SOEN – 6841
Software Project Management

Amin Ranj Bar

Winter, 2021

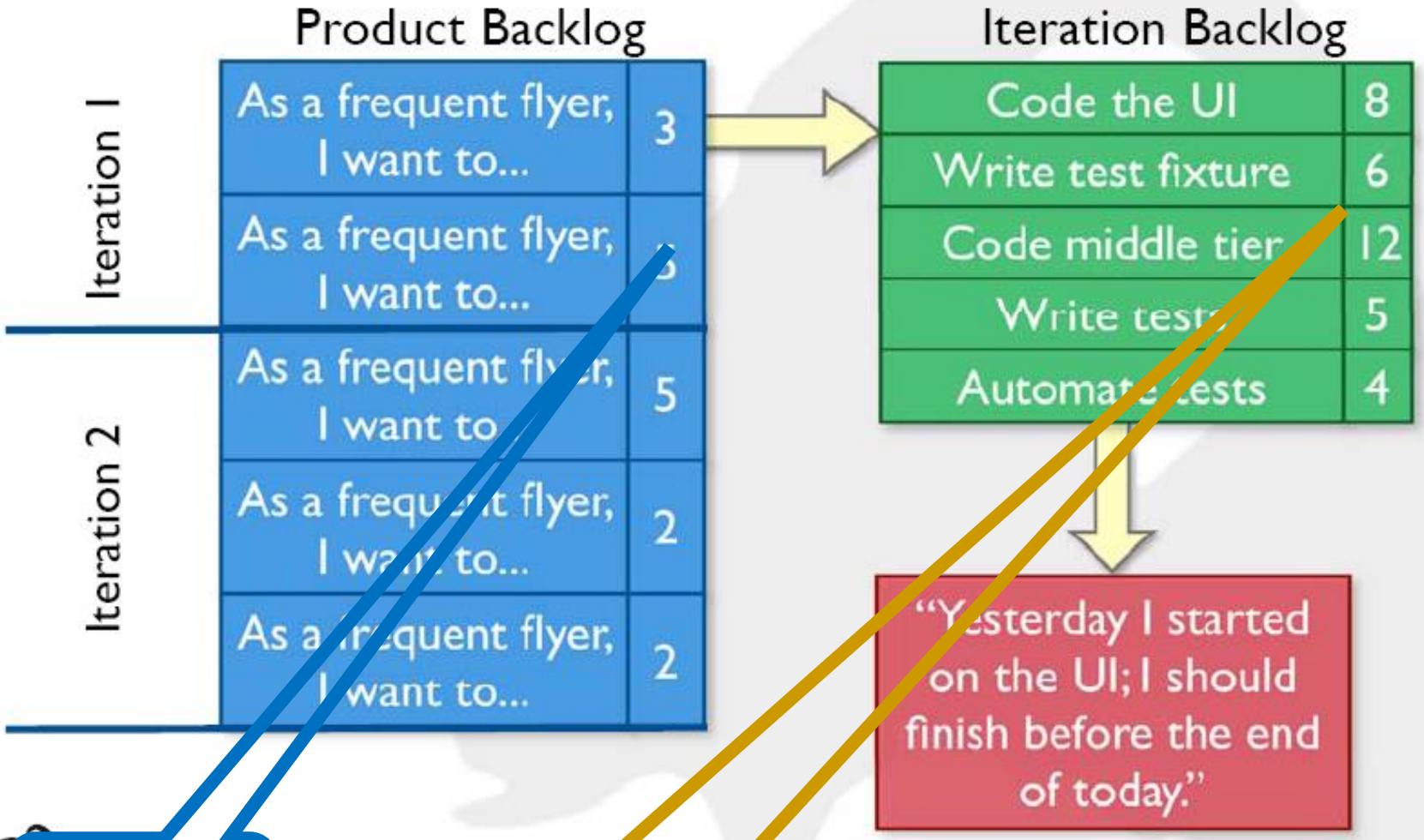
Overview

- Agile estimating and planning
- Project scope and documentation
- Budgeting. Accurate Estimating.

Release Plan v.s. Iteration Plan

	Release Plan	Iteration Plan
Planning horizon	3–9 months	1–4 weeks
Items in plan	User stories	Tasks
Estimated in	Story points or ideal days	Ideal hours

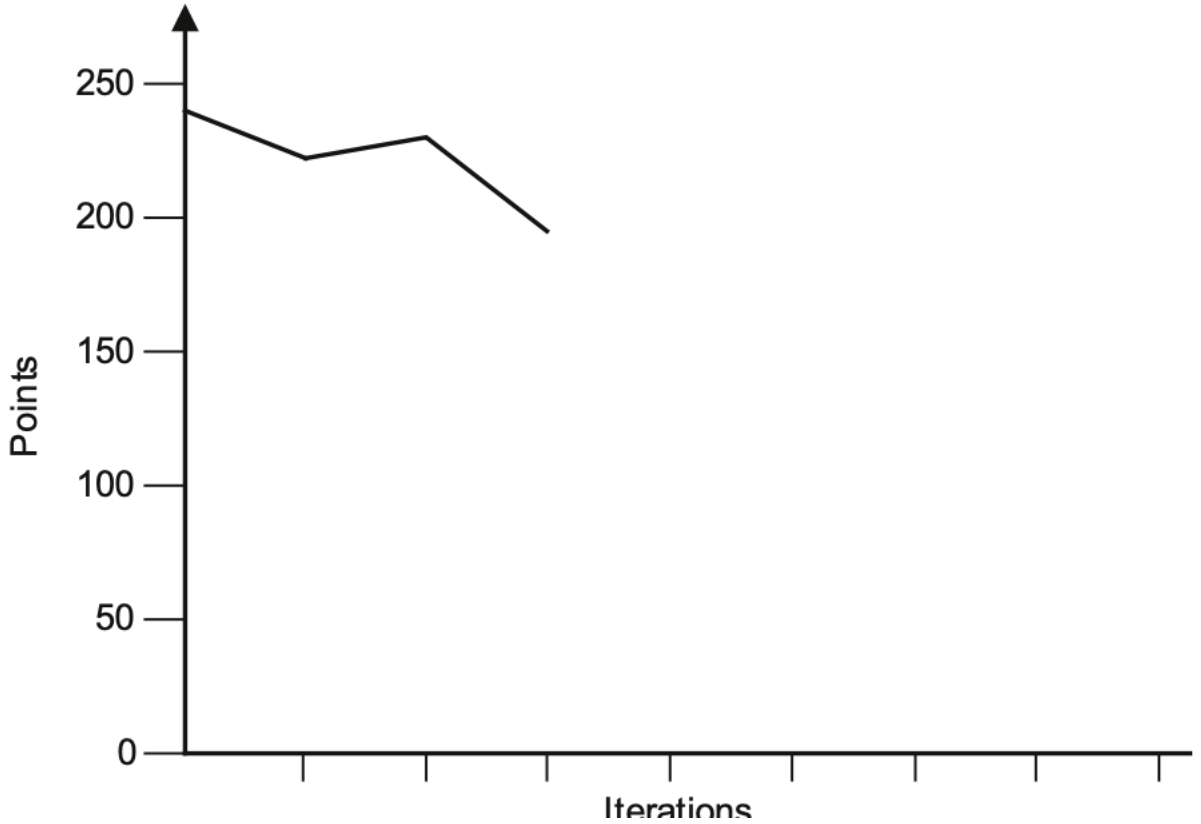
Relating the different planning levels



Tracking and communicating

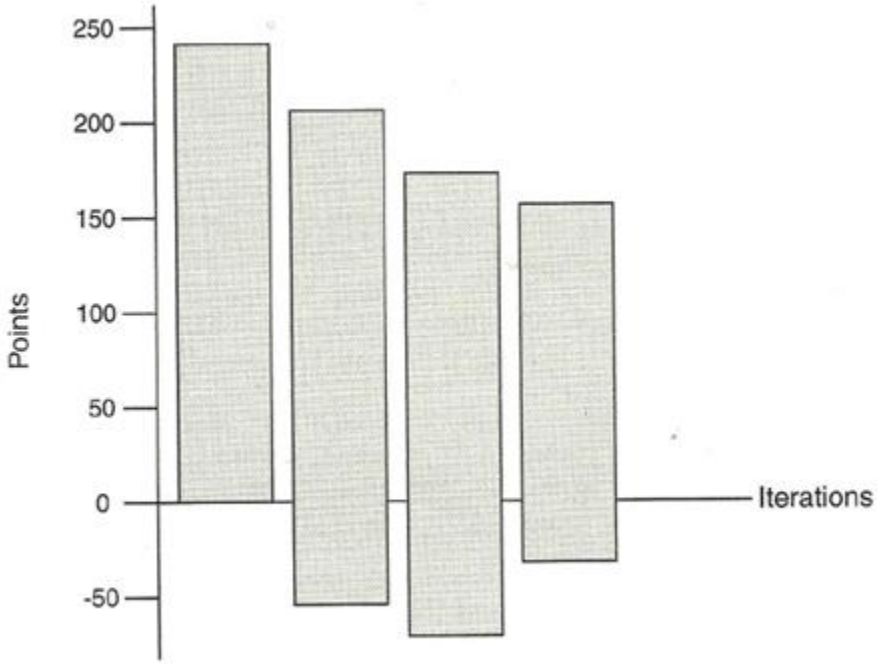
- Monitoring the release plan
- Monitoring the iteration plan
- Communicating about plans

Monitoring the release plan: A release burndown chart



Monitoring the release plan

Release Burndown Bar Chart

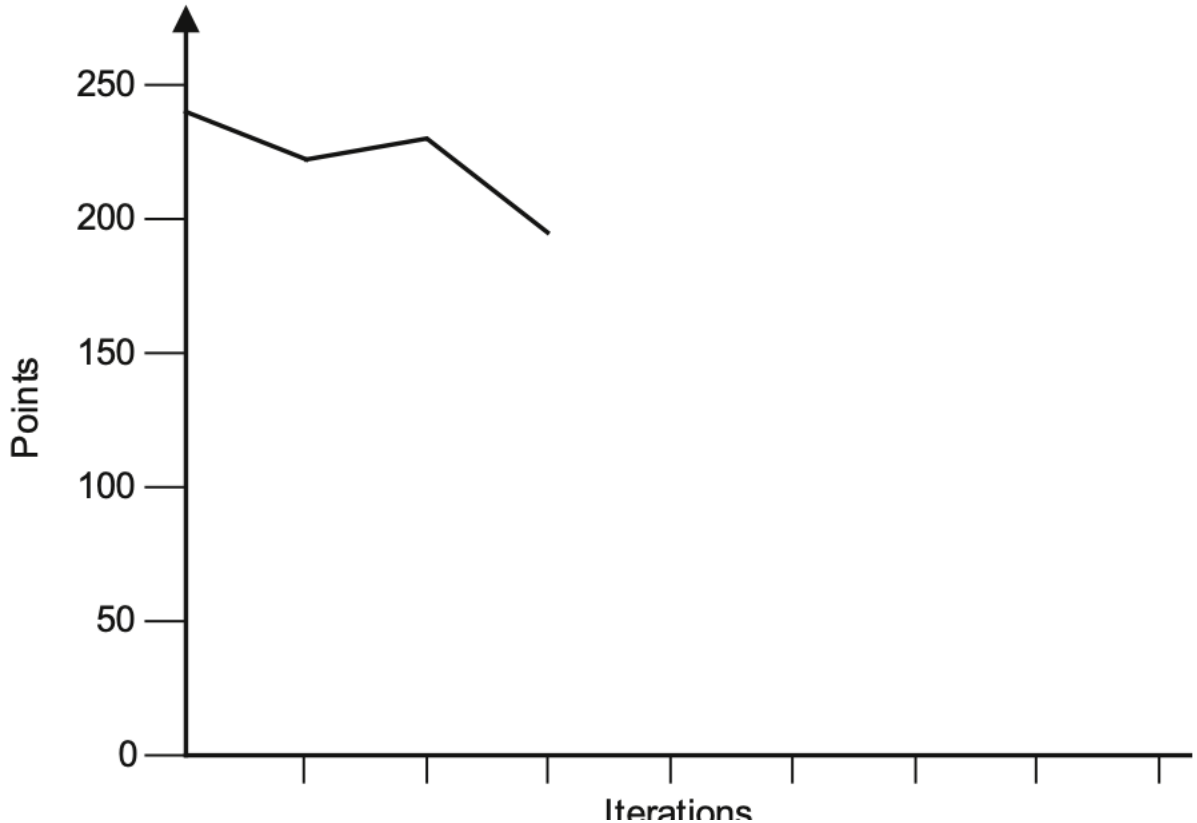


- Any time work is completed, the top is lowered
- When work is re-estimated, the top moves up or down
- When work is added, the bottom is lowered
- When work is removed, the bottom is raised

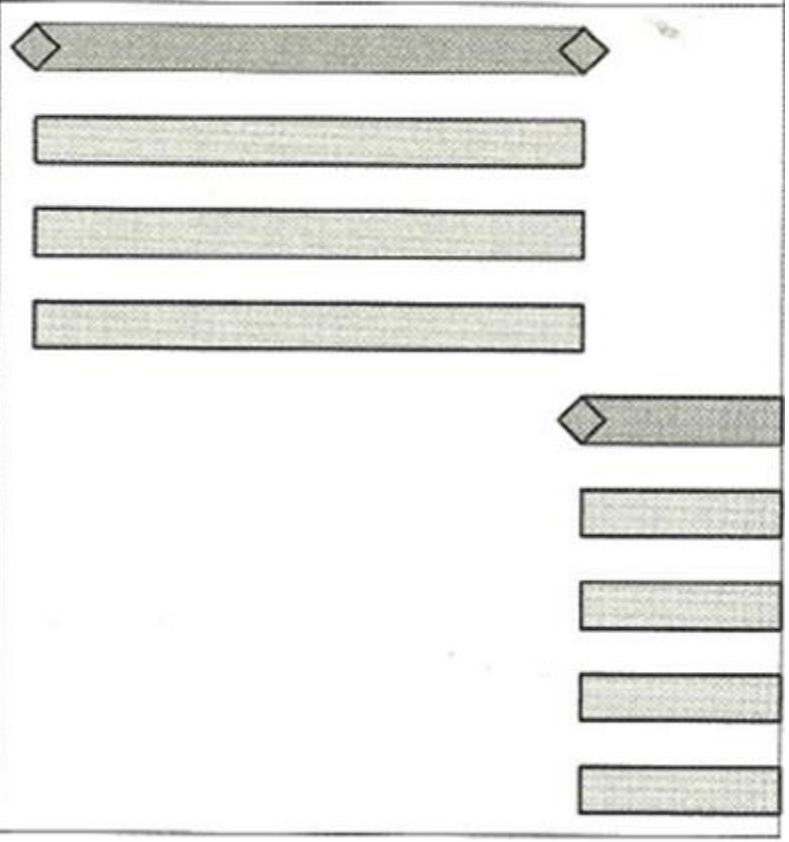
Monitoring the iteration plan

Story	To Do	Tests Ready	In Process	To Verify	Hours
As a user, I can... 5	Code the... 8 Code the... 5 Test the... 6	√	Code the... SC 6 Code the... DC 4	Code the... LC 4	33
As a user, I can... 2	Code the... 8 Code the... 5				13
As a user, I can... 3	Code the... 3 Code the... 6	√	Code the... MC 4		13

Monitoring the iteration plan: an iteration burndown chart

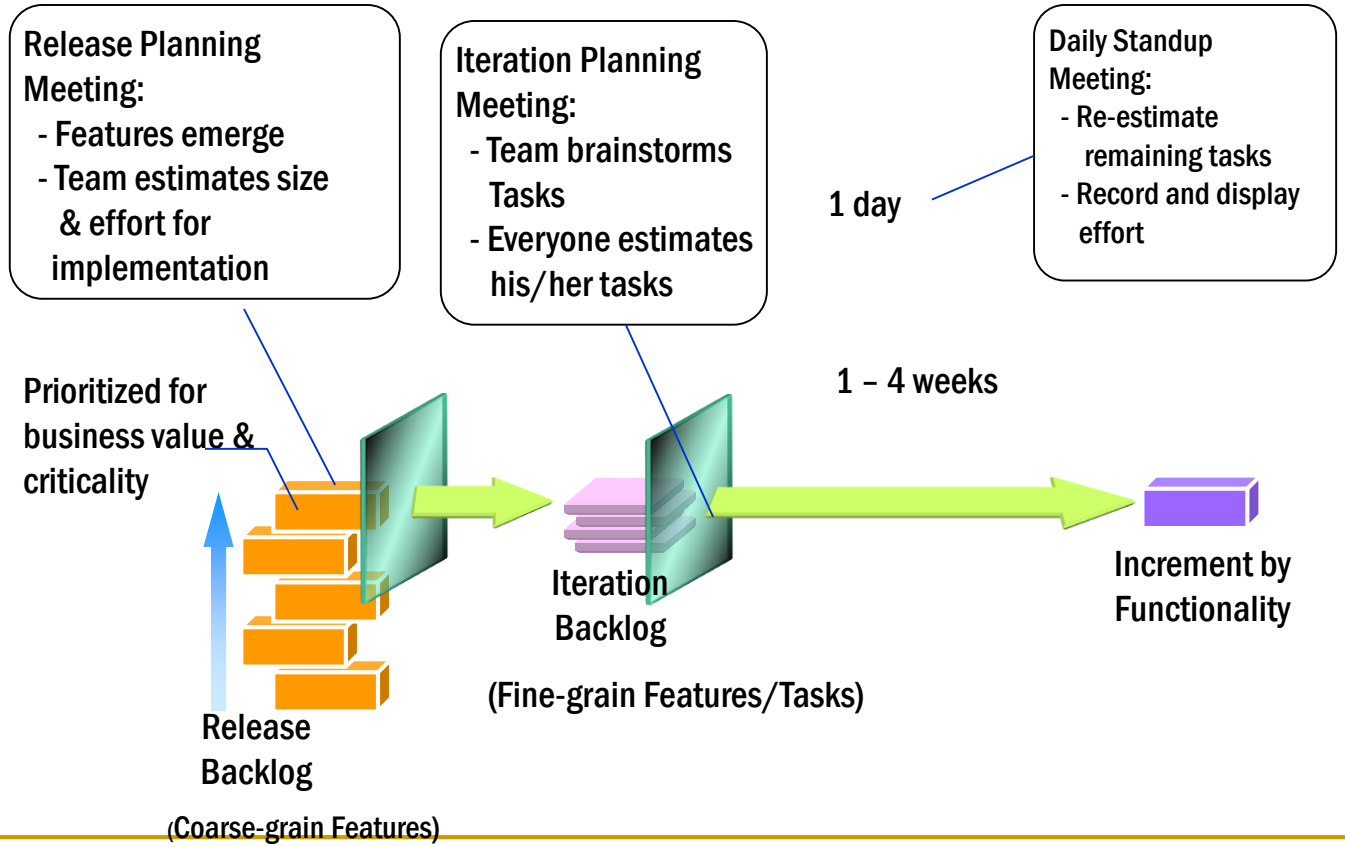


Communicating about plans

ID	Description	Start	End	Chart
1	Iteration 1	July 1	July 14	
2	As a user, I want ...	July 1	July 14	
3	As a user, I want ...	July 1	July 14	
4	As a user, I want ...	July 1	July 14	
5	Iteration 2	July 15	July 28	
6	As a user, I want ...	July 15	July 28	
7	As a user, I want ...	July 15	July 28	
8	As a user, I want ...	July 15	July 28	
9	As a user, I want ...	July 15	July 28	

Planning & Estimation in Agile

"Estimates will never be anything other than approximate, however hard you try."
- Beck & Fowler (2000)



Project Vision/Charter

What is needed next?
A detailed plan of action so that we can create a budget

The highest level document of a project

Content outline:

- Name of the project
- Purpose of the project and business case
- Project team, key business stakeholders, other team members
- Project approach (SDL, technologies chosen and why)
- Project goal date and high-level milestones
- Cost assumptions based on research

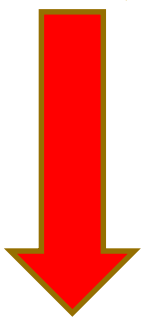
Research phase is done. Next step?

- Research phase outcome: Project charter or vision document (High-level plan)
- Next: Final Discovery
  - determine final business requirements (clear and detailed)
  - Accurate estimate of the budget based on the business requirements
- Outcome: A plan described as Scope of Work (detailed level plan)

From High-level to Detailed-level Planning: Plan for the Plan

High-level Plan

How? Plan for the Plan



Detailed-level Plan

When to Plan-for-the-Plan?

- The High-level goal is defined. Significant research is in place.
- A final list of business requirements does **NOT** exist

What are the Plan-for-the-Plan activities?

- Brainstorming with knowledgeable team members
- Getting it all down on paper
- Identifying further breakout sessions

The Plan for the Plan: overview of everything that a project plan must have

Example of a

Plan-for-the-Plan

- Study Pages 82-84, textbook

Elicitation of the Plan for the Plan: ask key questions

Examples:

- What are all the components and steps of this project?
- How is one part of the project depended on another?
- If you had to guess, when would each one need to be completed to make the goal date?
- What are the assumptions and constraints?
- What is the right level of detail of the business requirements?

Outcome of Final Discovery: A Plan answering the following questions:

1. What specifically are we going to do?
2. What features will the system have?
3. What features will be left out?
4. Who is doing what?
5. What are the dependencies we must consider?
6. In what order must the interrelated parts (identified in 5) go in because one thing must be done before another can start?
7. How does all of this fit into a timeline?
8. What are the risks?
9. What is the budget for all of this?

Controlling scope v.s. Blue Sky

- A must in software projects
  - Do something modest, get it right
  - Only then do more

Timeline management

- Bottom-up (given a list of requirements, estimate the schedule)
- Top-down (given a fixed deadline, estimate the amount of requirements)

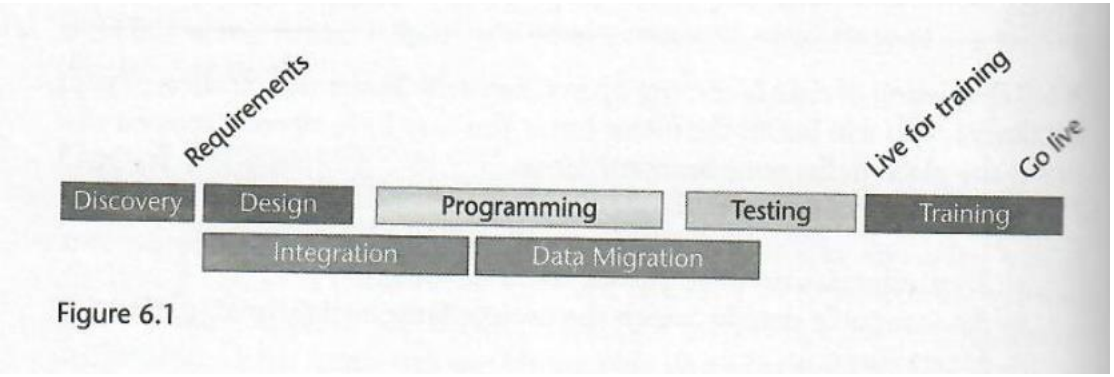


Figure 6.1

Project Management Metrics

- How the company will know the project is successful?
  - SUCCESS METRICS are derived from project goals through Goal-Question(Indicator)-Metric approach
  - Example of a goal: sales numbers are expected to increase 20% within 6 months
  - Success Indicator (graph to generate for the upper management): trend of sales over time.
  - Interpretation: the project is successful if an increase of at least 20% is observed within the indicated period.
  - Metrics (data to collect): number of sales reported monthly for the last 6 months

Valuable Advices and Best Practices on Scope

Valuable Advices and Best Practices:

- Visualize the scope (demos, mockups, etc.) – **budget for it!**
- Create mockups that are:
  - Highly visual
  - Broad in scope
- Budget in lots of time for mockups and demos.
- Target early-stage problems immediately.

The Scope Document overview

The Scope Document should include:

- Project summary
- Project deliverables
- Out of scope
- Constraints
- Assumptions
- Risks
- Timeline
- Budget
- Success Metrics (Key Performance Indicators (KPI))

Why Effort & Cost Estimation?

- Planning new development work (**schedule, effort, cost, people**)
- Estimating how many features you can deliver within a specific development iteration
- Estimating time needed for defect correction work (maintenance)
- Estimating number of developers
- …

Fundamental Principle 1 of Estimation

- All estimates are based on:
  - a set of assumptions that must be realized, and
  - a set of constraints that must be satisfied.

Assumptions and Constraints

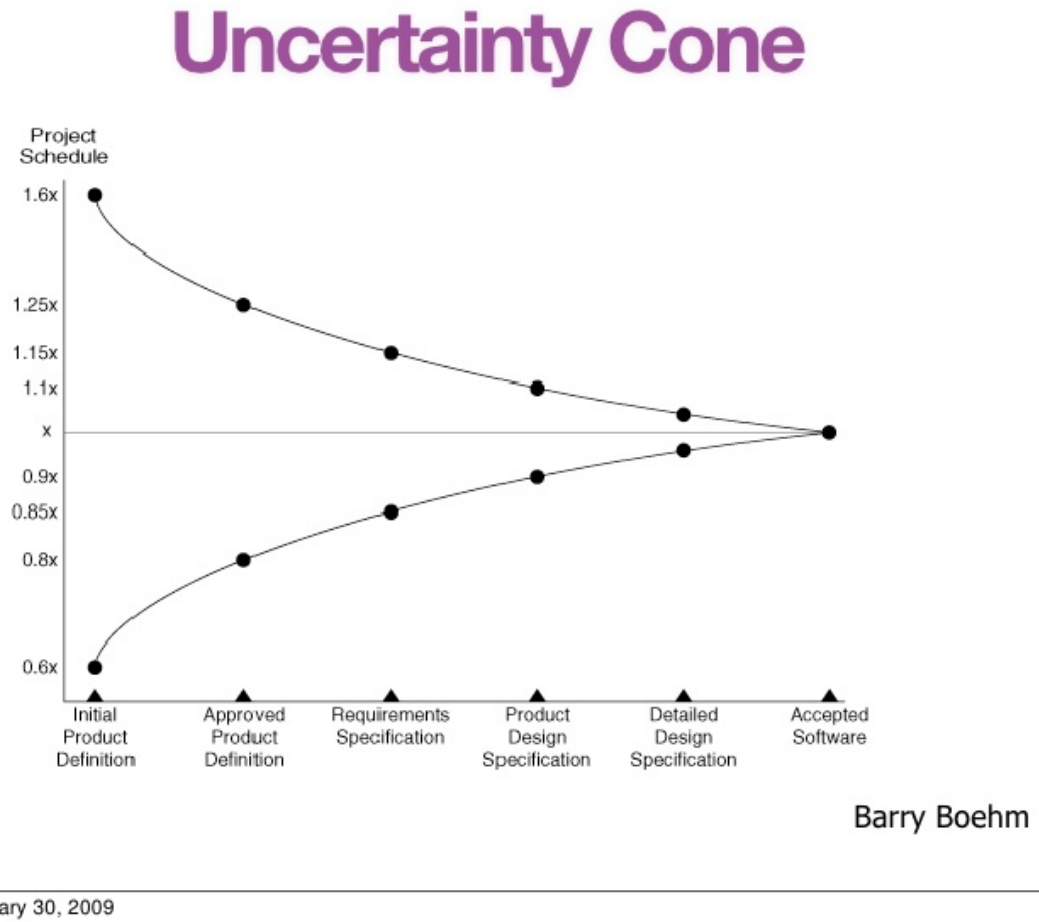
- An *assumption* is a statement that is taken to be true without verifying, or being able to verify, the truth of the statement
  - for example, it might be assumed that the productivity factor on the next project will be 500 delivered source lines of code per staff-month (500 DSLOC/SM)
- A *constraint* is an externally imposed condition that must be observed
  - for example, the project might be constrained to 5 people for 6 months

Fundamental Principle 2 of Estimation

- Projects must be re-estimated periodically as understanding grows and aperiodically as project parameters change.

Notion of "Cone of uncertainty" in software project effort estimation

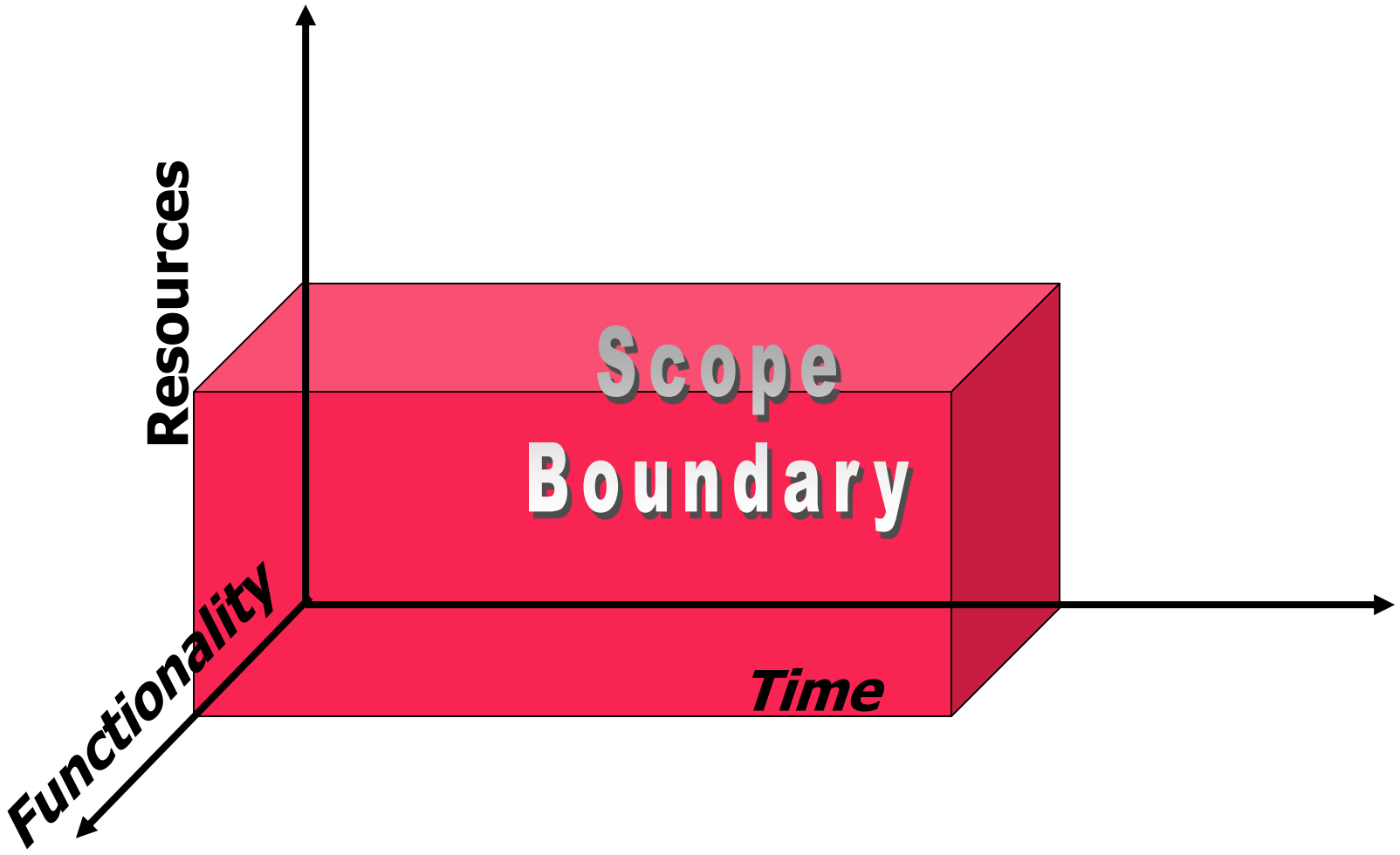
- At the beginning of a project, comparatively little is known about the product or work results, and so estimates are subject to large uncertainty.
- As more research and development is done, more information is learned about the project, and the uncertainty then tends to decrease, reaching 0% when all residual risk has been terminated or transferred.



Art v.s. Science of Software Estimation

- **Science:**
  - Complex formulas, requires experience
  - Requires estimation tools for calibration and estimation
  - Goal: accuracy to achieve +- **15**% (RE: O.I.Q. regulations)
- **Art:**
  - Rules of thumb, simple formulas, 5th grade arithmetic at most
  - Accuracy: +- 25%

Scope Management : Dimensions



Scope Management : Time, Resources, Functionality

Scoping: define the boundaries of the project

- **Time is "Soft" Boundary**
  - Time (raw) estimation
  - Subject to change
- **Resources**
  - People (Remember Brook's Law!)
- **Functionality**
  - Features

Scope assessment: Very Simplified Equation

- Let
- **R:** resources available for project **(# of developers**)
- **T:** development time (**hours**) available to work on project
- **Effort(f) :** effort **(person-hours)** required to realize feature *f*.

Effort: "Productive" Time Only (2)

- **Industrial statistics**: In a typical 40-hour workweek, developers spend 20-25 hours a week in meetings, consulting on other projects, assisting marketing, or performing, etc. – tasks they generally MUST do.
  - Average task hours/week is **12-15** person-hours.
  - Few teams are able to achieve an average of 20 or more staff hours per week per developer without affecting the **quality** of the work

Effort: "Productive" Time Only

The daily recording of only the "productive" effort, (including overtime), expended by a person on project related tasks.

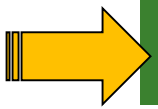
- Example:
  - a person works on a specific project from 8am until 5pm with 1 hour lunch break will record 8 hours of WORK EFFORT.
  - when the "non-productive" tasks have been removed, (coffee, liase with other teams, team meetings, administration, read magazine, etc.), only 5 hours of **PRODUCTIVE EFFORT** may be recorded.

How to use effort in planning? Examples

- Example 1: *A person working full time on a small program for 1 day:*
  - **Duration of the activity = 1 workday**
  - **Productive Effort = 5 person-hours**
- Example 2: *Five persons working full time on a feature for six weeks in a project, 25 hours of work per week. Effort estimate?*
  - **Duration = 6 weeks**
  - **Productive Effort = (5 persons * 6 weeks * 25 hours/week) = 750 person-hours**

Project Scope Management

- How to assess project scope?
  - Use **Ratio (%)** as an indicator



$$\frac{\sum_{f \in Feature} effort(f)}{R \times T}$$

- Project scope Ratio **<=** 100% is acceptable:

$$\sum_{f \in Fetures} effort(f) \leq R \times T$$

Overscoping: Very Simplified Equation

- Let
- **R:** resources available for project **(# of developers**)
- **T:** development time (**hours**) available to work on project
- **Effort(f) :** effort **(person-hours)** required to realize feature *f*.
- A project is in jeopardy (**overscoped**) if

$$\sum_{f \in Feature} effort(f) \geq R \times T$$

Fixing Overscoping By Adding More People? Brook's Law

- Just add more resources (people)?! Brooks' law (Mythical Man-Month):
  - Adding labor to a late project will make it later.
  - Why? More people means …
    - Ramp-up time (means new and exiting people unproductive).
    - Communication, coordination, … overheads.

Project Scope Management

- Project scope **>>** 100% is a **problem.**
  - What can we do in such a situation?
- Solution: **REDUCE SCOPE**
  - Define a feature subset, ***B*** (release/project **baseline**) such that

$$\sum_{f \in B} effort(f) \leq R \times T$$

Requirements Baseline

Primary technique to scope management is to *establish a high-level requirements baseline* for the project (release).

- Definition:

  The itemized set of features intended to be delivered in a specific version of the application

- Must be agreed upon by primary stakeholders, especially customer and developer representatives.

Well-Estimated Projects

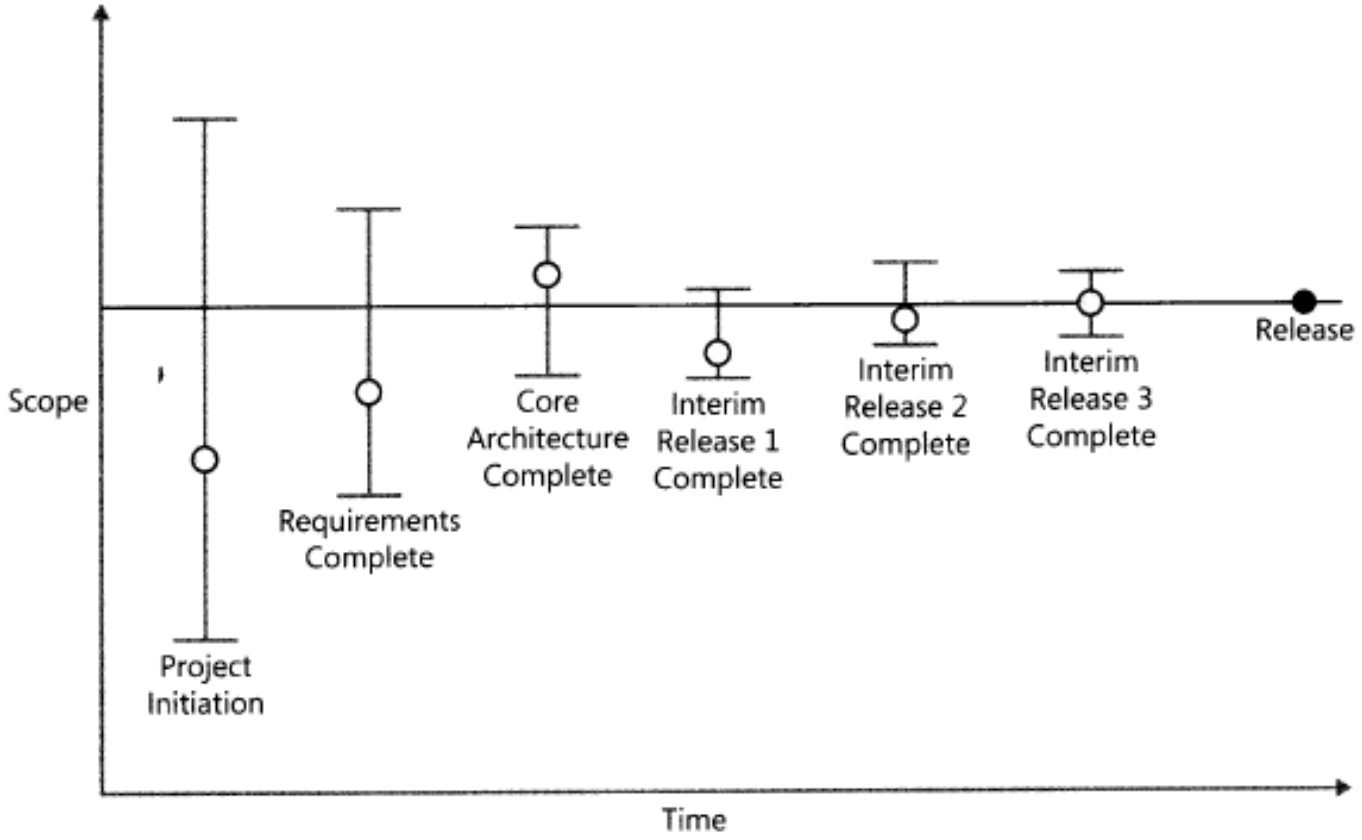


Figure 16-5 A well-estimated project. The single-point estimates miss the mark, but the ranges all include the eventual outcome.

THE PROJECT MANAGER

Has the primary responsibility and accountability for project estimating!

Budgeting

Budgeting Types:

- Comparative
- Bottom-up
- Top-down

COMPARATIVE BUDGETING: Expert Judgment / Delphi Estimation Method

1. A coordinator gives each expert the information on which to base the estimate
2. Experts work alone and submit their estimates and their rationales to the coordinator
3. The coordinator prepares an anonymous report that contains the estimates and rationales of each estimator and gives the report to each estimator and asks each to submit a second estimate
4. The procedure continues until the estimates stabilize
   - usually after 3 or 4 rounds
   - If there is small disparity in the stabilized estimates, they can be used as the range of estimates
   - If there is wide disparity in the stabilized estimates, the estimators meet to discuss and resolve their disagreements

Expert Judgment / Delphi Estimation

- To derive the task-level estimates, have the people who will actually do the work create the estimates
- Separate large tasks into smaller tasks
- Create both Best Case and Worst Case estimates to simulate thinking about the full range of possible outcomes.
- ExpectedCase =

  [BestCase + (4* MostLikelyCase) + WorstCase] / 6.

Example of Expert Judgement

Estimated Days to Complete				
Feature	**Best Case**	**Most Likely Case**	**Worst Case**	**Expected Case**
Feature 1	1.25	1.5	2.0	**1.54**
Feature 2	1.5	1.75	2.5	**1.83**
Feature 3	2.0	2.25	3.0	**2.33**
Feature 4	0.75	1	2.0	**1.13**
Feature 5	0.5	0.75	1.25	**0.79**
Feature 6	0.25	0.5	0.5	**0.46**
Feature 7	1.5	2	2.5	**2.00**
Feature 8	1.0	1.25	1.5	**1.25**
Feature 9	0.5	0.75	1.0	**0.75**
Feature 10	1.25	1.5	2.0	**1.54**
TOTAL	**10.5**	**13.25**	**18.25**	**13.62**

Why Best Case and Worst Case estimates ?

Table 9-1 Example of Developer Single-Point Estimates

Feature	Estimated Days to Complete
Feature 1	1.5
Feature 2	1.5
Feature 3	2.0
Feature 4	0.5
Feature 5	0.5
Feature 6	0.25
Feature 7	2.0
Feature 8	1.0
Feature 9	0.75
Feature 10	1.25
TOTAL	**11.25**

Table 9-2 Example of Individual Estimation Using Best Case and Worst Case

	Estimated Days to Complete	
Feature	**Best Case**	**Worst Case**
Feature 1	1.25	2.0
Feature 2	1.5	2.5
Feature 3	2.0	3.0
Feature 4	0.75	2.0
Feature 5	0.5	1.25
Feature 6	0.25	0.5
Feature 7	1.5	2.5
Feature 8	1.0	1.5
Feature 9	0.5	1.0
Feature 10	1.25	2.0
TOTAL	10.5[1]	18.25

Single estimate: 11.25 v. s. Range [10.5 … 18.25]

Note: Single is usually very closed to "Best"

Team Effort Estimate for a Project

- Have each team member estimate pieces of the project individually, and then meet to compare estimates
- Discuss the differences among the individual results
- Average your estimates
- Arrive at a consensus estimate ***E*** that the whole group accepts
- Calculate a rang
  - At the beginni approximately within +- 25%
  - [min, max]: ***[E * 75%, E * 125%]***
  - When the project advances the range is within +- 15% [min, max]: ***[E * 85%, E * 115%]***

Why the effort range is wider at the beginning of the project?

Bottom-up Budget

- Estimated budget for a project is calculated approximately as

$$Cost = \sum_{f \in B} effort(f) \times salary(+overhead)$$

where

- *B* is the baseline of the requirements (features)
- *f* is a feature (requirement)
- **Effort(f) :** effort **(in person-hours, or stuff-hours, person-weeks, etc.)** required to realize feature *f*.
- Overhead: hosting costs, setup fees, licensing fees, QA, training, etc.

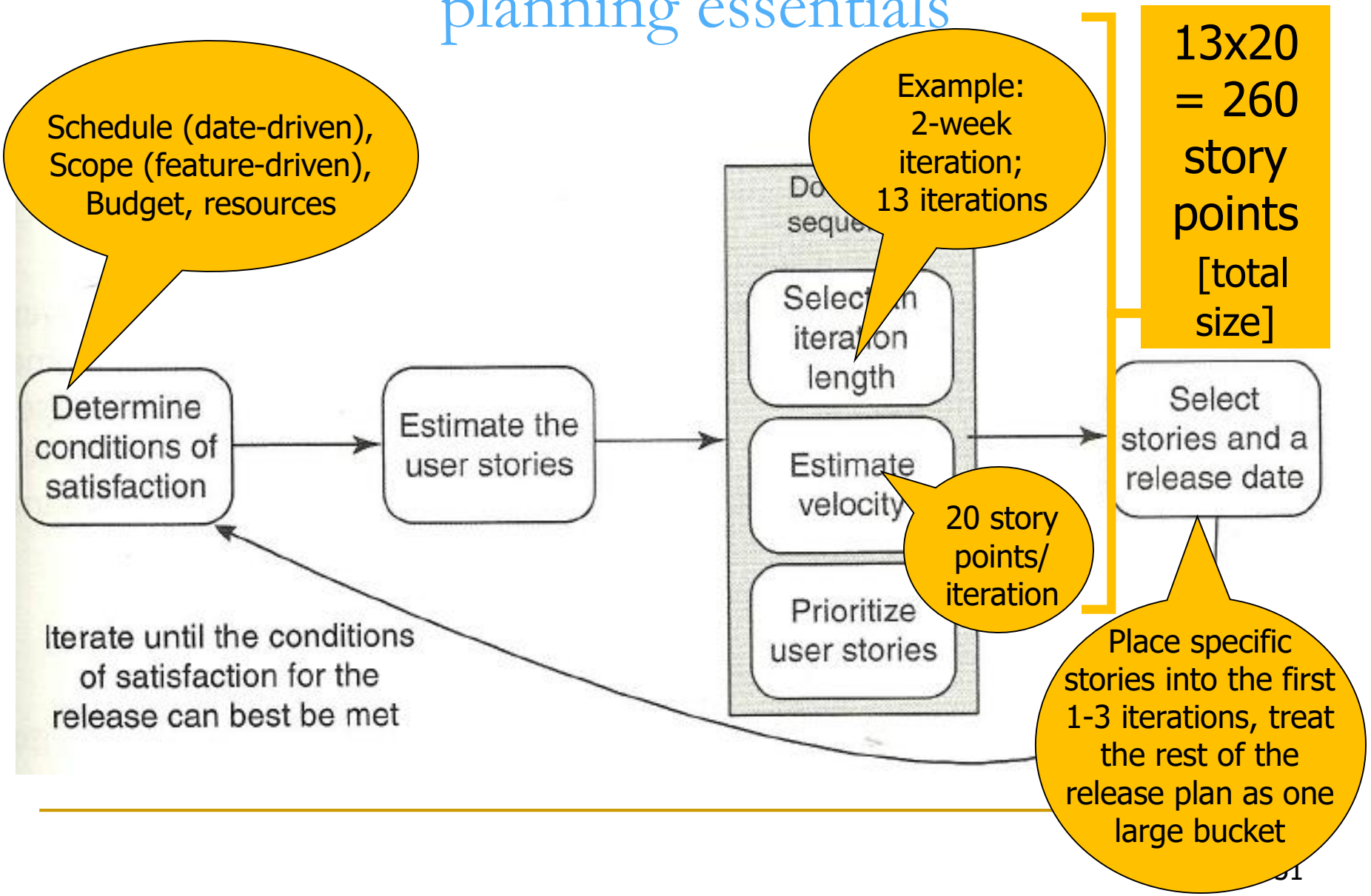
Top-down budget

- There is no list of item to work on
- Large chunks of the project scope are estimated in terms of weeks or months
- Where the list of requirements is clarified, comparative budgeting starts one by one (Delphi method)

Effective comparative Estimating matching "almost Agile" development process:

1. Establish a Base
   - Programming **60%**
   - Ongoing discovery **15%**
   - Unit testing and developing **10%**
   - Project management **15%**
2. Assign **percentages** to each item (experience)

Agile (re) estimating revisited: Release planning essentials



Summary

- Estimating and planning in agile
- Project scope and documentation
- Budgeting. Accurate Estimating.