



Recall The Team Skills

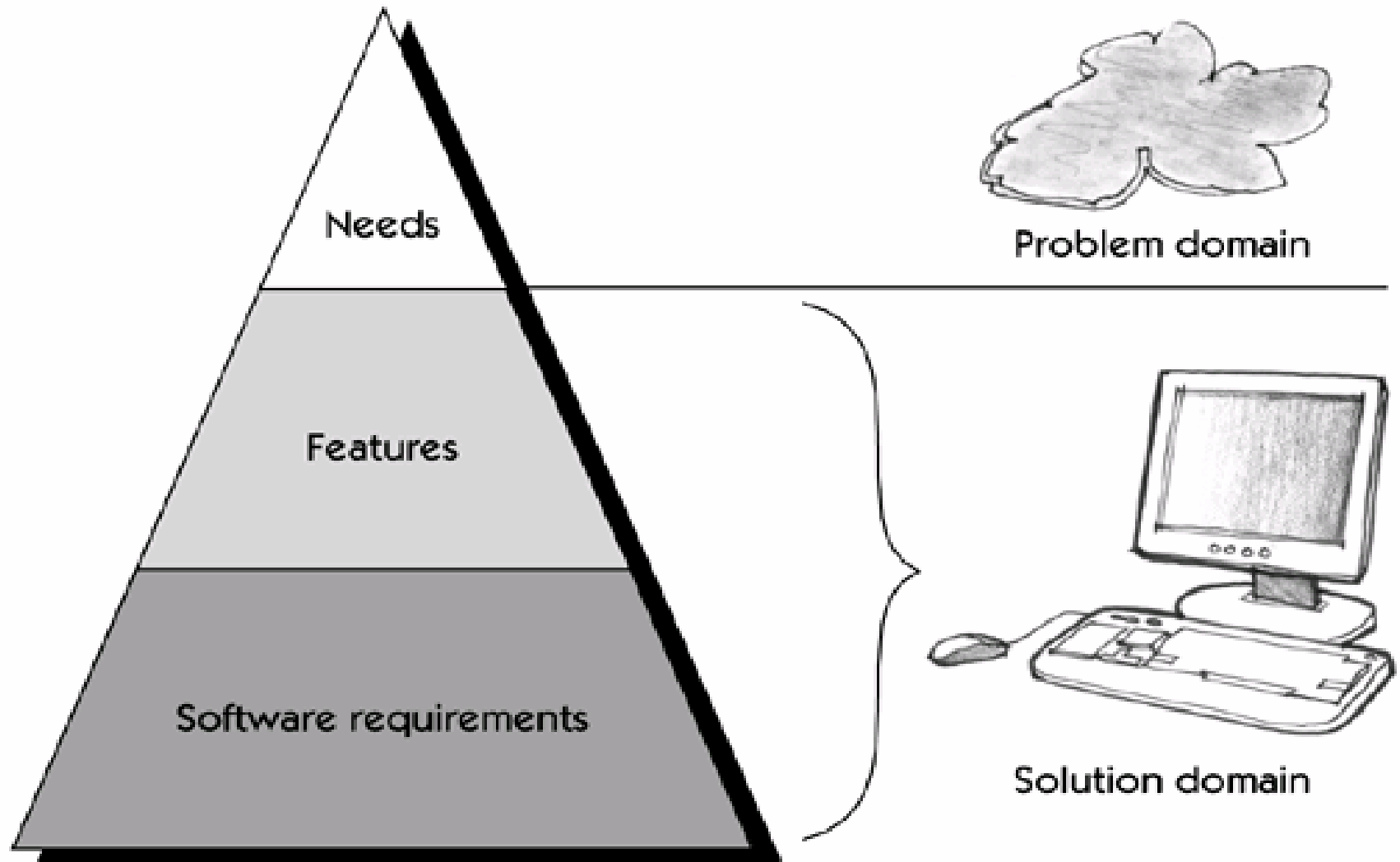
1. Analyzing the Problem (with 5 steps)
 1. Gain agreement on the problem definition.
 2. Understand the root causes
 3. Identify the stakeholders and the users.
 4. Define the solution system boundary.
 5. Identify the constraints
 2. Understanding User and Stakeholder Needs
 3. Defining the System
 4. Managing Scope
 5. Refining the System Definition
 6. Building the Right System
-

Chapter 9

The Features of a Product or System

- ❑ Stakeholders and user needs
- ❑ Features and examples
- ❑ Attributes of features

Recall: The Requirements Pyramid






Stakeholder and User Needs

- **A stakeholder need** is a reflection of the business, personal, or operational problem that must be addressed in order to justify consideration, purchase, or use of a new system.
 - The development team will **build a better system only if it understands** the true needs of the stakeholder.
 - That knowledge will give the team the information it needs to make **better decisions** in the definition and implementation of the system.
-




Stakeholder and User Needs

- Often, these stakeholder and user needs will be vague and ambiguous. For example:
 - "I need easier ways to understand the status of my inventory"
 - "I'd like to see a big increase in the productivity of sales order entry"
 - These statements set an important context for all the activities that follow.
 - Therefore, it is important to spend some significant time and energy trying to understand them.
- 



Features

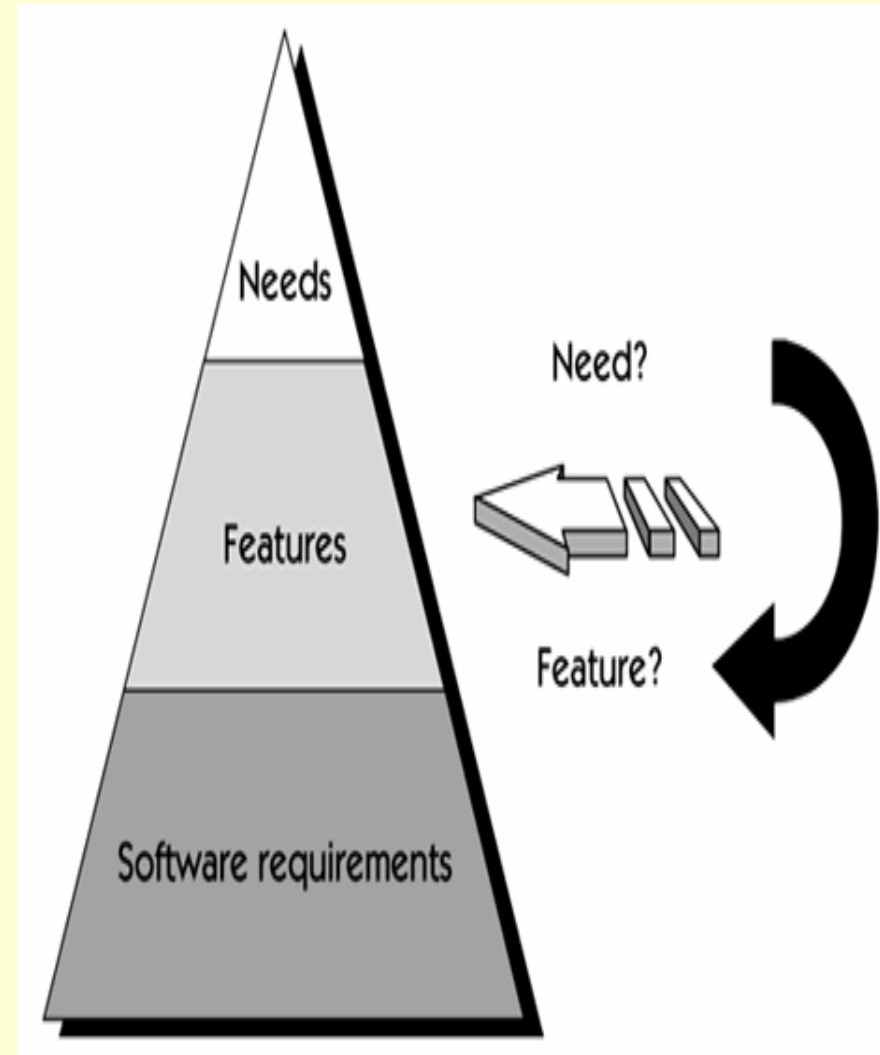
- A **feature** is a service the system provides to fulfill one or more stakeholder needs. It is
 - A **High-level expressions** of desired system behaviour.
 - A **convenient way** to describe functionality without getting bogged down in detail.
 - Features are **often not well defined** and may even be in conflict with one another. Example:
 - “I want an increased order processing rates” and
 - “I want to provide a far more user-friendly interface to help our new employees learn the system”
- 

Examples of Features

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 XP compatibility

Needs and Features

- Without an understanding of the need behind the feature, there will be a real risk.
- If the feature does not solve the real need for any reason, then the system may fail to meet the users' objectives even though the implementation delivered the feature they requested.






Managing the complexity of the system

- 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 features.
 - Although, fewer than 50 is preferred.
 - Later on, these features will be refined to get the software requirements.
-



Managing the complexity of the system

- 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 of Features

- **Attributes** are data elements that help provide additional information about the features.
- They are
 - used to relate the features to other types of project information.
 - used to track, prioritize, and manage the features.





Attributes of Features

Attribute	Description
Status	Tracks progress during definition of the project baseline and subsequent development. <i>Example:</i> Proposed, Approved, Incorporated status states.
Priority/benefit	Assists in managing scope and determining priority. 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. <i>Example:</i> Critical, Important, Useful rankings.
Effort	Helps establish realistic schedules, goals, and costs. Estimating the number of team- or person-weeks, lines of code or function points, or just the general level of effort helps set expectations of what can and cannot be accomplished in a given time frame. <i>Example:</i> Team months, or Low, Medium, High levels of effort.
Risk	Indicates a measure of the probability that the feature will cause undesirable events, such as cost overruns, schedule delays, or even cancellation. <i>Example:</i> High, Medium, Low risk level.
Stability	Reflects a measure of the probability that the feature will change or that the team's understanding of the feature will change. This attribute helps establish development priorities and determine those items for which additional elicitation is the appropriate next action. <i>Example:</i> High, Medium, Low stability.
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. <i>Example:</i> Version 2.3.
Assigned to	Indicates who will be responsible for implementing the feature. In many projects, features will be assigned to "feature teams" responsible for further elicitation, writing the software requirements, and perhaps even implementation.
Reason	Helps 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.



Key Points

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