Dean Leffingwell & Don Widrig

Managing Software requirements

MANAGING SOFTWARE REQUIREMENTS
A UNIFIED APPROACH
DEAN LEFFINGWELL
DON WIDRIG
Ed Yourdon
"The most practical book ever written on requirements. The authors have obviously 'been there.' A must read for every practitioner."
—Al Davis, Omni-Vista, Inc.

The Rock problem

- You might have clients who demand:

“Bring me a rock”

- But when you deliver the rock, the customer looks at it for a moment and says,

"Yes, but, actually, what I really wanted was a small blue rock."

The Rock problem continues…

- And then when you deliver the small blue rock…

“Oh! but I wanted a small blue marble and not a rock”

We need to prevent failures

- More than half of the software systems projects today are substantially over budget and behind schedule,
- 25%–33% of the projects are canceled before completion,
- Often at a staggering cost.

REMEMBER

We got to get it right the first time itself

Coz there’s no time for another iteration

A software team consists of…

- Development team
  - *analysts, developers, tester and QA personnel,*
  - *Project management, documentation folks*
- “Customer" team
  - *users and other stakeholders*
  - *marketing*
  - *management*

It is extremely crucial that…

- members of both teams,
  - including the nontechnical members of the external team,
- master the skills required to
- successfully define & manage requirements process for your new system—
- simple reason that
  - *they are the ones who create the requirements in the* first place and who ultimately determine the success or failure of the system

If you were a building contractor

- You need a detailed discussion with the owner
- The Government authorities to get sanctions
- Even the neighbours (sometimes) to ensure that it meets the requirements
- And is not a nuisance for anybody

Our blueprints

- Engineering drawings help in depicting requirements of a building

- Even technical drawings related to software systems can be created in a way that a layman can understand!

So let’s arm ourselves with some skills

To build the perfect marble for our customers!

Sustainable success requires a combination of…

- A pragmatic process for defining and managing the requirements for the software
- A solid methodology for the design and development of software
- The application of various proven, innovative, techniques for verifying and validating that the software was safe and effective
- Extraordinary skills and commitment on the part of both the software development and software quality assurance teams

Chapter 1

The Requirements Problem

Key Points

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

Develop quality software (on time and on budget) that meets customers' real needs

- The Standish Group study 1994:
  - $250 billion is spent each year on IT application development of approximately 175,000 projects.
  - Average cost of a development project:
    - large company - $2,322,000; medium company - $1,331,000, small company - $434,000….
  - 31% of projects get canceled before completion
  - 52.7% of projects cost 189% of their original estimates….

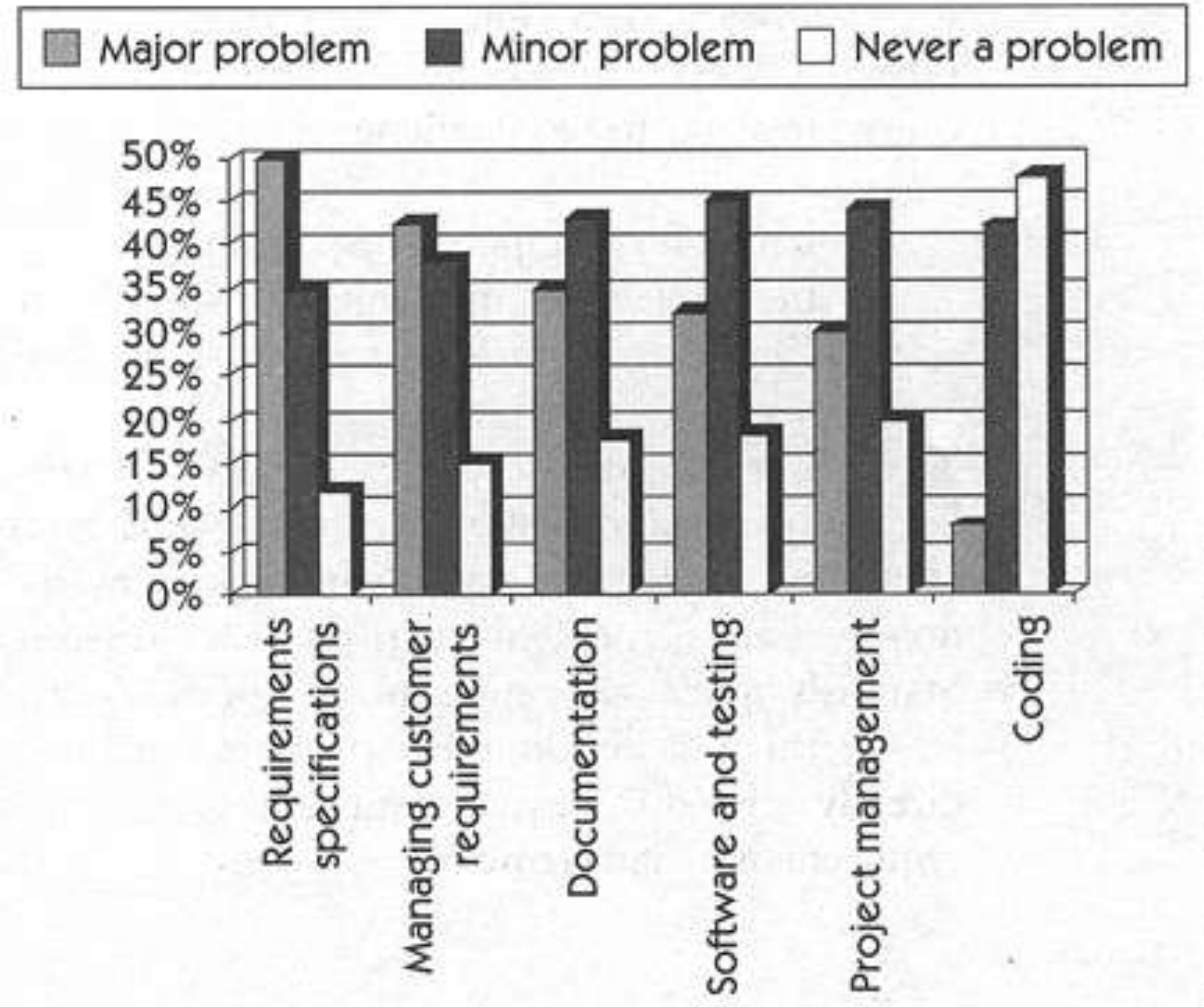
3 factors that cause software to be challenged: *The Standish Group*

- Lack of user input: 13 percent of all projects
- Incomplete requirements and specifications: 12 percent of projects
- Changing requirements and specifications: 12 percent of all projects
    - *unrealistic schedule or time frame (4 percent)*
    - *inadequate staffing and resources (6 percent)*

Primary success factors

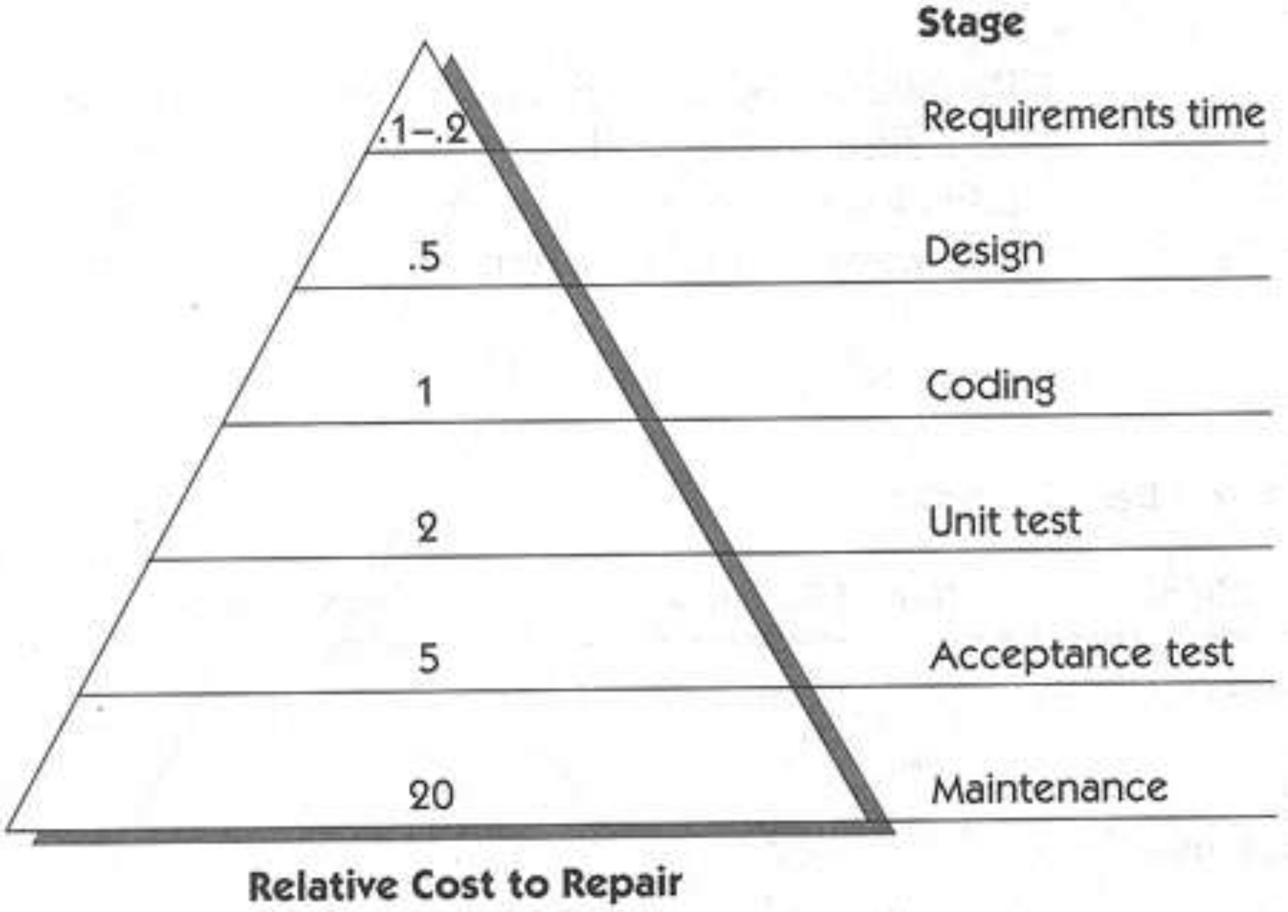
- User involvement: 16 percent of all successful projects
- Executive management support: 14 percent of all successful projects
- Clear statement of requirements: 12 percent of all successful projects

European Software Process Improvement Training Initiative (ESPITI) survey (1995)



Relative cost to repair a defect at different lifecycle phases

- Many errors are not detected until it is made



To repair a defect we experience cost in…

- Re-specification
- Redesign
- Recoding
- Retesting
- Change orders
- Corrective action
- Scrap (code and design that had to be thrown away)
- Recall of defective versions from users
- Warranty costs
- Product liability (if the customer sues for damages)
- Service costs
- Documentation

So with that background, let’s move to Chapter 2

Introduction to Requirements Management

Key points that will be covered

- A requirement: *capability the system must deliver*
- Requirements management: *process of eliciting, organizing, and documenting requirements*
- Our problem: *understand users' problems in their culture and their language and to build systems that meet their needs*
- A feature: *service the system provides to fulfill one or more stakeholder needs*
- A use case: *describes a sequence of actions, performed by a system, that yields a result of value to a user*

What is a requirement?

- Dorfman and Thayer (1990)
  - A software capability needed by the user to solve a problem to achieve an objective
  - A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation

Requirements management

  - a systematic approach to **eliciting, organizing, and documenting** the requirements 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
- When many members are involved in parts of the software system, there’s more management to be done

Applications of RM Techniques

- Software application categories
  - IS and other applications developed for use within a company
    - payroll system
  - Software we develop and sell as commercial products – Independent software vendors
    - MS Office
  - Software that runs on computers embedded in other devices, machines, or complex systems – software embedded system applications
    - Automobiles, mobile phones

The ROAD MAP
We are embarking on a journey

Develop quality software—on time and on budget—that meets customers' real needs

Problem domain – land of the problem

- home of real users and other stakeholders
- people whose needs must be addressed to build the perfect system
  - Have business or technical problems
  - A set of team skills are used to understand the problem to be solved

Stakeholder needs

- Elicit needs of our primary users
- And every other stakeholder associated with the software system
  - *Remember the Building contractor example!*

Moving towards solution domain

- Define a solution to user’s problems
  - computers, programming, operating systems, networks, and processing nodes

Features of the system

- State what we learned in the problem domain and how we intend to deliver that via the solution
  - User’s language
  - *Features: a service that the system provides to flfill one or more stakeholder needs.*

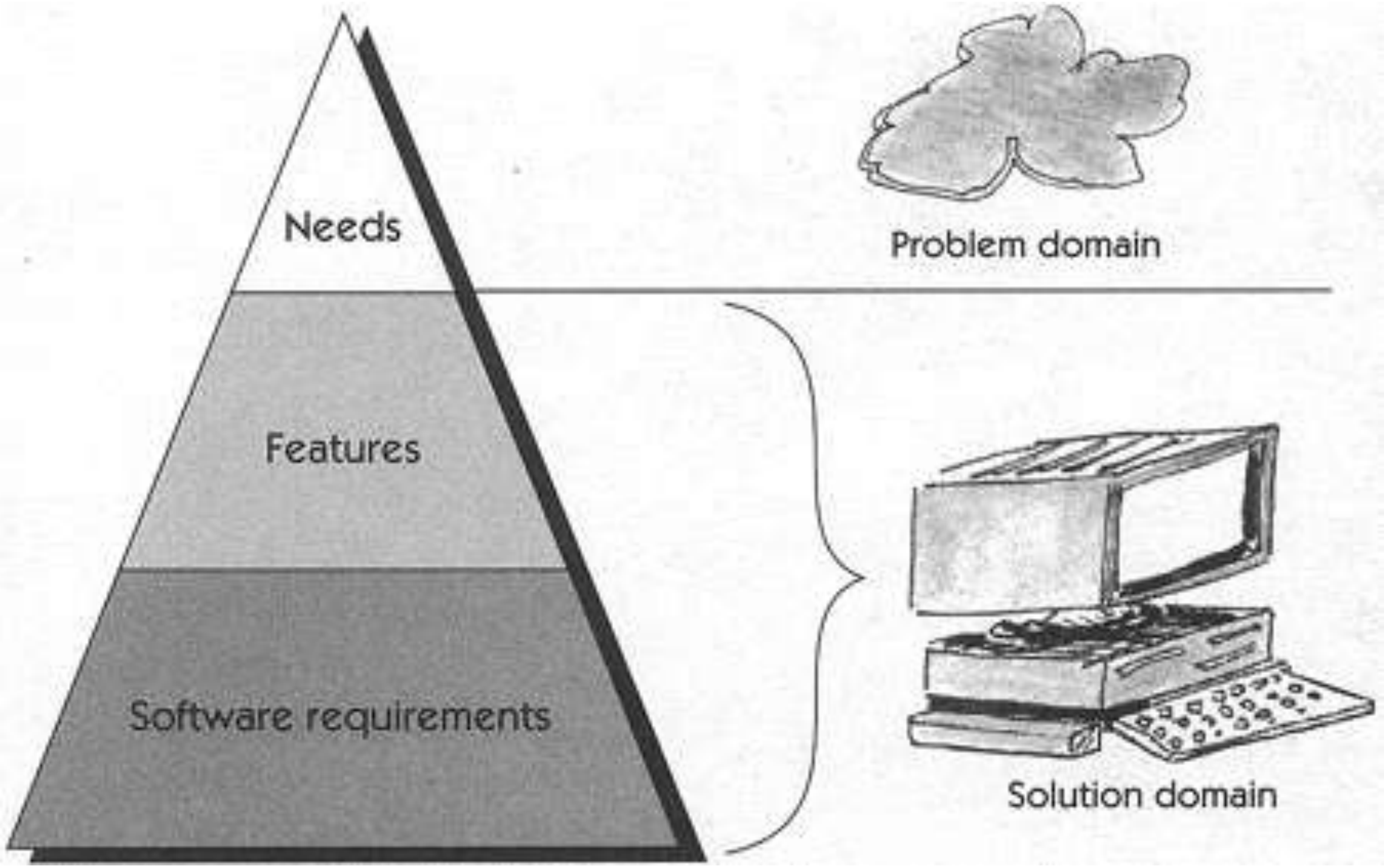
Software requirements

- specific requirements that we will need to impose on the solution

Introduction to use cases

- Use case describes a sequence of actions, performed by a system, that yields a result of value to a user
- describes a series of user/system interactions that helps the user accomplish something

The problem / solution domain



"Computer programming is a human activity."

—(Weinberg 1971)

Chapter 3

The Software Team

Key points covered

- Requirements management touches every team member, albeit in different ways.
- Effective requirements management can only be accomplished by an effective software team
- Six team skills are needed for requirements management

We have to make the Team thing work!

Requisite Team Skills for Effective Requirements Management

1. Analyzing the Problem
2. Understanding User Needs
3. Defining the System
4. Managing scope
5. Refining system definition
6. Building the right system

Team Skill 1: ANALYZING THE PROBLEM

Chapter 4
The 5 steps in Problem Analysis

Key points

- Problem analysis: process of understanding real-world problems and user's needs and proposing solutions to meet those needs
- The goal is to gain a better understanding, before development begins, of the problem.
- To identify the root cause, ask the people directly involved.
- Identifying the actors on the system is a key step in problem analysis

Problem analysis

- Problem: difference b/w things as perceived and things as derived
- A way to address the problem: change user's desire or perception
- Else bridge the gap b/w perception and reality
- Goal: gain better understanding of the problem being solved

Steps in problem analysis

1. Gain agreement on the problem definition.
2. Understand the root causes—the problem behind the problem.
3. Identify the stakeholders and the users.
4. Define the solution system boundary.
5. Identify the constraints to be imposed on the solution.

1. Gain agreement on the problem definition

- *simply write the problem down and see whether everyone agrees*
- Problem statement format
  - The problem of *(describe the problem)* affects *(identify stakeholders affected by the problem),* the result of which *(describe the impact of this problem on stakeholders and business activity).* Benefits of *(indicate the proposed solution and list a few key benefits).*

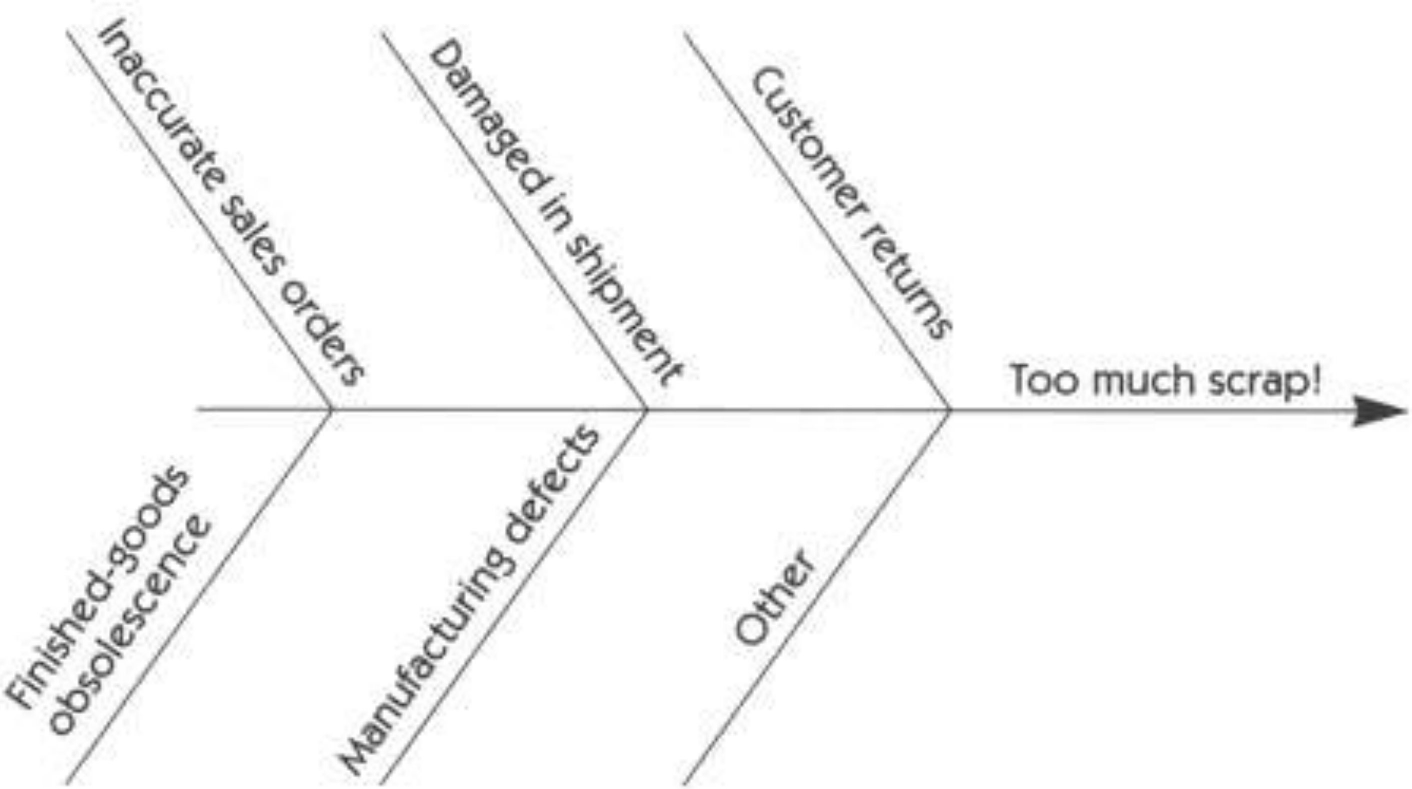
2: Understand the Root Causes—The Problem Behind the Problem

- Root cause analysis – Fishbone diagram
- Ask the people directly involved – what’s the root cause
- Or a detailed investigation of each contributing problem and to quantify its individual impact
- Contribution of each root cause: Pareto chart or simple histogram

GoodsAreUs – mail order catalogue company

- Manufactures and sells a variety of inexpensive, miscellaneous items for home and personal use.
- Problem: insufficient profitability
- TQM techniques applied
- Finds root cause as *cost of nonconformance* - cost of all of the things that go wrong and produce waste, scrap, and other excess costs (negative-value activities)
- *What factors lead to too much scrap?*

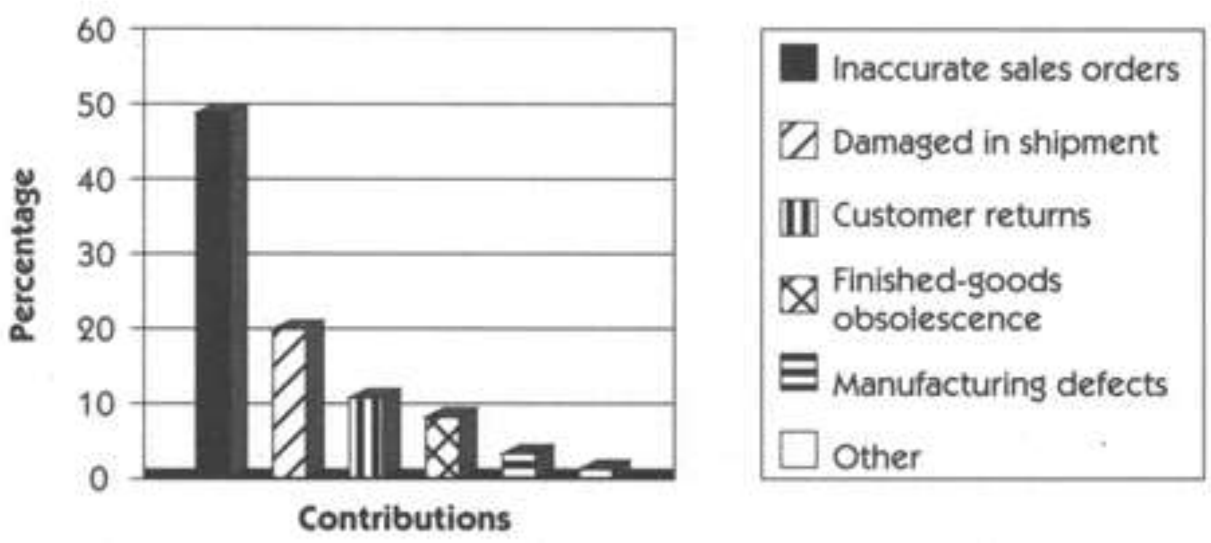
Fishbone Diagram



- But is that enough? *Not really*

Determine contribution of each root cause

- Pareto chart or simple Histogram



- the existing sales order system was defective – *New Sales Order system!*

Sales order problem statement

Elements	Description
The problem of	inaccuracies in sales orders
affects	sales order personnel, customers, manufacturing, shipping, and customer service,
the result of which is	is increased scrap, excessive handling costs, customer dissatisfaction, and decreased profitability
Benefits of	a new system to address the problem include • Increased accuracy of sales orders at point of entry • Improved reporting of sales data to management • And, ultimately, higher profitability

- *Circulate the problem statement to stakeholders* for comment and feedback
- Further Fishbone diagram can reveal what errors cause inaccurate sales order
- Thus help *define features of the new system*

Step 3: Identify the Stakeholders and the Users

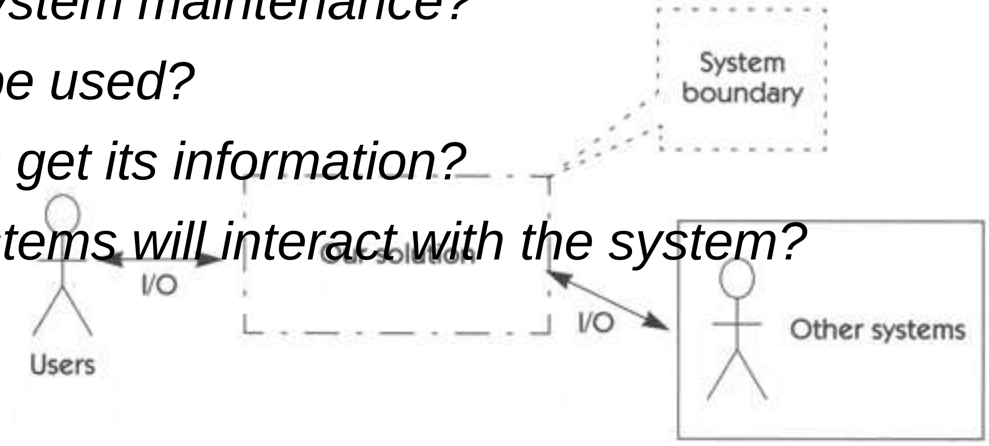
- *anyone who could be materially affected by the implementation of a new system or application*
  - Who are the users of the system?
  - Who is the customer (economic buyer) for the system?
  - Who else will be affected by the outputs that the system produces?
  - Who will evaluate and bless the system when it is delivered and deployed?
  - Are there any other internal or external users of the system whose needs must be addressed?
  - Who will maintain the new system?
  - Is there anyone else?

In our Example: Sales Order System

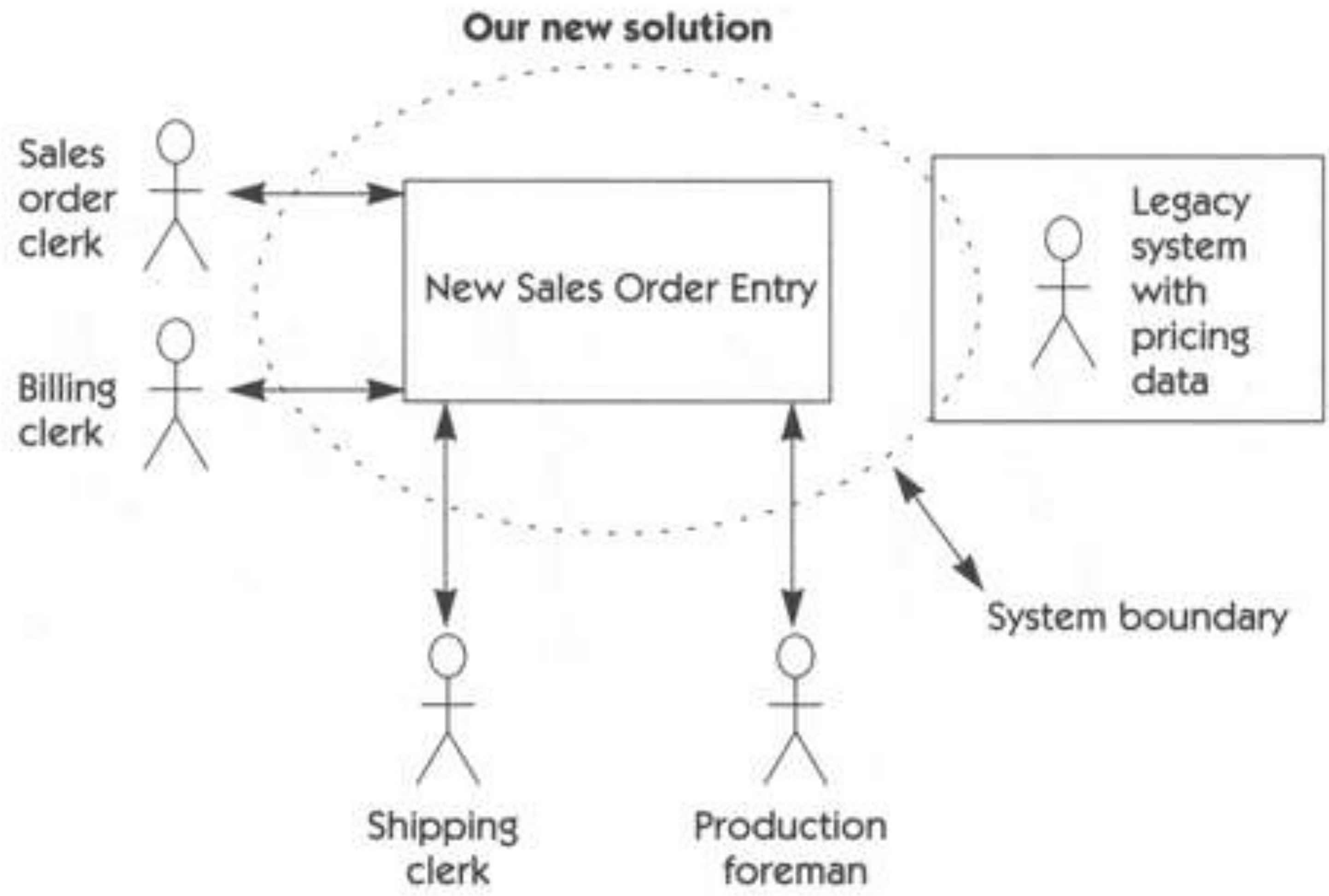
Users	Stakeholders
Sales order entry clerks	MIS Director & development team
Sales order supervisor	CFO
Production control	Production manager
Billing clerk	

Step 4: Define the Solution System Boundary

- border between the solution and the real world that surrounds the solution
- How do we find the actors - someone or something, outside the system, that interacts with the system
  - *Who will supply, use, or remove information from the system?*
  - *Who will operate the system?*
  - *Who will perform any system maintenance?*
  - *Where will the system be used?*
  - *Where does the system get its information?*
  - *What other external systems will interact with the system?*



Create a System perspective



Step 5: Identify the Constraints to Be Imposed on the Solution

- Constraint: *a restriction on the degree of freedom we have in providing a solution*
  - Some of these constraints become requirements of the new system
  - Other constraints will affect resources, implementation plans, and project plans
- Understand sources of constraints
- determine the impact of each constraint on the potential solution spaces

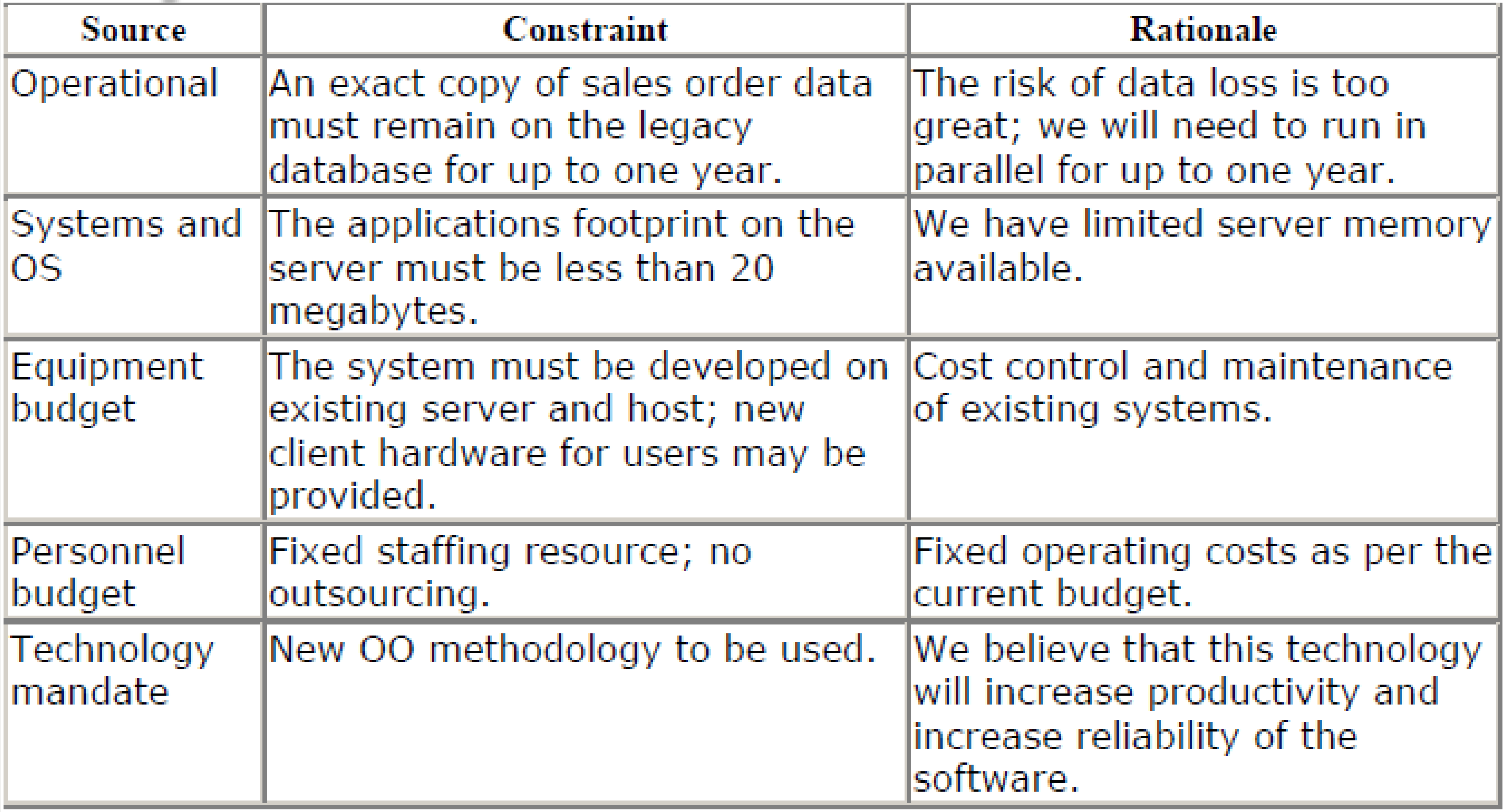
Potential system constraints

Source	Sample considerations
Economic	•What financial or budgetary constraints are applicable? •Are there costs of goods sold or any product pricing considerations? •Are there any licensing issues?
Political	•Are there internal or external political issues that affect potential solutions? •Interdepartmental problems or issues?
Technical	•Are we restricted in our choice of technologies? •Are we constrained to work within existing platforms or technologies? •Are we prohibited from any new technologies? •Are we to use any purchased software packages?

Potential system constraints contd...

Source	Sample considerations
System	•Is the solution to be built on our existing systems? •Must we maintain compatibility with existing solutions? •What operating systems and environments must be supported?
Environmental	•Are there environmental or regulatory constraints? •Legal? •Security requirements? •What other standards might we be restricted by?
Schedules and resources	•Is the schedule defined? •Are we restricted to existing resources? •Can we use outside labor? •Can we expand resources? Temporary? Permanent?

Constraints of Sales Order System



Source	Constraint	Rationale
Operational	An exact copy of sales order data must remain on the legacy database for up to one year.	The risk of data loss is too great; we will need to run in parallel for up to one year.
Systems and OS	The applications footprint on the server must be less than 20 megabytes.	We have limited server memory available.
Equipment budget	The system must be developed on existing server and host; new client hardware for users may be provided.	Cost control and maintenance of existing systems.
Personnel budget	Fixed staffing resource; no outsourcing.	Fixed operating costs as per the current budget.
Technology mandate	New OO methodology to be used.	We believe that this technology will increase productivity and increase reliability of the software.

For independent software vendors, problem analysis involves…

- Identifying market opportunities and market segments
- Identifying classes of potential users and their particular needs
- Studying the demographics of the potential user base
- Understanding potential demand, pricing, and pricing elasticity
- Understanding sales strategies and distribution channels

Team Skill 1: ANALYZING THE PROBLEM

Chapter 5
Business Modeling

Key points

- Business modeling is a problem analysis technique especially suitable for the IS/IT environment.
- Helps define systems and their applications.
- A business use case model, consisting of actors and use cases, is a model of the intended functions of the business.
- A business object model describes the entities that deliver the functionality to realize the business use cases, and how these entities interact.

IS / IT environment is far more complex than we think

- Businesses are complex
- A system is not just an interface b/w a computer and 1 or 2 users but b/w:
  - organizations, business units, departments, functions, WAN, the corporate intranet and extranet, customers, users, human resources, MRP systems, inventory, management systems, and more

We need a technique to determine answers to the following questions

- Why build a system at all?
- Where should it be located?
- How can we determine what functionality is optimum to locate on a particular system?
- When should we use manual-processing steps or workarounds?
- When should we consider restructuring the organization itself in order to solve the problem?

That technique is Business Modeling

- Purpose is 2-fold
  - To understand the structure and dynamics of the organization
  - To ensure that customers, end users, and developers have a common understanding of the organization
- This can help to define
  - where software applications can improve the productivity of the business
  - assist in determining requirements for those applications

Business modeling techniques

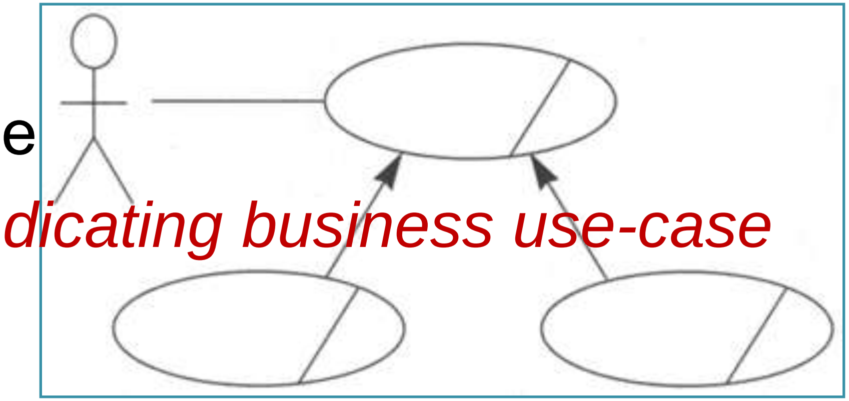
- Object Oriented techniques
- Unified Modeling Language (UML)

Business Modeling using UML concepts

- goals of business models is to develop a model of the business that can be used to drive application development
- Two key modeling constructs
  - business use-case model
  - business object model
    - the two models provide a comprehensive overview of how business works
    - development team can focus on areas where systems can be provided to improve overall efficiency
    - what changes are needed to implement the new system

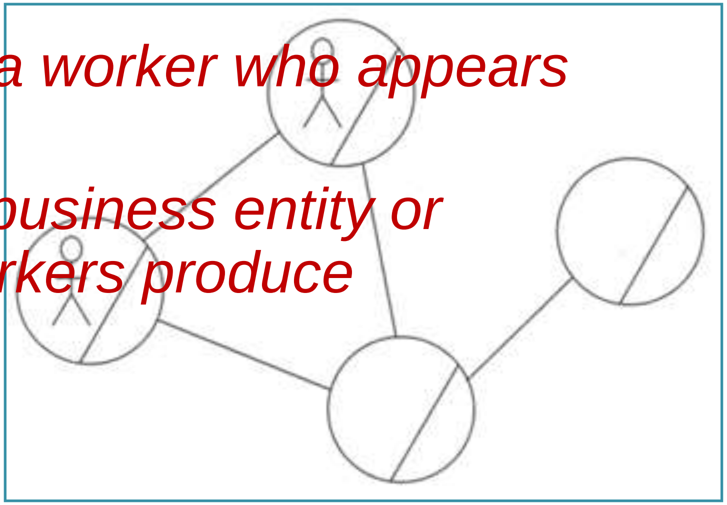
Business Use Case Model

- model of intended functions of business used as an input to identify roles and deliverables in the organization
- consists of
  - actors—users and systems that interact with the business
  - use cases—sequences of events by which the actors interact with the business elements to get their job done
  - Here’s an example
  - *Oval with slash indicating business use-case*



Business Object Model

- Describes entities—departments, paychecks, systems—and how they interact to deliver the functionality necessary to realize the business use cases
- also includes business use-case realizations
- show how business use cases are "performed" in terms of interacting business workers and business entities
    - *actor-circle icon represents a worker who appears within the business process*
    - *slashed circle represents a business entity or something that business workers produce*



Team Skill 1: ANALYZING THE PROBLEM

Chapter 6. Systems Engineering of Software-Intensive Systems

Key points

- Systems engineering is a problem analysis technique suitable for embedded systems development.
- helps us understand requirements imposed on software applications that run within the solution system.
- Requirements flowdown helps us ensure that all system requirements are filled by a subsystem or a set of subsystems collaborating
- Today, the system must often be optimized for software costs rather than for hardware costs

Embedded-systems business

- Instead of departments, people, and processes,
- the domains consist of
  - connectors and power supplies, racks of equipment, electronic and electrical components, hydraulic and fluidic handling devices, other software systems, mechanical and optics subsystems, and the like…
- Here Systems Engineering is more appropriate

What is Systems Engineering?

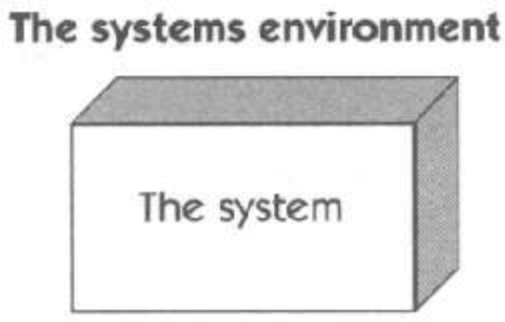
- helps us understand the requirements that are going to be imposed on any software applications that run within the solution system
    - INCOSE Systems Engineering Practices working group (INCOSE 1993) defined a basic set of *eight systems engineering principles*

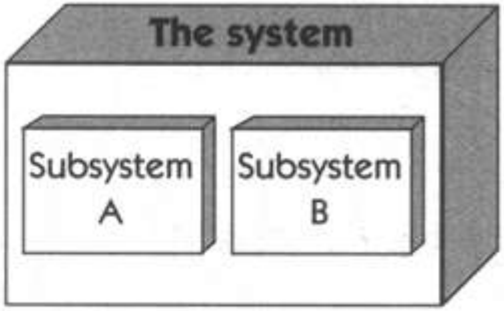
8 Systems Engineering principles

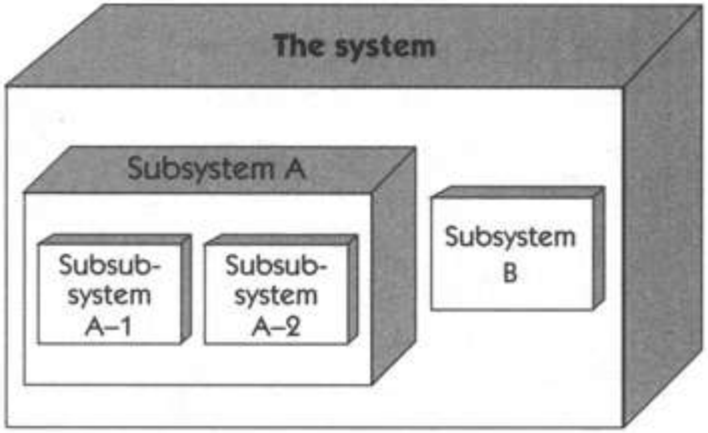
1. Know the problem, know the customer, and know the consumer.
2. Use effectiveness criteria based on needs to make the system decisions.
3. Establish and manage requirements.
4. Identify and assess alternatives so as to converge on a solution.
5. Verify and validate requirements and solution performance.
6. Maintain the integrity of the system.
7. Use an articulated and documented process.
8. Manage against a plan.

Complex system decomposed to smaller problems

- The job is done right when:
  - Distribution and partitioning of functionality are optimized to achieve the overall functionality of the system with minimal costs and maximum flexibility.
  - Each subsystem can be defined, designed, and built by a small, or at least modest-sized, team





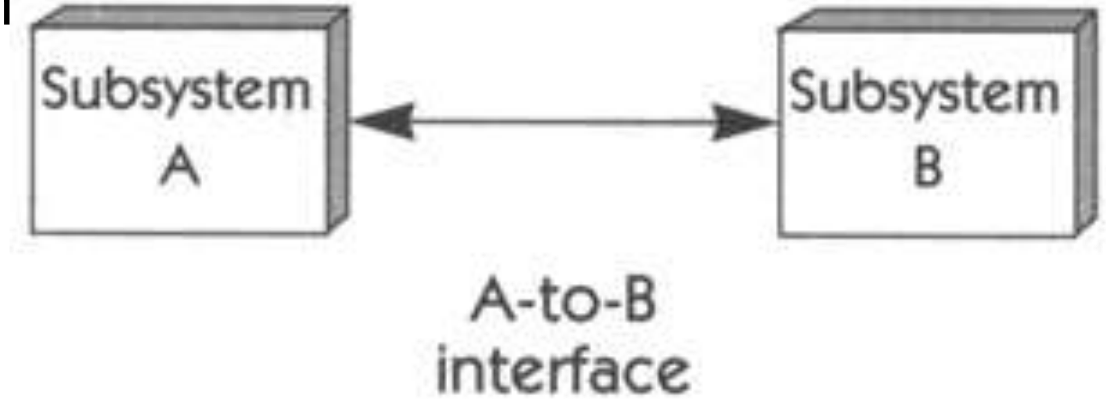


Complex system decomposed to smaller problems

- The job is done right when:
  - Each subsystem can be manufactured within the physical constraints and available technologies
  - Each subsystem can be reliably tested as a subsystem
  - Appropriate deference is given to the physical domain—the size, weight, location, and distribution of the subsystems—that has been optimized in the overall system context.

Derived requirements of subsystems

- **Subsystem requirements**: Imposed on subsystems but do not necessarily provide a direct benefit to the end user ("Subsystem A must execute the algorithm that computes the wind speed of the aircraft")
- **Interface requirements**: Subsystems need to communicate with one another to accomplish an overall result – share data/power / a useful computing algorithm



Moving to our HOLIS Case Study

Preliminary user needs

- HOLIS will need to support "soft" key switches—programmable key switches used to activate the lighting features in various rooms.
- Program HOLIS from a remote center
- HOLIS be programmable from their home PCs and that they be provided with the ability to do all of the installation, programming, and maintenance themselves.
- System provide a simple, push-button control panel–type interface they can use to change HOLIS programming, vacation settings, and so on, without having to use a PC.
- HOLIS needs to provide an emergency-contact system of some kind.

Problem statement for Lumenations

Elements	Description
The problem of	slowing growth in the company's core professional theater marketplaces
affects	the company, its employees, and its shareholders,
the result of which is	is unacceptable business performance and lack of substantive opportunities for growth in revenue and profitability.
Benefits of	new products and a potential new marketplace for the company's products and services include • Revitalizing the company and its employees • Increased loyalty and retention of the company's distributors • Higher revenue growth and profitability • Upturn in the company's stock price

Problem statement for Homeowners

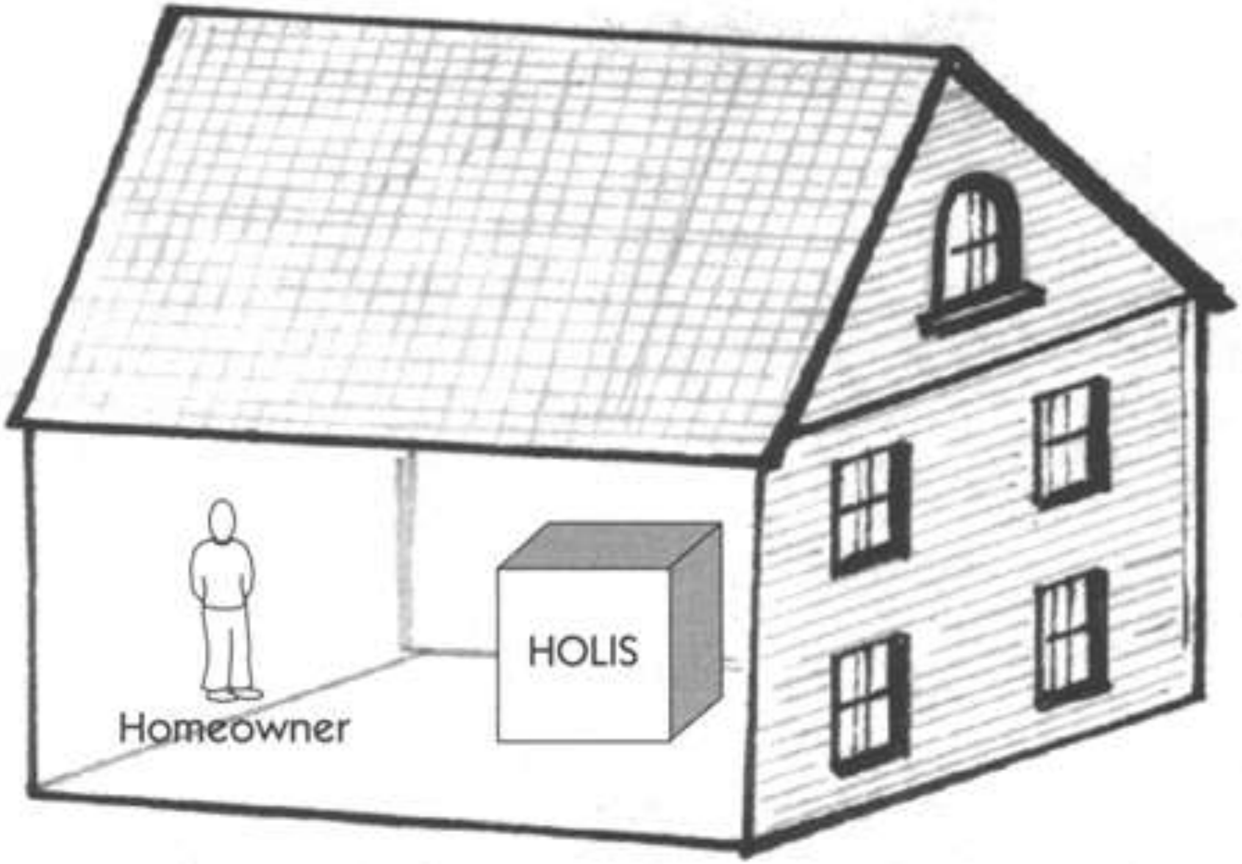
Elements	Description
The problem of	the lack of product choices, limited functionality, and high cost of existing home lighting automation systems
affects	the homeowners of high-end residential systems,
the result of which is	is unacceptable performance of the purchased systems or, more often than not, a decision "not to automate."
Benefits of	the "right" lighting automation solution could include • Higher homeowner satisfaction and pride of ownership • Increased flexibility and usability of the residence • Improved safety, comfort, and convenience

Problem statement for Distributors

Elements	Description
The problem of	the lack of product choices, limited functionality, and high cost of existing home lighting automation systems
affects	the distributors and builders of high-end residential systems,
the result of which is	is few opportunities for marketplace differentiation and no new opportunities for higher-margin products.
Benefits of	the "right" lighting automation solution could include • Differentiation • Higher revenues and higher profitability • Increased market share

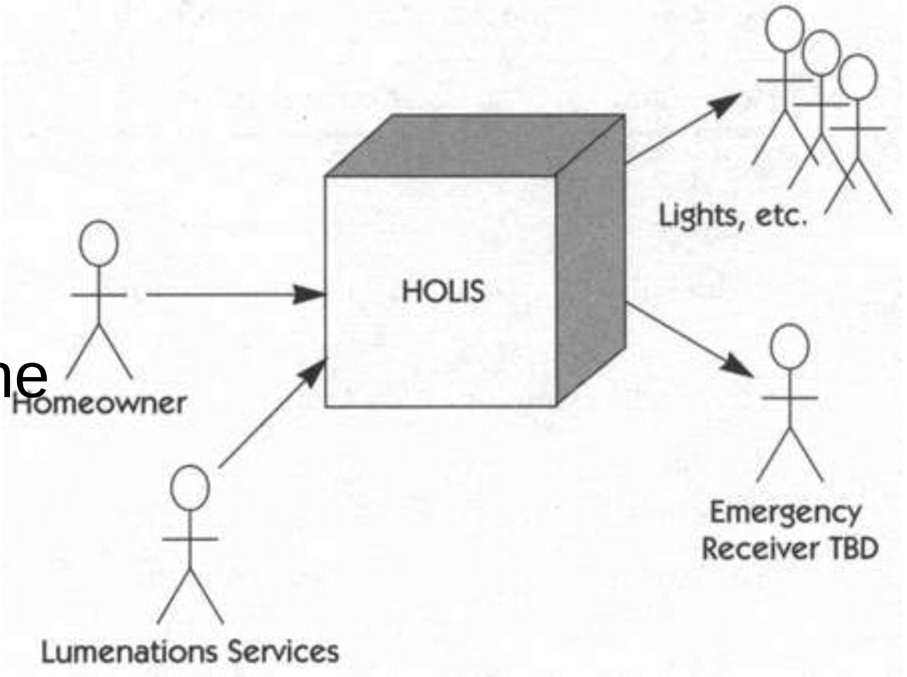
HOLIS: The System, Actors, and Stakeholders

- *System context:HOLIS in its environment*



HOLIS with actors

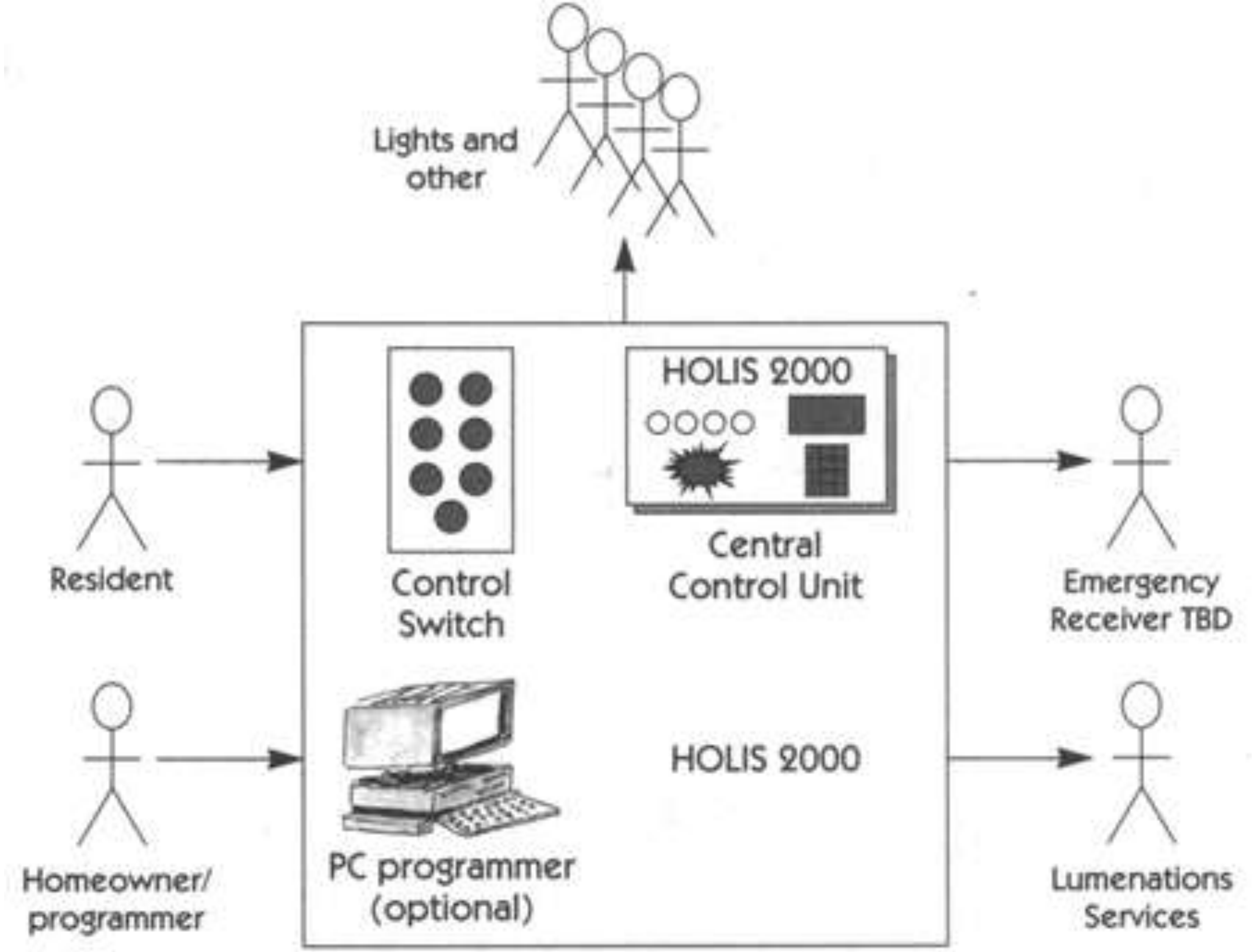
- The homeowner who uses HOLIS to control the lighting
- The various lights that HOLIS, in turn, controls
- Lumenations Services, the manufacturer who can remotely dial HOLIS and perform remote programming
- Emergency Receiver, an undefined actor who will likely receive emergency messages



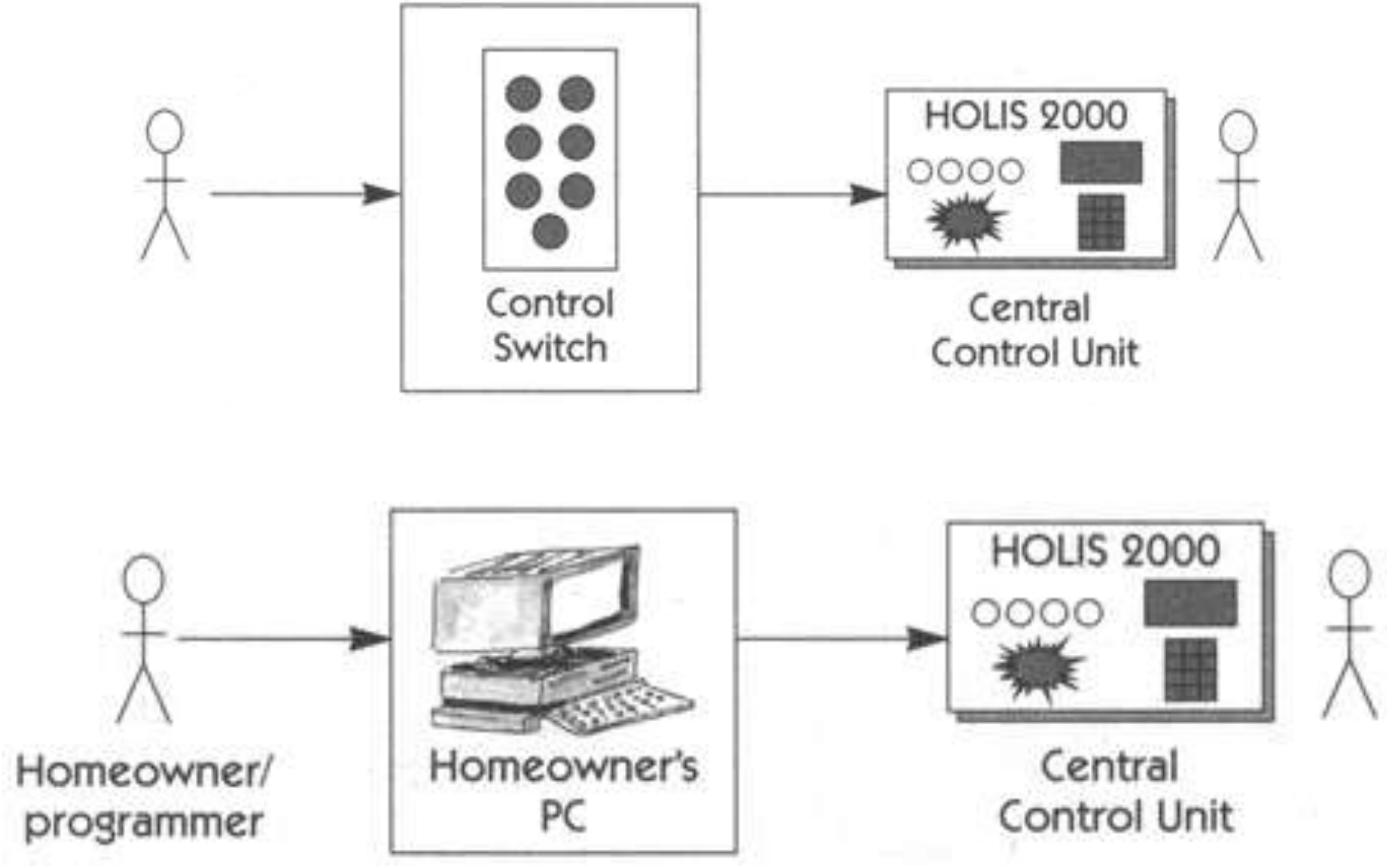
Non-actor stakeholders of HOLIS

Item name	Comments
EXTERNAL	
Distributors	Lumenations' direct customer
Builders	Lumenations' customer's customer: the general contractor responsible to the homeowner for the end result
Electrical Contractors	Responsible for installation and support
INTERNAL	
Development team	Lumenation's team
Marketing/product management	Will be presented by Cathy, product manager
Lumenations general management	Funding and outcome accountability

HOLIS sub-systems

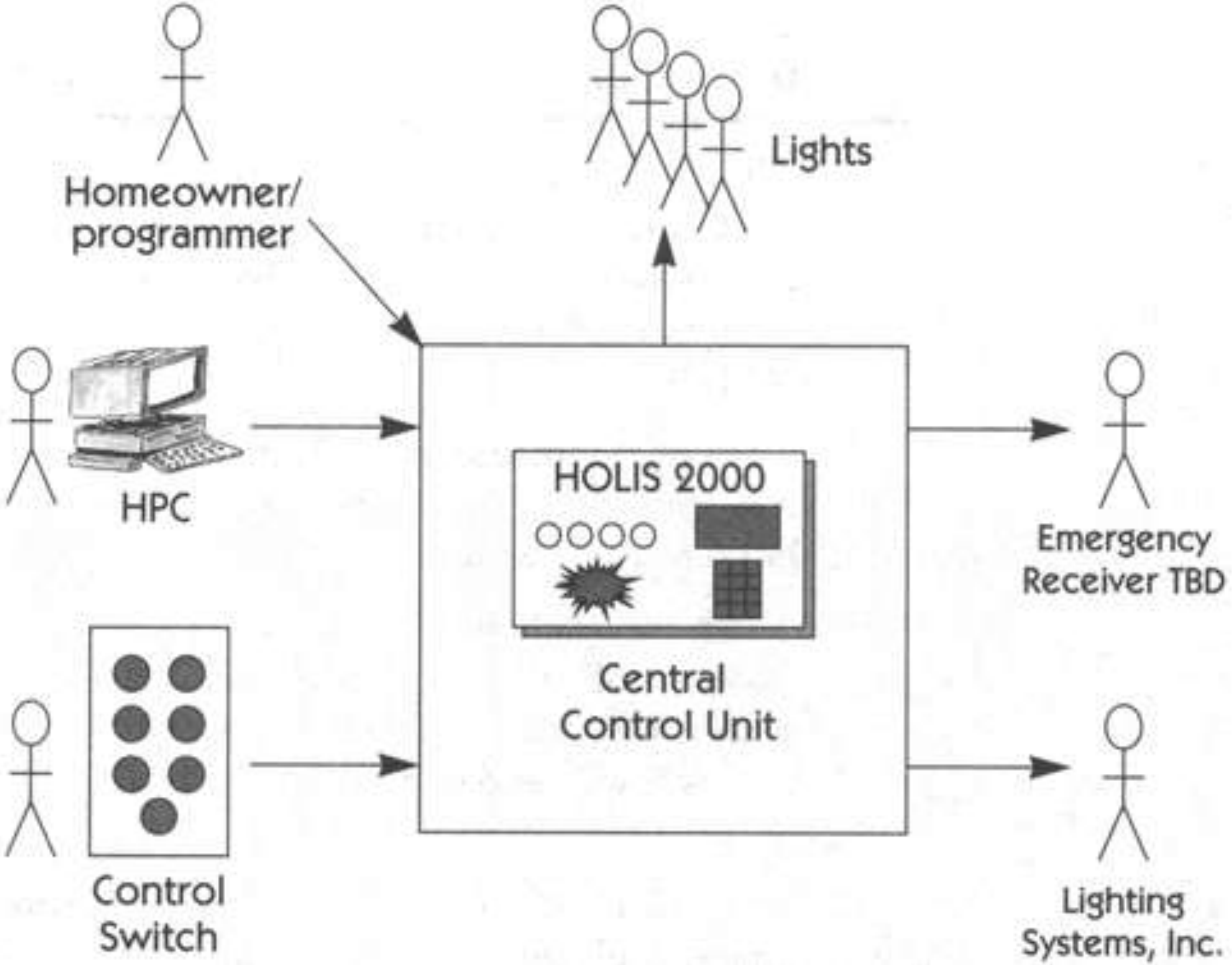


Unique requirements for each of HOLIS's three subsystems

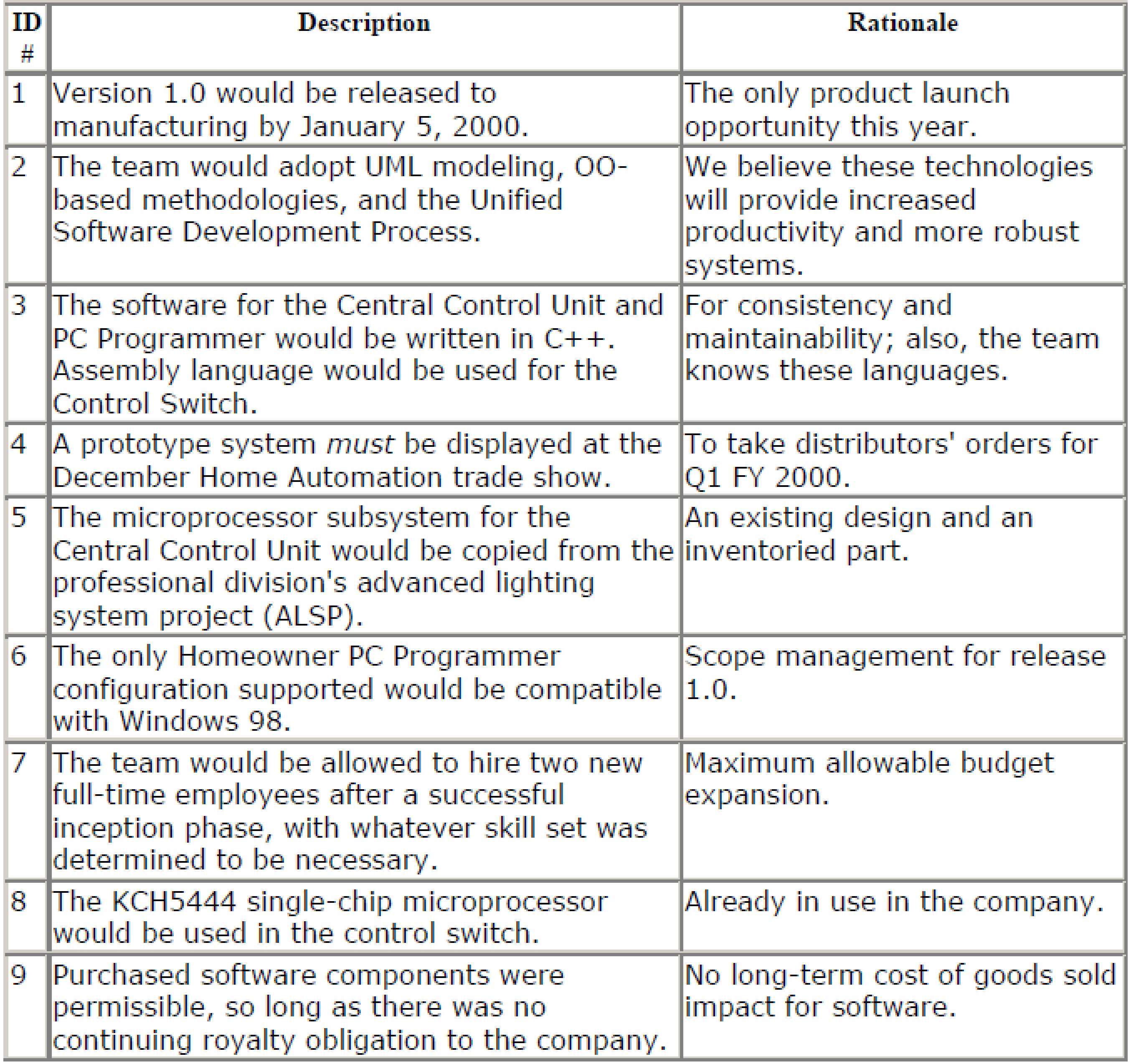


Here, CCU is an actor, which is actually a sub-system!

Central Control Unit subsystem with actors



Constraints for HOLIS project



ID #	Description	Rationale
1	Version 1.0 would be released to manufacturing by January 5, 2000.	The only product launch opportunity this year.
2	The team would adopt UML modeling, OO-based methodologies, and the Unified Software Development Process.	We believe these technologies will provide increased productivity and more robust systems.
3	The software for the Central Control Unit and PC Programmer would be written in C++. Assembly language would be used for the Control Switch.	For consistency and maintainability; also, the team knows these languages.
4	A prototype system *must* be displayed at the December Home Automation trade show.	To take distributors' orders for Q1 FY 2000.
5	The microprocessor subsystem for the Central Control Unit would be copied from the professional division's advanced lighting system project (ALSP).	An existing design and an inventoried part.
6	The only Homeowner PC Programmer configuration supported would be compatible with Windows 98.	Scope management for release 1.0.
7	The team would be allowed to hire two new full-time employees after a successful inception phase, with whatever skill set was determined to be necessary.	Maximum allowable budget expansion.
8	The KCH5444 single-chip microprocessor would be used in the control switch.	Already in use in the company.
9	Purchased software components were permissible, so long as there was no continuing royalty obligation to the company.	No long-term cost of goods sold impact for software.

Team Skill 1: Summary

- Gain agreement on the problem definition.
- Understand the root causes of the problem.
- Identify the stakeholders and users whose collective judgment will ultimately determine the success or failure of your system.
- Determine where the boundaries of the solution are likely to be found.
- Understand the constraints that will be imposed on your team and on the solution.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 7. The Challenge of Requirements Elicitation

Key Points

- Requirement elicitation is complicated by three endemic syndromes.
- The "Yes, But" syndrome stems from human nature and the users' inability to experience the software as they might a physical device.
- Searching for requirements is like searching for "Undiscovered Ruins"; the more you find, the more you know remain.
- The "User and the Developer" syndrome reflects the profound differences between the two, making communication difficult.

Barriers to elicitation

- The "Yes, But" Syndrome
  - we always observe two immediate, distinct, and separate reactions when the users see the system implementation for the first time
- The solution
  - Elicit the “Yes, But responses much early”

Barriers to elicitation

- The "Undiscovered Ruins" Syndrome
  - *The more that are found, the more you know remain*
- To overcome this
  - *We need to take a decision: we have discovered enough!*

Barriers to elicitation

- The "User and the Developer" Syndrome
  - communication gap between the user and

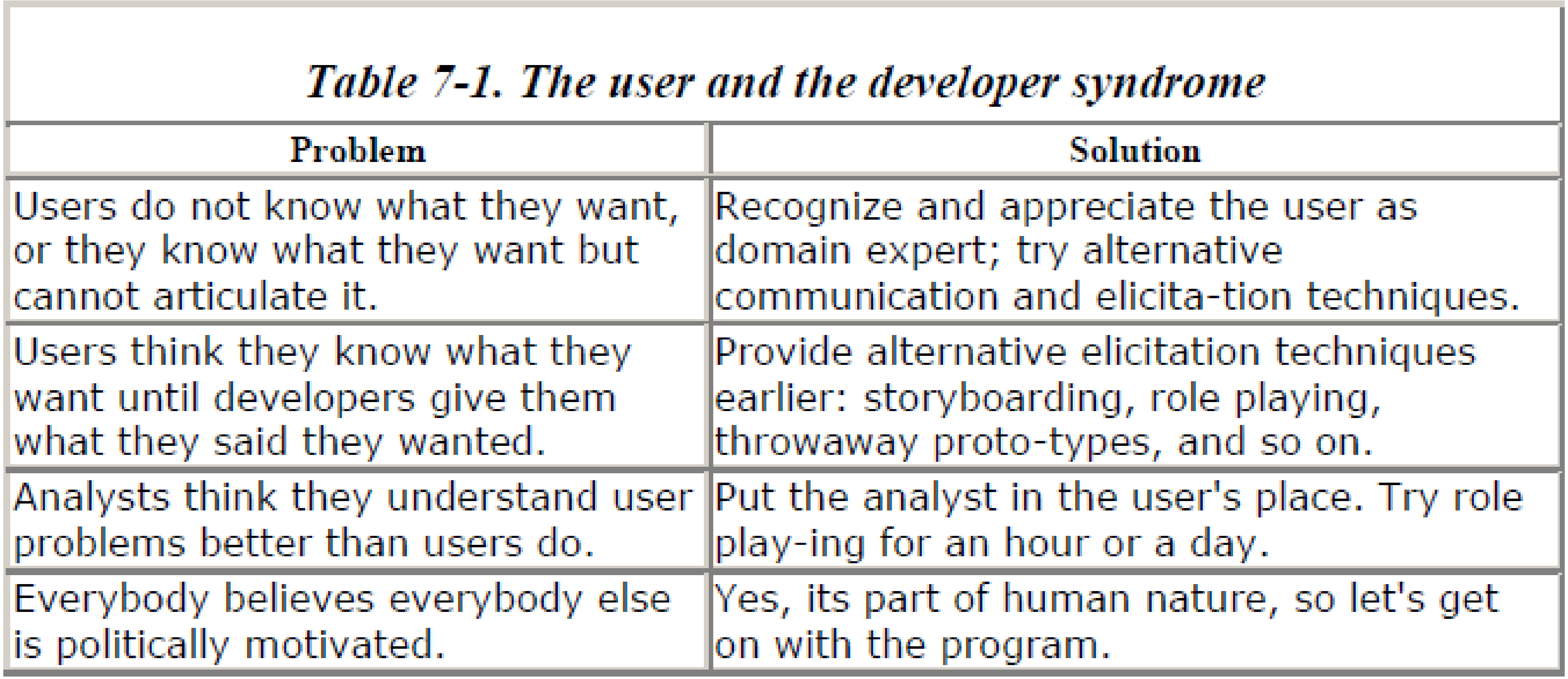


Table 7-1. The user and the developer syndrome

Problem	Solution
Users do not know what they want, or they know what they want but cannot articulate it.	Recognize and appreciate the user as domain expert; try alternative communication and elicita-tion techniques.
Users think they know what they want until developers give them what they said they wanted.	Provide alternative elicitation techniques earlier: storyboarding, role playing, throwaway proto-types, and so on.
Analysts think they understand user problems better than users do.	Put the analyst in the user's place. Try role play-ing for an hour or a day.
Everybody believes everybody else is politically motivated.	Yes, its part of human nature, so let's get on with the program.

Techniques for Requirements Elicitation

- Interviewing and questionnaires
- Requirements workshops
- Brainstorming and idea reduction
- Storyboards
- Use cases
- Role playing
- Prototyping

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 8. The Features of a Product or System

Key Points

- The development team needs to play a more active role in eliciting the requirements for the system.
- Product or system features are high-level expressions of desired system behavior.
- System features should be limited to 25–99, with fewer than 50 preferred.
- Attributes provide additional information about a feature

Development team needs to be more involved

- user needs will be vague and ambiguous
  - “I need easier ways to understand the status of my inventory”
- Do not mention real need or requirement of the system
- high-level expressions of desired system behavior the *features* of the system
- Needs and features are closely related
- Team needs to understand the real need behind a feature

Here are a few examples

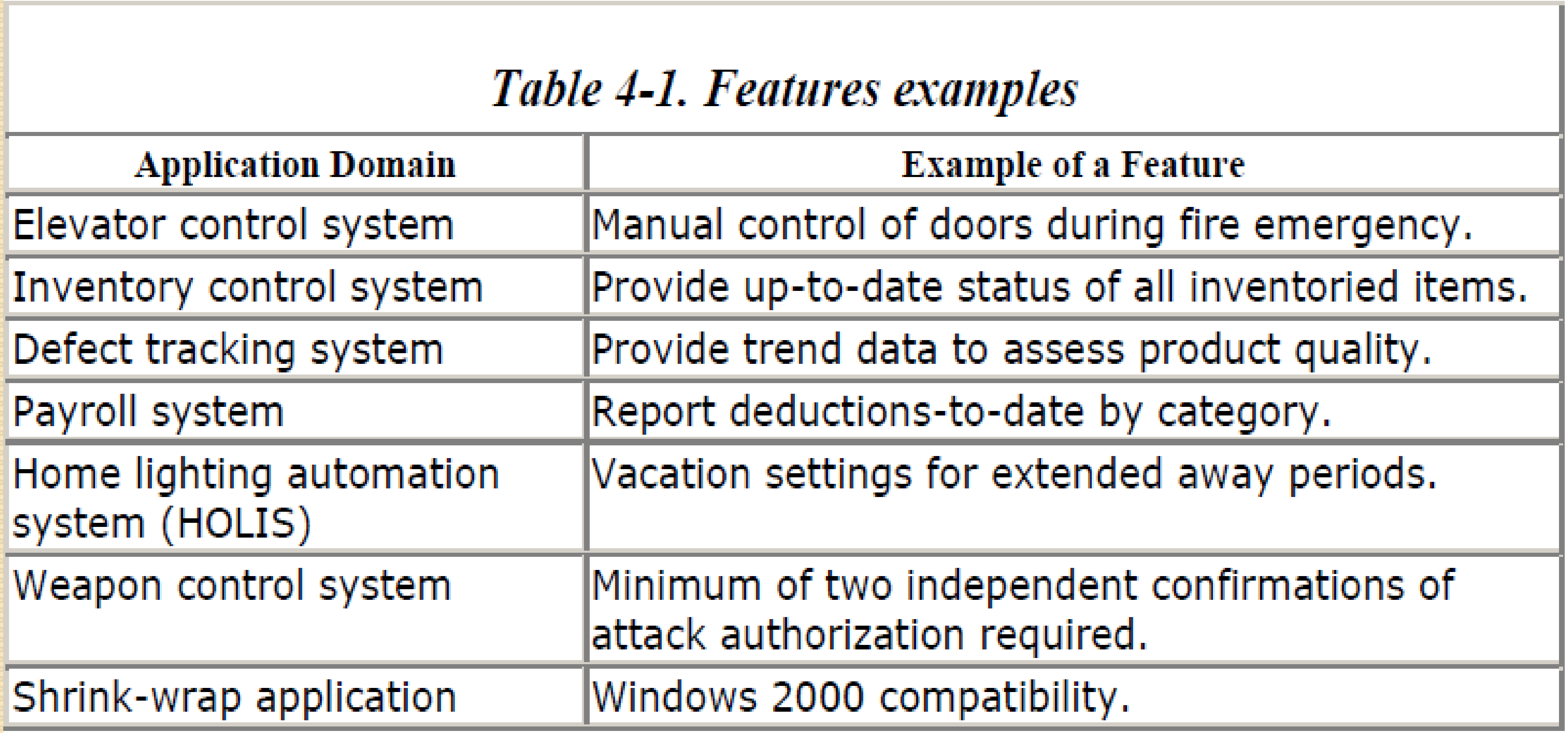


Table 4-1. Features examples

Application Domain	Example of a Feature
Elevator control system	Manual control of doors during fire emergency.
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.
Home lighting automation system (HOLIS)	Vacation settings for extended away periods.
Weapon control system	Minimum of two independent confirmations of attack authorization required.
Shrink-wrap application	Windows 2000 compatibility.

After enumerating possible features

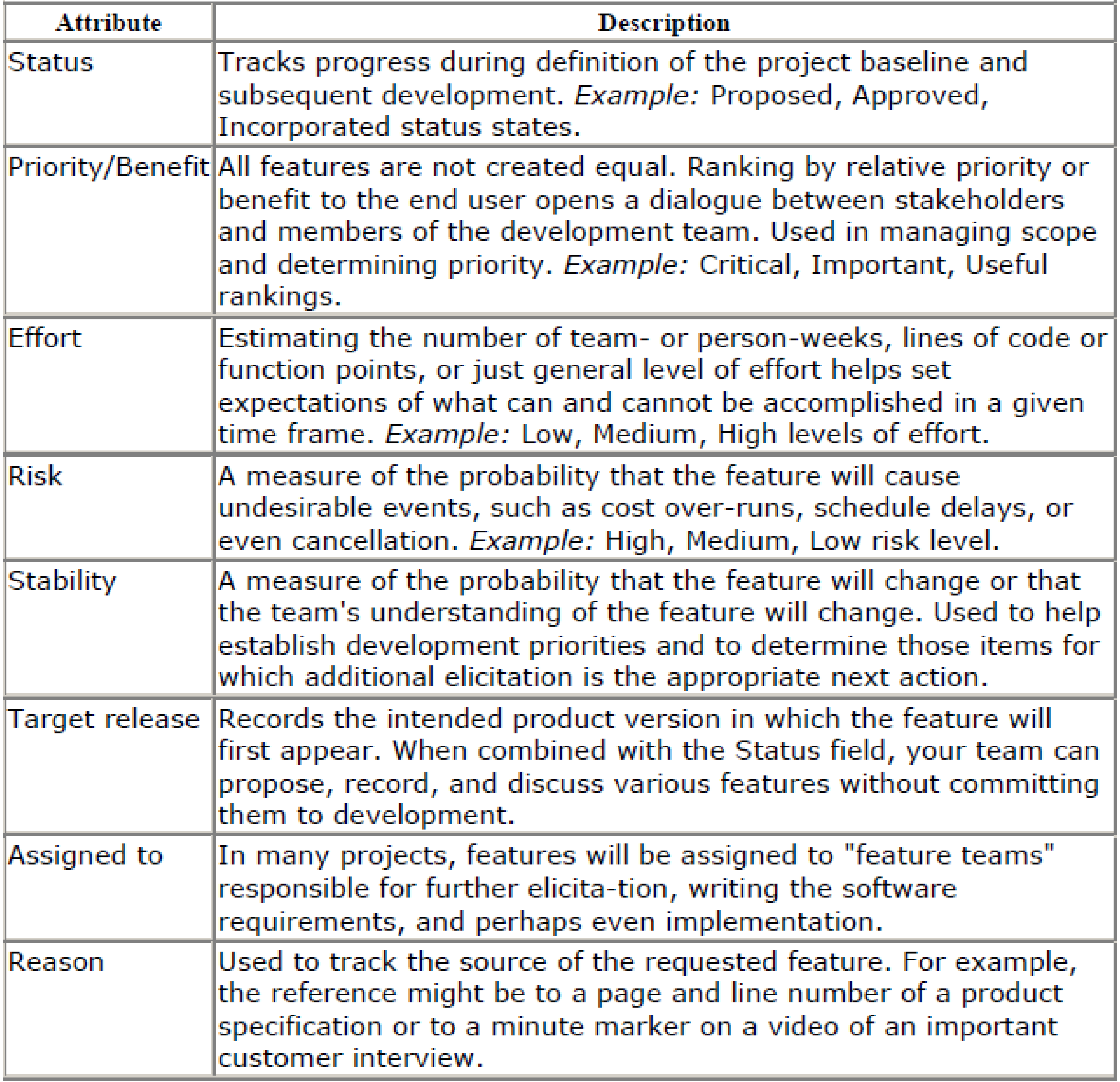
preferably 25 – 99 features

- Scope the features
  - "defer to a later release,"
  - "implement immediately,"
  - "reject entirely," or
  - "investigate further."

Attributes of product features

- used to relate the feature or requirements data to other types of project information
  - to track (name or unique identifier, state, history data, allocated from, traced-to, and so on),
  - to prioritize (priority field),
  - to manage (status) the features proposed for implementation

  - Example: *version number* might be used to record the specific software release in which we intend to implement a specific feature



Attribute	Description
Status	Tracks progress during definition of the project baseline and subsequent development. *Example:* Proposed, Approved, Incorporated status states.
Priority/Benefit	All features are not created equal. Ranking by relative priority or benefit to the end user opens a dialogue between stakeholders and members of the development team. Used in managing scope and determining priority. *Example:* Critical, Important, Useful rankings.
Effort	Estimating the number of team- or person-weeks, lines of code or function points, or just general level of effort helps set expectations of what can and cannot be accomplished in a given time frame. *Example:* Low, Medium, High levels of effort.
Risk	A measure of the probability that the feature will cause undesirable events, such as cost over-runs, schedule delays, or even cancellation. *Example:* High, Medium, Low risk level.
Stability	A measure of the probability that the feature will change or that the team's understanding of the feature will change. Used to help establish development priorities and to determine those items for which additional elicitation is the appropriate next action.
Target release	Records the intended product version in which the feature will first appear. When combined with the Status field, your team can propose, record, and discuss various features without committing them to development.
Assigned to	In many projects, features will be assigned to "feature teams" responsible for further elicita-tion, writing the software requirements, and perhaps even implementation.
Reason	Used to track the source of the requested feature. For example, the reference might be to a page and line number of a product specification or to a minute marker on a video of an important customer interview.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 9. Interviewing

Key Points

- Interviewing is a simple and direct technique.
- Context-free questions can help achieve bias-free interviews.
- Then, it may be appropriate to search for undiscovered requirements by exploring solutions.
- Convergence on some common needs will initiate a "requirements repository" for use during the project.
- A questionnaire is no substitute for an interview.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 10. Requirements Workshop

Key Points

- The requirements workshop is perhaps the most powerful technique for eliciting requirements.
- It gathers all key stakeholders together for a short but intensely focused period.
- The use of an outside facilitator experienced in requirements management can help ensure the success of the workshop.
- Brainstorming is the most important part of the workshop.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 11. Brainstorming & Idea Reduction

Key Points

- Brainstorming involves both idea generation and idea reduction.
- The most creative, innovative ideas often result from combining multiple, seemingly unrelated ideas.
- Various voting techniques may be used to prioritize the idea created.
- Although live brainstorming is preferred, Web-based brainstorming may be a viable alternative in some situations.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 12. Storyboarding

Key Points

- The purpose of storyboarding is to elicit early "Yes, But" reactions.
- Storyboards can be passive, active, or interactive.
- Storyboards identify the players, explain what happens to them, and describe how it happens.
- Make the storyboard sketchy, easy to modify, and unshippable.
- Storyboard early and often on every project with new or innovative content.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 13. Applying Use-Case

Key Points

- Use cases, like storyboards, identify the who, what, and how of system behavior.
- Use cases describe the interactions between a user and a system, focusing on what the system "does" for the user.
- The use-case model describes the totality of the system's functional behavior.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 14. Role Playing

Key Points

- Role playing allows the development team to experience the user's world from the user's perspective.
- A scripted walkthrough may replace role playing in some situations, with the script becoming a live storyboard.
- Class-Responsibility-Collaboration (CRC) cards, often used in object-oriented analysis, are a derivative of role playing.

Team Skill 2: UNDERSTANDING USER NEEDS

Chapter 15. Prototyping

Key Points

- Prototyping is especially effective in addressing the "Yes, But" and "Undiscovered Ruins" syndromes.
- A software requirements prototype is a partial implementation of a software system, built to help developers, users, and customers better understand system requirements.
- Prototype the "fuzzy" requirements: those that, although known or implied, are poorly defined and poorly understood.

Team Skill 3: DEFINING THE SYSTEM

Chapter 16. Organizing Requirements Information

Key Points

- For nontrivial applications, requirements must be captured and recorded in a document database, model, or tool.
- Different types of projects require different requirements organization techniques.
- Complex systems entail requirements specification for each subsystem.

Multi-person effort demands documentation

- Requirements *must* be captured and documented
- All parties must reach agreement about what system is being built
- The *requirements specification* for a system or application describes the external behavior of that system

Requirements specification

- Can be contained in a:
  - Document
  - Database
  - use-case model
  - Requirements repository
  - or a combination of these elements
- Due to system complexity, it cannot be recorded in a single monolithic document

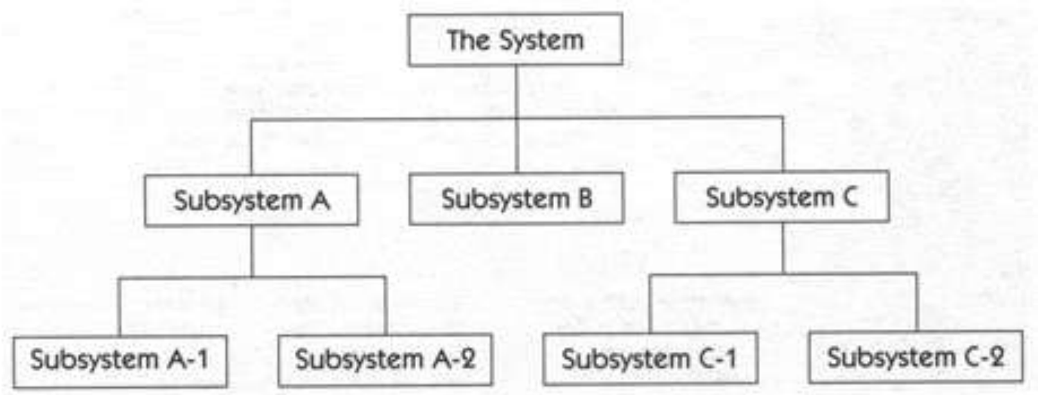
Parent documents required

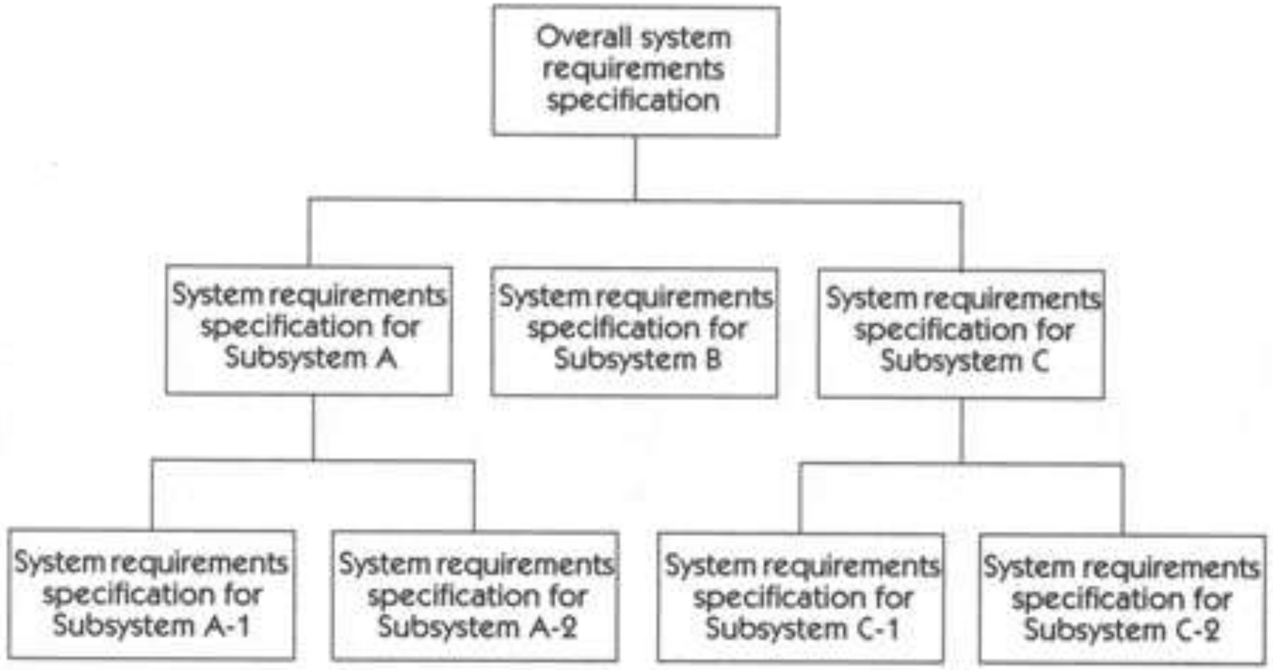
- System Requirements Specification
  - Parent document defining overall system requirements
    - hardware, software, people, and procedures
- Vision document
  - defines the features of the system in general terms
- Product family requirements
  - defines the full set of requirements for a family of products
- Business requirements / Marketing requirements document
  - describes the overall business requirements and business environment in which the product will reside

System Requirements Specification (SRS)

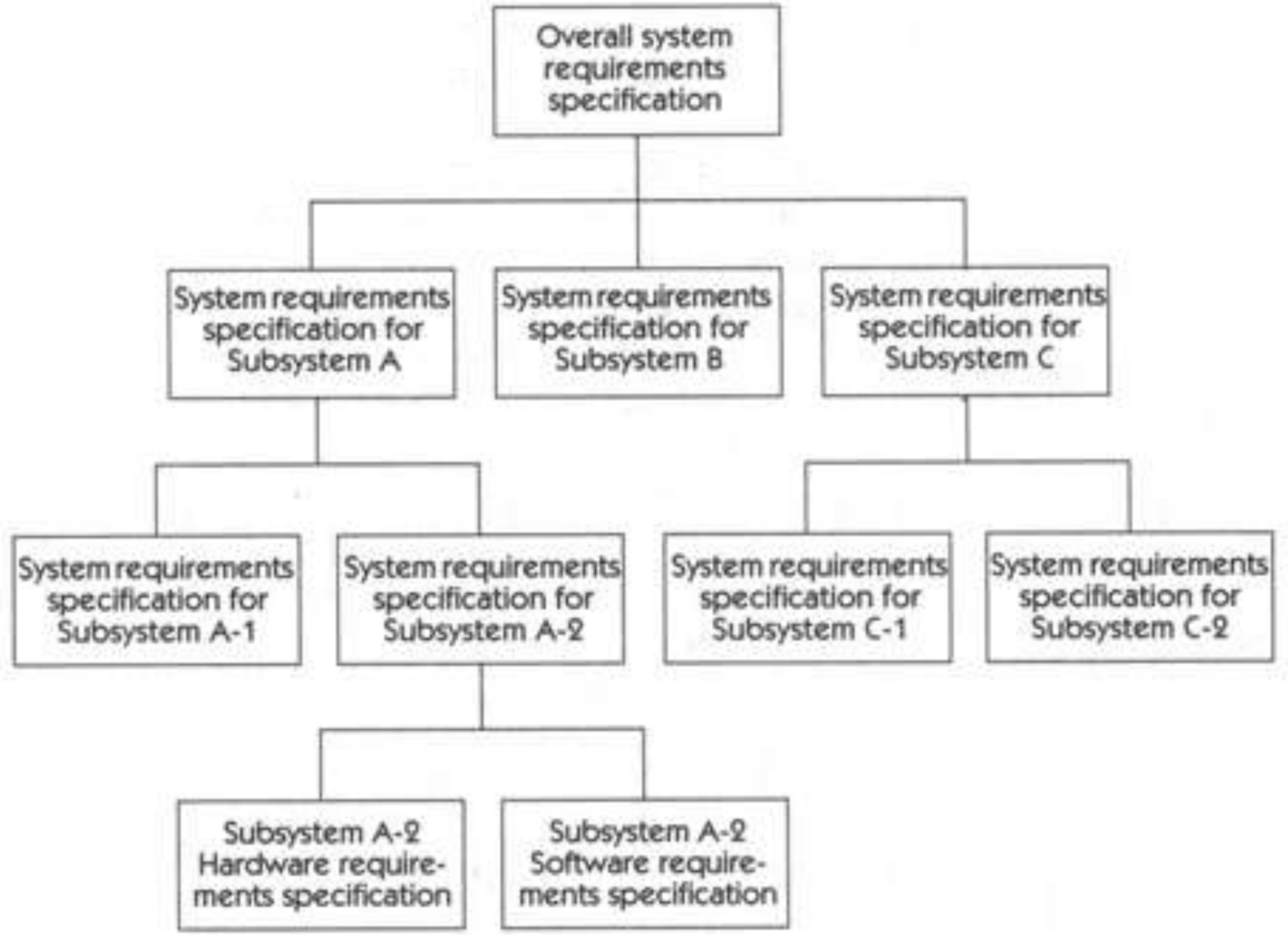
- Defines requirements of software piece
- Specific details shown
- Just one specific application and for one specific release
- defines the external behavior of the system being built

Here is a complex system & its requirements




Overall system requirements specification
System requirements specification for Subsystem A
System requirements specification for Subsystem B
System requirements specification for Subsystem C
System requirements specification for Subsystem A-1
System requirements specification for Subsystem A-2
System requirements specification for Subsystem C-1
System requirements specification for Subsystem C-2

Hierarchy of resulting specifications, including software and hardware levels

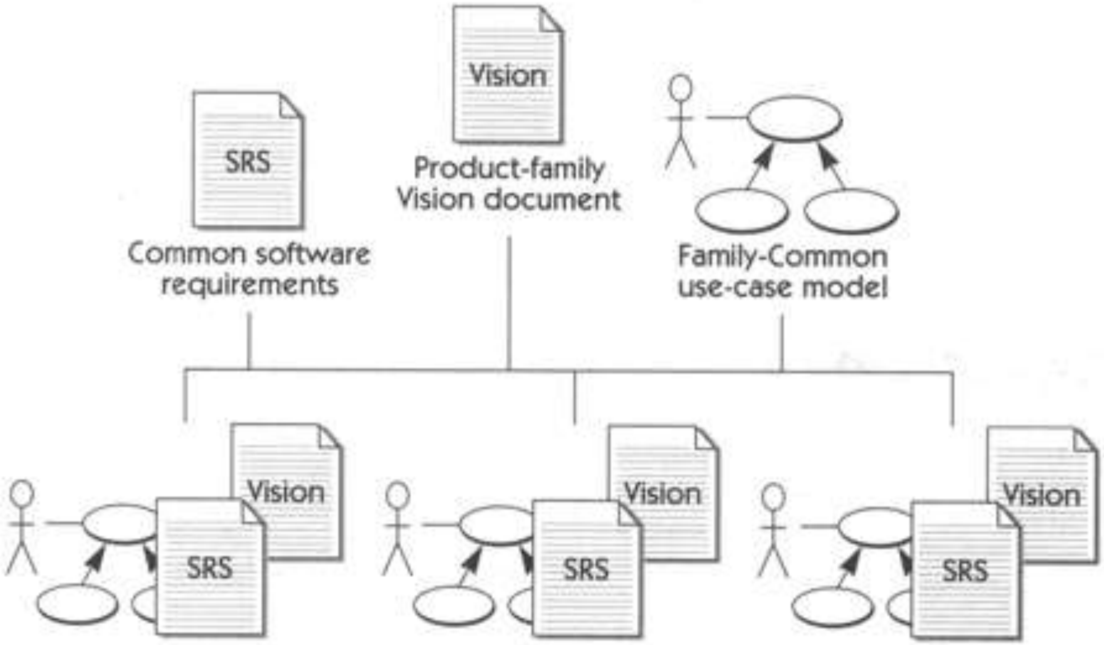

Overall system requirements specification
System requirements specification for Subsystem A
System requirements specification for Subsystem B
System requirements specification for Subsystem C
System requirements specification for Subsystem A-1
System requirements specification for Subsystem A-2
System requirements specification for Subsystem C-1
System requirements specification for Subsystem C-2
Subsystem A-2 Hardware requirements specification
Subsystem A-2 Software requirements specification

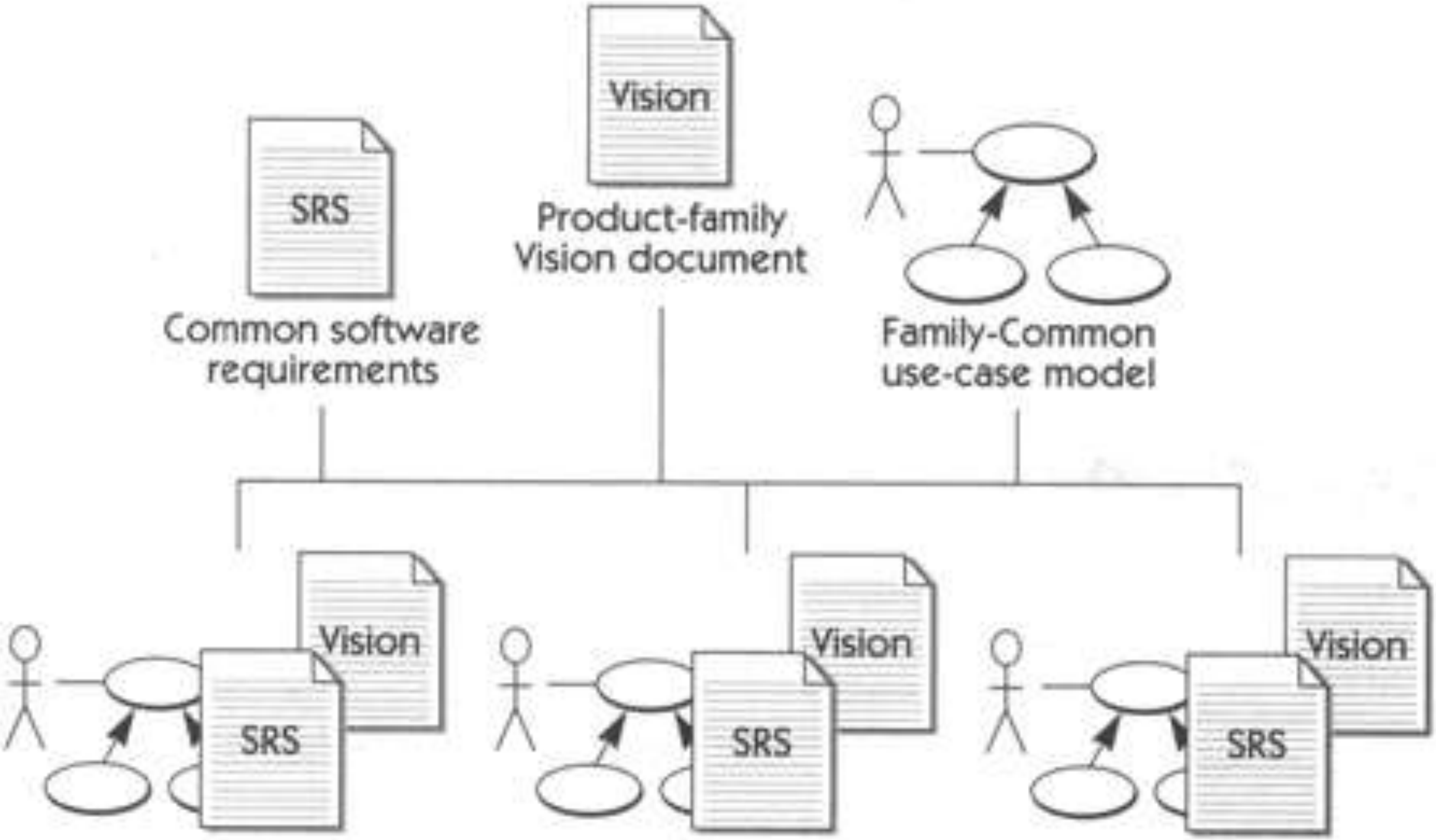
Organizing Requirements for Product Families

- Develop a product-family *Vision document:* describes the ways in which the products are intended to work together; other features that could be shared.
- To understand the shared-usage model: develop a set of *use cases* showing how the users will interact with various applications running together.
- Develop *common software requirements specification:* defines specific requirements for shared functionality, such as menu structures and communication protocols.
- For each product in the family, develop a *Vision document, software requirements specification,* and a *use-case model* that defines its specific

Requirements organization for a software product family

- Changes in parent document needs to be recorded individually in each sub-set
- Even use a requirements tool to manage tl

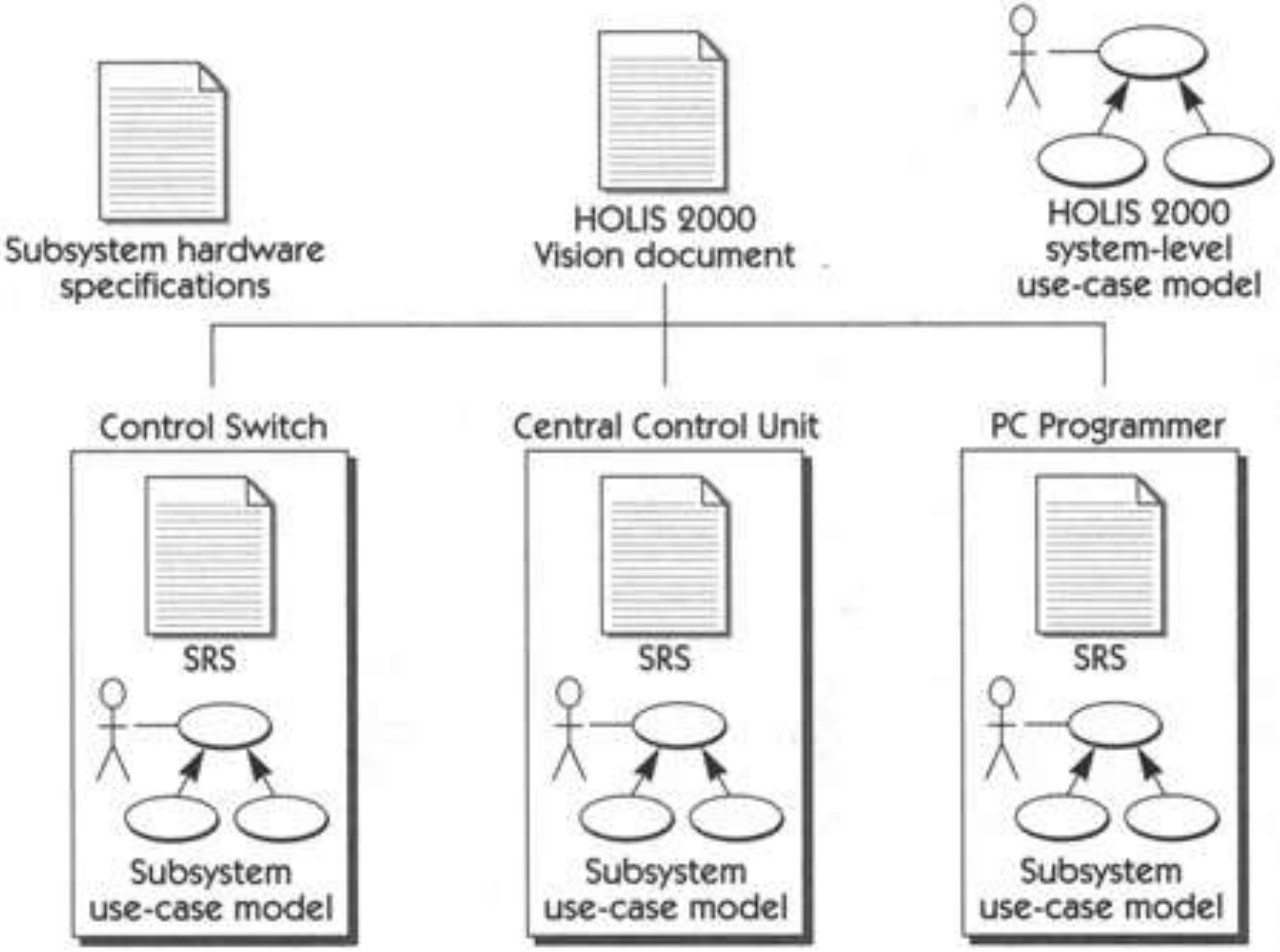

Vision
Product-family Vision document
SRS
Common software requirements
Family-Common use-case model
Vision
SRS
Vision
SRS
Vision
SRS


Vision
Product-family
Vision document
SRS
Common software
requirements
Family-Common
use-case model
Vision
SRS
Vision
SRS
Vision
SRS

Case Study - HOLIS

- The ***Vision document:*** short-term and longer-term visions for HOLIS, including basic system-level requirements and the features that are being proposed.
- The ***system-level use-case model*** how various actors in the system interact with HOLIS.
- Common hardware requirements—size, weight, power, packaging—for HOLIS's three subsystems in a **single *hardware requirements specification.***
- As each subsystem of HOLIS is quite software intensive, the team decided to develop a ***software requirements specification*** for each of the three subsystems, as well as a ***use-case model*** for how each subsystem interacts with its various actors.

Case Study - HOLIS


Subsystem hardware specifications
HOLIS 2000 Vision document
HOLIS 2000 system-level use-case model
Control Switch
SRS
Subsystem use-case model
Central Control Unit
SRS
Subsystem use-case model
PC Programmer
SRS
Subsystem use-case model