Unit 4 Estimation and Budgeting of Projects

Structure:

4.1 Introduction
Objectives
4.2 Software Cost Estimation
4.3 COCOMO Model
4.4 Budgeting
4.5 Summary
4.6 Terminal Questions
4.7 Answers

4.1 Introduction

Dear student, in the last unit you studied about project planning. In this unit you are going to study about project estimation and budgeting. Software cost estimation plays a vital role in project management. The reason for this strong emphasis on software cost estimation is that it provides the vital link between the general concepts and techniques of economic analysis and the particular world of software engineering. There is no good way to perform a software cost-benefit analysis, breakeven analysis, or make-or-buy analysis without some reasonably accurate method of estimating software costs, and their sensitivity to various product, project, and environmental factors. Software cost estimation techniques also provides an essential part of the foundation for good software management.

In this unit we will highlight an introduction to cost estimation, cost estimation methods, comparison of cost estimation methods, COCOMO cost estimation models (Basic, Intermediate and COCOMOII). We are also going to discuss the estimation of software budgeting and its three methods like NPV, payback methods and ROI.

Objectives:

After studying this unit, you should be able to:

- discuss various issues in software cost estimation
- explain the COCOMO model of software development
- describe software budgeting

4.2 Software Cost Estimation

Dear student, cost in a project is due to the requirements for software, hardware and human resources. The bulk of the cost of software development is due to the human resources needed, and most cost estimation procedures focus on this aspect. Most cost estimates are determined in terms of person-months (PM).

As the cost of the project depends on the nature and characteristics of the project, at any point, the accuracy of the estimate will depend on the amount of reliable information we have about the final product. When the project is being initiated or during the feasibility study, we have only some idea of the data the system will get and produce and the major functionality of the system. There is a great deal of uncertainty about the actual specifications of the system. As we specify the system more fully and accurately, the uncertainties are reduced and more accurate cost estimates can be made. Despite the limitations, cost estimation models have matured considerably and generally give fairly accurate estimates. Software cost estimation techniques also provides an essential part of the foundation for good software management.

Software project management begins with a set of activities that are collectively called *project planning.* Before the project can begin, the manager and the software team must estimate the work to be done, the resources that will be required, and the time that will elapse from start to finish. Whenever estimates are made, we look into the future and accept some degree of uncertainty as a matter of course.

According to Frederick Brooks, although estimating is as much art as it is science, this important activity need not be conducted in a haphazard manner. Useful techniques for time and effort estimation do exist. Process and project metrics can provide historical perspective and powerful input for the generation of quantitative estimates. Past experience (of all people involved) can aid immeasurably as estimates are developed and reviewed. Because estimation lays a foundation for all other project planning activities and project planning provides the road map for successful software engineering, we would be ill-advised to embark without it.

Steps for Estimation

Let's discuss various steps involved in cost estimation.

Step 1: Establish Objectives

- Key the estimating objectives to the needs for decision making information.
- Balance the estimating accuracy objectives for the various system components of the cost estimates.
- Re-examine estimating objectives as the process proceeds, and modify them where appropriate.

Step 2: Plan for Required Data and Resources

If we consider the software cost-estimation activity as a mini project, then we automatically cover this problem by generating a project plan at an early stage. The mini plan includes an early set of notes on the why, what, when, who, where, how, how much, and whereas of your estimating activity.

Step 3: Pin Down Software Requirements

It is important to have a set of software specifications that are as unambiguous as possible (subject to qualifications with respect to our estimating objectives). The best way to determine to what extent a software specification is cost table is to determine to what extent it is testable. A specification is testable to the extent that one can define a clear pass/fail test for determining whether or not the developed software will satisfy the specification. In order to be testable, specifications must be specific, unambiguous, and quantitative wherever possible.

Step 4: Work Out as Much Details as Feasible

"As feasible" here means "as is consistent with our cost-estimating objectives". In general, the more detail to which we carry out our estimating activities, the more accurate our estimates will be, for three main reasons: a) the more detail we explore, the better we understand the technical aspects of the software to be developed; b) the more pieces of software we estimate, the more we get the law of large numbers working for us to reduce the variance of the estimate; c) the more we think through all the functions the software must perform, the less likely we are to miss the costs of some of the more unobtrusive components of the software

Step 5: Use Several Independent Techniques and Sources

None of the alternative techniques for software cost estimation is better than the others from all aspects, their strengths and weaknesses are complementary. It is important to use a combination of techniques, in order to avoid the weakness of any single method and to capitalize on their joint strengths.

Step 6: Compare and Iterate Estimates

The most valuable aspect of using several independent cost-estimation techniques is the opportunity to investigate why they give different estimates. An iteration of the estimates after finding why they give different estimates may converge to a more realistic estimate.

Step 7: Follow-up

Once a software project is started, it is essential to gather data on its actual costs and progress and compare these to the estimates because of: Software estimating inputs are imperfect (sizing estimates, cost driver ratings). It is important to update the cost estimate with the new knowledge of the cost drivers by comparing the estimates to actual, providing a more realistic basis for some projects does not exactly fit the estimating model. Both near-term project-management feedback and long-term model-improvement feedback of any estimates-versus-actual differences are important.

Software Projects tend to be volatile as components are added, split up, rescoped, or combined in unforeseeable ways as the project progresses. The project manager needs to identify these changes and generate a more realistic update of the estimated upcoming costs. Software is an evolving field. Estimating techniques are all calibrated on previous projects, which may not have featured the future projects environment. It is important to sense differences due to these trends and incorporate them into improved project estimates and improved estimating techniques continuing to manage the project. Software estimating techniques are imperfect. For long-range improvements, we need to compare estimates to actual and use the results to improve the estimating techniques.

Software Cost Estimation Methods

A number of methods have been used to estimate software costs.

Algorithimic Models

These methods provide one or more algorithms which produce a software cost estimate as a function of a number of variables which relate to some software metric (usually its size) and cost drivers.

Expert Judgement

This method involves consulting one or more experts, perhaps with the aid of an expert-consensus mechanism such as the Delphi technique.

Analogy Estimation

This method involves reasoning by analogy with one or more completed projects to relate their actual costs to an estimate of the cost of a similar new project.

Top-down Estimation

An overall cost estimate for the project is derived from global properties of the software product. The total cost is then split up among the various components.

Bottom-up Estimation

Each component of the software job is separately estimated, and the results aggregated to produce an estimate for the overall job.

Parkinson's Principle

A Parkinson principle ('Work expands to fill the available volume") is invoked to equate the cost estimate to the available resources.

Price to Win

The cost estimation developed by this method is equated to the price believed necessary to win the job. The estimated effort depends on the customer's budget and not on the software functionality.

The comparison of different cost estimation methods have been shown in the Table 4.1.

Table 4.1: Comparison of Methods

Method	***Strengths***	***Weaknesses***
Algorithmic model	• Objective, repeatable, analyzable formula • Efficient, good or sensitivity analysis • Objectively calibrated to experience	• Subjective inputs • Assessment of exceptional circumstances • Calibrated to past, not future
Expert judgment	• Assessment of representative ness, interactions, exceptional circumstances	• No better than participants • Biases, incomplete recall
Analogy	• Based on representative experience	• Representative ness of experience
Parkinson & Price to win	• Correlates with some experience • Often gets the contract	• Reinforces poor practice • Generally produces large overruns
Top-down	• System level focus • Efficient	• Less detailed basis • Less stable

Bottom-up	• More detailed basis • More stable • Fosters individual commitment	• May overlook system level costs • Requires more effort

Cost Estimation Guidelines

The following guidelines will help you in cost estimation.

1) Assign the initial estimating task to the final developers.
2) Delay finalizing the initial estimate until the end of a thorough study.
3) Anticipate and control user changes.
4) Monitor the progress of the proposed project.
5) Evaluate proposed project progress by using independent auditors.
6) Use the estimate to evaluate project personnel.
7) Computing management should carefully and approve the cost estimate.
8) Rely on documented facts, standards, and simple arithmetic formulas rather than guessing, intuition, personal memory, and complex formulas.
9) Don't rely on cost estimating software for an accurate estimate.

Self Assessment Questions

1. Cost in a project includes software, hardware and human resources. (True / False)
2. Most cost estimates are determined in terms of __________.
3. ‘Work expands to fill the available volume’ is __________ principle.
    a) Parkinson’s
    b) James’s
    c) John’s
    d) None of the above

4.3 COCOMO Model

The ***COnstructive COst MOdel (COCOMO)*** is the most widely used software estimation model in the world. It was developed by Barry Boehm of TRW and first published in his book Software Engineering Economics in 1981. The COCOMO model predicts the effort and duration of a project based on inputs relating to the size of the resulting systems and a number of "cost drives" that affect productivity. The most fundamental calculation in the COCOMO model is the use of the Effort Equation (Equation 1) to estimate the number of Person-Months (PM) required developing a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity.

$PM = C \times (KDSI)^n$ ——— Equation 1

Where PM = number of person-month (=152 working hours),
C = a constant,
KDSI = thousands of "delivered source instructions" (DSI) and
n = a constant.

COCOMO is defined in terms of three different models: the **Basic model**, the **Intermediate model**, and the **detailed model**. The more complex models account for more factors that influence software projects, and make more accurate estimates. In the COCOMO model, one of the most important factors contributing to a project's duration and cost is the Development Mode. Every project is considered to be developed in one of three modes.

Organic Mode: The project is developed in a familiar, stable environment, and the product is similar to previously developed products. The product is relatively small, and requires little innovation.

Semidetached Mode: The project's characteristics are intermediate between Organic and Embedded.

Embedded Mode: The project is characterized by tight, inflexible constraints and interface requirements. An embedded mode project will require a great deal of innovation.

COCOMO avoids estimating labor costs in dollars because of the large variations between organizations in what is included in labor costs, and because person-months are a more stable quantity than dollars, given current inflation rates and international money fluctuations. In order to convert COCOMO person-month estimates into dollar estimates, the best compromise between simplicity and accuracy is to apply a different average dollar per person-month figure for each major phase, to account for inflation and the differences in salary level of the people required for each phase.

The Basic COCOMO Model

Basic COCOMO model estimates the software development effort using only a single predictor variable (size in DSI) and three software development modes. Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is necessarily limited because of its lack of factors which have a significant influence on software costs.

The Basic COCOMO estimates are within a factor of 1.3 only 29% of the time, and within a factor of 2 only 60% of the time.

The Basic COCOMO Model estimate for annual software maintenance is calculated in terms of **Annual Change Traffic** (**ACT**). **ACT** is the fraction of the software product's source instructions which undergo change during a (typical) year, either through addition or modification.

$ACT = (DSI_{Modified} + DSI_{Added}) / (KDSI)$
$PM_{AM} = 1.0 * ACT \times PM_D$
$FSP_{M} = (PM)_{AM} / 12$

where, **PM_{AM}** is the estimated annual maintenance effort, **PM_D** is the estimated development effort, **FSP_M** is the FSP level required for maintenance. Table 4.2 shows the Effort Equations for the three development modes in Basic COCOMO Model.

Table 4.2: Effort Equations for Three Development Modes in Basic COCOMO Model

Development Mode	Basic Effort Equation	Basic Schedule Equation
Organic:	Effort = 2.4 x $(KDSI)^{1.05}$	TDEV = 2.5 x $(Effort)^{0.38}$
Semidetached:	Effort = 3.0 x $(KDSI)^{1.12}$	TDEV = 2.5 x $(Effort)^{0.35}$
Embedded:	Effort = 3.6 x $(KDSI)^{1.20}$	TDEV = 2.5 x $(Effort)^{0.32}$

The Intermediate COCOMO Model

The Intermediate model use an **Effort Adjustment Factor (EAF)** and slightly different coefficients for the effort equations than the Basic model. You can apply Intermediate COCOMO across the entire software product for easily and roughly cost estimation during the early stage, or apply it at the software product component level for more accurate cost estimation in more detailed stages. The Intermediate COCOMO estimates are within 20% of the actual 68% of the time. There are two primary limitations which may become significant, particularly in detailed cost estimates for large software projects. Table 4.3 shows the Effort Equations for the three development modes in Intermediate COCOMO Model.

Table 4.3: Effort Equations for Three Development Modes in Intermediate COCOMO Model.

Development Mode	**Basic Effort Equation**	**Basic Schedule Equation**
Organic:	**Effort = EAF x 3.2 * $(KDSI)^{1.05}$**	**TDEV = 2.5 x $(Effort)^{0.38}$**
Semidetached:	**Effort = EAF x 3.0 * $(KDSI)^{1.12}$**	**TDEV = 2.5 x $(Effort)^{0.35}$**
Embedded:	**Effort = EAF x 2.8 * $(KDSI)^{1.20}$**	**TDEV = 2.5 x $(Effort)^{0.32}$**

EAF = Effort Adjustment Factor, when the EAF is 1, we call the Effort is nominal.

The Intermediate model produces better results because you supply settings for 15 **Cost Drivers** that determine the effort and duration of software projects. The **Cost Drivers** include factors such as **Product Complexity**, **Programmer Capability**, and **“Use of Software Tools”.** The **Effort adjustment Factor** is simply the product of the Effort Multipliers corresponding to each of the cost drivers for your project. The Intermediate model also produces better results than the Basic model because the system can be divided into "components". DSI values and Cost Drivers can be chosen for individual components, instead of for the system as a whole. COCOMO can estimate the staffing, cost, and duration of each of the components – allowing you to experiment with different development strategies, to find the plan that best suits your needs and resources.

There are many candidate factors to consider in developing a better model for estimating the cost of a software project. There are two principles to reduce the large number of candidate factors to a relatively manageable number of factors for practical cost estimation:

- General Significance: this tends to eliminate factors which are significant only in a relatively small fraction of specialized situations.
- Independence: this tends to eliminate factors which are strongly correlated COCOMO model uses 15 cost drivers based on these principles. These cost drivers are grouped into four categories: software product attributes, computer attributes, personnel attributes, and project attribute.

Introduction to COCOMO II

The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. It has evolved into a more comprehensive estimation model, called *COCOMO II.* COCOMO II is tuned to modern software life cycles. The original COCOMO model has been very successful, but it doesn't apply to newer software development practices as well as it does to traditional practices. COCOMO II targets the software projects of the 1990s and 2000s, and will continue to evolve over the next few years.

The primary objectives of the COCOMO II effort are::

- To develop a software cost and schedule estimation model tuned to the life cycle practices of the 1990's and 2000's.
- To develop software cost database and tool support capabilities for continuous model improvement.

COCOMO II is actually a hierarchy of estimation models that address the following areas:

Application Composition Model: Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.

Early Design Stage Model: Used once requirements have been stabilized and basic software architecture has been established.

Post-architecture-stage Model: Used during the construction of the software. Like all estimation models for software, the COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy: object points, function points, and lines of source code.

Self Assessment Questions

4. COCOMO model was developed by Parkinson. (True / False)
5. COCOMO stands for _______________.
6. In __________ mode the project is characterized by tight, inflexible constraints and interface requirements.
   a) Organic
   b) Embedded
   c) Semidetached
   d) None of the above

4.4 Budgeting

Budgeting in a business sense is the planned allocation of available funds to each department within a company. Budgeting allows executives to control overspending in less productive areas and put more company assets into areas which generate significant income or good public relations. Budgeting is usually handled during meetings with accountants, financial experts and representatives from each department affected by the budgeting. Software budgeting is also the same.

Capital Budgeting

Capital budgeting (or investment appraisal) is the planning process used to determine a firm's long term investments such as new machinery, replacement machinery, new plants, new products, and research and development projects.(EX: IT projects)

Many formal methods are used in capital budgeting, including the techniques such as

- Net present value
- Profitability index
- Internal rate of return
- Modified Internal Rate of Return, and
- Equivalent annuity.

These methods use the incremental cash flows from each potential investment, or *project*. Techniques based on accounting earnings and accounting rules are sometimes used - though economists consider this to be improper – such as the *accounting rate of return,* and "return on investment." Simplified and hybrid methods are used as well, **such as *payback period* and *discounted payback period*.**

1) Net Present Value (NPV)

Net present value (NPV) is a standard method for the financial appraisal of long-term projects. Used for capital budgeting, and widely throughout economics, it measures the excess or shortfall of cash flows, in present value (PV) terms, once financing charges are met. By definition,
NPV = Present value of net cash flows. For its expression, see the formula section below.

Definition: NPV

The net-present-value (NPV) method is a discounted-cash-flow approach to capital budgeting that computes to all expected future cash flows using a minimum desired rate of return.

Formula:

Each cash inflow/outflow is discounted back to its PV. Then they are summed. Therefore

$$\mathrm{NPV} = \sum_{t=1}^{n} \frac{C_t}{(1+r)^t} - C_0$$

Where

t – the time of the cash flow

n – the total time of the project

r – the discount rate

C_t – the net cash flow (the amount of cash) at time t.

C_0 – the capital outlay at the beginning of the investment time ($t = 0$)

In the above formula, choosing an appropriate discount rate is crucial to the NPV calculation. A good practice of choosing the discount rate is to decide the rate which the capital needed for the project could return if invested in an alternative venture. If, for example, the capital required for Project A can earn five percent elsewhere, use this discount rate in the NPV calculation to allow a direct comparison to be made between Project A and the alternative. Obviously, NPV value obtained using variable discount rates with the years of the investment duration is more reflecting to the real situation than that calculated from a constant discount rate for the entire investment duration.

For some professional investors, their investment funds are committed to target a specified rate of return. In such cases, that rate of return should be selected as the discount rate for the NPV calculation. In this way, a direct comparison can be made between the profitability of the project and the desired rate of return.

The rate used to discount future cash flows to their present values is a key input of this process. Most firms have a well defined policy regarding their capital structure. So the weighted average cost of capital (after tax) is appropriate for use with all projects. Alternately, higher discount rates can be used for more risky projects. Another method is to apply higher discount rates to cash flows occurring further along the time span, to reflect the yield curve premium for long-term debt.

With a particular project, if C_t is a positive value, the project is in the status of cash inflow in the time of t. If C_t is a negative value, the project is in the status of cash outflow in the time of t. Appropriately risked projects with a positive NPV should be accepted. This does not necessarily mean that they should be undertaken since NPV at the cost of capital may not account for opportunity cost, i.e. comparison with other available investments. In financial theory, if there is a choice between two mutually exclusive alternatives, the one yielding the higher NPV should be selected.

Table 4.4 sums up the NPV's various situations.

Table 4.4: Decision Making based on NPV's Value

If...	***It means...***	***Then...***
NPV > 0	the investment would add value to the firm	the project should be accepted
NPV < 0	the investment would subtract value from the firm	the project should be rejected
NPV = 0	the investment would neither gain nor lose value for the firm	We should be indifferent in the decision whether to accept or reject the project. This project adds no monetary value. Decision should be based on other criteria, e.g. strategic positioning or other factors not explicitly included in the calculation.

Example:

X Corporation must decide whether to introduce a new product line. The new product will have startup costs, operational costs, and incoming cash flows over six years. This project will have an immediate (t=0) cash outflow of \$100,000 (which might include machinery, and employee training costs). Other cash outflows for years 1-6 are expected to be \$5,000 per year. Cash inflows are expected to be \$30,000 per year for years 1-6.

All cash flows are after-tax, and there are no cash flows expected after year 6. The required rate of return is 10%. The present value (PV) can be calculated for each year:

T=0 - \$100,000 / 1.10^0 = -\$100,000 PV.
T=1 (\$30,000 - \$5,000)/ 1.10^1 = \$22,727 PV.
T=2 (\$30,000 - \$5,000)/ 1.10^2 = \$20,661 PV.
T=3 (\$30,000 - \$5,000)/ 1.10^3 = \$18,783 PV.
T=4 (\$30,000 - \$5,000)/ 1.10^4 = \$17,075 PV.
T=5 (\$30,000 - \$5,000)/ 1.10^5 = \$15,523 PV.
T=6 (\$30,000 - \$5,000)/ 1.10^6 = \$14,112 PV.

The sum of all these present values is the **net present value**, which equals \$8,881. Since the NPV is greater than zero, the corporation should invest in the project.

2) Rate of return (ROR)
In finance, rate of return (ROR) or return on investment (ROI), or sometimes just return, is the ratio of money gained or lost on an investment relative to the amount of money invested. The amount of money gained or lost may be referred to as interest, profit/loss, gain/loss, or net income/loss. The money invested may be referred to as the asset, capital, principal, or the cost basis of the investment.

ROI is also known as **rate of profit**. **Return** can also refer to the monetary amount of gain or loss. ROI is the return on a past or current investment, or the estimated return on a future investment. ROI is usually given as a percent rather than decimal value.

ROI stands for "return on investment," one of the great mysteries of online advertising, and indeed, advertising in general. ROI is trying to find out what the end of result of the expenditure (in this case, an ad campaign) is. A lot depends on the goal of the campaign, building brand awareness, increasing sales, etc. Early attempts at determining ROI in Internet advertising relied heavily on the click-rate of an ad.

ROI does not indicate how long an investment is held. However, ROI is most often stated as an annual or annualized rate of return, and it is most often stated for a calendar or fiscal year. In this article, “ROI” indicates an *annual or annualized rate of return*, unless otherwise noted.

ROI is used to compare returns on investments where the money gained or lost – or the money invested – are not easily compared using monetary values. For instance, a \$1,000 investment that earns \$50 in interest obviously generates more cash than a $100 investment that earns $20 in interest, but the $100 investment earns a higher return on investment.

- $50/$1,000 = 5% ROI
- $20/$100 = 20% ROI

Since rates of return are percentages, negative rates cannot be averaged with positive rates for purposes of calculating monetary returns. However, it is common practice in finance to estimate monetary returns by averaging periodic rates of return; these estimations are most useful when the averaged periodic returns are all positive, all negative, or have low variances.

3. Payback Model

Payback time, or *payback period,* is the time it will take to recoup, in the form of cash inflows from operations, the initial dollars invested in a project.

$$P = I \div O$$

Where I is the cash inflow and O is the cash outflow.

Example:

Assume that $12,000 is spent for a machine with an estimated useful life of 8 years.

Annual savings of $4,000 in cash outflows are expected from operations.

What is the payback period?

P = I ÷ O = $12,000 ÷ $4,000 = 3 years

Self Assessment Questions

7. Budgeting in a business sense is the planned allocation of available funds to each department within a company. (True / False)
8. ROI stands for _______________.
9. _______________ is a standard method for the financial appraisal of long-term projects.

4.5 Summary

Let's summarize the important points covered in this unit:

- The software project planner must estimate three things before a project begins: how long it will take, how much effort will be required, and how many people will be involved. In addition, the planner must predict the resources (hardware and software) that will be required and the risk involved.

- Accurate project estimates generally use at least two of the three techniques just noted.
- By comparing and reconciling estimates derived using different techniques, the planner is more likely to derive an accurate estimate.
- In this unit we have highlighted the COCOMO cost estimation models. It is the model used so frequently in the industry.
- We have discussed the estimation of software budgeting and its three methods like NPV, payback methods and ROI.

4.6 Terminal Questions

1. What is cost estimation? Explain different Cost Estimation methods.
2. Compare the Cost Estimation Methods in detail.
3. What is COCOMO? Explain COCOMO model in detail.
4. What is COCOMO II? Explain in brief.
5. What is budgeting? Explain the following budgeting techniques
   a) NPV b) ROI c) Payback Models

4.7 Answers

Self Assessment Questions

1. True
2. Person-Months (PM)
3. a) Parkinson's
4. False
5. COnstructive COst MOdel
6. b) Embedded
7. True
8. Return On Investment
9. Net Present Value (NPV)

Terminal Questions

1. The project cost includes the requirements for software, hardware and human resources. The bulk of the cost of software development is due to the human resources needed, and most cost estimation procedures focus on this aspect. Most cost estimates are determined in terms of person-months (PM). (Refer Section 4.2)

2. A number of methods have been used to estimate software costs such as Algorithmic Models, Expert Judgment, Analogy Estimation etc. (Refer Section 4.2)
3. COCOMO stands for COnstructive COst MOdel, which is the most widely used software estimation model in the world. It was developed by Barry Boehm of TRW and first published in his book Software Engineering Economics in 1981. The COCOMO model predicts the effort and duration of a project based on inputs relating to the size of the resulting systems and a number of "cost drives" that affect productivity. (Refer Section 4.3)
4. The original COCOMO model became one of the most widely used and has evolved into a more comprehensive estimation model, called COCOMO II. COCOMO II is tuned to modern software life cycles. (Refer Section 4.3)
5. Budgeting in a business sense is the planned allocation of available funds to each department within a company. Budgeting allows executives to control overspending in less productive areas and put more company assets into areas which generate significant income or good public relations. (Refer Section 4.4)