Eötvös Loránd Tudományegyetem
Informatikai Kar

Requirements Engineering

Attila Kovács, egyetemi tanár
attila.kovacs@inf.elte.hu

Contents

- Fundamentals of RE – challenges of IT projects, Cobb’s paradox, BA and RE, requirements classification, levels and types, conflicts between requirements, ISO models for quality, Furps, Kano models, glossary for RE, core activities in RE
- RE in different development models, Traditional vs Agile RE,
- Main requirement activities: inception, elicitation, analysis and negotiation, documentation, validation and RE management
- Tools for RE, Traceability, Baseline, Change Management
- Specific documentation technique: sentence templates

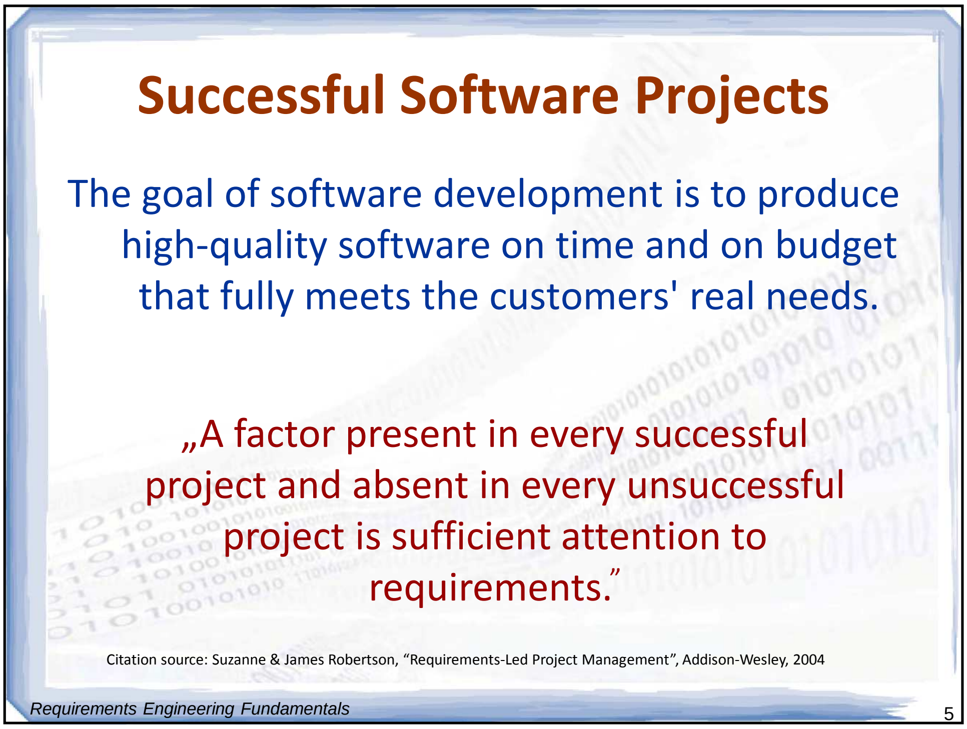
Basic Concepts and Terminology

3

Enterprise Challenges

- Speed
  - Need to change rapidly
  - Can handle change quickly
- Productivity
  - Key to profitability
- Knowledge and its management
  - Market knowledge
  - Customer knowledge (internal and external)
  - Product creation
    - Competitive understanding
    - Organizational capabilities
    - Creating value
  - Growing, Managing and Retaining Knowledge
    - Generations „X, Y, Z” dilemma

4



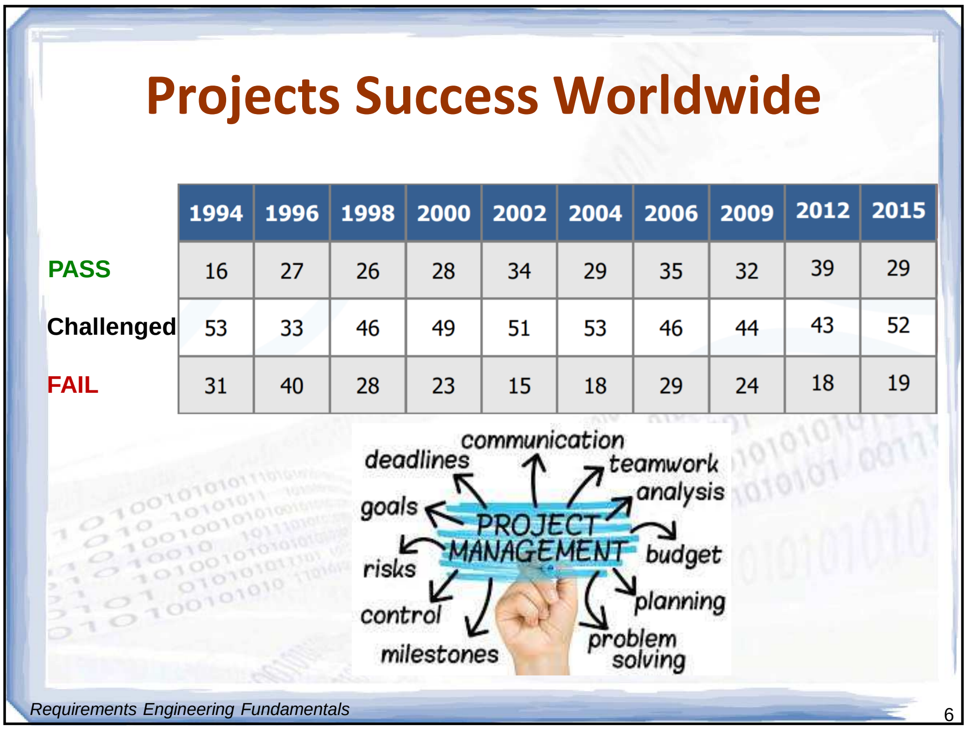
Successful Software Projects

The goal of software development is to produce high-quality software on time and on budget that fully meets the customers' real needs.

„A factor present in every successful project and absent in every unsuccessful project is sufficient attention to requirements.”

Citation source: Suzanne & James Robertson, “Requirements-Led Project Management”, Addison-Wesley, 2004

5

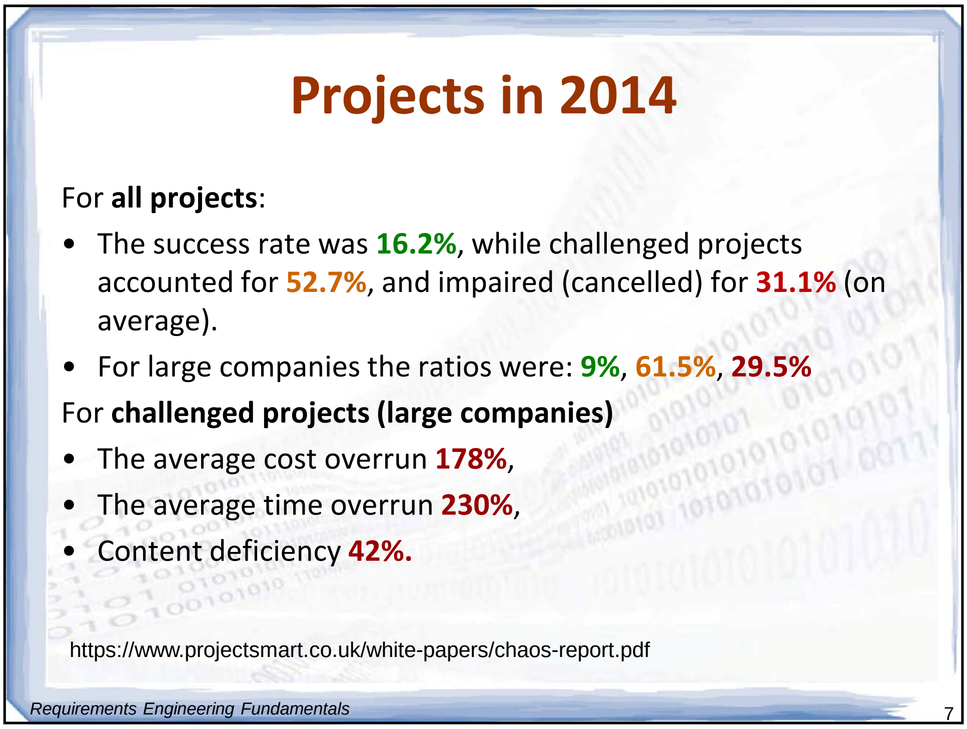


Projects Success Worldwide

	1994	1996	1998	2000	2002	2004	2006	2009	2012	2015
PASS	16	27	26	28	34	29	35	32	39	29
Challenged	53	33	46	49	51	53	46	44	43	52
FAIL	31	40	28	23	15	18	29	24	18	19

communication
deadlines
teamwork
analysis
goals
PROJECT MANAGEMENT
budget
risks
planning
control
problem solving
milestones

6



Projects in 2014

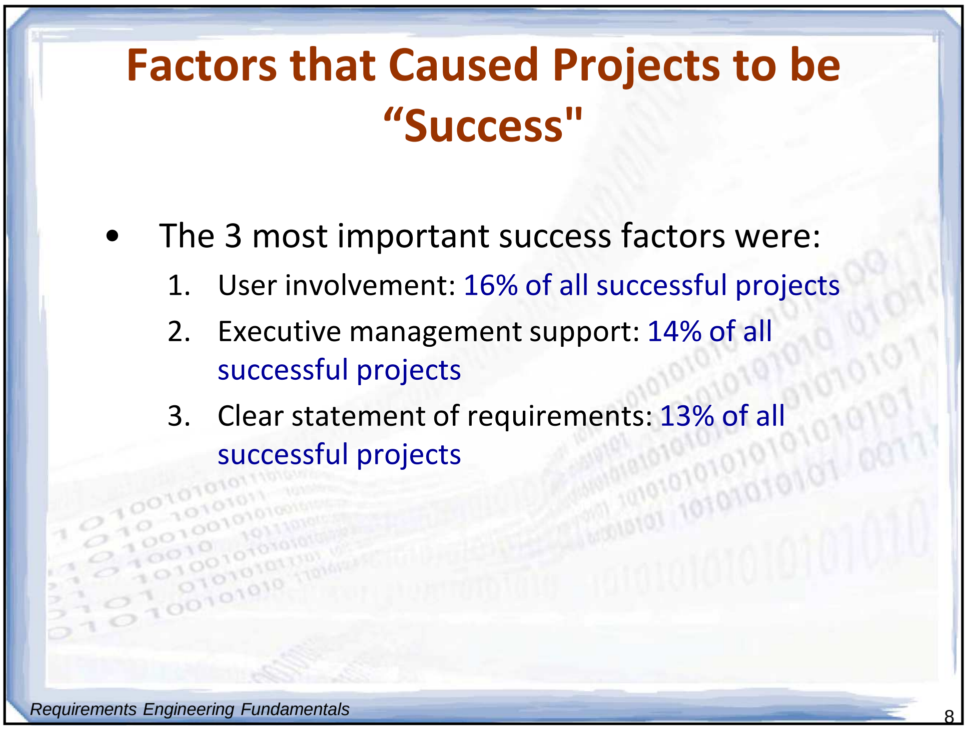
For **all projects**:

- The success rate was **16.2%**, while challenged projects accounted for **52.7%**, and impaired (cancelled) for **31.1%** (on average).
- For large companies the ratios were: **9%**, **61.5%**, **29.5%**

For **challenged projects (large companies)**

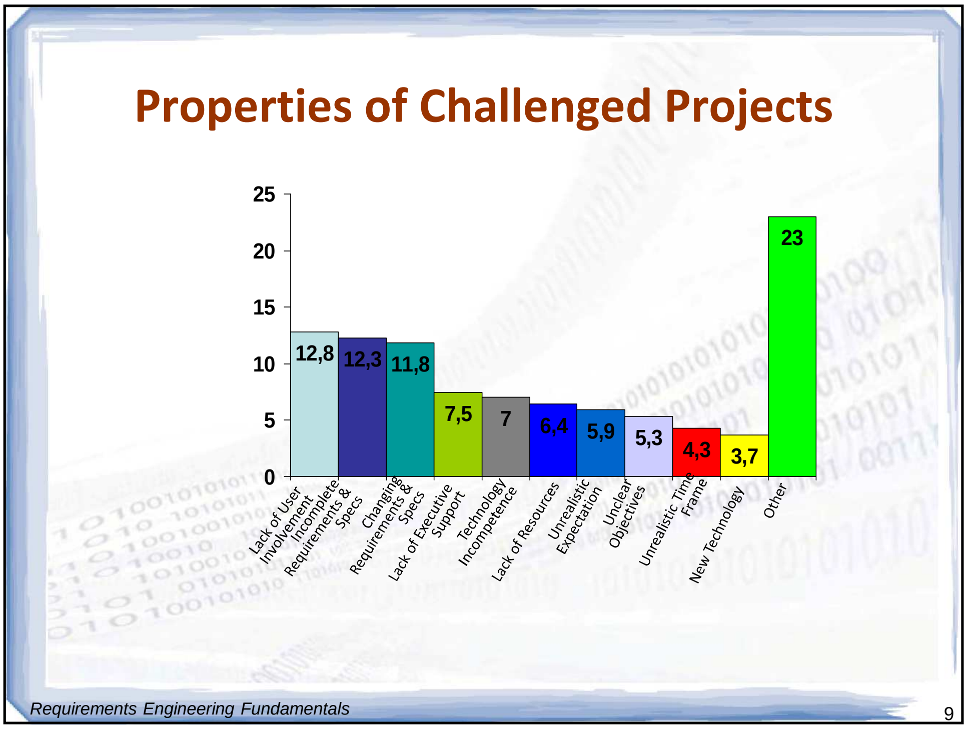
- The average cost overrun **178%**,
- The average time overrun **230%**,
- Content deficiency **42%.**

https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

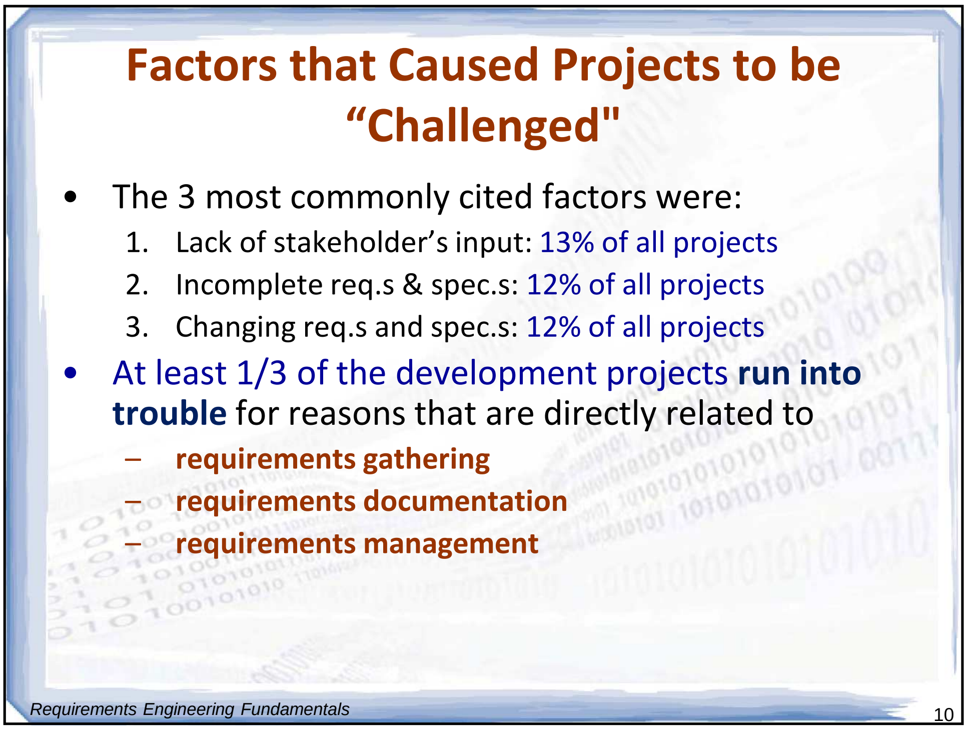


Factors that Caused Projects to be "Success"

- The 3 most important success factors were:
  1. User involvement: 16% of all successful projects
  2. Executive management support: 14% of all successful projects
  3. Clear statement of requirements: 13% of all successful projects

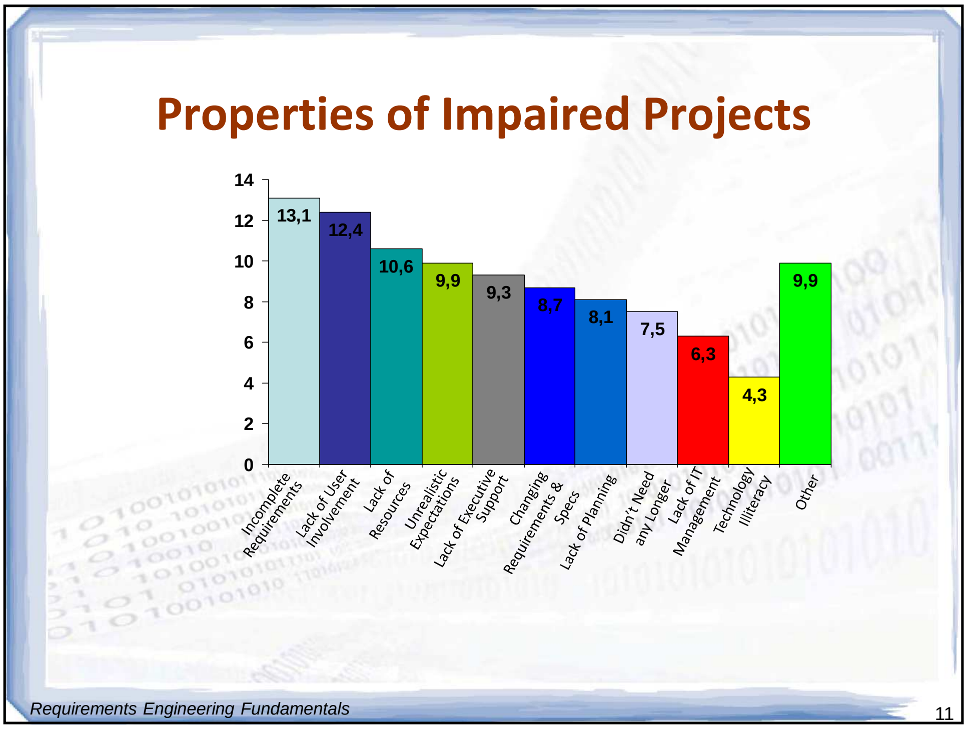


Properties of Challenged Projects

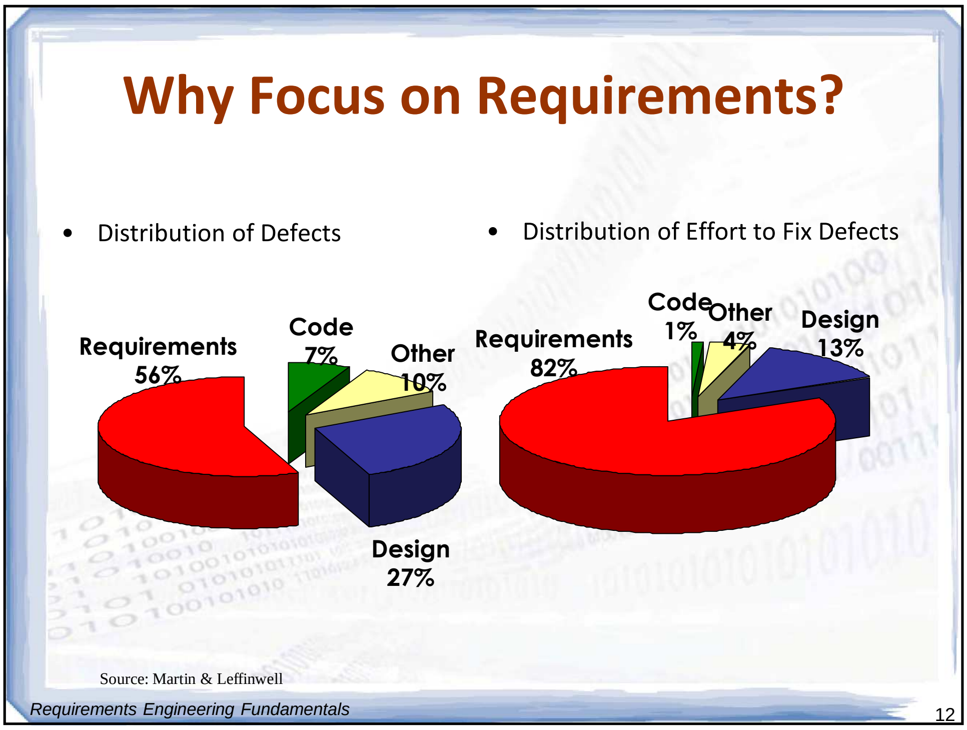


Factors that Caused Projects to be “Challenged"

- The 3 most commonly cited factors were:
  1. Lack of stakeholder’s input: 13% of all projects
  2. Incomplete req.s & spec.s: 12% of all projects
  3. Changing req.s and spec.s: 12% of all projects
- At least 1/3 of the development projects **run into trouble** for reasons that are directly related to
  - **requirements gathering**
  - **requirements documentation**
  - **requirements management**



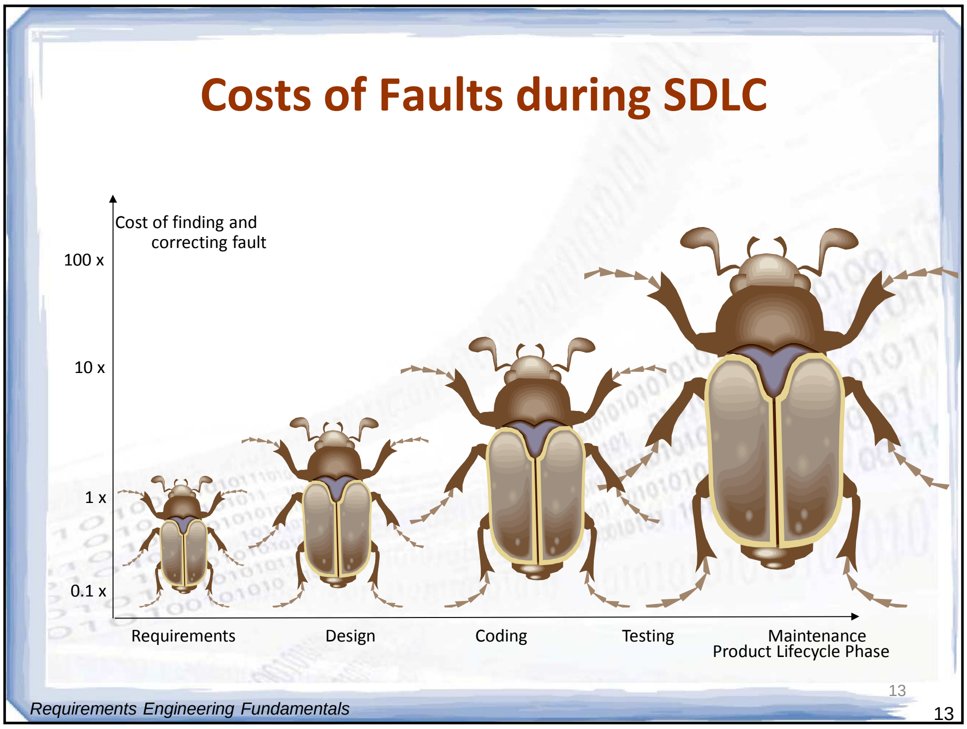
Properties of Impaired Projects



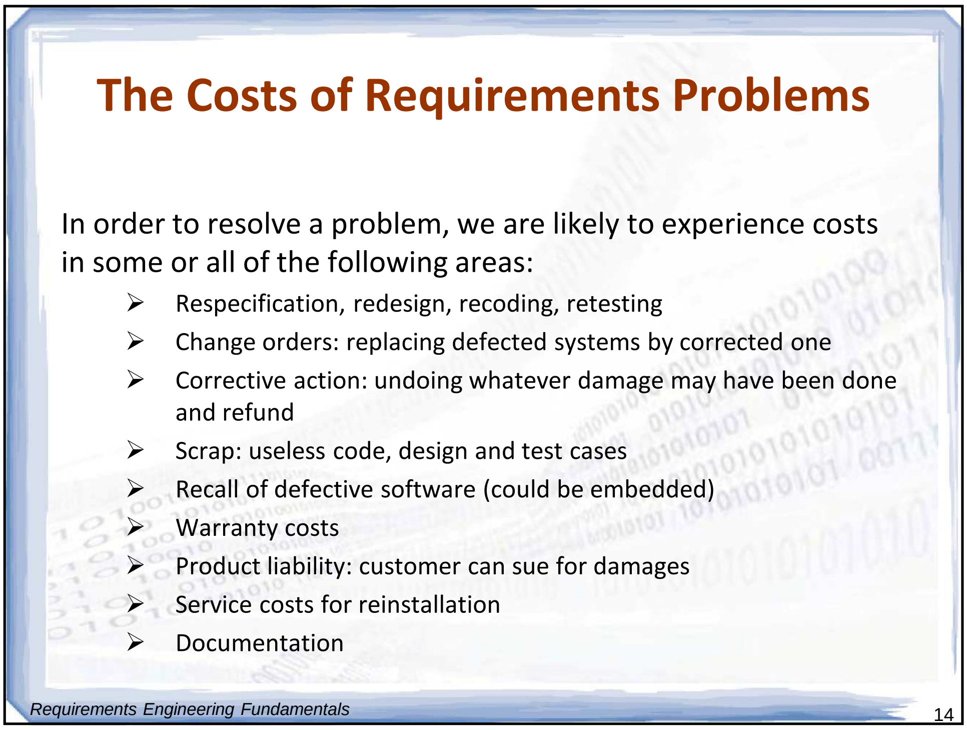
Why Focus on Requirements?

- Distribution of Defects
- Distribution of Effort to Fix Defects

Source: Martin & Leffinwell



Costs of Faults during SDLC

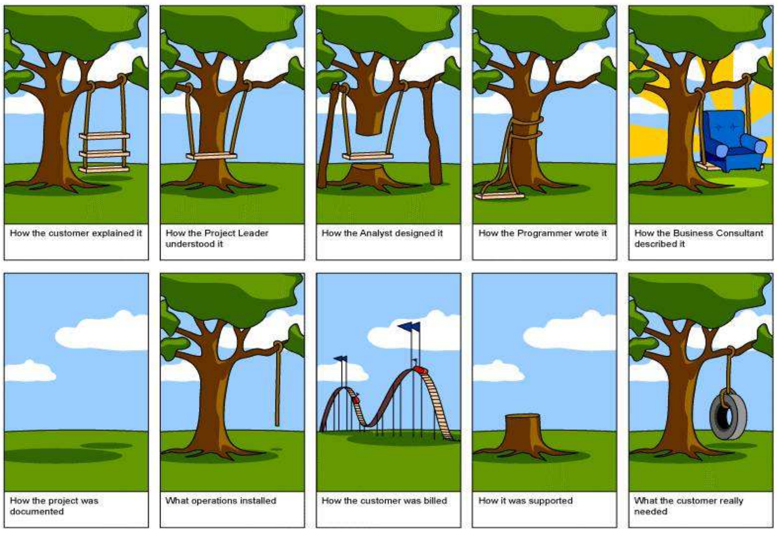


The Costs of Requirements Problems

In order to resolve a problem, we are likely to experience costs in some or all of the following areas:

- Respecification, redesign, recoding, retesting
- Change orders: replacing defected systems by corrected one
- Corrective action: undoing whatever damage may have been done and refund
- Scrap: useless code, design and test cases
- Recall of defective software (could be embedded)
- Warranty costs
- Product liability: customer can sue for damages
- Service costs for reinstallation
- Documentation

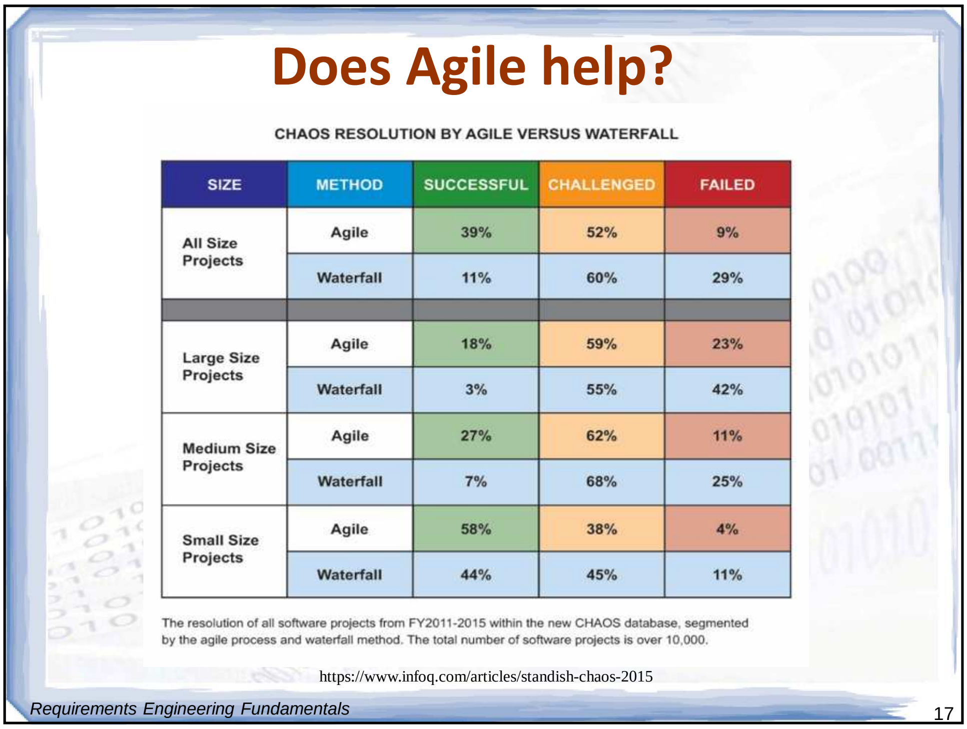
To Build the „Right System”…


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Cobb’s Paradox

„We know why projects fail, we know how to prevent their failure - so why do they still fail?”

Martin Cobb, 1995.



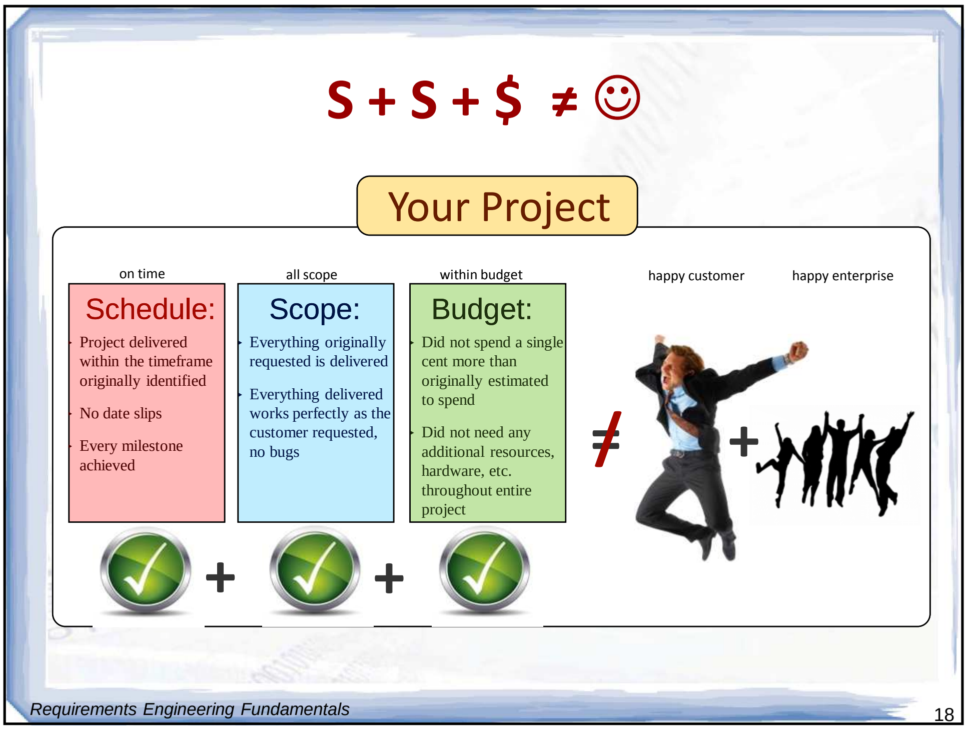
Does Agile help?

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011-2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

https://www.infoq.com/articles/standish-chaos-2015



S + S + $ ≠ ☺

Your Project

on time

Schedule:
- Project delivered within the timeframe originally identified
- No date slips
- Every milestone achieved

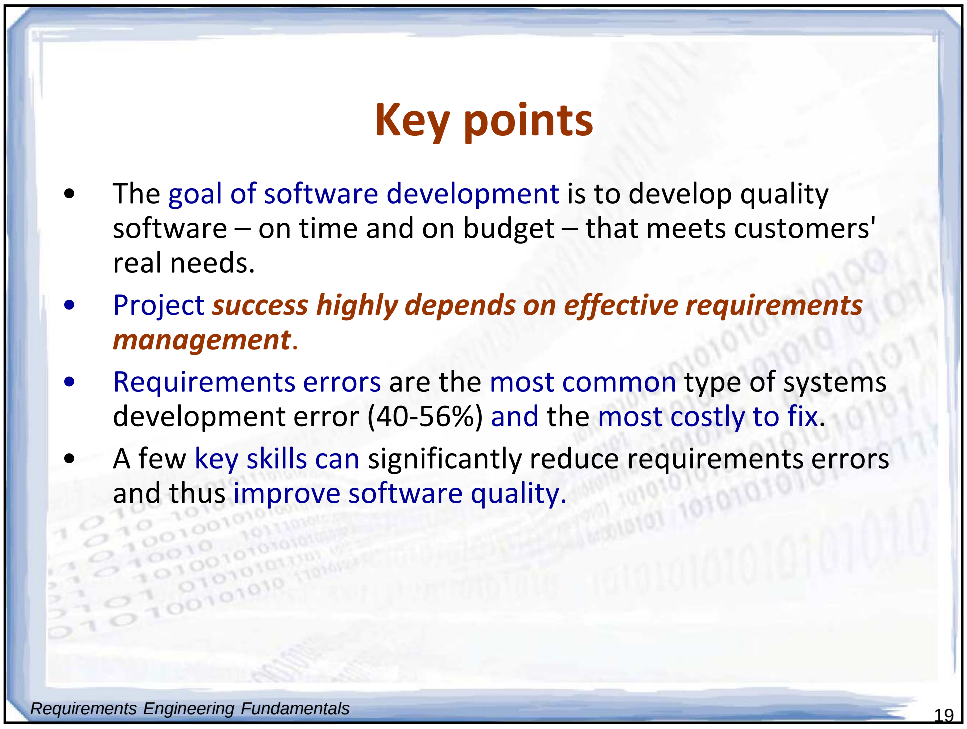
all scope

Scope:
- Everything originally requested is delivered
- Everything delivered works perfectly as the customer requested, no bugs

within budget

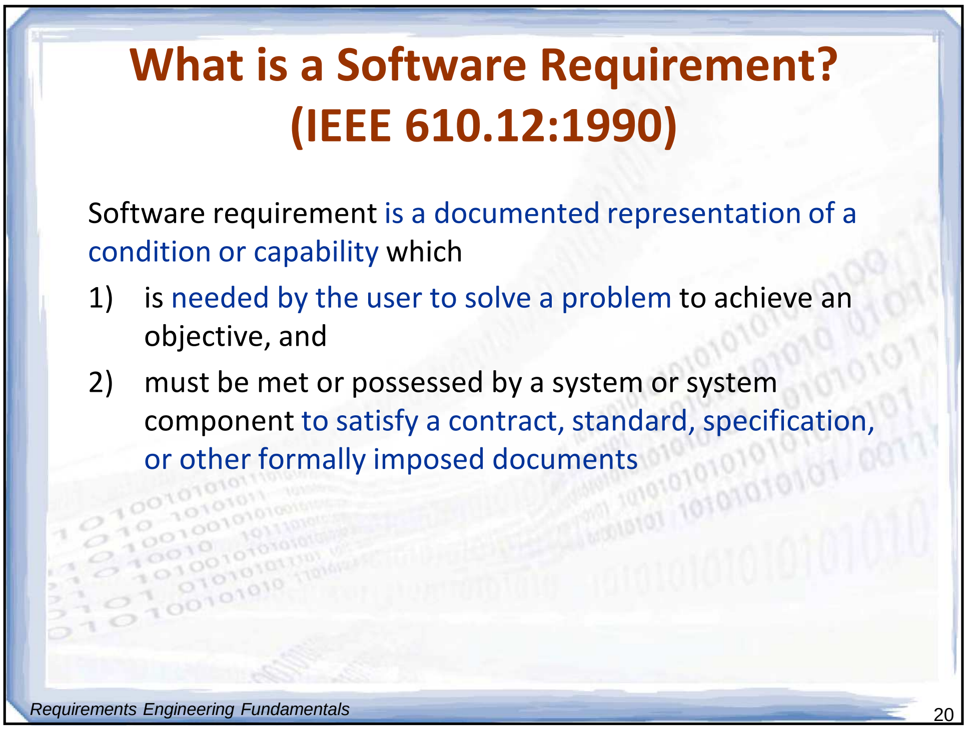
Budget:
- Did not spend a single cent more than originally estimated to spend
- Did not need any additional resources, hardware, etc. throughout entire project

✓ + ✓ + ✓ ≠ happy customer + happy enterprise



Key points

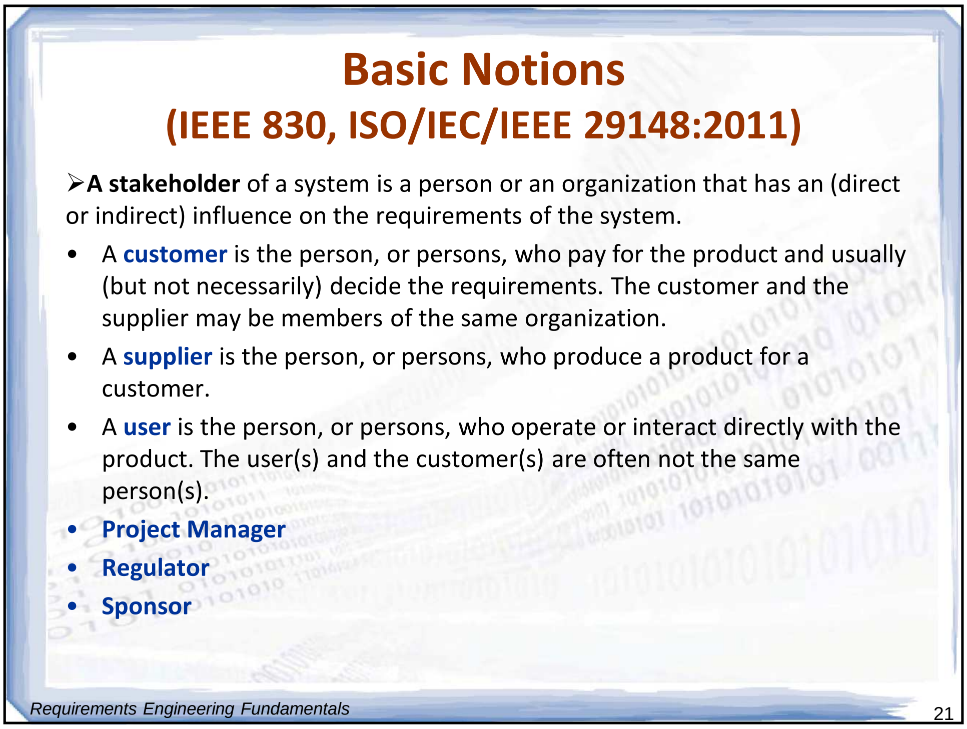
- The goal of software development is to develop quality software – on time and on budget – that meets customers' real needs.
- Project ***success highly depends on effective requirements management***.
- Requirements errors are the most common type of systems development error (40-56%) and the most costly to fix.
- A few key skills can significantly reduce requirements errors and thus improve software quality.



What is a Software Requirement? (IEEE 610.12:1990)

Software requirement is a documented representation of a condition or capability which

1) is needed by the user to solve a problem to achieve an objective, and
2) must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents

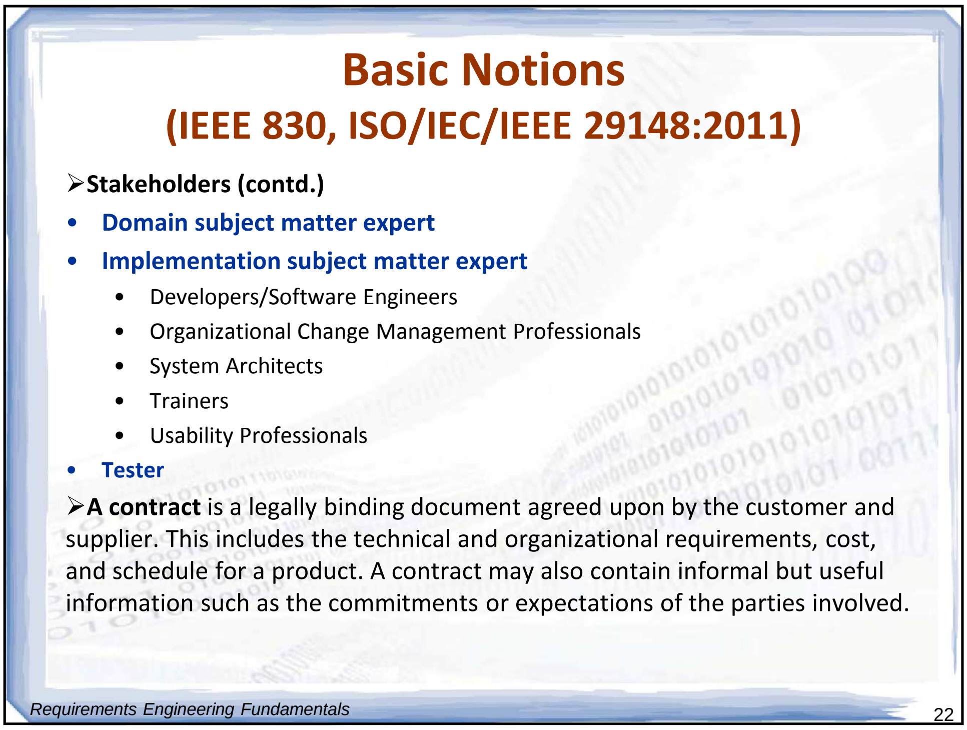


Basic Notions

(IEEE 830, ISO/IEC/IEEE 29148:2011)

➢**A stakeholder** of a system is a person or an organization that has an (direct or indirect) influence on the requirements of the system.

- A **customer** is the person, or persons, who pay for the product and usually (but not necessarily) decide the requirements. The customer and the supplier may be members of the same organization.
- A **supplier** is the person, or persons, who produce a product for a customer.
- A **user** is the person, or persons, who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).
- **Project Manager**
- **Regulator**
- **Sponsor**



Basic Notions

(IEEE 830, ISO/IEC/IEEE 29148:2011)

➢**Stakeholders (contd.)**

- **Domain subject matter expert**
- **Implementation subject matter expert**
  - Developers/Software Engineers
  - Organizational Change Management Professionals
  - System Architects
  - Trainers
  - Usability Professionals
- **Tester**

➢**A contract** is a legally binding document agreed upon by the customer and supplier. This includes the technical and organizational requirements, cost, and schedule for a product. A contract may also contain informal but useful information such as the commitments or expectations of the parties involved.

What is Requirements Engineering?

Requirements Engineering (RE) is a systematic and disciplined approach to the **specification and management of requirements** of a software system with the following goals:

- *knowing the relevant requirements, achieving a consensus among the stakeholders, documenting them according to given standards and managing them systematically*
- *understanding and documenting the stakeholder's desires, specifying and managing requirements to miminize risks of delivering erroneous systems*

Compared to wikipedia:

Requirements engineering refers to the process of defining, documenting and maintaining requirements to the sub-fields of systems engineering and software engineering.

The Requirements Engineer

- **Business analyst (BA)** is someone who **analyzes** an organization or **business domain** and documents its business or processes or systems, assessing the business model or its integration with technology. The BA may also support the development of training material, participates in the implementation, and provides post-implementation support**. BA may overlap into roles such as project manager or consultant**. BA does not always work in IT-related projects, as BA skills are often required in **marketing and financial roles** as well.
- Those Business Analysts who work solely on developing software systems are called IT Business Analysts or **Technical Business Analysts** or **Requirements Engineer**.

Requirements Classification Scheme

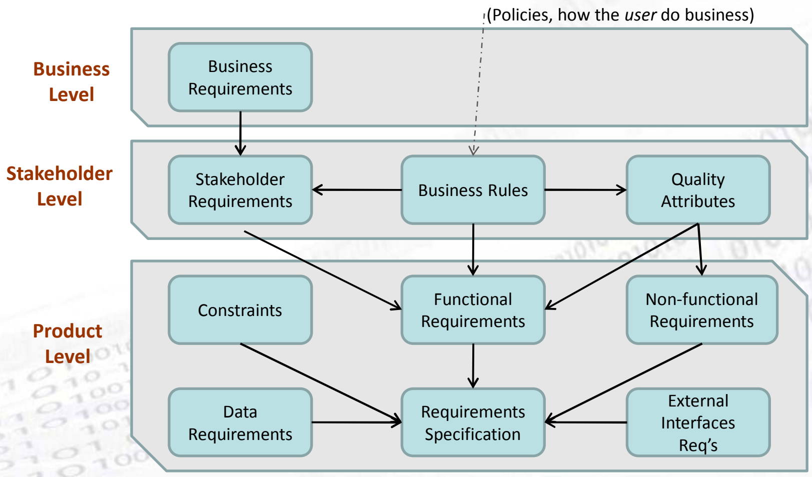
Business requirements are higher-level statements of the goals, objectives or needs of the enterprise.

Stakeholder requirements are statements of the needs of a particular stakeholder or class of stakeholders. They describe the needs that a given stakeholder has and how the stakeholder will interact with a solution.

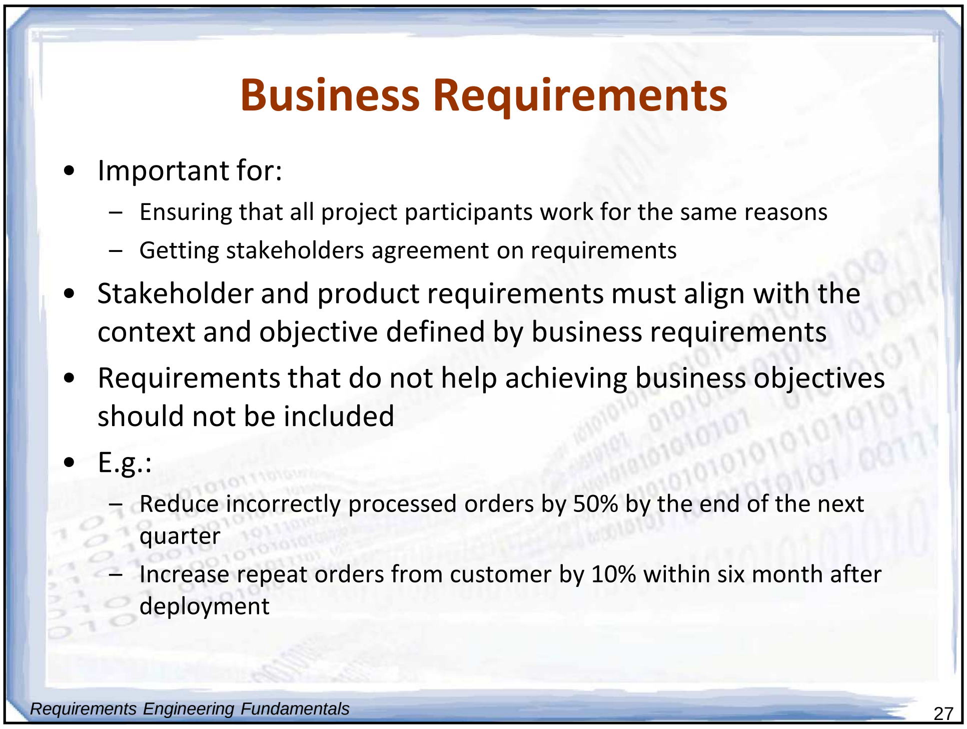
Product Requirements describe the characterizes of specific neeeds that meet business requirements and stakeholder requirements.

Transition Requirements describe capabilities that the solution must have in order to facilitate transition from current state of the enterprise to a desired future state, but that will not be needed once the transition is complete.

Levels and Types of Requirements

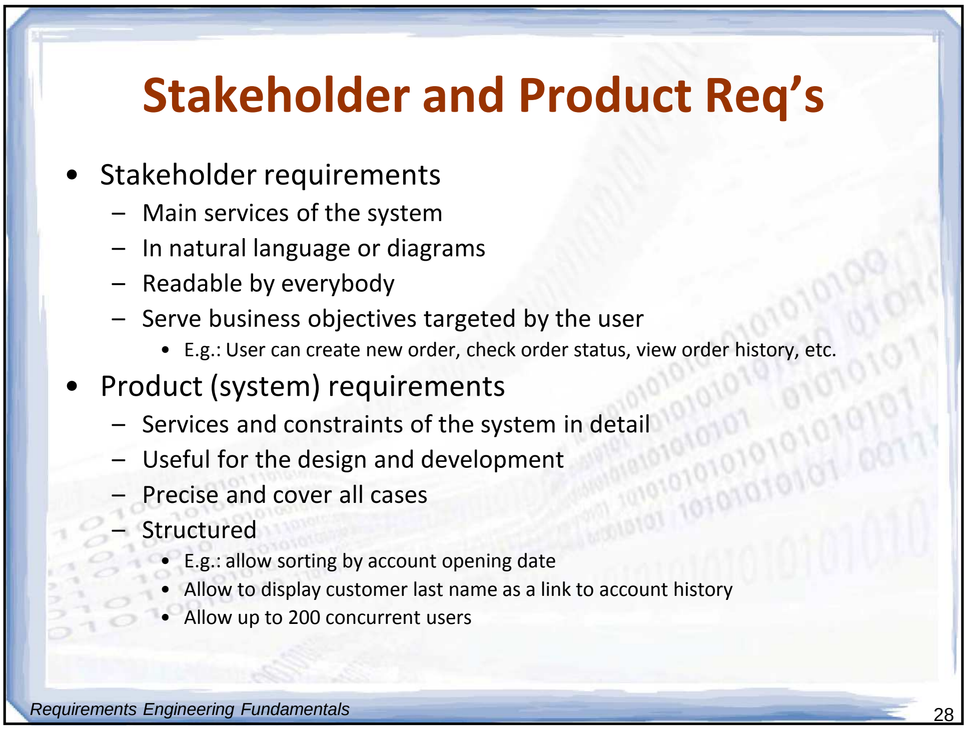


Origin: Karl Wiegers, Software Requirements, 2004



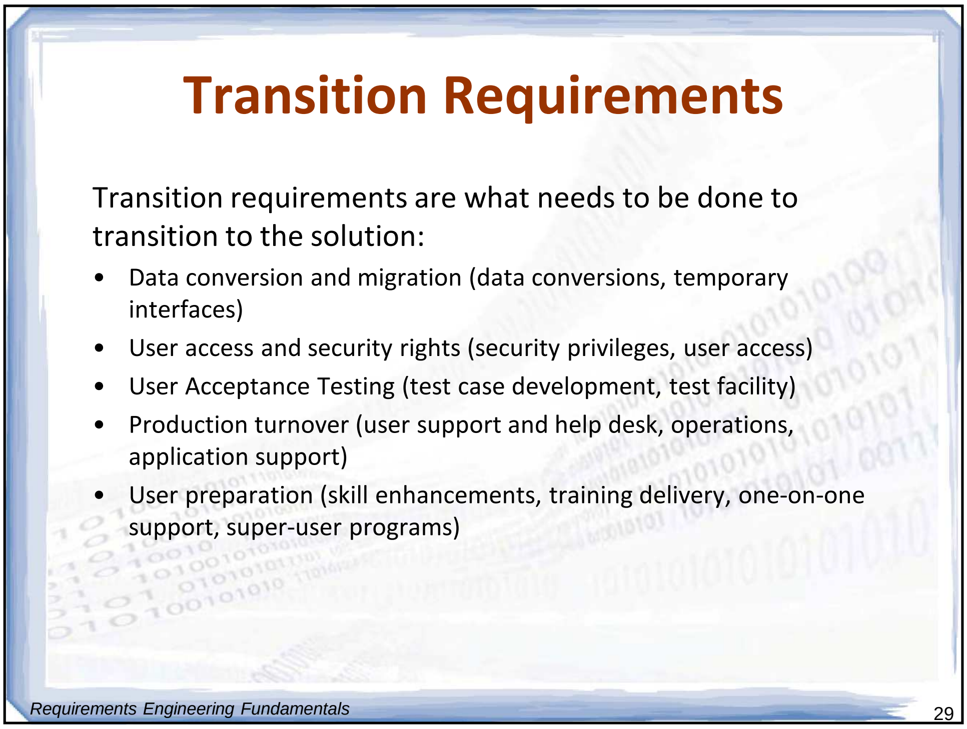
Business Requirements

- Important for:
  - Ensuring that all project participants work for the same reasons
  - Getting stakeholders agreement on requirements
- Stakeholder and product requirements must align with the context and objective defined by business requirements
- Requirements that do not help achieving business objectives should not be included
- E.g.:
  - Reduce incorrectly processed orders by 50% by the end of the next quarter
  - Increase repeat orders from customer by 10% within six month after deployment



Stakeholder and Product Req’s

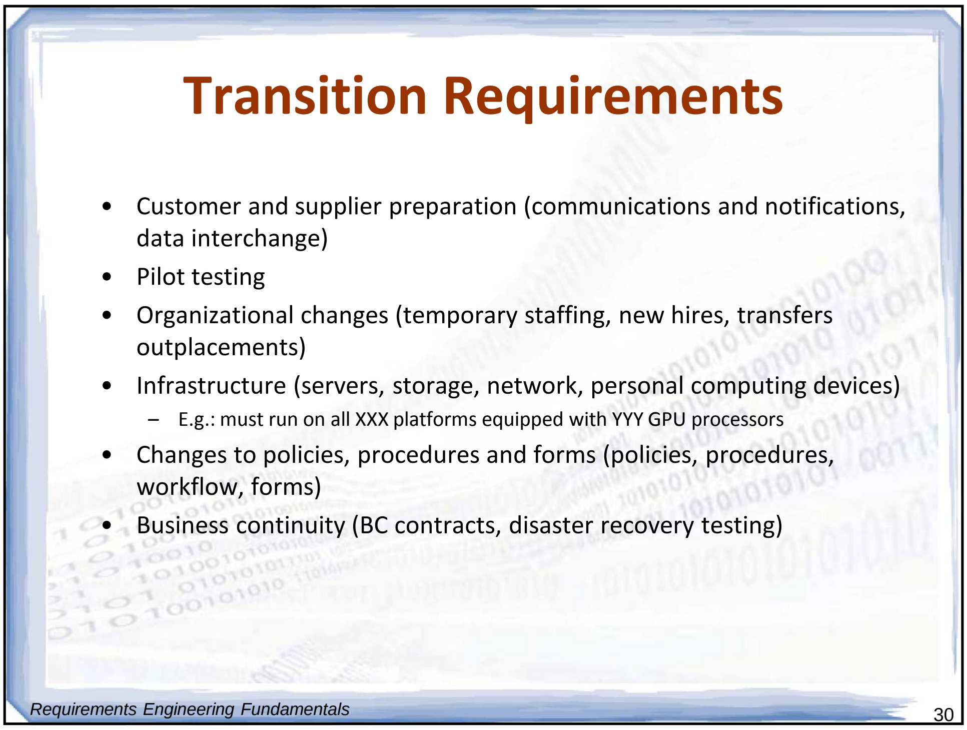
- Stakeholder requirements
  - Main services of the system
  - In natural language or diagrams
  - Readable by everybody
  - Serve business objectives targeted by the user
    - E.g.: User can create new order, check order status, view order history, etc.
- Product (system) requirements
  - Services and constraints of the system in detail
  - Useful for the design and development
  - Precise and cover all cases
  - Structured
    - E.g.: allow sorting by account opening date
    - Allow to display customer last name as a link to account history
    - Allow up to 200 concurrent users



Transition Requirements

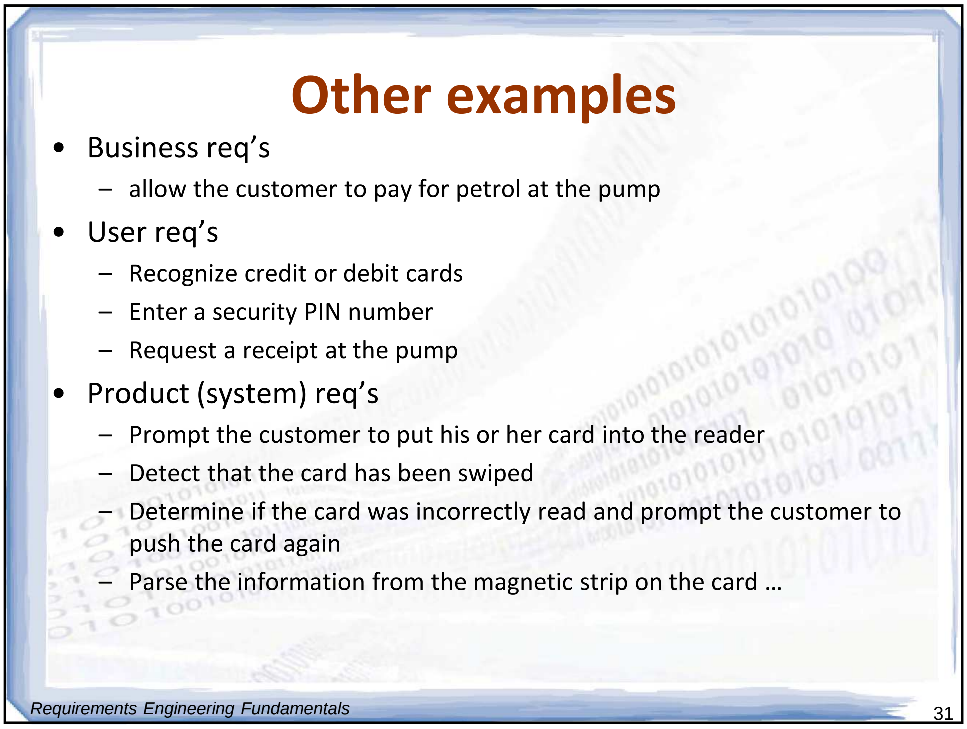
Transition requirements are what needs to be done to transition to the solution:

- Data conversion and migration (data conversions, temporary interfaces)
- User access and security rights (security privileges, user access)
- User Acceptance Testing (test case development, test facility)
- Production turnover (user support and help desk, operations, application support)
- User preparation (skill enhancements, training delivery, one-on-one support, super-user programs)



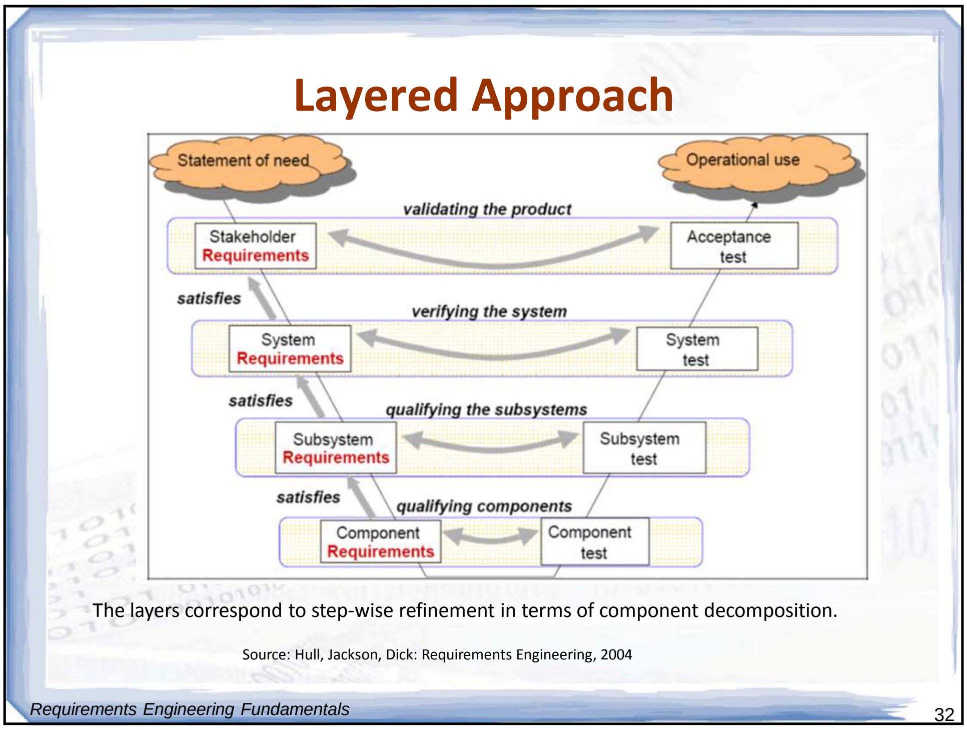
Transition Requirements

- Customer and supplier preparation (communications and notifications, data interchange)
- Pilot testing
- Organizational changes (temporary staffing, new hires, transfers outplacements)
- Infrastructure (servers, storage, network, personal computing devices)
  - E.g.: must run on all XXX platforms equipped with YYY GPU processors
- Changes to policies, procedures and forms (policies, procedures, workflow, forms)
- Business continuity (BC contracts, disaster recovery testing)



Other examples

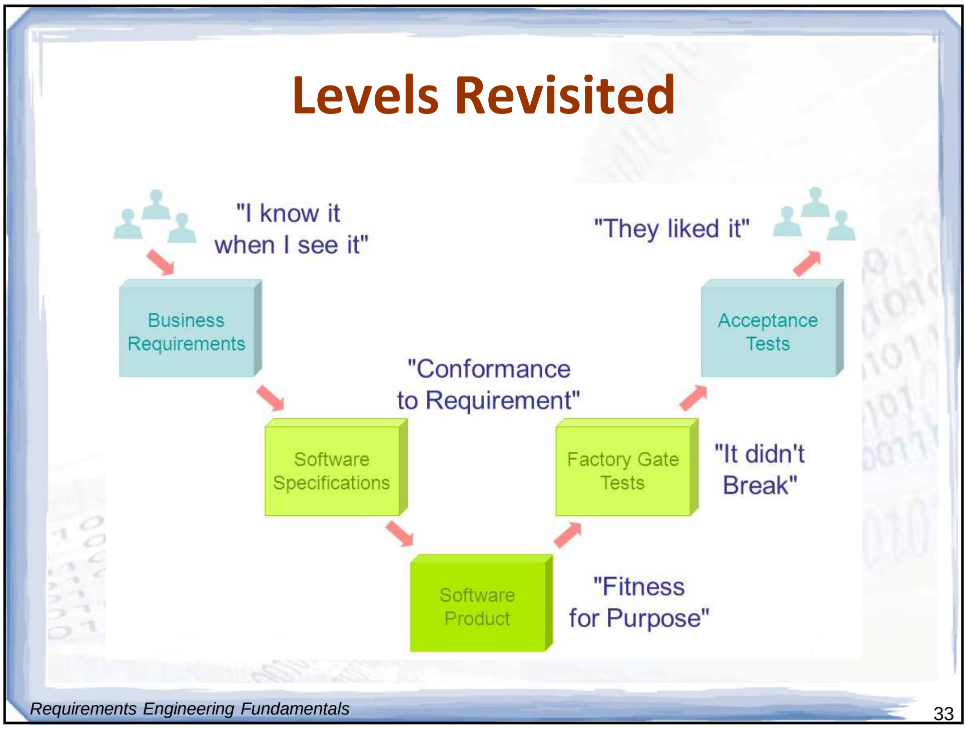
- Business req’s
  - allow the customer to pay for petrol at the pump
- User req’s
  - Recognize credit or debit cards
  - Enter a security PIN number
  - Request a receipt at the pump
- Product (system) req’s
  - Prompt the customer to put his or her card into the reader
  - Detect that the card has been swiped
  - Determine if the card was incorrectly read and prompt the customer to push the card again
  - Parse the information from the magnetic strip on the card …



Layered Approach

The layers correspond to step-wise refinement in terms of component decomposition.

Source: Hull, Jackson, Dick: Requirements Engineering, 2004



Levels Revisited

"I know it when I see it"

Business Requirements

"Conformance to Requirement"

Software Specifications

Software Product

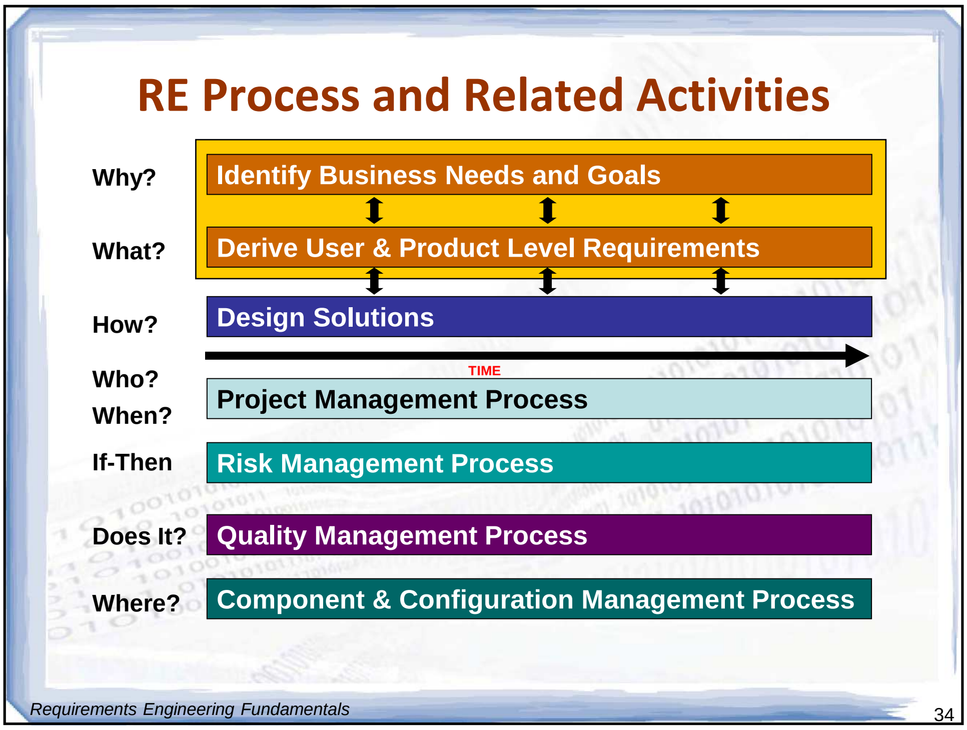
"Fitness for Purpose"

Factory Gate Tests

"It didn't Break"

Acceptance Tests

"They liked it"



RE Process and Related Activities

Why? **Identify Business Needs and Goals**

What? **Derive User & Product Level Requirements**

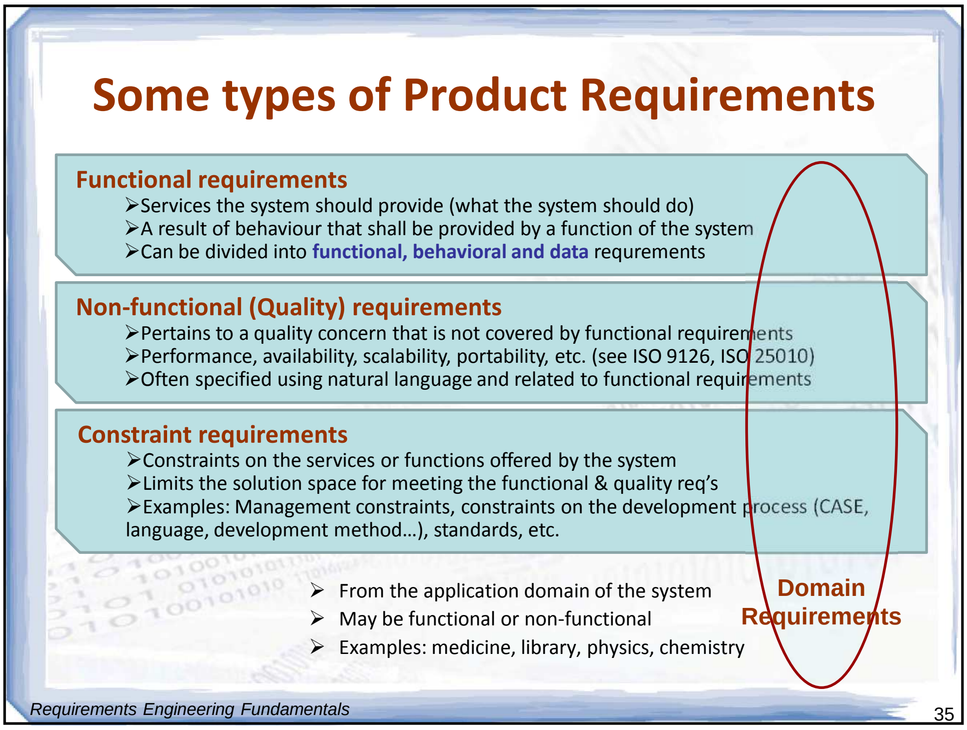
How? **Design Solutions**

Who? When? **Project Management Process**

If-Then **Risk Management Process**

Does It? **Quality Management Process**

Where? **Component & Configuration Management Process**



Some types of Product Requirements

Functional requirements

- ➢Services the system should provide (what the system should do)
- ➢A result of behaviour that shall be provided by a function of the system
- ➢Can be divided into **functional, behavioral and data** requrements

Non-functional (Quality) requirements

- ➢Pertains to a quality concern that is not covered by functional requirements
- ➢Performance, availability, scalability, portability, etc. (see ISO 9126, ISO 25010)
- ➢Often specified using natural language and related to functional requirements

Constraint requirements

- ➢Constraints on the services or functions offered by the system
- ➢Limits the solution space for meeting the functional & quality req's
- ➢Examples: Management constraints, constraints on the development process (CASE, language, development method…), standards, etc.

- From the application domain of the system
- May be functional or non-functional
- Examples: medicine, library, physics, chemistry

Domain Requirements

Exercise

Non-functional Requirements

It is very important to be able to test/verify/check non-functional requirements such as

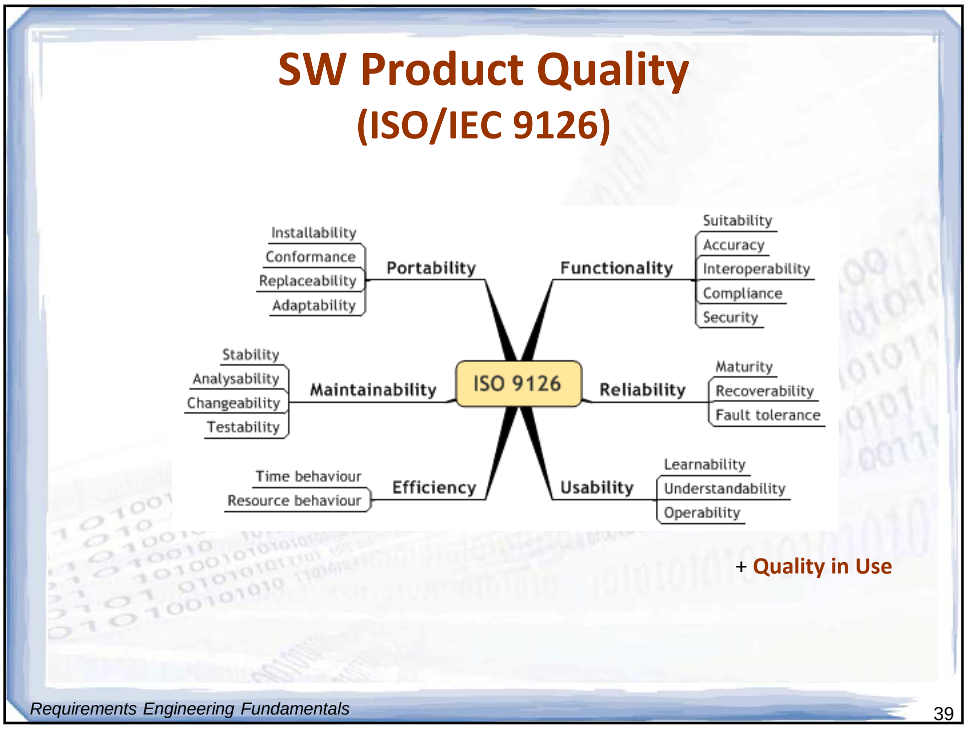
Property	**Measure**
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

NFR Interaction

- Conflicts between different non-functional requirements are common in complex systems
- Spacecraft system
  - To minimise weight, the number of separate chips in the system should be minimised
  - To minimise power consumption, lower power chips should be used
  - However, using low power chips may mean that more chips have to be used.

Which is the most critical requirement?

→ **Priorities!**

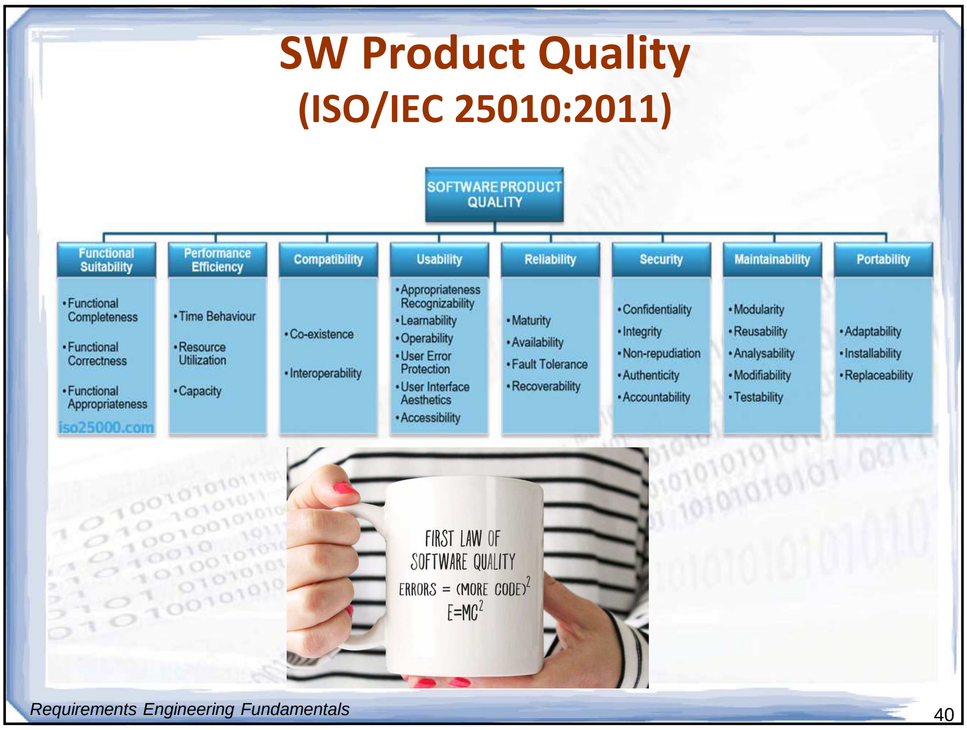


SW Product Quality
(ISO/IEC 9126)

ISO 9126

- **Portability**
  - Installability
  - Conformance
  - Replaceability
  - Adaptability
- **Functionality**
  - Suitability
  - Accuracy
  - Interoperability
  - Compliance
  - Security
- **Maintainability**
  - Stability
  - Analysability
  - Changeability
  - Testability
- **Reliability**
  - Maturity
  - Recoverability
  - Fault tolerance
- **Efficiency**
  - Time behaviour
  - Resource behaviour
- **Usability**
  - Learnability
  - Understandability
  - Operability

+ **Quality in Use**

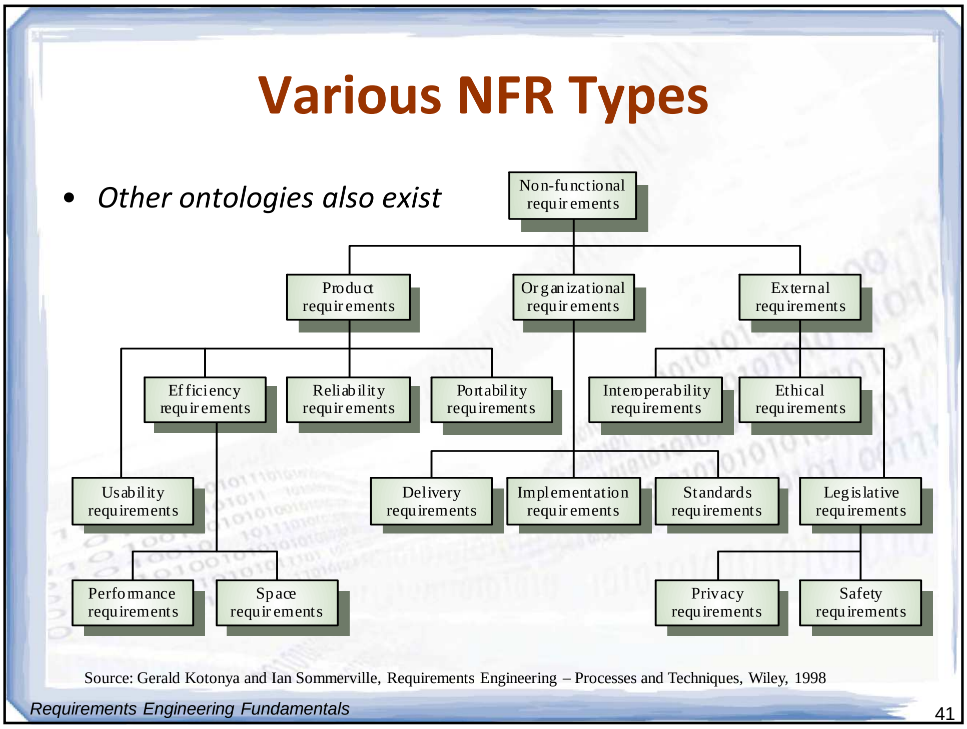


SW Product Quality
(ISO/IEC 25010:2011)

SOFTWARE PRODUCT QUALITY

Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintainability	Portability
• Functional Completeness • Functional Correctness • Functional Appropriateness	• Time Behaviour • Resource Utilization • Capacity	• Co-existence • Interoperability	• Appropriateness Recognizability • Learnability • Operability • User Error Protection • User Interface Aesthetics • Accessibility	• Maturity • Availability • Fault Tolerance • Recoverability	• Confidentiality • Integrity • Non-repudiation • Authenticity • Accountability	• Modularity • Reusability • Analysability • Modifiability • Testability	• Adaptability • Installability • Replaceability

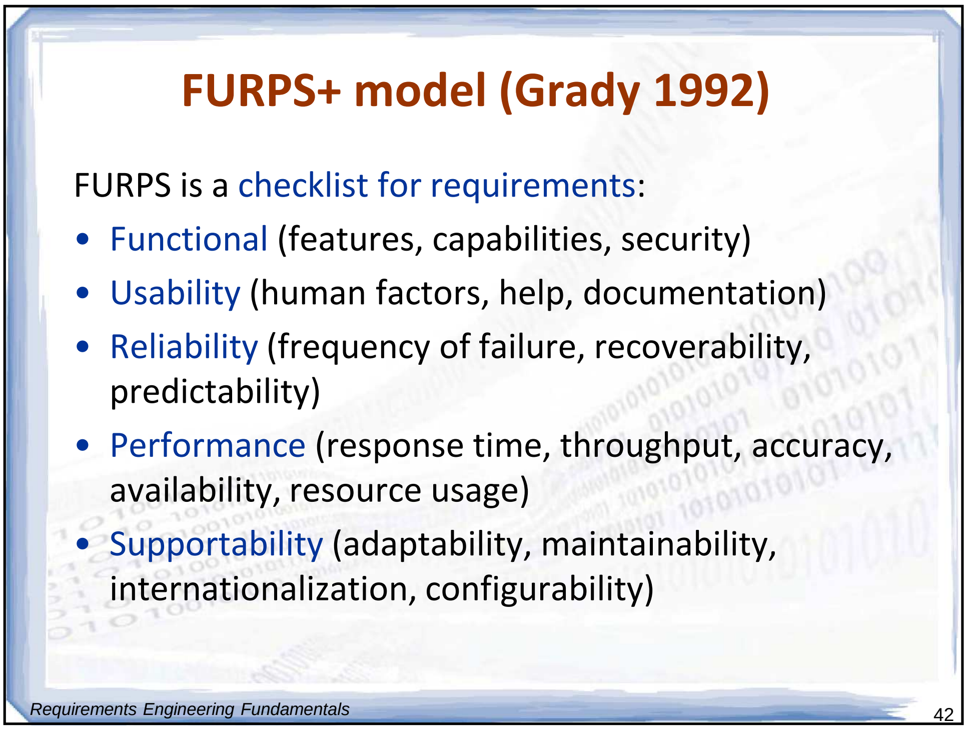
iso25000.com



Various NFR Types

- *Other ontologies also exist*

Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998



FURPS+ model (Grady 1992)

FURPS is a checklist for requirements:

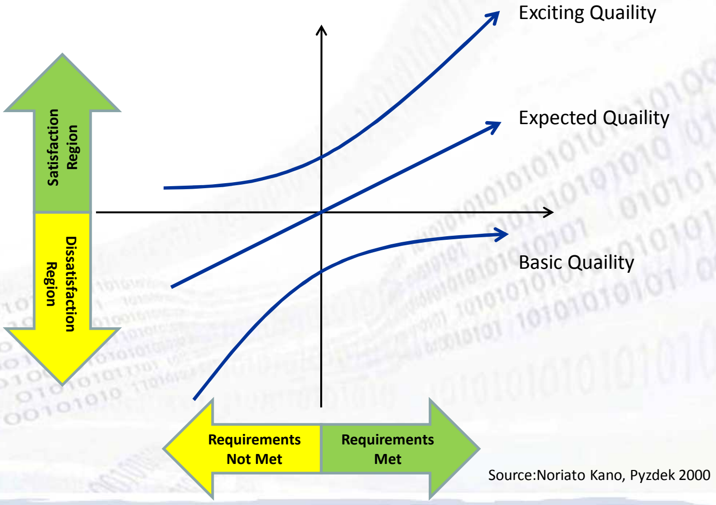
- Functional (features, capabilities, security)
- Usability (human factors, help, documentation)
- Reliability (frequency of failure, recoverability, predictability)
- Performance (response time, throughput, accuracy, availability, resource usage)
- Supportability (adaptability, maintainability, internationalization, configurability)

What's with the + in FURPS+?

And don't forget….

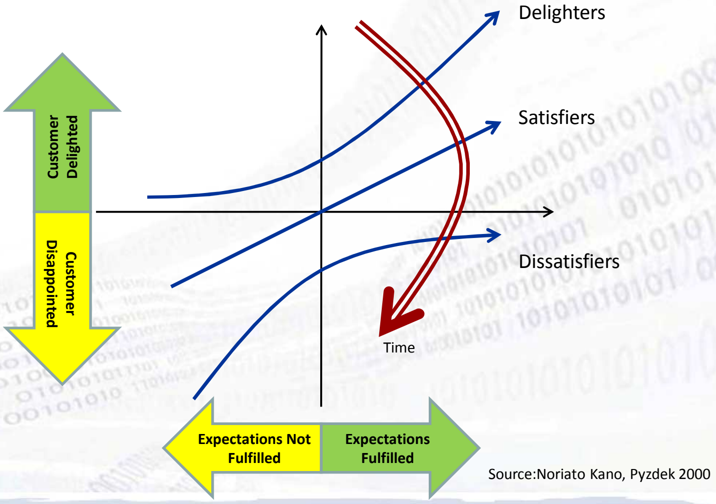
- Implementation (resource limitation, language and tools, hardware)
- Interface (constraints posed by interfacing with external systems)
- Operations (system management in its operational setting)
- Packaging (for example, a physical box)
- Legal (licencing)

Kano Model for Quality Requirements



Source:Noriato Kano, Pyzdek 2000

Kano Model for Requirements Categorization



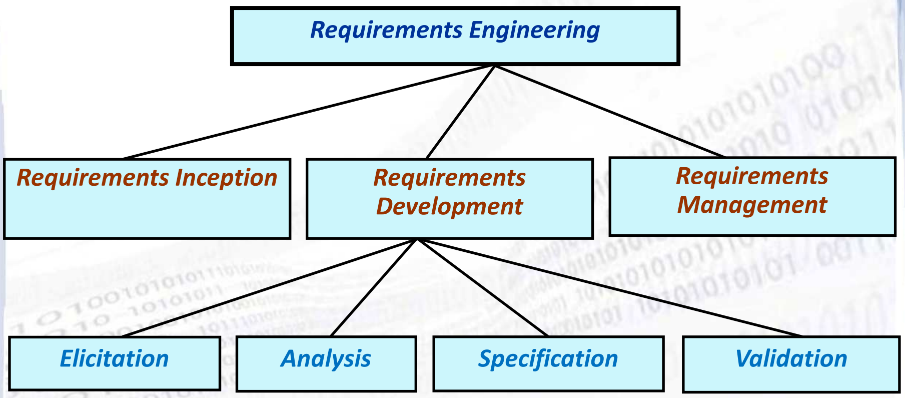
Source:Noriato Kano, Pyzdek 2000

Kano Model for Requirements Categorization

➢Dissatisfiers (normal req's): properties that are self-evident and represent subconscious knowledge
➢Satisfiers (expected req's): explicitly demanded system properties (conscious knowledge)
➢Delighters (exciting req's): properties that the stakeholder does not know and cause a pleasant and useful surprise (unconscious knowledge)

Over time delighters turn into satisfiers and finally into dissatisfiers.

Requirements Engineering Activities

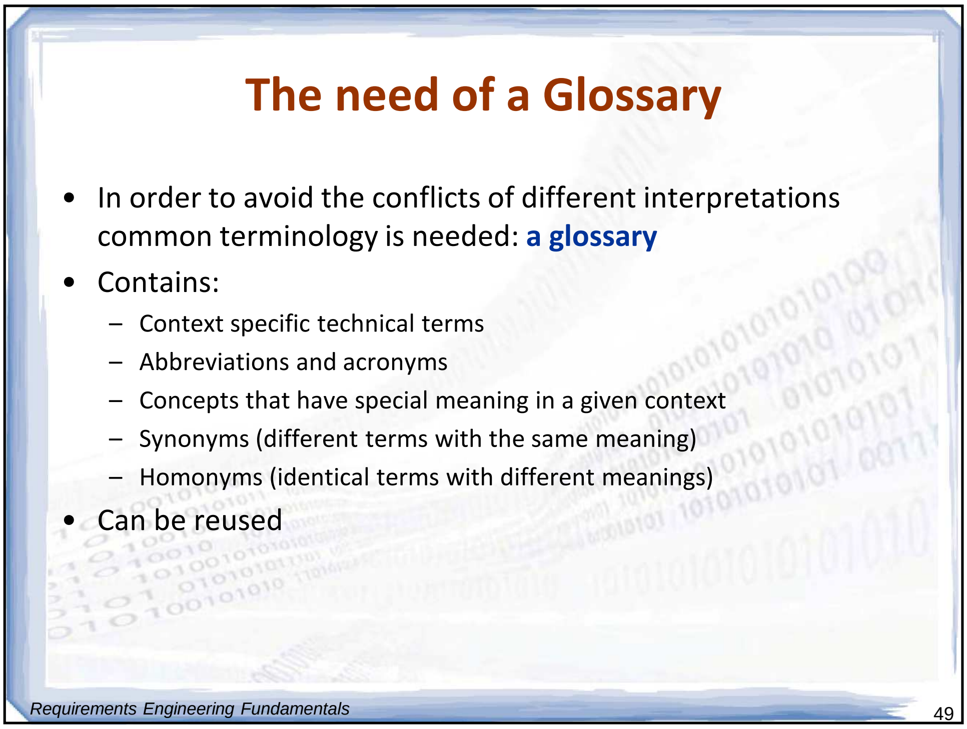


Source: Larry Boldt, Trends in Requirements Engineering People-Process-Technology, Technology Builders, Inc., 2001

Core Activities

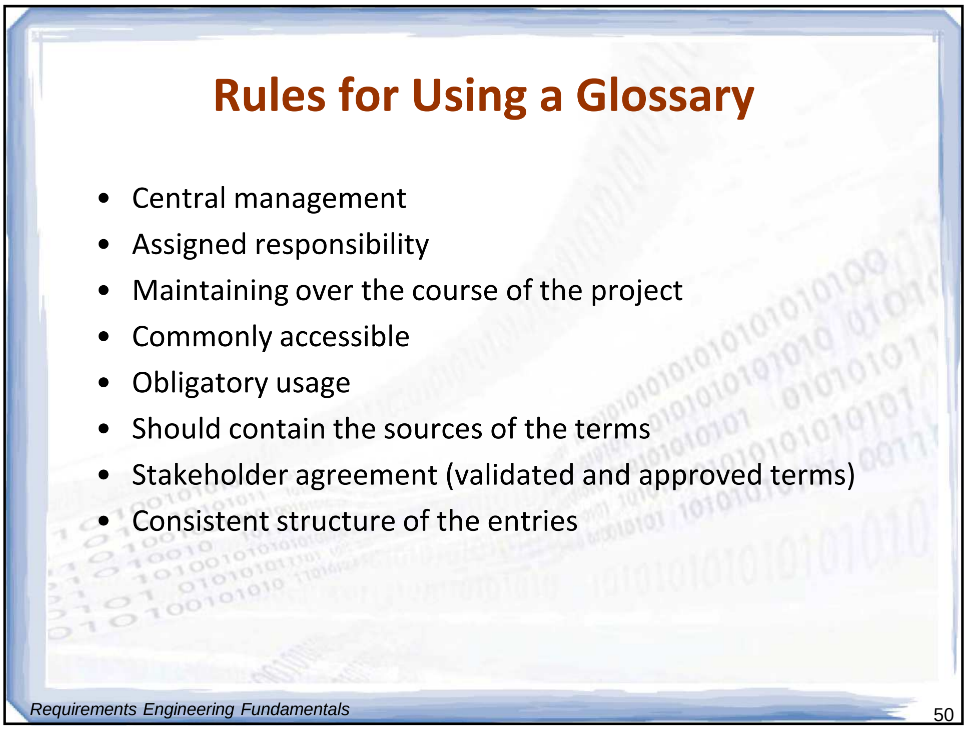
- **Req. Inception** starts the process (business need, market opportunity, great idea, ...), business case, feasibility study, system scope, main risks, etc.
- **Req. Development** is a systematic approach for
  - **Eliciting,**
  - **Analyzing & Negotiating,**
  - **Documenting and**
  - **Validating**

  the requirements of the system
- **Req. Management** Process establishes and maintains **agreement** between the customer and the project team **on the changing requirements** of the system.



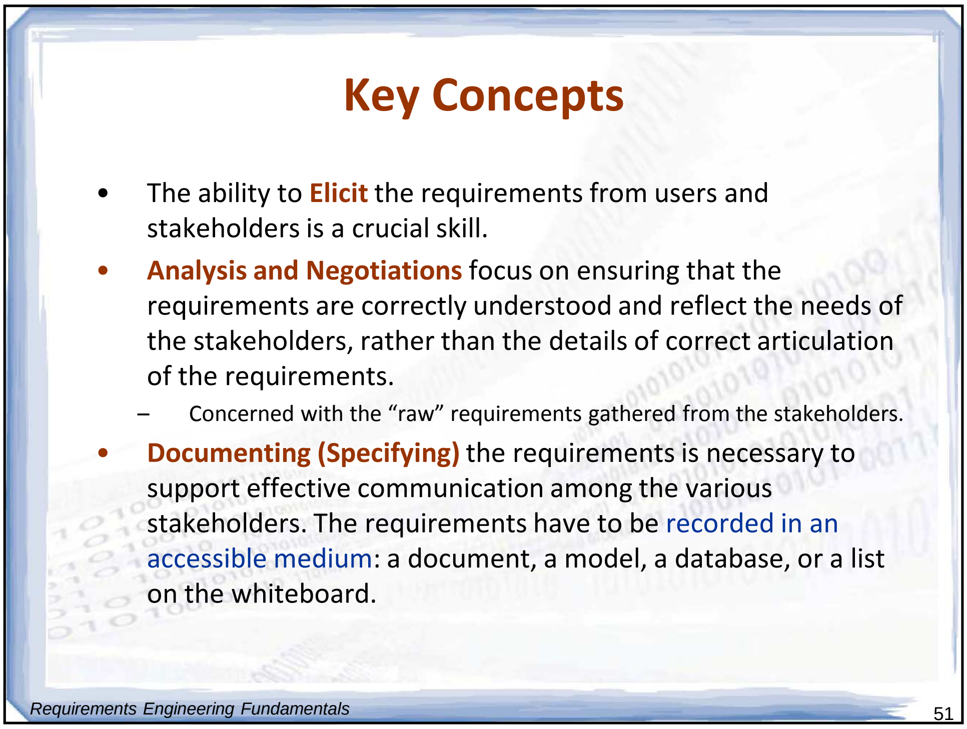
The need of a Glossary

- In order to avoid the conflicts of different interpretations common terminology is needed: **a glossary**
- Contains:
  - Context specific technical terms
  - Abbreviations and acronyms
  - Concepts that have special meaning in a given context
  - Synonyms (different terms with the same meaning)
  - Homonyms (identical terms with different meanings)
- Can be reused



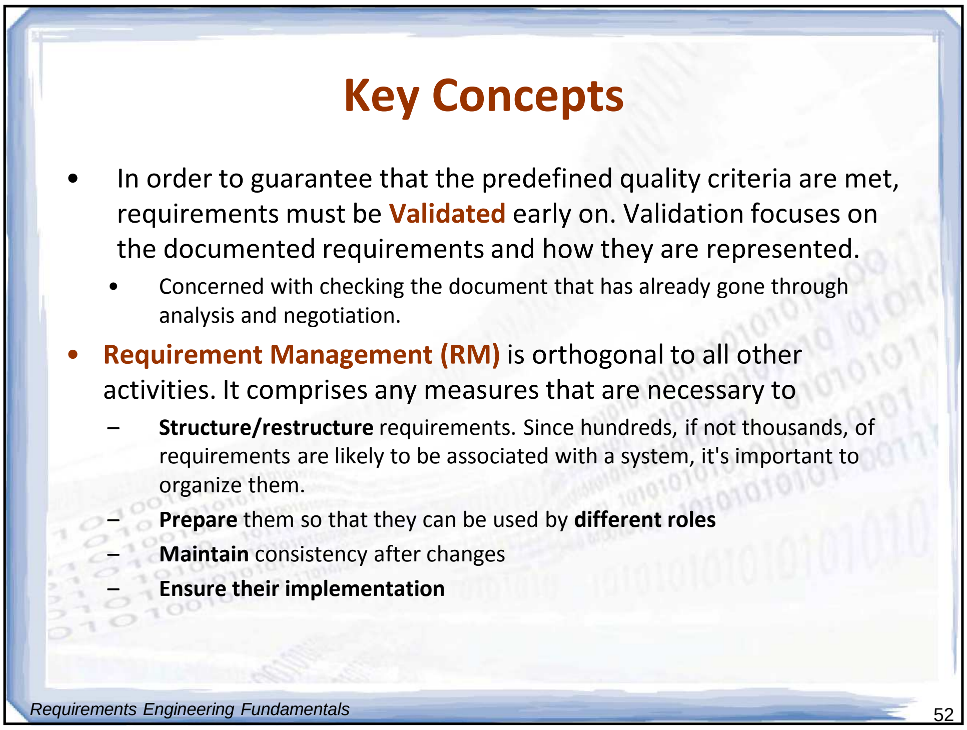
Rules for Using a Glossary

- Central management
- Assigned responsibility
- Maintaining over the course of the project
- Commonly accessible
- Obligatory usage
- Should contain the sources of the terms
- Stakeholder agreement (validated and approved terms)
- Consistent structure of the entries



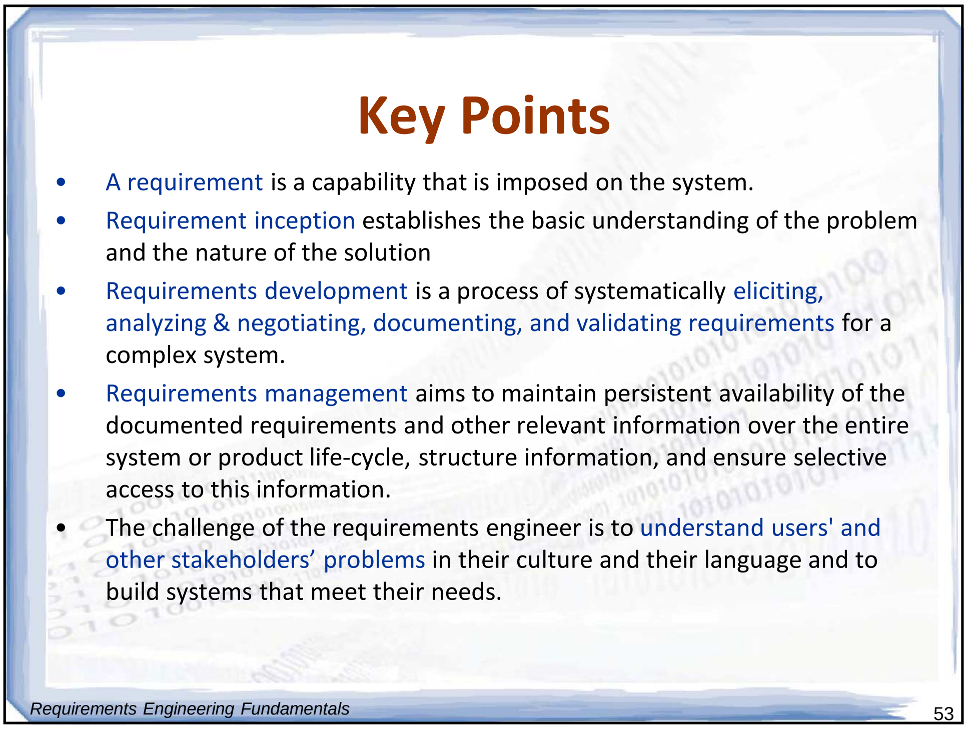
Key Concepts

- The ability to **Elicit** the requirements from users and stakeholders is a crucial skill.
- **Analysis and Negotiations** focus on ensuring that the requirements are correctly understood and reflect the needs of the stakeholders, rather than the details of correct articulation of the requirements.
  - Concerned with the “raw” requirements gathered from the stakeholders.
- **Documenting (Specifying)** the requirements is necessary to support effective communication among the various stakeholders. The requirements have to be recorded in an accessible medium: a document, a model, a database, or a list on the whiteboard.



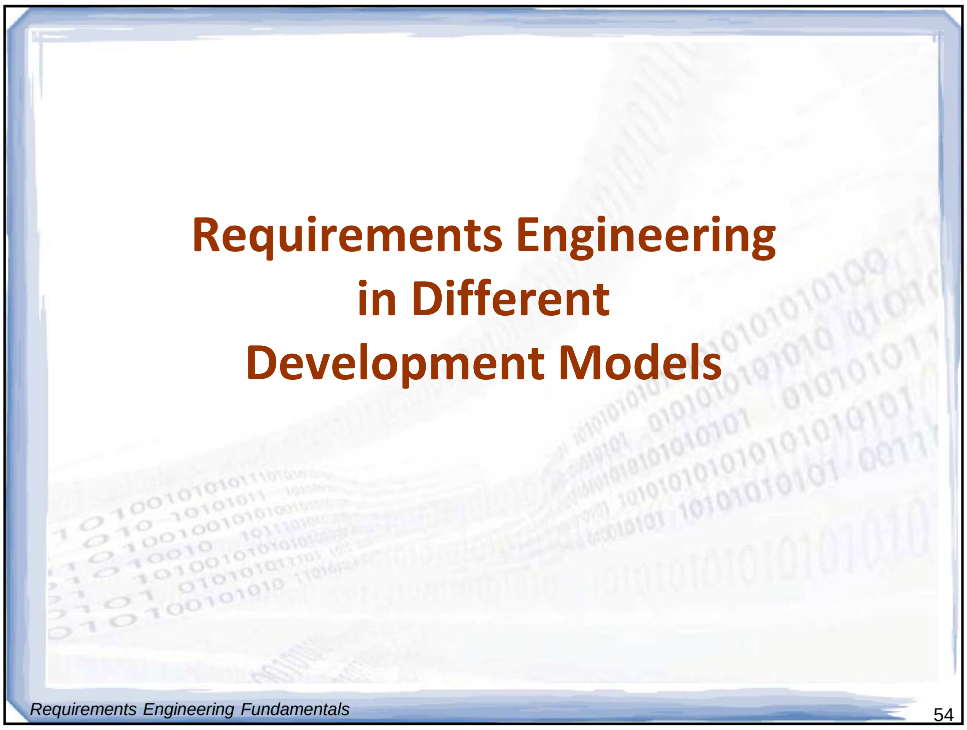
Key Concepts

- In order to guarantee that the predefined quality criteria are met, requirements must be **Validated** early on. Validation focuses on the documented requirements and how they are represented.
  - Concerned with checking the document that has already gone through analysis and negotiation.
- **Requirement Management (RM)** is orthogonal to all other activities. It comprises any measures that are necessary to
  - **Structure/restructure** requirements. Since hundreds, if not thousands, of requirements are likely to be associated with a system, it's important to organize them.
  - **Prepare** them so that they can be used by **different roles**
  - **Maintain** consistency after changes
  - **Ensure their implementation**

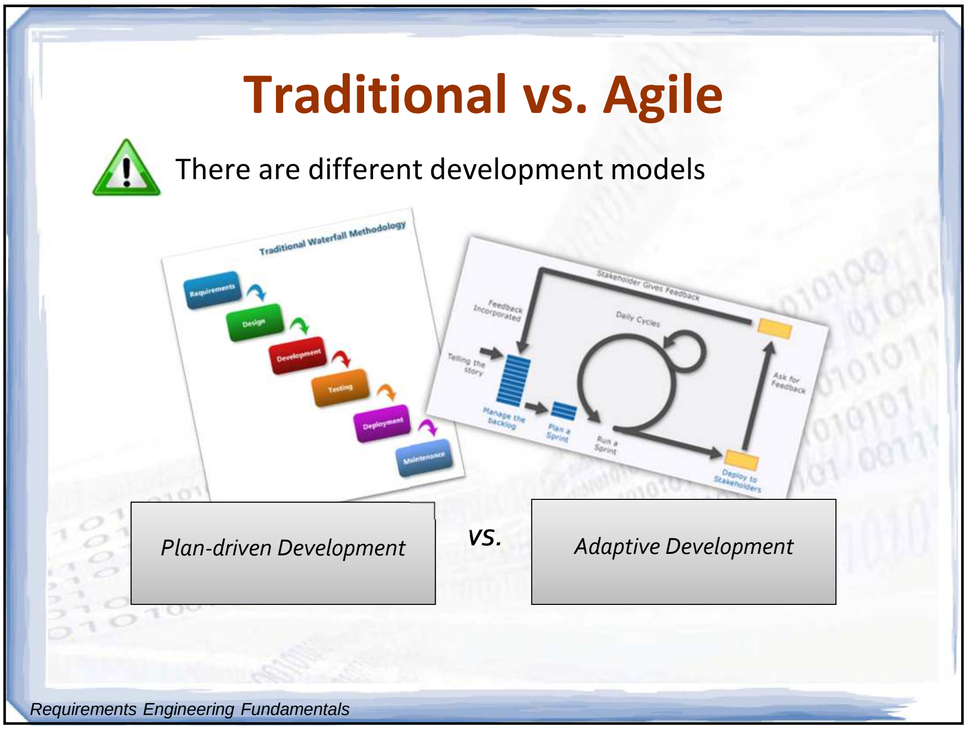


Key Points

- A requirement is a capability that is imposed on the system.
- Requirement inception establishes the basic understanding of the problem and the nature of the solution
- Requirements development is a process of systematically eliciting, analyzing & negotiating, documenting, and validating requirements for a complex system.
- Requirements management aims to maintain persistent availability of the documented requirements and other relevant information over the entire system or product life-cycle, structure information, and ensure selective access to this information.
- The challenge of the requirements engineer is to understand users' and other stakeholders' problems in their culture and their language and to build systems that meet their needs.



Requirements Engineering in Different Development Models



Traditional vs. Agile

There are different development models

Plan-driven Development vs. *Adaptive Development*

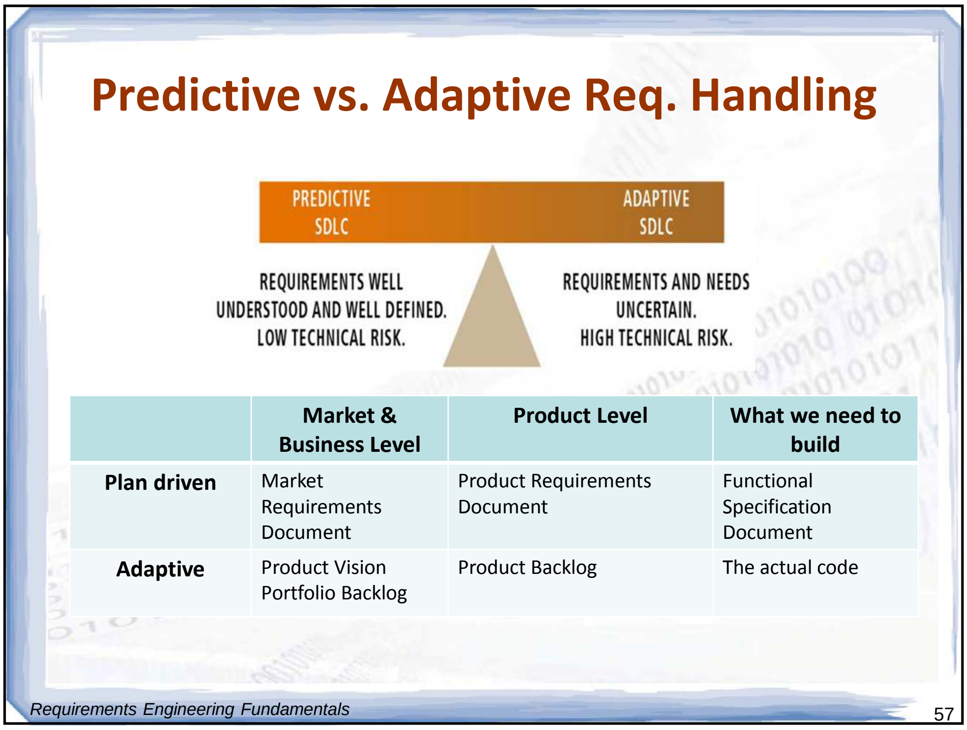
Adaptive Development

The Agile Manifesto

Individuals and interactions	over	Processes and Tools
Working Product	over	Comprehensive Documentation
Customer Collaboration	over	Contract Negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

www.agilemanifesto.org



Predictive vs. Adaptive Req. Handling

PREDICTIVE SDLC

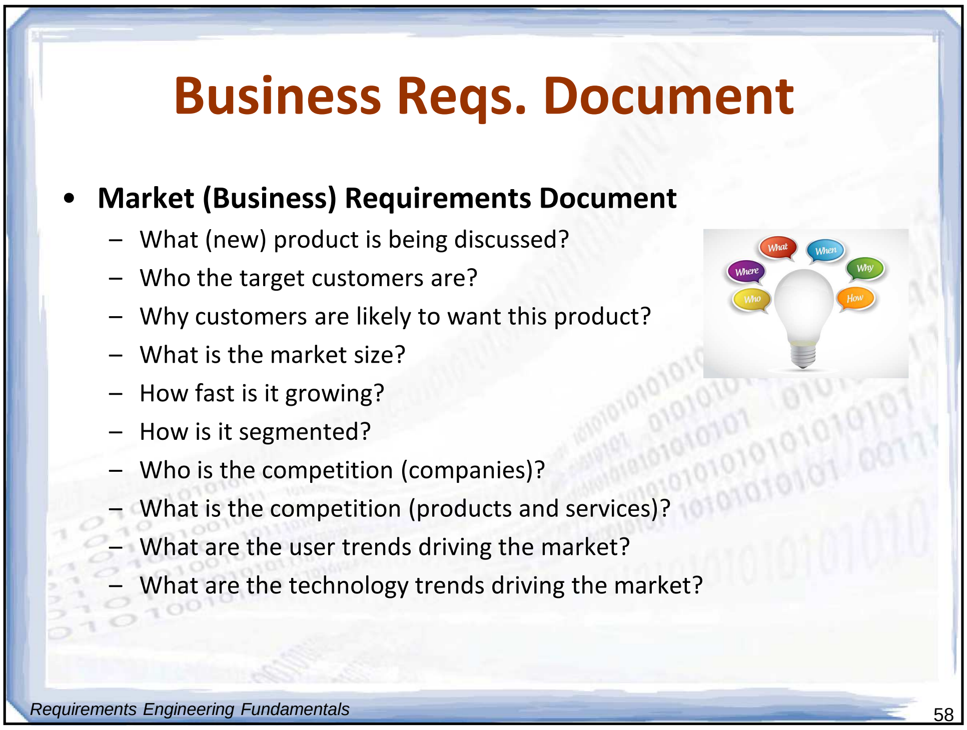
ADAPTIVE SDLC

REQUIREMENTS WELL UNDERSTOOD AND WELL DEFINED. LOW TECHNICAL RISK.

REQUIREMENTS AND NEEDS UNCERTAIN. HIGH TECHNICAL RISK.

	Market & Business Level	Product Level	What we need to build
Plan driven	Market Requirements Document	Product Requirements Document	Functional Specification Document
Adaptive	Product Vision Portfolio Backlog	Product Backlog	The actual code

Requirements Engineering Fundamentals



Business Reqs. Document

- **Market (Business) Requirements Document**
  - What (new) product is being discussed?
  - Who the target customers are?
  - Why customers are likely to want this product?
  - What is the market size?
  - How fast is it growing?
  - How is it segmented?
  - Who is the competition (companies)?
  - What is the competition (products and services)?
  - What are the user trends driving the market?
  - What are the technology trends driving the market?

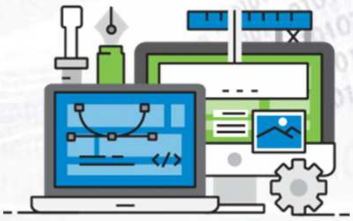
Requirements Engineering Fundamentals

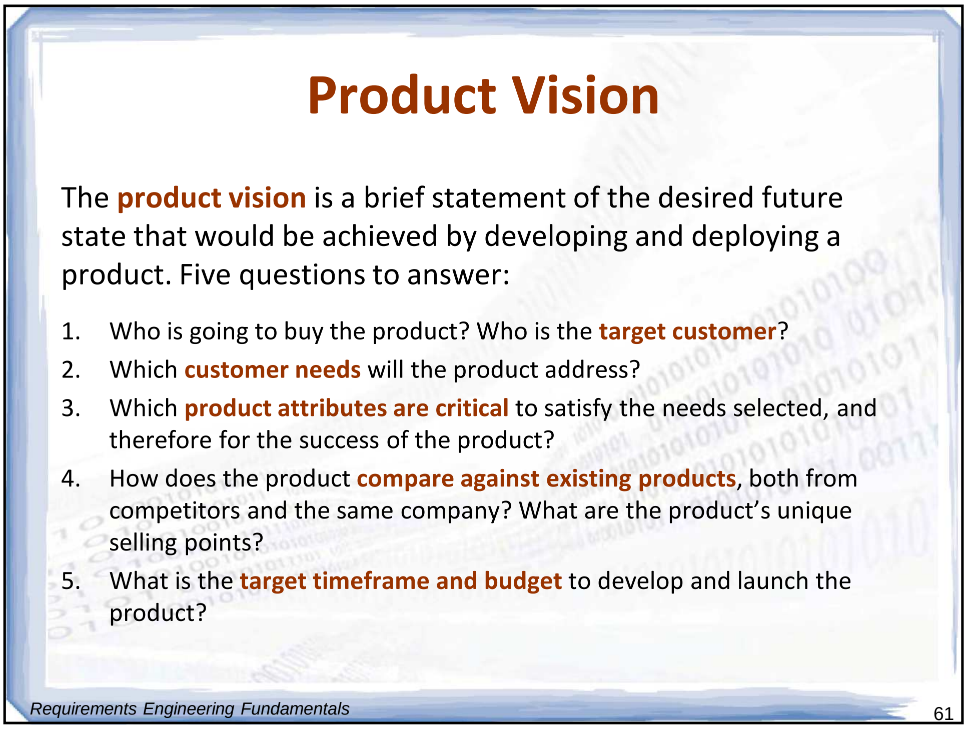
Product Reqs. Document

- **Product Requirements Document**
  - Title & author information
  - Purpose and scope, from both a technical and business perspective
  - Stakeholder identification
  - Market assessment and target demographics
  - Product overview and use cases
  - Requirements, including
    - Functional requirements (e.g. what a product should do)
    - Usability requirements
    - Technical requirements (e.g. security, network, platform, integration)
    - Environmental requirements
    - Transition and support requirements
    - Interaction requirements (e.g. how the product should work with other systems)
  - Assumptions, constraints, dependencies
  - High level workflow plans, timelines and milestones
  - Evaluation plan and performance metrics

Functional Reqs. Document

- **Functional Specification Document**
  - Describes how each functional requirement should be implemented considering the other requirements
  - High level guideline to the SW architects/implementers on how to write code
  - It is internal, not shared with customer
  - It is technical

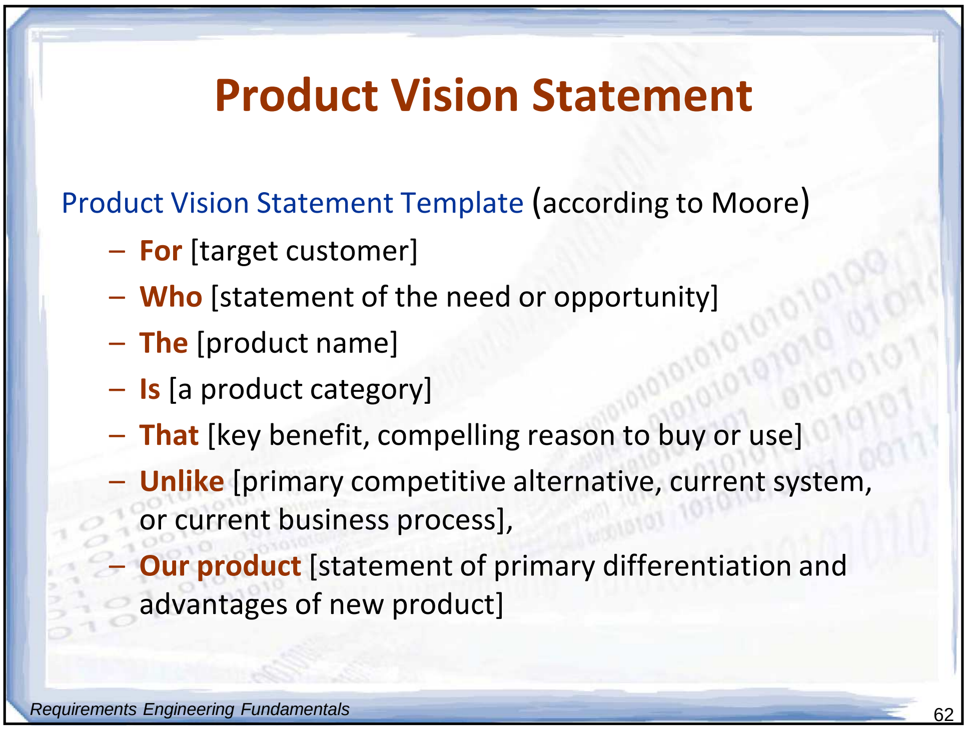




Product Vision

The **product vision** is a brief statement of the desired future state that would be achieved by developing and deploying a product. Five questions to answer:

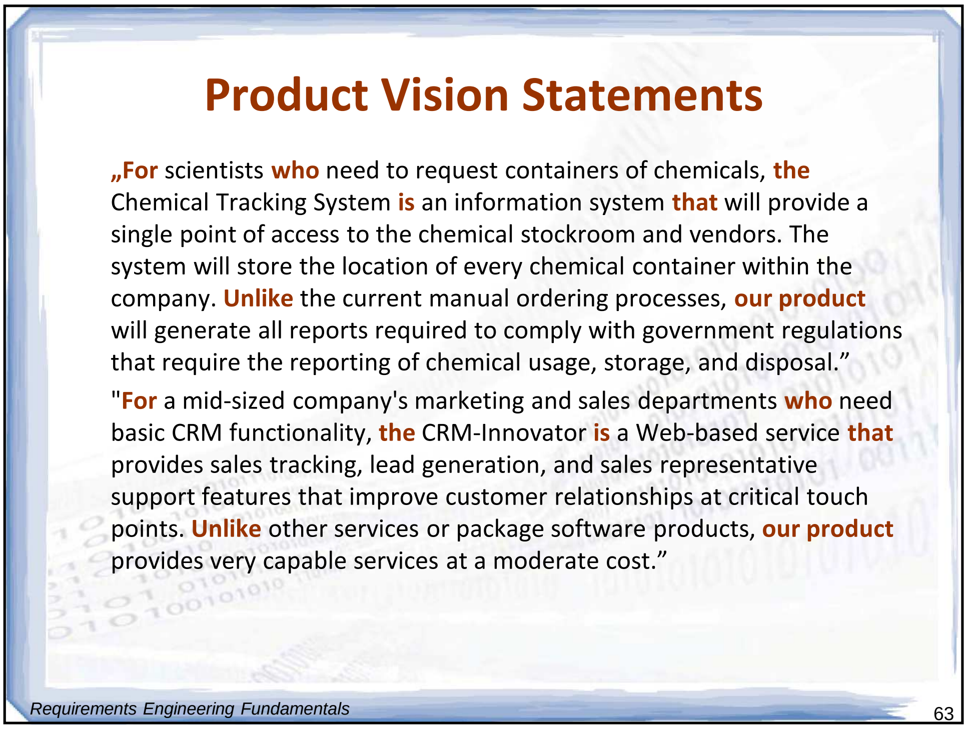
1. Who is going to buy the product? Who is the **target customer**?
2. Which **customer needs** will the product address?
3. Which **product attributes are critical** to satisfy the needs selected, and therefore for the success of the product?
4. How does the product **compare against existing products**, both from competitors and the same company? What are the product's unique selling points?
5. What is the **target timeframe and budget** to develop and launch the product?



Product Vision Statement

Product Vision Statement Template (according to Moore)

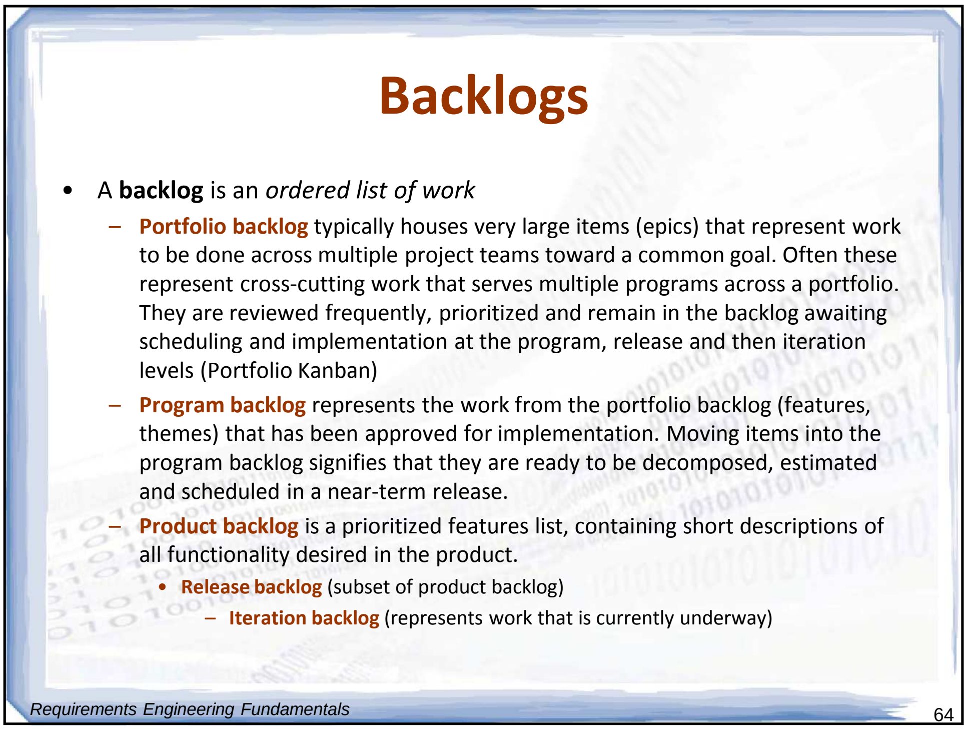
- **For** [target customer]
- **Who** [statement of the need or opportunity]
- **The** [product name]
- **Is** [a product category]
- **That** [key benefit, compelling reason to buy or use]
- **Unlike** [primary competitive alternative, current system, or current business process],
- **Our product** [statement of primary differentiation and advantages of new product]



Product Vision Statements

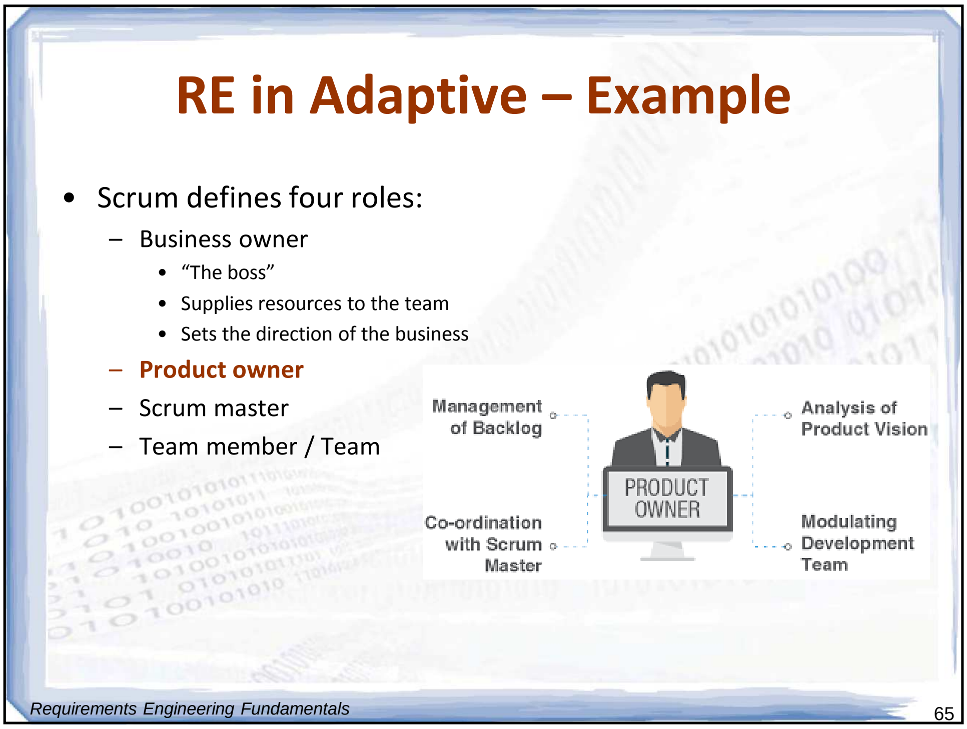
„For scientists **who** need to request containers of chemicals, **the** Chemical Tracking System **is** an information system **that** will provide a single point of access to the chemical stockroom and vendors. The system will store the location of every chemical container within the company. **Unlike** the current manual ordering processes, **our product** will generate all reports required to comply with government regulations that require the reporting of chemical usage, storage, and disposal."

"**For** a mid-sized company's marketing and sales departments **who** need basic CRM functionality, **the** CRM-Innovator **is** a Web-based service **that** provides sales tracking, lead generation, and sales representative support features that improve customer relationships at critical touch points. **Unlike** other services or package software products, **our product** provides very capable services at a moderate cost."



Backlogs

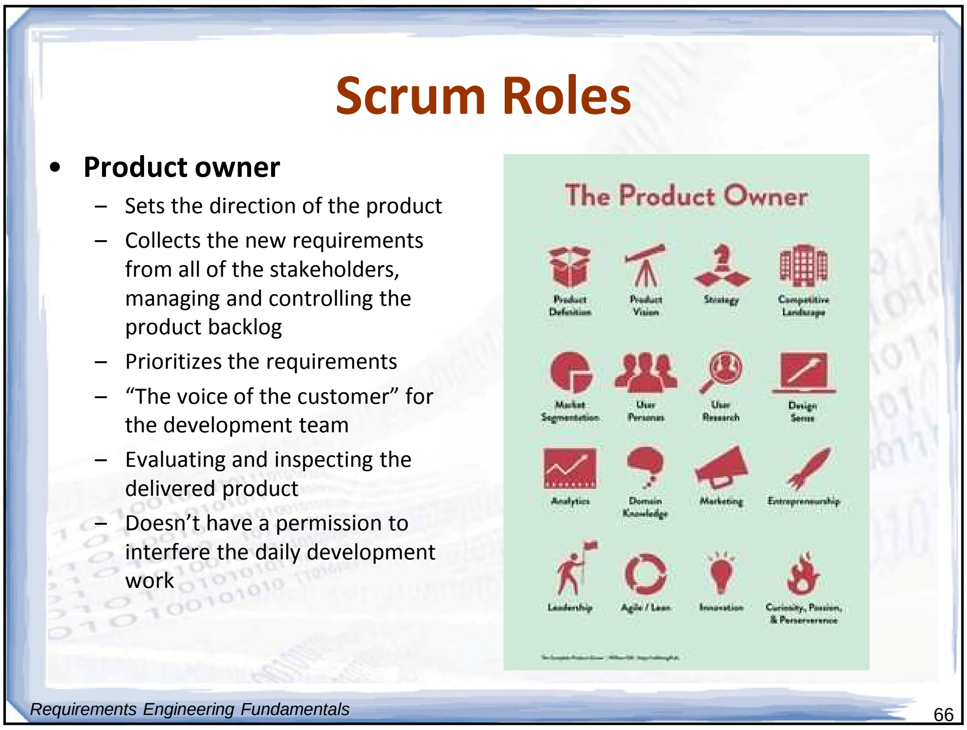
- A **backlog** is an *ordered list of work*
  - **Portfolio backlog** typically houses very large items (epics) that represent work to be done across multiple project teams toward a common goal. Often these represent cross-cutting work that serves multiple programs across a portfolio. They are reviewed frequently, prioritized and remain in the backlog awaiting scheduling and implementation at the program, release and then iteration levels (Portfolio Kanban)
  - **Program backlog** represents the work from the portfolio backlog (features, themes) that has been approved for implementation. Moving items into the program backlog signifies that they are ready to be decomposed, estimated and scheduled in a near-term release.
  - **Product backlog** is a prioritized features list, containing short descriptions of all functionality desired in the product.
    - **Release backlog** (subset of product backlog)
      - **Iteration backlog** (represents work that is currently underway)



RE in Adaptive – Example

- Scrum defines four roles:
  - Business owner
    - "The boss"
    - Supplies resources to the team
    - Sets the direction of the business
  - Product owner
  - Scrum master
  - Team member / Team

Management of Backlog
Analysis of Product Vision
PRODUCT OWNER
Co-ordination with Scrum Master
Modulating Development Team



Scrum Roles

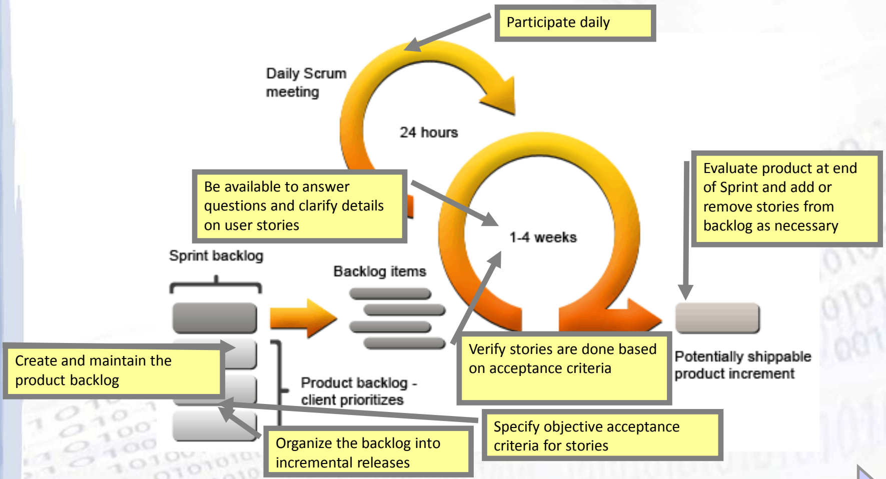
- **Product owner**
  - Sets the direction of the product
  - Collects the new requirements from all of the stakeholders, managing and controlling the product backlog
  - Prioritizes the requirements
  - "The voice of the customer" for the development team
  - Evaluating and inspecting the delivered product
  - Doesn't have a permission to interfere the daily development work

Product Owner

- Subject Matter Expert
  - Understand the domain well enough to envision a product
  - Answer technical questions on the domain for those creating the product
- End User Advocate
  - Describe the product with understanding of users and use, and a product that best serves both
- Customer Advocate
  - Understand the needs of the business buying the product and select a mix of features valuable to the customer
- Business Advocate
  - Understand the needs of the organization paying for the software's construction and select a mix of features that serve their goals
- Communicator
  - Capable of communicating vision and intent – deferring detailed feature and design decisions to be made just in time
- Decision Maker
  - Given a variety of conflicting goals and opinions, be the final decision maker for hard product decisions

The Product Owner role is generally filled by a single person supported by a collaborative team

PO Responsibilities


Daily Scrum meeting
24 hours
1-4 weeks
Sprint backlog
Backlog items
Product backlog - client prioritizes
Potentially shippable product increment

Participate daily

Be available to answer questions and clarify details on user stories

Evaluate product at end of Sprint and add or remove stories from backlog as necessary

Create and maintain the product backlog

Verify stories are done based on acceptance criteria

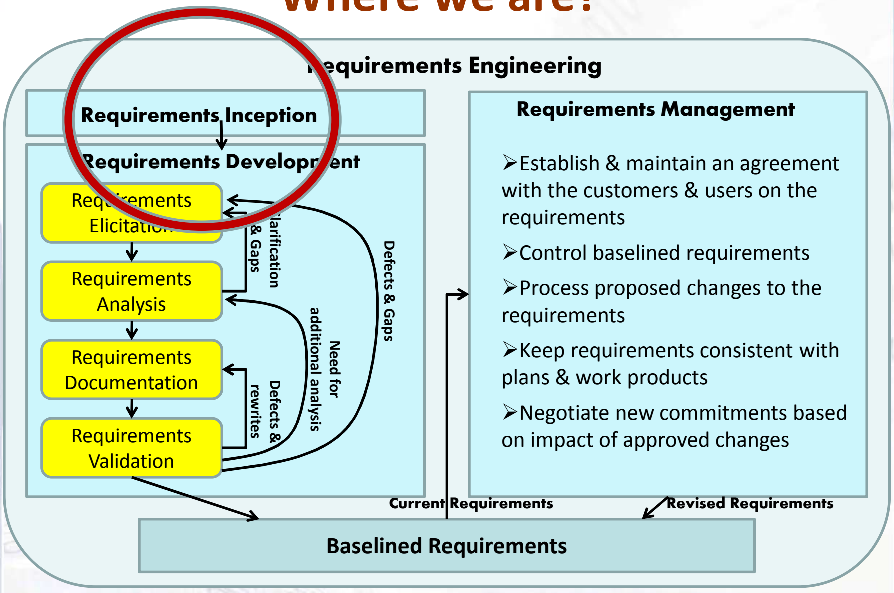
Specify objective acceptance criteria for stories

Organize the backlog into incremental releases

- Communicate Business Goals, Customer Goals, End User Goals
- Coordinate involvement of Subject Matter Experts, users, and business stakeholders
- Coordinate with other product owners to insure coherence of product and releases

Requirements Inception

Where we are?



Requirements Engineering

Requirements Inception

Requirements Development

Requirements Elicitation

Requirements Analysis

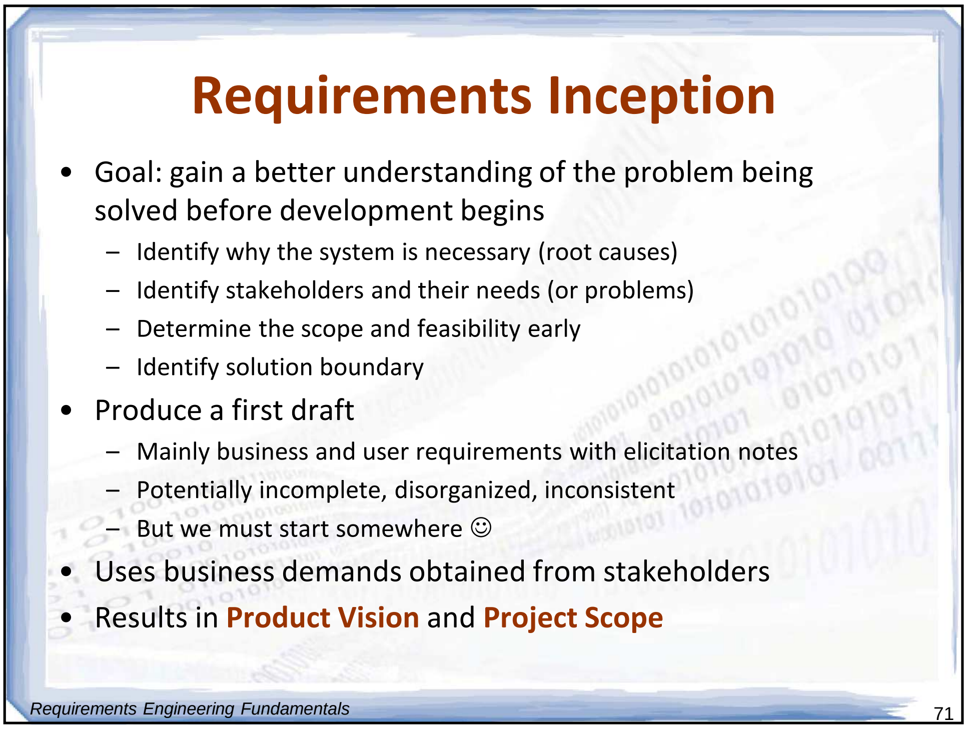
Requirements Documentation

Requirements Validation

Requirements Management

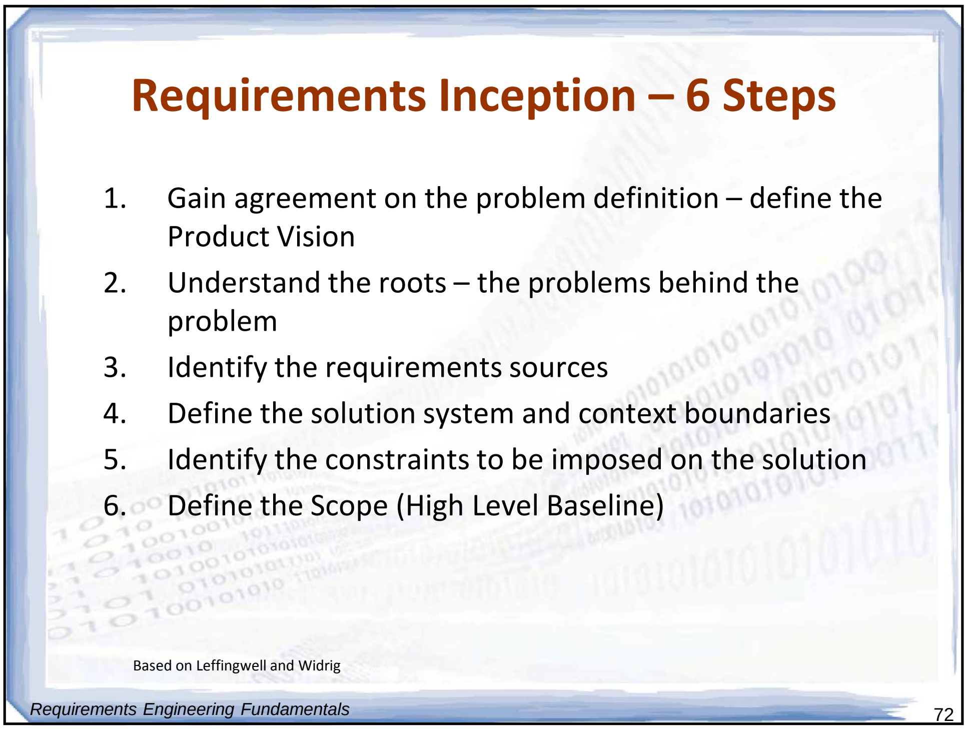
- ➢Establish & maintain an agreement with the customers & users on the requirements
- ➢Control baselined requirements
- ➢Process proposed changes to the requirements
- ➢Keep requirements consistent with plans & work products
- ➢Negotiate new commitments based on impact of approved changes

Baselined Requirements



Requirements Inception

- Goal: gain a better understanding of the problem being solved before development begins
  - Identify why the system is necessary (root causes)
  - Identify stakeholders and their needs (or problems)
  - Determine the scope and feasibility early
  - Identify solution boundary
- Produce a first draft
  - Mainly business and user requirements with elicitation notes
  - Potentially incomplete, disorganized, inconsistent
  - But we must start somewhere ☺
- Uses business demands obtained from stakeholders
- Results in **Product Vision** and **Project Scope**



Requirements Inception – 6 Steps

1. Gain agreement on the problem definition – define the Product Vision
2. Understand the roots – the problems behind the problem
3. Identify the requirements sources
4. Define the solution system and context boundaries
5. Identify the constraints to be imposed on the solution
6. Define the Scope (High Level Baseline)

Based on Leffingwell and Widrig

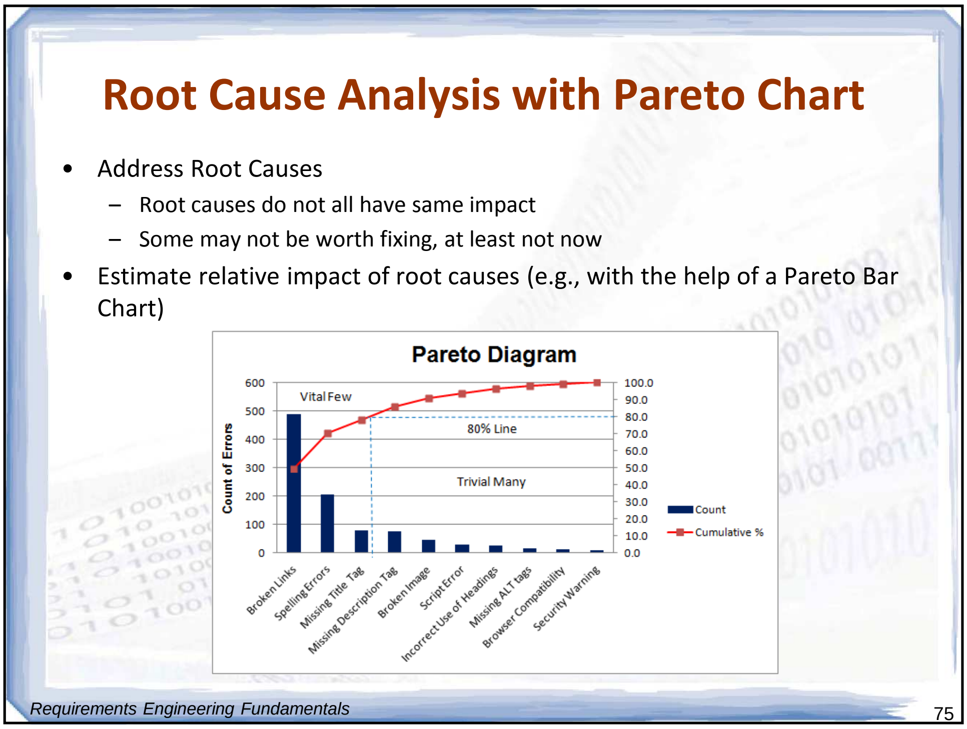
Step 1 – Gain Agreement

Document the problem and seek agreement

- Ask stakeholders to write a problem statement in an agreed format (Vision in Agile)
- Statement should include
  - What the problem is
  - Who is affected by it?
  - What is the impact?
  - Is there a proposed solution?
  - What are the key benefits?

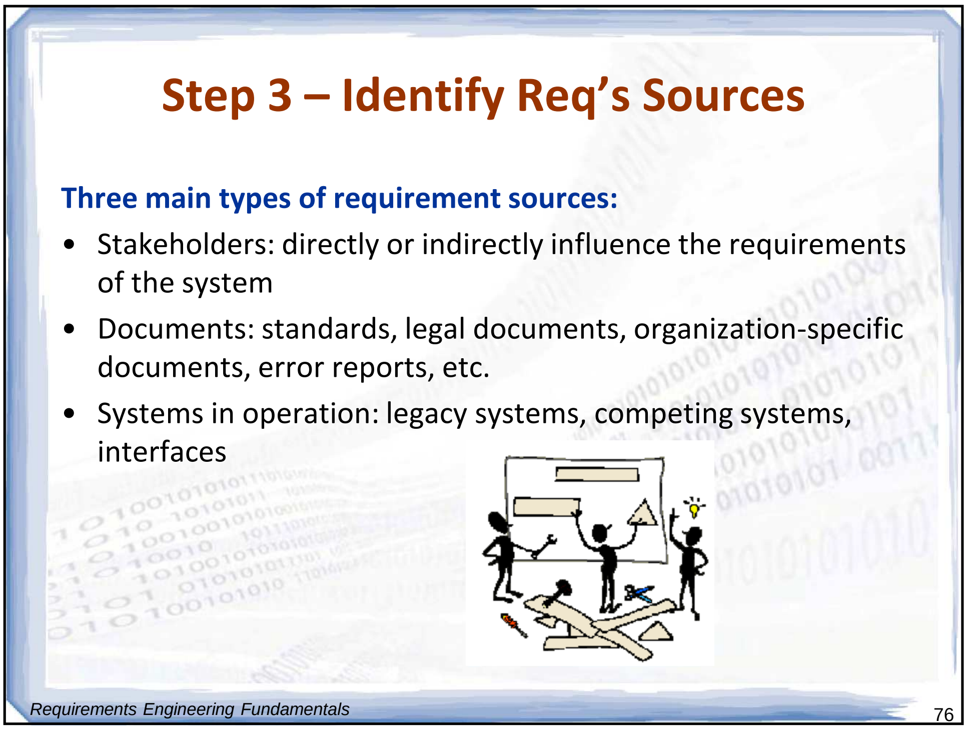
Step 2 – Understand Root Causes

- There is often a problem behind the problem
- **Root cause analysis** consists of finding underlying causes that may not be immediately apparent
- Example: Our e-commerce site is not profitable
  - Why is it not profitable?
  - Poor site design?
  - Bad pricing?
  - Poor customer management after the sale?
  - Some or all of the above?



Root Cause Analysis with Pareto Chart

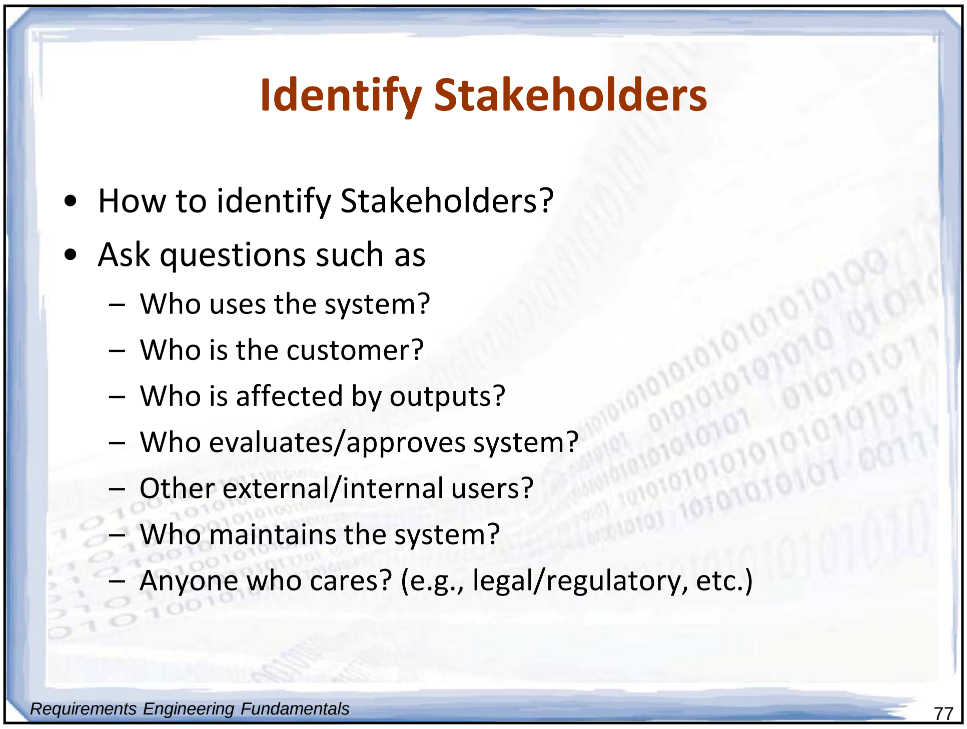
- Address Root Causes
  - Root causes do not all have same impact
  - Some may not be worth fixing, at least not now
- Estimate relative impact of root causes (e.g., with the help of a Pareto Bar Chart)



Step 3 – Identify Req's Sources

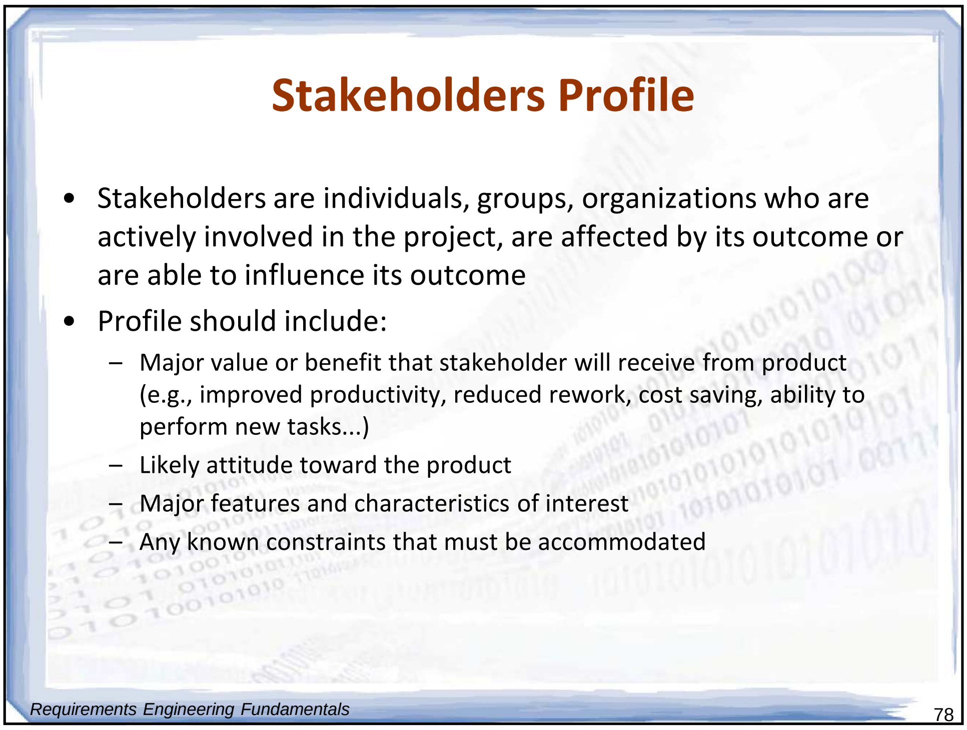
Three main types of requirement sources:

- Stakeholders: directly or indirectly influence the requirements of the system
- Documents: standards, legal documents, organization-specific documents, error reports, etc.
- Systems in operation: legacy systems, competing systems, interfaces



Identify Stakeholders

- How to identify Stakeholders?
- Ask questions such as
  - Who uses the system?
  - Who is the customer?
  - Who is affected by outputs?
  - Who evaluates/approves system?
  - Other external/internal users?
  - Who maintains the system?
  - Anyone who cares? (e.g., legal/regulatory, etc.)



Stakeholders Profile

- Stakeholders are individuals, groups, organizations who are actively involved in the project, are affected by its outcome or are able to influence its outcome
- Profile should include:
  - Major value or benefit that stakeholder will receive from product (e.g., improved productivity, reduced rework, cost saving, ability to perform new tasks...)
  - Likely attitude toward the product
  - Major features and characteristics of interest
  - Any known constraints that must be accommodated

Stakeholders Responsibility

- Introduce the Requirements Engineer to the application domain
- Supply the Requirements Engineer with requirements
- Make timely decisions
- Respect the Requirements Engineer’s estimates (costs, feasibility) and process that has been instated
- Prioritize requirements
- Inspect the documents made by the Requirements Engineer (e.g. prototypes)
- Communicate changes immediately

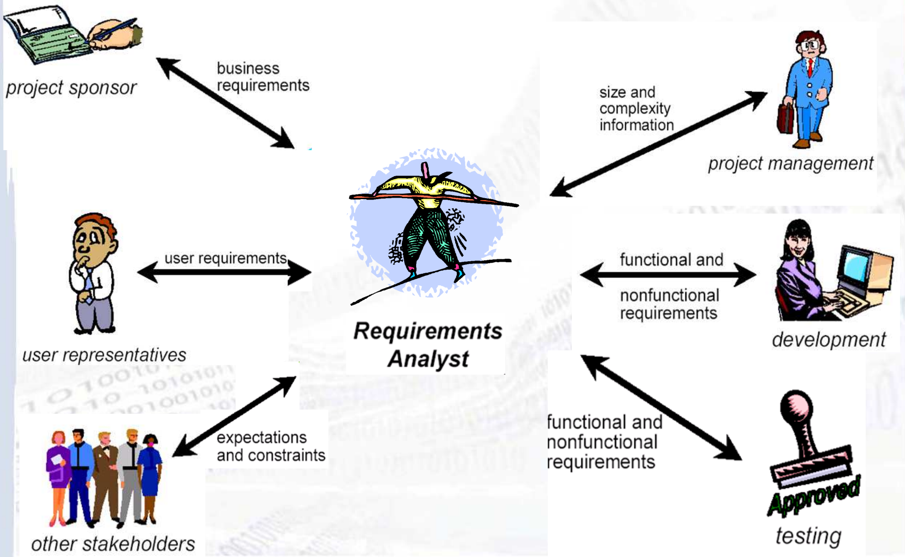
Requirements Engineer

- The Requirements Engineer is responsible for understanding the needs of users and other stakeholders whose lives will be affected by the solution
- Necessary Capabilities of the Requirements Engineer:
  - Analytic Thinking
  - Empathy
  - Communication Skills
  - Conflict Resolution Skills
  - Moderation Skills
  - Self-Confidence
  - Persuasiveness (meggyőzőerő)

Requirements Engineer

- The Requirements Engineer
  - Speaks the language of the stakeholders
  - Become thoroughly familiar with the application domain
  - Creates the requirements document
  - Maintains relationships with the stakeholders
  - Able to present ideas, alternatives and their realizations
  - Allows stakeholders to demand properties that make the system more simple
  - Ensures that the system satisfies all kinds of the stakeholders' demands

Putting RE in wider Context



project sponsor

business requirements

size and complexity information

project management

user requirements

user representatives

Requirements Analyst

functional and nonfunctional requirements

development

expectations and constraints

other stakeholders

functional and nonfunctional requirements

Approved

testing

Techniques that the Req. Eng. should know

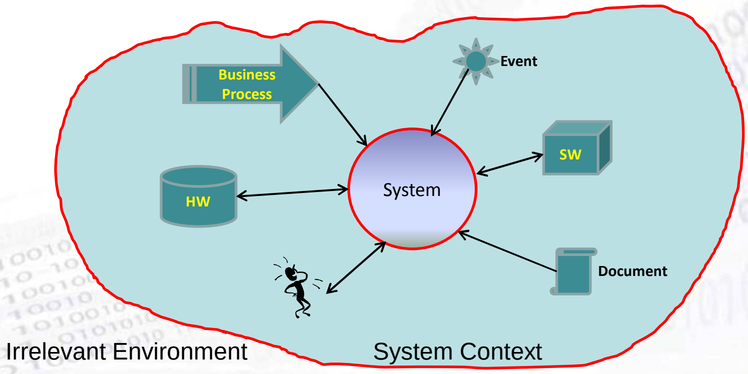
- Acceptance and Evaluation Criteria definition
- Brain Storming
- Business Rules Analysis
- Data Dictionary and Glossary
- Data Flow Diagram
- Data Modeling
- Decision Analysis
- Document Analysis
- Interviews
- Metrics and Key Performance Indicators
- Non-functional requirements Analysis
- Organizational Modeling
- Problem Tracking
- Problem Modeling
- Requirements Workshops
- Scenarios and Use Cases

Step 4 – Define Boundaries (1)

- The aim of this step is to clearly define the boundaries of the system to the **system context** and the boundary of the system context to the **irrelevant environment.**
- Typical aspects **within** the system context are stakeholders, documents, standards, other systems interacting with the system to be developed.
- This is the basis of the systematic elicitation. Various elicitation techniques support the requirements engineer in ascertaining the knowledge of the stakeholders

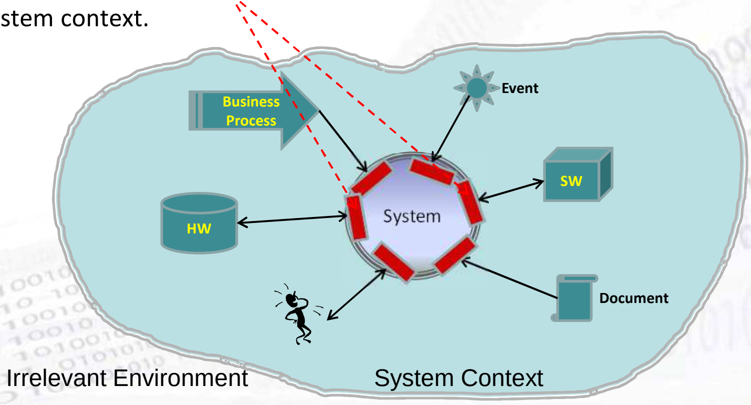
Step 4 – Define Boundaries (2)

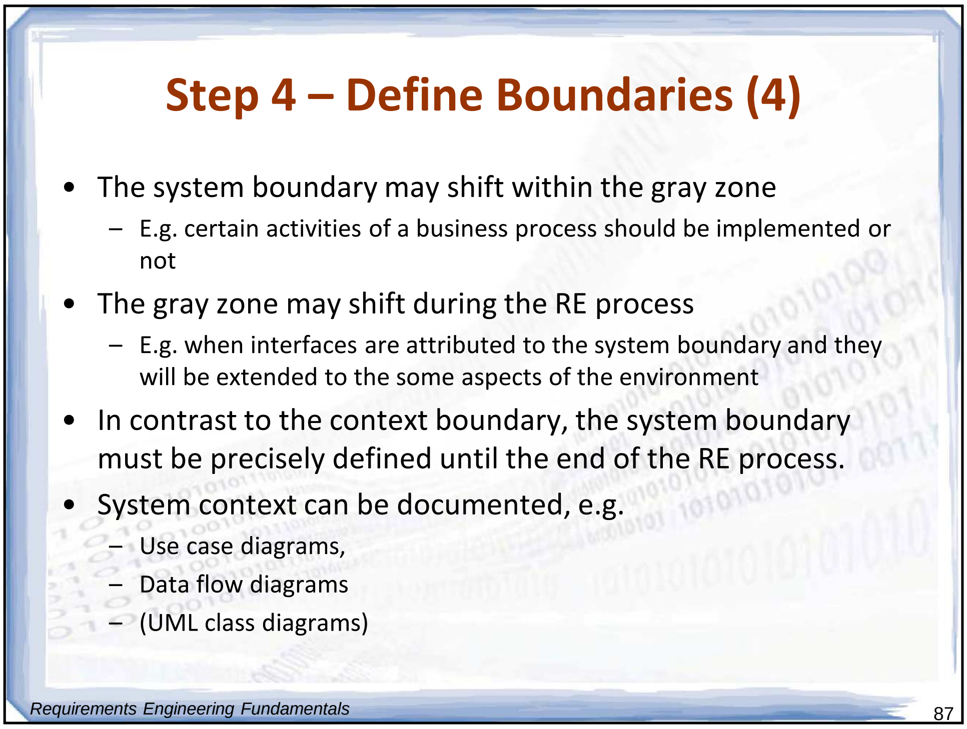
- The part of the reality (system environment) that is relevant for the req's of the system is the **System Context**.



Step 4 – Define Boundaries (3)

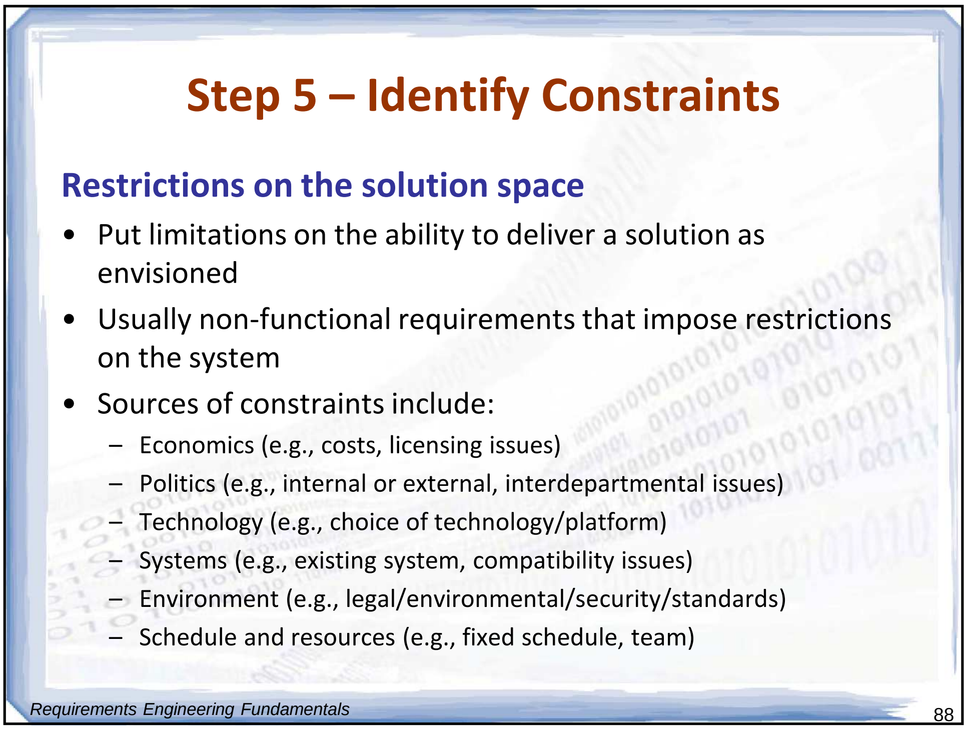
- Sources (provide input) and sinks (receive output) can be used to identify the **interfaces** of the system. There is a „grey zone" between system and system context.





Step 4 – Define Boundaries (4)

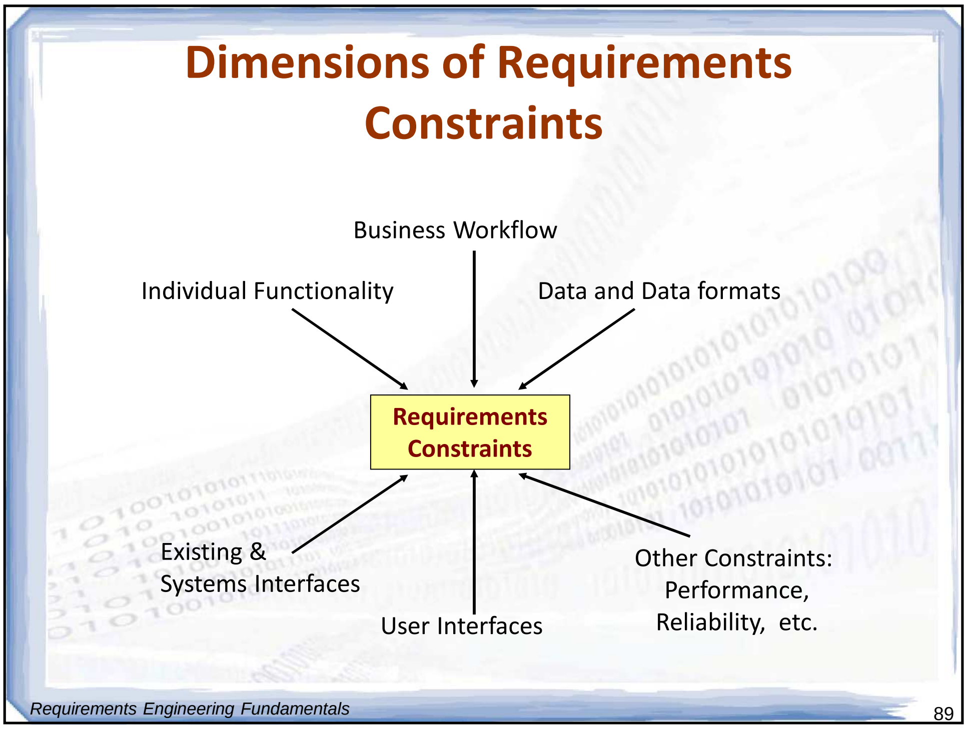
- The system boundary may shift within the gray zone
  - E.g. certain activities of a business process should be implemented or not
- The gray zone may shift during the RE process
  - E.g. when interfaces are attributed to the system boundary and they will be extended to the some aspects of the environment
- In contrast to the context boundary, the system boundary must be precisely defined until the end of the RE process.
- System context can be documented, e.g.
  - Use case diagrams,
  - Data flow diagrams
  - (UML class diagrams)



Step 5 – Identify Constraints

Restrictions on the solution space

- Put limitations on the ability to deliver a solution as envisioned
- Usually non-functional requirements that impose restrictions on the system
- Sources of constraints include:
  - Economics (e.g., costs, licensing issues)
  - Politics (e.g., internal or external, interdepartmental issues)
  - Technology (e.g., choice of technology/platform)
  - Systems (e.g., existing system, compatibility issues)
  - Environment (e.g., legal/environmental/security/standards)
  - Schedule and resources (e.g., fixed schedule, team)



Dimensions of Requirements Constraints

Business Workflow

Individual Functionality

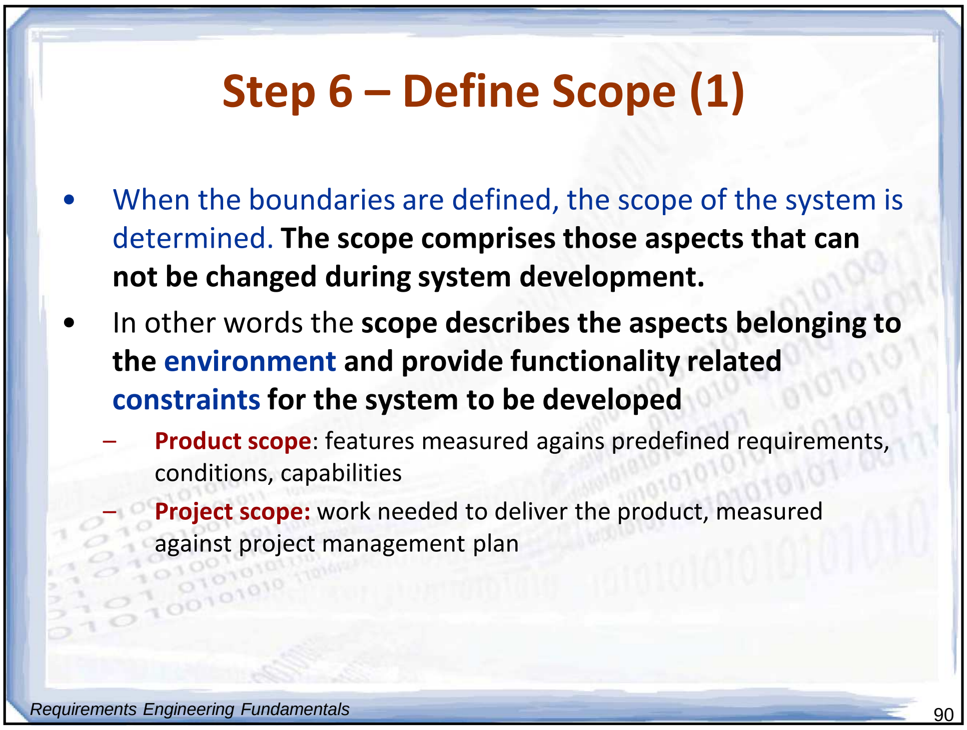
Data and Data formats

Requirements Constraints

Existing &
Systems Interfaces

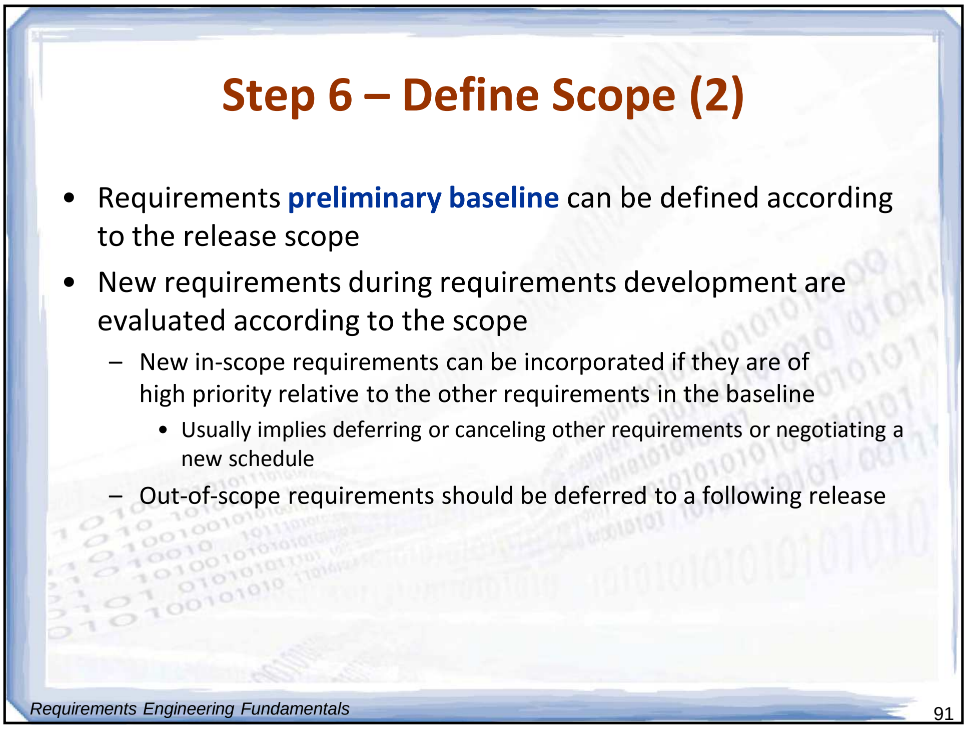
User Interfaces

Other Constraints:
Performance,
Reliability, etc.



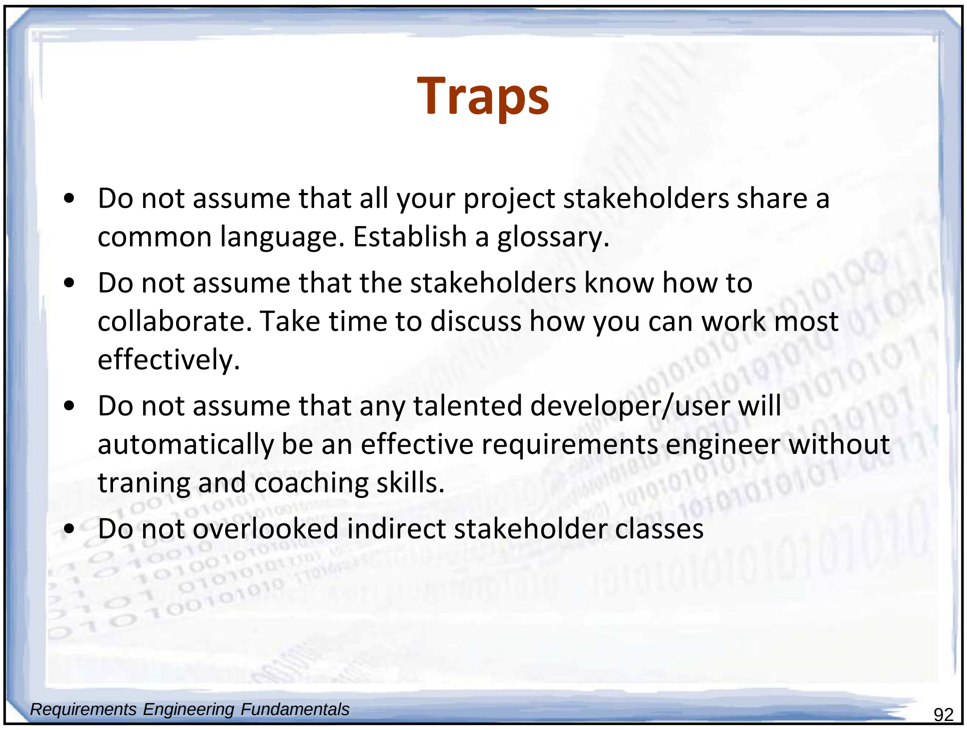
Step 6 – Define Scope (1)

- When the boundaries are defined, the scope of the system is determined. **The scope comprises those aspects that can not be changed during system development.**
- In other words the **scope describes the aspects belonging to the environment and provide functionality related constraints for the system to be developed**
  - **Product scope**: features measured agains predefined requirements, conditions, capabilities
  - **Project scope:** work needed to deliver the product, measured against project management plan



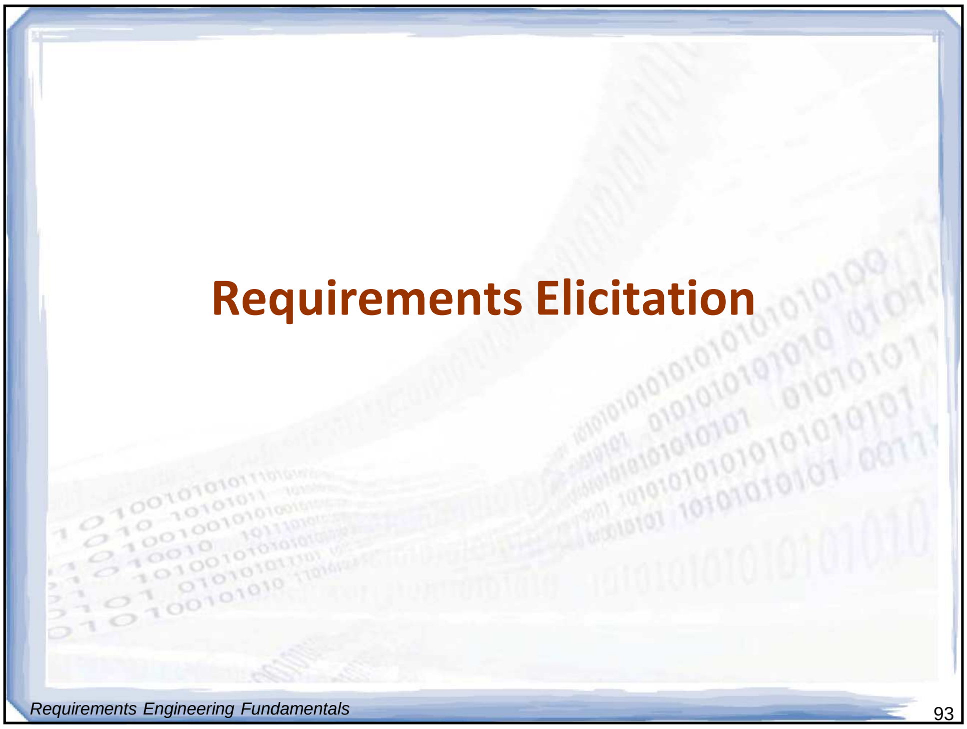
Step 6 – Define Scope (2)

- Requirements **preliminary baseline** can be defined according to the release scope
- New requirements during requirements development are evaluated according to the scope
  - New in-scope requirements can be incorporated if they are of high priority relative to the other requirements in the baseline
    - Usually implies deferring or canceling other requirements or negotiating a new schedule
  - Out-of-scope requirements should be deferred to a following release

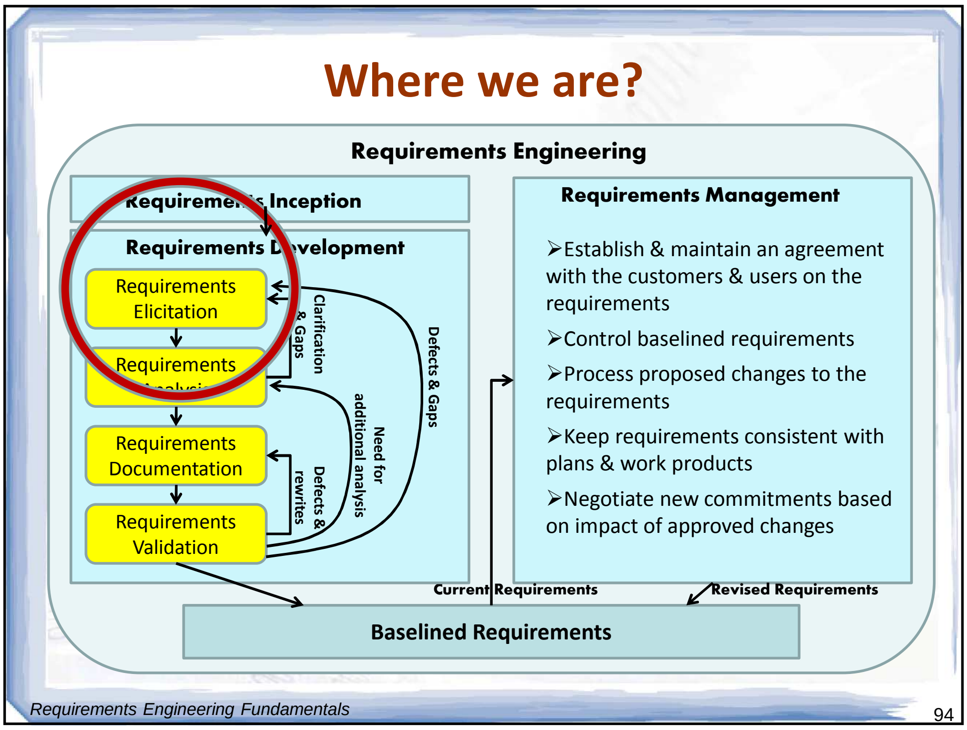


Traps

- Do not assume that all your project stakeholders share a common language. Establish a glossary.
- Do not assume that the stakeholders know how to collaborate. Take time to discuss how you can work most effectively.
- Do not assume that any talented developer/user will automatically be an effective requirements engineer without traning and coaching skills.
- Do not overlooked indirect stakeholder classes

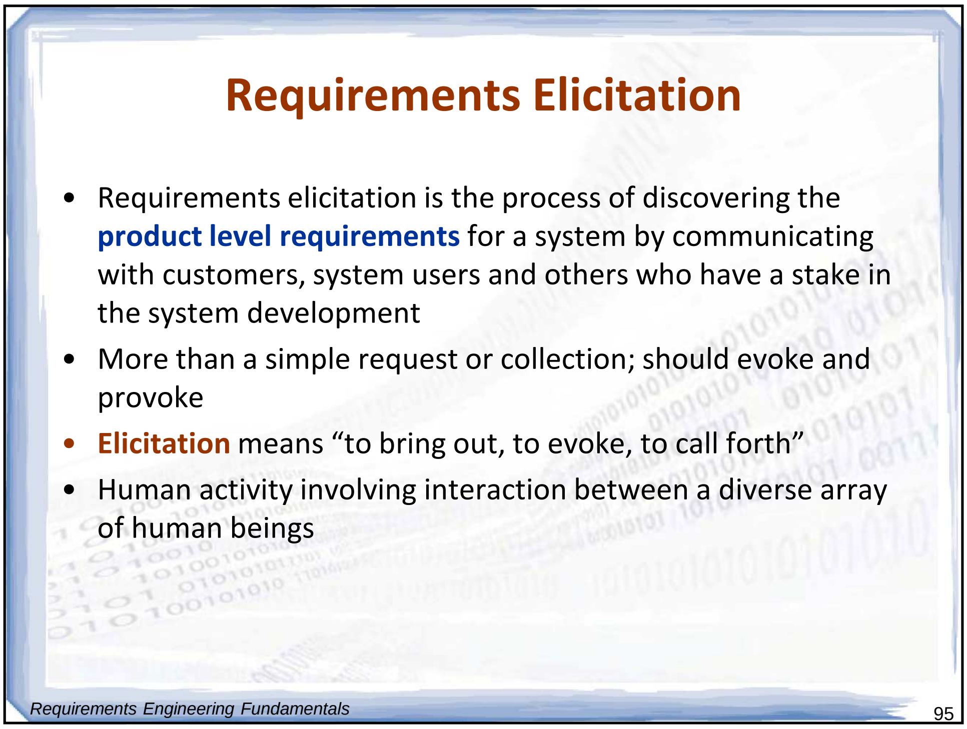


Requirements Elicitation



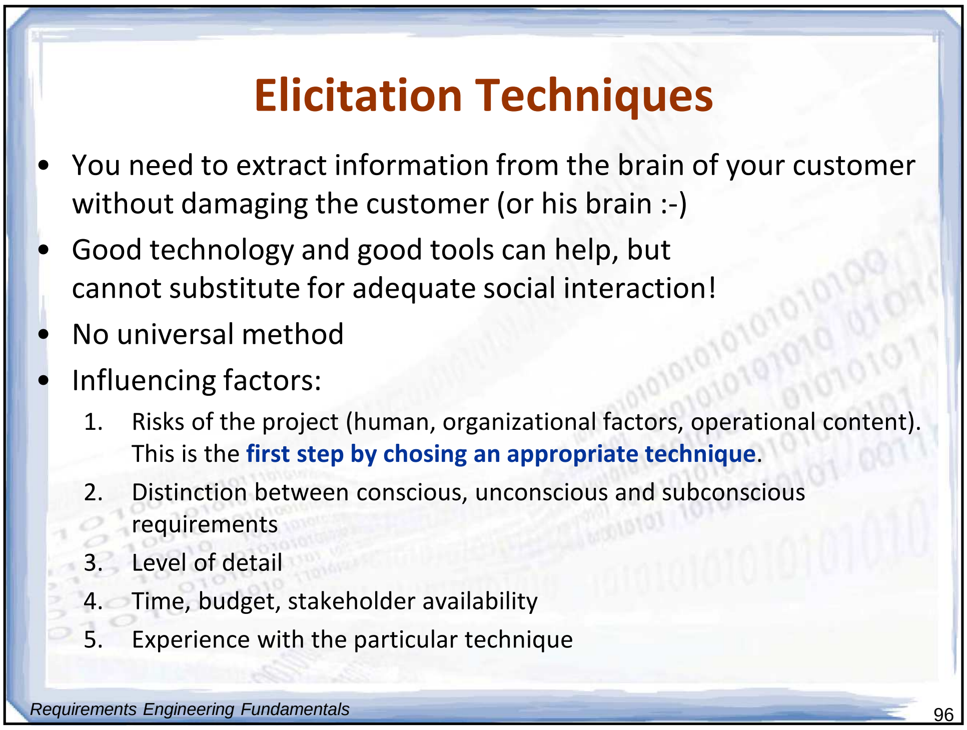
Where we are?

- ➢Establish & maintain an agreement with the customers & users on the requirements
- ➢Control baselined requirements
- ➢Process proposed changes to the requirements
- ➢Keep requirements consistent with plans & work products
- ➢Negotiate new commitments based on impact of approved changes



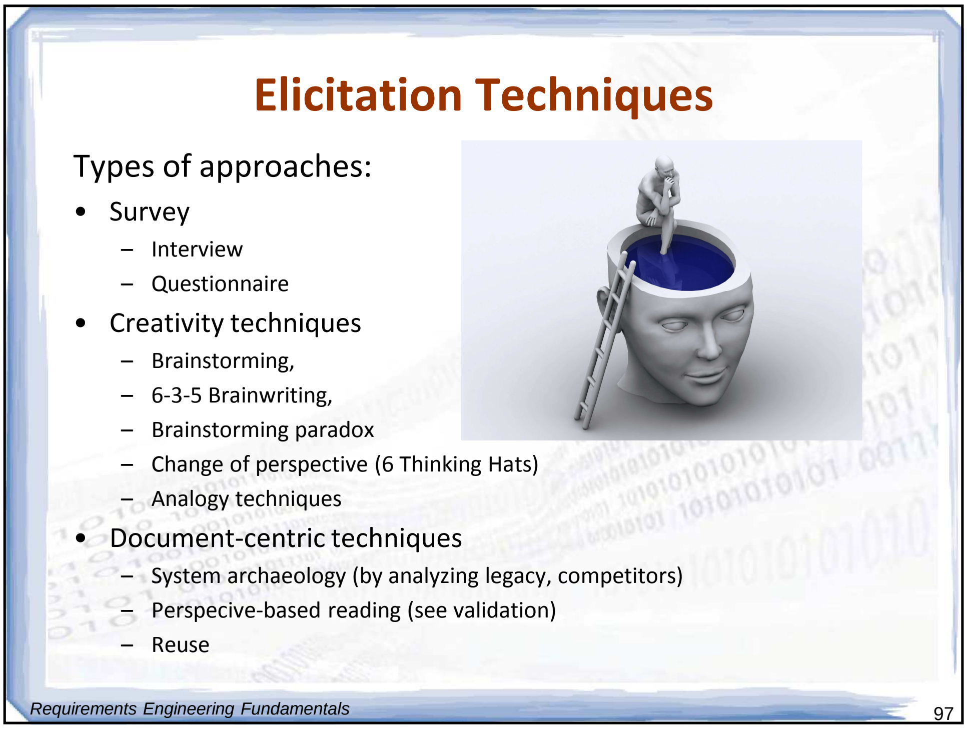
Requirements Elicitation

- Requirements elicitation is the process of discovering the **product level requirements** for a system by communicating with customers, system users and others who have a stake in the system development
- More than a simple request or collection; should evoke and provoke
- **Elicitation** means “to bring out, to evoke, to call forth”
- Human activity involving interaction between a diverse array of human beings



Elicitation Techniques

- You need to extract information from the brain of your customer without damaging the customer (or his brain :-)
- Good technology and good tools can help, but cannot substitute for adequate social interaction!
- No universal method
- Influencing factors:
  1. Risks of the project (human, organizational factors, operational content). This is the **first step by chosing an appropriate technique**.
  2. Distinction between conscious, unconscious and subconscious requirements
  3. Level of detail
  4. Time, budget, stakeholder availability
  5. Experience with the particular technique

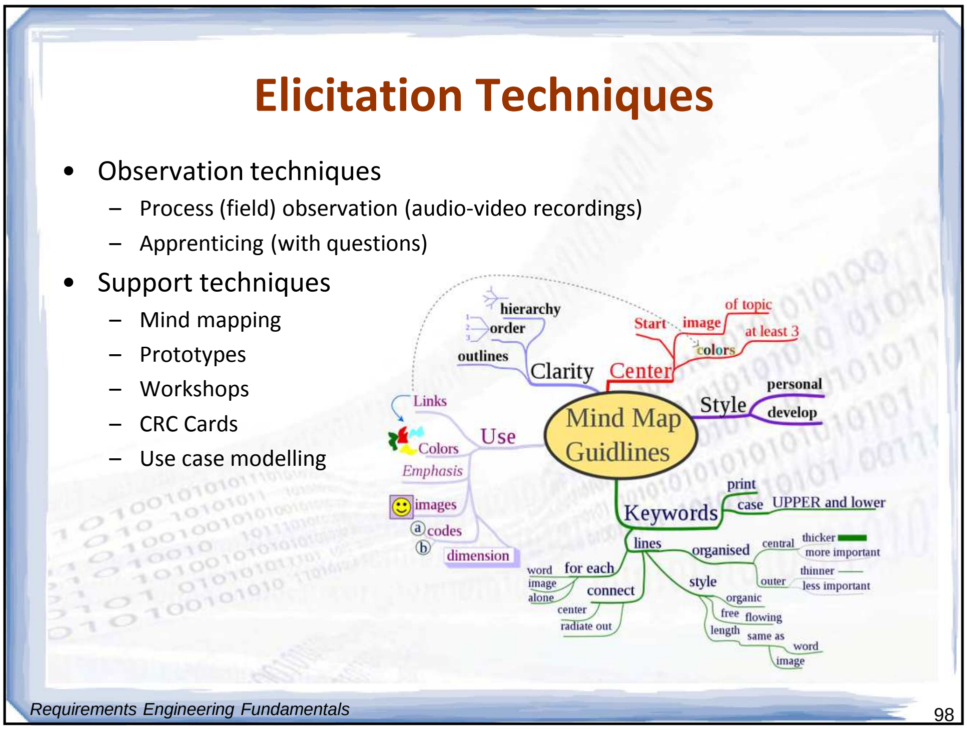


Elicitation Techniques

Types of approaches:

- Survey
  - Interview
  - Questionnaire
- Creativity techniques
  - Brainstorming,
  - 6-3-5 Brainwriting,
  - Brainstorming paradox
  - Change of perspective (6 Thinking Hats)
  - Analogy techniques
- Document-centric techniques
  - System archaeology (by analyzing legacy, competitors)
  - Perspecive-based reading (see validation)
  - Reuse

97



Elicitation Techniques

- Observation techniques
  - Process (field) observation (audio-video recordings)
  - Apprenticing (with questions)
- Support techniques
  - Mind mapping
  - Prototypes
  - Workshops
  - CRC Cards
  - Use case modelling

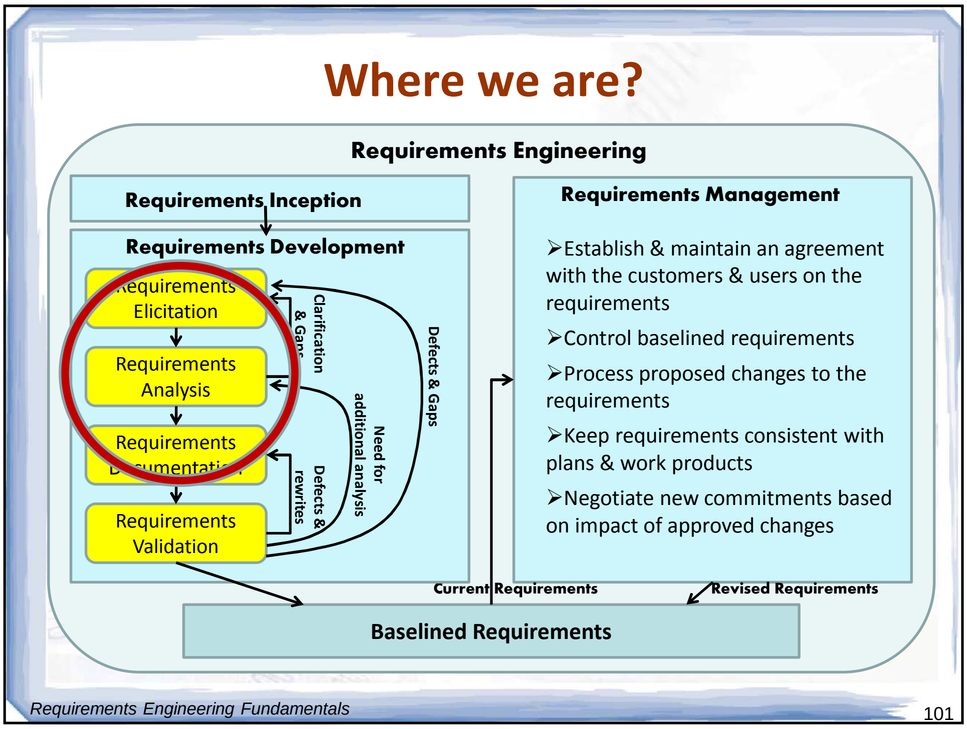
Mind Map
Guidlines
Clarity
hierarchy
order
outlines
Center
Start
image
of topic
colors
at least 3
Style
personal
develop
Use
Links
Colors
Emphasis
images
codes
dimension
Keywords
print
case
UPPER and lower
lines
organised
central
thicker
more important
outer
thinner
less important
style
organic
free
flowing
length
same as
word
image
for each
word
image
alone
connect
center
radiate out

98

Requirements Elicitation Guideline

- Assess System Feasibility
- Be sensitive to organizational and political considerations
- Record requirements sources
- Identify and consult system stakeholders
- Use appropriate techniques to elicit requirements
- Collect requirements from multiple viewpoints
- Prototype poorly understood requirements
- Look for domain constraints
- Reuse requirements

Analysis and Negotiation



Where we are?

Requirements Engineering

Requirements Inception

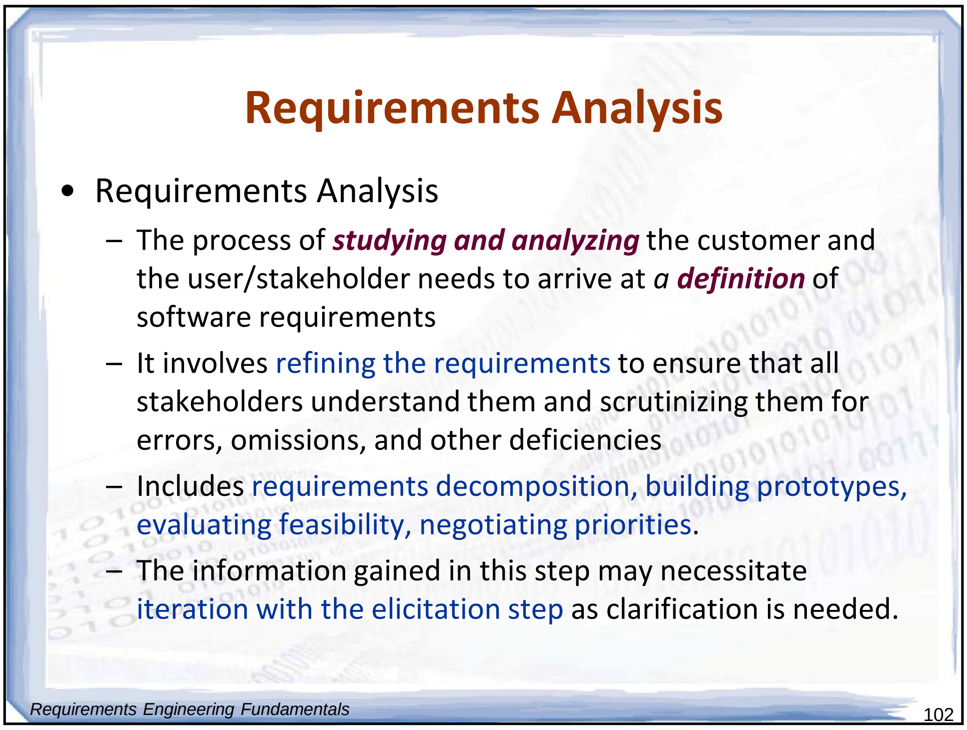
Requirements Development

Requirements Management

- Establish & maintain an agreement with the customers & users on the requirements
- Control baselined requirements
- Process proposed changes to the requirements
- Keep requirements consistent with plans & work products
- Negotiate new commitments based on impact of approved changes

Baselined Requirements

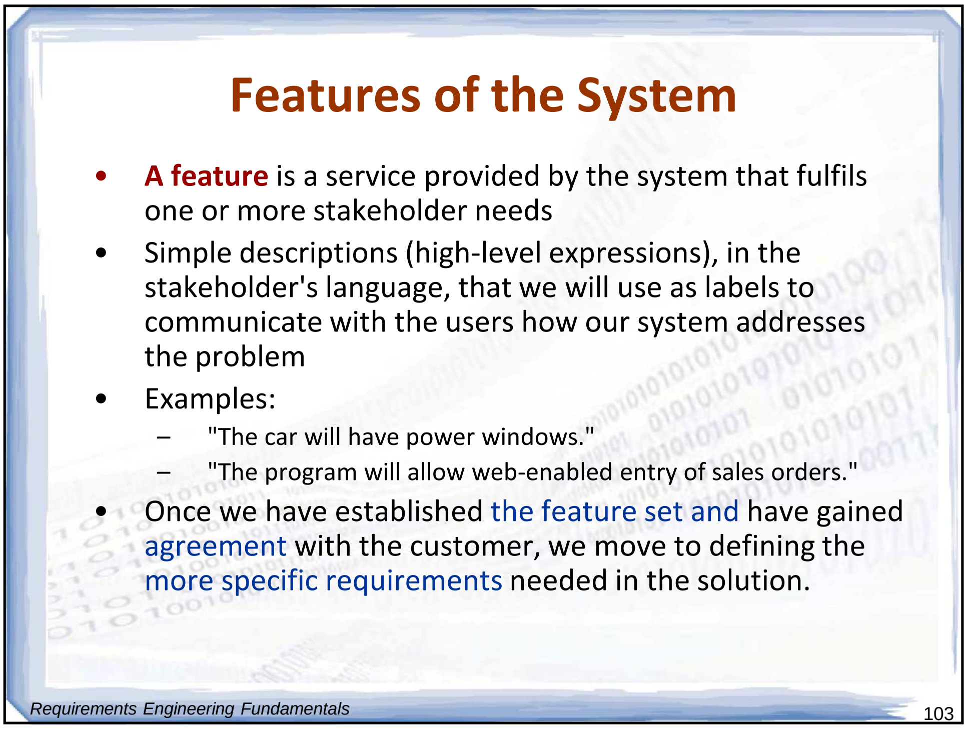
Requirements Engineering Fundamentals 101



Requirements Analysis

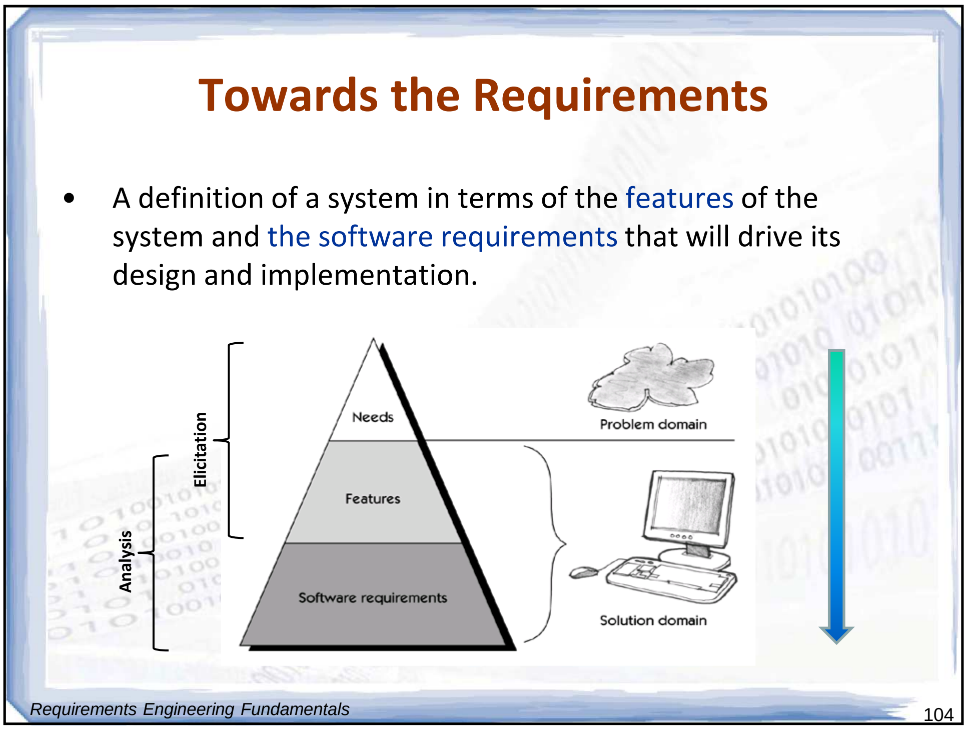
- Requirements Analysis
  - The process of ***studying and analyzing*** the customer and the user/stakeholder needs to arrive at *a* ***definition*** of software requirements
  - It involves refining the requirements to ensure that all stakeholders understand them and scrutinizing them for errors, omissions, and other deficiencies
  - Includes requirements decomposition, building prototypes, evaluating feasibility, negotiating priorities.
  - The information gained in this step may necessitate iteration with the elicitation step as clarification is needed.

Requirements Engineering Fundamentals 102



Features of the System

- **A feature** is a service provided by the system that fulfils one or more stakeholder needs
- Simple descriptions (high-level expressions), in the stakeholder's language, that we will use as labels to communicate with the users how our system addresses the problem
- Examples:
  - "The car will have power windows."
  - "The program will allow web-enabled entry of sales orders."
- Once we have established the feature set and have gained agreement with the customer, we move to defining the more specific requirements needed in the solution.



Towards the Requirements

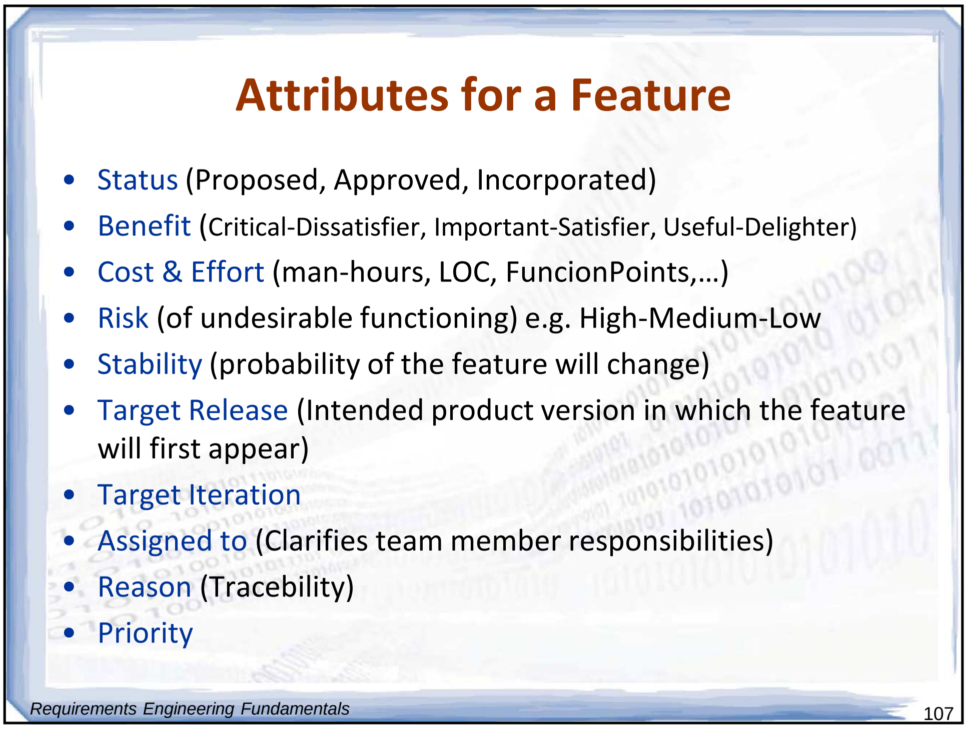
- A definition of a system in terms of the features of the system and the software requirements that will drive its design and implementation.

Managing complexity (1)

- By picking the level of abstraction which depends on the number of features
- Recommendation:
  - for any new system or an increment to an existing one, the number of **features should be between 25-99.**
  - Although, **fewer than 50 is preferred**.
  - Later on, these features will be refined to get the software requirements.

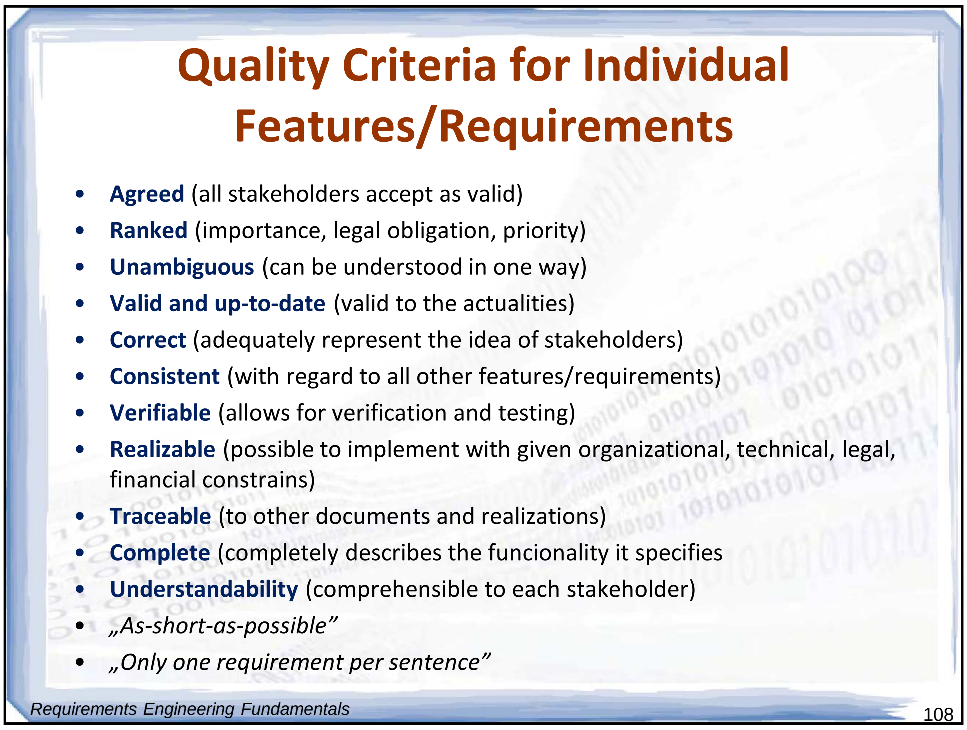
Managing complexity (2)

- In this way, the information will be
  - Small and manageable
  - Comprehensive and complete for
    - Product definition, communication with stakeholders,
    - Scope management and Project management
- Decision can be made for each feature to either
  - Postpone to a later release,
  - Implement immediately,
  - Reject entirely, or
  - Investigate further



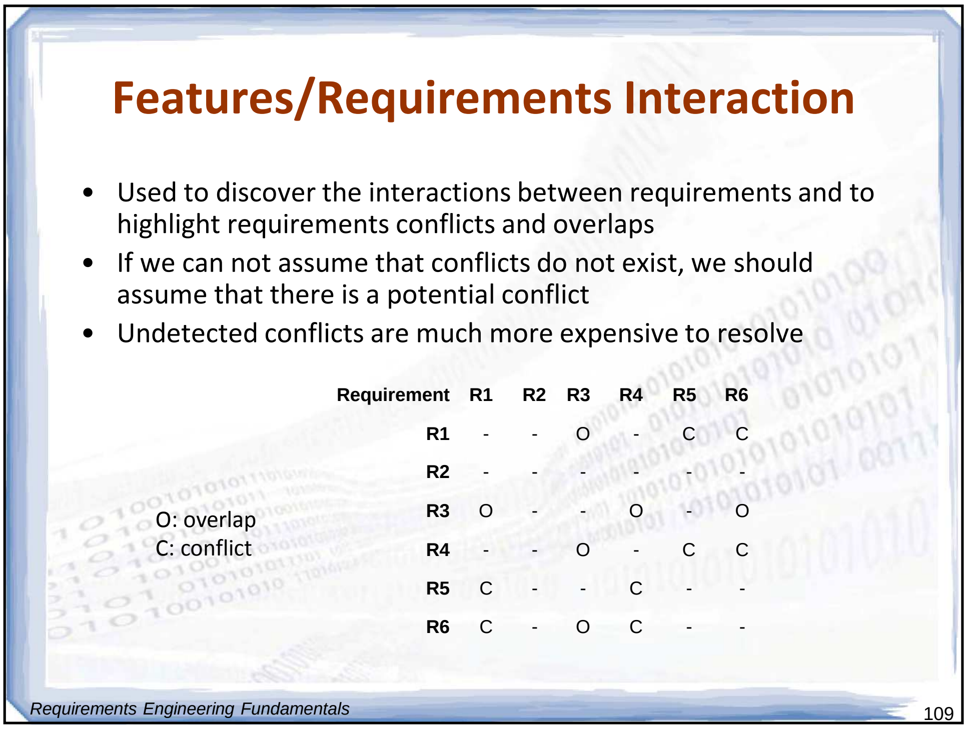
Attributes for a Feature

- Status (Proposed, Approved, Incorporated)
- Benefit (Critical-Dissatisfier, Important-Satisfier, Useful-Delighter)
- Cost & Effort (man-hours, LOC, FuncionPoints,…)
- Risk (of undesirable functioning) e.g. High-Medium-Low
- Stability (probability of the feature will change)
- Target Release (Intended product version in which the feature will first appear)
- Target Iteration
- Assigned to (Clarifies team member responsibilities)
- Reason (Tracebility)
- Priority



Quality Criteria for Individual Features/Requirements

- **Agreed** (all stakeholders accept as valid)
- **Ranked** (importance, legal obligation, priority)
- **Unambiguous** (can be understood in one way)
- **Valid and up-to-date** (valid to the actualities)
- **Correct** (adequately represent the idea of stakeholders)
- **Consistent** (with regard to all other features/requirements)
- **Verifiable** (allows for verification and testing)
- **Realizable** (possible to implement with given organizational, technical, legal, financial constrains)
- **Traceable** (to other documents and realizations)
- **Complete** (completely describes the funcionality it specifies
- **Understandability** (comprehensible to each stakeholder)
- *„As-short-as-possible”*
- *„Only one requirement per sentence”*

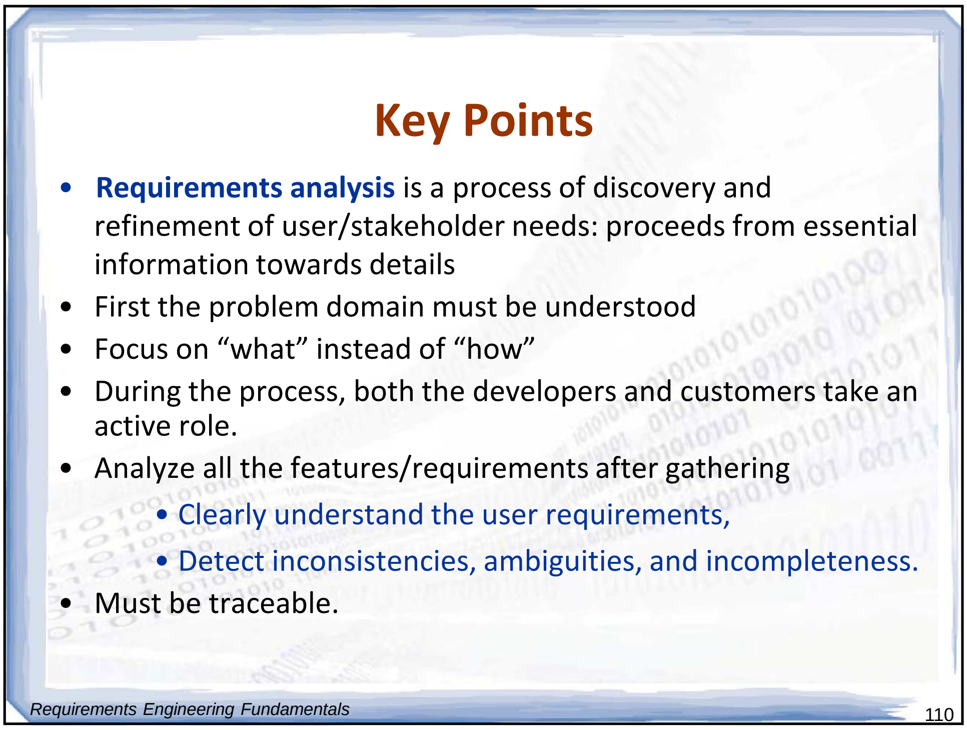


Features/Requirements Interaction

- Used to discover the interactions between requirements and to highlight requirements conflicts and overlaps
- If we can not assume that conflicts do not exist, we should assume that there is a potential conflict
- Undetected conflicts are much more expensive to resolve

O: overlap
C: conflict

Requirement	R1	R2	R3	R4	R5	R6
R1	-	-	O	-	C	C
R2	-	-	-	-	-	-
R3	O	-	-	O	-	O
R4	-	-	O	-	C	C
R5	C	-	-	C	-	-
R6	C	-	O	C	-	-

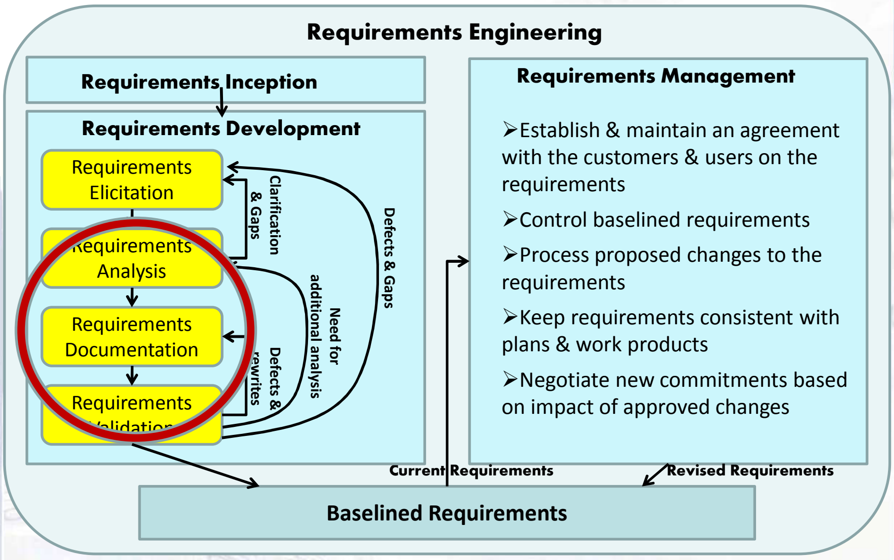


Key Points

- **Requirements analysis** is a process of discovery and refinement of user/stakeholder needs: proceeds from essential information towards details
- First the problem domain must be understood
- Focus on “what” instead of “how”
- During the process, both the developers and customers take an active role.
- Analyze all the features/requirements after gathering
    - Clearly understand the user requirements,
    - Detect inconsistencies, ambiguities, and incompleteness.
- Must be traceable.

Requirements Documentation

Where we are?



Requirements Engineering

Requirements Inception

Requirements Development

Requirements Elicitation

Requirements Analysis

Requirements Documentation

Requirements Validation

Clarification & Gaps

Defects & rewrites

Need for additional analysis

Defects & Gaps

Requirements Management

- ➢Establish & maintain an agreement with the customers & users on the requirements
- ➢Control baselined requirements
- ➢Process proposed changes to the requirements
- ➢Keep requirements consistent with plans & work products
- ➢Negotiate new commitments based on impact of approved changes

Current Requirements

Revised Requirements

Baselined Requirements

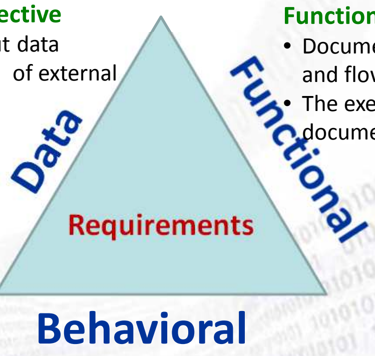
Documenting Requirements

- A requirement specification (documentation) is a systematically represented collection of requirements, typically for a system or component, that satisfies given criteria.
- Reasons for documenting requirements:
  - Requirements are the basis for system development
  - Requirements have a legal relevance
  - Requirements documents are complex
  - Requirements must be accessible to all involved parties
- Requirements should be documented in a way that they meet the quality demands of all involved

Perspectives of Requirements

Static-Structural Perspective

- Documents input-output data
- Static Data Dependency of external system services

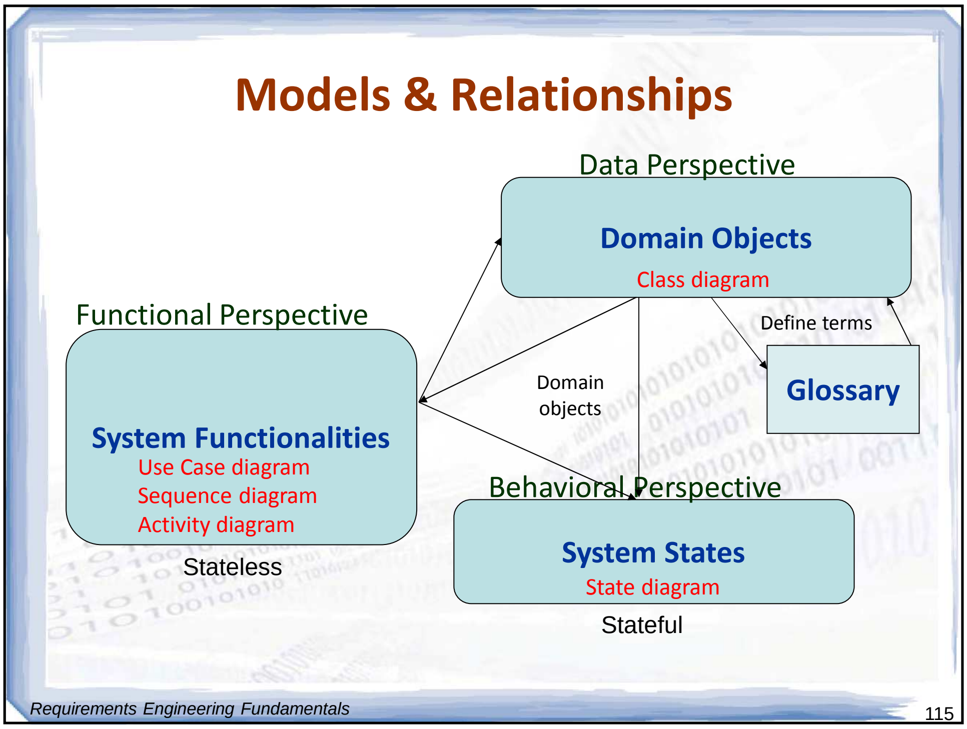


Functional Perspective

- Documents the data processing and flows
- The execution order is also documented

Behavioral Perspective

- Documents the reactions of the system upon Events,
- Conditions and Effects



Models & Relationships

Data Perspective

Domain Objects

Class diagram

Functional Perspective

System Functionalities

Use Case diagram

Sequence diagram

Activity diagram

Define terms

Glossary

Domain objects

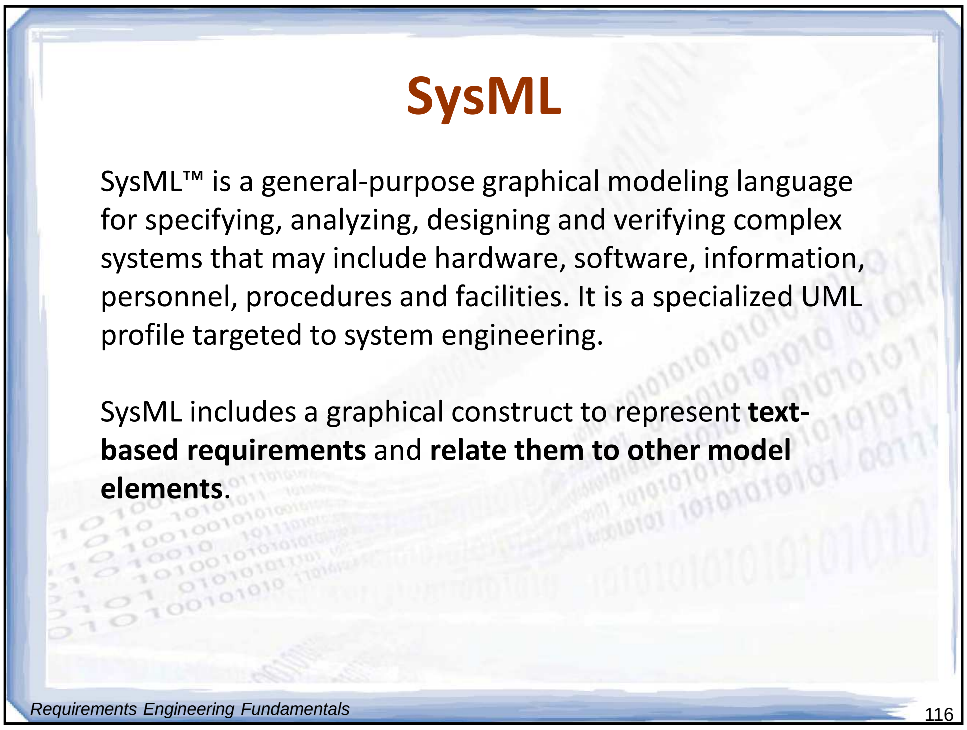
Behavioral Perspective

System States

State diagram

Stateless

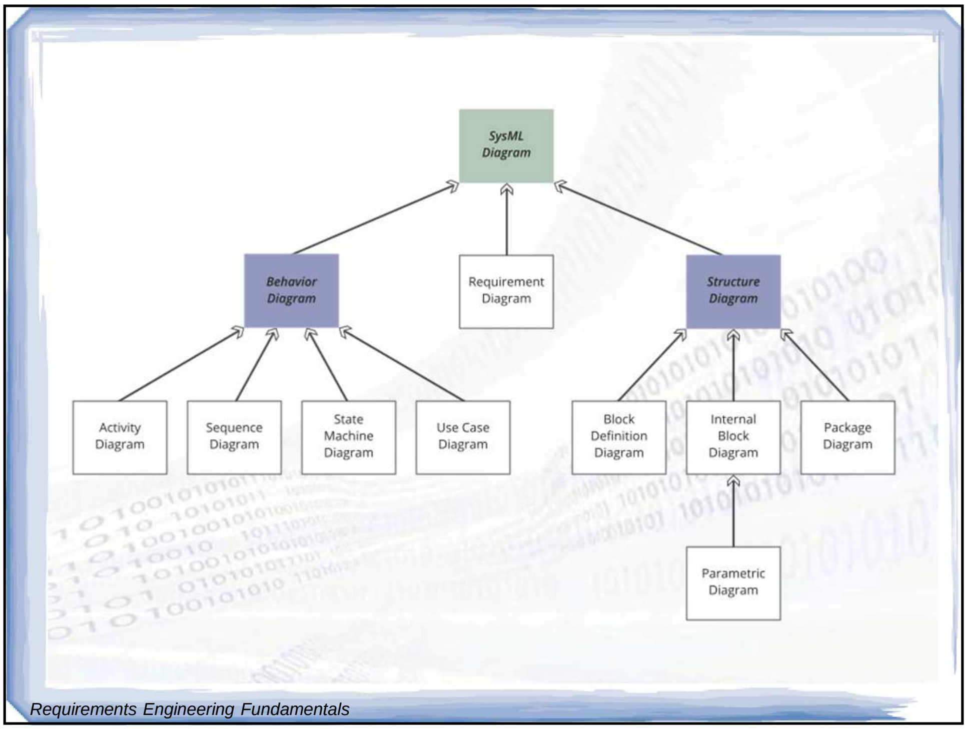
Stateful



SysML

SysML™ is a general-purpose graphical modeling language for specifying, analyzing, designing and verifying complex systems that may include hardware, software, information, personnel, procedures and facilities. It is a specialized UML profile targeted to system engineering.

SysML includes a graphical construct to represent **text-based requirements** and **relate them to other model elements**.



SysML Diagram

Behavior Diagram

Requirement Diagram

Structure Diagram

Activity Diagram

Sequence Diagram

State Machine Diagram

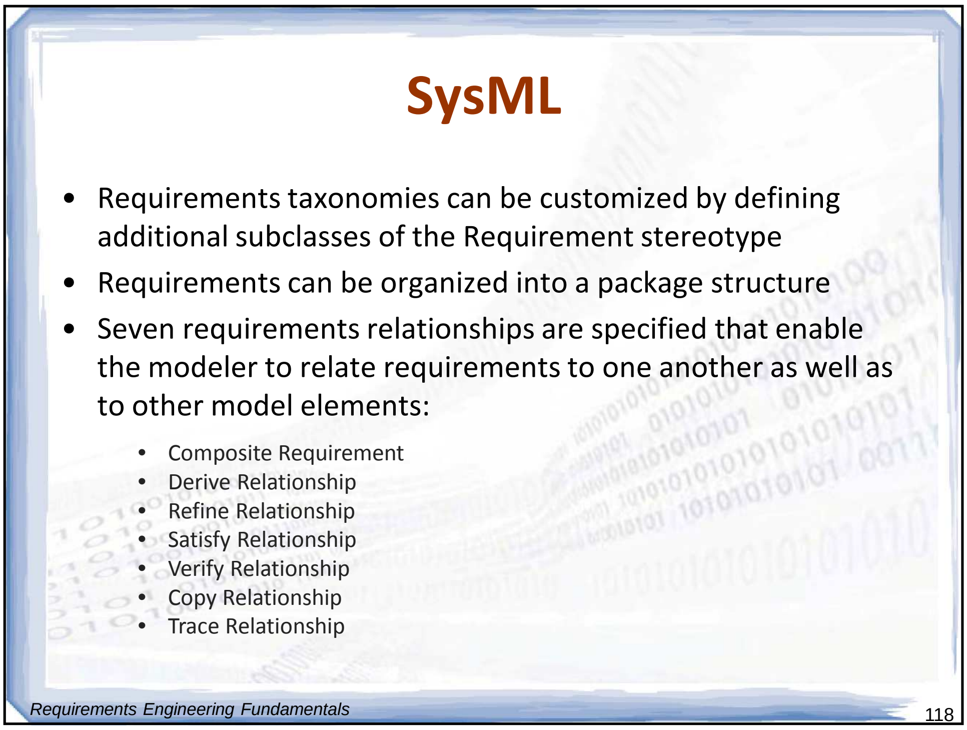
Use Case Diagram

Block Definition Diagram

Internal Block Diagram

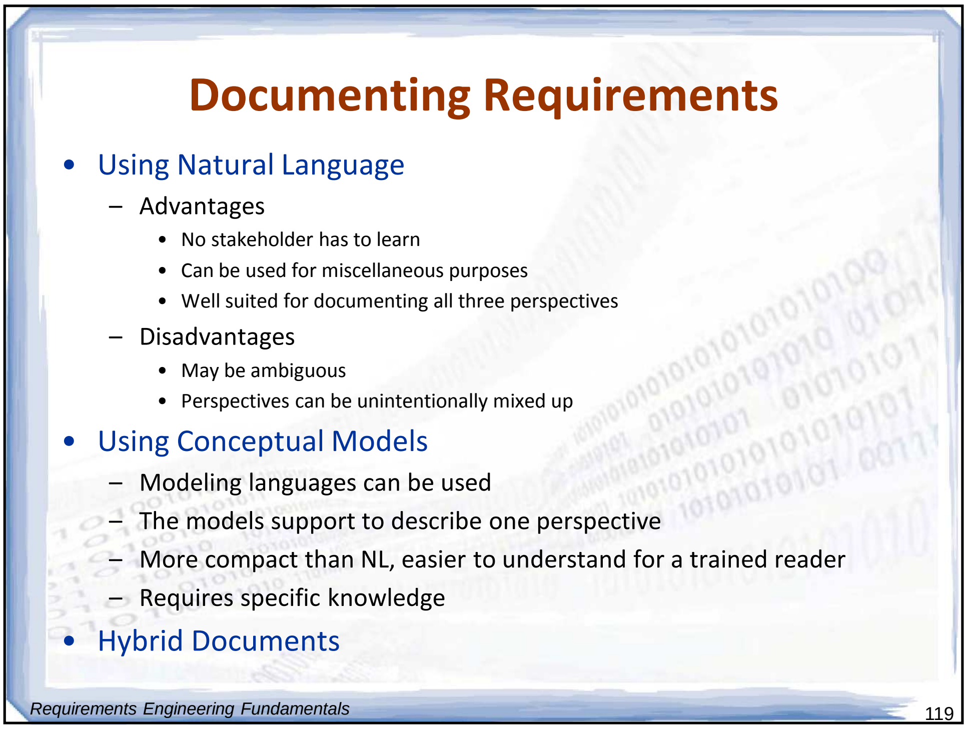
Package Diagram

Parametric Diagram



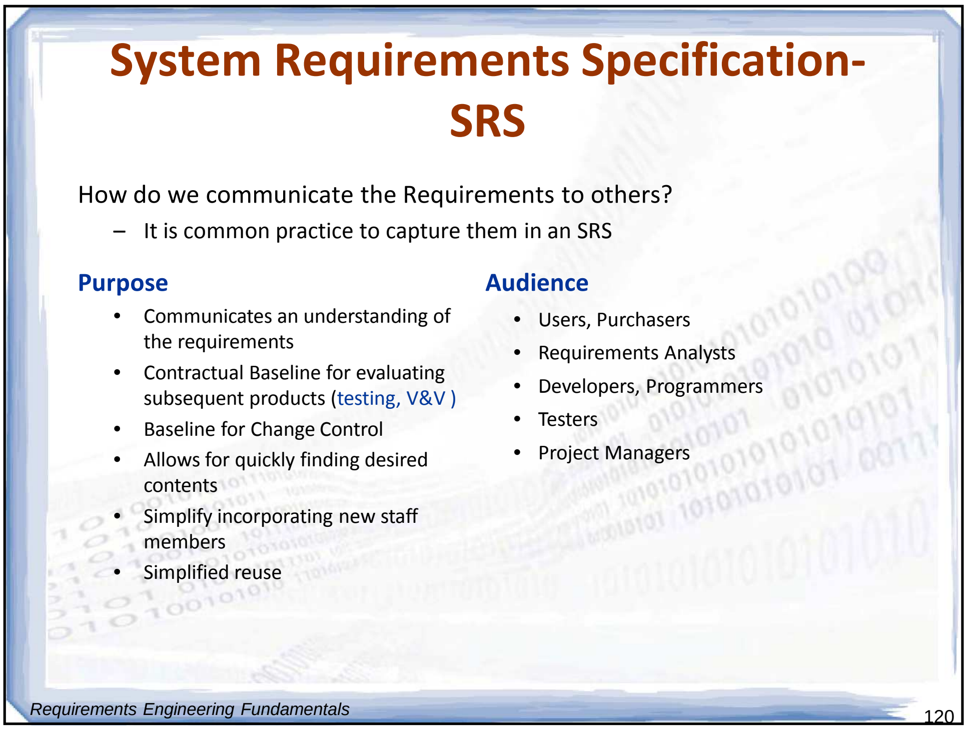
SysML

- Requirements taxonomies can be customized by defining additional subclasses of the Requirement stereotype
- Requirements can be organized into a package structure
- Seven requirements relationships are specified that enable the modeler to relate requirements to one another as well as to other model elements:
  - Composite Requirement
  - Derive Relationship
  - Refine Relationship
  - Satisfy Relationship
  - Verify Relationship
  - Copy Relationship
  - Trace Relationship



Documenting Requirements

- Using Natural Language
  - Advantages
    - No stakeholder has to learn
    - Can be used for miscellaneous purposes
    - Well suited for documenting all three perspectives
  - Disadvantages
    - May be ambiguous
    - Perspectives can be unintentionally mixed up
- Using Conceptual Models
  - Modeling languages can be used
  - The models support to describe one perspective
  - More compact than NL, easier to understand for a trained reader
  - Requires specific knowledge
- Hybrid Documents



System Requirements Specification-SRS

How do we communicate the Requirements to others?

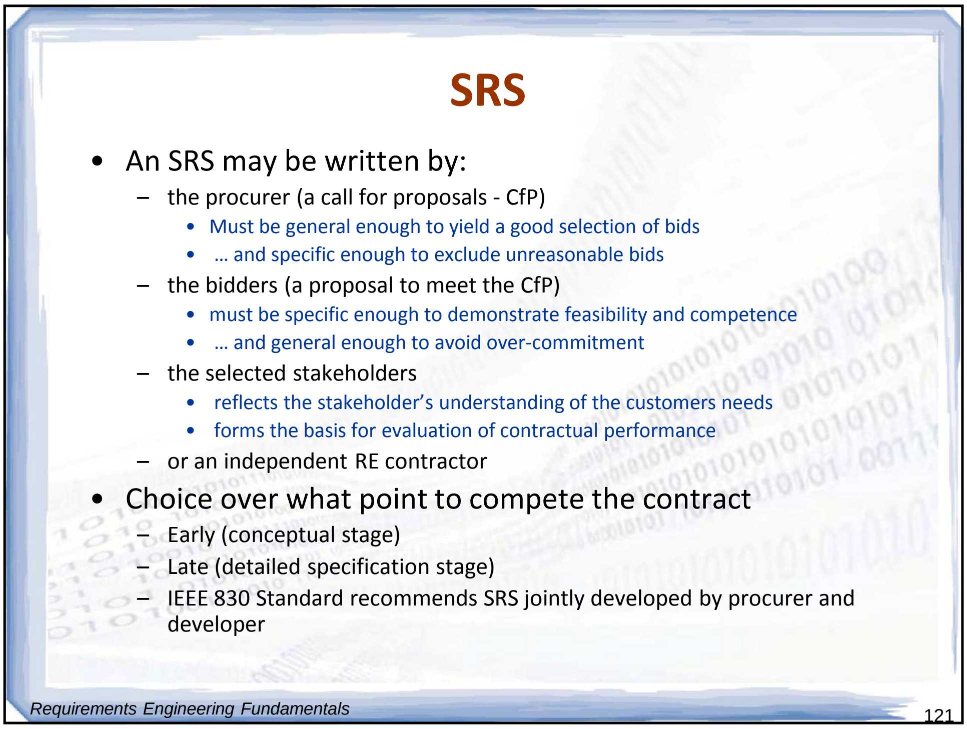
- It is common practice to capture them in an SRS

Purpose

- Communicates an understanding of the requirements
- Contractual Baseline for evaluating subsequent products (testing, V&V)
- Baseline for Change Control
- Allows for quickly finding desired contents
- Simplify incorporating new staff members
- Simplified reuse

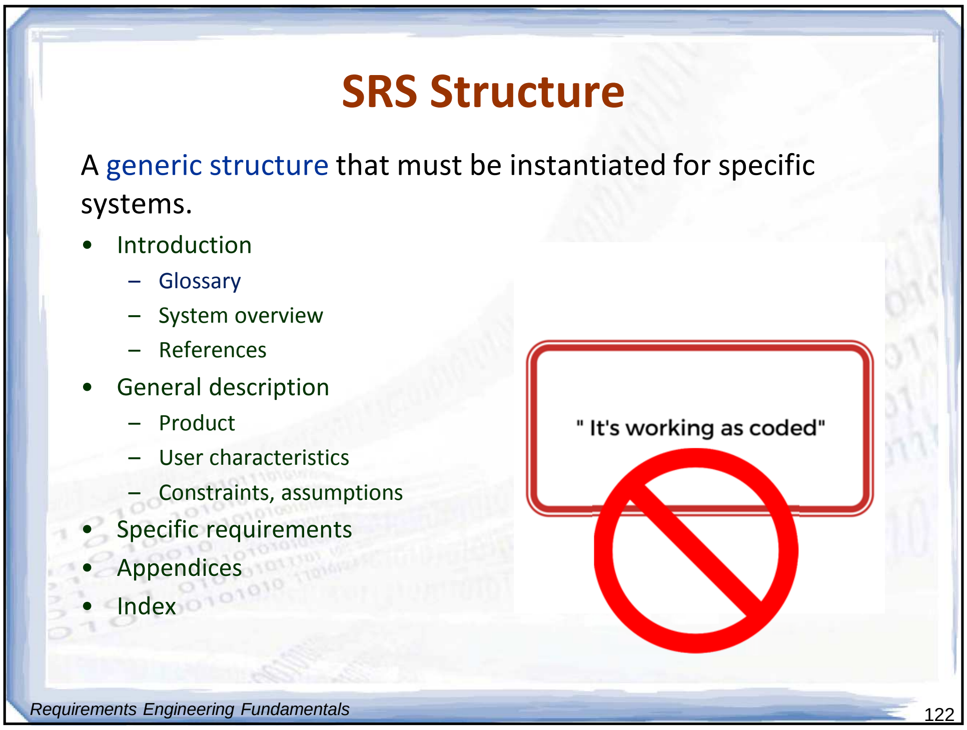
Audience

- Users, Purchasers
- Requirements Analysts
- Developers, Programmers
- Testers
- Project Managers



SRS

- An SRS may be written by:
  - the procurer (a call for proposals - CfP)
    - Must be general enough to yield a good selection of bids
    - … and specific enough to exclude unreasonable bids
  - the bidders (a proposal to meet the CfP)
    - must be specific enough to demonstrate feasibility and competence
    - … and general enough to avoid over-commitment
  - the selected stakeholders
    - reflects the stakeholder's understanding of the customers needs
    - forms the basis for evaluation of contractual performance
  - or an independent RE contractor
- Choice over what point to compete the contract
  - Early (conceptual stage)
  - Late (detailed specification stage)
  - IEEE 830 Standard recommends SRS jointly developed by procurer and developer

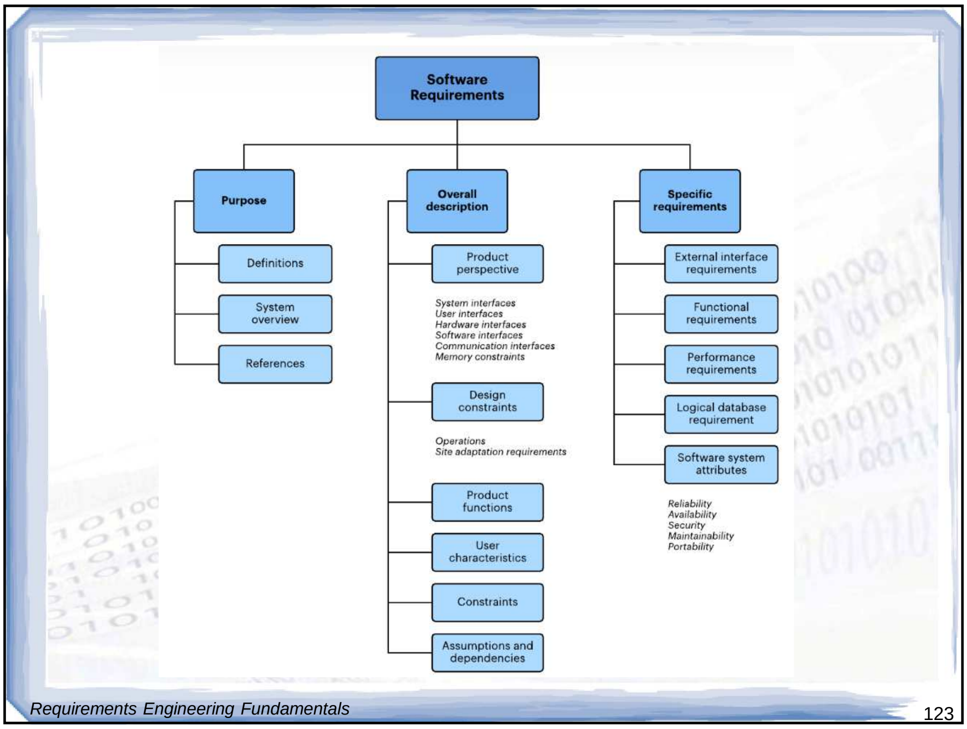


SRS Structure

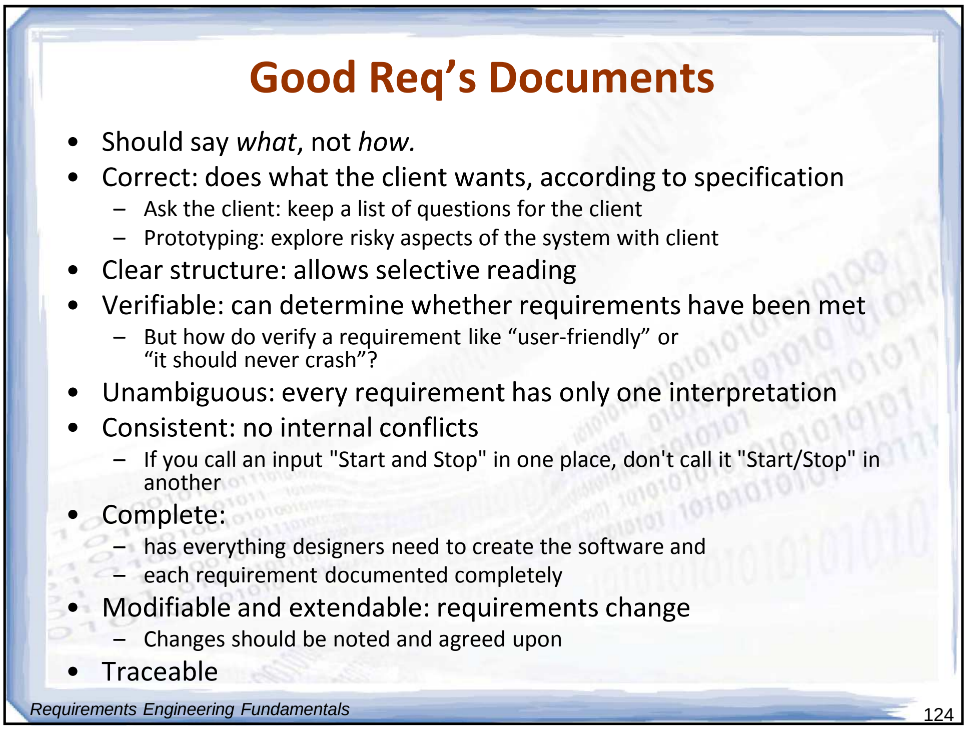
A generic structure that must be instantiated for specific systems.

- Introduction
  - Glossary
  - System overview
  - References
- General description
  - Product
  - User characteristics
  - Constraints, assumptions
- Specific requirements
- Appendices
- Index

" It's working as coded"

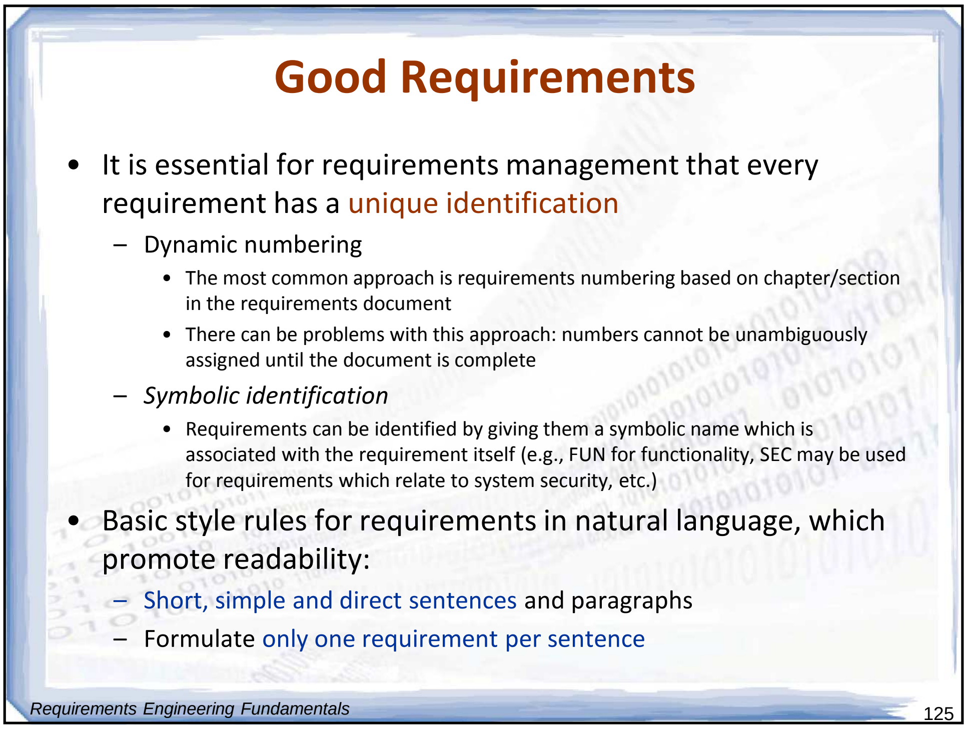


- **Software Requirements**
  - **Purpose**
    - Definitions
    - System overview
    - References
  - **Overall description**
    - Product perspective
      - *System interfaces*
      - *User interfaces*
      - *Hardware interfaces*
      - *Software interfaces*
      - *Communication interfaces*
      - *Memory constraints*
    - Design constraints
      - *Operations*
      - *Site adaptation requirements*
    - Product functions
    - User characteristics
    - Constraints
    - Assumptions and dependencies
  - **Specific requirements**
    - External interface requirements
    - Functional requirements
    - Performance requirements
    - Logical database requirement
    - Software system attributes
      - *Reliability*
      - *Availability*
      - *Security*
      - *Maintainability*
      - *Portability*



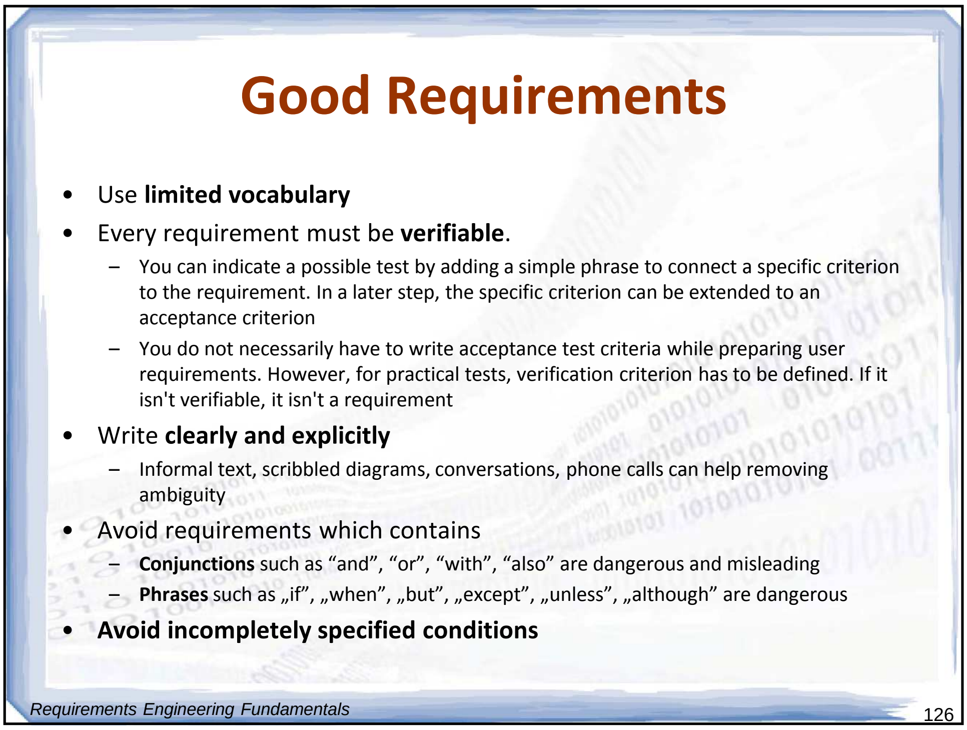
Good Req's Documents

- Should say *what*, not *how.*
- Correct: does what the client wants, according to specification
  - Ask the client: keep a list of questions for the client
  - Prototyping: explore risky aspects of the system with client
- Clear structure: allows selective reading
- Verifiable: can determine whether requirements have been met
  - But how do verify a requirement like "user-friendly" or "it should never crash"?
- Unambiguous: every requirement has only one interpretation
- Consistent: no internal conflicts
  - If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another
- Complete:
  - has everything designers need to create the software and
  - each requirement documented completely
- Modifiable and extendable: requirements change
  - Changes should be noted and agreed upon
- Traceable



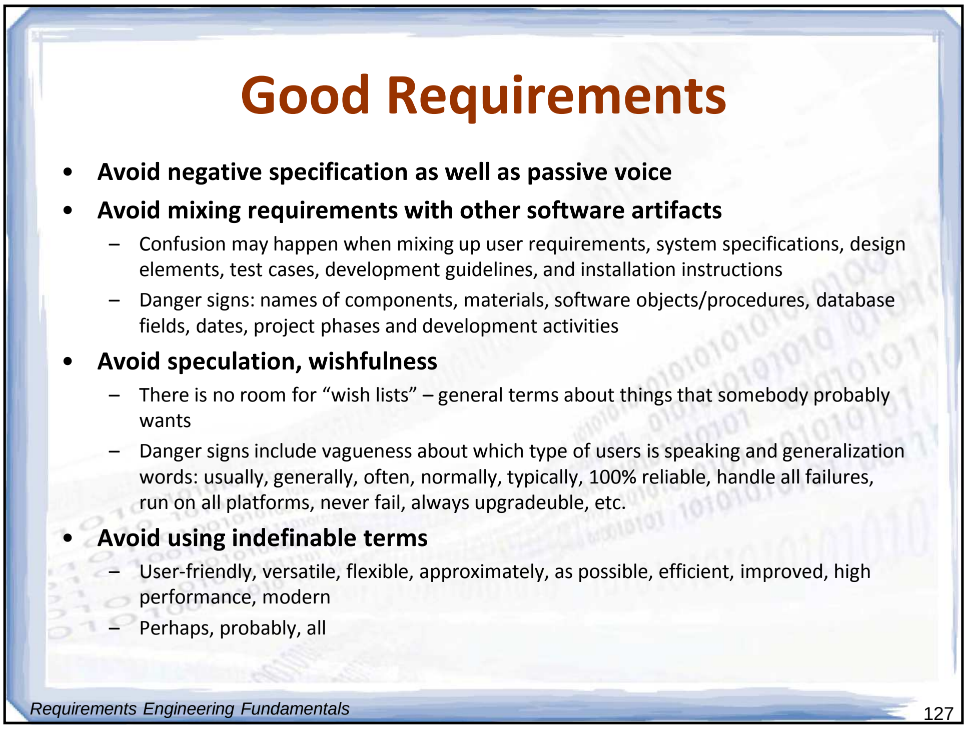
Good Requirements

- It is essential for requirements management that every requirement has a unique identification
  - Dynamic numbering
    - The most common approach is requirements numbering based on chapter/section in the requirements document
    - There can be problems with this approach: numbers cannot be unambiguously assigned until the document is complete
  - *Symbolic identification*
    - Requirements can be identified by giving them a symbolic name which is associated with the requirement itself (e.g., FUN for functionality, SEC may be used for requirements which relate to system security, etc.)
- Basic style rules for requirements in natural language, which promote readability:
  - Short, simple and direct sentences and paragraphs
  - Formulate only one requirement per sentence



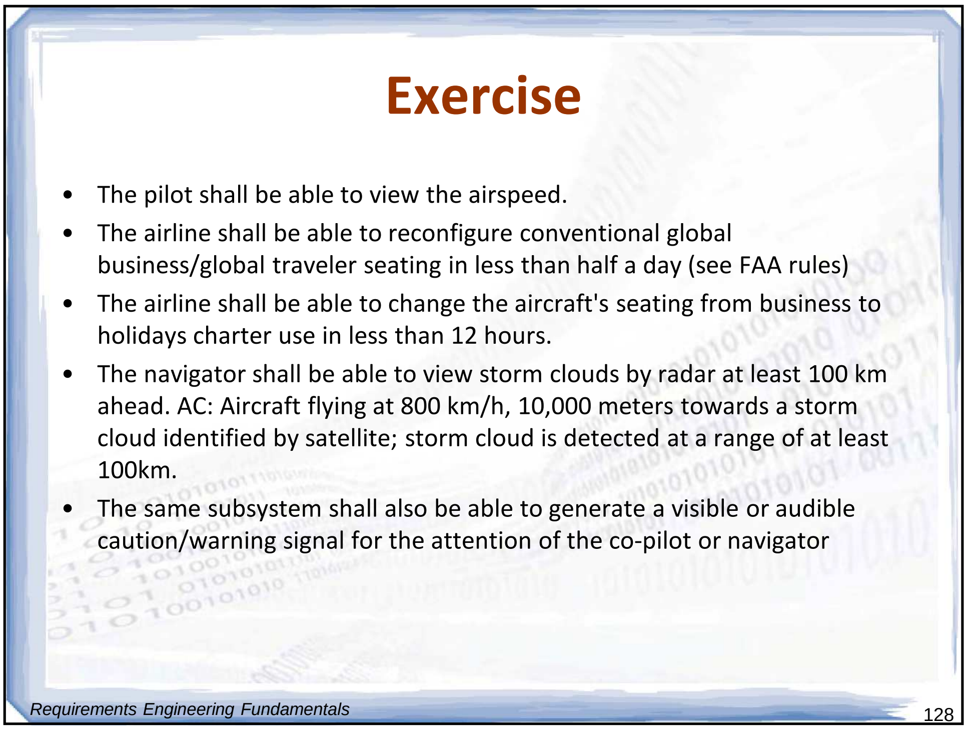
Good Requirements

- Use **limited vocabulary**
- Every requirement must be **verifiable**.
  - You can indicate a possible test by adding a simple phrase to connect a specific criterion to the requirement. In a later step, the specific criterion can be extended to an acceptance criterion
  - You do not necessarily have to write acceptance test criteria while preparing user requirements. However, for practical tests, verification criterion has to be defined. If it isn't verifiable, it isn't a requirement
- Write **clearly and explicitly**
  - Informal text, scribbled diagrams, conversations, phone calls can help removing ambiguity
- Avoid requirements which contains
  - **Conjunctions** such as “and”, “or”, “with”, “also” are dangerous and misleading
  - **Phrases** such as „if”, „when”, „but”, „except”, „unless”, „although” are dangerous
- **Avoid incompletely specified conditions**



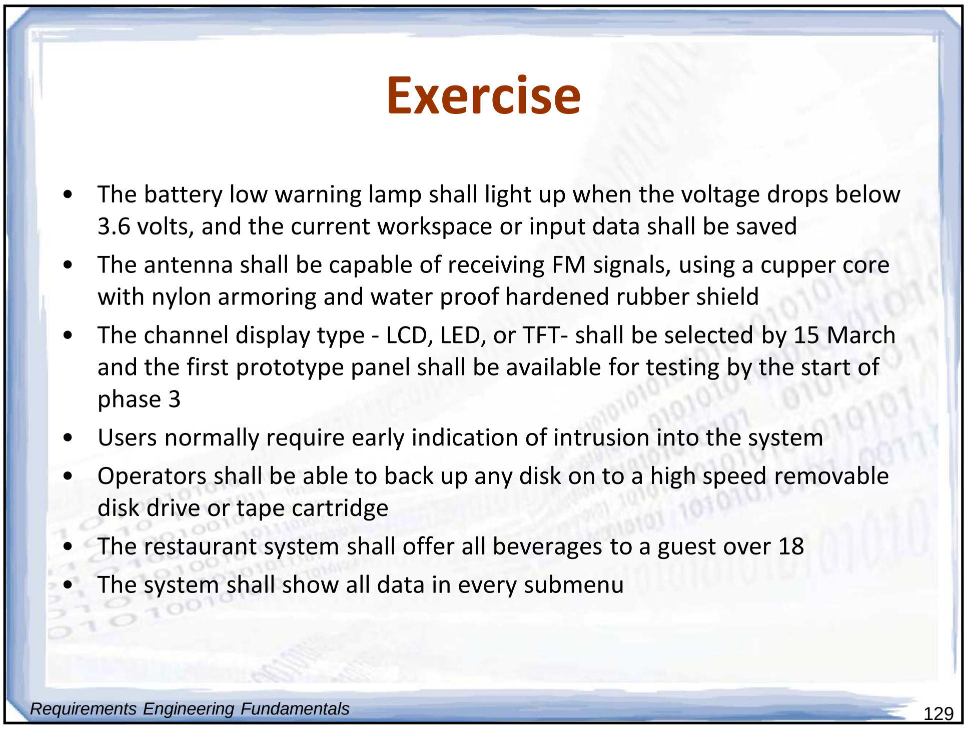
Good Requirements

- **Avoid negative specification as well as passive voice**
- **Avoid mixing requirements with other software artifacts**
  - Confusion may happen when mixing up user requirements, system specifications, design elements, test cases, development guidelines, and installation instructions
  - Danger signs: names of components, materials, software objects/procedures, database fields, dates, project phases and development activities
- **Avoid speculation, wishfulness**
  - There is no room for "wish lists" – general terms about things that somebody probably wants
  - Danger signs include vagueness about which type of users is speaking and generalization words: usually, generally, often, normally, typically, 100% reliable, handle all failures, run on all platforms, never fail, always upgradeuble, etc.
- **Avoid using indefinable terms**
  - User-friendly, versatile, flexible, approximately, as possible, efficient, improved, high performance, modern
  - Perhaps, probably, all



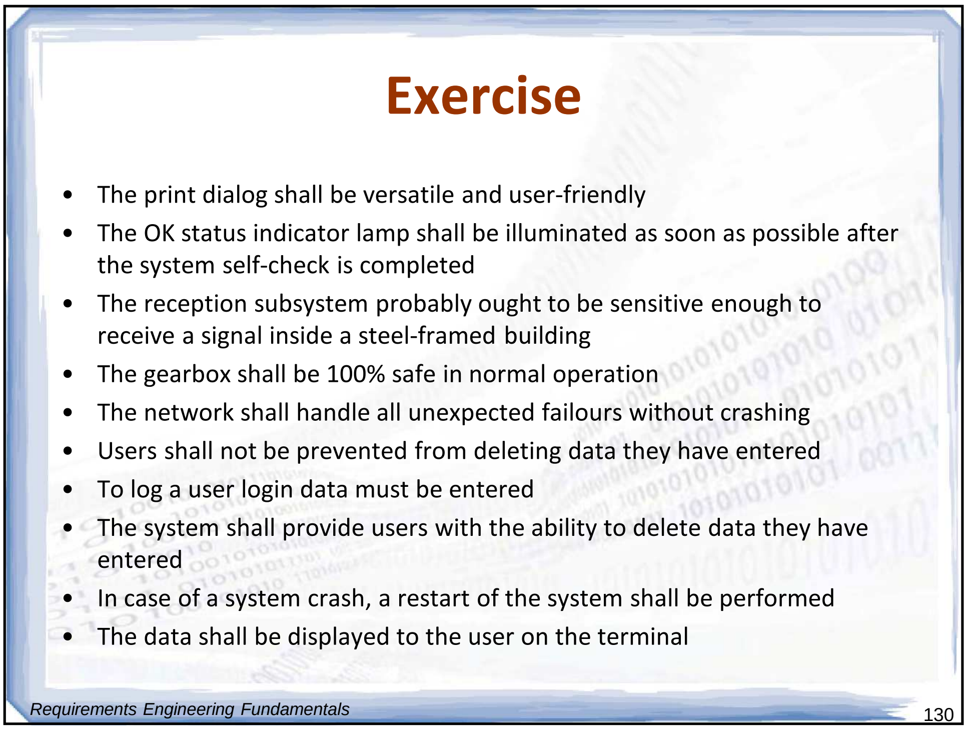
Exercise

- The pilot shall be able to view the airspeed.
- The airline shall be able to reconfigure conventional global business/global traveler seating in less than half a day (see FAA rules)
- The airline shall be able to change the aircraft's seating from business to holidays charter use in less than 12 hours.
- The navigator shall be able to view storm clouds by radar at least 100 km ahead. AC: Aircraft flying at 800 km/h, 10,000 meters towards a storm cloud identified by satellite; storm cloud is detected at a range of at least 100km.
- The same subsystem shall also be able to generate a visible or audible caution/warning signal for the attention of the co-pilot or navigator



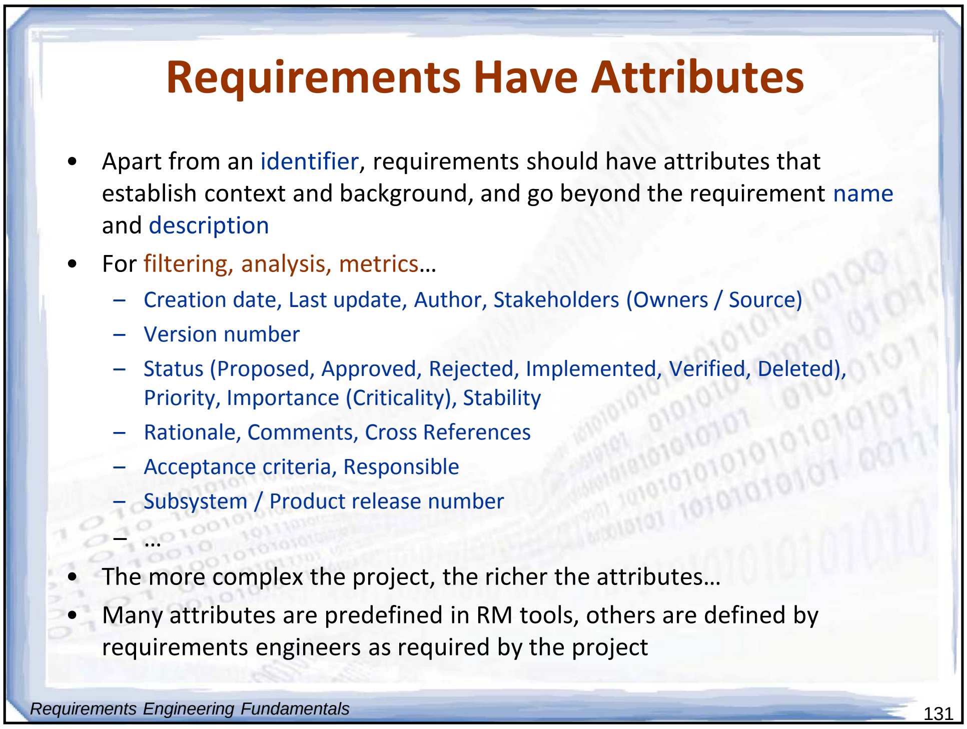
Exercise

- The battery low warning lamp shall light up when the voltage drops below 3.6 volts, and the current workspace or input data shall be saved
- The antenna shall be capable of receiving FM signals, using a cupper core with nylon armoring and water proof hardened rubber shield
- The channel display type - LCD, LED, or TFT- shall be selected by 15 March and the first prototype panel shall be available for testing by the start of phase 3
- Users normally require early indication of intrusion into the system
- Operators shall be able to back up any disk on to a high speed removable disk drive or tape cartridge
- The restaurant system shall offer all beverages to a guest over 18
- The system shall show all data in every submenu



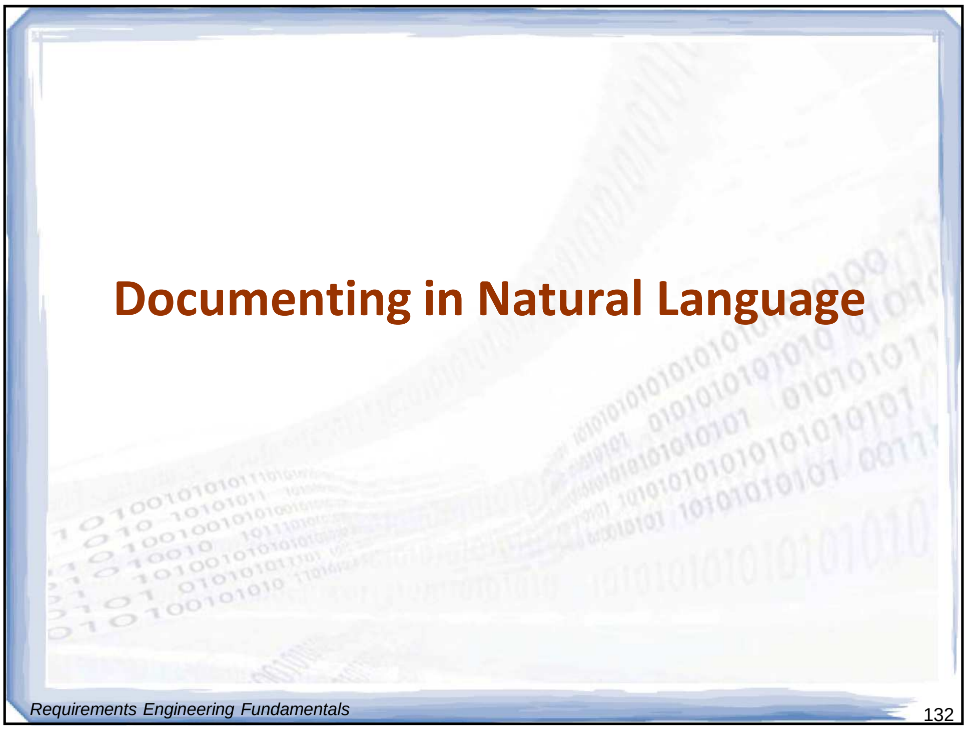
Exercise

- The print dialog shall be versatile and user-friendly
- The OK status indicator lamp shall be illuminated as soon as possible after the system self-check is completed
- The reception subsystem probably ought to be sensitive enough to receive a signal inside a steel-framed building
- The gearbox shall be 100% safe in normal operation
- The network shall handle all unexpected failours without crashing
- Users shall not be prevented from deleting data they have entered
- To log a user login data must be entered
- The system shall provide users with the ability to delete data they have entered
- In case of a system crash, a restart of the system shall be performed
- The data shall be displayed to the user on the terminal



Requirements Have Attributes

- Apart from an identifier, requirements should have attributes that establish context and background, and go beyond the requirement name and description
- For filtering, analysis, metrics…
  - Creation date, Last update, Author, Stakeholders (Owners / Source)
  - Version number
  - Status (Proposed, Approved, Rejected, Implemented, Verified, Deleted), Priority, Importance (Criticality), Stability
  - Rationale, Comments, Cross References
  - Acceptance criteria, Responsible
  - Subsystem / Product release number
  - …
- The more complex the project, the richer the attributes…
- Many attributes are predefined in RM tools, others are defined by requirements engineers as required by the project



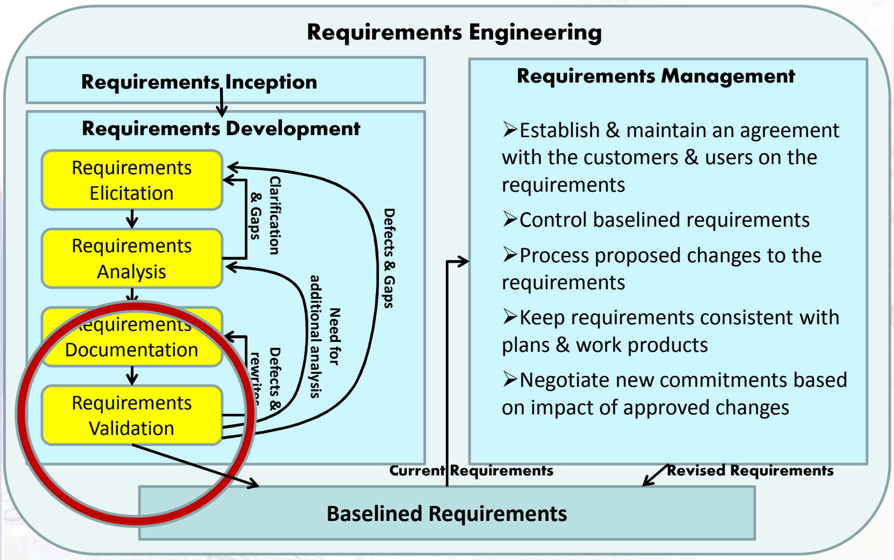
Documenting in Natural Language

Advantages and Disadvantages

- Natural languages requirements are
  - Not formalized
  - May be ambiguous
  - May be interpreted differently
- Using templates
  - Simple, easy to understand
  - Reduces language effects
  - Supports the author in achieving qualitative and unambiguous requirements
  - Low cost

Requirements Validation

Where we are?



Requirements Engineering

Requirements Inception

Requirements Development

Requirements Elicitation

Requirements Analysis

equirements Documentation

Requirements Validation

Requirements Management

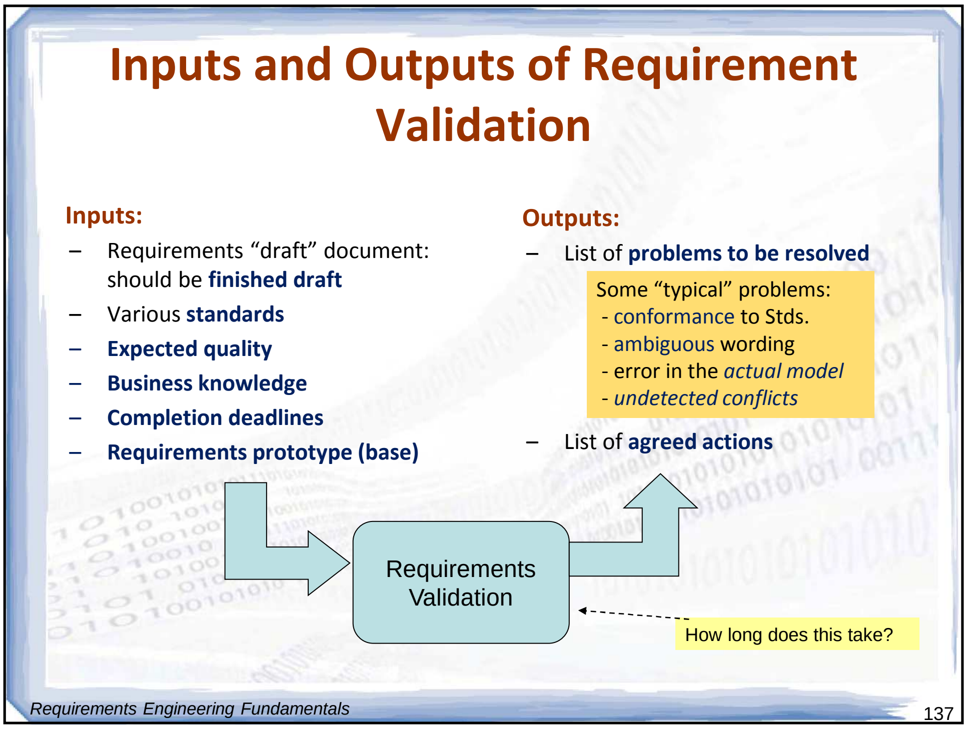
- ➢Establish & maintain an agreement with the customers & users on the requirements
- ➢Control baselined requirements
- ➢Process proposed changes to the requirements
- ➢Keep requirements consistent with plans & work products
- ➢Negotiate new commitments based on impact of approved changes

Baselined Requirements

Requirements Validation

The aim is to check and certify that the requirements represent the system that is to be implemented. It differs from the requirements analysis step and/or negotiation step:

1. Analysis and Negotiations
   - Focus on ensuring that the requirements are *correctly understood* and *reflect the needs* of the stakeholders, rather than the details of correct articulation of the requirements.
   - Concerned with the *“raw” requirements* gathered from the stakeholders.
   - “Have we got the right requirements?”
2. Validation focuses on the documented requirements and how they are represented.
   - Concerned with checking the document that has already gone through analysis and negotiation.
   - “Have we represented (got) the requirements right (correctly)?”



Inputs and Outputs of Requirement Validation

Inputs:

- Requirements “draft” document: should be **finished draft**
- Various **standards**
- **Expected quality**
- **Business knowledge**
- **Completion deadlines**
- **Requirements prototype (base)**

Outputs:

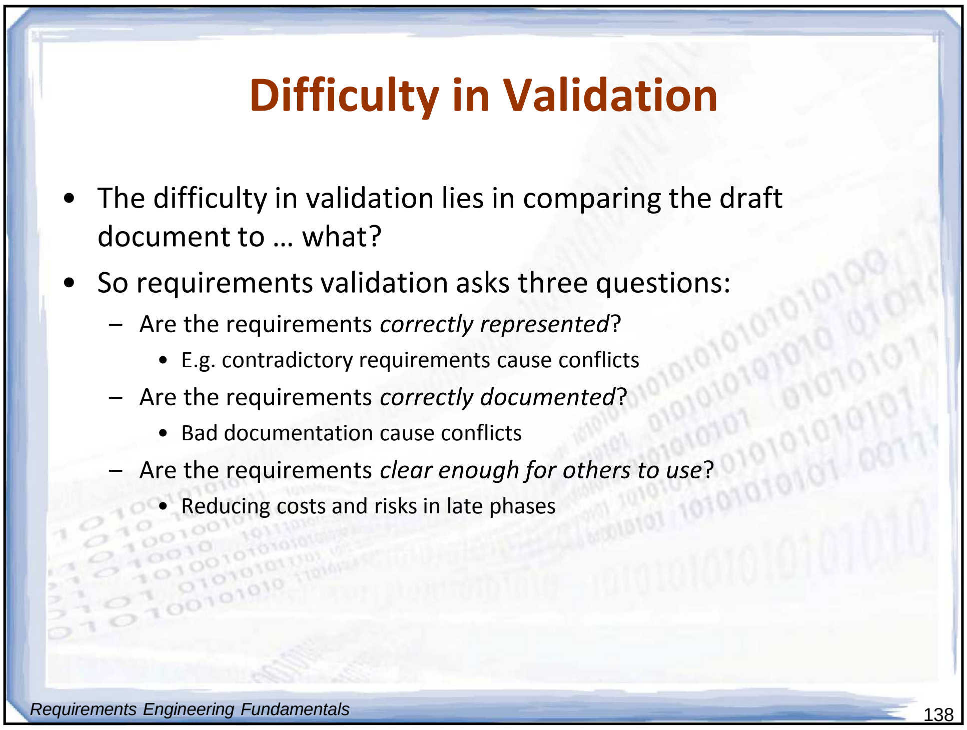
- List of **problems to be resolved**

Some “typical” problems:
- conformance to Stds.
- ambiguous wording
- error in the *actual model*
- *undetected conflicts*

- List of **agreed actions**

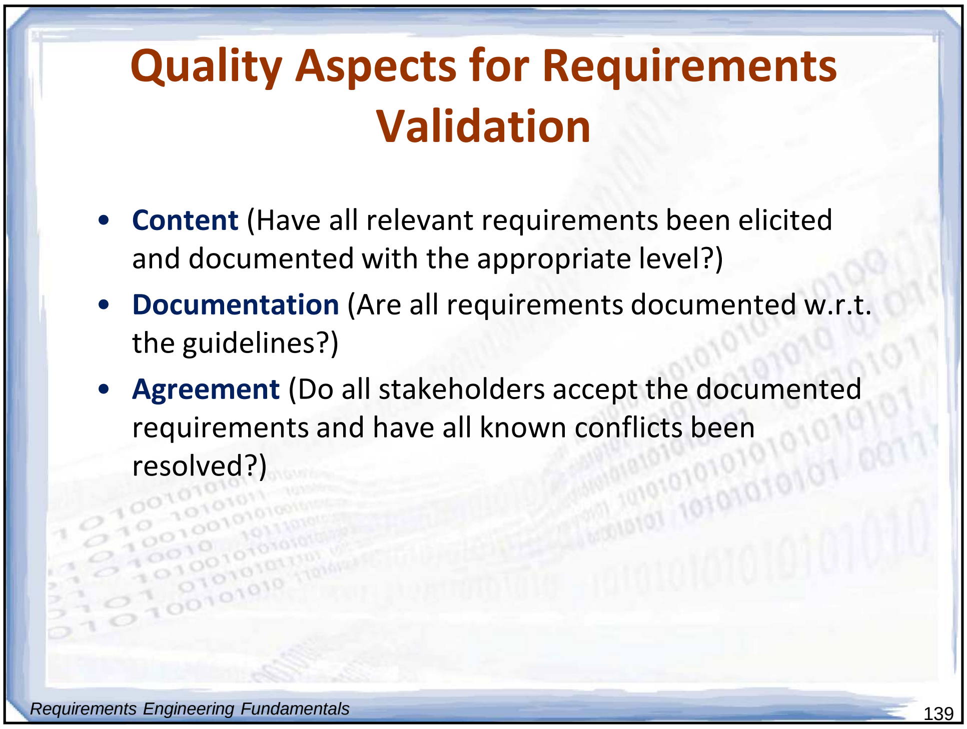
Requirements Validation

How long does this take?



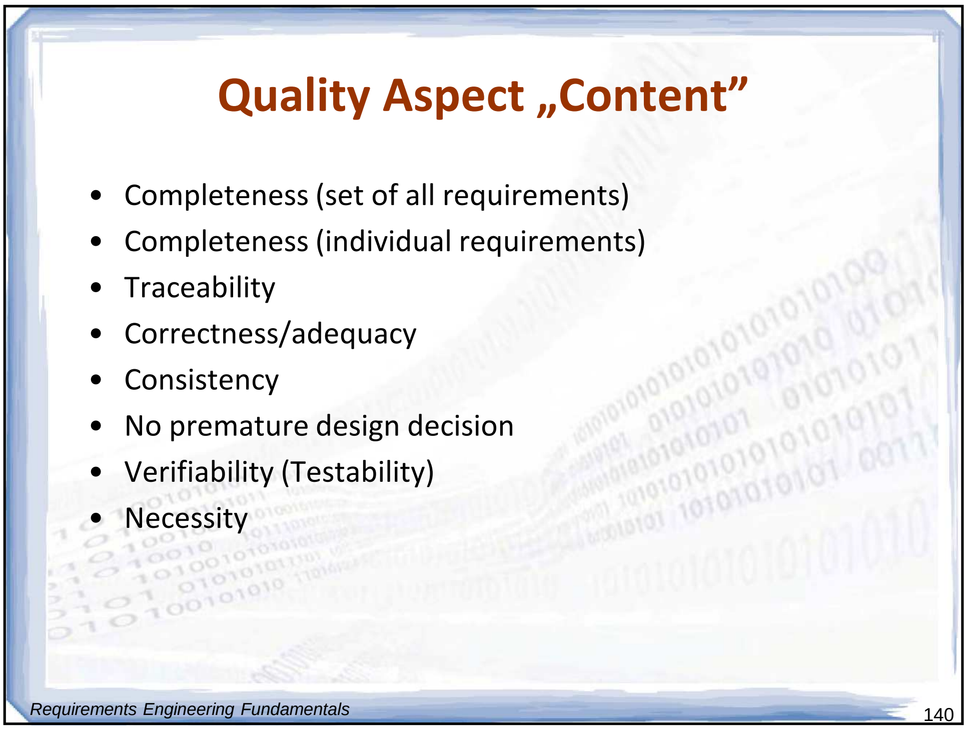
Difficulty in Validation

- The difficulty in validation lies in comparing the draft document to … what?
- So requirements validation asks three questions:
  - Are the requirements *correctly represented*?
    - E.g. contradictory requirements cause conflicts
  - Are the requirements *correctly documented*?
    - Bad documentation cause conflicts
  - Are the requirements *clear enough for others to use*?
    - Reducing costs and risks in late phases



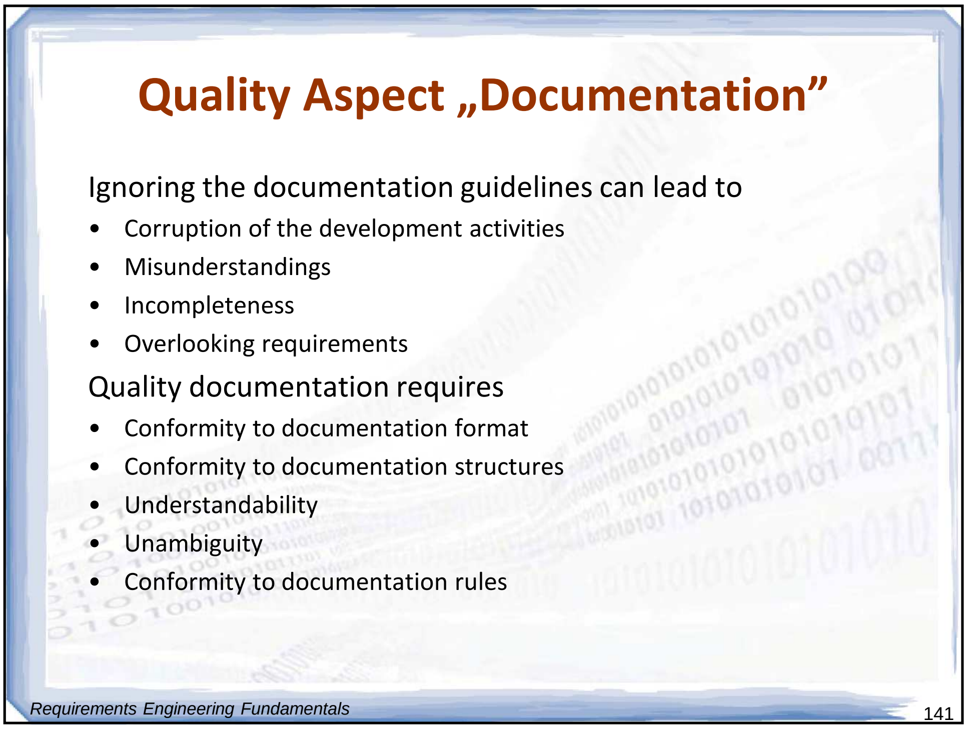
Quality Aspects for Requirements Validation

- **Content** (Have all relevant requirements been elicited and documented with the appropriate level?)
- **Documentation** (Are all requirements documented w.r.t. the guidelines?)
- **Agreement** (Do all stakeholders accept the documented requirements and have all known conflicts been resolved?)



Quality Aspect „Content”

- Completeness (set of all requirements)
- Completeness (individual requirements)
- Traceability
- Correctness/adequacy
- Consistency
- No premature design decision
- Verifiability (Testability)
- Necessity



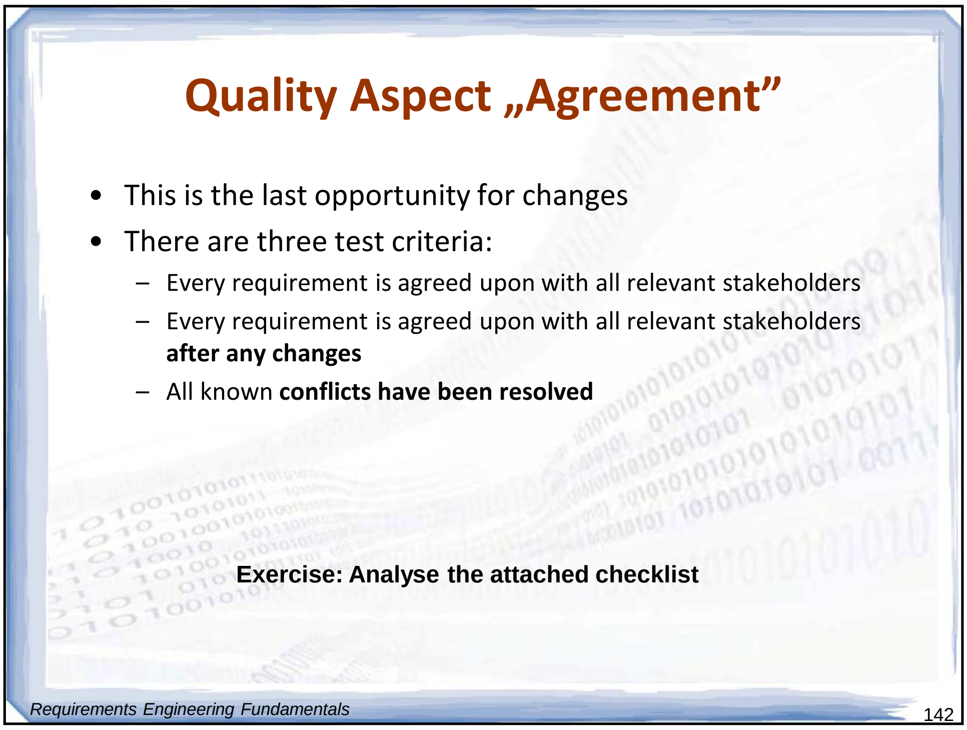
Quality Aspect „Documentation”

Ignoring the documentation guidelines can lead to

- Corruption of the development activities
- Misunderstandings
- Incompleteness
- Overlooking requirements

Quality documentation requires

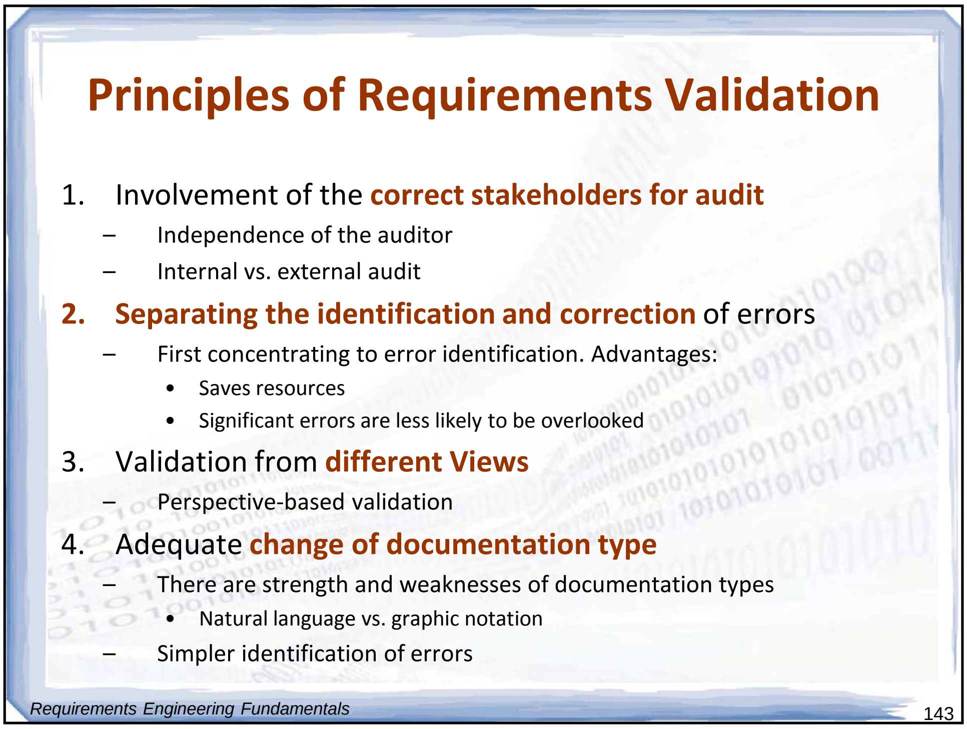
- Conformity to documentation format
- Conformity to documentation structures
- Understandability
- Unambiguity
- Conformity to documentation rules



Quality Aspect „Agreement”

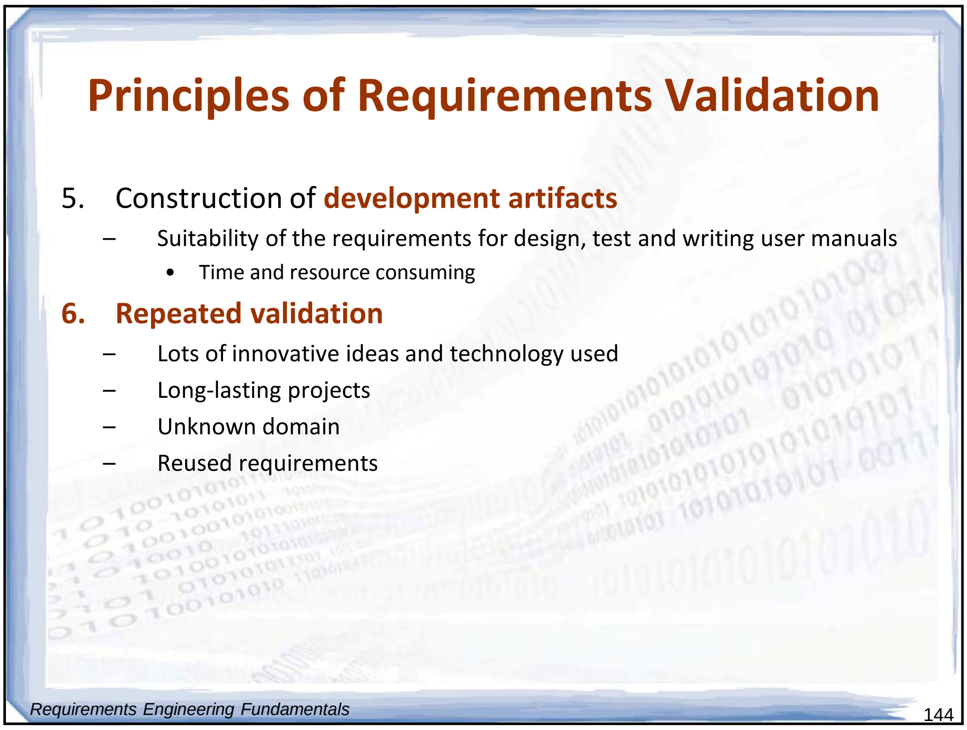
- This is the last opportunity for changes
- There are three test criteria:
  - Every requirement is agreed upon with all relevant stakeholders
  - Every requirement is agreed upon with all relevant stakeholders **after any changes**
  - All known **conflicts have been resolved**

Exercise: Analyse the attached checklist



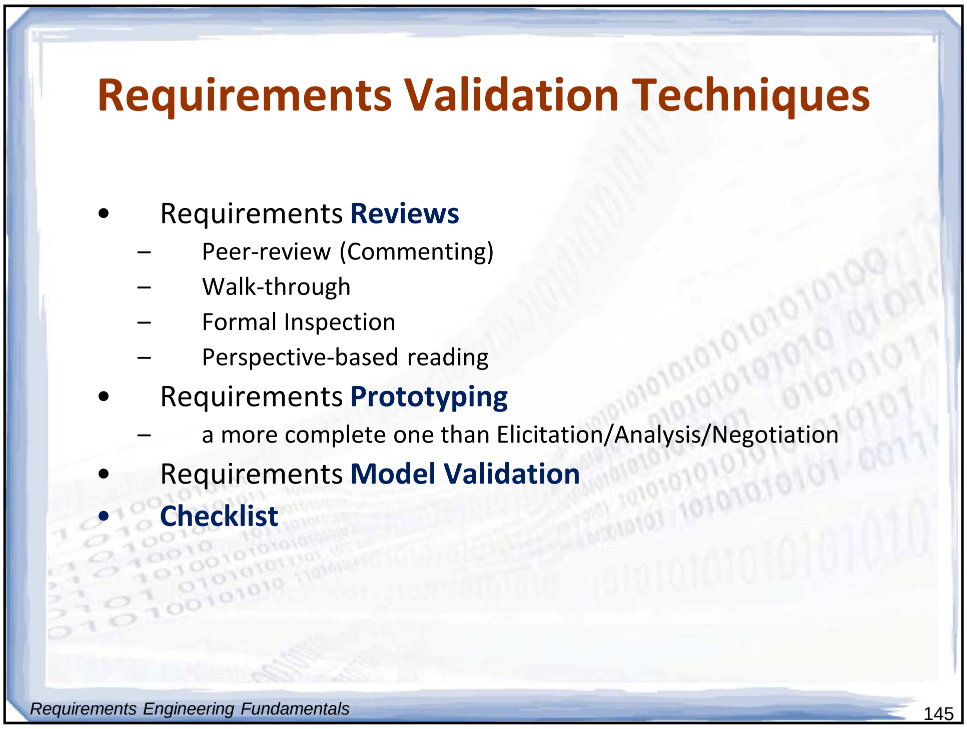
Principles of Requirements Validation

1. Involvement of the **correct stakeholders for audit**
   - Independence of the auditor
   - Internal vs. external audit
2. **Separating the identification and correction** of errors
   - First concentrating to error identification. Advantages:
     - Saves resources
     - Significant errors are less likely to be overlooked
3. Validation from **different Views**
   - Perspective-based validation
4. Adequate **change of documentation type**
   - There are strength and weaknesses of documentation types
     - Natural language vs. graphic notation
   - Simpler identification of errors



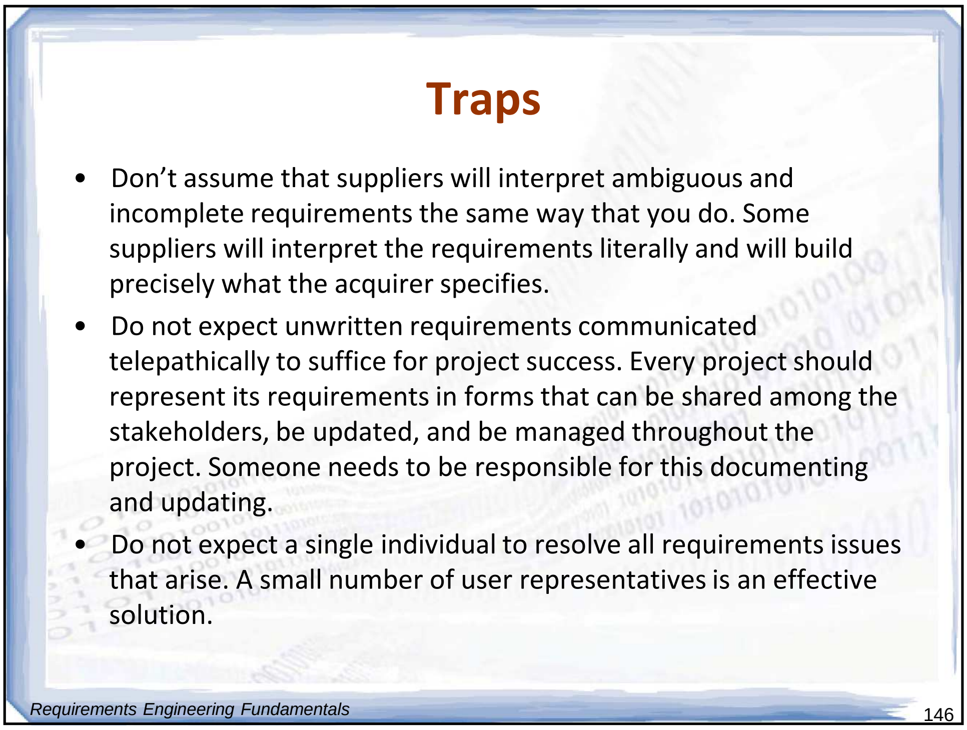
Principles of Requirements Validation

5. Construction of **development artifacts**
   - Suitability of the requirements for design, test and writing user manuals
     - Time and resource consuming
6. **Repeated validation**
   - Lots of innovative ideas and technology used
   - Long-lasting projects
   - Unknown domain
   - Reused requirements



Requirements Validation Techniques

- Requirements **Reviews**
  - Peer-review (Commenting)
  - Walk-through
  - Formal Inspection
  - Perspective-based reading
- Requirements **Prototyping**
  - a more complete one than Elicitation/Analysis/Negotiation
- Requirements **Model Validation**
- **Checklist**

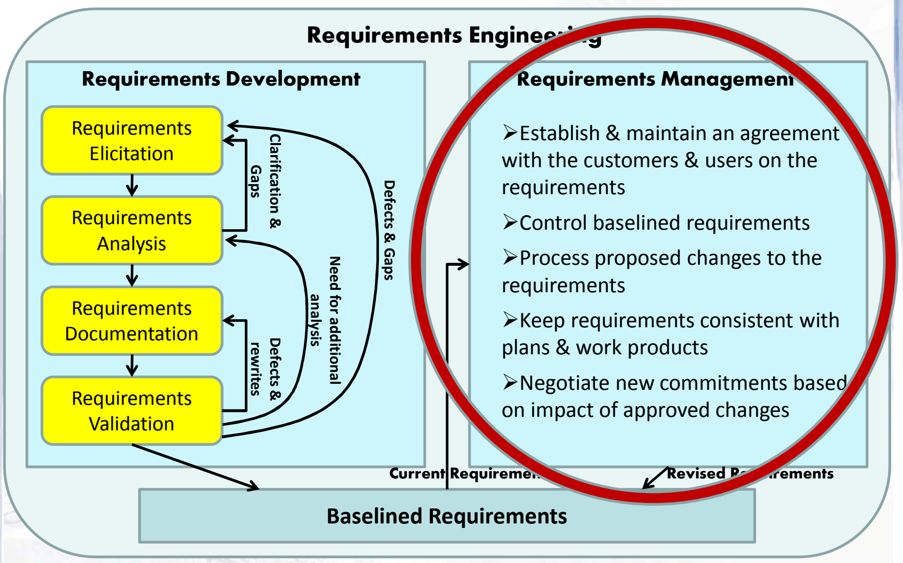


Traps

- Don’t assume that suppliers will interpret ambiguous and incomplete requirements the same way that you do. Some suppliers will interpret the requirements literally and will build precisely what the acquirer specifies.
- Do not expect unwritten requirements communicated telepathically to suffice for project success. Every project should represent its requirements in forms that can be shared among the stakeholders, be updated, and be managed throughout the project. Someone needs to be responsible for this documenting and updating.
- Do not expect a single individual to resolve all requirements issues that arise. A small number of user representatives is an effective solution.

Requirements Management and Supporting Tools

Where we are?



- Establish & maintain an agreement with the customers & users on the requirements
- Control baselined requirements
- Process proposed changes to the requirements
- Keep requirements consistent with plans & work products
- Negotiate new commitments based on impact of approved changes

Requirements Management

A systematic approach to eliciting, organizing, and documenting the requirement of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.

Leffingwell & Widrig 1999, p.16

Some Problems Due to Changing Requirements

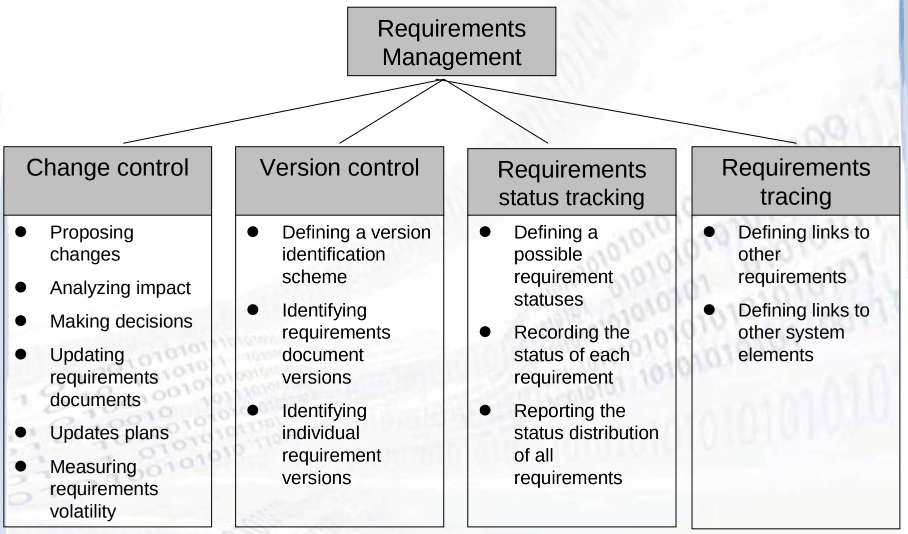
- Requirements are changing towards the end of development without any impact assessment
- Unmatched/outdated requirements specifications causing confusion and unnecessary rework
- Time spent coding, writing test cases or documentation for requirements that no longer exist

Req’s Management Activities (1)

Requirements management includes all activities intended to maintain the integrity and accuracy of expected req’s

- Manage changes to **agreed requirements**
- Manage changes to **baseline** (increments)
- Keep project plans **synchronized** with requirements
- Control **versions** of individual requirements and versions of requirements documents
- Manage **relationships** between requirements
- Managing the **dependencies** between the requirements document and other documents produced in the systems engineering process
- Track requirements **status**

Req’s Management Activities (2)



Requirements Management

Change control

- Proposing changes
- Analyzing impact
- Making decisions
- Updating requirements documents
- Updates plans
- Measuring requirements volatility

Version control

- Defining a version identification scheme
- Identifying requirements document versions
- Identifying individual requirement versions

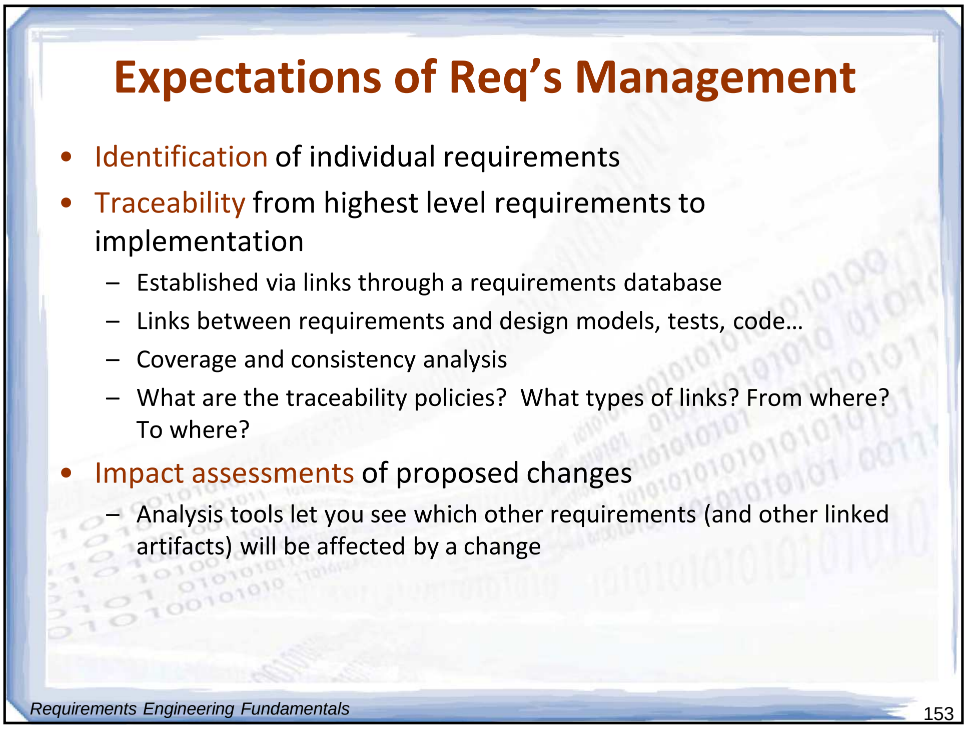
Requirements status tracking

- Defining a possible requirement statuses
- Recording the status of each requirement
- Reporting the status distribution of all requirements

Requirements tracing

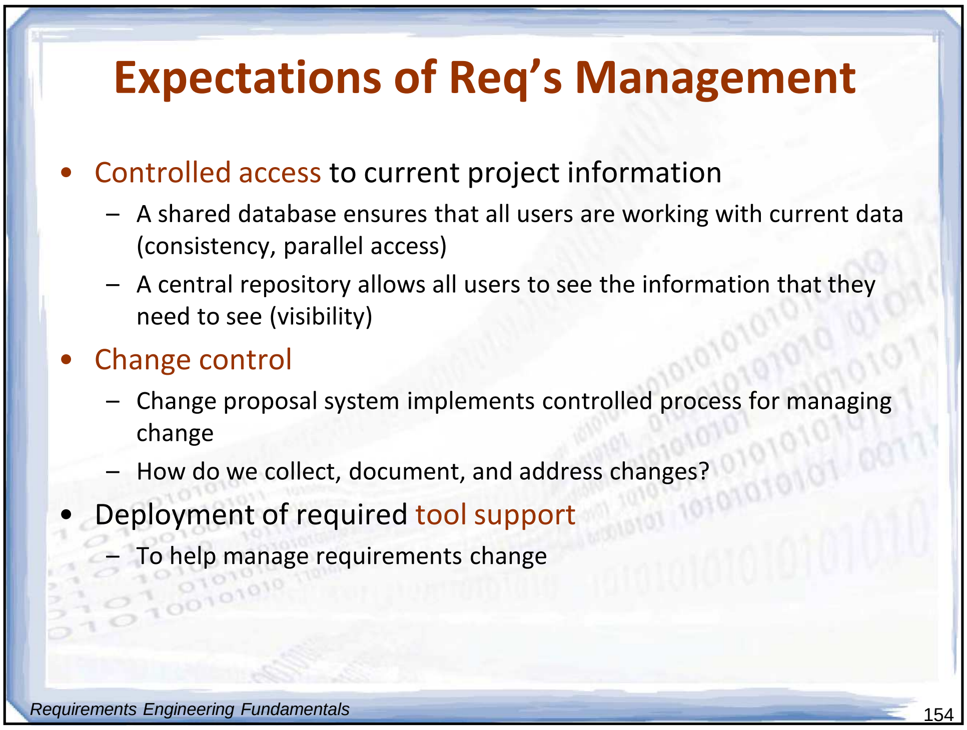
- Defining links to other requirements
- Defining links to other system elements

Source: Wiegers, 1999



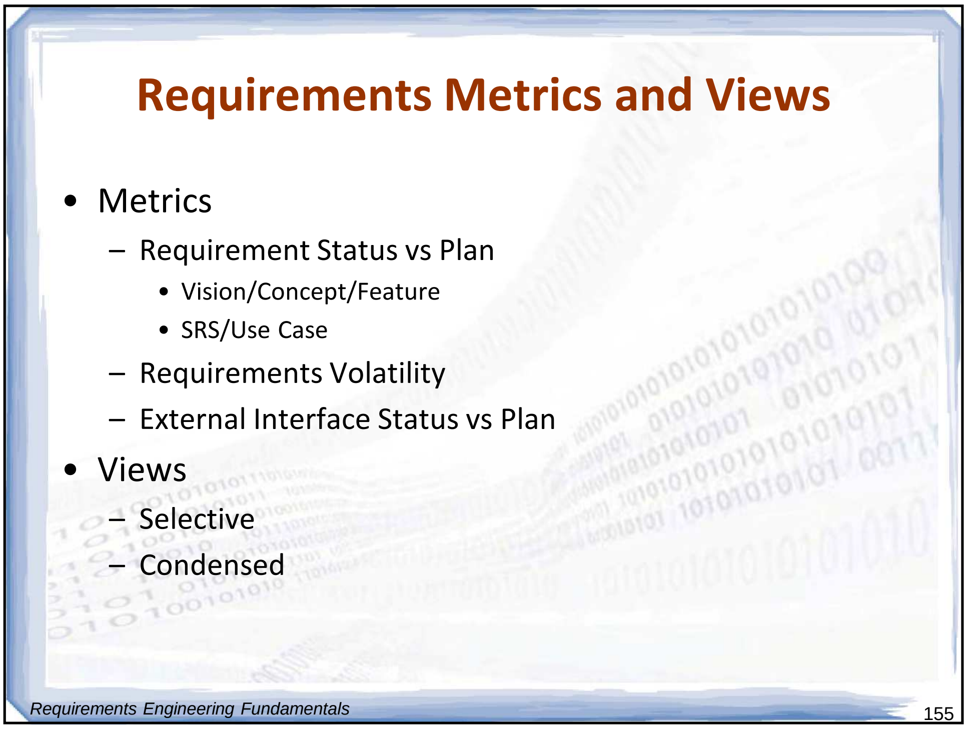
Expectations of Req's Management

- Identification of individual requirements
- Traceability from highest level requirements to implementation
  - Established via links through a requirements database
  - Links between requirements and design models, tests, code…
  - Coverage and consistency analysis
  - What are the traceability policies? What types of links? From where? To where?
- Impact assessments of proposed changes
  - Analysis tools let you see which other requirements (and other linked artifacts) will be affected by a change



Expectations of Req's Management

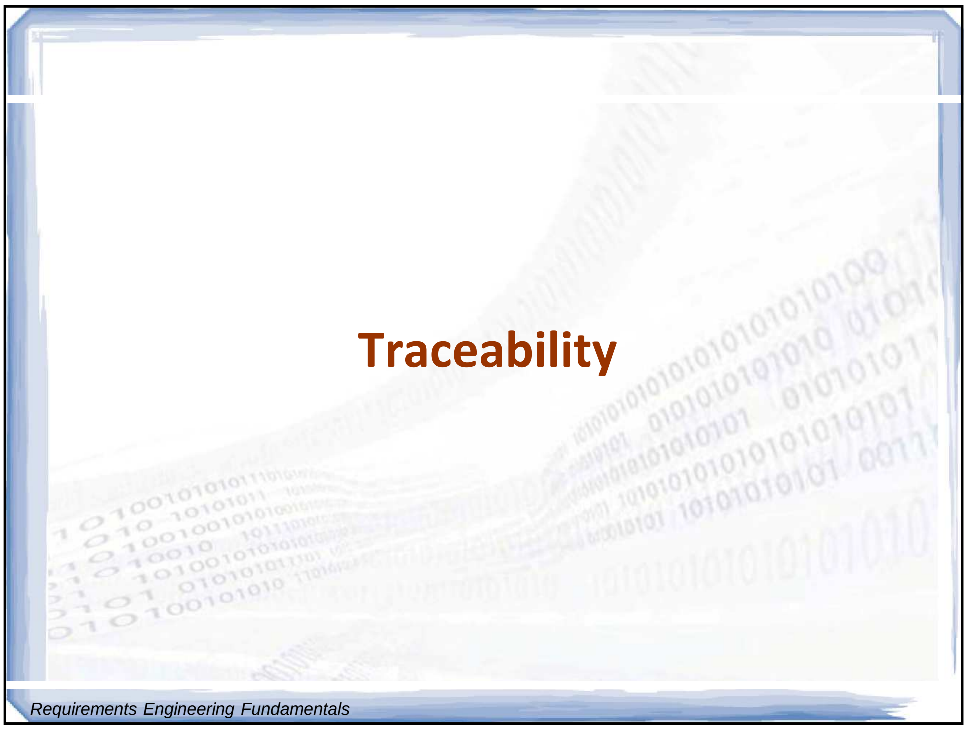
- Controlled access to current project information
  - A shared database ensures that all users are working with current data (consistency, parallel access)
  - A central repository allows all users to see the information that they need to see (visibility)
- Change control
  - Change proposal system implements controlled process for managing change
  - How do we collect, document, and address changes?
- Deployment of required tool support
  - To help manage requirements change



Requirements Metrics and Views

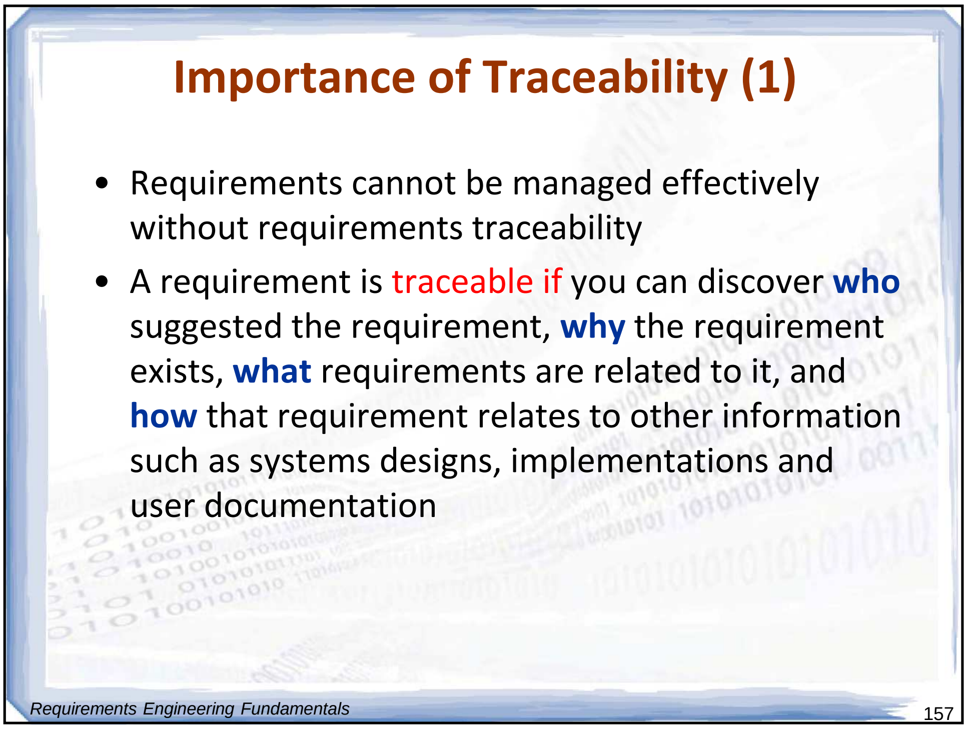
- Metrics
  - Requirement Status vs Plan
    - Vision/Concept/Feature
    - SRS/Use Case
  - Requirements Volatility
  - External Interface Status vs Plan
- Views
  - Selective
  - Condensed

155



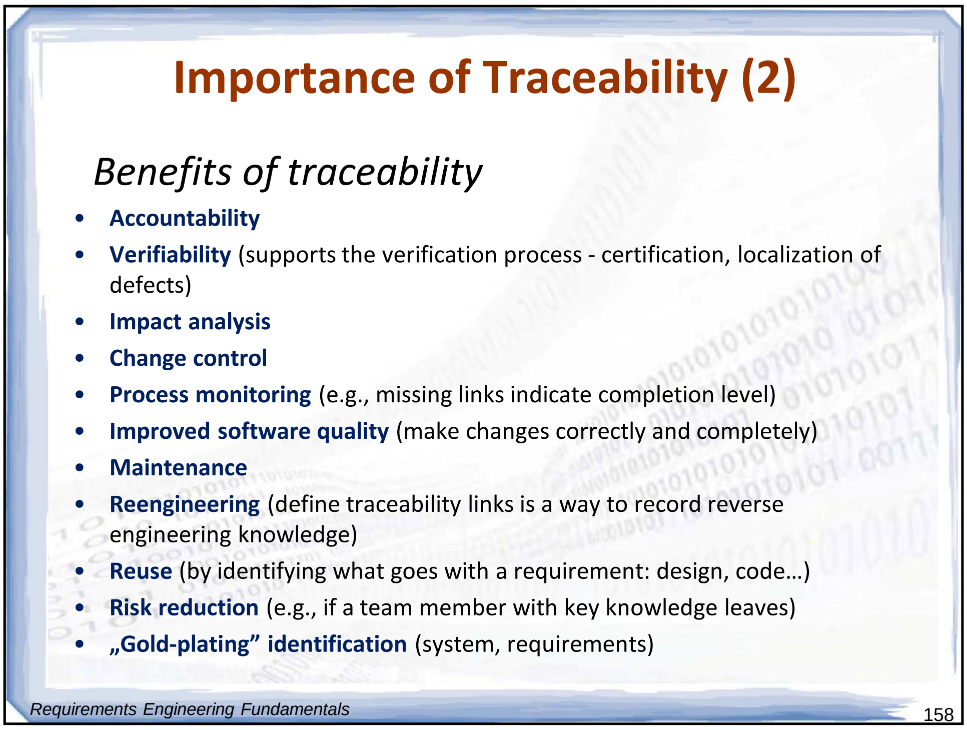
Traceability

156



Importance of Traceability (1)

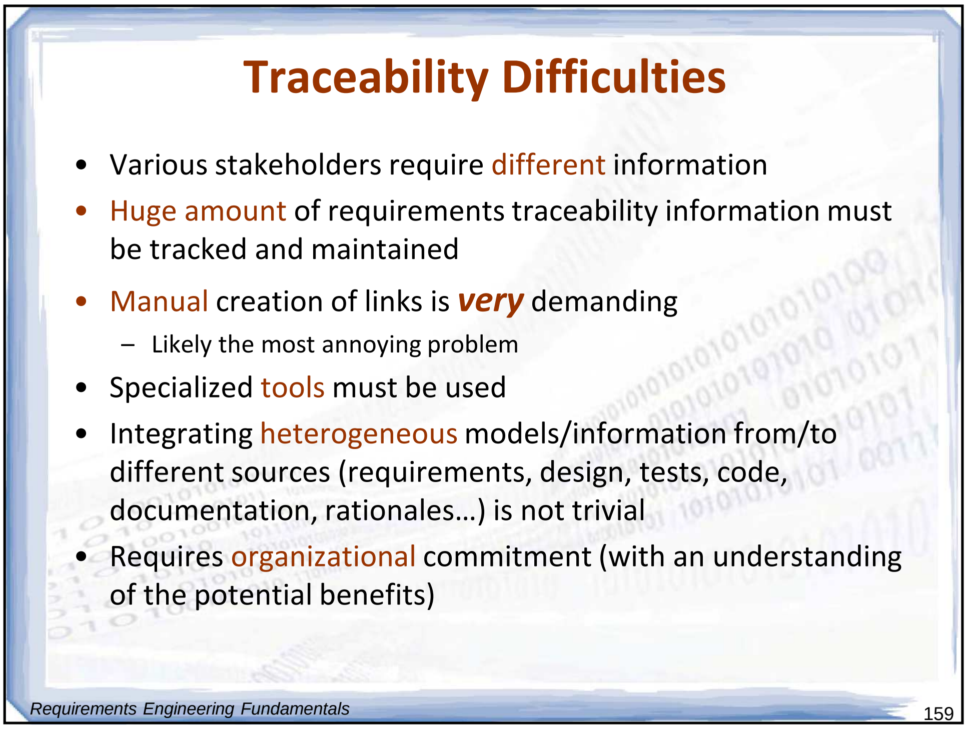
- Requirements cannot be managed effectively without requirements traceability
- A requirement is traceable if you can discover **who** suggested the requirement, **why** the requirement exists, **what** requirements are related to it, and **how** that requirement relates to other information such as systems designs, implementations and user documentation



Importance of Traceability (2)

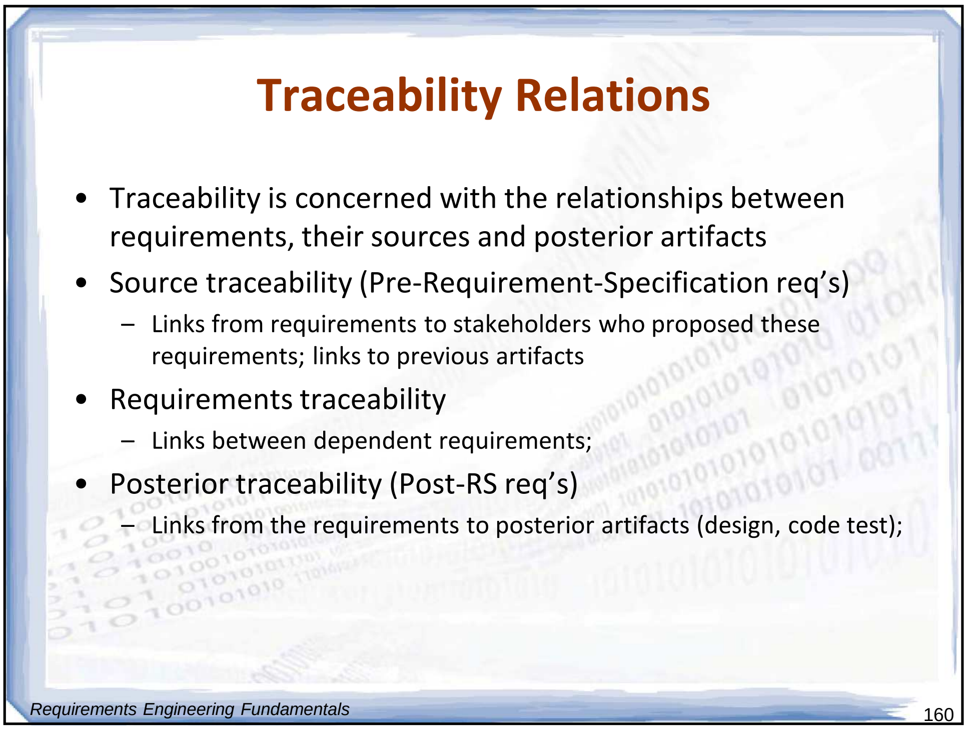
Benefits of traceability

- **Accountability**
- **Verifiability** (supports the verification process - certification, localization of defects)
- **Impact analysis**
- **Change control**
- **Process monitoring** (e.g., missing links indicate completion level)
- **Improved software quality** (make changes correctly and completely)
- **Maintenance**
- **Reengineering** (define traceability links is a way to record reverse engineering knowledge)
- **Reuse** (by identifying what goes with a requirement: design, code…)
- **Risk reduction** (e.g., if a team member with key knowledge leaves)
- **„Gold-plating” identification** (system, requirements)



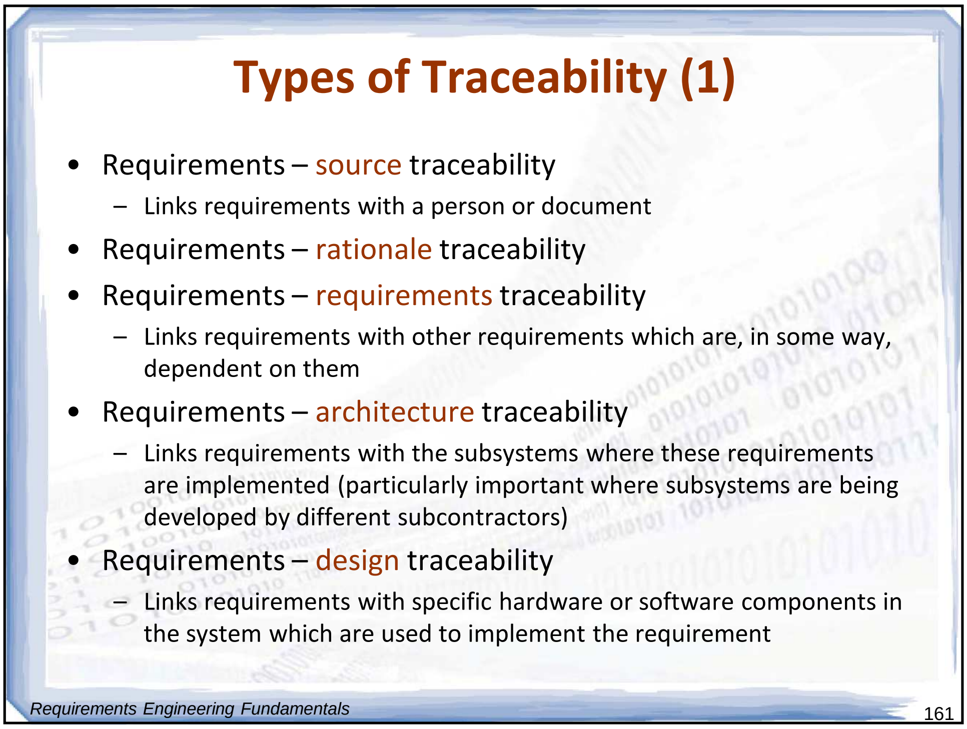
Traditional Difficulties

- Various stakeholders require different information
- Huge amount of requirements traceability information must be tracked and maintained
- Manual creation of links is ***very*** demanding
  - Likely the most annoying problem
- Specialized tools must be used
- Integrating heterogeneous models/information from/to different sources (requirements, design, tests, code, documentation, rationales…) is not trivial
- Requires organizational commitment (with an understanding of the potential benefits)



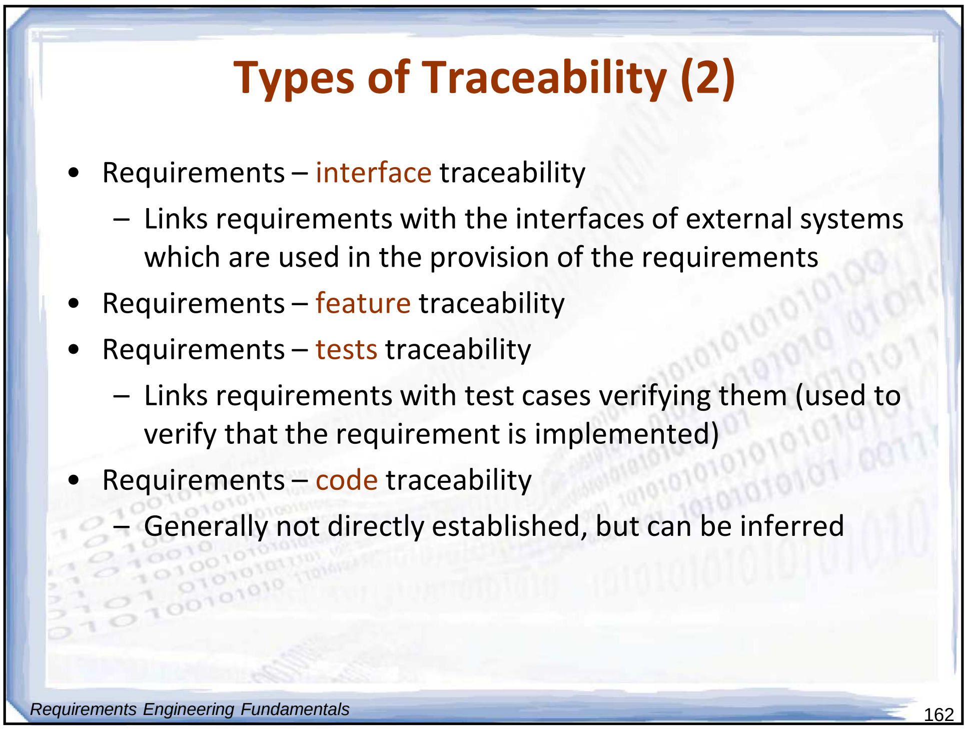
Traceability Relations

- Traceability is concerned with the relationships between requirements, their sources and posterior artifacts
- Source traceability (Pre-Requirement-Specification req’s)
  - Links from requirements to stakeholders who proposed these requirements; links to previous artifacts
- Requirements traceability
  - Links between dependent requirements;
- Posterior traceability (Post-RS req’s)
  - Links from the requirements to posterior artifacts (design, code test);



Types of Traceability (1)

- Requirements – source traceability
  - Links requirements with a person or document
- Requirements – rationale traceability
- Requirements – requirements traceability
  - Links requirements with other requirements which are, in some way, dependent on them
- Requirements – architecture traceability
  - Links requirements with the subsystems where these requirements are implemented (particularly important where subsystems are being developed by different subcontractors)
- Requirements – design traceability
  - Links requirements with specific hardware or software components in the system which are used to implement the requirement

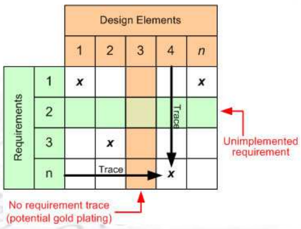


Types of Traceability (2)

- Requirements – interface traceability
  - Links requirements with the interfaces of external systems which are used in the provision of the requirements
- Requirements – feature traceability
- Requirements – tests traceability
  - Links requirements with test cases verifying them (used to verify that the requirement is implemented)
- Requirements – code traceability
  - Generally not directly established, but can be inferred

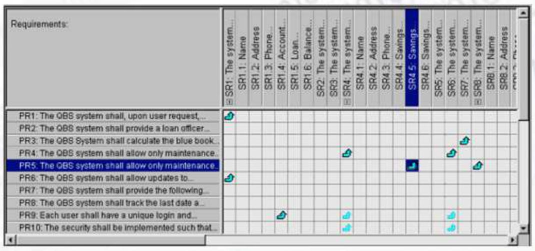
Representation – Traceability Matrix

- Define links between pairs of elements
  - E.g., requirements to requirement, use case to requirement, requirement to test case…
- Can be used to defined relationships between pairs
  - E.g., specifies/is specified by, depends on, is parent of, constrains…



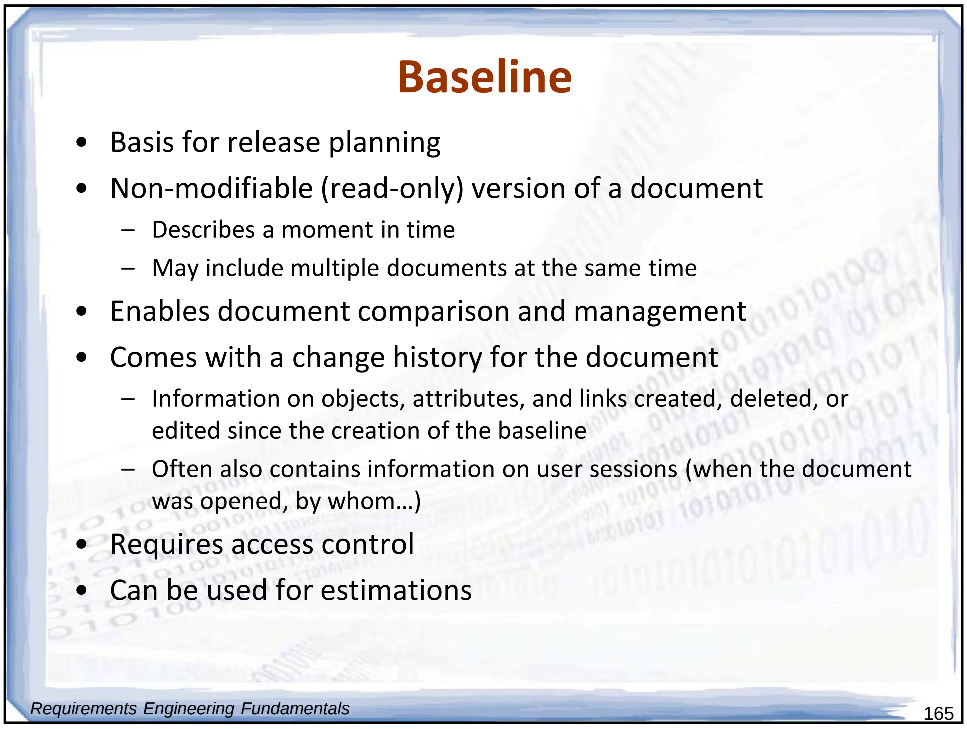
		Design Elements				
		1	2	3	4	n
Requirements	1	x				x
	2					
	3		x			
	n				x	

Trace
Trace
Unimplemented requirement
No requirement trace (potential gold plating)



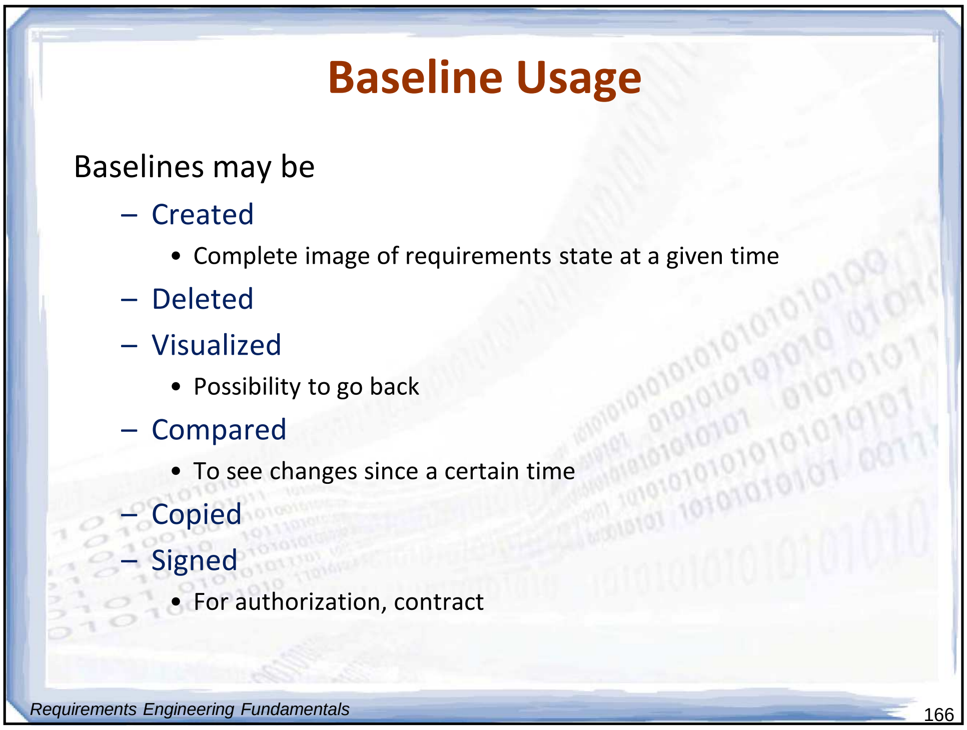
Requirements:	SR1: The system...	SR1.1: Name	SR1.2: Address	SR1.3: Phone...	SR1.4: Account...	SR1.5: Loan...	SR1.6: Balance...	SR2: The system...	SR3: The system...	SR4: The system...	SR4.1: Name	SR4.2: Address	SR4.3: Phone...	SR4.4: Savings...	SR4.5: Savings...	SR4.6: Savings...	SR5: The system...	SR6: The system...	SR7: The system...	SR8: The system...	SR8.1: Name	SR8.2: Address
PR1: The QBS system shall, upon user request,...	●																					
PR2: The QBS system shall provide a loan officer...																						
PR3: The QBS System shall calculate the blue book...																			●			
PR4: The QBS system shall allow only maintenance...										●								●				
PR5: The QBS system shall allow only maintenance...															●					●		
PR6: The QBS system shall allow updates to...	●																					
PR7: The QBS system shall provide the following...																						
PR8: The QBS system shall track the last date a...																						
PR9: Each user shall have a unique login and...					●					●								●				
PR10: The security shall be implemented such that...										●								●				

Baselines



Baseline

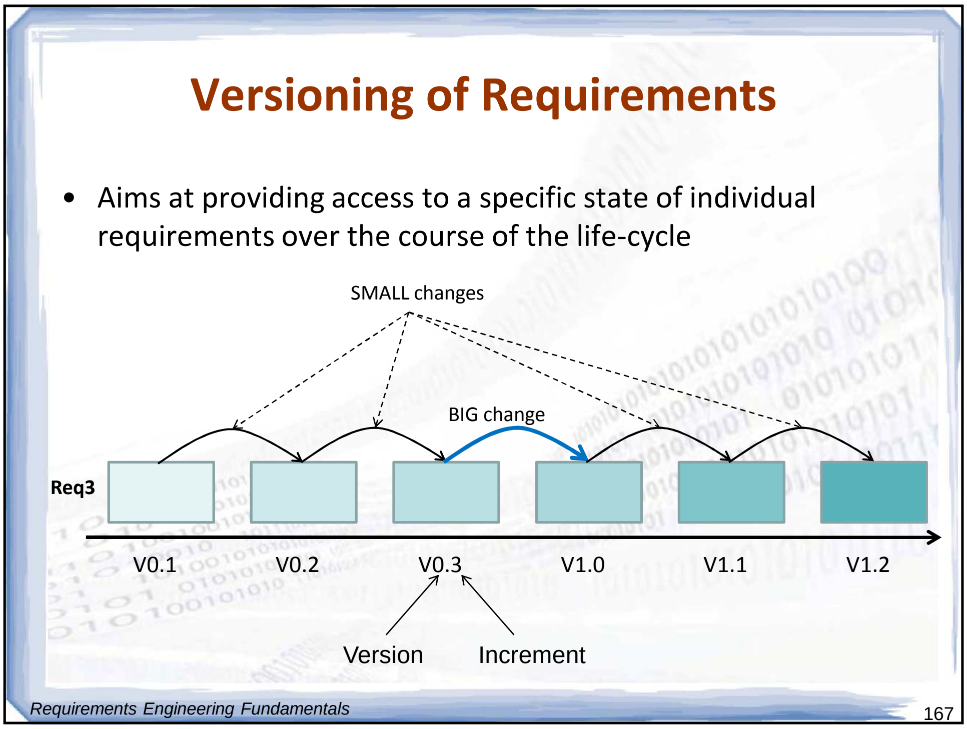
- Basis for release planning
- Non-modifiable (read-only) version of a document
  - Describes a moment in time
  - May include multiple documents at the same time
- Enables document comparison and management
- Comes with a change history for the document
  - Information on objects, attributes, and links created, deleted, or edited since the creation of the baseline
  - Often also contains information on user sessions (when the document was opened, by whom…)
- Requires access control
- Can be used for estimations



Baseline Usage

Baselines may be

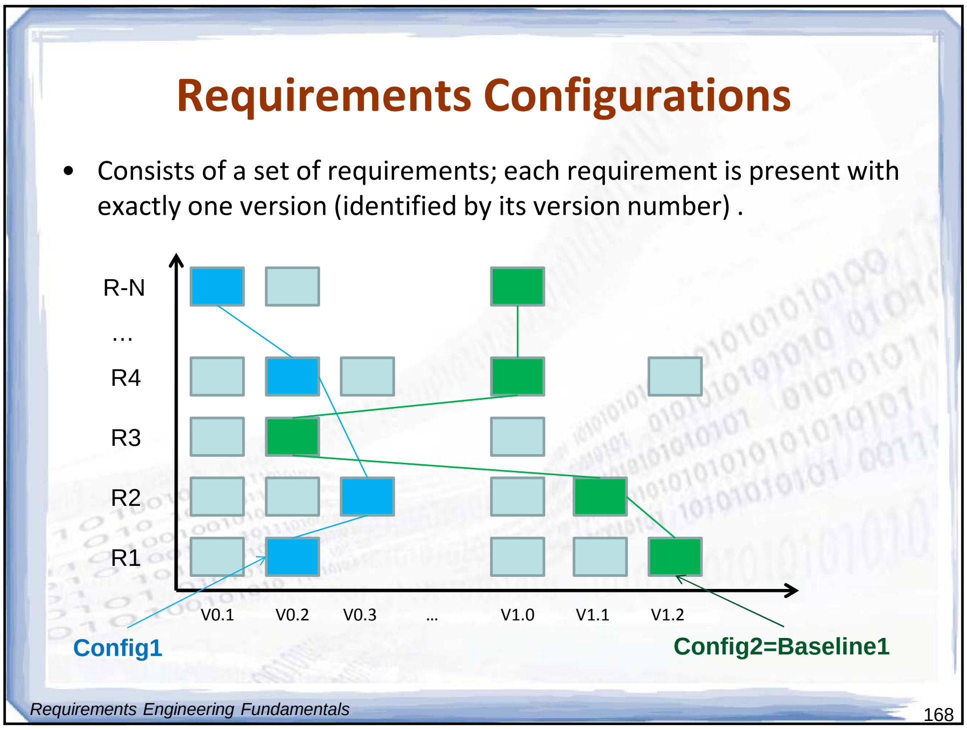
- Created
  - Complete image of requirements state at a given time
- Deleted
- Visualized
  - Possibility to go back
- Compared
  - To see changes since a certain time
- Copied
- Signed
  - For authorization, contract



Versioning of Requirements

- Aims at providing access to a specific state of individual requirements over the course of the life-cycle

167

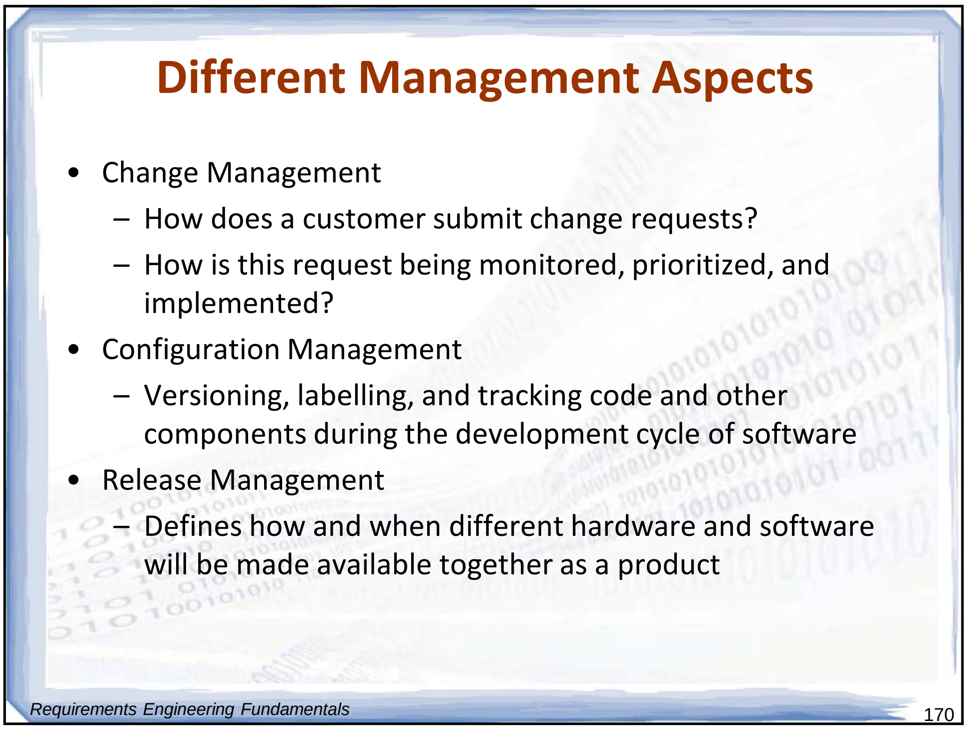


Requirements Configurations

- Consists of a set of requirements; each requirement is present with exactly one version (identified by its version number) .

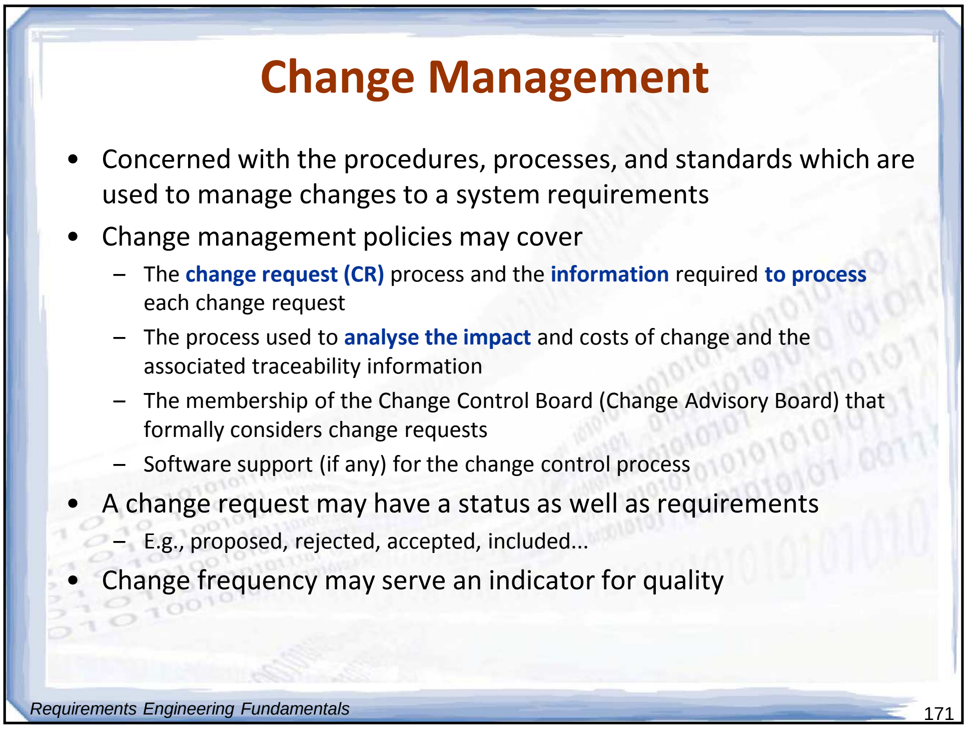
168

Change Management



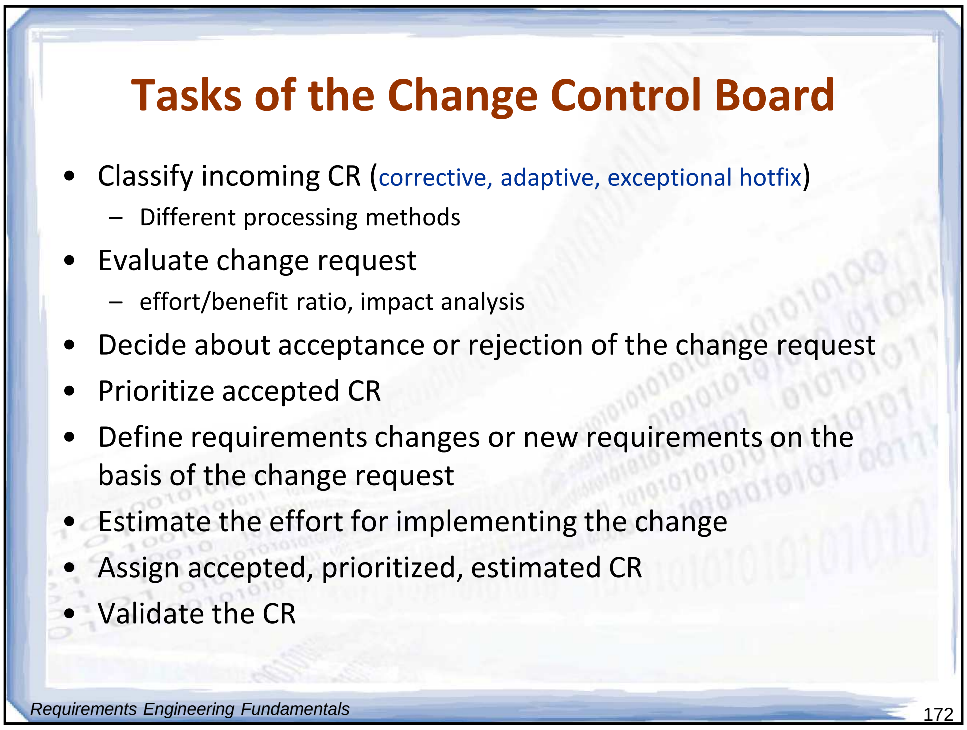
Different Management Aspects

- Change Management
  - How does a customer submit change requests?
  - How is this request being monitored, prioritized, and implemented?
- Configuration Management
  - Versioning, labelling, and tracking code and other components during the development cycle of software
- Release Management
  - Defines how and when different hardware and software will be made available together as a product



Change Management

- Concerned with the procedures, processes, and standards which are used to manage changes to a system requirements
- Change management policies may cover
  - The **change request (CR)** process and the **information** required **to process** each change request
  - The process used to **analyse the impact** and costs of change and the associated traceability information
  - The membership of the Change Control Board (Change Advisory Board) that formally considers change requests
  - Software support (if any) for the change control process
- A change request may have a status as well as requirements
  - E.g., proposed, rejected, accepted, included…
- Change frequency may serve an indicator for quality



Tasks of the Change Control Board

- Classify incoming CR (corrective, adaptive, exceptional hotfix)
  - Different processing methods
- Evaluate change request
  - effort/benefit ratio, impact analysis
- Decide about acceptance or rejection of the change request
- Prioritize accepted CR
- Define requirements changes or new requirements on the basis of the change request
- Estimate the effort for implementing the change
- Assign accepted, prioritized, estimated CR
- Validate the CR

Change Request Form

- Proposed changes are usually recorded on a change request form which is then passed to all of the people involved in the analysis of the change
- Change request forms may include
  - Identifier, Title, Justification (Reasons)
  - Date, Customer/Requester, Product/System including version
  - Description of change request including rationale
  - Priority (in the customer's opinion)
  - Signature fields (for validaton)
  - Status
  - Impact analysis (description and status)
  - Responsible
  - Comments

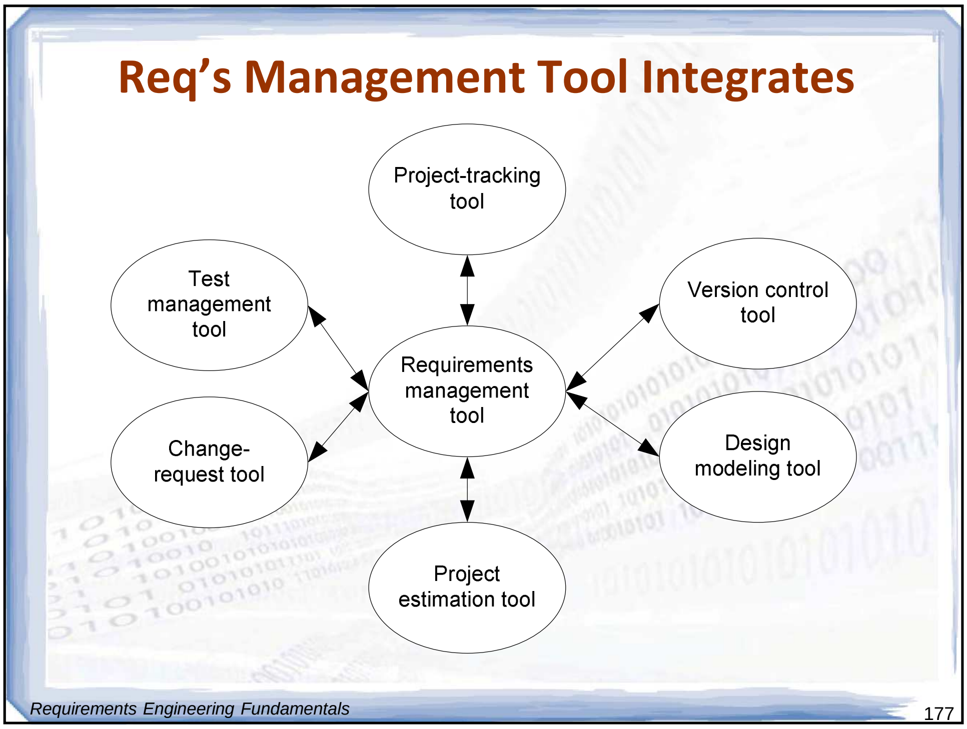
Requirements Management Tools

What Kind of Tool Do We Need?

- Different companies will use different tools, which may or may not be tailored to the requirements management task
- Specialized tools and standard office applications
  - Word processor (Microsoft Word with templates…)
  - Spreadsheet (Microsoft Excel…)
  - Industrial-strength, commercial RM tools
    - IBM/Telelogic DOORS, IBM Requisite Pro, Borland CaliberRM…
  - Internal tools
  - Open source RM tools
    - OSRMT: http://sourceforge.net/projects/osrmt
  - Bug tracking tools (free or not)
    - Bugzilla…
  - Collaboration tools (free or not)
    - TWiki…

What Should We Look For in a Tool?

- Types/attributes for requirements and links
- Specifications and models
- Version and change management
- Database repository
- Traceability
- Analysis (impact, completeness, style, differences…)
- Automatic inspection of requirements (according to rules)
- Visualization and reports
- Manage different information
- Requirements document and other reports generation
- Monitoring of requirements statuses
- Access control
- Import/export
- Communication with stakeholders
- Scripting language (for automation)
- Reuse of requirements, models, projects



Req's Management Tool Integrates

Project-tracking tool

Test management tool

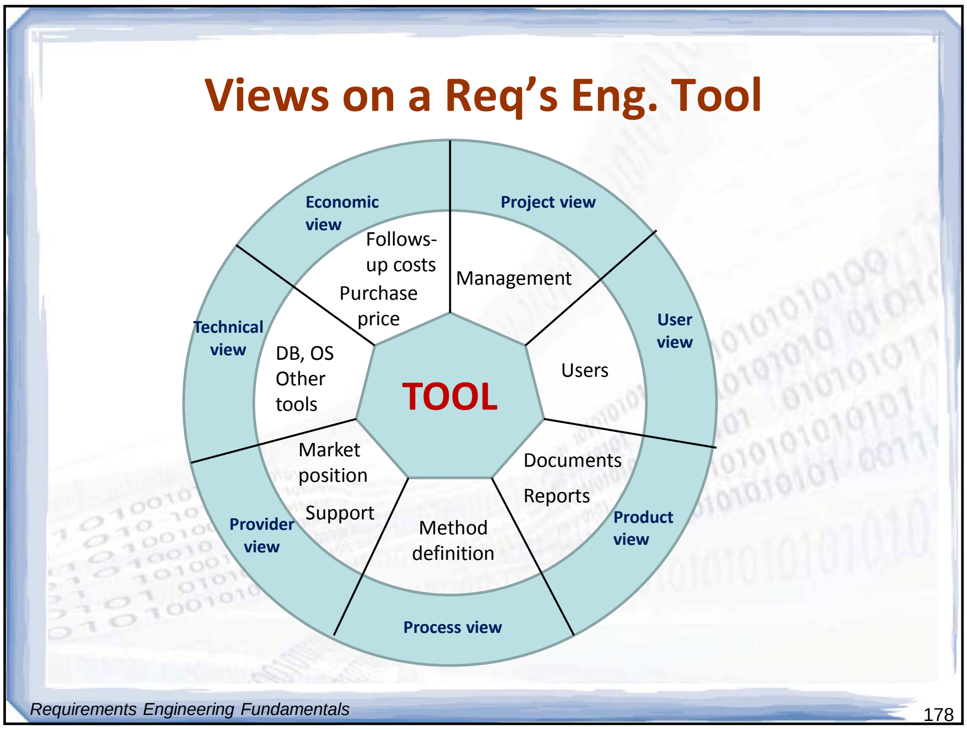
Version control tool

Requirements management tool

Change-request tool

Design modeling tool

Project estimation tool



Views on a Req's Eng. Tool

TOOL

Economic view

Follows-up costs

Purchase price

Project view

Management

User view

Users

Product view

Documents

Reports

Process view

Method definition

Provider view

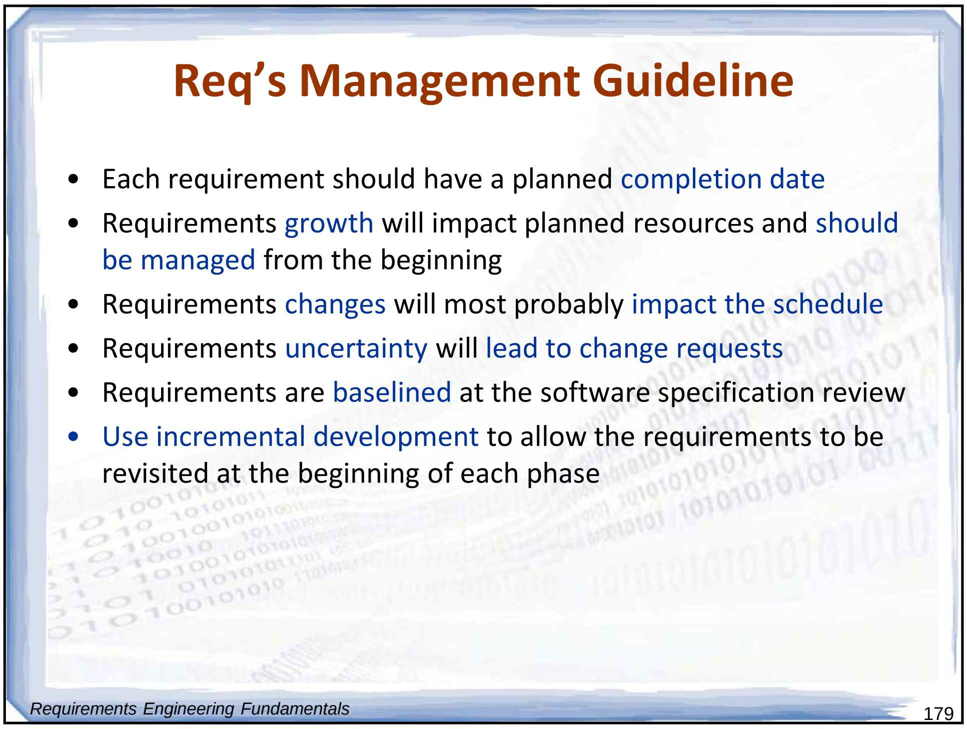
Market position

Support

Technical view

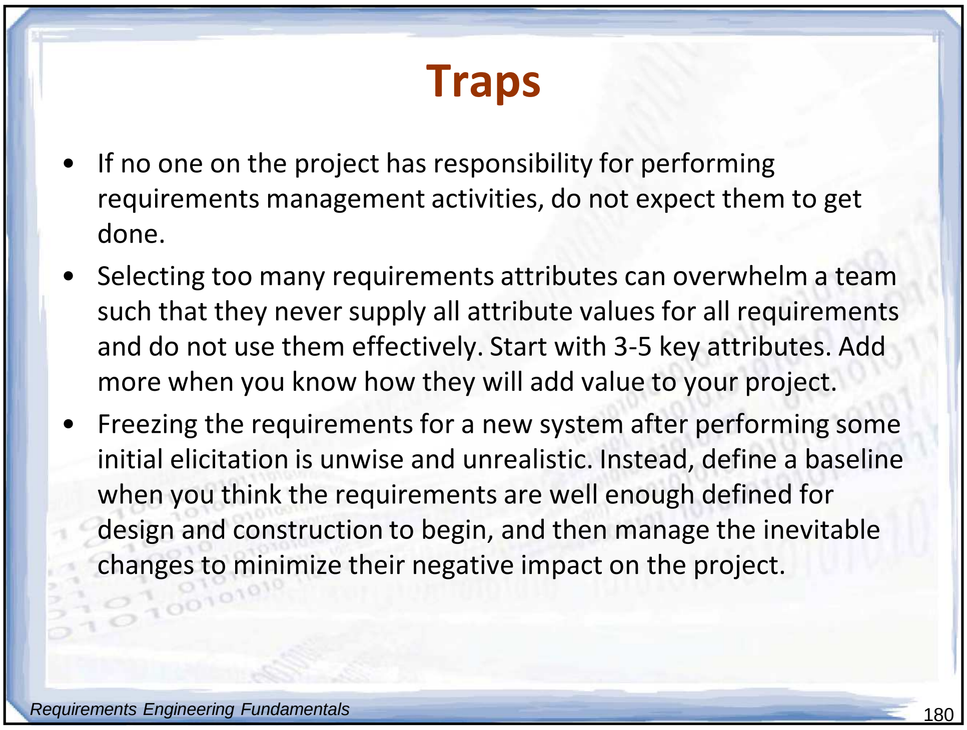
DB, OS

Other tools



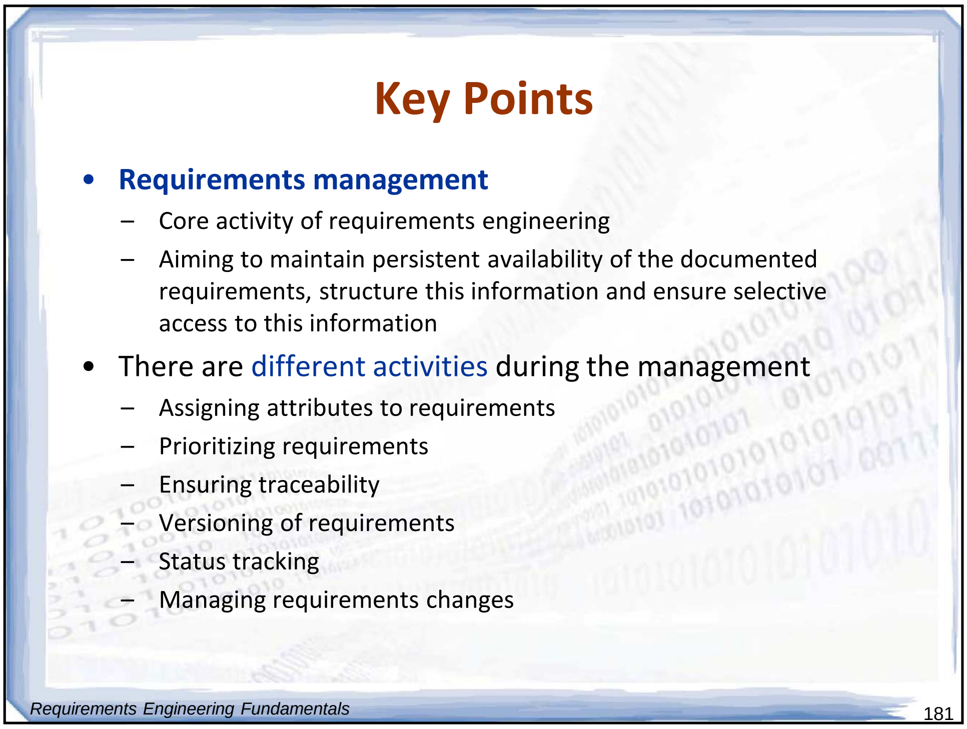
Req's Management Guideline

- Each requirement should have a planned completion date
- Requirements growth will impact planned resources and should be managed from the beginning
- Requirements changes will most probably impact the schedule
- Requirements uncertainty will lead to change requests
- Requirements are baselined at the software specification review
- Use incremental development to allow the requirements to be revisited at the beginning of each phase



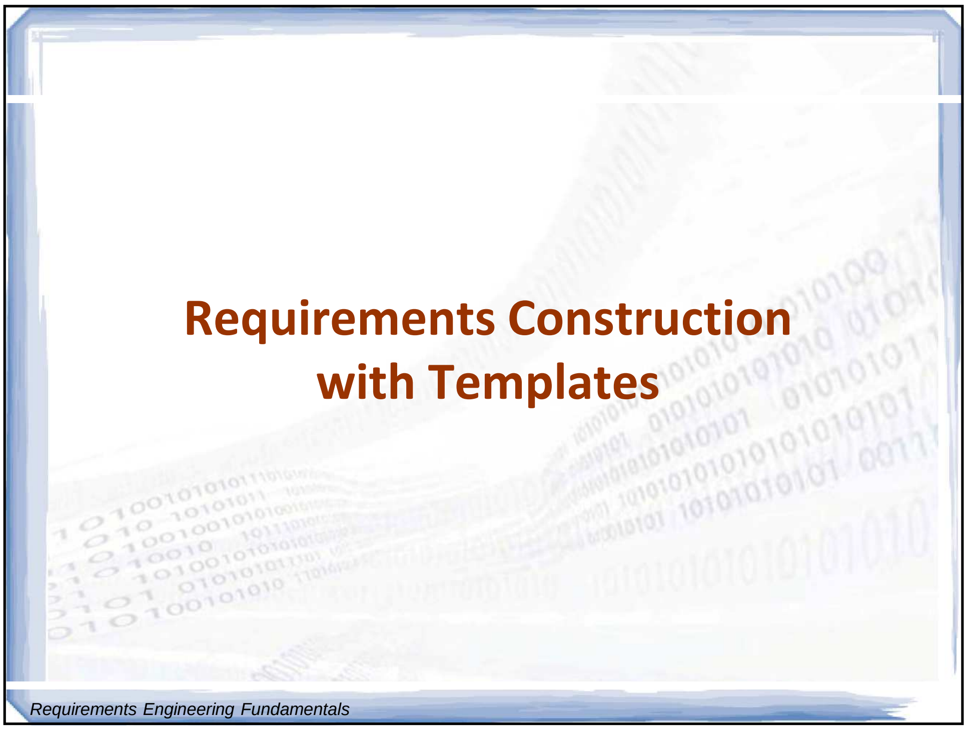
Traps

- If no one on the project has responsibility for performing requirements management activities, do not expect them to get done.
- Selecting too many requirements attributes can overwhelm a team such that they never supply all attribute values for all requirements and do not use them effectively. Start with 3-5 key attributes. Add more when you know how they will add value to your project.
- Freezing the requirements for a new system after performing some initial elicitation is unwise and unrealistic. Instead, define a baseline when you think the requirements are well enough defined for design and construction to begin, and then manage the inevitable changes to minimize their negative impact on the project.



Key Points

- **Requirements management**
  - Core activity of requirements engineering
  - Aiming to maintain persistent availability of the documented requirements, structure this information and ensure selective access to this information
- There are different activities during the management
  - Assigning attributes to requirements
  - Prioritizing requirements
  - Ensuring traceability
  - Versioning of requirements
  - Status tracking
  - Managing requirements changes



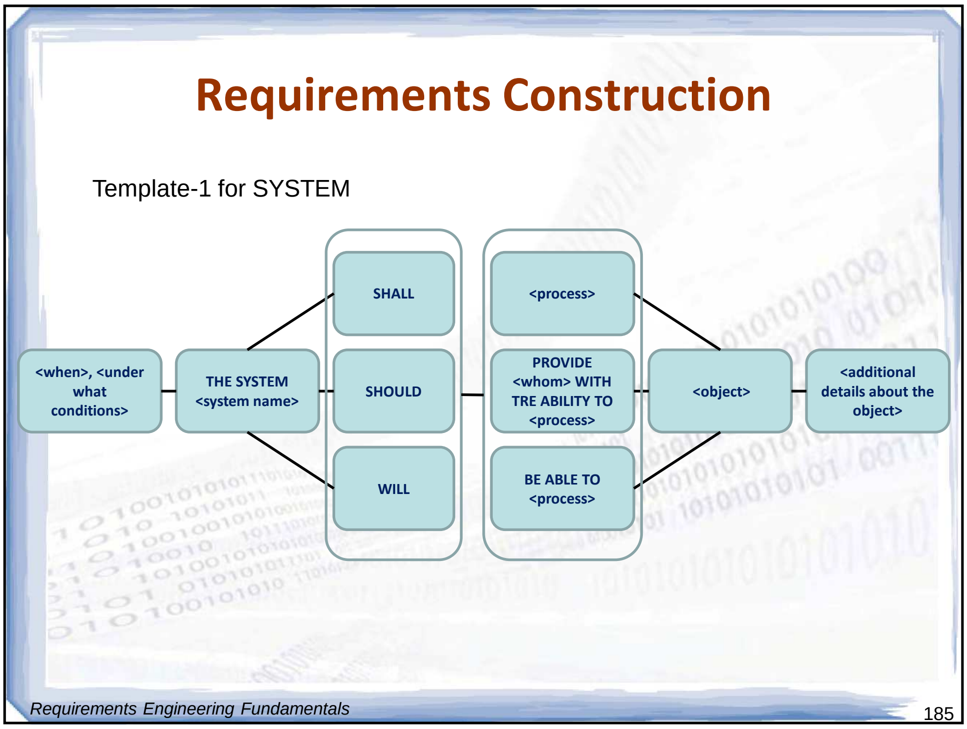
Requirements Construction with Templates

Requirements Construction Using Templates

- A requirements template is a blueprint for the syntactic structure of individual requirements.
- Steps of a correct requirements template application:
  - Determine the legal obligations (SHALL, SHOULD, WILL)
    SHALL – must, SHOULD – not obligatory provision, WILL - desired
  - Determine the required process (<PROCESS>)
  - Characterize the activity of the system
    - Autonomous system activity (<PROCESS>)
    - User interaction (PROVIDE <whom> WITH THE ABILITY TO <process>)
      what the system provides to specific users
    - Interface requirement (BE ABLE TO <process>), i.e. the system reacts while triggering other events
  - Insert Objects (e.g. PRINT what and where)
  - Determine Logical and Temporal Conditions (IF, AS SOON AS, WHEN)

Verbs for process names

- Acquire (megszerez)
- Add
- Adjudicate (megítél)
- Assess (felbecsül)
- Calculate
- Cancel
- Change
- Check
- Conduct
- Control
- Create
- Delete
- Determine
- Identify
- Maintain
- Manage
- Merge
- Modify
- Obtain
- Plan
- Query
- Record
- Receive
- Request
- Remove
- Report
- Reject
- Review
- Roll back
- Select
- Specify
- Submit
- Update
- Validate
- Verify
- View

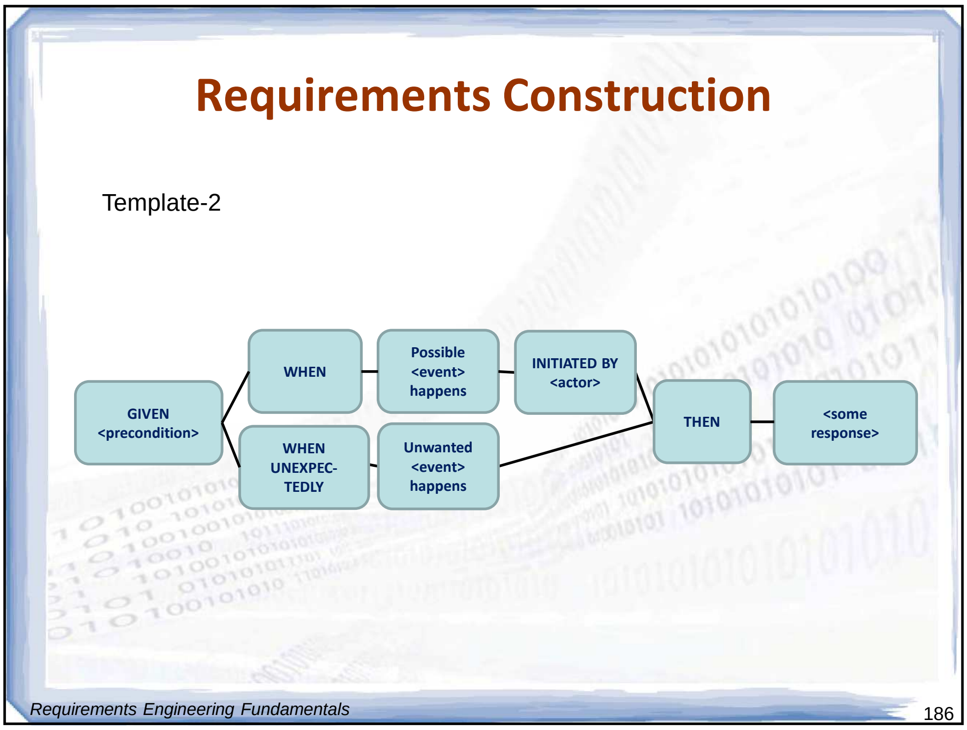


Requirements Construction

Template-1 for SYSTEM

<when>, <under what conditions>
THE SYSTEM <system name>
SHALL
SHOULD
WILL
<process>
PROVIDE <whom> WITH TRE ABILITY TO <process>
BE ABLE TO <process>
<object>
<additional details about the object>

185

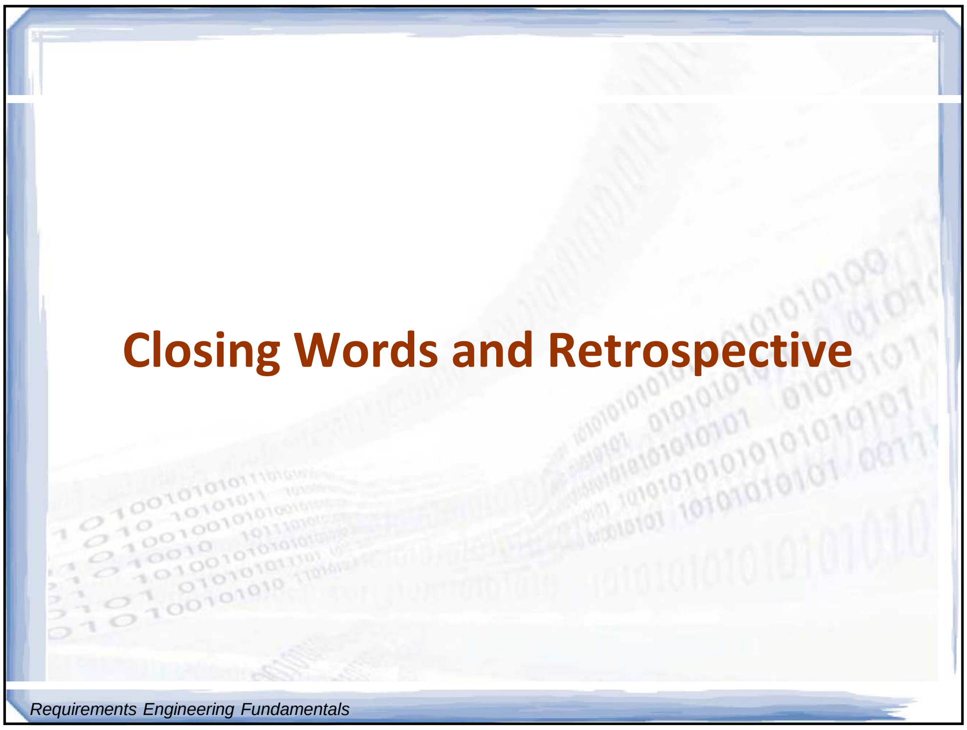


Requirements Construction

Template-2

186

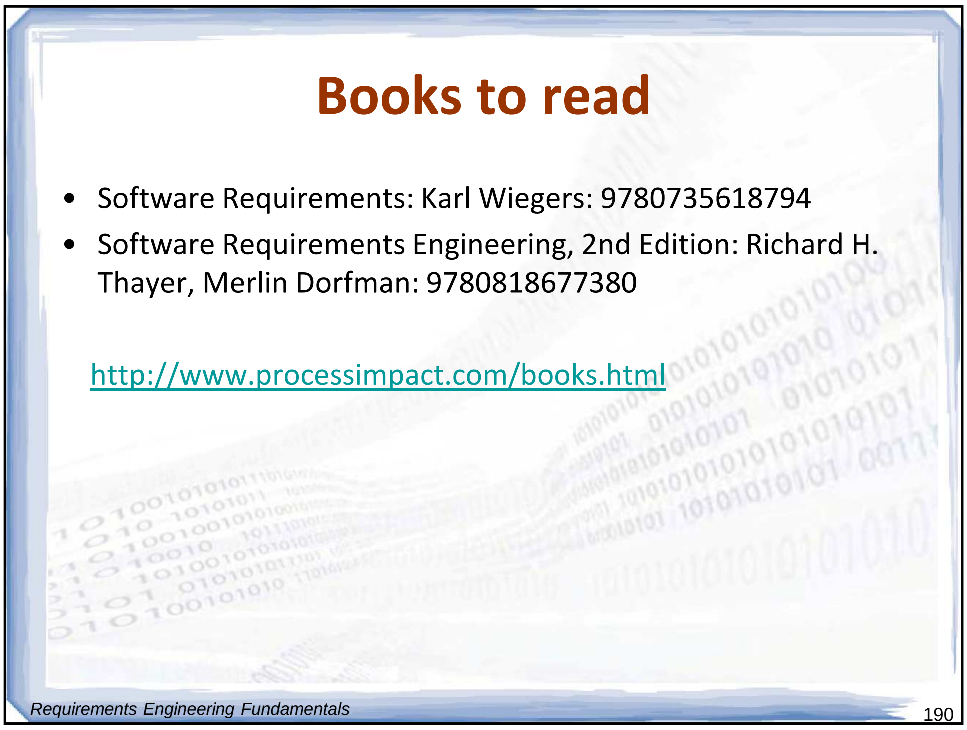
Exercise



Closing Words and Retrospective

Web pages to visit

- www.ireb.org
  - » Professional qualifications (IREB Diploma)
- www.BCS.org
  - » Professional qualifications (ISEB Diploma)
- www.smart-BA.com
  - » Articles
  - » Sample analysis documentation
- www.ModernAnalyst.com
  - » Excellent community
  - » Articles, Forums
  - » Sample analysis documentation
- www.RequirementsNetwork.com
  - » Another excellent community
  - » Articles, Forums
- www.TheIIBA.org
  - » Professional organisation
  - » Professional qualifications (CBAP)



Books to read

- Software Requirements: Karl Wiegers: 9780735618794
- Software Requirements Engineering, 2nd Edition: Richard H. Thayer, Merlin Dorfman: 9780818677380

http://www.processimpact.com/books.html

obrigado
Dank U
Merci
mahalo
Köszi
спасибо
Grazie
Thank you
mauruuru
Takk
Gracias
Dziękuję
Děkuju
danke
Kiitos