Hugbúnaðarverkefni 2 / Software Project 2

3. Software Estimation

HBV601G – Spring 2019

Matthias Book

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ
IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz Prep

- Please prepare a small scrap of paper with the following information:

ID: ____@hi.is Date: _____

1. a) ____ b) ____

2. a) ____ - ____ b) ____ - ____
   c) ____ - ____ d) ____ - ____
   e) ____ - ____

- During class, I'll show you questions that you can answer with a few numbers
  - No calculation or elaboration necessary
- Hand in your scrap at end of class
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
  - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam

In-Class Quiz #1 Solution

- **Indicate if the following are (E)pics, (F)eatures, (N)eeds, (R)equirements or (T)asks:**

a) As a customer, I can let the shop store my credit card information so I can pay more conveniently for subsequent purchases. (R)

b) Define acceptance test (T)

c) In addition to physical store, sell goods also in an online shop. (E)

d) Payment by credit card (F)

e) Payment in the online shop needs to be effortless, so customers have no reason to change their mind and bail out of a purchase at the last second (N)

f) Cancel an order (F)

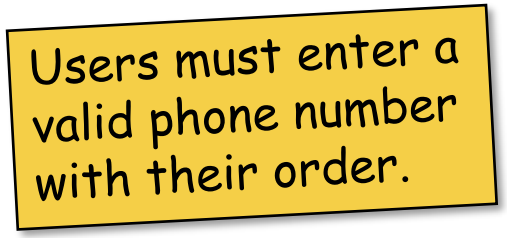
Software Estimation

see also:

- McConnell: Software Estimation, Ch. 1, 3-5, 7-13
- Leffingwell: Agile Software Requirements, Ch. 8

Accurate Estimates Are Based on Precise Requirements

A seemingly trivial requirement:



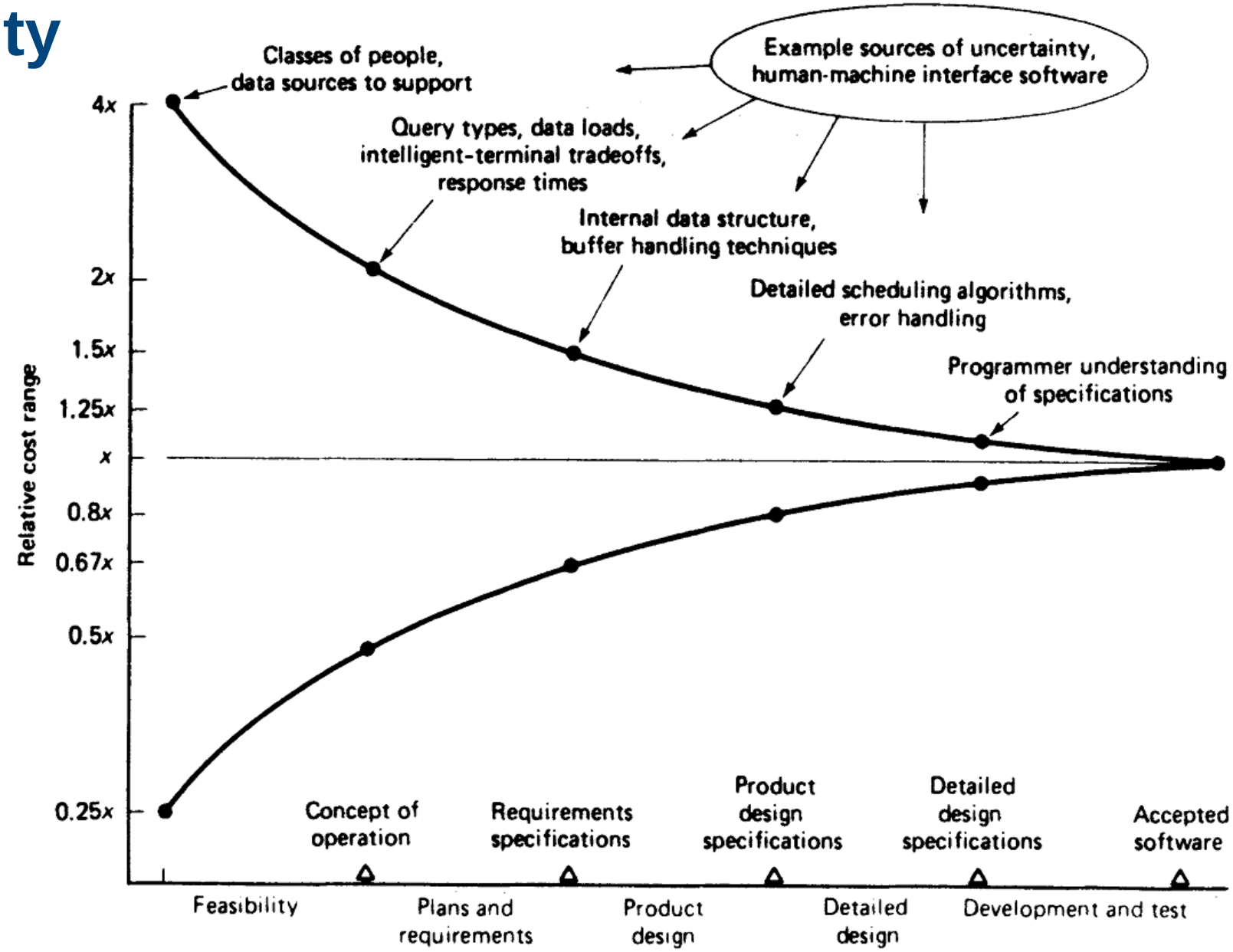
But we need to understand:

- This is not just about *entering*, but about *validating* a number using some “phone number checker (PNC)”
- If the customer wants a PNC, will he want the version we can build in 2 hours, 2 days, or 2 weeks?
- If we implement the cheap version of the PNC, will the customer later want the expensive version after all?
- Can we use an off-the-shelf PNC, or are there design constraints that require us to develop our own?
- Do the PNC and the Address Checker need to interact? How long will it take to integrate them?
- How will the PNC be designed?
- How long will it take to code the PNC?
- What will the quality level of the PNC be?
- How long will it take to debug and correct mistakes made in the PNC implementation?

Each of these can introduce a factor 10 of difference in complexity, quality, time etc.

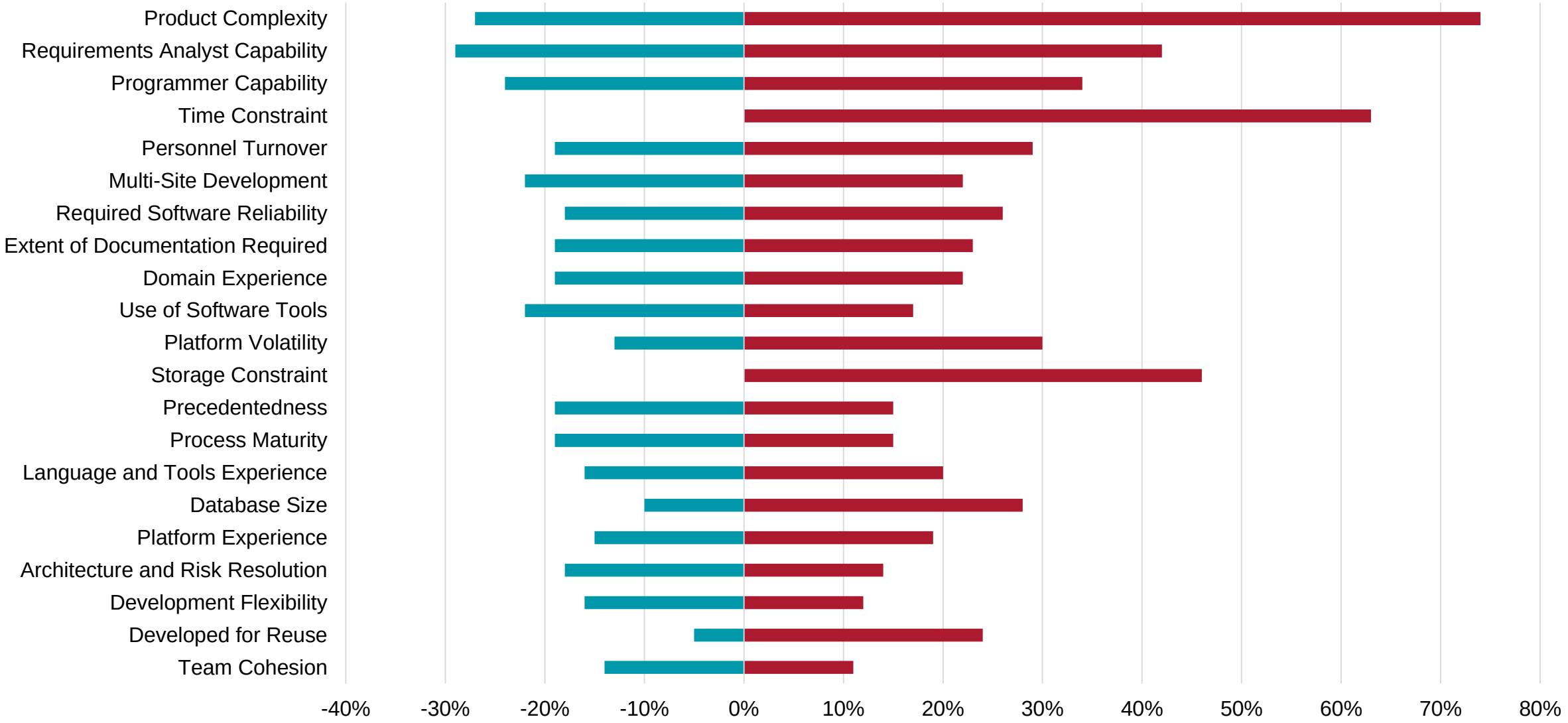
Cone of Uncertainty

- A measure for the maximum possible estimation accuracy over the course of a project
  - Your cone may be wider
  - i.e. your accuracy may be worse!
- The cone does not narrow by itself
  - Continuous active work is required to eliminate sources of uncertainty in the project.



Barry Boehm: Software Engineering Economics. IEEE Transactions on Software Engineering, Vol. SE-10 (1), pp. 4-21 (1984)

Impact Bandwidth of Factors Affecting Project Effort



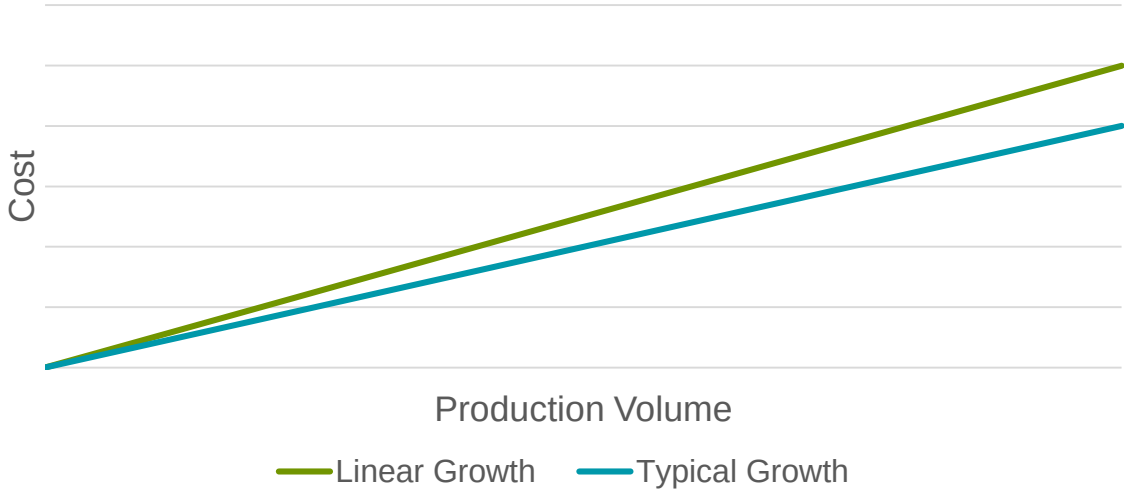
Barry Boehm et al.: Software Cost Estimation with Cocomo II. Addison-Wesley, 2000

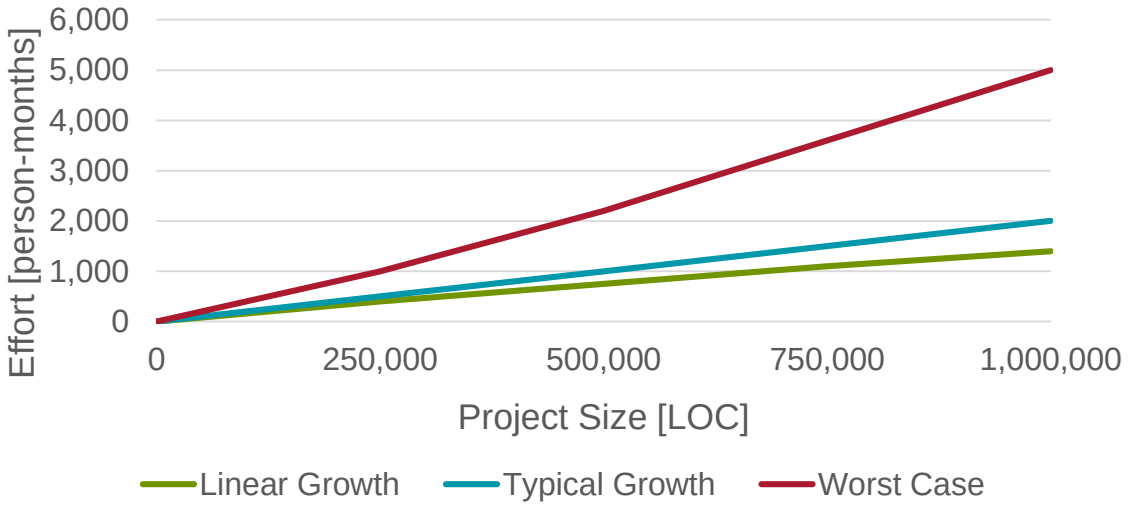
Diseconomies of Scale

- Many manufacturing domains can benefit from **economies of scale:** "The more units we produce, the smaller the cost of each unit."

- In software engineering, we have to deal with a **diseconomy of scale:** "The larger a system we build, the higher the cost of building each part."

- Effort does not scale linearly with project size, but exponentially.

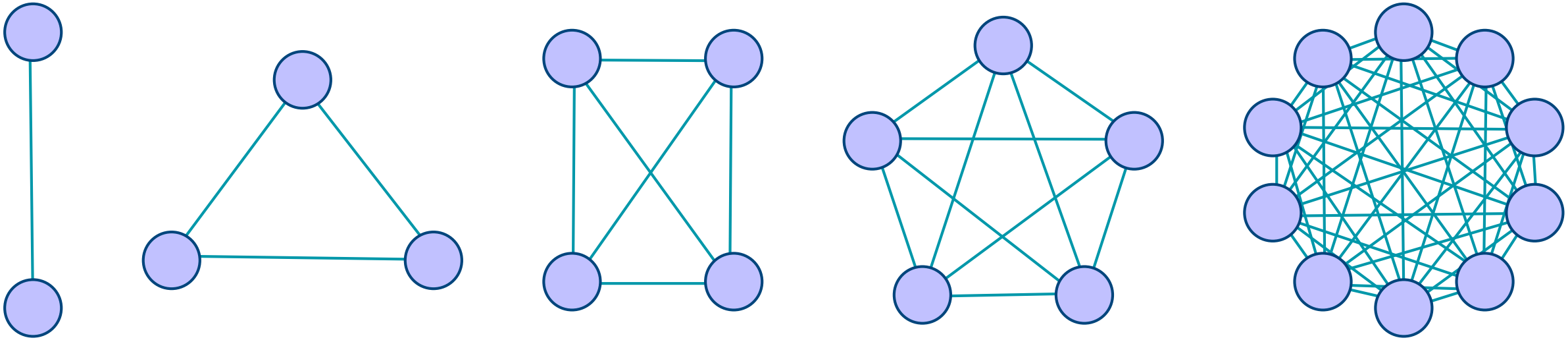




Barry Boehm et al.: Software Cost Estimation with Cocomo II. Addison-Wesley, 2000 (source for bottom chart)

Diseconomies of Scale

- A system that is 10 times as large requires *more* than 10 times as much effort.
  - A linear increase in components (or a linear increase in team size) results in an exponential increase in **relationships** between components (or people).
  - Of course, efficient component design and encapsulation (or effective team structures) can reduce the number of actual relationships considerably,
  - but the cognitive complexity that the team has to deal with remains high.

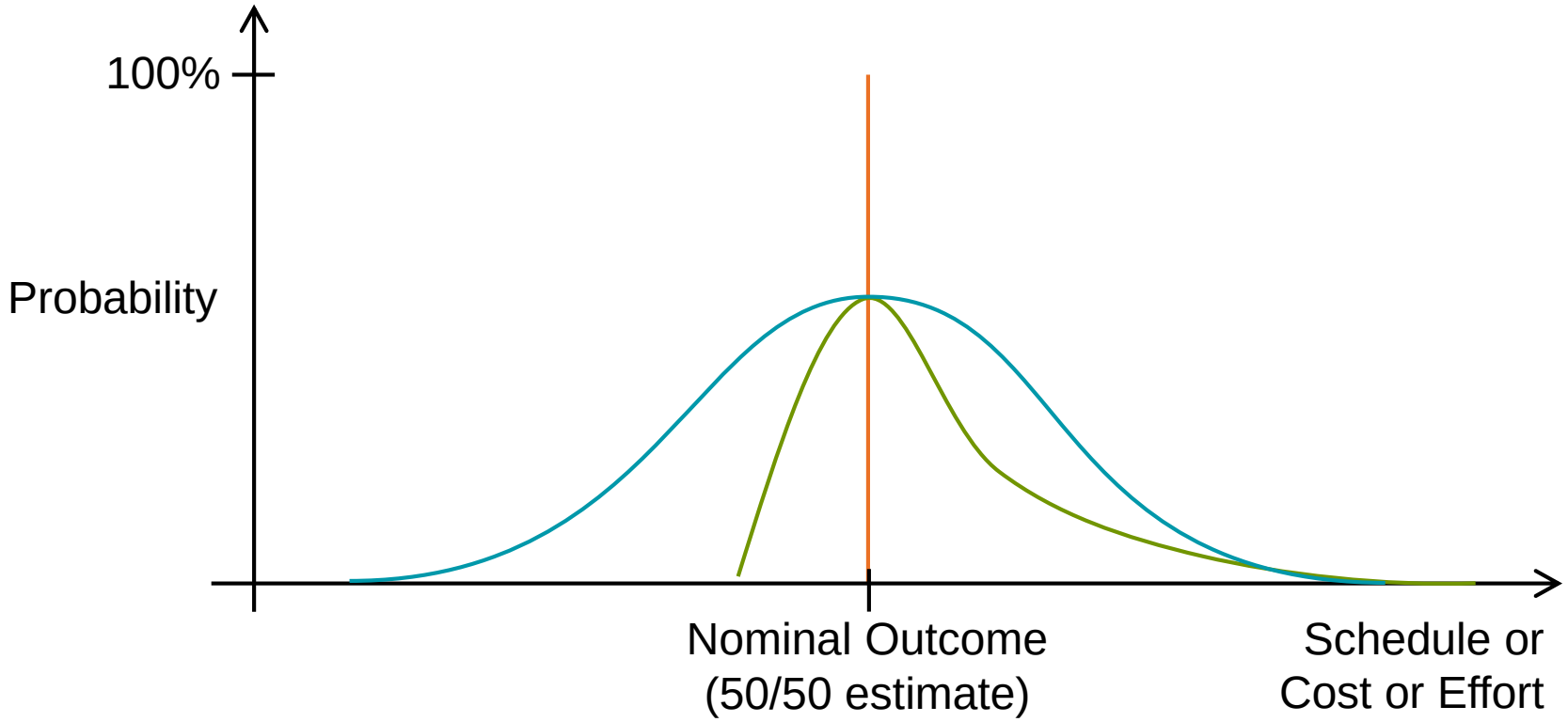


Difficulties in Software Estimation

- **Caution when interpreting management requests**
  - Determine whether you are expected to estimate, or figure out how to hit a target
- **Caution when preparing estimates**
  - Hard to quantify the subject
  - Lots of confounding factors
  - Temptation to tweak the estimate
- **Caution when letting management interpret your estimates**
  - Your estimates may be understood as precise commitments

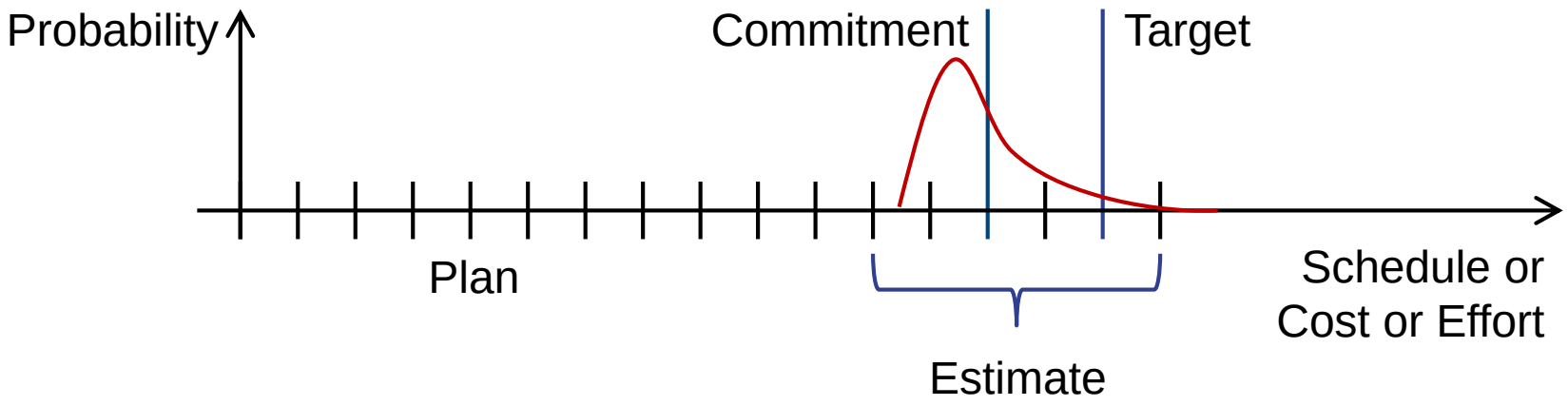
Estimates as Probability Statements

- ~~Single point estimate: Assumes that estimate will accurately predict result~~
- ~~Bell curve: Disregards limits of how efficiently a project team can complete work~~
- ✓Realistic curve: There's a limit to how *well* a project can go, but not how *bad*



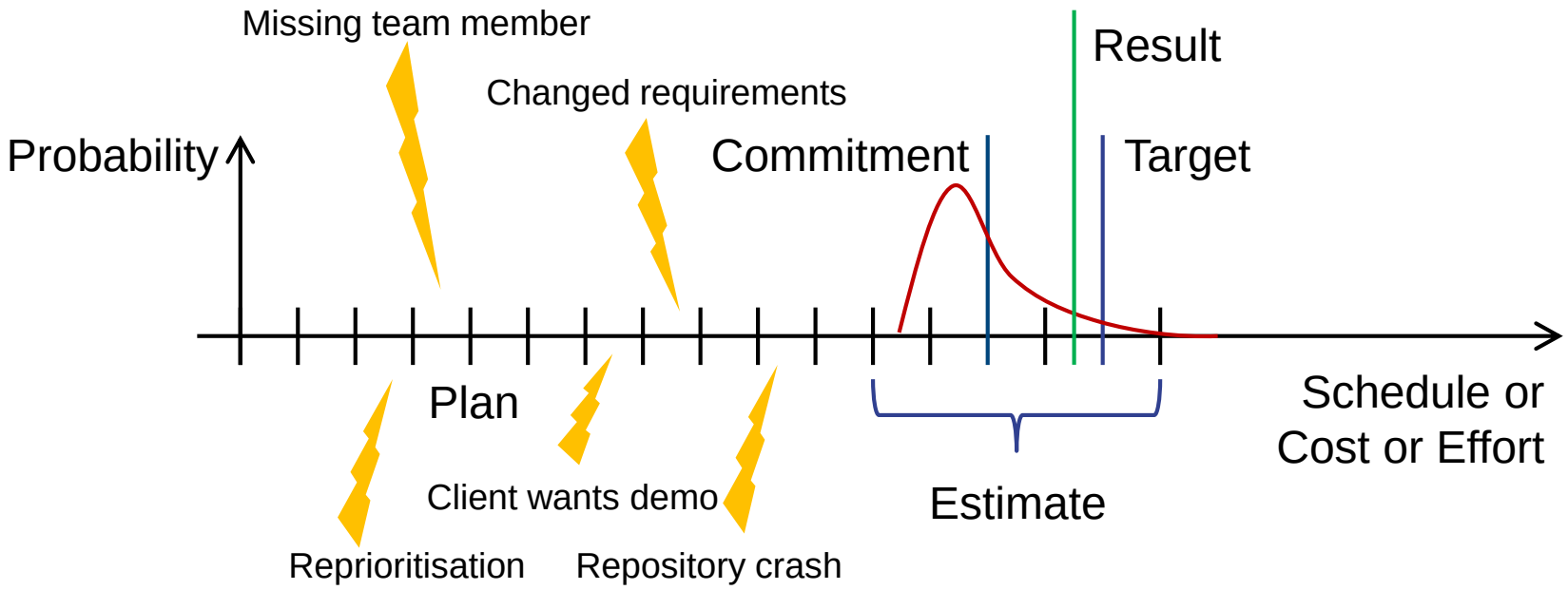
Terminology

- **Target:** fixed value (deadline, budget etc.) determined by external factors
- **Estimate:** preliminary prediction of a future value/interval
- **Commitment:** Intention of reaching a certain future value
- **Plan:** Agreement of steps to achieve the commitment



Life Is What Happens While You Are Making Other Plans

- Some project events make (the foundations of) prior estimates obsolete.



The Purpose of Software Estimation

- The aim of effort estimates is not to predict the project result, but to **judge whether the project target is sufficiently realistic** to be **achievable through corresponding project management.**
  - Estimates do not need to be extremely precise, as long as they are **useful**.
- Experience: If the initial target and the initial estimate are within ~20% of each other, the project manager will have enough maneuvering room (in terms of features, schedule, staffing etc.) to meet the project's business goals.
- If the gap between target and estimate is larger, it is very unlikely the project can be completed successfully.
  - ➢ Bring target into better alignment with reality before the project has a chance of success.
  - **RESIST TEMPTATION / PRESSURE** to align estimate better with target!
    - Your first estimate is rough but honest.
    - Revisions not based on additional data are just kidding yourself and endangering your project.

Estimation Approaches Discussed in This Class

A. Projection from counts
   1. Concrete items
   2. Function points

B. Individual expert judgment

C. Group expert judgment
   1. Wideband Delphi
   2. Planning Poker

D. Qualitative estimation

A_1) Projection from Counts

1. Find a suitable type of item to count
   - Highly correlated with size of software you're estimating
   - Available early in development cycle
   - Statistically meaningful number of items available (>20)
     - If we are talking about less, the individual differences between these items become too dominant
   - Countable with minimal effort
   - Historical averages based on comparable parameters(!) available
     - Similar product, similar team, similar technology, similar complexity…
2. Count items of interest
3. Calculate effort from item counts, based on average effort per item

Countable Items for Projecting Estimates

- Marketing requirements
- Engineering requirements
- Features
- Use cases
- User stories
- Function points
- Change requests
- Web pages
- Reports
- Dialog boxes

- Database tables
- Classes
- Defects found
- Configuration settings
- Lines of code already written
- Test cases already written
- …

- **Projection:** Multiply counted items with average effort per item observed in historical projects of similar kind

In-Class Quiz #2a: Projecting Estimates from Counts

1.a) Counting defects late in project

- For the last 250 defects we fixed, we needed 2 hours per defect on average.
- 400 defects are currently open in our project.

- How many hours will it approximately take us to fix the open defects?

1.b) Counting web pages

- So far, each dynamic web page in the project cost us a total of 40 hours to design, code and test, on average.
- We have 10 web pages left to build.

- How many hours will it approximately take us to finish this?

A_2) Function Points (FP)

- Synthetic measure of software size assigned to counts of various types of items:
- **External inputs**
  - Screens, forms, dialog boxes, control signals through which an end user or other system adds, deletes, or changes our system's data
- **External outputs**
  - Screens, reports, graphs, control signals that our system generates for use by an end user or other system.
- **Internal logical files**
  - Major logical groups of end-user data or control information completely controlled by our system, e.g. a single flat file or a single table in a database
- **External interface files**
  - Files controlled by other systems with which our system interacts; this includes each major logical group of data or control information entering or leaving our system

Function Points

- Each system characteristic is classified by complexity (low / medium / high) and assigned an according number of function points:

System Characteristic	Low Complexity	Medium Complexity	High Complexity
External inputs	3	4	6
External outputs	4	5	7
Internal logical files	4	10	15
External interface files	5	7	10

- FPs are an abstract, relative measure of complexity, not absolute size/time/effort
- Formulas exist to translate FPs into lines of code, based on industry experience
  - e.g. in Java, 1 FP requires at least 40, typically 55, at most 80 lines of code
- More useful: Use organizational experience to translate FP to expected effort

Recommendation: Avoid in practice – use story points if you want an abstract metric

Lines of Code (LOC)

Traditional size measure found in many tools and textbooks

Advantages

- Data easily collected and evaluated by tools
- Lots of historical data exists
- Effort per line of code is roughly constant across languages
  - (*Effect* per line of code obviously is not!)
- Most convertible “currency” between projects
  - But make sure projects are comparable to draw meaningful conclusions!

Disadvantages

- LOC don’t reflect
  - Diseconomies of scale in software development effort
  - Difference in programmer productivity
  - Difference in expressiveness of programming languages
- Using LOC for estimating non-coding work (requirements, design etc.) is counterintuitive
- Which Lines of Code to count?

Recommendation: Avoid in practice – mostly useful for academic purposes

In-Class Quiz #2b / Experiment: Estimation Confidence

2. Estimate a range in which you are 90% confident the actual value will be:

a) Latitude of San Francisco:
- _____ – _____ °N

b) Birth year of Fridtjof Nansen:
- _____ – _____

c) Area of Iceland:
- _____ – _____ km^2

d) Production cost of movie “Avatar”:
- _____ – _____ million US$

e) Registered students at HÍ:
- _____ – _____

Quiz inspired by “Software Estimation” by Steve McConnell (Microsoft Press, 2006). The original is © 2006 Steve McConnell.

Experiment: Estimation Confidence

How many of the following actual values fall inside your estimation range?

a) Latitude of San Francisco:
- 37°47’ N

b) Birth year of Fridtjof Nansen:
- 1861

c) Area of Iceland:
- 103.000 km^2

d) Production cost of movie “Avatar”:
- 237 million US$

e) Registered students at HÍ:
- 12.470 (as of 20 Oct 2018)

In range	Percentage
0 of 5	0%
1 of 5	20%
2 of 5	40%
3 of 5	60%
4 of 5	80%
5 of 5	100%

Quiz inspired by “Software Estimation” by Steve McConnell (Microsoft Press, 2006). The original is © 2006 Steve McConnell.

Lesson: Estimation Accuracy

- Remember you were asked to provide a range that you were *90% confident* to include the actual value.

- Reflect:
  - Did you subconsciously try to make the estimate “better” by tightening the range?
  - Were you still highly confident that your range included the actual value?

- Perceived estimation confidence is usually much higher than actual confidence
  - Be cautious with statements like “90% confidence” – usually your estimate is far less reliable
  - Choose your estimation range deliberately
  - Don’t make the range narrower than necessary
  - The estimation range should reflect your inconfidence

Estimation vs. Commitment

- In software projects, people (and possibly yourself) will assume that your estimates can be directly translated into commitments.

Estimation vs. Commitment

- Your estimate is much less reliable than you think, but other people even think it's much more reliable!
  - Carefully manage expectations and be clear about your confidence

…9 months later:

B) Individual Expert Judgment

- Let an expert (typically: developer) come up with an estimate for each feature
  - Caution: subjective, possibly based on non-transferable experience / incorrect assumptions
- Avoid developer single-point estimates
  - Usually, these reflect the developer's subconscious *best case* assumption!
- Create three-point estimates: Best Case, Most Likely Case, Worst Case
  - This will stimulate developers to think about full range of possible outcomes for each feature
- Calculate expected case from three-point estimates for each feature:
  - ExpectedCase = (BestCase + 4 * MostLikelyCase + WorstCase) / 6 (basic PERT formula)
  - ExpectedCase = (BestCase + 3 * MostLikelyCase + 2 * WorstCase) / 6
    - (alternative formula if you feel your team's "most likely" estimates tend to be too optimistic)

Example: Calculating Expected Cases

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

[Days to Complete]

Checklist for Individual Estimates

- Is what's being estimated clearly defined?
- Does the estimate include all the kinds of work needed to complete the task?
- Does the estimate include all the functionality areas needed to complete the task?
- Is the estimate broken down into enough detail to expose hidden work?
- Did you look at documented facts from past work or estimate purely from memory?
- Is the estimate approved by the person who will actually do the work?
- Is the assumed productivity similar to what has been achieved on similar assignments?
- Does the estimate include a Best Case, Worst Case, and Most Likely Case?
- Is the Worst Case really the worst case? Does it need to be made even worse?
- Is the Expected Case computed appropriately from other cases?
- Are you aware of the assumptions that are underlying this estimate?
- Has the situation changed since the estimate was prepared?

Training Your Estimation Accuracy Over Time

To improve estimation accuracy:

- Compare actual results to estimated results
  - Calculate Magnitude of Relative Error (MRE) for each feature estimate:
    MRE = | (ActualResult – EstimatedResult) / ActualResult |
    - The average of your MREs per sprint should decrease over time
  - Check if actual results were in range between best case and worst case
    - The percentage of actual results that are in this range should increase over time
  - Check if 50% of expected cases were overrun and 50% of them were underrun
    - If not, your individual estimates tend to be too optimistic or too pessimistic
      - Work on improving your best, most likely and worst case estimation accuracy
- Try to understand
  - What went right and what went wrong
  - What you overlooked
  - How to avoid making those mistakes in the future

Example: Calculating Magnitude of Relative Error

Feature	Best Case	Most Likely Case	Worst Case	Expected Case	Actual Outcome	MRE	Between best and worst case
1	1.25	1.5	2.0	1.54	2	23%	Yes
2	1.5	1.75	2.5	1.83	2.5	27%	Yes
3	2.0	2.25	3.0	2.33	1.25	87%	No
4	0.75	1	2.0	1.13	1.5	25%	Yes
5	0.5	0.75	1.25	0.79	1	21%	Yes
6	0.25	0.5	0.5	0.46	0.5	8%	Yes
7	1.5	2	2.5	2.00	3	33%	No
8	1.0	1.25	1.5	1.25	1.5	17%	Yes
9	0.5	0.75	1.0	0.75	1	25%	Yes
10	1.25	1.5	2.0	1.54	2	23%	Yes
Total				**13.62**	**16.25**	**29% avg.**	**80% yes**

[Days to Complete]

Assignment 1: Project Plan and Requirements

Note: Updated from last week's version.

- By **Sun 3 Feb**, submit a **project outline** in Ugla, containing:
  - Project vision (in form of a RUP Vision document, Data Sheet or Press Release)
  - Product backlog (prioritized user stories)
  - User Story estimates (three-point estimates of expected cases, using basic PERT formula)
  - Project schedule (dates for sprints, milestones, assignments 2 and 3)

- On **Thu 7 Feb**, present and **explain** your project outline to your tutor:
  - What considerations influenced your choice of scope, estimates and schedule?

- **Grading criteria:**
  - Project vision is clear and plausible (15% of this assignment's grade)
  - Product backlog describes requirements clearly and with realistic scope (40%)
  - Project estimates calculated using appropriate formulae (35%)
  - Project schedule is clear, realistic, and specifies dates for assignments 2 and 3 (10%)