SOFTWARE REQUIREMENTS ENGINEERING

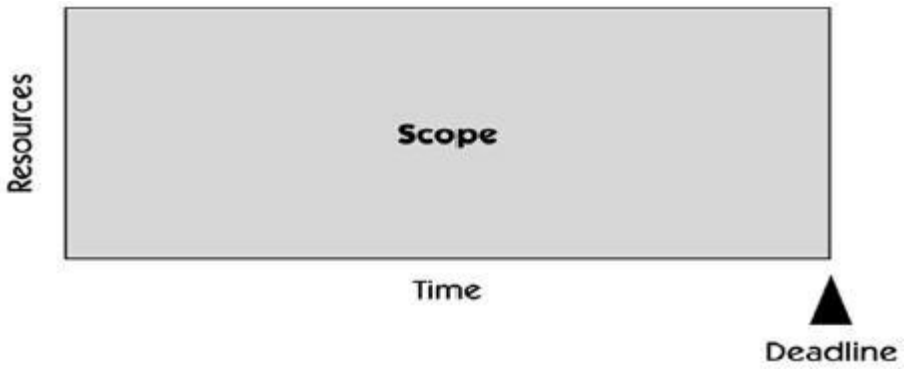
MANAGING THE SCOPE

Project Scope

- As with any professional activity, meeting commitments in application development involves making realistic assessments of project resources, time lines, and objectives before the activity begins. For software development, these factors combine to create the "scope" of the project.

- Project scope is a function of:
  - The functionality that must be delivered to meet the user's needs
  - The resources available to the project
  - The time available in which to achieve the implementation

- Figure below provides a perspective of the "box" we can use to represent project scope.



The Problem of Project Scope

- Resources, consisting primarily of the labor from requirement engineer, developers, testers, tech writers, quality assurance personnel, and others.

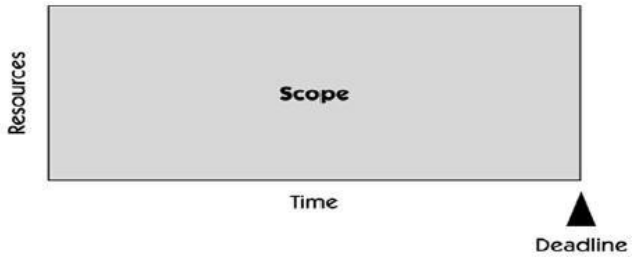
- Fred Brooks [1975] demonstrated that adding resources to a software project in order to increase the work output is a risky proposition at best. Indeed, **Brooks' law states that adding labor to a late software project makes it even later. [2]**

- Brooks points to two main factors that explain why it works this way:
  - ✓ It takes some time for the people added to a project to become productive. Brooks calls this the "ramp up" time.
  - ✓ Communication overheads increase as the number of people increases.

The Problem of Project Scope

- If we want to ensure our credibility and gain the confidence of our customers, it is important that we not slip the schedule.

- So, for purposes of scope analysis, we'll treat time as a fixed factor. The total functionality we can deliver is obviously limited by the available time (fixed) and the available resources (also fixed), so the achievable scope is the area of the box.



- 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. However, experience has shown that there is often a poor match between these factors in the scope equation.

The Problem of Project Scope

- What happens when a project proceeds with a 200 percent initial scope?

  - ✓ If the intended features of the applications were completely independent, which is unlikely, only half of them will be working when the deadline passes. The product provides only half of the intended utility. The features don't work together, and don't produce any useful combined functionality.

  - ✓ An application with reduced scope will be deployed in this case.

  - ✓ Consequences include::
    - Seriously unhappy customers
    - Missed marketing commitments and
    - Inaccurate manuals and promotional materials

The Problem of Project Scope

- What happens to software quality under these circumstances?

  - ✓ The code, which is rushed to completion near the end, is poorly designed and buggy
  - ✓ Testing is reduced to an absolute minimum or skipped
  - ✓ Documentations and help systems are eliminated
  - ✓ Customers take on both the testing and the quality assurance functions.
  - ✓ Soon, the customers react to our extraordinary efforts as follows:

    - **"Although we were initially disappointed by how late you were, now we are really unhappy because we just discovered that what you shipped us is junk."**

The Requirements Baseline

- A primary technique in ***scope management*** is to establish a **high-level requirements baseline** for the project (Figure 2).

- Baseline can be viewed as an itemized set of features intended to be delivered in a specific version of the application.

- This baseline for the next release must be agreed to by both the customer and the development team.

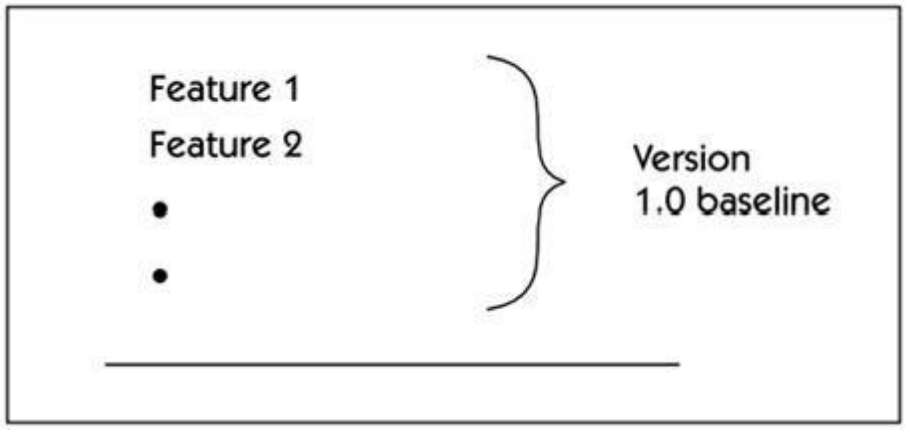


Figure 2:: Requirements baseline

The Requirements Baseline

- The first step in creating the baseline is simply to list the features that have been defined for the application.
- It is suggested that any new system, no matter how complex, can be described by a list of 25–50 features.
- If we follow requirements workshop or any other elicitation technique, we will get a list of proposed features. This list provides a high-level description of the capabilities of a new or revised system.

The Requirements Baseline

- For our example, let's consider a software product with the following eight features.

  - ✓ Feature 1: SQL database support
  - ✓ Feature 2: Multi-user security
  - ✓ Feature 3: Provide Modification facility to Employee
  - ✓ Feature 4: Employee Complete Information bank
  - ✓ Feature 5: Generation of Employee Report
  - ✓ Feature 6: Automatic Leave management support
  - ✓ Feature 7: Online Notification facility for Employee Information
  - ✓ Feature 8: Integration with other databases like MS Access

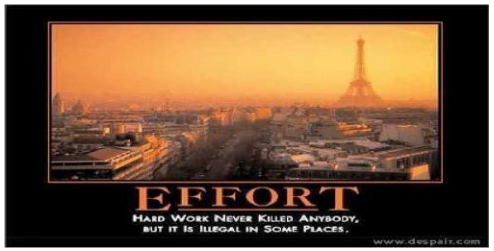
Setting Priorities

- Establishing the relative priorities for the feature set is important to scope management
- During prioritization, it is important that the customers and users, or other representatives—not the development team—set the initial priorities.
- In our project example, let's assume that we vote on the priority of each feature, using a critical-important-useful scale; the results of this exercise are shown in Table 1.

Table 1:: Prioritized Features List

Features	Priorities
Feature1: SQL database support	Critical
Feature4: Employee Complete Information bank	Critical
Feature6: Automatic Leave management support	Critical
Feature3: Provide Modification facility to Employee	Important
Feature2: Multiuser security	Important
Feature5: Generation of Employee Report	Important
Feature7: Online Notification facility for Employee Information	Useful
Feature8: Integration with other database	Useful

Assessing Effort



- The next step is to establish the **rough level of effort** implied by each feature of the proposed baseline.

- The team may wish to use "team-weeks" or "team-months" as a gross estimate of the total impact of a feature on the project. This rough estimate serves as a substitute for a more detailed estimate.

- Then, using these totals and the total time available for the project, the team can determine where to initially draw the baseline.

Assessing Effort

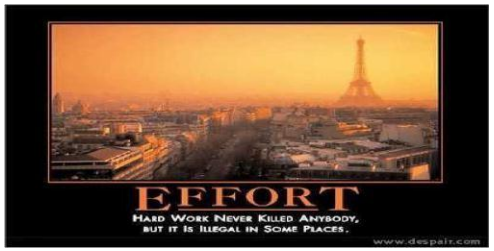


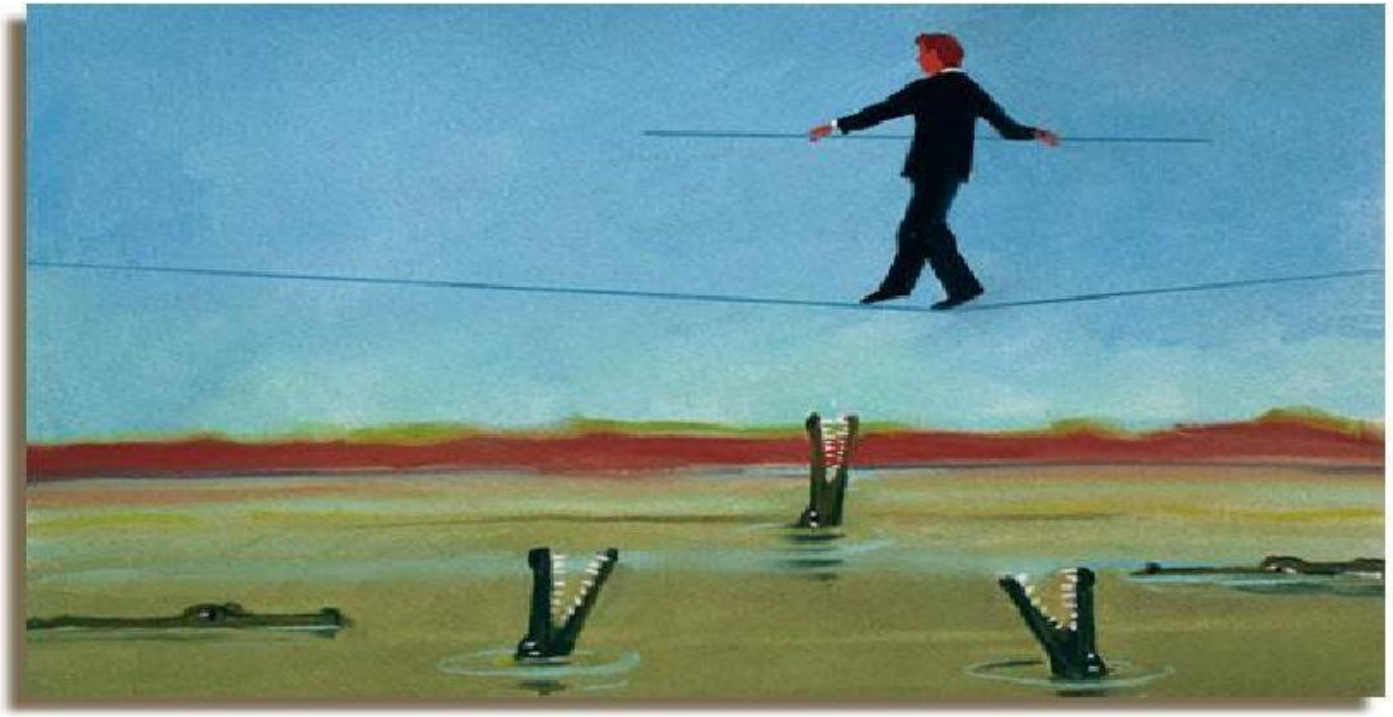
Table 2:: Prioritized Features List with Effort Estimates

Features	Priorities	Effort
Feature1:SQL database support	Critical	Medium
Feature 4: Employee Complete Information bank	Critical	High
Feature 6: Automatic Leave management support	Critical	Medium
Feature 3: Provide Modification facility to Employee	Important	Low
Feature 2: Multiuser security	Important	Medium
Feature 5: Generation of Employee Report	Important	Low
Feature 7: Online Notification facility for Employee Information	Useful	Low
Feature 8: Integration with other database	Useful	High

Adding the Risk Element

- Another aspect of managing scope is estimating the "risk" associated with each feature.

- In this context, we'll consider risk to be the probability that the implementation of a feature will cause an adverse impact on the schedule and/or the budget.



Adding the Risk Element

- Risk gives us a measure of the potential impact of including a particular feature within the project baseline. A high-risk feature has the potential to impact the project negatively.

- Table 3 shows the revised features list also including the Effort and Risk Estimates for the prioritized features on a scale of Low, Medium, High.

Table 3:: Prioritized Features List with Effort and Risk Estimates

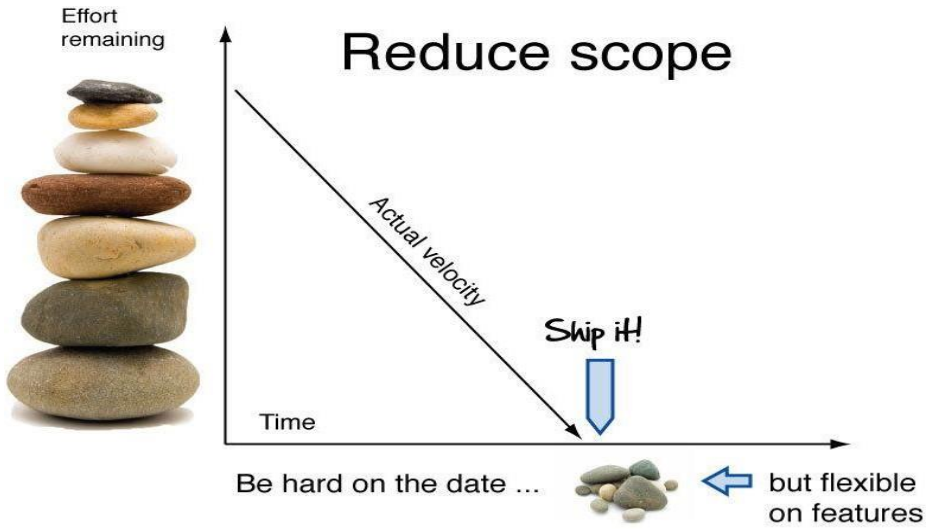
Features	Priorities	Effort	Risk Estimates
Feature1: SQL database support	Critical	Medium	Low
Feature 4: Employee Complete Information bank	Critical	High	Medium
Feature 6: Automatic Leave management support	Critical	Medium	Low
Feature 3: Provide Modification facility to Employee	Important	Low	Medium
Feature 2: Multiuser security	Important	Medium	High
Feature 5: Generation of Employee Report	Important	Low	Low
Feature 7: Online Notification facility for Employee Information	Useful	Low	Low
Feature 8: Integration with other database	Useful	High	Medium

Scope Reduction

- We now have ***a prioritized features set with associated relative effort and risk.***

- Note that there is often **little correlation** between priority and effort or between priority and risk. Indeed, many critical items require low effort; many items that are only useful are very difficult.

- For example, a feature that is critical priority, medium effort, and low risk may be a candidate for immediate resourcing.



Scope Reduction

- If expectations are properly set and managed, anything that can be accomplished beyond the baseline will be a bonus. Table 4 applies this simple heuristic to the baseline for our sample project.

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

Table 4:: Final Prioritization List

Features	Priorities	Effort	Risk Estimates
Feature 1: SQL database support	Critical	Medium	Low
Feature 4: Employee Complete Information bank	Critical	High	Medium
Feature 6: Automatic Leave management support	Critical	Medium	Low
Feature 3: Provide Modification facility to Employee	Important	Low	Medium
Baseline (features above this line are committed features)			
Feature2: Multiuser security	Important	Medium	High
Feature 5: Generation of Employee Report	Important	Low	Low
Feature 7: Online Notification facility for Employee Information	Useful	Low	Low
Feature 8: Integration with other database	Useful	High	Medium

Managing Your Customer

Engaging Customers to Manage Project Scope

- Reducing project scope to within shouting distance of available time and resources has the potential to create a pleasing relationship between the project team and the customers,.

- We can actively engage our customers in managing their requirements and project scope to ensure both the quality and the timeliness of the software outcomes.

Communicating the Result

- If the project scope must be reduced, ***make sure that the customer is a direct participant.*** A customer who is part of the process will own the result. A customer who is excluded from the process will be unhappy and tend to blame the developers for not trying hard enough.

- Engaging the customer in this **dialogue helps to avoid the embarrassment of missed schedules and missing features.** Any extra features accomplished beyond the baseline will be perceived positively.

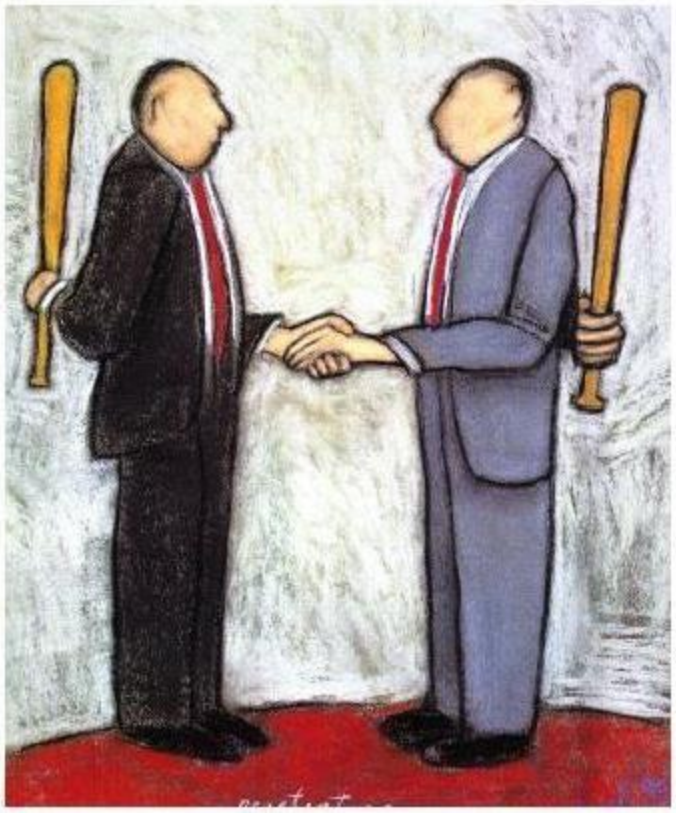
Negotiating with the Customer

- Almost all business processes require negotiation.

- Consider negotiating with a customer for a delivery date, negotiating your annual increase with your manager, negotiating an achievable quota for your sales team, or negotiating additional resources for your project.

Negotiating with the Customer

- On behalf of both your project and your customer's business objective, you will need to negotiate the scope commitment for your team.
- As you negotiate with your customer, your guiding principle in establishing the baseline should be under promise and over deliver
- *Doing so ensure that unavoidable problems during software development, unanticipated technological risks, changing requirements, delays in the availability of purchased components, a key team member's unanticipated leave, and so on can be accommodated within your project schedule.*



Managing the Baseline

- Successful development managers create margins for error in estimating effort and allow for time to include appropriate changes during the development cycle.
- These managers also resist feature creep, which Weinberg [1995] notes can increase scope by as much as 50 percent to 100 percent after the start of a project.
- With scope negotiated to an achievable level, and with development focused on the customer's "must haves," the team will establish credibility by meeting schedules with quality and utility.
- However, your customers want as much functionality as possible with each release of a software system to meet their business objectives but the demand for more and more functionality can compromise the quality

Managing the Baseline

- Perhaps most important, the high-level baseline can be used as part of an effective change management process.

- **Official Changes**
  - ✓ The features baseline provides an excellent mechanism or managing high-level change.
  - ✓ For example, when the customer requests a new system capability (an official change) and that capability is not part of the features baseline, the impact of the change must be assessed before including the new feature in the baseline.
  - ✓ If the resources and schedule are fixed, the project team must engage the customer in a decision-making process that prioritizes the new feature relative to the other features scheduled for the release.

Managing the Baseline

- If the new feature is critical, it must, by definition, be included in the release, and the customer and the project team should jointly determine which features will be excluded from the release or at least lowered in priority.

- If the feature is important but not critical, the project team can proceed with the implementation of the feature on a best-efforts basis, allowing progress to dictate whether the feature makes the release.

Managing the Baseline

Un-Official Changes

- The problem of customer-initiated change may be the easiest scope management challenge to handle. It is externally focused, and the impact of change can be assessed and made clear to this external stakeholder.

- However, experience shows that another class of change threat known as —Un-official Change is even more challenging to the development process.

- Weinberg calls those un-official changes the —Requirements Leakage which are those requirements that entered the system without visibility to the team members responsible for managing to schedule, budget and quality criteria

- These Un-official changes contributes up to half of the total scope of one project. In other words, half of the total work product of the system was invested in requirements leakage

References

1. Managing Software Requirements: A Use Case Approach, Second Edition By Dean Leffingwell, Don Widrig, Addison- Wesley
2. http://en.wikipedia.org/wiki/Brooks's_law

For any query Feel Free to ask