Software Engineering COMP 201

Lecturer: **Sebastian Coope**

Ashton Building, Room G.18

E-mail: ***coopes@liverpool.ac.uk***

COMP 201 web-page:

http://www.csc.liv.ac.uk/~coopes/comp201

Software Cost Estimation

Software Cost Estimation

- **Software cost estimation** involves predicting the resources required for a software development process

Fundamental Estimation Questions

- How much **effort** is required to complete an activity?
- How much **calendar time** is needed to complete an activity?
- What is the **total cost** of an activity?
- Project estimation and scheduling and interleaved management activities

Software Cost Components

- Hardware and software costs
- Travel and training costs
- Effort costs (the dominant factor in most projects)
  - salaries of engineers involved in the project
  - Social and insurance costs
- Effort costs must take **overheads** into account
  - costs of building, heating, lighting
  - costs of networking and communications
  - costs of shared facilities (e.g library, staff restaurant, etc.)

Costing and Pricing

- Estimates are made to discover the cost, to the developer, of producing a software system
- There is not a simple relationship between the **development cost** and the **price charged** to the customer
- Broader organisational, economic, political and business considerations influence the price charged

Software Pricing Factors

Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Programmer Productivity

- A measure of the rate at which individual engineers involved in software development produce software and associated documentation
- Not quality-oriented although quality assurance is a factor in productivity assessment
- Essentially, we want to measure **useful functionality produced per time unit**

Productivity Measures

- **Size-related measures** based on some output from the software process. This may be lines of delivered source code, object code instructions, etc.
- **Function-related measures** based on an estimate of the functionality of the delivered software. Function-points are the best known of this type of measure

Measurement Problems

- Estimating the size of the measure
- Estimating the total number of programmer months which have elapsed
- Estimating contractor productivity (e.g. documentation team) and incorporating this estimate in overall estimate

Lines of Code

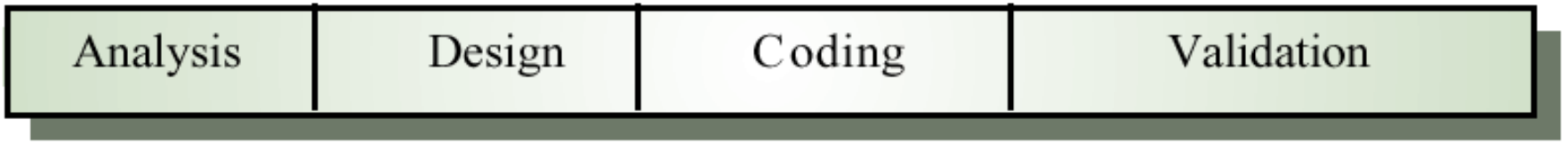
- What's a line of code?
  - The measure was first proposed when programs were typed on cards with one line per card
  - How does this correspond to statements as in Java which can span several lines or where there can be several statements on one line
- What programs should be counted as part of the system?
- Assumes linear relationship between system size and volume of documentation

Productivity Comparisons

- The lower level the language, the more productive the programmer
  - The same functionality takes more code to implement in a lower-level language than in a high-level language

High and Low Level Languages

Low-level language



High-level language

Function Points

- Based on a combination of program characteristics
  - external inputs and outputs
  - user interactions
  - external interfaces
  - files used by the system
- A weight is associated with each of these
- The function point count is computed by multiplying each raw count by the weight and summing all values

Object Points

- Object points are an alternative function-related measure to function points
- Object points are NOT the same as object classes
- The number of object points in a program is a weighted estimate of
  - The number of separate **screens** that are displayed
  - The number of **reports** that are produced by the system
  - The number of **modules** that must be developed

Productivity Estimates

- Real-time embedded systems, 40-160 LOC/P-month
- Systems programs , 150-400 LOC/P-month
- Commercial applications, 200-800 LOC/P-month
- In object points, productivity has been measured between 4 and 50 object points/month depending on type of application, tool support and developer capability

Factors Affecting Productivity

Factor	Description
Application domain experience	Knowledge of the application domain is essential for effective software development. Engineers whoalready understand a domain are likely to be the most productive.
Process quality	The development process used can have a significant effect on productivity. This is covered in Chapter 31.
Project size	The larger a project, the more time required for team communications. Less time is available for development so individual productivity is reduced.
Technology support	Good support technology such as CASE tools, supportive configuration management systems, etc. can improve productivity.
Working environment	As discussed in Chapter 28, a quiet working environment with private work areas contributes to improved productivity.

Quality and Productivity

- It could be argued that all metrics based on volume/unit time are flawed because they do not take **quality** into account
- Productivity may generally be increased at the cost of quality
- It is not clear how productivity/quality metrics are related
- If change is constant (**simplifying** and **improving** code for example) then an approach based on counting lines of code is not meaningful

Estimation Techniques

- There is no simple way to make an accurate estimate of the effort required to develop a software system
  - Initial estimates are based on inadequate information in a user requirements definition
  - The software may run on unfamiliar computers or use new technology
  - The people in the project may be unknown
- Project cost estimates may be self-fulfilling
  - The estimate defines the budget and the product is adjusted to meet the budget

Estimation Techniques

- Algorithmic cost modelling
- Expert judgement
- Estimation by analogy
- Parkinson's Law
- Pricing to win

Algorithmic Code Modelling

- A **formulaic approach** based on historical cost information and which is generally based on the **size of the software**
- Discussed later …

Expert Judgement

- One or more experts in **both** software development and the application domain use their experience to predict software costs. Process iterates until some consensus is reached.
- **Advantages**: Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems
- **Disadvantages**: Very inaccurate if there are no experts!

Estimation by Analogy

- The cost of a project is computed by comparing the project to a **similar project** in the same application domain
- **Advantages**: Accurate if project data available
- **Disadvantages**: Impossible if no comparable project has been tackled. Needs systematically maintained cost database

Parkinson's Law

- The project costs whatever resources are available
- **Advantages**: No overspend
- **Disadvantages**: System is usually unfinished

- Parkinson's Law states that work expands to fill the time available. The cost is determined by available resources rather than by objective statement.
- Example: Project should be delivered in 12 months and 5 people are available. Effort = 60 p/m

Pricing to Win

- The project costs whatever the customer has to spend on it
- **Advantages**: You get the contract
- **Disadvantages**: The probability that the customer gets the system he or she wants is small. Costs do not accurately reflect the work required

Top-Down and Bottom-Up Estimation

- Any of these approaches may be used top-down or bottom-up
- Top-down
  - Start at the system level and assess the overall system functionality and how this is delivered through sub-systems
- Bottom-up
  - Start at the component level and estimate the effort required for each component. Add these efforts to reach a final estimate

Estimation Methods

- Each method has strengths and weaknesses
- Estimation should be based on several methods
- If these do not return approximately the same result, there is insufficient information available
- Some action should be taken to find out more in order to make more accurate estimates
- Pricing to win is sometimes the only applicable method

Experience-Based Estimates

- Estimating is primarily experience-based
- However, new methods and technologies may make estimating based on experience inaccurate
  - Object-oriented rather than function-oriented development
  - Client-server systems rather than mainframe systems
  - Off the shelf components
  - Component-based software engineering
  - CASE tools and program generators

Pricing to Win

- This approach may seem **unethical** and **unbusinesslike**?
- However, when detailed information is lacking it may be the only appropriate strategy
- The project cost is agreed on the basis of an outline proposal and the development is constrained by that cost
- A detailed specification may be negotiated or an evolutionary approach used for system development

Algorithmic Cost Modelling

- Cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers
  - $\text{Effort} = A \times \text{Size}^B \times M$
  - A is an organisation-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process and people attributes
- Most commonly used product attribute for cost estimation is code size
- Most models are basically similar but with different values for A, B and M

Estimation Accuracy

- The size of a software system can only be known accurately when it is finished
- Several factors influence the final size
  - Use of COTS and components
  - Programming language
  - Distribution of system
- As the development process progresses then the size estimate becomes more accurate