Requirements

References:GeekCorps2004

John Vu (CMU)

Introduce

- First activity of the process
- Goal: find out what to build
- Communication between users and developers
  - thus: no complex notations possible (unless scientific domain)
  - mostly natural language
  - contract
- Methods for finding requirements
- Methods for formulating requirements
  - Scenarios, Use Cases, Mockups / Prototypes, Feature Lists
- Stakeholders
  - everybody interested in the product

What is Requirements Management?

What is Requirements Management?

- Process of discovering, documenting, and maintaining requirements
- Systematic, repeatable techniques to make sure Requirements are:
  - Complete
  - Consistent
  - Relevant

What is Requirements Management?

- In plain English,
  - Find out what the user wants
  - Organize this information
  - Document this information
  - Trace this information
  - Manage any changes
  - Meet the users needs
- Establish a process and follow it

What is Requirements Management?

- Most software development organizations claim to do this in some fashion
- But it's usually informal and carried out inconsistently from project to project
- CMM Level 1 vs. CMM Level 2

  = Defined Requirements Management Process

What is a Requirement?

What is a Requirement?

- A software capability needed by the user to solve a problem to achieve an objective
- Requirements must be met in order to ensure the success of a project

Sources of Requirements: Customers

- Interviewing customers
  - the people who pay you
  - other stakeholders
    - users
    - managers
- Problems:
  - customers may not know what they want / need
    - software is abstract and complex
  - may often change their minds
  - may not be able to express their needs in technical terms
    - Communication between computer scientists / regular people
- Techniques
  - mockups, prototypes
  - walkthroughs
  - differences to existing systems

Sources of Requirements: Market

- Evaluating competing products
  - what has been done before?
  - where is a niche for us?
  - but take care not to violate copyrights, trademarks, patents
- Evaluating own abilities
  - what are we better at than the competition?
  - what knowledge, skills do we have?
- Market survey
  - interview consumers
  - difficult to do right; polling companies
  - marketing and advertising: create the demand
  - consider future trends
- Problems:
  - people don't know what they want
    - example: video telephones
  - markets change quickly (UMTS)

Sources of Requirements: Standards

- Standards and interoperating systems
  - system standards, file formats, network protocols
  - usability standards
  - domain standards
- Official standards
  - written by a standards body
    - ANSI
    - ISO (e.g. Unicode)
    - IEEE (e.g. Posix)
- Industry standards
  - Wintel Platform
  - Java, Dot-Net
  - Wimp user interface

Kinds of Requirements

- Functional
  - features
  - user interface
  - input / output
- Non-functional
  - user qualities
    - performance
    - precision
    - reliability
  - developer qualities
    - maintainability
    - reusability
    - interoperability

The Requirements Problem

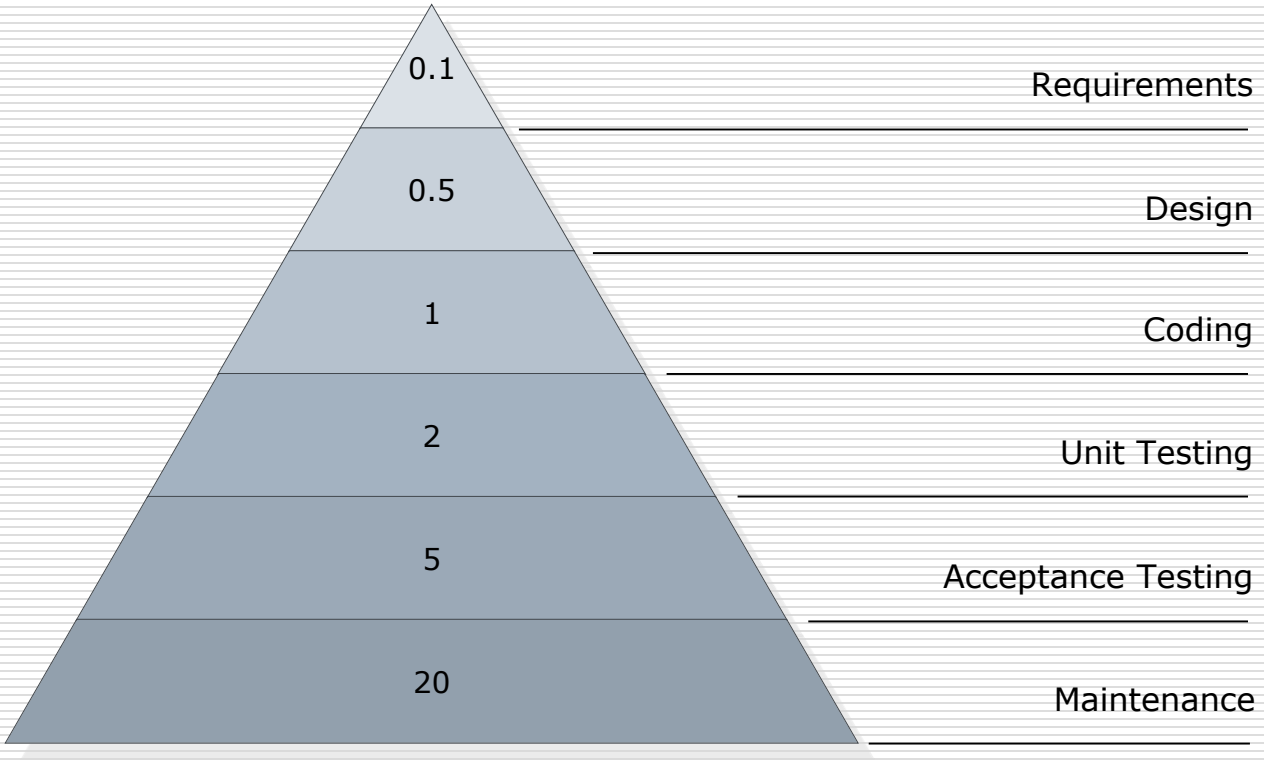
- In the USA, a study found that 31% of all software projects were canceled before completion

The Requirements Problems ?

The Requirements Problem

- Many systems are delivered late and over budget. They often don't do what users wanted or are often not used to full effectiveness.
- The first step to resolving any problem is to identify the *root cause*
- The same study identified the following root causes:
  - Lack of user input
  - Incomplete requirements and specifications
  - Changing requirements and specifications

Incremental Cost of Bad/Missing Requirements



Relative Cost To Repair Defect

Effects of problems in requirements:

- Re-specification
  - Redefine requirements again
- Re-design
  - Change models and dependencies
- Re-coding
  - Re-write code
- Re-testing
  - Change test plan, test case, retest
- Lots of extra effort and lost time/resources:
  - Product Recall / Throw away product
  - Legal liability / Angry Customers
  - Increase Maintenance cost
  - Documentation changes

Good Requirements Management ?

Good Requirements Management

- Prevent:
  - Project Cost and Schedule Overruns
  - Cancellation
- Successful projects have the following "success factors":
  - User involvement
  - Executive management support
  - Clear statement of requirements

Quality of Requirements: Consistency

- Consistency
  - no contradictions
  - the program won't work if requirements are contradictory
- Hard to guarantee
  - large sets of requirements
  - contradictions can be hidden
    - example:
      - Section 1.2 says: "no more than 128MB of memory needed"
        Section 5.8.9 says: "images should be rendered in real time"
      - => is this a contradiction???
- Formal logic: anything follows from a contradiction
- Problem:
  - often difficult to explain contradiction to customers
  - customers may want impossible things

Quality of Requirements: Manageability

- Resources must match requirements
  - can it be done in time?
  - with the money available?
  - with the skills we have?
- Risk management
  - requirements should be prioritized
  - ideally: alternatives in case it doesn't work out
  - often impossible to tell which requirements are possible
- Complexity management
  - don't do everything at once
  - iterative processes
  - prototyping

Quality of Requirements: Specificity

- Be as precise and detailed as possible
- Bad:
  - “program shall be fast”
  - “it takes a number as input”
- Good:
  - “the program shall give a response in less than 1s”
  - “it takes a 16-bit signed integer as input”
- Definitions
  - are often a good idea
  - example: “by 'Number', we always mean a 16-bit signed integer”
  - defined terms are often capitalized
- Rules about definitions
  - no circles

Quality of Requirements: No Implementation Bias

- Implementation bias:
  - giving information about the design
  - giving information about the code
- Use terminology of the domain
  - not technical terminology
  - you want to focus on WHAT
  - leave HOW for later
- Bad examples:
  - “store the checked-out books in an array”
  - “calculate the square root with Newton's algorithm”
- Good examples:
  - “the library knows which books are checked out”
  - “return the square root with 5-digit precision”

Requirements must be …

- **Correct**: They must represent the real need of customers and users.
- **Complete**: They include all the needed elements.
- Functionality, external interfaces, quality attributes and design constraints.
- **Clear**: They can be understood in the same way by all stakeholders with minimum supplementary explanation.
- **Concise**: They are stated simply, in the most minimal way possible to be understandable.
- **Consistent**: They do not conflict with other requirements.

Requirements must be …

- **Relevant**: They are necessary to meet a business need, goal, or objective.
- **Feasible**: They are possible to implement.
- **Verifiable**: There is a finite, cost effective technique for determining whether the requirement is satisfied.

The Software Team

- Requirements management touches every team member
- Effective requirements management can only be accomplished by an effective software team
- The capability of the team has the greatest impact on software production

Team Skills for Requirements ?

Team Skills for Requirements

- Six team skills for requirements management:
  - Analyzing the problem
  - Understanding user needs
  - Defining the system
  - Managing scope
  - Refining the system definition
  - Building the right system

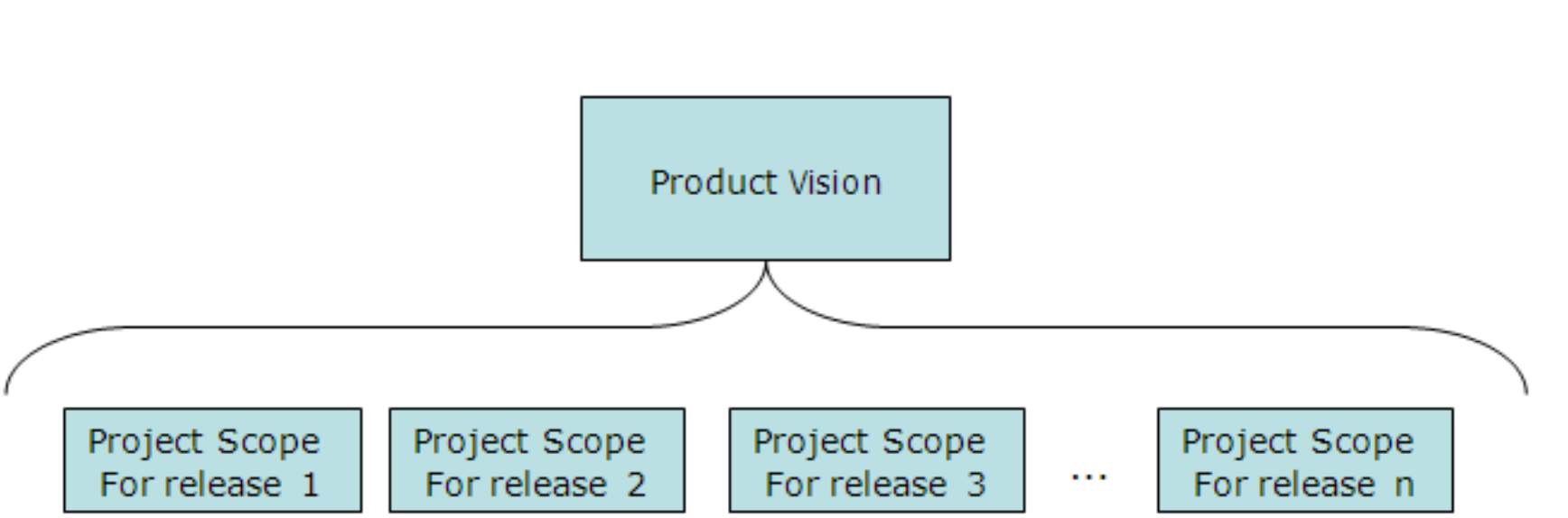
Requirement Engineer Skills:

- Listening skills
- Interviewing & questioning
- Analytical skills
- Facilitating skills
- Observation skills
- Writing skills
- Organizational skills
- Modeling skills
- Interpersonal skills
- Communication skills

From Vision To Project Scope

- The product vision aligns all stakeholders in a common direction. The vision describes what the product is about and what it eventually could become.
- The project scope identifies what portion of the long-term product vision the current project will address.
- The statement of scope draws the boundaries between what is in and what is out (the project limitation) as the details of a project's scope are represented by the requirements baseline.
- The project scope is the sum of all the project's deliverables and services to be provided to customers & users.

Vision & Scope



A Product Vision Statement

- A vision statement is a concise statement that defines the what, why, and who will buy or use the final product from the business point of view.
- The vision statement will help you to:
  - Ensure that the product definition aligns with the business goals and objectives.
  - Identify stakeholders – Who will buy or use the product?
  - What the product will do for its stakeholders?
  - What are the reasons to buy or use this product?
  - How can this product be distinguished from others?
  - Describe the state of the business – its strengths, weaknesses.
  - Describe the opportunities and risksto the business.

How To Write Vision Statement

- Define the following terms:
  1. **Target customers**: Describe the people who will use or buy the software.
  2. **Statement of need**: Describe what the target customer does or what the software will help them with.
  3. **Product name**: Provide the name of the product that you will create.
  4. **Product category**: Describe the type of product that you are building. Product categories might include internal application, COTS software, embedded software, business and finance software, game software, complex software, etc

How To Write Vision Statement

- Define the following terms:
  5. **Benefit Statement:** Describe what the product will do for the target customers or the justification for buying the product.
  6. **Competitive analysis:** Describe the key competing product available or the system or process that the product will replace.
  7. **Product differentiation**: Explain the differences between the product you are building and the competition.
- **Review all writings** to ensure that they define the what, why, and who from the business point of view.

Follow This Template …

- Insert statement to <>
- **For** <target customer> **Who** <statement of need> **the** <product name> **is a** <product category> **that** <benefit statement> **unlike** <competitive analysis>, **our product or service** <product differentiation>.
- Review and check if the vision aligns with the business goals and objectives.

Example 1

- **For** a software company **who** provides outsourcing services to global customers, **the** ABC company **is the** most advanced outsourcing company **that** has successfully provided services to hundreds of global clients. **Unlike** others who do not provide total solutions, **our services** have helped many clients achieve significant cost savings by integrating all applications into a cohesive total business solution that brings quality and efficiency for the entire operation.

Example 2

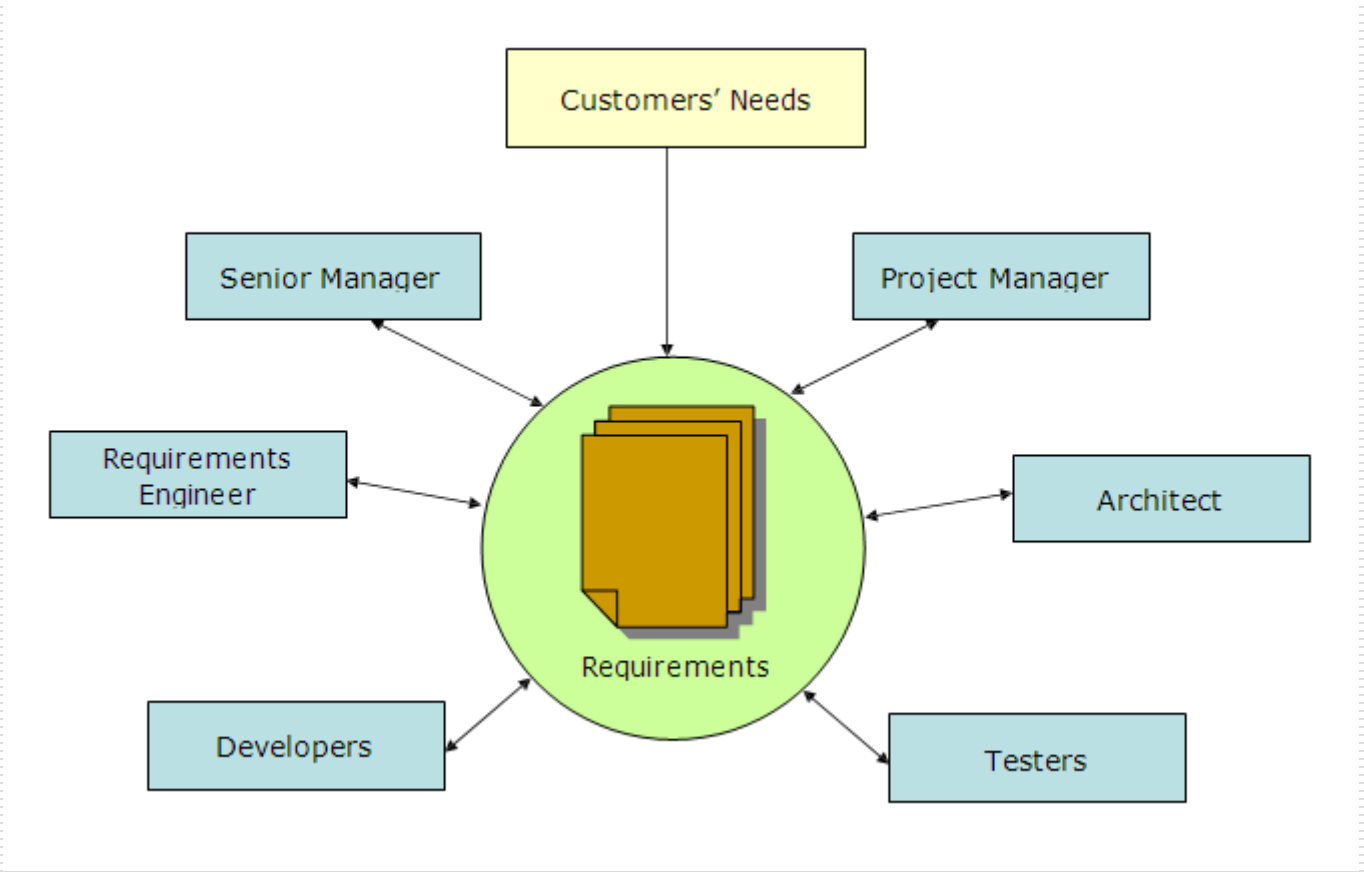
- **For** software users **who** need to request information from databases, **the** ABC Data Management **is** an information system that provided a single point of access to all data and information stored anywhere in company's multiple databases. **Unlike** other systems, **our system** not only locates, selects and provides information, but also generates required reports when needed, for use for further investigation and documentation.

Who will involve in Req Management ?

Requirements Gathering Team:

- Account Manager
  - Kickoff Meeting
    - Debriefs team on general business goals, introductions, organizational background etc.
    - Introduces Requirements Team to Customer counterparts
- Facilitator
  - Project Manager
    - Defines agenda
    - Keeps meeting on focus
    - Manages and Negotiates with Clients
- Note Taker
  - Responsible for capturing requirements
    - Project Manager/Team Member
- Domain Experts/Team Leads
  - Technical Architect
  - Software Lead
  - UI designer
  - Area Specialists (Expert on Platform X, Feature Y etc)

Requirements Development



Senior Manager

- Allocates resources (people, materials, budget).
- Ensures project goals and objectives align with business goals.
- Defines and approves overall vision & scope.
- Makes decisions about scope of project & schedules.
- Resolves conflicts in requirement priorities

Project Manager

- Coordinates users involvement.
- Acts as a liaison between project team and business management.
- Ensures that requirements engineers and architects have the resources, tools, training and knowledge to work on requirements activities.
- Defines business requirements and product vision.
- Oversees requirements prioritization.
- Monitors progress of requirements development.
- Reviews requirements documents to ensure that they adequately and completely represent user's needs.

Requirement Engineer

- Selects elicitation techniques and coordinates requirements activities.
- Collaborates with stakeholders to develop requirements.
- Defines user requirements, creates models and documents requirements.
- Verifies that requirements are correct, complete and consistent.
- Transforms user requirements to software requirements specification.

Architect

- Provides details about user needs, business rules and data.
- Represents the needs of users who may not be able to participate.
- Identifies integration issues between hardware, software, and systems.
- Ensures requirements align with product vision.
- Consults with other experts who have relevant requirements knowledge.
- Reviews requirements document to ensure that it adequately and completely represents user's needs.

Developer & Tester

- Provides details about design constraints and suggestions regarding the feasibility of nonfunctional requirements.
- Supports the writing of software requirements specifications.
- Reviews all requirements documents.
- Reviews software requirements specifications to ensure that they can be transformed into design.
- Ensures that requirements can be tested

Key project roles

	Requirement Development			Requirements Management
	Define Business Requirements	Develop User Requirements	Specify Software Requirements	
Senior Manager	Owner	Reviewer	Approver	Approver
Project Manager	Producer	Reviewer	Reviewer	Reviewer
Requirements Engineer	Reviewer	Producer	Producer	Producer
Architect	Reviewer	Approver	Owner	Owner
Developers	Reviewer	Reviewer	Reviewer	Reviewer
Testers	Reviewer	Reviewer	Reviewer	Reviewer

Owner: Provides complete and accurate information.
Approver: Approves and authorizes requirements.
Producer: Creates and maintains requirements.
Reviewer: Stays informed and provides feedback.

Some Key Stakeholders:

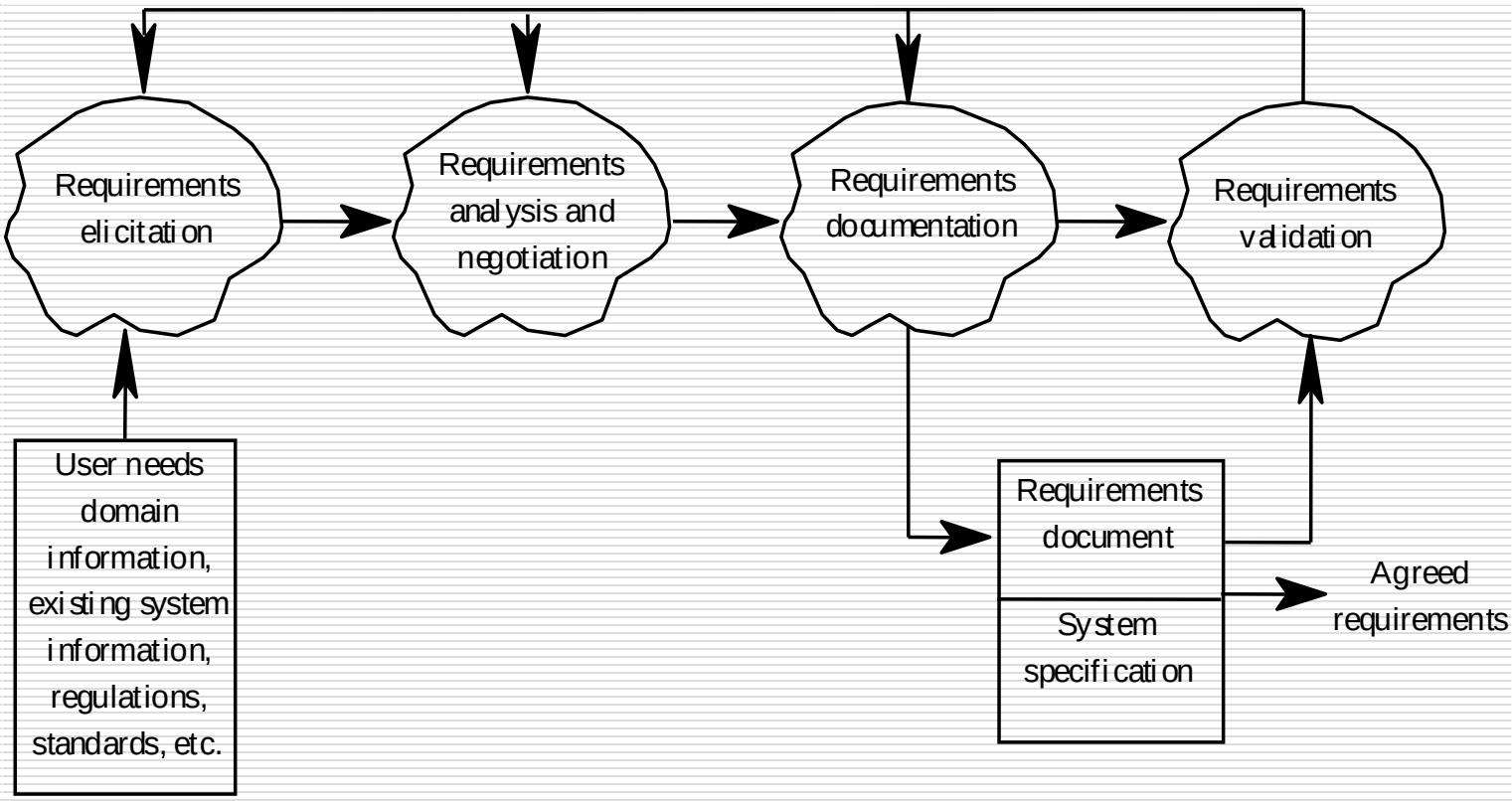
- **System End-users**
  - Who will use the system after it has been delivered
- **Customer**
  - Business requirements are being met
- **Domain experts**
  - Give essential background information about the system application domain
- **Project Manager**
  - Champion who manages requirements process. Makes sure needs are address.
  - Negotiates between stakeholders
- **Software engineers**
  - Responsible for system development
- **QA Test Engineer**
  - Responsible for testing system and ensuring quality

How much does it cost ? (Req Management Process)

Expected Effort of Requirements:

- Project Cost:
  - Expect to spend about 15% (up to 30%) of project schedule on Requirements Process
- More time spent on requirements, more time is saved in:
  - Design
  - Implementation
  - Testing

Sample Process:



Elicitation⇔Analysis⇔Negotiation:

- Characteristics:
  - Iterative
    - Step by Step, not one big step.
  - Inter-dependent processes
    - Each step depends on each other
  - Repeated
    - Each step has to be repeated
- Sample Outputs
  - Final Proposal
    - Agreed Requirements
  - Vision Document
  - Software Requirements Specification
  - High Level Design Models
    - UML
    - Network Diagrams
    - Object Models

Elicitation:

- Process of extracting requirements from Customer
  - Interview and Ask Questions:
    - Business requirements
    - Functional requirements
    - Other requirements
  - Use Tools:
    - Whiteboard (draw diagrams to help customer understand)
    - Use samples (other existing applications), company products as point of reference
    - Use UI diagrams to make things easier to understand
    - Simple prototypes
  - Take notes:
    - Can be rough notes, organize into formal documentation later

Parallel Tasks:

- Research
  - Potential Solutions/Approaches (platforms (J2EE vs .NET), existing software (DB, App server packages), hardware configuration that can solve customer problem
- Prototyping
  - Technical:
    - Rapid prototyping to understand technical challenges
      - Could be performed by Engineers not on Requirements Team who have spare time
  - UI Mockups
    - Designer can create mockups or use existing samples to help Customer understand application behavior
    - Sometimes useful for non-technical Clients
- Result:
  - Better understanding of technical challenges
  - Better understand effort required
  - Head start on design, implementation

Some good practice guidelines:

- Define a standard document structure
- Uniquely identify each requirement
- Track and trace.
  - Who originated this requirement?
  - Who changed it?
- Use checklists for requirements analysis
- Use prototyping to animate requirements
- Reuse requirements

Who will use requirements documents?

Who will use requirements documents?

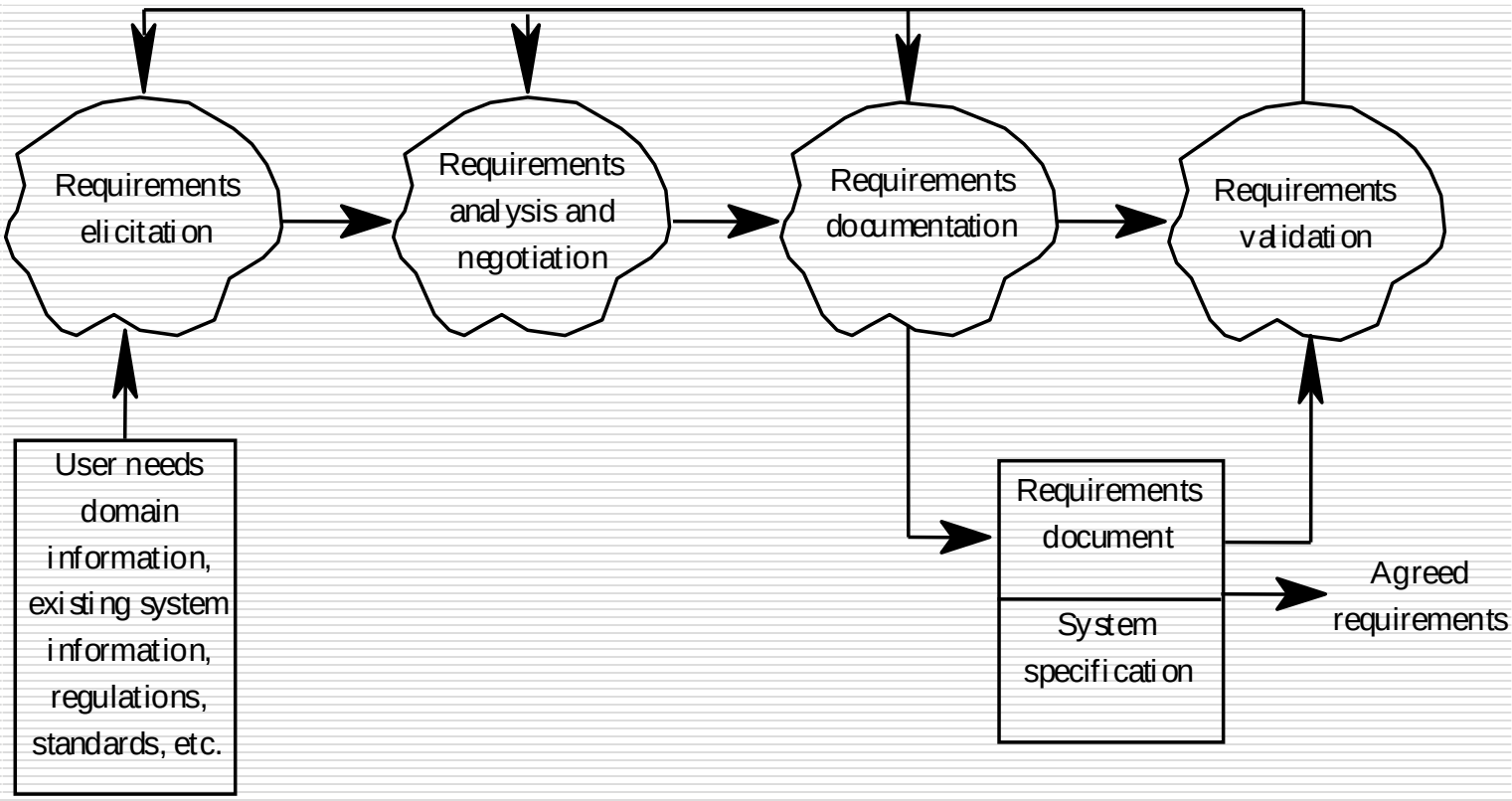
- System customers
  - Specify the requirements and read them to check they meet their needs
- Project managers
  - Use the requirements document to plan a bid for system and to plan the system development process
- System engineers
  - Use the requirements to understand the system being developed
- QA test engineers
  - Use the requirements to develop test plan/test cases for the system
- System maintenance engineers
  - Use the requirements to help understand the system

What are the challenges ?

Challenges:

- Customers do not know what they want
  - It is the team's job to elicit requirements and validate them with the customer
- There is a natural tendency for requirements to change over time
  - Track and Validate
  - Change control management
- The process of designing and building a system inherently raises new requirements
  - Track and Validate
  - Change Control management

Sample Process:



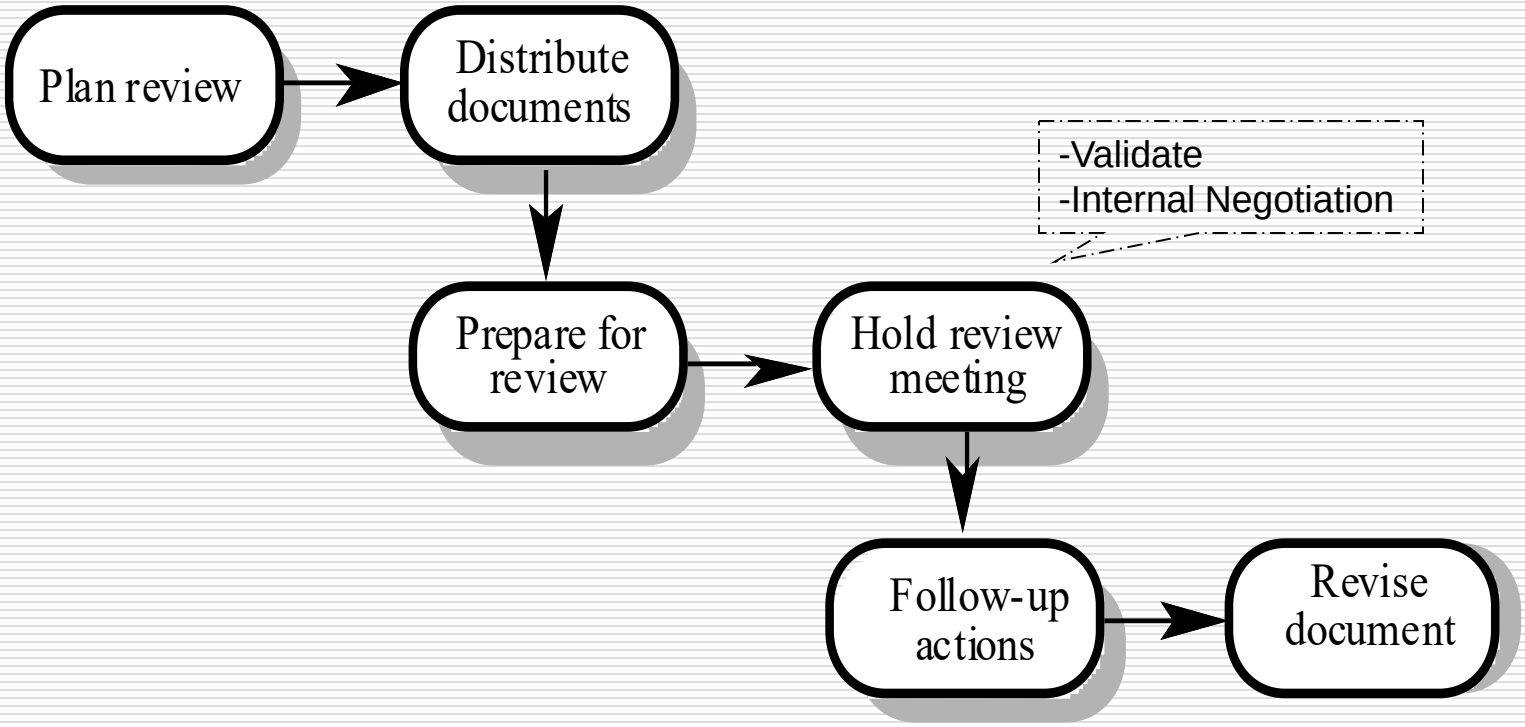
Validation:

- Checks that the requirements document is an acceptable description of the system to be implemented
- Checks a requirements document for:
  - Accuracy
    - Is this what Customer really needs?
    - Is this what Company is agreeing to do?
  - Completeness and consistency
    - Enough information to know what to implement?
  - Conformance to standards
  - Requirements conflicts
    - Does one requirement conflict with another?
  - Technical errors
    - Impossible to implement? Does not reflect real-world conditions of legacy system or 3rd party platform?
  - Unclear requirements
  - Testability

How to Validate Requirements?

- Validation
  - Making sure the requirements are right
    - desired
    - consistent
    - complete
- Discussing them with users
  - Showing them mockups, use cases, feature lists
  - Explaining them to the user
- Brainstorming
  - Is anything missing?
  - Creativity needed
- Role Playing, Walkthroughs
  - Developers play users
  - Go through use cases step-by-step, using mockups

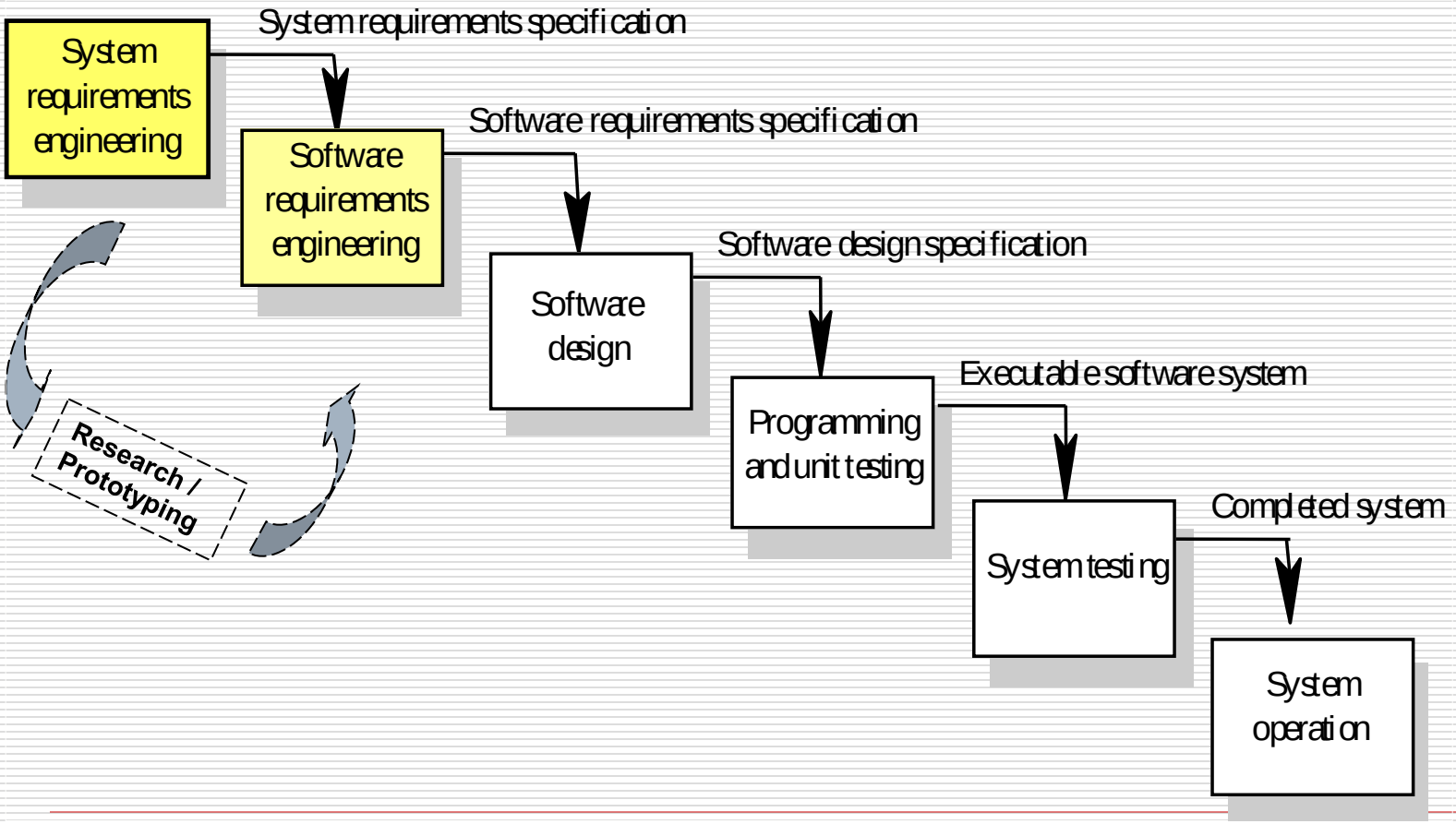
Requirements Review Process:



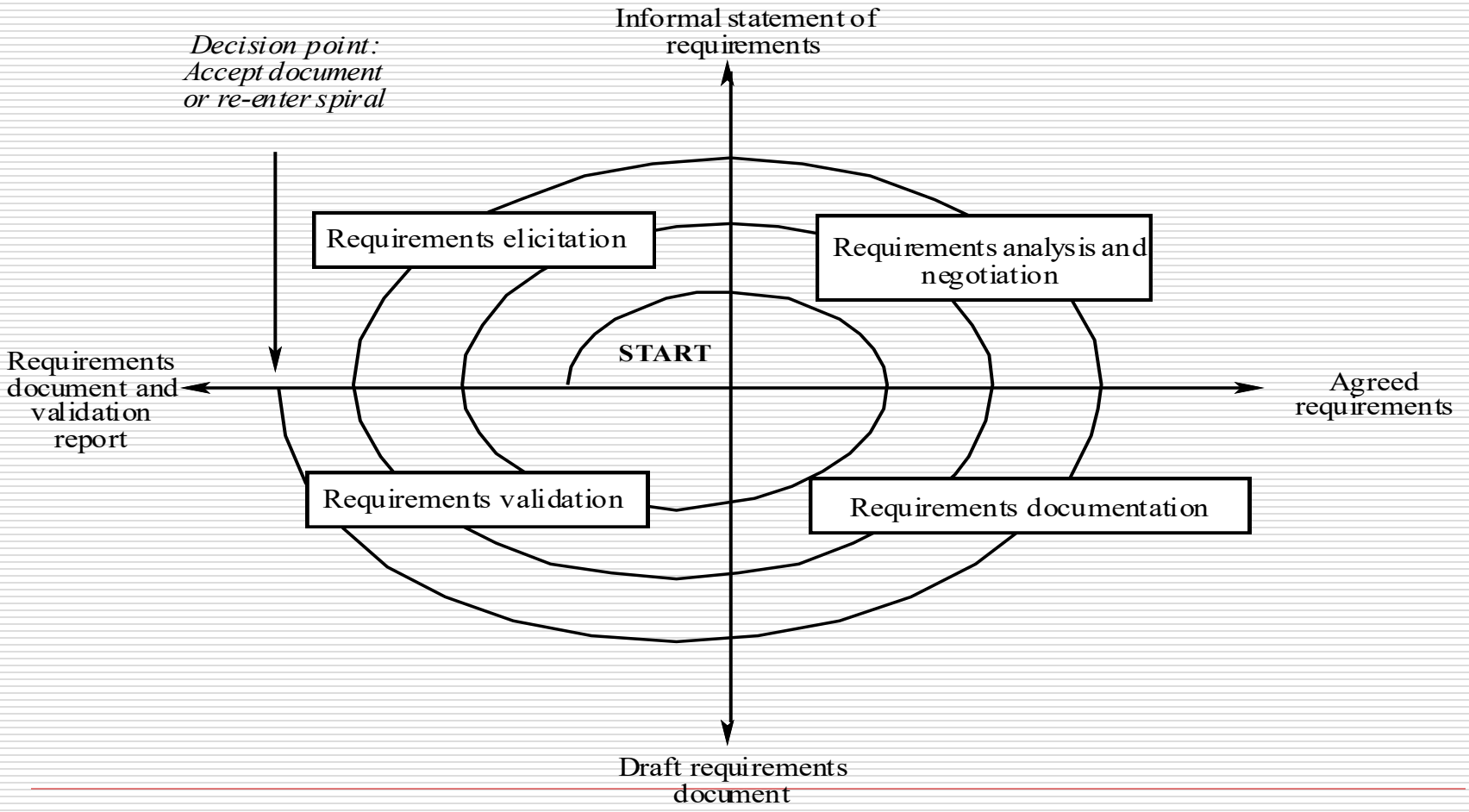
Negotiation:

- Prioritization
  - “Must Have”
    - Customer cannot meet business requirement without
  - “Need to have” (but not absolutely necessary)
  - “Good to Have”
- Sequencing
  - Which features first? Any critical paths?
  - Milestones
  - Phases
- Scheduling
  - How much time needed?
  - Customer requirements to launch
  - Service Provider’s requirements to implement
- Negotiation w/ Client
  - Defer “Good to Have” features to meet Client Deadline
  - Costs
  - Break large project into Phases

Software Requirements View:



Manage risks by repeating analysis, validation, documentation as part of process:



Analyzing the Problem

- Development teams spend too little time understanding
  - the real business problems,
  - the needs of the users and other stakeholders, and
  - the nature of the environment in which their applications must thrive
- Developers tend to forge ahead, focusing on the technology they will use in the solution

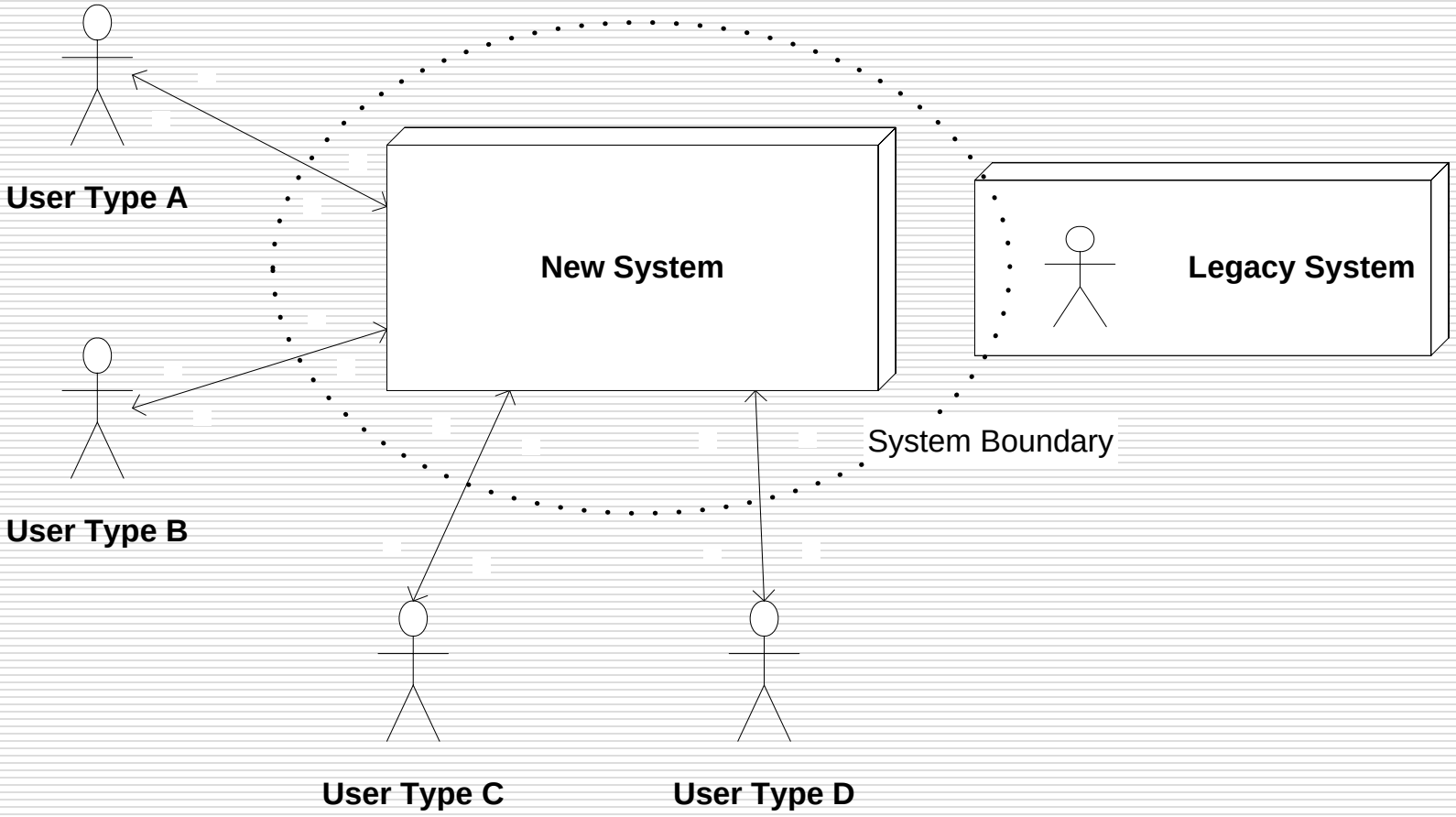
Analyzing the Problem

- Five steps of problem analysis:
  - Gain agreement on the problem definition
  - Understand the root cause of a problem
  - Identify the stakeholders and the users
  - Define the solution system boundary
  - Identify the constraints imposed on the solution

Analyzing the Problem

- Identifying the users and stakeholders (actors) is a key step:
  - Who will supply, use, or remove information from the system?
  - Who will operate the system?
  - Who will perform any system maintenance?
  - Where does the system get its information?
  - What other external systems will interact with the system?

Analyzing the Problem



Analyzing the Problem

- Constraints restrict the degrees of freedom available in the solution
  - Schedule
  - Budget
  - Operating systems
  - Databases
  - Purchased software
  - Technical issues

Understanding User Needs

- Techniques for eliciting user needs can range from simple and inexpensive (interviewing) to quite technical and somewhat expensive (prototyping)
- No one technique is perfect, but exposure to a variety of techniques and experience using them will help determine which are most effective

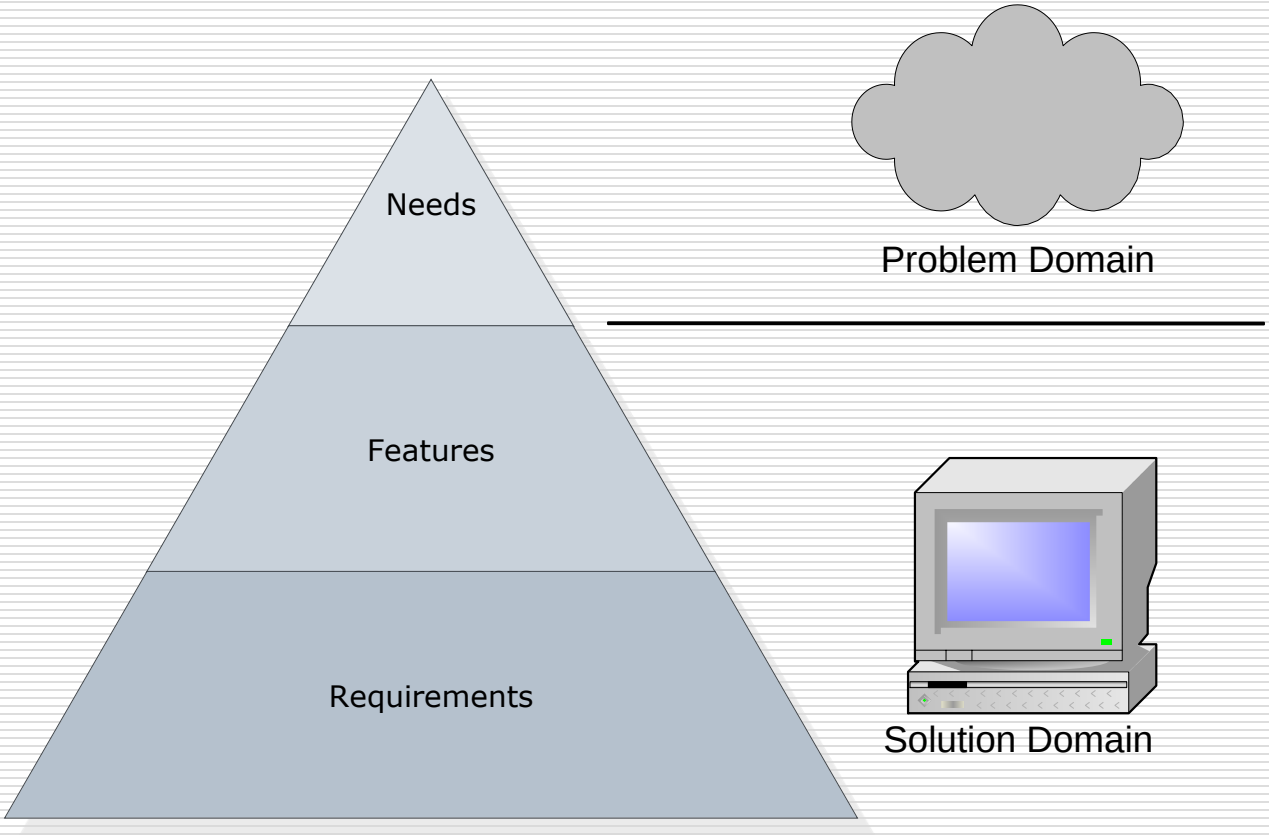
Understanding User Needs

- The “Yes, but” syndrome (hội chứng)
  - Users didn’t understand what was described earlier
  - Once they interact with production-level code, they discover it’s not exactly what they expected
  - Need to apply techniques that get the “buts” out sooner rather than later

Understanding User Needs

- Software development teams struggle with determining when requirements elicitation is “complete”
- Must be able to say at some point “we have discovered enough”
- Identifying all stakeholders in problem analysis should prevent any undiscovered requirements

Understanding User Needs



Understanding User Needs

- If a team doesn’t understand the need behind a feature, a risk exists
- Features are a high-level, convenient way to describe functionality
- Features are helpful for early product scope management and related negotiation and trade-off processes
- Investment may just be a list of phrases in natural language

Understanding User Needs

- Examples of features:

Application Domain	Example Feature
Inventory control System	Provide up-to-date status of all inventoried items
Defect tracking system	Provide trend data to assess product quality
Payroll system	Report deductions-to-date by category
Shrink-wrap application	Support English, German, and Vietnamese

Understanding User Needs

- Keep the feature set manageable, around 50 or so
- Start making decisions: “defer to a later release”, “implement now”, “reject completely”, or “investigate further”
- Easier to scope the project at this level rather than at requirements

Understanding User Needs

- Attach attributes to features to track, prioritize, and manage them:
  - Identifier
  - Status
  - Priority
  - Risk
  - Stability
  - Assigned to

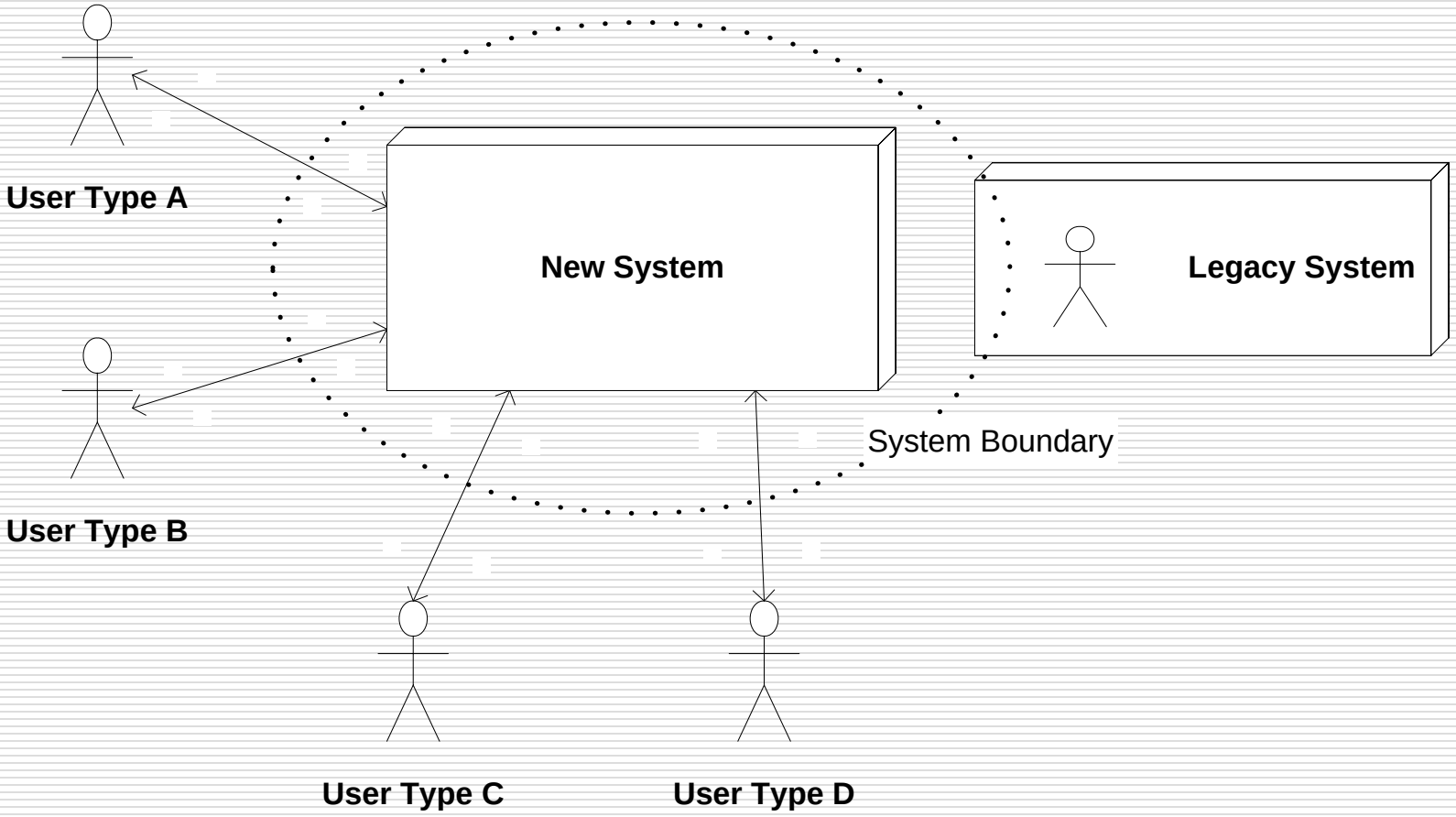
Understanding User Needs

- Techniques for eliciting user needs:
  - Interviewing
  - Workshops / User groups
    - Brainstorming and idea reduction
  - Storyboarding / UI Mockup
  - Use cases
  - CRC Cards
  - Prototyping

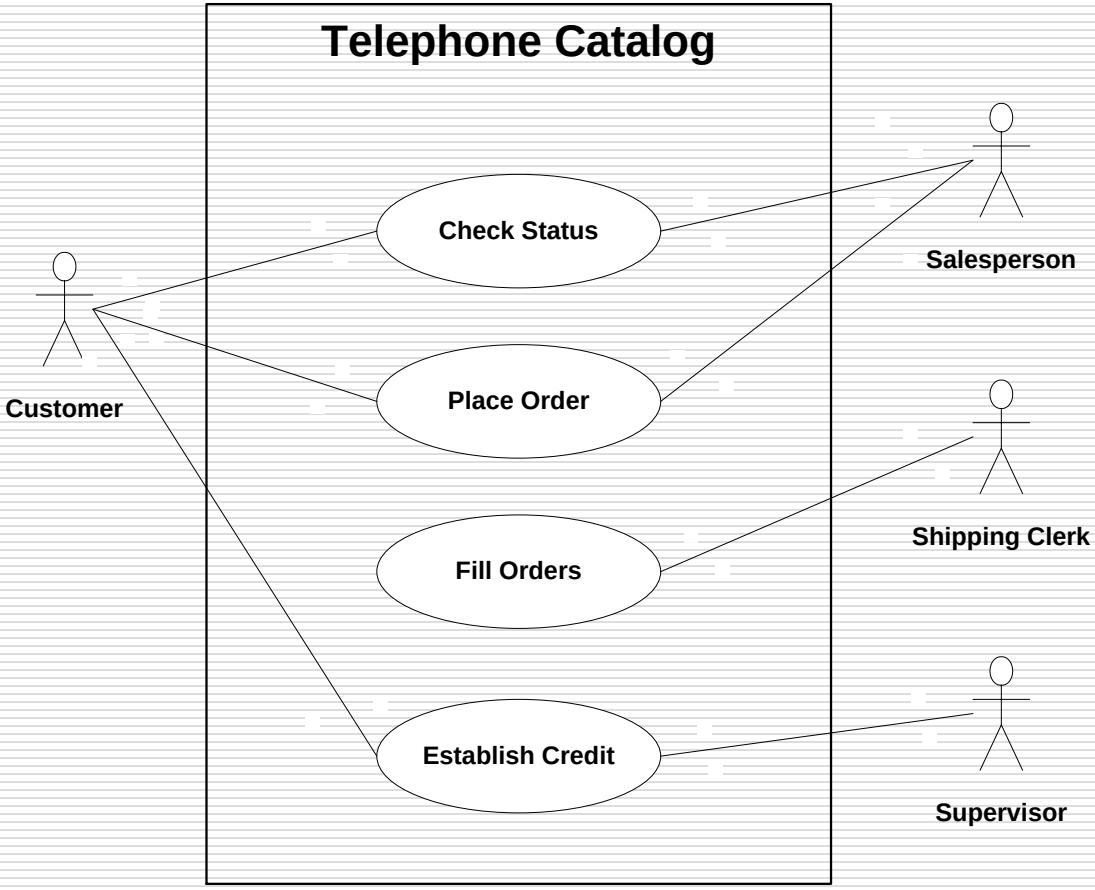
Applying Use Cases

- The use case technique is integral to Object-Oriented Software Engineering, a methodology developed by Ivar Jacobsen
- A use case describes a sequence of actions a system performs that yields a result of value to a particular actor
- Use case modeling parallels problem analysis

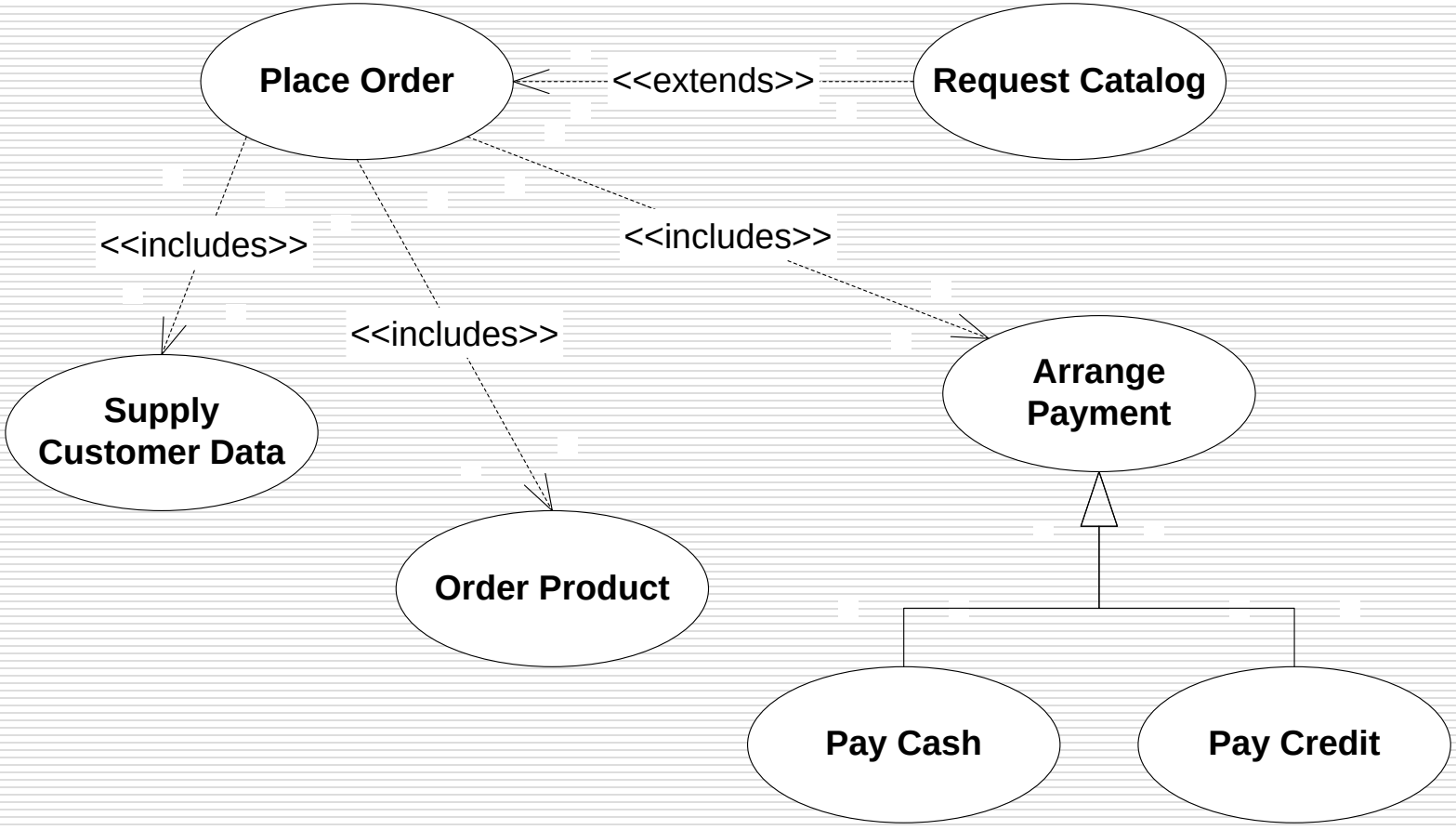
Applying Use Cases



Applying Use Cases



Applying Use Cases



Applying Use Cases

- In addition to a graphical representation of a use case, a textual description is useful
- Usually includes a name, identifier, author, summary, description of flow, open issues, links to other documents
- At this point, it may be useful to begin defining screens and other user interface elements

Applying Use Cases

UC007: Check Status

Primary Actor: Customer

Summary: A registered customer logs into website to check the status of an order

Flow:
1. Customer logs into site using Screen 14
2. Successful login displays Screen 15
3. Customer clicks button “View History” to see a list of orders, sorted by date
4. Customer selects item in list and then clicks button “View Status” to see Screen 24

Author: Nguyễn Hùng Sỏn

Reviewer: Nguyễn Tố Uyên

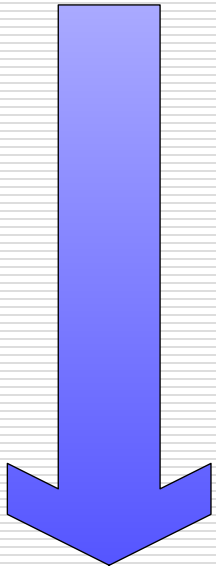
Applying Use Cases

- Use cases are not helpful for finding non-functional aspects of system requirements
  - Usability
  - Reliability
  - Performance
- At the end of this process, the system behavior should be well understood

Understanding User Needs

- The goal is to move from the natural language of the user to the technical language of the software team
- The team needs to be proactive in pushing user needs forward

Abstract Natural Language



Concrete Requirement

So far…

- We've developed the skills that focus on analyzing the problem before investing any serious effort in the solution
- We've understood that a variety of techniques are useful for understanding user needs, which live at the top of the requirements pyramid

Defining the System

- For any substantial system, requirements must be captured in a document, database, model, or tool
- For larger projects, a single document is usually not a good solution:
  - Systems are complex
  - Some documents are written in parallel
  - Business goals should be separated from technical goals

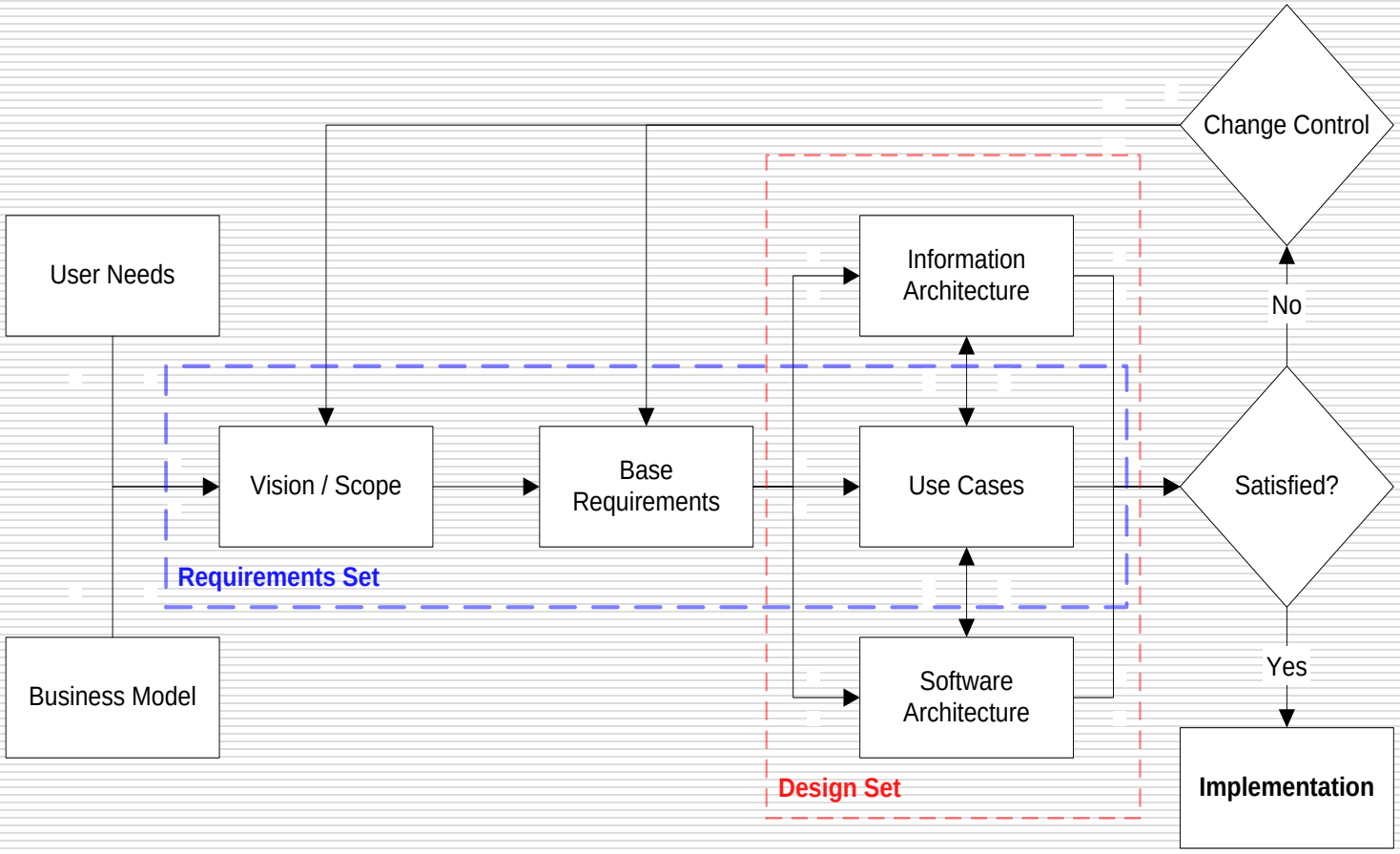
Defining the Vision

- Vision: A general description of both the problem and the solution
  - [Why?] Customer Business Requirement
  - [What?] Software Solution
- The Vision document combines modest elements of both marketing and software requirements
- Every project should have a Vision document, the first “official” document of the process

Defining the System

- The Vision document:
  - Identifies the business opportunity
  - States the problem to be solved
  - Identifies users and stakeholders
  - Provides a product overview
- The Vision document is “organic”
  - A living document that grows and changes with the project

From Vision to Requirements



Functional vs. Non Functional Requirements:

- Functional requirements
  - Use Cases
  - System Behavior
- Non functional requirements
  - Security
  - Speed
  - Usability
  - Other interacting systems
- Both are part of Requirements Process

Requirements Examples (Library System):

1. “The system shall maintain records of all library materials including books, serials, newspapers and magazines, video and audio tapes, reports, collections of transparencies, computer disks and CD-ROMS.”
2. “The system shall allow the users to search for an item by title, author, or ISBN.”
3. “The system’s user interface shall be displayed using Internet Explorer 5 and above”
4. “The system shall support at least 20 transactions per second.”
5. “There should be no more than 3 clicks from homepage to reach search results.”
6. “The access permissions for system data may only be changed by the system’s data administrator”

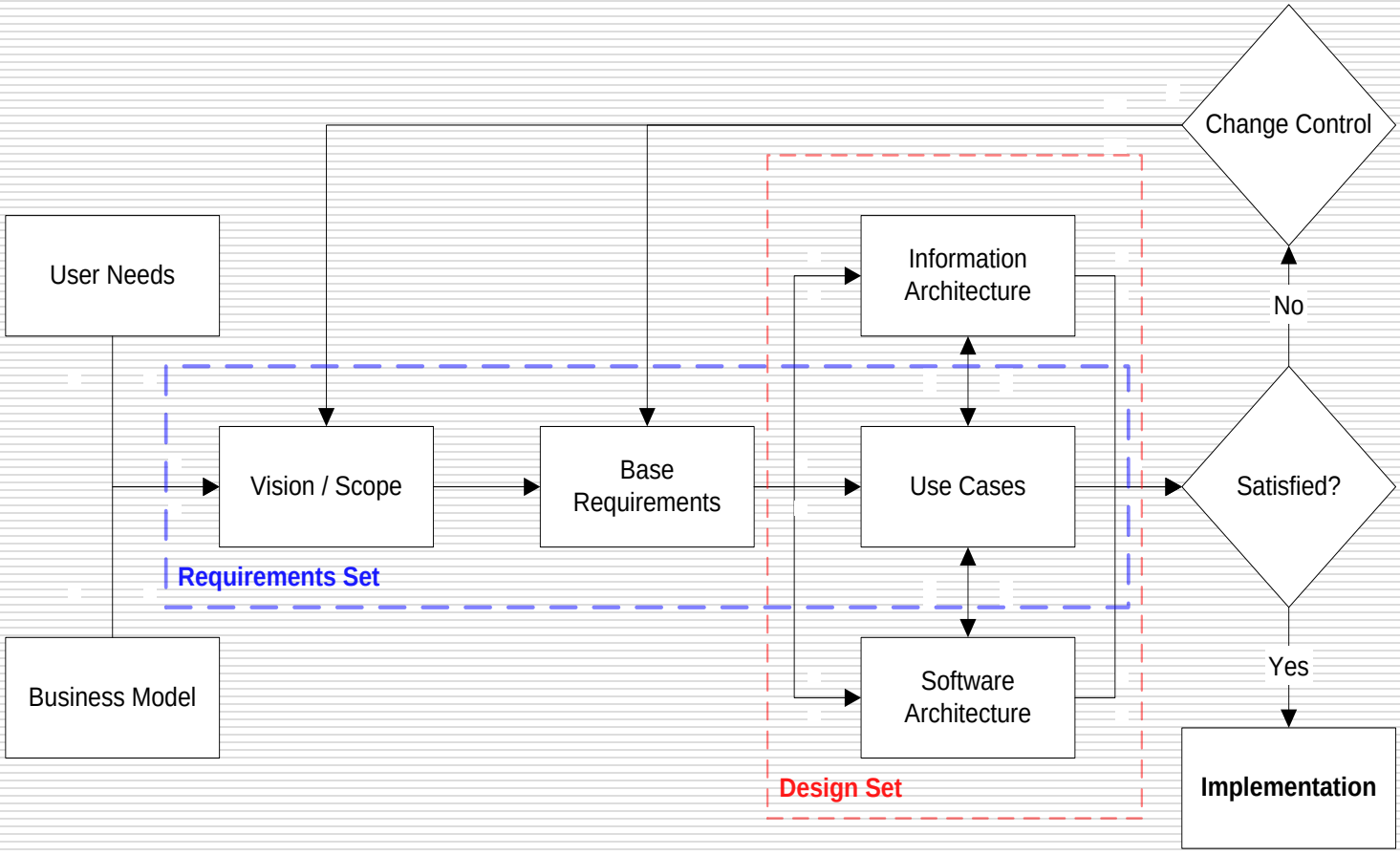
The previous list contained different types of requirements:

1. *"The system shall maintain records of all library materials…"*
   - [Very general requirements that set out in broad terms what the system should do]
2. *"The system shall allow the users to search for an item by title, author, or ISBN."*
   - [Functional requirements]
3. *"The system's user interface shall be displayed using Internet Explorer 5 and above"*
   - [Implementation requirements]
4. *"The system shall support at least 20 transactions per second."*
   - [Performance requirements to specify minimum acceptable performance]
5. *"There should be no more than 3 clicks..."*
   - [Usability]
6. "*The access permissions for system data may…"*
   - [Security]

How do requirements relate to design?

- Separate activities?
  - Focus on **What** vs. **How** initially
    - What: Abstract behavior of system feature
      - "User will be able to register"
    - How: How it is implemented
      - "System will persist user data using JDBC, Oracle and JavaBeans"
- In reality, often there are overlaps. Some design may be needed to articulate requirement.
  - System has to fit within existing environment (e.g. 'data has to be read from ORACLE DB')
  - Large systems decompose into subsystems which have their own constraints
  - Re use of existing software (will use Company's existing e-Commerce Suite to implement X,Y,Z)
  - Re use of approved 'patterns' or prior project solution

From Requirements to Design



Requirement Checklist:

- ☑ **Premature design**
  - Does the requirement include premature detailed design or implementation information? High level design is sometimes necessary.
- ☑ **Combined requirements**
  - Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?
- ☑ **Unnecessary requirements**
  - Is the requirement 'gold plating'? That is, is the requirement a cosmetic addition to the system which is not really necessary.
- ☑ **Use of non-standard hardware**
  - Does the requirement mean that non-standard hardware or software must be used? To make this decision, you need to know the computer platform requirements.
- ☑ **Conformance with business goals**
  - Is the requirement consistent with the business goals defined in the introduction to the requirements document? Requirements ambiguity
- ☑ **Requirements ambiguity**
  - Is the requirement ambiguous i.e. could it be read in different ways by different people? What are the possible interpretations of the requirement? Try to close any gaps that will allow for Scope Creep.
- ☐ ☑ **Requirements realism**
  - Is the requirement realistic given the technology which will be used to implement the system? Is the requirements realistic given other constraints (budget, schedule, etc)?
- ☐ ☑ **Requirements testability**
  - Is the requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

Requirements Process and CMM:

- **Initial level - 1**
  - No defined RE process. Problems include:
    - Volatility (no way to manage change)
    - Unsatisfied stakeholders
    - High rework costs.
    - High dependency on individual skills and experience.
- **Repeatable level - 2**
  - Defined standards for requirements documents and policies and procedures for requirements management.
- **Defined level - 3**
  - Defined RE process based on good practices and techniques. Active process improvement process in place.

References:

- **Managing Software Requirements: A Unified Approach**
  - by Dean Leffingwell and Don Widrig
- **An Introduction to Requirements Engineering**
  - by Gerald Kotonya and Ian Sommerville
- **Requirements Engineering**
  - by Csaba Veres
- **Requirements**
  - By Spencer Rugaber
- **Software Requirements Specification**
  - By Karl E. Wiegers