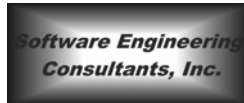


Requirements and Their Characteristics

Presented by
Emanuel R. Baker, Ph.D.
President
Software Engineering Consultants, Inc.
Los Angeles, California
(310) 278-7241
erbaker@swengcon.com



© Copyright 2006 Software Engineering Consultants, Inc.
Los Angeles, California
All Rights Reserved

Establishing the Requirements

Software Engineering
Consultants, Inc.

Topics to be Discussed

- **Importance of Requirements**
- Requirements Sources
- Types of Requirements
- The Requirements Definition Process
- Documenting Requirements
- Managing Requirements Over the Product Life-Cycle

Why Requirements Are Important

Software Engineering
Consultants, Inc.

- **Real measure of software quality: does it do what it is supposed to do?**
- **Defines what will be produced**
- **Greatest impact on software product quality**
 - Requirements errors propagate into redesign, recode, retest
 - Greatest source of errors
- **Good requirements management and development practices ensure a design that implements the real needs**

Req - 3

© 2006 Software Engineering Consultants, Inc.

Impact on Cost and Quality

Software Engineering
Consultants, Inc.

- **Requirements are Greatest Single Source of Errors**
 - Estimates from various researchers range from 25 – 50%
 - Standish Group reports (covering 1994 – 2004) estimate projects that meet budget, schedule, and requirements at an average of less than 20%
 - 2002 best year at about 28%
- **Cost of Requirements Error Fixes Grows Exponentially**
 - 100:1 Production Phase for larger programs (Boehm)
 - 20:1 for smaller programs (Boehm)
- **Requirements Problems Often Remain Undetected Until Validation Testing**
 - Lack of organized requirements definition process
 - Lack of peer reviews, formal reviews
 - Poorly implemented unit tests

Req - 4

© 2006 Software Engineering Consultants, Inc.

Topics to be Discussed

- Importance of Requirements
- Requirements Sources
- Types of Requirements
- The Requirements Definition Process
- Documenting Requirements
- Managing Requirements Over the Product Life-Cycle

Sources of Requirements

Basic Concepts

- Kinds of Requirements
 - Customer/user requirements
 - System level
 - Express required functionality as desired by the user
 - Developer requirements
 - Software-specific
 - Express required functionality from implementation perspective
- Explicit vs. Implicit Requirements
 - Explicit: Directly derived from the system/user requirements
 - Implicit: Not directly derivable, but necessary for implementation
 - Example: Constraints imposed by target computer

Customer/User Needs

Software Engineering
Consultants, Inc.

- **Elicited from People**
 - Driven by user operations
 - Minimal system constraints
 - Task analysis identifies user interface
- **Derived from System**
 - Driven by system interface
 - Highly constrained by environment



Req - 7

© 2006 Software Engineering Consultants, Inc.

Developer Requirements

Software Engineering
Consultants, Inc.

Requirements Engineering

- **Developer Requirements Derived from Customer Requirements¹**
 - Identification
 - Allocation
 - Decomposition – Methods
 - Risk assessment
 - Analysis
- **Traceability Techniques**
 - Tracks implementation of requirements
 - Ensures all are implemented
 - Distinguishes explicit from implicit requirements

¹ See December 2006 Crosstalk for discussion of requirements elicitation/development techniques

Req - 8

© 2006 Software Engineering Consultants, Inc.

Topics to be Discussed

- Importance of Requirements
- Requirements Sources
- **Types of Requirements**
- The Requirements Definition Process
- Documenting Requirements
- Managing Requirements Over the Product Life-Cycle

- **Requirements characteristics**
 - Necessary
 - Complete
 - Understandable/unambiguous
 - Consistent
 - Testable
 - Correct
 - Feasible
 - Traceable
- **Good requirements will exhibit all these characteristics**
- **What constitutes the completeness and testability of a requirement varies based on the type of requirement**

Key Characteristics

Software Engineering
Consultants, Inc.

- **Completeness**
 - Importance is reflected in ability to design
 - Without the required information, requirement cannot normally be implemented correctly
- **Testability**
 - Influences ability to do qualification testing
 - Qualification method is dependent on:
 - Type of requirement
 - Adequacy by which other requirements attributes are specified
- **Correctness**
 - If requirement is not correctly specified, software will not perform in accordance with user needs

Requirements Types

Software Engineering
Consultants, Inc.

- **Functional Requirements**
- **Performance Requirements**
- **Interface Requirements**
- **Data Requirements**
- **Security and Privacy Requirements**
- **Data Integrity Requirements**
- **“Ilities”**

Functional Requirements

Software Engineering
Consultants, Inc.

- Describes functions that the system or software must be capable of performing
- Specifies system behavior in terms of inputs (stimuli), outputs (responses), and functional relationships between them

Functional Requirements

Software Engineering
Consultants, Inc.

Examples

- Upon detection of an off-hook condition, the telephone system shall respond with a dial tone
- If no input is received for a 15-minute period, the application shall terminate
- Upon detection of a heat sensor reading above 104 degrees Fahrenheit, the system shall activate the CRISIS_SERVICE_NEEDED light at the Nurse's Station

Testing Functional Requirements

Software Engineering
Consultants, Inc.

Simple Go/No-Go Tests

- **Upon detection of an off-hook condition, the system shall respond with a dial tone**
 - Take the phone off the hook and wait for a dial tone
 - Repeat several times to ensure consistent behavior
- **If no input is received for a 15-minute period, the application shall terminate**
 - Terminate providing inputs to the software, via any external interface
 - Use a stop watch to time when the application terminates
 - Repeat several times to verify that 15-minute time limit has been met

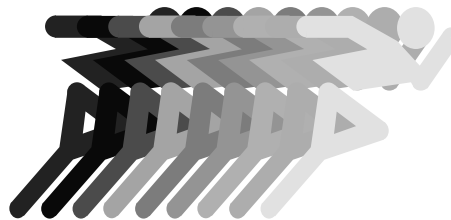
Req - 15

© 2006 Software Engineering Consultants, Inc.

Performance Requirements

Software Engineering
Consultants, Inc.

- **Specifies criteria by which the system must perform its functions with respect to:**
 - Response speed
 - Error-handling capability
 - Accuracy
 - Capacity



Req - 16

© 2006 Software Engineering Consultants, Inc.

Examples

- Upon detection of an off-hook condition, the system shall respond with a dial tone *within 2 ± 0.1 seconds*
- The system shall notify the user whenever an input is entered that is outside the allowable limits and shall not process it
- The trajectory model shall be capable of reconstructing the actual trajectory to *within ± 1 foot in the x, y, and z coordinates*
- The database management system shall be large enough to *contain 150 billion records*

Testing Performance Requirements

- Validation of performance requirements require more complex tests
- Examples:
 - Upon detection of an off-hook condition, the system shall respond with a dial tone *within 2 ± 0.1 seconds*
 - Requires same test as for functional testing, except that time measurement must be added to the test process
 - Must also define how many tests must be performed to validate 0.1 second constraint
 - The system shall notify the user whenever an input is entered that is outside the allowable limits and shall not process it
 - Enter valid and invalid inputs
 - In range
 - At the limits
 - Above and below the limits
 - Look for an error message when invalid entry made
 - Verify that software did not process input

Interface Requirements

Software Engineering
Consultants, Inc.

- **Two Primary Types of Interface Requirements**
 - User Interface Requirements
 - Specifies interfaces between the user and the system
 - System Interface Requirements
 - Specifies interfaces between the application(s) under development and other applications or system software with which it must interact

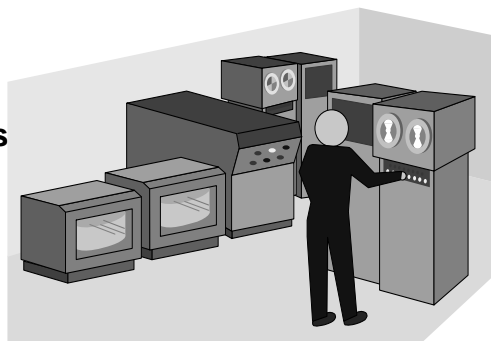
Req - 19

© 2006 Software Engineering Consultants, Inc.

User Interface Requirements

Software Engineering
Consultants, Inc.

- **User Input Devices**
 - Keyboards
 - Mice
 - Bar Code Readers
 - Fingers
 - Sensors
- **User Output Devices**
 - Displays
 - Printers
 - Speakers
 - Sensors



Req - 20

© 2006 Software Engineering Consultants, Inc.

Examples

- **Input**
 - The system shall have a menu subsystem for user access to functions. The user shall not be required to press more than one key in selecting a menu item.
 - The user-input subsystem shall be graphical. The user shall be able to click on an icon to launch an application. Each application shall have a main menu bar at the top of the screen that contains pull-down menus. It shall be...
- **Output**
 - The user shall be able to select report options that will provide either hard copy of the report, or a screen display of the report.
 - The output shall be capable of driving a digital video recorder, using the following formats:

Testing User Interface Requirements

- **Validations of user interface requirements tend to be relatively simple**
- **Examples**
 - The system shall have a menu subsystem for user access to functions. The user shall not be required to press more than one set of key strokes in selecting a menu item.
 - Can be performed by combination of inspection and simple test
 - Look for existence of a menu
 - Look for instructions or symbols that indicate that a combination of several keys activate menu items (e.g., "ALT+D" for data entry menu item)
 - Test each menu item
 - The user shall be able to select report options that will provide either hard copy of the report, or a screen display of the report.
 - Can be performed by inspection

System Interface Requirements

Software Engineering
Consultants, Inc.

Examples

- The application shall be a windows application operating under the Microsoft Windows XP Professional Operating System.
- Battlefield force data will be provided by sensors in the field operated by the XYZ agency. The data shall be transferred via a MIL-STD-1553B data bus. The data to be received and their formats shall be as follows:

Testing System Interface Requirements

Software Engineering
Consultants, Inc.

- **Validating system interface requirements can be very complex**
- **Examples**
 - The application shall be a windows application capable of operating under the Microsoft Windows XP Professional Operating System.
 - Microsoft Windows XP specifies interface requirements for all applications operating under it
 - Deliverable software would have to be tested to verify that it was compatible with all Microsoft Windows XP constraints
 - Battlefield force data will be provided by sensors in the field operated by the XYZ agency. The data shall be provided by a MIL-STD-1553B data bus.
 - Data inputs would have to have a validity test for the source of the data to verify it came from XYZ sensors
 - Tests of the inputs would be required to ensure that protocol and format requirements specified in MIL-STD-1553 were met

Data Requirements

Software Engineering
Consultants, Inc.

Data requirements concern identification and organization of data to be operated on by system.

- **System level (as identified by user in the system/user specification)**
 - Global data
 - Identifies high level data entities used by the system
 - Identified without concern for organization
- **Application level (as described by the developers in the software requirements specification)**
 - Identifies data elements operated on by the software
 - Organizes data by their relationships to each other
 - Normalization not performed at this level (design issue)

Examples of Data Requirements

Software Engineering
Consultants, Inc.

- **Data element identification**
 - **HC_B_ID Number(22) Customer Identification Number**
 - Customer Identification Number is data element identifier and HC_B_ID is associated mnemonic
 - Number(22) indicates the type of data (numeric) and associated field length
- **Data relationships to each other**
 - “The XYZ CSCI shall have the capability to record notes and communications associated with the budget (or any of its revisions) for the project.”
 - “The XYZ CSCI shall associate each project’s subcontract numbers and subcontractor identifiers with the prime contract identification number.”

Testing Data Requirements

Software Engineering
Consultants, Inc.

- **Qualification testing of data requirements often overlooked**
 - Lack of recognition of importance
 - Lack of understanding of how to do it
- **Validation can range from simple to complex**
- **Examples**
 - Data element identification
 - Can be accomplished by inspection of data definitions (simple)
 - Need to know how to navigate through DBMS to find data definitions (may not be so simple)
 - Data relationships
 - Need familiarity with database design techniques (e.g., entity relationship diagrams) to establish whether specified relationships have been correctly designed (not so simple)
 - Need familiarity with DBMS to verify if relationships have been correctly implemented (may not be so simple)

Req - 27

© 2006 Software Engineering Consultants, Inc.

Security and Privacy Requirements

Software Engineering
Consultants, Inc.

- **Specifies requirements related to the security of data and information processed by or maintained by the system**
 - User Identification and Authentication
 - Access control
 - Discretionary
 - Mandatory
 - Information Labelling
 - Object Reuse
 - Audit Mechanisms
 - Encryption

Req - 28

© 2006 Software Engineering Consultants, Inc.

Examples

Software Engineering
Consultants, Inc.

- The system shall require users to identify themselves and present a valid password before obtaining system access
- Users shall be allowed to control sharing of owned objects with other users
- Users shall not be allowed access to information classified above their security level
- The system shall not assign system memory resources to a user until said memory has been cleared of all contained data
- The system shall be able to create, maintain, and protect from modification or unauthorised access or destruction, an audit trail of accesses to protected objects

Testing Security and Privacy Requirements

Software Engineering
Consultants, Inc.

- Qualification testing of secure or classified databases may often involve very complex tests
- Examples
 - Users shall not be allowed access to information classified above their security level
 - These tests are very complex to ensure that unauthorized access has been prevented
 - Testing requirements specified in DoD security manual
 - The system shall not assign system memory resources to a user until said memory has been cleared of all contained data
 - Involves multiple overwrites of memory locations
 - Testing requirements specified in DoD security manual

Data Integrity Requirements

Software Engineering
Consultants, Inc.

- **Specifies Requirements on the Correctness, Authenticity, Accuracy and Precision of Data**
 - Prevent authorized users from entering improper information
 - Prevent authorized users from making improper modifications
 - Prevent unauthorized users from modifying data
 - Maintain internal and external data consistency

Examples

Software Engineering
Consultants, Inc.

- **Users shall only be able to enter a valid state abbreviation**
- **When a user requests a change to the customer account balance, the system shall inform the user's superior of the request and take no action until the user's superior authorizes the change to the customer's account balance**
- **The system shall maintain a list of users, together with a list of system functions each user is allowed to perform. The system shall not allow a user to perform a function unless the function is on the user's allowed list.**

Testing Data Integrity Requirements

Software Engineering
Consultants, Inc.

- **Validation of the implementation of these requirements is similar to performance requirements**
- **Example**
 - Users shall only be able to enter a valid state abbreviation
 - Would be performed by entering valid and invalid abbreviations
 - Non-alphabetic characters should also be entered
 - Only valid abbreviations should be accepted by the software

“Ility” Requirements

Software Engineering
Consultants, Inc.

- **Requirements that Characterize the Service Needs of the System**
- **“Ilities” Include:**
 - **Availability**
 - Relates to permissible down-time for the system
 - Example: The Credit Card Transaction System shall be available 99.99999% of the time
 - **Reliability**
 - Probability of failure-free operation of a software system for a specified time in a specified environment
 - Relates to how dependable the software is, i.e. are there designed in errors that will impact system availability
 - Example: The software shall be capable of operating for 1000 hours without encountering a software failure
 - **Maintainability**
 - Relates to how difficult it is to correct, modify, or change software (includes facilities, tools, equipment, personnel, documentation, etc).
 - Example: A maintainer shall be capable of locating and correcting a fault in $2 \pm 1/2$ hours, 95% of the time.

Testing “Ility” Requirements

Software Engineering
Consultants, Inc.

- **Validation of “ility” requirements involve long and complicated tests**
- **Examples**
 - **Software reliability**
 - **Require many hours of tests**
 - **Require multiple input conditions**
 - **Involve measurement of time duration between failures**
 - **Software maintainability**
 - **Would require many hours of tests**
 - **Could require seeding the software with many different kinds of bugs**
 - **Some easy to find and fix**
 - **Others difficult to find and fix**
 - **Maintainability rarely specified in quantitative terms**
 - **Rarely tested**

Establishing the Requirements

Software Engineering
Consultants, Inc.

Topics to be Discussed

- **Importance of Requirements**
- **Requirements Sources**
- **Types of Requirements**
- **The Requirements Definition Process**
- **Documenting Requirements**
- **Managing Requirements Over the Product Life-Cycle**

Many Conflicting Factors Need to be Balanced



System Definition Phase - 1

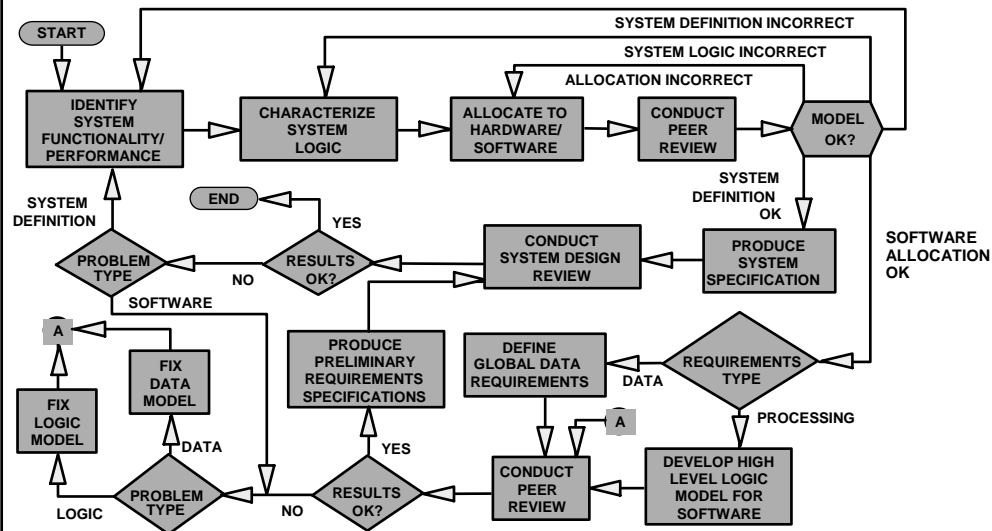
- **User Activities**
 - Identify overall system functionality
 - Identify overall system performance requirements
 - Produce system specification

- **Development Organization Activities**
 - Characterize system logic
 - Allocate between hardware and software (where applicable)
 - Define global data requirements
 - Develop high level logic model for software
 - Produce preliminary processing requirements specifications
 - Produce software development plans

System Definition Phase - 2

- **Techniques**
 - Structured analysis
 - JAD sessions
 - Object-oriented requirements analysis
 - Hierarchical decomposition
 - Functional flow analysis
 - Prototyping
 - Trade-off studies
 - Simulations
- **Reviews**
 - Peer
 - System review (formal)

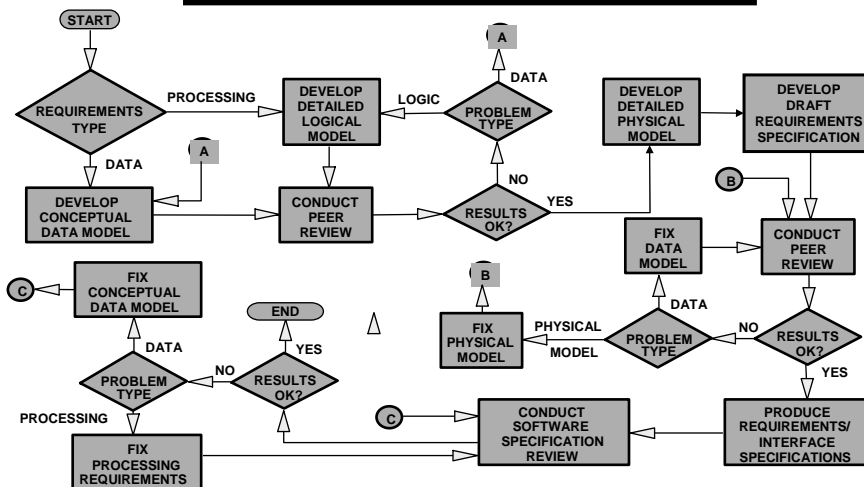
System Definition Phase - 3



Software Requirements Analysis Phase - 1

- **Activities**
 - Develop conceptual data model
 - Develop detailed logical model
 - Develop detailed physical model
 - Produce "final" processing requirements specifications
 - Software Requirements Specification (SRS)
 - Interface Requirements Specification (IRS)
- **Techniques**
 - Use cases
 - Structured analysis
 - Object-oriented requirements analysis
 - Hierarchical decomposition
 - Trade-off studies
 - Simulations
- **Reviews**
 - Peer
 - Software Specification Review

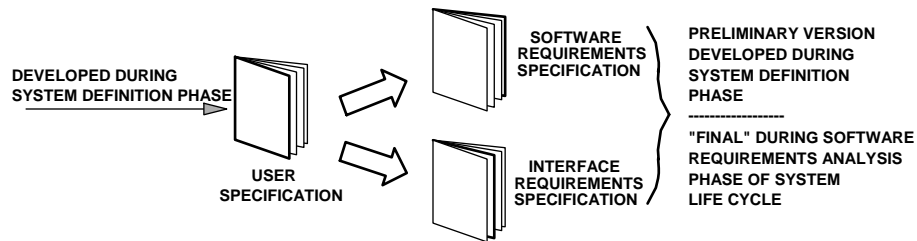
Software Requirements Analysis Phase - 2



Topics to be Discussed

- Importance of Requirements
- Requirements Sources
- Types of Requirements
- The Requirements Definition Process
- Documenting Requirements
- Managing Requirements Over the Product Life-Cycle

Requirements Documents - 1



| REQUIREMENTS DOCUMENT | FOCUS | CONTENT REQUIREMENTS |
|--|-------------|---|
| SYSTEM SPECIFICATION | USER | <ul style="list-style-type: none"> • SPECIFIES SYSTEM FUNCTIONALITY AND PERFORMANCE WITHOUT REGARD TO SOFTWARE • DEFINES DATA TO BE USED FOR SYSTEM • IDENTIFIES INTERFACING SYSTEMS AND ASSOCIATED DATA |
| SOFTWARE REQUIREMENTS SPECIFICATION (SRS) | USER/DEVEL. | <ul style="list-style-type: none"> • SPECIFIES SOFTWARE FUNCTIONALITY AND PERFORMANCE REQUIREMENTS • SPECIFIES ALL OTHER TYPES OF REQUIREMENTS (e.g., PRIVACY, DATA SECURITY) • DEFINES CONCEPTUAL DATA MODEL • IDENTIFIES DESIGN CONSTRAINTS • SPECIFIES USER INTERFACE |
| INTERFACE REQUIREMENTS SPECIFICATION (IRS) | USER/DEVEL. | <ul style="list-style-type: none"> • SPECIFIES INTERFACING DATA ELEMENTS AND CHARACTERISTICS (SYSTEM INTERFACES) • FOR SIMPLE INTERFACES, IRS MAY BE COMBINED WITH SRS |

Requirements Documents - 2

Software Engineering
Consultants, Inc.

- **System Specification Content Areas**
 - Description of interfacing systems
 - Definition of system data to be used
 - Functional/performance requirements for system as a whole
 - “Ilities” for the system as a whole
 - Qualification requirements for system
- **Software Requirements Specification Content Areas**
 - External interface requirements*
 - Software functional requirements
 - Software performance requirements
 - Design constraints
 - Conceptual data model description
 - Installation/site/user-unique data
 - “Ilities” allocated to the software
 - Qualification requirements

* ONLY IF NO SEPARATE IRS IS PROVIDED -- USE IRS REQUIREMENTS FOR CONTENT OF THIS SECTION

Requirements Documents - 3

Software Engineering
Consultants, Inc.

- **Interface Requirements Specification Content Areas**
 - External interface descriptions (can be diagrams)
 - Interface requirements (e.g., protocols, execution concurrency, if applicable)
 - Interface data descriptions

Topics to be Discussed

- Importance of Requirements
- Requirements Sources
- Types of Requirements
- The Requirements Definition Process
- Documenting Requirements
- Managing Requirements Over the Product Life-Cycle

Managing Requirements Risk

- **Surprise-driven ---> "Risk/Analysis-driven"**
 - Requirements Management is response to history of software project surprises
- **Simplified definition**
 - Requirements Management is collection of management techniques to focus resources on critical success factors
 - Early identification and quantification of these factors is major emphasis
- **Benefits focus on Requirements phase**
 - Early identification and prioritization expands viable options and minimizes corrective costs
 - Management control is enhanced by focusing review and reporting process on early "actionable" issues

Source: Boehm

Prioritizing

- **Relative Necessity**
 - Importance (Essential, Desirable, Optional)
 - Basis for prioritizing
- **Relative Stability -- Stable or Volatile**
 - Estimate of continued volatility
 - Also used for prioritizing
- **Risk**
 - May define development strategy
 - Higher risk requirements candidates for deferral to later releases

Baselining

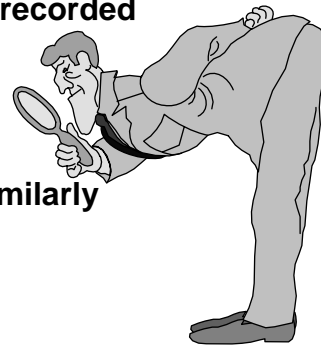
- **Foundation for Requirements Management**
- **Tradeoff - individual efficiency vs. system volatility**
- **Baselines**
 - Initial baseline coincides with first agreement on user requirements as stated in User Specification
 - Next baseline coincides with first agreement on software requirements as stated in Software Requirements Specification (and Interface Requirements Specification, if applicable)
- **Change management must support (not hinder) evolving requirements**
 - Effective control with ease of operation
 - Continuous change must be accommodated
 - Approved changes must include a tracking method
- **Requirements Management is responsible for baseline integrity**

Avoiding Requirements Creep

- **Need well-established initial baseline**
 - Change-of-scope starts with knowing the initial scope
- **Implement defined requirements management methods**
 - Explicit process is best protection against uncontrolled changes
 - Impact analysis of proposed changes provides concrete assessment of change
 - Definitive links between changes & specific impact
- **Orderly basis for user accommodation**
 - Budget increase
 - Schedule changes -- Not always longer
 - Requirements adjustment

Traceability

- **Origins of every requirement must be recorded**
 - Original user needs
 - Developer requirements
 - Derived
 - Implicit
- **Maintenance/updated requirements similarly**
 - Traceable to approved change initiation record
- **Tracing to implementation - matrix**



Requirements Verification

Absolutely essential to verify requirements before implementation in design/code

- **Requirements on individual requirements**
 - Correct
 - Unambiguous
 - Verifiable
- **Requirements on the set of requirements**
 - Feasible
 - Complete
 - Consistent
 - Design Independent
 - Understandable by Customers

Summary

- **Requirements are the driver for a successful product development**
 - Major source of errors
 - Major source of budget/schedule problems
- **Specifying requirements correctly involves:**
 - Knowing the characteristics of good requirements
 - Knowing their types
 - Specifying their attributes accordingly
- **Important to have an effective requirements definition process**
- **Important to document them**
- **Important to manage them over the product life cycle**