Bournemouth University

ssrc
software systems research centre

Estimation and Prediction in Computing

Dr Keith Phalp

Questions and Overview

- What is estimation (a recap – hopefully)?
- Why do we need (usually cost) estimation?
- Why is estimation difficult for computing (as opposed to other engineering disciplines)?
- What are the main approaches which have been used (historically, where you worked).
- Which are more generic (can be applied widely)?
- Summary of techniques.
- *An estimation exercise.*
- Extras: Planning and delivery of estimation.

Cost Estimation

- Estimation in Computing is ‘typically’ (historically) about estimating project costs, and related durations.
- Prediction of the “most likely” value (s) (that’s the theory).
  - Many input variables (often referred to as cost drivers).
  - Output in terms of effort (i.e., person hours / months / years).
- Different development environments (ideally) determine which variables included in the cost value.
- Potential for over and under estimates
  - trick is not by too much (*20 times under for example*).
  - ***Do you think over / under estimating is roughly equal?***

Why we need to estimate

- Lots of sources for similar figures, but one report shows that:
  - 55% of projects over budget
  - 53% of projects cost 189% more than initial estimates (Standish Report(1994))
- This is actually one of the more ‘positive’ sets of reports, others have much higher ‘failure rates’, for projects.
  - General consensus is that things haven’t really improved (continue to have high profile examples).
- Need to make predictions in order to plan and control computing projects, e.g. effort, number of features, defects and reliability, capacity, availability, ….

What we need to estimate

- Bottom line: we need to know total costs, plus components of the total:
  - Labour (Effort) – ***usually the major cost***
  - Equipment (Computers, Software, ...)
  - Consumables
  - Expenses
  - Subcontracts
  - Overhead costs
  - Etc..
- Also need to separate out (why)
  - Development or project costs.
  - Project timescales and staffing required for the project
  - ***Is double effort double time or double people or neither?***

Diseconomies of scale

- Productivity levels tend to decrease as projects get larger.
  - Engineers spend relatively more time communicating.
  - Relatively more design and integration activity is required, since there are more component parts.
  - Relatively more effort is spent in verification and validation.
  - The management effort per engineer increases.
- Also evidence of *economies* of scale in *some* organisations.
- ***What happens if I had more staff to a project which is running late? (Why might this be?)***
- Therefore, important to get it right at the beginning.
  - Though also problems with trying to estimate too early - ***what might these be?***

Estimation in Computing

- Within the computing context the majority of our projects are ‘one-offs’, a new network for a particular site, a new system for company x, and technology has changed markedly since the last time.
- There are similarities of course, and this may well have a bearing on the kinds of estimation that we can use in the wider context.
- In addition, we are plagued by change and uncertainty.

- Lots of other things we might also wish to ‘predict’, e.g., errors, intrusions etc.

Change and uncertainty

- *Change and uncertainty typify the environment for Computing projects.*
- Operating environment, other software, networks, business models.
- The old (or new) legacy systems problem …
- Deadline, budget constraints, priorities.
- Staffing. Also access to client's staff.
- Technology (again often changing even during the project…, or may even be working towards other new technologies).
- Functional (or other) requirements change (during the project and when delivered: Features, Reliability, Performance, Usability…
  - and if we have to write software this is intrinsically difficult to predict.

Software is Description

- *An inherent problem with our domain (industry)*
- With hardware engineering (cars, aeroplanes, computers etc.) you have to **design** it and then **build** it. But . .
  - **"To build software is to build a machine simply by describing it"**

    (Michael Jackson (the software requirements one), 1995)
- You can't see it, you can't touch it and you can't weigh it.
- Its the holes.. (Richard Feynman). It's "think-ware".

- Inappropriateness of manufacturing paradigm.
- All of our building is actually like building the prototype in manufacturing; the replication (manufacturing is trivial).

Estimation Difficulties

- If that's not enough, estimation brings its own issues.
  - lack of data, and what exists may be unsystematic
  - "noise"
  - phenomena not well understood
  - interactions *among* variables are often complex
  - lack of expertise
  - past experience is used inconsistently if used at all [though we may seek to change that].
  - hard to [incorporate] all the human factors influencing cost, e.g.. political problem, people factors.
  - *Tom DeMarco, T., (1982) Controlling Software Projects. Management, Measurement & Estimation. Yourdon Press, NY, 1982*

So how do we estimate?

- Expert Judgement.
- Bottom up (activity based) approaches (or top down).
- Statistical methods (e.g., regression models).
- Parametric (algorithmic), e.g., Function Points, COCOMO
- Analogy. Use data from similar, past projects (often with the help of other approaches).
- Natural computing approaches, e.g., neural networks, rule based, data mining, CBR, genetic algorithms, PSO, hybrids.
- Hybrid approaches (of the above).
- *Price to win*
- *Other approach (consulting the stars?)*

Your experience of estimation

- Expert Judgement
- Bottom up
- Top Down
- Statistical methods
- Function Point
- COCOMO
- Other Parametric
- Analogy
- ANNs
- ANNs (neural nets)
- Rule based (expert systems)
- CBR
- GAs
- Hybrid approaches (of the above).
- *Price to win*
- *Other DIY approach*

Expert Judgement

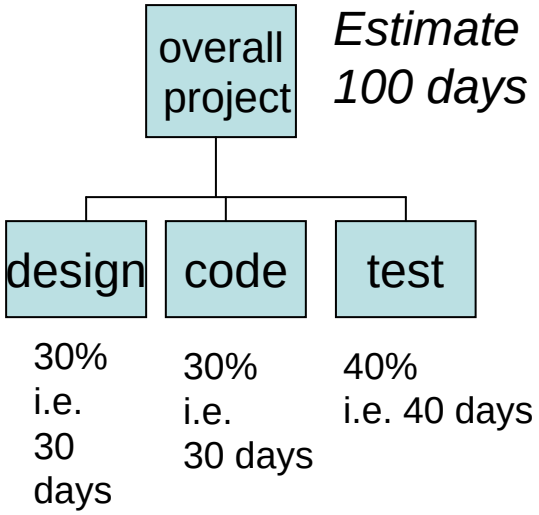
- For whole system or part of system.
- May involve several experts, drawing on their experiences.
- Could ask an outside consultant.
  - [***Any disadvantages?***]
- Reach a consensus (if possible).
  - Delphi Poll is one method.
  - ***We will examine Delphi in further detail later.***

Bottom up

- Bottom-up. Estimate effort for system components and/or activities, combine for overall project estimate.
- Strengths
  - Could involve allocated engineers; those responsible for development of each unit assess the costs.
  - Allows for differences between system components
  - Can be used for cost tracking and useful when no past project data to go on.
- Weaknesses
  - Could overlook project activities such as integration or configuration management
  - Requires detailed requirements or spec, so not always possible early on in life cycle, plus time consuming.

Top down

- Derive effort estimate for project as a whole (often using effort drivers).
- Distribute estimate to components or project activities.
- Advantages
  - Requires minimal project detail
  - Relatively fast and easy to implement
  - Focus on system level activities
- Disadvantages
  - Tend to overlook low level components
  - No detailed basis



Statistical Approaches

- Traditionally estimation assumed probabilistic distributions.
- A typical (common) statistical approach is linear regression.
- A simple version finds a line of fit (can be multi-dimensional) and uses this to predict further point estimates.
  - Other approaches, e.g., PCA can be used to find which variables appear to have most impact on the estimate.
- Often used as a baseline for comparison with other (more advanced) approaches.
- Interestingly while computing uses many approaches to prediction other disciplines can still be very reliant on the statistical approach to prediction (or estimation).

Parametric Models

- Mathematical equations to perform software estimation (using parameters).
- Equations are based on theory or historical data.
- Use inputs (cost drivers) such as source LOC.
- Accuracy can be improved by calibrating the model to the specific environment.
- COCOMO (lines of code) and function points most well known examples.
  - Though many adaptations, e.g., use case points.

Parametric Approach

- Using formulae derived from past projects.
- Inputs are project characteristics, quantified.
- Outputs are estimated project cost/time.
- Key benefit is repeatability.
- Requires a calibrated model based on a large set of good quality data from past projects *in this organisation*. [**Why?**]
- Not good for a new type of project. [**Why?**]

Building parametric models

- Examples of system characteristics
  - no of screens x 4 hours
  - no of reports x 2 days
  - no of entity types x 2 days
- The quantitative relationship between the input and output products of a process can be used as the basis of a parametric model.
- Models vary in their sophistication.

Simplified Parametric Model

- Use some historical data
- A simplistic model for an estimate
  - estimated effort = (system size) / productivity, e.g.,

    system size = lines of code

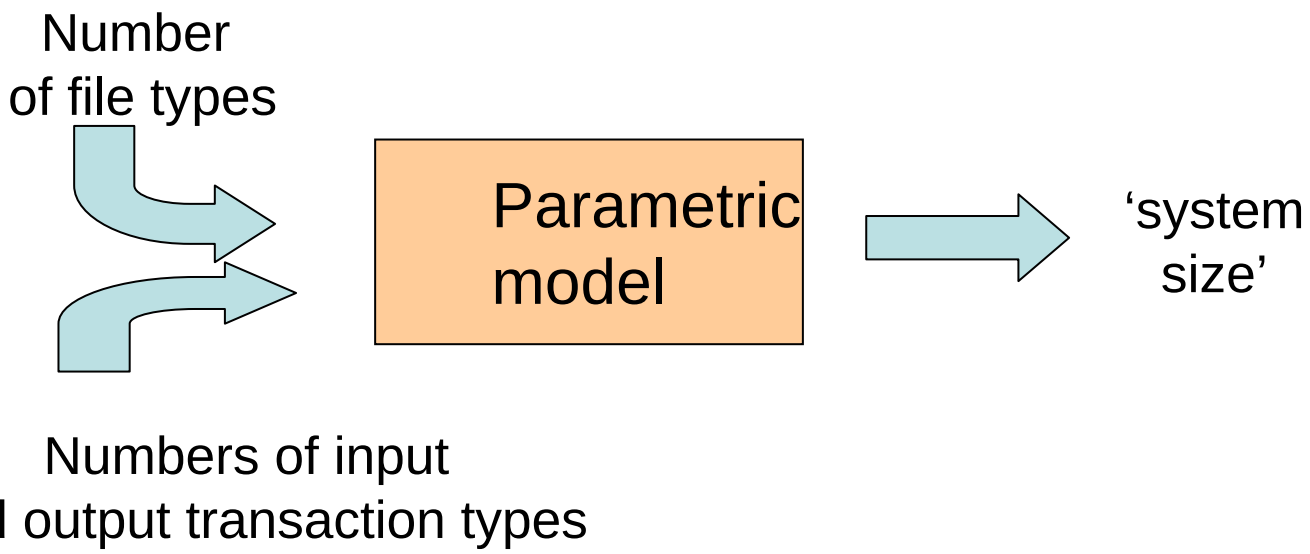
    productivity = lines of code per day

    productivity = (system size) / effort

    based on past projects

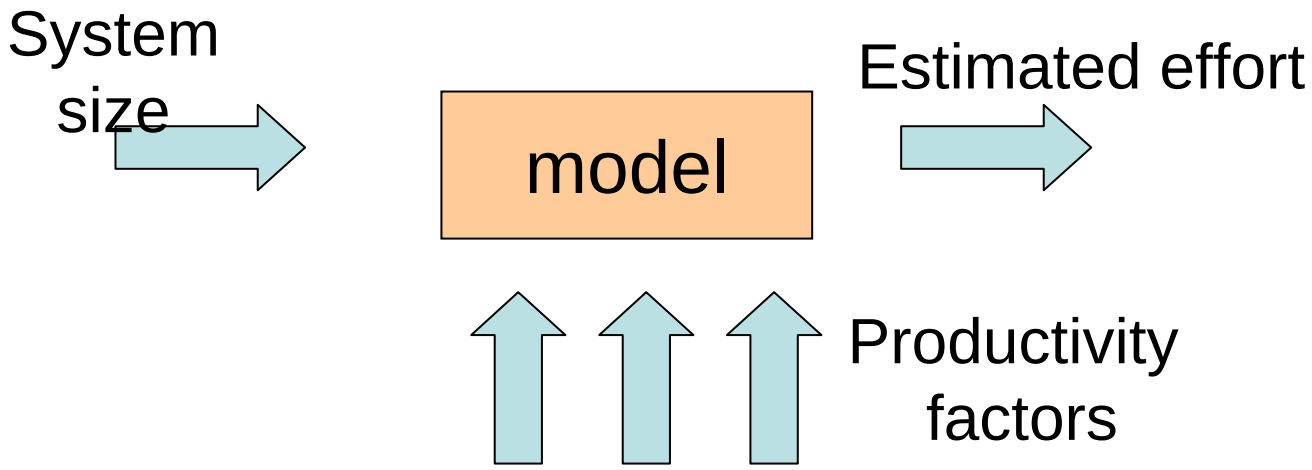
Focus of models - Size

- Some models focus on task or system size e.g., Function Points, Bang model.
- FPs originally used to estimate Lines of Code, rather than effort.



Effort and productivity

- Other models focus on productivity or effort as outputs, e.g., COCOMO
- Lines of code (or latterly FPs etc) an input.



Parametric Models: Evaluation

Advantages

- Generally easy to use, tools and support
- Repeatable
- Helps us to understand the factors (cost drivers) that have an impact on cost (effort) and time (schedule).

Disadvantages

- Calibration and validation is an issue
- Historical data not necessarily relevant to new projects
- Can be misleading.
- Some suggestions that models can be used (tweaked) to come up with the answer that the manager wants in terms of resources.
- *Inputs are often unknowns themselves. .. "if I knew the lines of code I'd know how big it was".*

COCOMO II

- Inputs
  - Estimated system size (lines of code) - from expert judgement (or Function Points?).
  - Project characteristics: examples - system complexity, engineers' experience, quality of tools used
- Allows for code reuse and adaptation.
- Published formulae calibrated from 161 projects.

Reference: Boehm B.W. *et al*. (2000) *Software Cost Estimation with COCOMO II*, Prentice Hall.

See Boehm (1981) for the original COCOMO model.

Function Points

- Size of system based on analysis of requirements, specifically:
  - Number of inputs and outputs.
  - Number of user transactions.
  - Number of stored entities accessed.
- Effort estimates adjusted using project characteristics.
- Good results for certain application domains.

See for example Hughes & Cotterell (2002) pp90-93.

Analogy

- Often uses “Case-based reasoning” approach, and some use terms synonymously.
- Use data from past projects - cost/time/effort.
- Find “similar” projects. ***What does this mean?***
  - Criteria? ***How do we measure similarity?***
- May have to scale up/down.
- May have to adjust for changed circumstances.
- Relies on good quality information being retained for past projects. *(What would this be?)*

The most similar project (s)

- Let’s suppose I have (unusually) kept records of past projects.
- A similar project might be a good start for estimating my new project.
- Which is the most similar?
- How would I determine this?
- Typically have a number of categories (often numeric or categorical).
- A simple measure would make no assumptions (or weighting of these variables).

Angel tool (BU)

- Shell
- *n* features (continuous or categorical)
- Brute force search for optimal subset of features —
- O((2**n) -1)
- Measures Euclidean distance (standardised dimensions)
- Uses *k* nearest cases.
- Simple adaptation strategy (weighted mean).
- With k=1 becomes a NN technique

Analogy Evaluation

Advantages

- Based on actual project data
- Studies show it out performs algorithmic (parametric)
- methods
- Even better when used with expert judgement

Disadvantages

- Impossible if no comparable project had been tackled
- in the past.
- How well does the previous project represent this one?

Further approaches

- To some extent mirrors developments in natural computing (AI), with expert systems (rules), logic approaches, nets, CBR, and latterly GAs, PSO and various hybrids.

- These approaches can be used for many prediction and estimation problems, as well as for computing projects.

- Strong history within BU and latterly Software Systems of examination of these approaches, from traditional routes in cost estimation and software metrics, through empirical evaluation of approaches, CBR and analogy (Angel tool) to further uses of AI approaches to prediction and estimation.

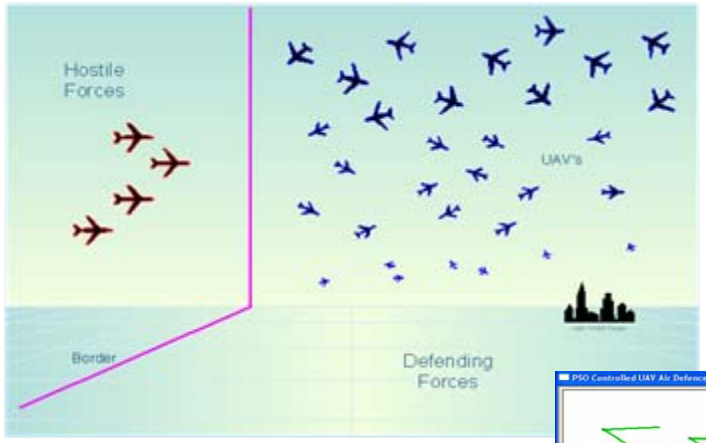
Some BU work on prediction

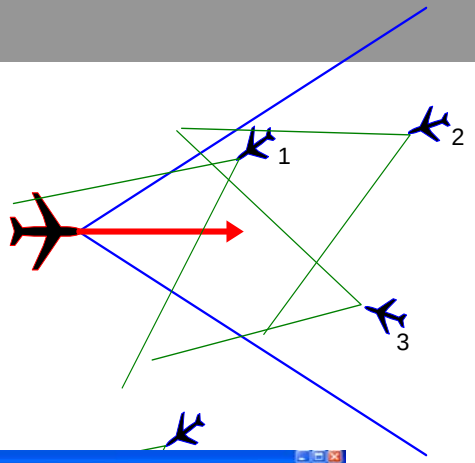
- E.g., well cited early paper:
  - *Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M. and Webster, S. (2000), An Investigation of Machine Learning Based Prediction Systems, Journal of Systems and Software*
- A number of PhDs in such areas, e.g.,
  - PhD: Premraj, R., Meta-Data to Enhance Case-Based Prediction: completed April 2006
  - PhD: Banks, A., A Naturally Inspired Guidance System for Unmanned Autonomous Vehicles Employed in a Search Role: completed Oct 2009. Early work by the team gained best paper award.
  - Current work in SSRC to use natural computing approaches for network based intrusion detection.
- Previous EC funded project had used approaches to detect fraud in mobile (phone) networks.

Natural Search Strategies for Un-manned Autonomous Vehicles

Bournemouth University

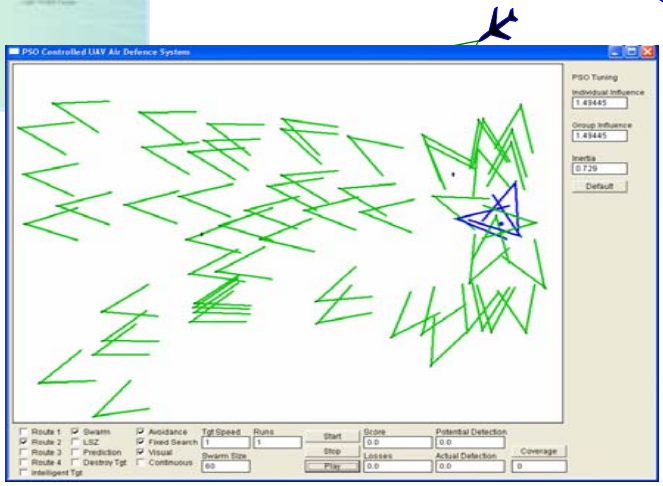
ssrc software systems research centre





Key

- Velocity Vector
- LSZ Neighbourhood
- UAV RADAR Range
- Hostile Aircraft
- Interceptor UAV



Machine Learning for Network Based Intrusion Detection

- Intrusion detection in this context refers to detecting malicious behaviour in computer systems or computer networks.
- Magnitude of data is generally so immense that it becomes an impossible task for a human; the amount of data will grow quicker than it can be analysed.
- One of the main goals of IDSs is automating the detection process.
- Found many results from previous research were contradictory, finding that the performance of the techniques varies significantly in different circumstances, as use of the data set is altered.
- Empirical investigation reveals what impact various issues with the data set has on the performance of the techniques, explores ways of dealing with these issues, and discusses implications.
- Class imbalance has been found to be a significant challenge to most of the machine learning classifiers adopted in the literature.
- One part of this research focused on demonstrating that this is indeed an issue for intrusion detection, affecting the performance of the classifiers.
- Further from this, a novel approach to learning from imbalanced data has been proposed, using multi-objective genetic algorithms to evolve artificial neural networks and classifier ensembles.

Stages in Delphi

1. Experts receive spec + estimation form
2. Discussion of product + estimation issues
3. Experts produce individual estimate
4. Estimates tabulated and returned to experts
5. Only expert's personal estimate identified
6. Experts meet to discuss results
7. Estimates are revised
8. Cycle continues until an acceptable degree of convergence is obtained.

- *Can also use a fixed number of rounds, or other means to agree acceptable convergence.*
- *For computing projects, there are typically a number of components estimated, e.g.,, from WBS, but the idea can be used for a variety of applications (domains).*
- *Used Delphi on Masters Integrating Studies (see separate slides).*

Summary

- Why we need to estimate.
- Difficulties and issues within estimation
- A variety of techniques from expert ‘opinion’, through statistical and parametric approaches, to analogy and natural computing approaches.
- CBR and analogy plus natural approaches appear more generic (if suitable data) and can be applied to wide range of estimation and prediction problems.
- Also found to perform well against traditional approaches.
- However, cannot under-estimate the human element (at least as a sanity check).
- *Collective approaches also bring further power (e.g., Delphi).*

An exercise

- We are to use Delphi to attempt to produce an estimate.
- We will use teams and rounds.

- Extras follow on estimating process ->

Who estimates?

- Project manager, developer, or specialist
  - Can use experience of past projects
  - Usually experienced
- Consultant
  - Can provide unbiased estimate
  - Tend to use empirical, algorithmic or other “non-expert” methods of estimation
  - Unlikely that a model will outperform an expert
    - *So why are models so popular?*
  - Possible lack of historical data to calibrate the model
- ***Your experience, who estimated project cost?***

Further issues

- A study of 598 (mostly development projects) found that:
  - 35% do not make an estimate
  - 50% do not record project data
  - 57% do not use cost accounting
  - 80% projects have overrun budget and/or timescale
  - Mean budget / timescale overrun is 50%
- *Heemstra, FJ "Software Cost Estimation" IST 34 (10) 1992 p627-39*
- Some improvements, though again recording tends to be poor despite the prevalence of ISO, TickIT, CMM, ITIL, etc.

When to Estimate

- This might seem obvious to you, but people see this as a one shot deal.
- Cost estimation should continue throughout the project (to refine or update estimates)
- Effective monitoring and control of project costs is necessary to verify and improve accuracy of estimates
- When to measure is as important as what to measure (and people need to understand this in terms of their process, even augmenting their process models).

Planning for Estimation

- Establish Plan
  - What data should be collected
  - Why are we gathering this data (our goals)
- Cost estimation for initial requirements
- Use several methods, if time and resources allow
  - Spread the risk
  - If wide variation in methods, then re-evaluate the information used
- Re-estimates, re-plan
- Final assessment of cost estimation at project close