



# Recall The Team Skills

1. Analyzing the Problem (with 5 steps)
  2. Understanding User and Stakeholder Needs
  3. Defining the System
    - A Use Case Primer
    - Organizing Requirements Information
    - The Vision Document
  4. **Managing Scope**
  5. Refining the System Definition
  6. Building the Right System
- 



# Team Skill 4: Managing scope

- Ch 18: Establishing project scope
- Ch 19: Managing your customer




# *Chapter 18*

## *Establishing Project Scope*

- Project scope problem
- The Requirements Baseline
- Setting Priorities
- Assessing Effort
- Adding the Risk Element
- Reducing Scope



# The Problem of Project Scope

- Project scope (complexity) depends on:
    - The functionality that has to meet the user's needs
    - The resources available to the project
    - The time available for the implementation
  - Project scope derives from the following elements.
    - Resources consisting of the labor: developers, testers, tech writers, quality assurance personnel, and others.
    - Time is perhaps a "soft" boundary that is subject to change if the available resources are inadequate to achieve the desired functionality.
- 




# The Problem of Project Scope

- If the effort required to implement the system features is equal to the resources available during the scheduled time, the project has an achievable scope, and we have no problem.
- But what if not ...?






# The Problem of Project Scope

- What happens when a project proceeds with a 200% initial scope?
  - If the intended features of the application were completely independent, which is unlikely, only half of them will be working when the deadline passes.
  - If some of the application features do depend on others, at deadline time nothing useful will work.
- 




# The Problem of Project Scope

- What happens to software quality during either of these outcomes?
  - The code, which is rushed to completion near the end, is poorly designed and bug-ridden;
  - testing is reduced to an absolute minimum or skipped entirely;
  - and documentation and help systems are eliminated.
- 



# The Hard Question

- The hard question: How does one manage to reduce scope and keep the customers happy?
  - Brooks' law states that adding labor to a late software project makes it even later.
  - We need another way to solve the scope problem
  - **Solution:** If we truly begin the development effort with an expectation of 200% scope, it will be necessary to reduce the project scope by as much as a factor of two in order to have any chance of success. ... But how?
- 





# Solution: Managing scope

Feature 1

Feature 2

to be implemented

Feature 3

..

----- Requirement baseline -----

...

Feature 4

Feature 5


to be postponed for future

Feature 6





# The Requirements Baseline

- The requirements baseline is the itemized set of features intended to be delivered in a specific version of the application.
  - The baseline must ...
    - Be at least "acceptable" to the customer
    - Have a reasonable probability of success, in the team's view
  - How do we define it?
- 




# Setting Priorities

- During prioritization, it is important that the customers and users, product managers, or other representatives — not the development team — set the initial priorities.






# Assessing Required Effort

- The next step is to establish the rough level of effort implied by each feature of the proposed baseline.
  - The best we can do is to determine a "rough order of magnitude" (High, Medium, Low) of the level of required effort for each feature. Why?
    - Little useful information is available yet on which to estimate the work
    - No detailed requirements or design output on which to base an estimate.
- 



# Adding the Risk Element

- The Risk associated with each feature is the probability that the implementation of a feature will cause an adverse impact on the schedule and/or the budget.
  - A high-risk feature has the potential to impact the project negatively, even if all other features can be accomplished within the allotted time.
  - The development team establishes risk based on any heuristic it is comfortable with, using the same low-medium-high scale used to assess effort
- 

# Example: Prioritized Features List with Effort and Risk Estimates

Feature	Priority	Effort	Risk
Feature 1: External relational database	Critical	Medium	Low
Feature 4: Portability to a new OS release	Critical	High	Medium
Feature 6: Import of external data by style	Critical	Low	High
Feature 3: Ability to clone a project	Important	High	Medium
Feature 2: Multiuser security	Important	Low	High
Feature 5: New project wizard	Important	Low	Low
Feature 7: Implementation of tool tips	Useful	Low	High
Feature 8: Integration with a version-manager subsystem	Useful	High	Low

# Reducing Scope

## Scope Prioritization Techniques


Attributes	Consider
<i>Priority:</i> Critical	Alarm! Establish immediate risk-mitigation strategy; resource immediately; focus on feasibility with architecture.
<i>Effort:</i> High	
<i>Risk:</i> High	
<i>Priority:</i> Critical	A likely critical resource-constrained item; resource immediately.
<i>Effort:</i> High	
<i>Risk:</i> Low	
<i>Priority:</i> Critical	Resource as a safety factor, or defer until later.
<i>Effort:</i> Low	
<i>Risk:</i> Low	



# Example: Final Prioritized Features List

Feature	Priority	Effort
Feature 1: External relational database support	Critical	Medium
Feature 4: Portability to a new OS release	Critical	High
Feature 6: Import of external data by style	Critical	Low
Feature 3: Ability to clone a project	Important	High
<b>Baseline</b> (features above this line are committed features)		
Feature 2: Multiuser security	Important	Low
Feature 5: New project wizard	Important	Low
Feature 7: Implementation of tool tips	Useful	Low
Feature 8: Integration with a version-manager subsystem	Useful	High


🌐 Features below the baseline are now future features and will be considered in later releases.







# Making the cut

- Is it enough to do all critical features?
  - Can we include some important ones?
  - Are there any dependent features?
  - Useful usually can be cut out.
  - Many possible future cuts.
  - Cuts can be changed as project progress.
- 



# Reading Assignment

- Read the scope management for HOLIS in pages 218-222.





# Key Points

- **Project scope** is a combination of product functionality, project resources, and available time.
  - **Brooks' law** states that adding labor to a late software project makes it even later.
  - If the **effort required** to implement the system features is **equal to the resources** available during the scheduled time, the project has an **achievable scope**.
  - **Over scoped projects are typical**. In many projects, it will be necessary to reduce the scope by as much as a factor of two.
  - **The first step** in establishing project scope is to establish a high-level **requirements baseline**.
- 