Software Engineering

Week # 6

LECTURER: ABDULRAHMAN QAIM

Software pricing

Estimates are made to discover the cost, to the developer, of producing a software system.

- You take into account, hardware, software, travel, training and effort costs.

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.

Factors affecting software pricing

Factor	Description
Market opportunity	A development organization 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 organization the opportunity to make a greater profit later. The experience gained may also help it develop new products.
Cost estimate uncertainty	If an organization is unsure of its cost estimate, it may increase its price by a 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.

Factors affecting software pricing

Factor	Description
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can 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 smaller than normal profit or break even than to go out of business. Cash flow is more important than profit in difficult economic times.

Estimation techniques

Organizations need to make software effort and cost estimates. There are two types of technique 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.

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.

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.

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.

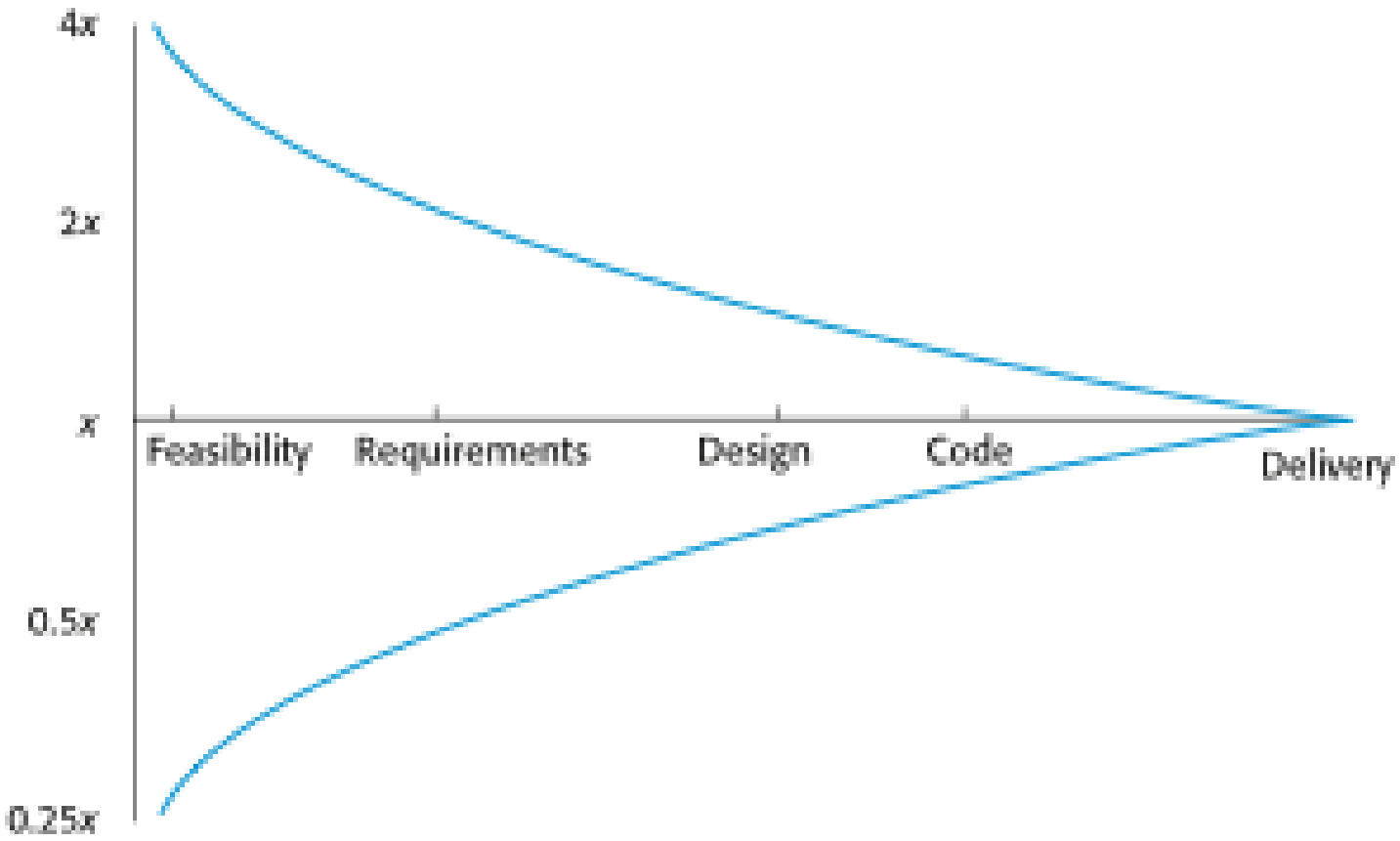
Several factors influence the final size

- Use of Commercial Off the shelf (COTS) 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.

Estimate uncertainty



The COCOMO 2 model

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.

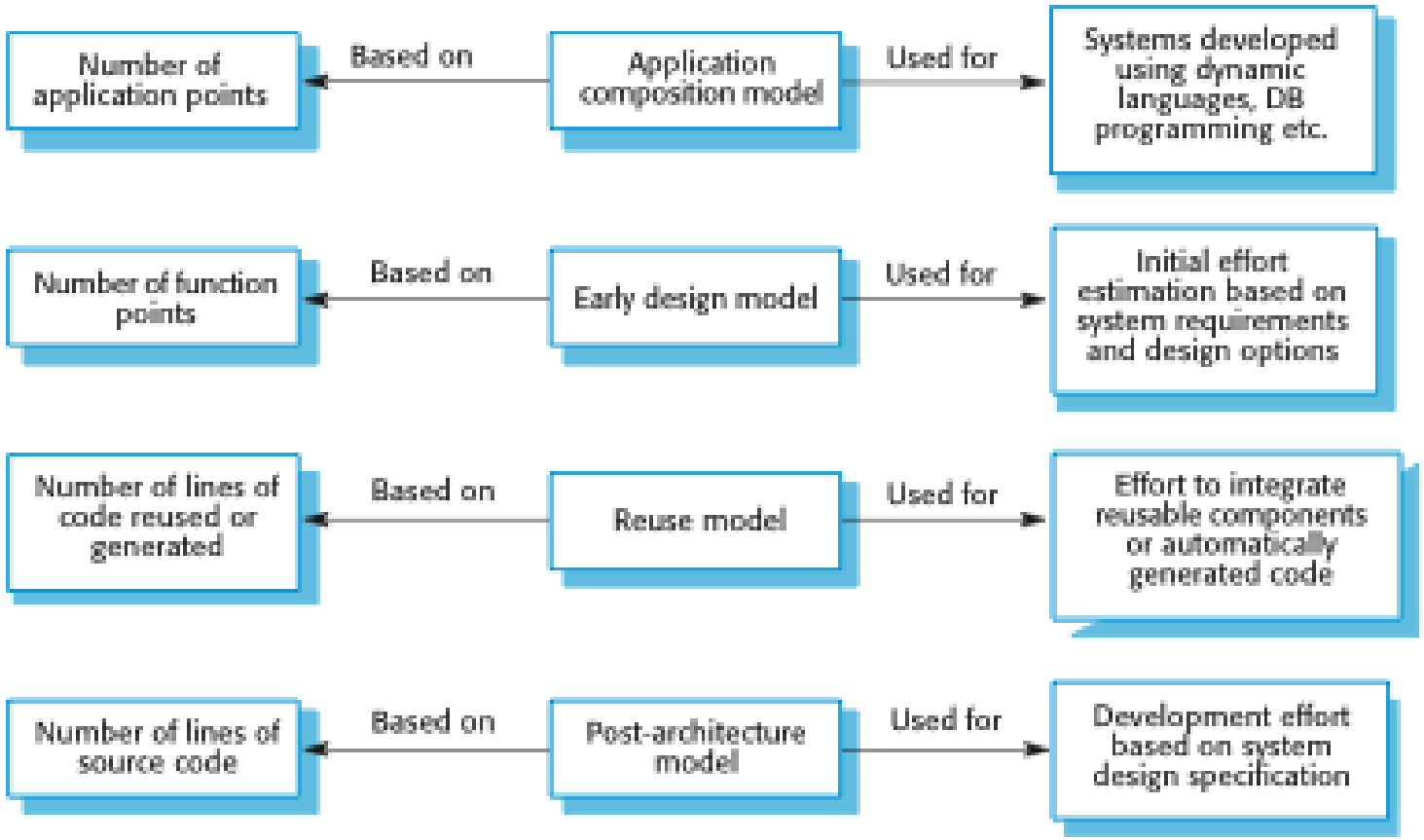
COCOMO 2 models

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.
- Reuse model. Used to compute the effort of integrating reusable components.
- Post-architecture model. Used once the system architecture has been designed and more information about the system is available.

COCOMO estimation models



COCOMO estimation models

1. Organic

A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. Semi-detached

A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environments lie in between organic and embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.

3. Embedded

A software project requiring the highest level of complexity, creativity, and experience requirement falls under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

COCOMO estimation models

1. **Cost Estimation:** To help with resource planning and project budgeting, COCOMO offers a methodical approach to software development cost estimation.
2. **Resource Management:** By taking team experience, project size, and complexity into account, the model helps with efficient resource allocation.
3. **Project Planning**: COCOMO assists in developing practical project plans that include attainable objectives, due dates, and benchmarks.
4. **Risk management**: Early in the development process, COCOMO assists in identifying and mitigating potential hazards by including risk elements.
5. **Support for Decisions**: During project planning, the model provides a quantitative foundation for choices about scope, priorities, and resource allocation.
6. **Benchmarking**: To compare and assess various software development projects to industry standards, COCOMO offers a benchmark.
7. **Resource Optimization:** The model helps to maximize the use of resources, which raises productivity and lowers costs.

Types of Cocomo Model

1. Basic Model

E = a(KLOC)^b

Time = c(Effort)^d

Person required = Effort/ time

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Types of Cocomo Model

2. Intermediate Model

Product attributes:

1. Required software reliability extent
2. Size of the application database
3. The complexity of the product
4. Run-time performance constraints
5. Memory constraints
6. The volatility of the virtual machine environment
7. Required turnabout time
8. Analyst capability
9. Software engineering capability
10. Application experience
11. Virtual machine experience
12. Programming language experience
13. Use of software tools
14. Application of software engineering methods
15. Required development schedule

Types of Cocomo Model

3. Detailed Model

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort. The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model