Aston University
BIRMINGHAM UK

Software Project Management

Unit 2: Measurement, estimation & data analysis

Thais Webber
Richard Lee

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

Outline

Aston University
BIRMINGHAM UK

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

Reminder: software process issue

- **Strategic**: organisational level (long term)
  - reducing costs
  - expanding customer base
  - increasing user satisfaction
  - improving productivity
  - reducing time-to-market
- **Tactical**: project level (medium term)
  - Quality, Time, Cost, Productivity
  - Process varies from project to project - compare:
    - RAD* of low-cost single-user desktop tool
    - enterprise-wide IS project
- **Operational**: activity level (short term)
  - Day-to-day, week-to-week activities which vary greatly depending on the project's process

Role of measurement, estimation & data analysis

Aston University
BIRMINGHAM UK

- To support reliable quantitative comparisons, evaluations, predictions and decision-making relevant to software projects and their artefacts

Project resource estimation	C, T
Project health tracking	Q, C, T, P
Productivity measures and models	P
Quality measures and models	Q
Structural and complexity measures	Q
Evaluation of models, methods and tools	Q, C, T, P

Key: C = cost; T = time; Q = quality; P = productivity

At operational level (short-term)

- collecting ‘raw’ data
- measuring estimating & predicting
- analysing data (e.g., statistically)
- developing & applying quantitative models
- interpreting (e.g., the outcomes of data analysis)
- experimenting
- evaluating

At tactical level (medium term)

Aston University
BIRMINGHAM UK

- Obtaining meaningful information that facilitates effective management of software projects
- Better PLANNING
- Improved UNDERSTANDING
- More effective CONTROL

At strategic level (longer-term)

Aston University
BIRMINGHAM UK

- Contribute to organisation-wide improvement of software processes, and hence to organisation's strategic aims
- Diagnose what is not so good
- Determine changes needed to improve
- Monitor the effects of applying those changes

Use as a management tool to answer questions

Aston University
BIRMINGHAM UK

- **How well are we doing currently?** (on this project, or projects in general)
- **This (such and such) has an atypical feature … Why is that?** (Can we make some positive use of this fact?)
- **Do (…X) and (…Y) exhibit some kind of consistent relationship?** (If so, what is it? Could we make good use of it?)
- **How do (...A) and (...B) compare? Is one 'better' than the other?** (If so, why?)
- **Why has (such and such) happened** (or keeps happening)?
- **If we make (such and such) a change, will (such and such) be the (hoped-for) outcome?**

What is measurement?

Aston University
BIRMINGHAM UK

- Measurement is the process of assigning a value to some attribute of an entity, where the value is obtained on a particular scale
  - e.g. elapsed time of the testing phase of a software package.
- Measurement may require a measurement instrument and may be carried out by a measurer

What to measure?

Aston University
BIRMINGHAM UK

- Product
  - Specifications, designs, test plans, modules, programs...
- Process
  - Change control, design, project scheduling, review...
- Resource
  - Personnel, money, hardware, software...
- The fact that we can name an attribute does not make it easy or even possible to measure:
  - Complexity, productivity, quality,
  - Length of time before the program stops (the halting problem)

Units

Aston University
BIRMINGHAM UK

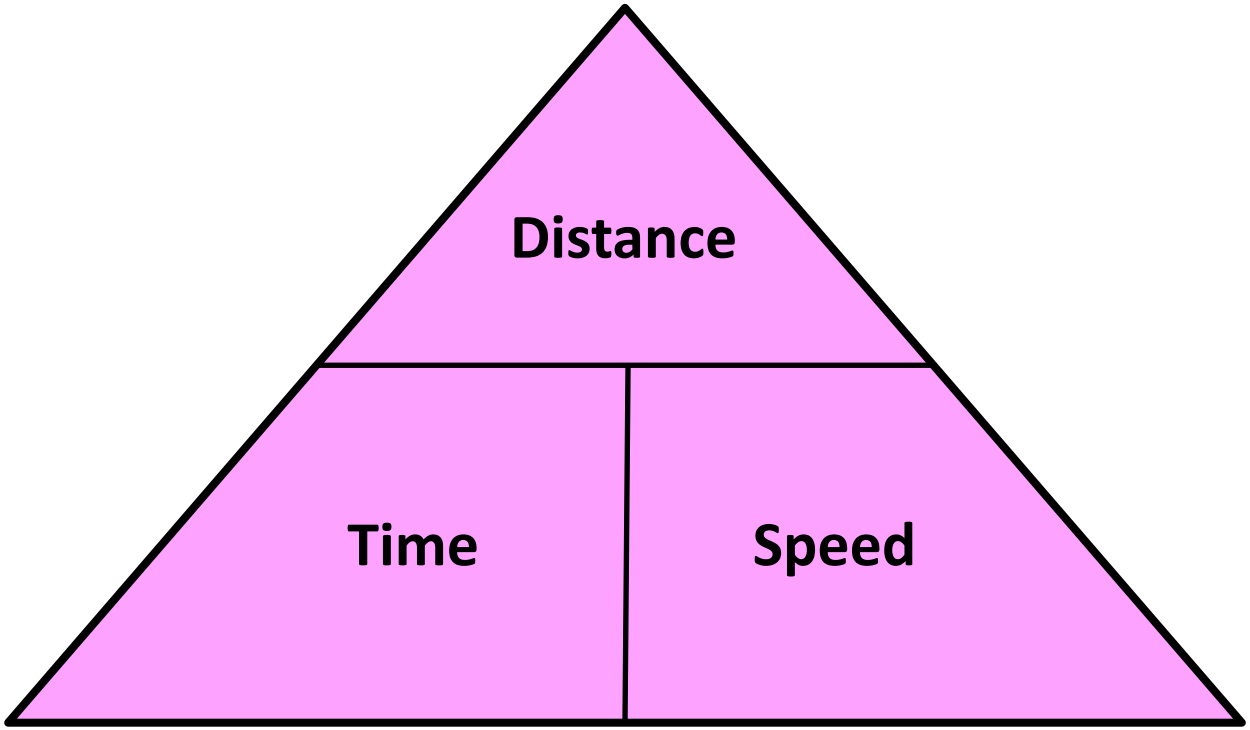
- A measurement unit is a scalar quantity, defined and adopted by convention, with which any other quantity of the same kind can be compared to express the ratio of the two quantities as a number.
  - For example, length can be measured in cm, inches, miles, nanometres...
  - Software size can be measured in characters, bytes, modules, classes, source lines of code (SLOC)

Example

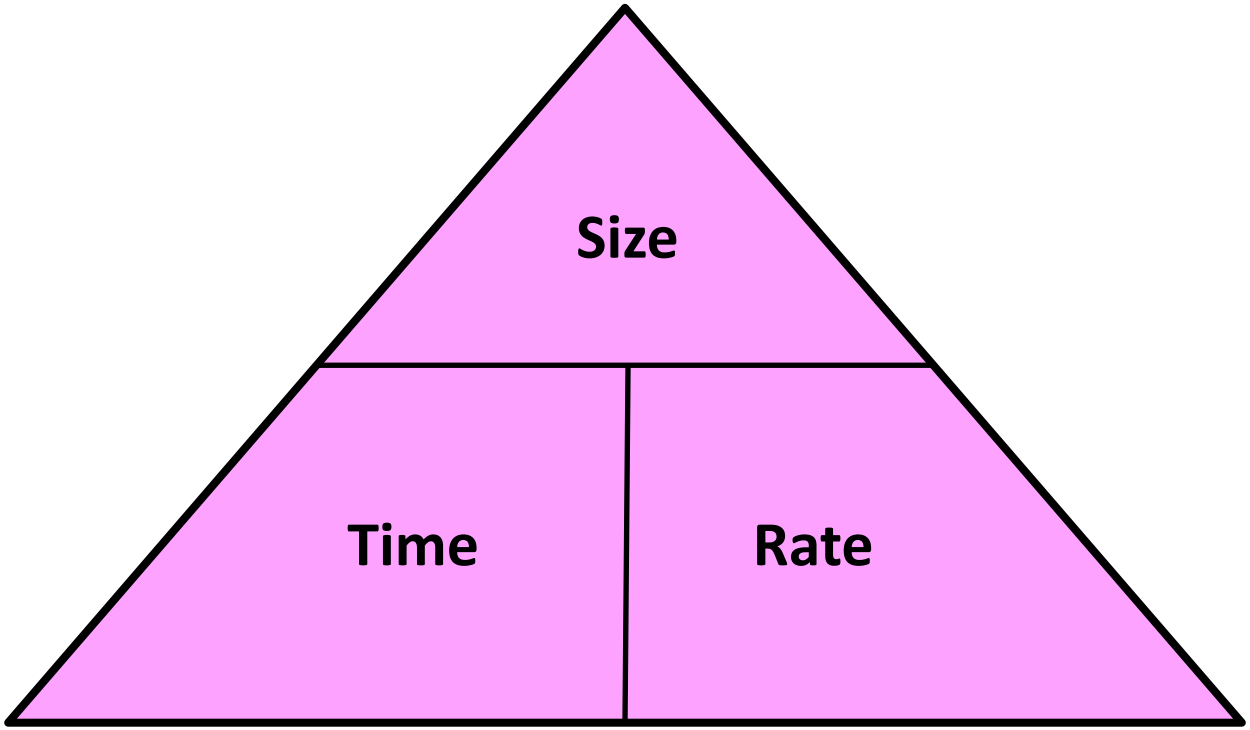
Aston University
BIRMINGHAM UK

- The size **S** of a Java package to be developed has been estimated at **10,000 SLOC**.
- The average rate **R** at which Java code is produced is **500 SLOC per developer-day**.
- The number of (full-time) developers **N** assigned to the task is **2**
- **How long will it take them to develop the package?**

Example



Example



Direct measurement

Aston University
BIRMINGHAM UK

- An attribute that can be measured without the need to measure another attribute:
  - Length of source code
  - Duration of testing process
  - Number of defects discovered

Indirect measurement

Aston University
BIRMINGHAM UK

- An indirect measurement is calculated from direct measurements, typically via addition, multiplication, division:
  - Length of a project (length of development + length of testing)
  - Developer-days (number of developers * number of days)
  - Programmer productivity (SLOC / time spent)
  - Module defect density (number of defects / module size)
  - System spoilage (effort spent fixing defects / total effort spent on project)

Measurement validity

- A measurement is valid if it is:
  - **accurate** – measures the true value of the attribute
  - **reliable** – repeated measurements give similar values
  - **meaningful** – the measurement answers a question that we want answered

Measurement accuracy

Aston University
BIRMINGHAM UK

- Accuracy is important, but needs to be tempered by context
- Acceptability of accuracy varies
  - If we're measuring file size, we expect 100% accuracy
  - If we're measuring number of defects, perhaps ± 10%
- Timeliness is more important
  - An approximate number of defects tells us whether the software is ready to ship
  - A precise number of defects doesn't add value, but may take time to determine (and can never be guaranteed as accurate)

Scale types

Aston University
BIRMINGHAM UK

- We need some way of understanding one measurement in relation to other measurements of the same type of attribute.

- The five types of scale that are available are as follows:
  - Nominal
  - Ordinal
  - Interval
  - Ratio
  - Absolute

- Data on a different scale conveys a different level of precision

Nominal

Aston University
BIRMINGHAM UK

- This means that the data can only be categorised
- When we compare two measurements, we can say "they are in the same category" or "they are not"
- For example:
  - types of defect (data-flow error, erroneous module interface, incomplete module specification)
  - process model (waterfall, prototyping, incremental)
  - operating system (Windows, MacOS, Linux)

Ordinal

Aston University
BIRMINGHAM UK

- We can categorise **and rank** data into order, but we don't have a any indication of the gap between two ranked items.
- For example:
  - Programmer skill (beginner, intermediate, expert)
  - Survey questions on quality (strongly disagree ... strongly agree)
  - Top-five paying clients
- We know, for example, that an intermediate programmer has greater skill than a beginner programmer, but we do not know how that gap compares with the intermediate-expert gap

Interval

Aston University
BIRMINGHAM UK

- We can categorise and rank data points
- We can infer equal intervals between them
- There is no 'zero point'
- For example:
  - Progress across time (**P1** = 23/09/2024, **P2** = 30/09/2024, **P3** = 07/10/2024)
  - Credit scores (no one has a score of zero)
  - Celsius (there is a zero on this scale, but that does not mean a lack of temperature)
- We can say "one week later", but we can't say "twice as late"

Ratio

Aston University
BIRMINGHAM UK

- We can categorise and rank data points
- We can infer equal intervals between them
- There is a true 'zero point'
- For example:
  - Source lines of code
  - Number of days delayed from deadline
  - Number of full-time programmers
- If you can say "there are twice as many" or "there is twice as much", and if zero has a specific meaning, it's a ratio scale

Absolute

Aston University
BIRMINGHAM UK

- A subset of 'ratio'
- Cannot be determined indirectly from any other measurements
- Typically used for counting
- For example:
  - Number of defects detected by a certain test method
  - Number of lines of code reused from a previous project

Admissible transformations

- Nominal scale
  - M = {RAD, Incremental, Waterfall, Spiral, Prototyping}
  - M' = {1, 18, 6, 5, 36}
- Ordinal scale
  - M = {1, 2, 3, 4, 5, 6} **e.g. programmer skill**
  - M' = {2, 6, 12, 20, 42, 88}
- Interval scale
  - M = {1, 2, 3, 4} **e.g. dates at which progress is regularly reviewed**
  - M' = {7, 9, 11, 13}
- Ratio scale
  - M = {0, 1, 2, 3} **e.g. number of hours spent on a task**
  - M' = {0, 60, 120, 180}
- Absolute scale
  - M = {0, 1, 2, 3} **e.g. number of lines of code re-purposed from a previous project**
  - M' = N/A

Which scale?

Aston University
BIRMINGHAM UK

1. You are assessing the programming languages used in a software project. The data includes: Python, Java, C++, Ruby, and PHP.
2. You are rating the quality of customer service on a scale of 1 to 5, with 1 being "Poor" and 5 being "Excellent."
3. You are measuring the time taken by a computer program to execute various tasks in seconds.
4. You are measuring temperature in degrees Celsius, where 0°C represents freezing point.
5. You are counting the number of errors in a software code.

Case study 1: software complexity

Aston University
BIRMINGHAM UK

- What do we mean by complexity?
    a. size
    b. control flow
    c. structure
    d. interface
    e. connection
    f. algorithm
- Software complexity: SC = F(a, b, c, d, e, f)
- How would we calculate complexity?
- Any attempt to reach a single-value complexity is likely to fail

Case study 2: developer productivity

Aston University
BIRMINGHAM UK

- Software productivity is a key management concern, considering:
  - cost containment
  - monitoring performance
  - comparison
  - assessment
- Straightforwardly, Pr = output / input
- In software, Pr = S / E (size / effort)
- What problems can arise from using SLOC
- What problems can arise from using defects resolved

The problem with productivity

Aston University
BIRMINGHAM UK

- Validity issues with Pr = SLOC / E
- It's an indirect measurement, but of what?
  - SLOC ignores quality
  - SLOC ignores difficulty/complexity
  - SLOC ignores functionality/utility
  - SLOC ignores task value
  - SLOC ignores those who do not directly write code
- SLOC is easy to measure, and superficially insightful
- What might be a better measurement?

Aston University
BIRMINGHAM UK

Software Project Management

Unit 2: Measurement, estimation & data analysis

Thais Webber
Richard Lee