# Software Project Management

## Unit 2: Measurement, estimation & data analysis

Thais Webber
Richard Lee

# Unit objectives

**When you have completed this unit, you will be able to:**

- Appreciate the key role of measurement, estimation and data analysis in software project management
- Understand measurement theory and use appropriate scale types to quantify the attributes of software projects
- Understand the need for estimates and use estimates to judge software project characteristics including size, effort and cost

# Outline

- Role of measurement, estimation and data analysis
- Key measurement concepts and measurement theory
- Theory and practice of estimation and prediction

# What is estimation?

- **Estimation** is the process of predicting an expected value for some attribute (whose value is not known)

- The **uncertainty** is expressed in terms of probabilities

- **Estimates** are forecasts or predictions; we estimate what we cannot measure
  - the entity whose attribute we estimate may not exist yet or is not sufficiently mature to be measured accurately
  - not because we don't understand what we are trying to measure

# What is estimation?

- Estimation is closely linked to measurement because estimates are based (explicitly or implicitly) on data obtained from measurement
- Representational validity and scale types are still just as relevant
- The most common use of estimation in software projects is **resource prediction**

# Resource prediction

- When needed?
  - determining if a project proposal is feasible resource-wise
  - bidding for a contract
  - planning a project: predicting cost and duration, etc.
  - tracking resource expenditure during a project

**If estimates are poor, the resulting losses can take a company out of business**

# Modelling uncertainty

- In principle, measurements are 'exact'
  - predictions are uncertain
- An estimate of an attribute A lies within some overall range of possible values on the scale X to which the (actual) value of A belongs
  - different values on X will have different 'likelihoods' of corresponding to the actual value of A
- **Estimates and probability are closely linked**
  - **Reminder**: a probability is formally expressed as a value in the interval [0.0, 1.0], though you will also see it as a percentage (i.e., in the interval [0.0, 100.0])

# Estimates and probability

- **Continuous variables**
  - Example: the duration of a software project, the weight of a server
  - If we estimate that attribute A will have the specific value a, then a is termed a ‘point estimate’, and the probability of our estimate being precisely correct is zero
  - If we estimate that A will have a value that lies in a non-empty interval , then the probability of being correct can be non-zero
- **Discrete variables**
  - Example: number of people, SLOC
  - It is still likely that there is uncertainty - each of the possible values $a_1$ , $a_2$ , …, $a_n$ may have a non-zero probability

# Confidence intervals

- **[L, U]** denotes an **interval**; all values **v** such that **L <= v <= U**
  - L is the **lower bound**
  - U is the **upper bound**
- The probability **P** of the actual value **a** (of attribute **A**) lying in the interval **[L, U]** is written as:

$$P(a \in [L,U])$$

- If P(project_duration_in_months ∊ [8, 12]) = 0.95, then [8, 12] is a 95% confidence interval

# Interval notation for estimates

- (L, M, U) is a triple spanning the interval [L,U] where M, the **central value** of the triple, is the **expected value** of the estimate

- If M in (L, M, U) is the **mid-point** of the interval [L, U] (it does not have to be), then:

$$M - L = U - M = \delta$$

and (L, M, U) = (M - δ, M, M + δ), which is more usually written:

$$M \pm \delta$$

**Example**: the estimate for the number of defects in a software application is 20 ± 3 per thousand SLOC

# Improved estimates

- Ways to improve on an expert judgement:
  - **Group estimation (the Delphi method)**
  - **Task decomposition**
  - **Analogy**
  - **Formal modelling**

# The Delphi method



- **Iterative prediction technique involving a group of experts**
  - Each expert provides an estimate in each iteration
  - The estimates from all experts are “summarised” (e.g., averaged) and the result is communicated back to the experts...
  - ... and used to come up with a new set of expert estimates in the next iteration

- Estimates obtained by a group of experts are more accurate than those provided by a single expert

# Estimation by decomposition

- **Decompose**
  - a project's software process into phases
  - a project phase into 'activities' or 'work packages'
  - a system into separate subsystems or 'functional units'
- **Top-down**: Estimate an overall value and then proportionately distribute this over the components of the distribution.
  - When is this appropriate? When the project completion time is fixed
- **Bottom-up**: Estimate a value for individual components, and then obtain the overall figure (e.g. by summing them).
  - When is this appropriate? e.g., for an open-ended or research project

# Formulae



- **S size** - example units SLOC (but it could also be number of modules)
- **EpS** (effort per size) - example of units used: pm/SLOC
- **SpE** (size per effort) called **Productivity** - example of units: SLOC/pm
- **CpS** unit cost - example of units used: £/SLOC
- **CpE** unit cost rate - example of units used: £/pm
- **Effort = S x EpS** - example working units: SLOC x pm/SLOC → pm
- **Effort = S/SpE** working the units: (SLOC) / (SLOC/pm) → pm
- **Cost = S x CpS** example of units £ but it could also be € (EUROS) - example working the units: SLOC x £/SLOC → £
- **Cost = Effort x CpE** - example working the units: pm x £/pm → £

pm = person-month, but it could be person-week (for example)

# Worked example 1

- Attributes estimated: system size, development effort and cost
- Decomposition: system into ‘functional units’/ subsystems (USI, GRD, DBMS)
- Strategy: bottom-up with multiple conversion factors
- **Delphi average = (Min + 4 * Likely + Max) / 6**

# Worked example 2

- Attributes estimated: development effort and cost
- Two dimensional decomposition: system into subsystems; process into phases
- Strategy: bottom-up using expert judgement
- What will this project cost?

| Function | Requirement | Design | Code | Test | Total |
|---|---|---|---|---|---|
| USI | 1.0 | 2.0 | 0.5 | 3.5 | 7.0 |
| GRD | 1.5 | 11.0 | 4.0 | 10.5 | 27.0 |
| DBMS | 2.0 | 6.0 | 3.0 | 4.0 | 15.0 |
| **Total pm** | 4.5 | 19.0 | 7.5 | 18.0 | 49.0 |
| **CpE (£/pm)** | 5,200 | 4,800 | 4,250 | 4,500 | |

# Estimation by analogy

Aston University BIRMINGHAM UK

1. Look for a past project (or unit) **Q** that is 'very similar' to the new project **P**
2. Obtain recorded effort **value*** **E** for **Q**: **Q.E**
3. Identify orthogonal factors accounting for significant differences between **Q** and **P** & expected to impact **P**'s effort
4. Adjust **Q.E** to take into account factors identified in **(3)**:

$$P.E = Q.E * (Rf_1 * Rf_2 * \ldots) + (Af_1 + Af_2 + \ldots)$$

What might be typical ratio factors $\mathbf{Rf_i}$ or additive factors $\mathbf{Af_i}$?

- E.g., size or development rate for ratio factors; different components between P and Q for additive factors

* effort might be time, cost, development time, etc.

# Worked example

- We will develop a new graphics project G
- We have developed in the past a similar project called Q
- Suppose we know from project Q, that Q.E = 100pm
  - to develop Q, 100 person-months were needed

- Question: What is the effort that you predict for the new project G?
- One significant difference between G and Q is size
  - **Q.S = 25,000 SLOC**
  - **G.S ≈ 20,000 SLOC ± 2,500 SLOC**

# Finding analogous projects

Aston University BIRMINGHAM UK

- Need database of relevant past project data

- What is an ‘analogous project’?
  - Need attributes that can be measured at estimation time, that are reasonably independent and correlated with variable you are trying to predict
  - Analogue identification mostly by expert judgement

- Identify adjustment factor(s) and estimate them quantitatively for new project

# Summary of key points in this unit



- Measurement, estimation and data analysis are key software project management tools
  - they support reliable quantitative comparisons, evaluations and decision-making
- The values of software project/artefact attributes can be established
  - using measurement for existing entities
  - using estimation/prediction for entities that will exist in the future

# Tips for FYP this week

- Consider the Top-Down Decomposition of tasks
- Useful tools: Trello - https://trello.com/en & Gantt Chart
- SMART Goals / Objectives

# Unit objectives

Aston University BIRMINGHAM UK

**When you have completed this unit, you will be able to:**

- Appreciate the key role of measurement, estimation and data analysis in software project management
- Understand measurement theory and use appropriate scale types to quantify the attributes of software projects
- Understand the need for estimates and use estimates to judge software project characteristics including size, effort and cost

Aston University
BIRMINGHAM UK

# Software Project Management

## Unit 2: Measurement, estimation & data analysis

Thais Webber
Richard Lee