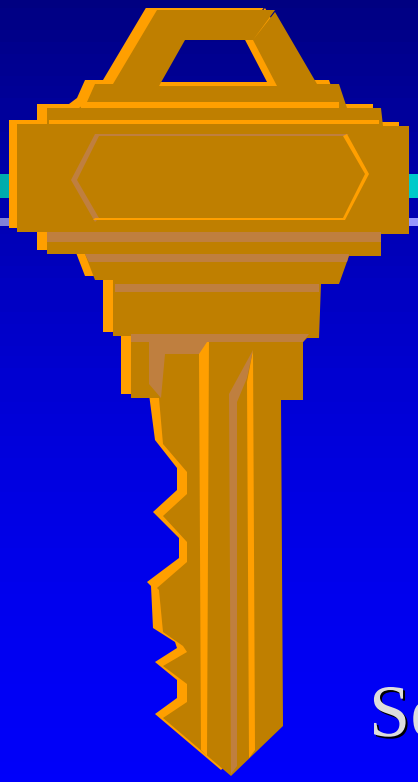
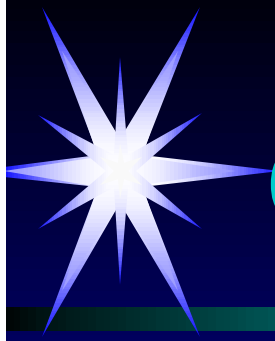
Writing Better Requirements
The Key to a Successful Project



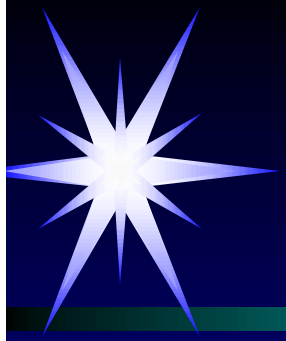
Kimberly Roberts
Senior Application Engineer
kimberly_roberts@qssinc.com

QUALITY SYSTEMS & SOFTWARE



Objectives

- Learn how good requirements definition can benefit your organization.
- Learn what the differences are between User Requirements, System Requirements and Architectural Design Requirements
- Explore the anatomy of a requirement and characteristics of a good requirement



Why projects fail

- Incomplete requirements - 13.1%
- Lack of user involvement - 12.4%
- Lack of resources - 10.6%
- Unrealistic expectations - 9.9%
- Lack of executive support - 9.3%
- Changing reqs/specs - 8.7%
- Lack of planning - 8.1%
- Didn’t need it any longer - 7.5%

Sources: Standish Group
Scientific American

Why projects succeed

- User involvement - 15.9%
- Management support - 13.9%
- Clear statement of reqs - 13.0%
- Proper planning - 9.6%
- Realistic expectations - 8.2%
- Smaller milestones - 7.7%
- Competent staff - 7.2%
- Ownership - 5.3%

Sources: Standish Group
Scientific American

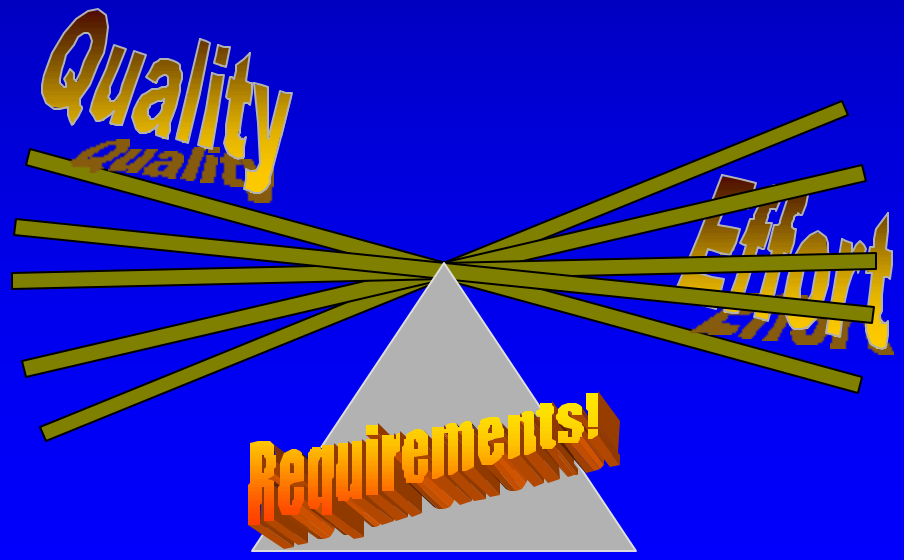
And the question is……

How do we make a project successful?

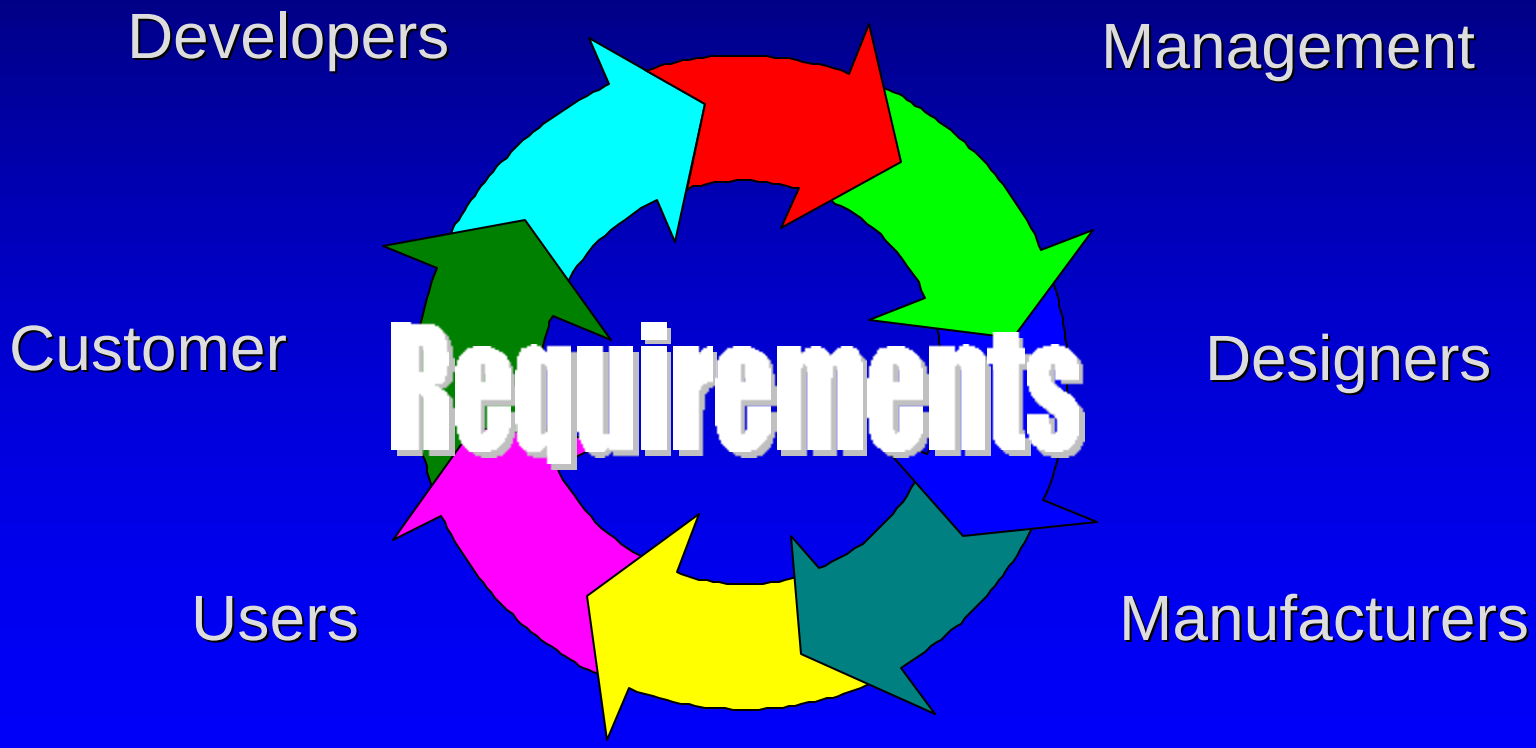
Write Better Requirements….

It’s a HIGH LEVERAGE activity!!

Requirements allow you to IMPROVE Quality with LESS effort.



Requirements are how we communicate



What are Requirements?

(They are the "To-Do" List of the Project Team)

- List of WHAT the Users need
- List of WHAT the System must do to Satisfy their needs
- List of WHAT components must be built
- List of WHAT each component must DO, and HOW they will INTERACT

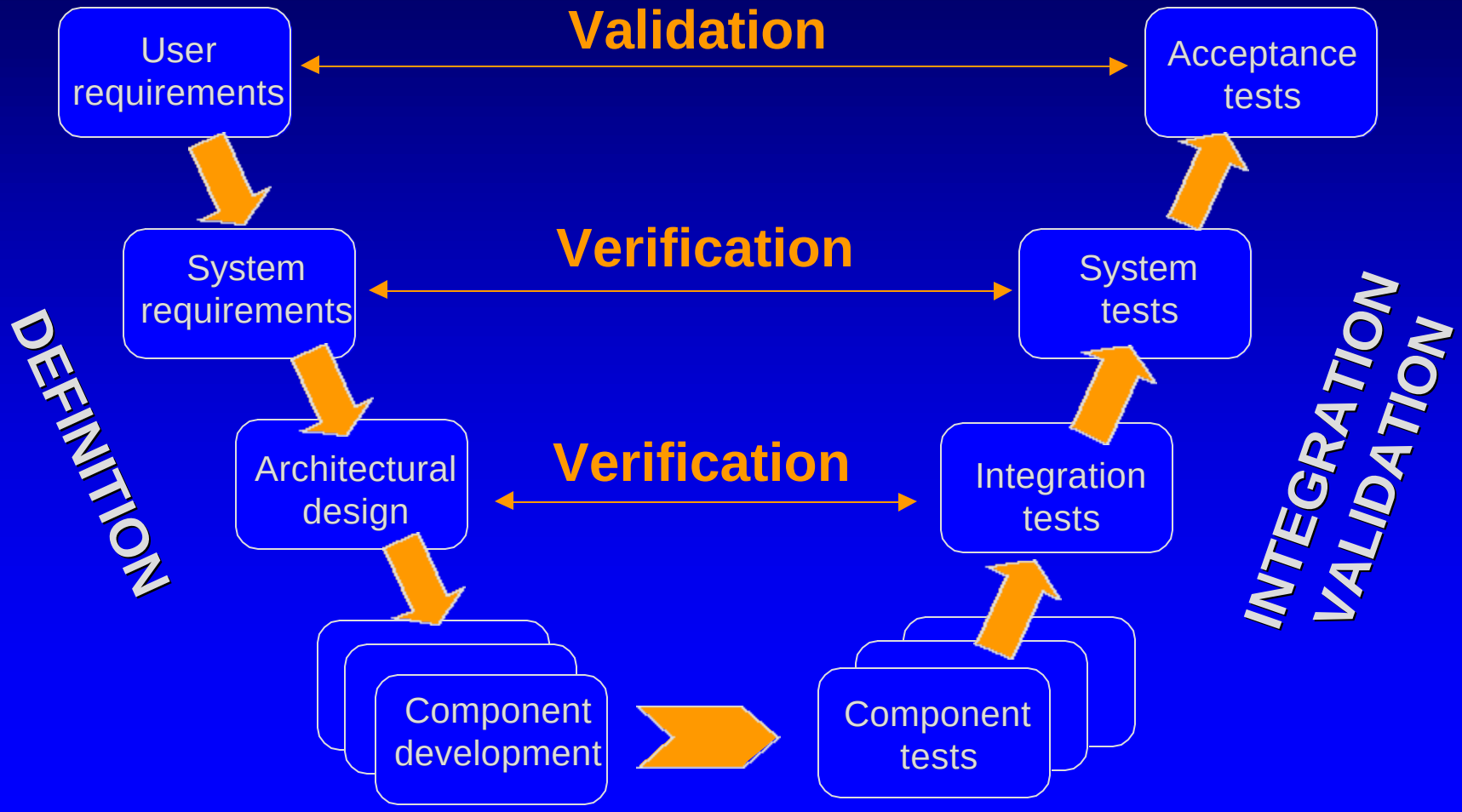
Requirements define the quality of the system

When are Good Requirements Essential?

- If you are building anything intended for someone else's use
- If you are building systems that do more than one thing
- If you are building anything that will be used more than once
- If you are building something that interacts with other systems
- If you are building a system that requires a specified level of performance
- If you are building anything that involves payment or other consideration (in other words, a *contract*)

Good requirements can make a difference

Requirements Ensure You Build the Right Project the Right Way



Basic Types of Requirements

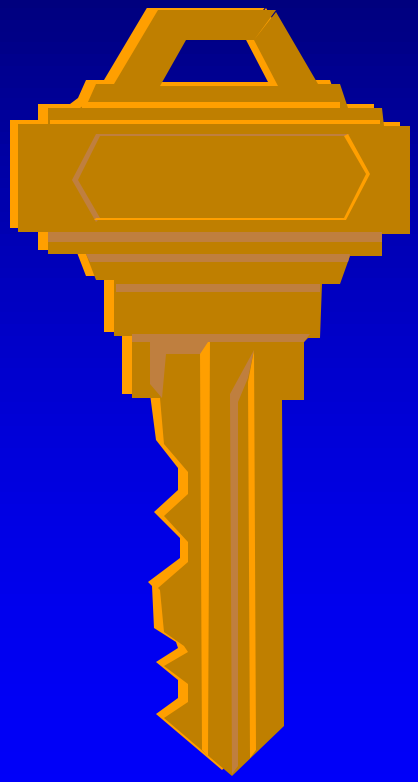
- **TYPES OF REQUIREMENTS**
  - **Functional Requirements**
  - **Non-Functional Requirements**
  - **Constraints**
  - **Design Guidelines**
- **TYPES OF DOCUMENTS**
  - **User or Business Needs Requirements**
  - **System or Functional Requirements**
  - **Architectural Design**
    - **“Element” Requirements (Software & Hardware)**
    - **Interface Definition Requirements**

Document Definitions

- **User/Business Needs Requirements**
  - **What the System should do *from the Users Perspective***
- **System/Functional Requirements**
  - **What the System will do *from the builders Perspective***
- **Component Requirements**
  - **Design level list of all things that each component must do. Any Programmer/Engineer can build one from this document**
- **Interface Requirements**
  - **Description of the *Protocol and Physical Interface* between Components of the System**

Architectural Details

Key Number 1 : Know where we are going.



USER/BUSINESS NEEDS REQUIREMENTS

Capture user requirements to understand what user problems your product will solve.

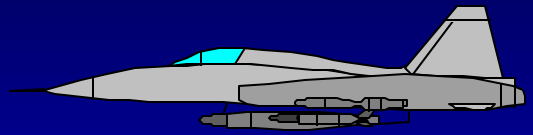
If we don't know where we're going….

- we never get there.
- we think we're there but in reality we are not.
- we finally get there but it takes a long time to arrive.

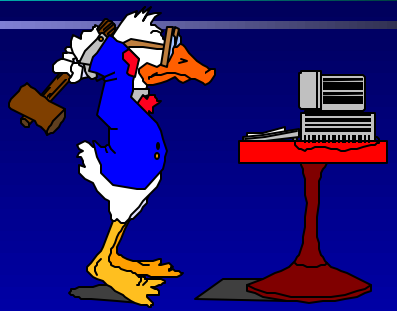
We have the time to do it over and over again but we never have the time to do it right the first time!

How do we know where we're going?

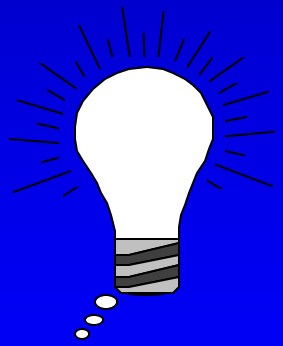
Maintainers



End users

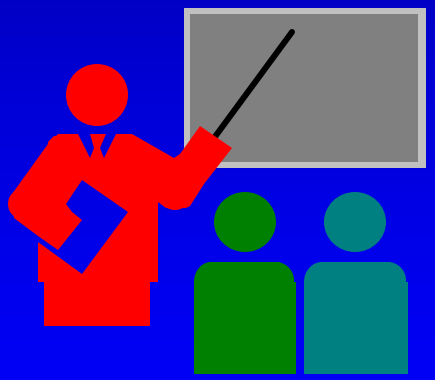


Testers



Marketing

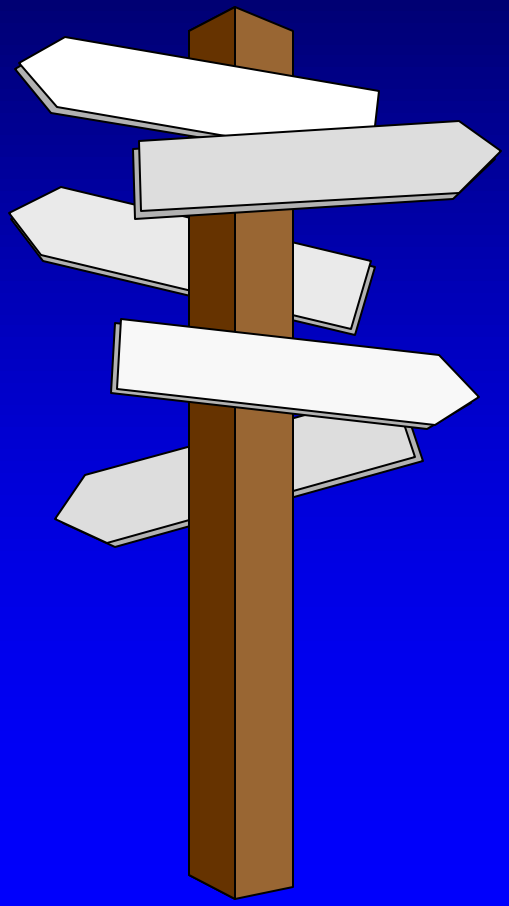
Customer



Corporate executives

The different types of users tell us!

Good user requirements are essential!



Without good user requirements there is no clear direction for building a product….
Product team members can head in different directions!

User Needs

- Generated by:
  - Business Analysis
- Generated From:
  - User Requests
  - Business Rules

Corp Business Rules

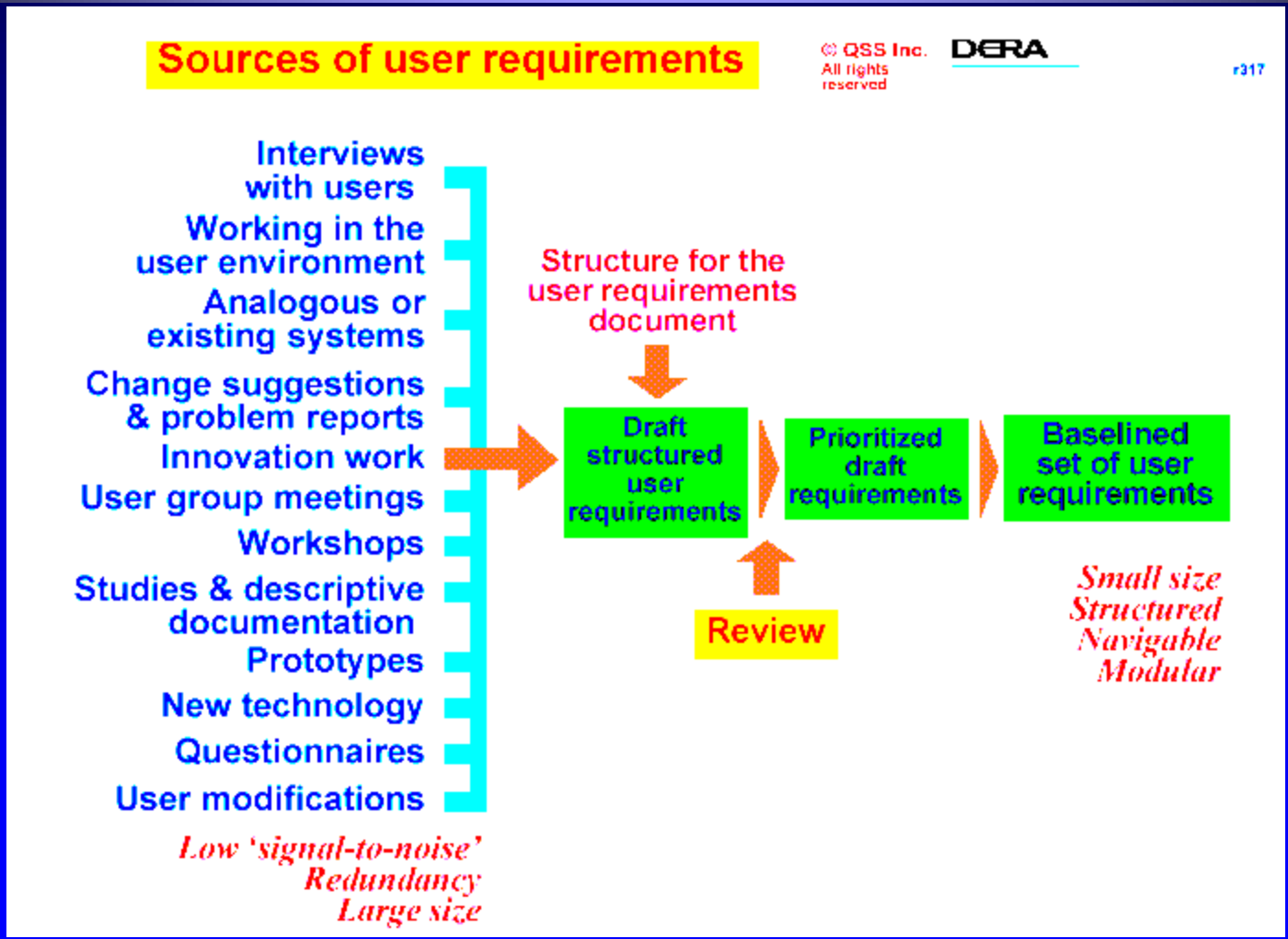
End User Rqmts

RFP Contract

View By:

- **Priority**
- **Importance**
- **Acceptable**
- ***You* Choose**

Where do User Needs Come From?



Anatomy of a User Requirement

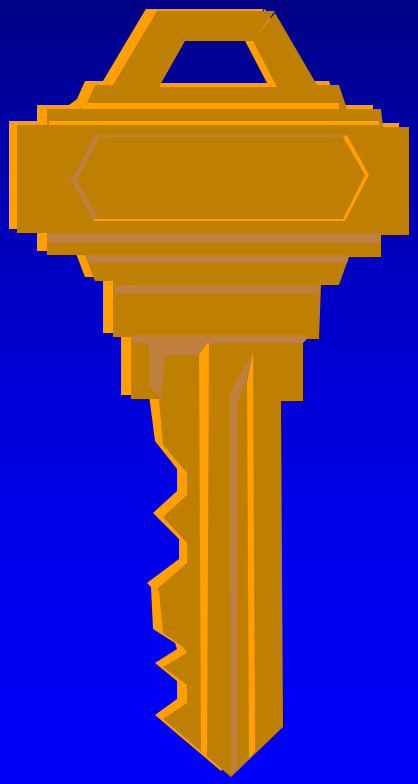
"The order entry clerk shall be able to complete ten customer orders in less than 2 hours"

This requirement identifies a real user type and an end result.

It also defines the success criteria in measurable terms.

The challenge is to seek out the user type, end result, and success measure in every requirement.

Key Number 2 : Know What We Are Doing



SYSTEM/FUNCTIONAL REQUIREMENTS

"What do WE need to know to build this?"

What Makes Up a System?

Components of System Requirements:

Key requirements in here

- Descriptive Elements
- Functional or Behavioral Breakdown
- Performance
- Interfaces
- Non-Functional Requirements
- Traceability to User Requirements

The core of the system requirements document

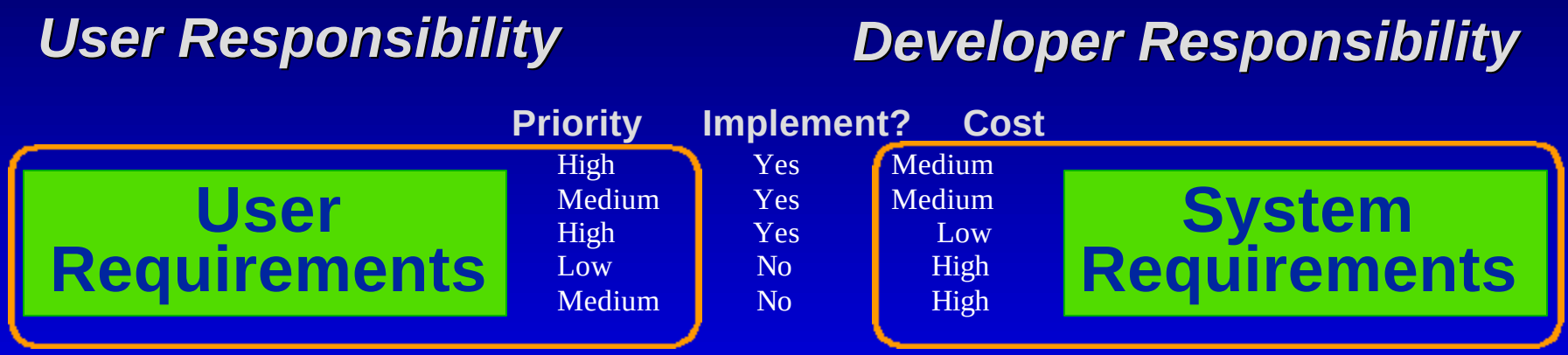
Use attributes

Often a large part of a document required for content to “make sense”

System Requirements Document

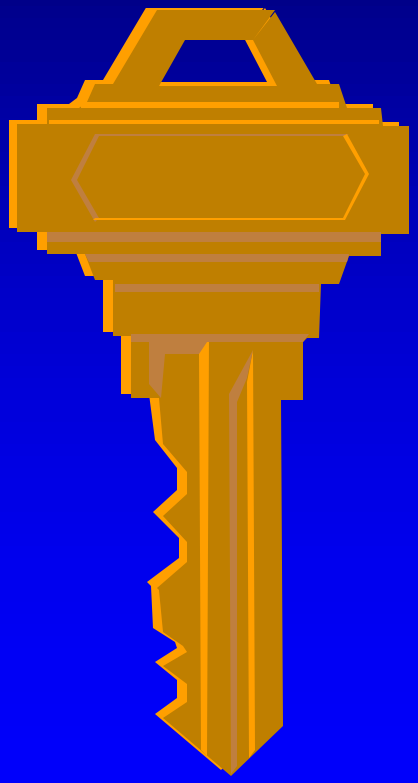
- Defines a model of the system to be built - not the system
- Defines some mixture of functionality, behavior, performance, and systems constraints
- It’s a model, not a complete system definition
- Organized by functionality or logical layout
- As implementation free as possible
- Every statement verifiable, with level and nature of test as attributes
- Defines professional constraints -- e.g. from different disciplines, working environment, induced environment, safety, etc.
- Owned by systems engineers, viewable to everyone including customers and designers

Different Attributes Between User and System Requirements



- Users are responsible for prioritizing a user requirement
- Developers for costing the requirement
- In the end, the user representative should decide whether cost is worth the result

Key Number 3 : Know What to Build and Implement



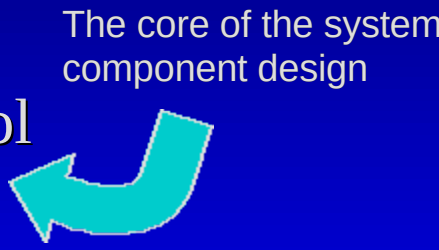
ARCHITECTURAL DESIGN REQUIREMENTS

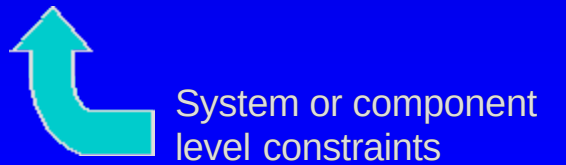
Decomposition to detailed design and subsystem components

What Makes up the Design?

Components of Architectural Design:

- Descriptive Elements
- Component Behavior/Control
- Component Functionality
- Component Interfaces
- Component Layout
- Dependencies & Resources
- Test Criteria
- Traceability to System Requirements





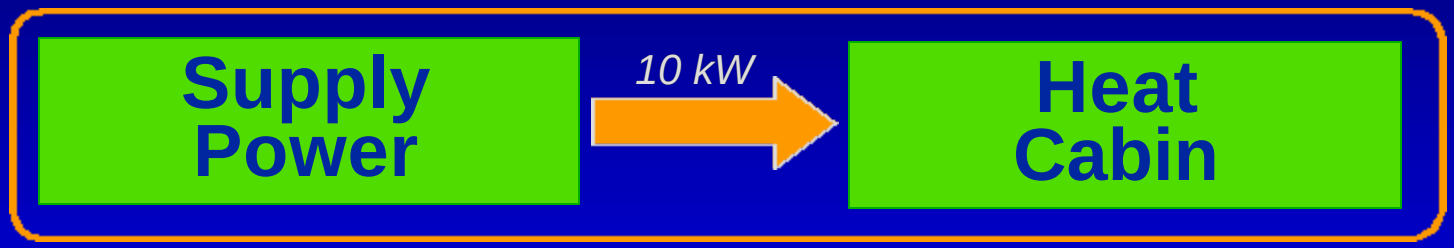
Architectural Design Document

- Defines WHAT is to be built, how it behaves, how it is laid out
- Defines key components, interfaces, control structures & external systems
- Detailed enough for optimization, partitioning to contractors, and definition of configuration item structure
- Basis for all milestones
- States what the system/component IS, not just its functionality
- Every statement verifiable and ready for integration tests with attributes to track states and methods of verification
- Created by designers, owned by developers, viewable by all
- Cost estimation should be well definable at this stage

Design versus System Level Requirements

System Requirements

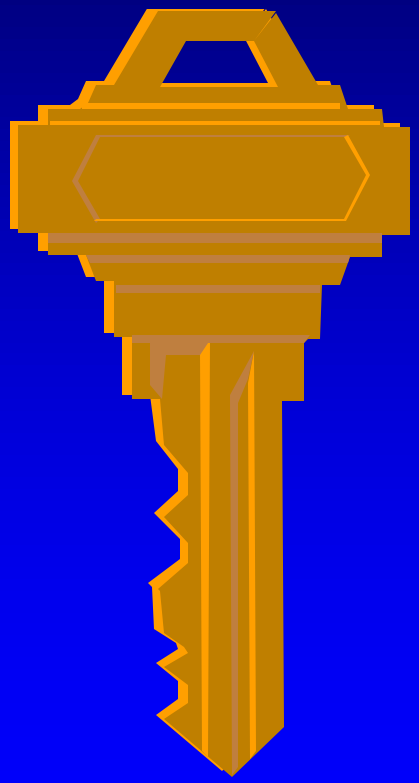
Functional definition of two functions and an interface



Architectural Design

Equivalent interface definition in design phase

Key Number 4 : Take Command



WRITE BETTER REQUIREMENTS!

The Anatomy of a Better Requirement…..

Typical Requirement Problems

- Inappropriate Level
- Poor Definition
- Lack of Structure and Control
- Undefined Ownership
- No Traceability
- No Completion Criteria

What is a Requirement?

- **Must form a complete sentence (not a bullet list of buzz words, list of acronyms, or "sound bites")**
- **States a subject and predicate where the subject is a user**
- **Consistent use of the "to be" verb:**
  - ***shall*, *will* or *must* to show mandatory nature**
  - ***should* or *may* to show optionality**
- **Has end result**
- **Contains success criteria or is testable in nature**

Types of Requirements

- Functional Requirements
  - Capability that the system must perform
- Non-Functional Requirements
  - Conditions that must be met that are not explicit capabilities (ie: the system must run with 32M of RAM)
- Constraints/Guidelines

Characteristics of Good Requirements

Each Individual Requirement Should Be:

- **Clear** — Avoid confusion
- **Brief** — Short and simple
- **Verifiable** — Testable and verifiable
- **Traceable** — Uniquely identified and can be tracked

Each Requirement Should Be Annotated with Attributes:

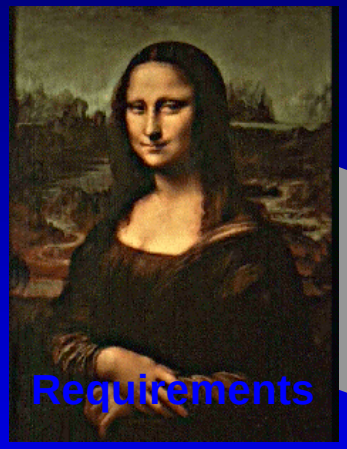
- **Priority** — Emphasize important requirements
- **Source** — Who requires it?
- **Urgency** — When?
- **Identity** — Requirements must be uniquely identifiable
- **Comments** — Clarification recorded when needed
- **Query/Response** — All can benefit from discussions

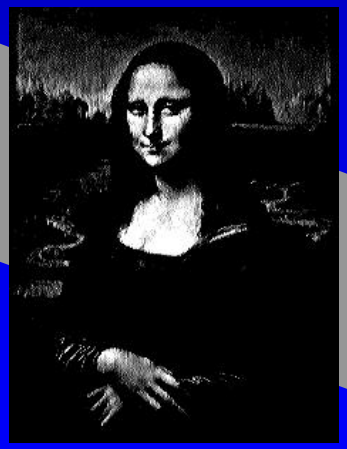
Characteristics of Good Documents

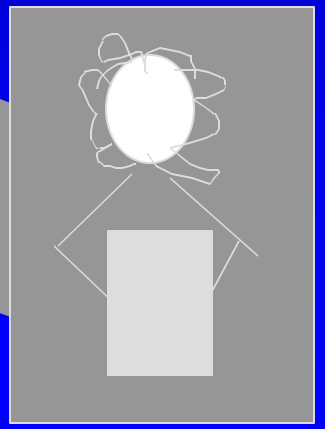
Each Collection of Requirements Should Be:

- **Complete** — Are all requirements expressed?
- **Balanced** — Are requirements at a consistent level of detail?
- **Modular** — Can system be changed without unforeseen side effects?
- **Correct** — Are user needs correctly expressed?
- **Consistent** — No contradictions exist between requirements?
- **Realistic** — Can we really build it?
- **Role/State/Mode** — Are requirements associated with user roles/system states?
- **Type Inclusive** — Functional, Non-Functional, Constraints, Guidelines

Traceability Ensures Quality







Without traceability products can drift away from what the user requires

"Quality is conformance to requirements.

Verifiability Ensures Quality

- Track the Source and Status of your Requirements with Attributes
  - Record the source (when created, rationale, original text)
  - Record urgency (mandatory, useful, optional, luxury)
  - Record acceptance (proposed, reviewed, accepted, rejected)
  - Identify verification method (test, demonstration, inspection, simulation, analysis)
  - Identify constraints (safety, performance, reliability, corporate standards, business rules)
  - Record discussion (comments, action items, proposed change, state of review process)

Analysis Now Made Simple

Once this wealth of information is captured and well written, the status of a project can be easily determined:

Show all high priority requirements for the pilot on the flight control subsystem:

Priority = High
User = Pilot
Subsystem = Flight Control

Show all the requirements maintenance users proposed that could effect safety and performance:

User = Maintenance
Status = Proposed
Constraints = Safety & Performance

The keys to writing better requirements lead to successful projects!

www.qssinc.com