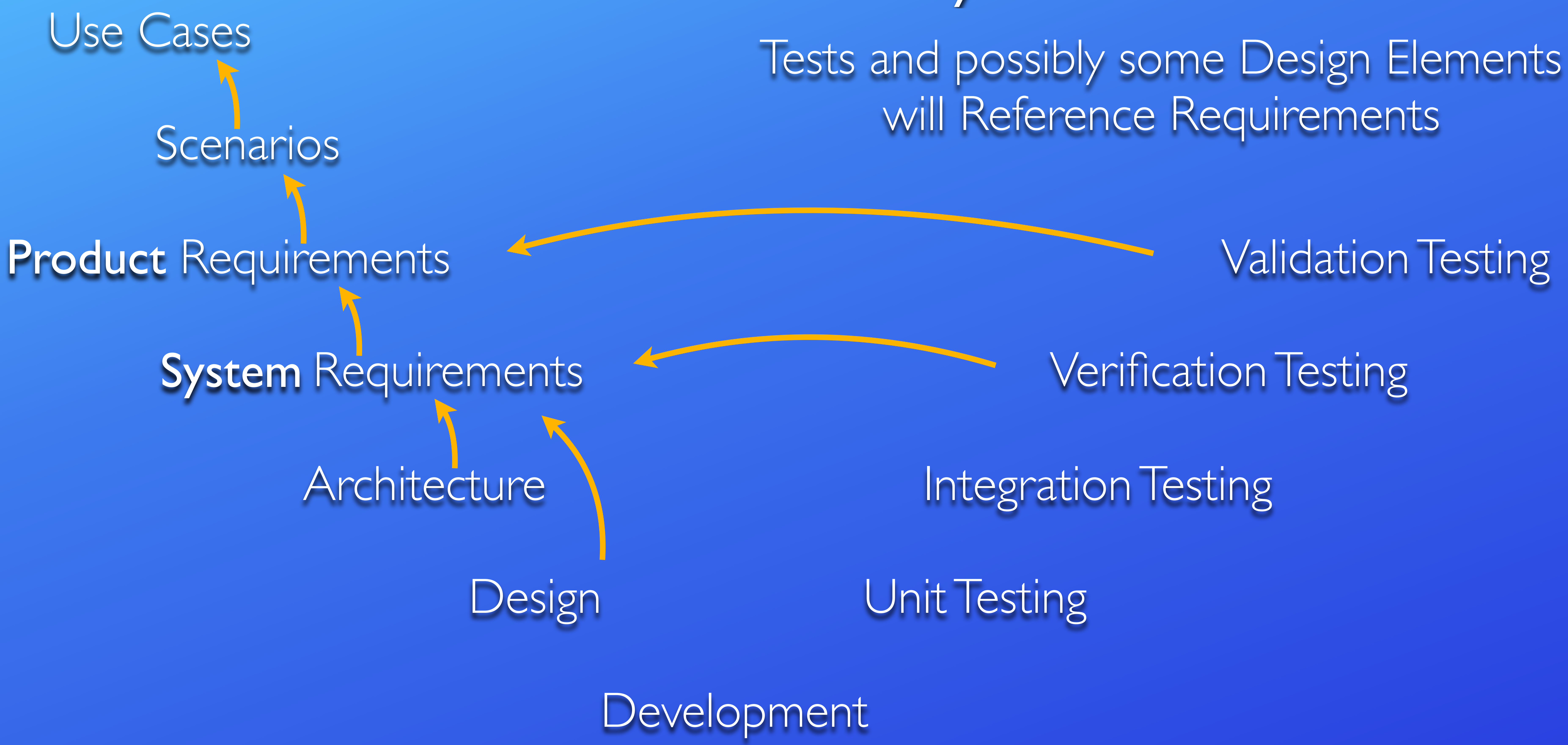
Integration Testing

Design

Unit Testing

Development

Traceability



Use Cases

Scenarios

Product Requirements

System Requirements

Architecture

Design

Development

Unit Testing

Integration Testing

Verification Testing

Validation Testing

Tests and possibly some Design Elements
will Reference Requirements

V & V

- Verification tells us that we've built the system right
- Validation tells us the we've built the right system

Organizing Requirements

A Product Requirements Document
Describes the Intended Use of the Product

A System Requirements Specification describes what
the System will do and how well it will do it

For larger systems, subordinate Requirements Documents describe
enough detail to build a testable subsystem

Organizing Requirements

A Product Requirements Document
Describes what we will Validate

A System Requirements Specification indicates
What to Test in the Verification Process

Subordinate Requirements Documents describe
What must be Tested to Enable Verification

Organizing Requirements

- At some point in time, requirements are reviewed and a "baseline" established
- Requirements will still change or be added
- After the baseline has been set, changes will affect the project's schedule and cost
    - Each change must be reviewed and the impact considered

Organizing Requirements

- A Complete end-to-end Requirements Specification is a daunting task, especially for complex or large systems
- Requirements Specification must be a team effort

None of us is as smart as all of us

Bibliography

1. *IEEE Standard Glossary of Software Engineering Terminology*, IEEE STD 610.12-1990, http://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html
2. *Mastering the Requirements Process—Third Edition* by Suzanne Robertson and James Robertson, Addison-Wesley, 2013. ISBN 978-0-321-81574-3
3. *Meeting the Challenges of Requirements Engineering*, Bill Thomas, Published in SEI Interactive, March, 1999, http://www.sei.cmu.edu/news-at-sei/features/1999/mar/Spotlight.mar99.pdf
4. *Software Engineering Spring 2005 Lecture 4*, New York University, Published on the web Spring 2005, http://www.cs.nyu.edu/courses/spring05/V22.0474-001/lec/lec4_h4.pdf
5. *Writing Quality Requirements*, Wiegers, Karl E., http://www.processimpact.com/articles/qualreqs.html

Bibliography

6. *Writing Good Requirements*, Hooks, Ivy, http://www.complianceautomation.com/papers/writingreqs.htm, INCOSE 1993

7. *Writing Good Requirements*, Karl Wiegers, Published in Software Development Magazine, May 1999, http://www.cs.bgu.ac.il/~elhadad/se/requirements-wiegers-sd-may99.html.

8. *Writing Good Requirements (A Requirements Working Group Information Report)*, Ivy Hooks, Published in Proceedings of the Third International Symposium of the INCOSE, Volume 3, 1993, http://www.itmweb.com/essay543.htm.

9. *Writing good requirements is a lot like writing good code*, Jim Heumann, IBM, 30 June 2004, http://www-106.ibm.com/developerworks/rational/library/5170.html.

doug murray consulting

Managing Requirements

Thank you for your attention