

Agile Planning and Cost Estimation



**Idaho State
University**

Computer
Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR



Topics covered

- Agile Planning
- Estimation techniques
- COCOMO cost modeling



Agile planning

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
 - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes



Agile planning stages

- Release planning, which looks ahead for several months and decides on the features that should be included in a release of a system.
- Iteration planning, which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2-4 weeks of work for the team.



Approaches to agile planning

- Planning in Scrum
- Based on managing a project backlog (things to be done) with daily reviews of progress and problems
- The planning game
 - Developed originally as part of Extreme Programming (XP)
 - Dependent on user stories as a measure of progress in the project



Story-based planning

- The planning game is based on user stories that reflect the features that should be included in the system
- The project team read and discuss the stories and rank them in order of the amount of time they think it will take to implement the story.
- Stories are assigned 'story points' reflecting their size and difficulty of implementation
- The number of story points implemented per day is measured giving an estimate of the team's 'velocity'
- This allows the total effort required to implement the system to be estimated.

The planning game





Release and iteration planning

- Release planning involves selecting and refining the stories that will reflect the features to be implemented in a release of a system and the order in which the stories should be implemented.
- Stories to be implemented in each iteration are chosen, with the number of stories reflecting the time to deliver an iteration (usually 2 or 3 weeks).
- The team's velocity is used to guide the choice of stories so that they can be delivered within an iteration.



Task allocation

- During the task planning stage, the developers break down stories into development tasks.
 - A development task should take 4–16 hours.
 - All of the tasks that must be completed to implement all of the stories in that iteration are listed.
 - The individual developers then sign up for the specific tasks that they will implement.
- Benefits of this approach:
 - The whole team gets an overview of the tasks to be completed in an iteration.
 - Developers have a sense of ownership in these tasks and this is likely to motivate them to complete the task.



Software delivery

- A software increment is always delivered at the end of each project iteration.
- If the features to be included in the increment cannot be completed in the time allowed, the scope of the work is reduced.
- The delivery schedule is never extended.



Agile planning difficulties

- Agile planning is reliant on customer involvement and availability
- This can be difficult to arrange, as customer representatives sometimes have to prioritize other work and are not available for the planning game.
- Furthermore, some customers may be more familiar with traditional project plans and may find it difficult to engage in an agile planning process.



Agile planning applicability

- Agile planning works well with small, stable development teams that can get together and discuss the stories to be implemented.
- However, where teams are large and/or geographically distributed, or when team membership changes frequently, it is practically impossible for everyone to be involved in the collaborative planning that is essential for agile project management.

Estimation Techniques

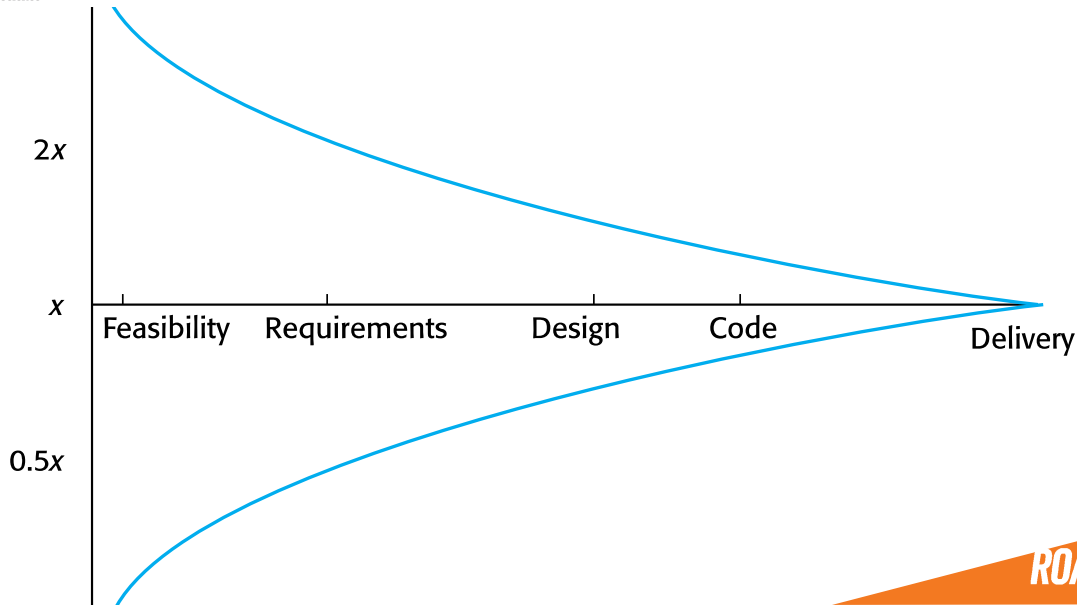


Estimation techniques

- Organizations need to make software effort and cost estimates. There are two types of techniques that can be used to do this:
 - Experience-based techniques: The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.
 - Algorithmic cost modeling: In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.



Estimate uncertainty





Experience-based approaches

- Experience-based techniques rely on judgments based on experience of past projects and the effort expended in these projects on software development activities.
- Typically, you identify the deliverables to be produced in a project and the different software components or systems that are to be developed.
- You document these in a spreadsheet, estimate them individually and compute the total effort required.
- It usually helps to get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate.



Problem with experience-based approaches

- The difficulty with experience-based techniques is that a new software project may not have much in common with previous projects.
- Software development changes very quickly and a project will often use unfamiliar techniques such as web services, application system configuration or HTML5
- If you have not worked with these techniques, your previous experience may not help you to estimate the effort required, making it more difficult to produce accurate costs and schedule estimates.



Algorithmic cost modeling

- Cost is estimated as a mathematical function of product, project, and process attributes whose values are estimated by project managers:
 - $Effort = A \times Size^B \times M$
 - A is an organization-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process and people attributes.
- The most commonly used product attribute for cost estimation is code size
- Most models are similar but they use 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 reused systems and components;
 - Programming language;
 - Distribution of system;
- As the development process progresses then the size estimate becomes more accurate
- The estimates of the factors contributing to B and M are subjective and vary according to the judgment of the estimator.



Effectiveness of algorithmic models

- Algorithmic cost models are a systematic way to estimate the effort required to develop a system. However, these models are complex and difficult to use.
- There are many attributes and considerable scope for uncertainty in estimating their values.
- This complexity means that the practical application of algorithmic cost modeling has been limited to a relatively small number of large companies, mostly working in defense and aerospace systems engineering.

COCOMO cost modeling



COCOMO cost modeling

- An empirical model based on project experience
- Well-documented, 'independent' model which is not tied to a specific software vendor
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2.
- COCOMO 2 takes into account different approaches to software development, reuse, etc.

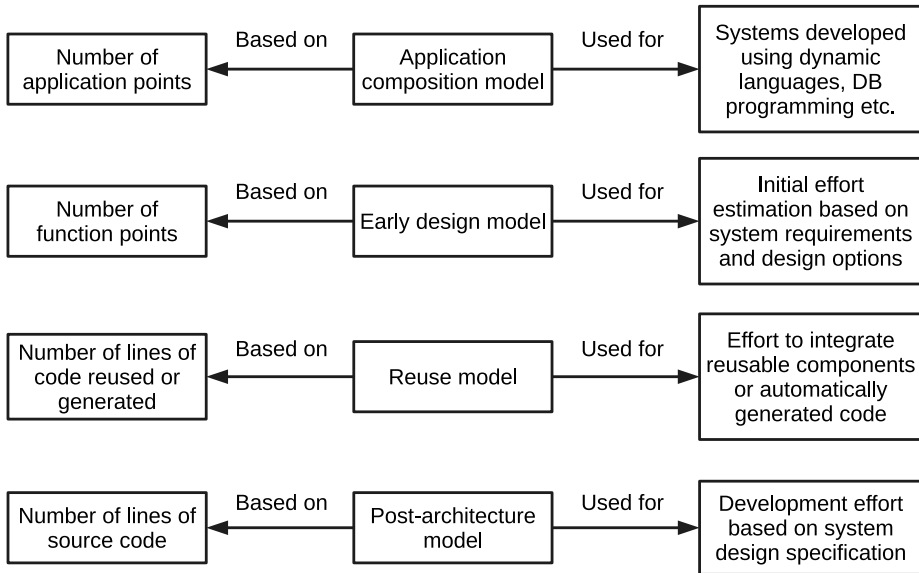


COCOMO 2 models

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO 2 are:
 - **Application composition model.** Used when software is composed from existing parts.
 - **Early design model.** Used when requirements are available but design has not yet started.
 - **Post-architectural model.** Used once the system architecture has been designed and more information about the system is available.



COCOMO estimation models





Application composition model

- Supports prototyping projects and projects where there is extensive reuse.
- Based on standard estimates of developer productivity in application (object) points/month
- Takes software tool use into account
- Formula is:
 - $PM = (NAP \times (1 - \%resue/100))/PROD$
 - Where: PM is the effort in person-months, NAP is the number of application points and $PROD$ is the productivity.



Application-point productivity

Developer's experience and capability	Very low	Low	Nominal	High	Very high
ICASE maturity and capability	Very low	Low	Nominal	High	Very High
PROD (NAP/month)	4	7	13	25	50



Early design model

- Estimate can be made after the requirements have been agreed.
- Based on a standard formula for algorithmic models
- $PM = A \times Size^B \times M$, where:
 - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$;
 - $A = 2.94$ in initial calibration
 - $Size$ in KLOC
 - B varies from 1.1 to 1.24 depending on the novelty of the project, development flexibility, risk management approaches, and the process maturity.



Multipliers

- Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.
 - RCPX - product reliability and complexity;
 - RUSE - the reuse required
 - PDIF - platform difficulty
 - PREX - personnel experience
 - PERS - personnel capability
 - SCED - required schedule
 - FCIL - the team support facilities

Project duration and staffing

- As well as effort estimation, managers must estimate the calendar time required to complete a project and when staff will be required.
- Calendar time can be estimated using a COCOMO 2 formula:
 - $TDEV = 3 * (PM)^{(0.33+0.2*(B-1.01))}$
 - PM is the effort computation and B is the exponent computed as previously discussed (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project.
- The time require dis independent of the number of people working on the project.



Staffing requirements

- Staff required can't be computed by dividing the development time by the required schedule
- The number of people working on a project varies depending on the phase of the project
- The more people who work on the project, the more total effort is usually required.
- A very rapid build-up of people often correlates with schedule slippage.



Key points

- The agile planning game involves the whole team in project planning. The plan is developed incrementally and, if problems arise, it is adjusted so that software functionality is reduced instead of delaying the delivery of an increment.
- Estimation techniques for software may be experience-based, where managers judge the effort required, or algorithmic, where the effort required is computed from other estimated project parameters.
- The COCOMO II costing model is a mature algorithmic cost model that takes project, product, hardware and personnel attributes into account when formulating a cost estimate.



Are there any questions?