Lecture 05

Evaluating the Proposal

- Once the proposals are generated, they are evaluated
- It is necessary to check
  1. Technical Feasibility
  2. Operational Feasibility
  3. Economical Feasibility
- Risk Analysis
- Identify the areas (uncertain) of high risks for which uncertainty about getting successful outcome
  - Identify and Special attention take

Economical Feasibility

- **Cost Benefit Analysis**
- Cost : Place actual money value against any purchase or activity needed to implement the project
- Benefit : Place money value against any benefits that will accure from a new system created by the project
- Two steps
  1. Estimation of Costs and Benefits
  2. Determine whether a project is worthwhile or not

Estimation of Costs and Benefits

- Goal is to produce a list of what is required to implement the project
- List of the new system benefits
- Evaluating Costs
  - Tangible Items (Direct value can be attached)
  - Intangible Items (Direct value can’t be attached , guess)
- Evaluating Benefits
  - Tangible Items (Direct value can be attached)
  - Intangible Items (Direct value can’t be attached , guess)

Cost of Tangible Items

1. Equipment costs for the new system
   - Computing equipments costs, accommodation costs, furniture costs etc.
2. Personnel Costs – Analyst, Designers, Programmers etc.
3. Material Costs – Stationary, manuals, documents etc.
4. Conversion Cost – New forms, Procedures etc.
5. Training Costs – Users, developers etc.
6. Other Costs – Consultants costs, travel, management overhead etc.

- Value can’t be precisely determined & subjective judgment
  - How much save to complete the project earlier
  - Provide new information to the decision makers

Benefits of Tangible & Intangible Items

1. Measure by actual value
   - Reduce of production cost
   - Reduce the processing cost

2. Can’t be measure (only subjective judgement)
   - Increase the sales in the market
   - Wider market of marketing data
   - Maintain good business image

Determine whether a project is worthwhile or not

1. Payback method (Forward)
2. Present Value method(backward)

- **Payback method (Forward)**
  - Define time required to recover the initial cost
  - We know how much a project will cost to start
  - Know cost and benefits for each succeeding year
- **Example**
  - Cost at start is $50,000 (project tenure is 5 years)
  - Benefits - $10,000 at the end of 1st year
    - $20,000 at the end of 2nd year
    - $30,000 at the end of 3rd year
    - $10,000 at the end of 4th year
    - $5,000 at the end of 5th year

payback period is between 2nd and 3rd year

Determine whether a project is worthwhile or not (Cont.)

- We assume for banking system, and rate of interest is 10% per annum
- **Present Value method(backward)**
- Determine how much money it is worthwhile for investing now in order to receive a given return in some years time
- Project benefits are estimated for each year from today
- Then compute the present value of savings
- If the present cost exceeds the present value, then it is not worthwhile

Determine whether a project is worthwhile or not (Cont.)

- Example
  - At t=0, Initial investment is $50,000
  - At t=1, Benefit = $10,000 then the amount deposited in the 0th year= $10000/(1+10/100)
  - At t=2, Benefit = $20,000 then the amount deposited in the 0th year= $20000/ $(1+10/100)^2$
  - At t=3, Benefit = $30,000 then the amount deposited in the 0th year= $30000/$(1+10/100)^3$
  - At t=4, Benefit = $10,000 then the amount deposited in the 0th year= $10000/$(1+10/100)^4$
  - At t=5, Benefit = $5,000 then the amount deposited in the 0th year= $,5000/ $(1+10/100)^5$
- Sum of the present value > Initial investment

COCOMO (COnstructive COst MOdel) Model

- COCOMO I
- Barry Boehm introduce it
- Hierarchy of software estimation model
- Model takes the following form :
  1. Model1(Basic COCOMO) : is a static single valued model that computes software development effort (and cost) as a function of program size expressed in estimated line of code
  2. Model2(Intermediate COCOMO) :computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessments of product, hardware, personnel and project attributes

COCOMO (COnstructive COst MOdel) Model

- Model3(Advanced COCOMO) :incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design etc.) of the software engineering process

- COCOMO may applied to 3 classes of the software projects :
  1. Organic mode: Small, simple s/w project
     - Ex. Tharmal analysis program
  2. Semi-detached mode: Intermediate size and complexity
     - Ex. Transaction processing
  3. Embedded mode:

COCOMO (COnstructive COst MOdel) Model

- Basic COCOMO equation take the form :
- Effort= $E = a_b(KLOC)\exp(b_b)$ person-months
- Duration= $C_b(E)\exp(d_b)$ months
- Cost driver attributes:

1. Product attributes

   I. Required s/w reliability
   II. Size of application database
   III. Complexity of the product

COCOMO (COnstructive COst MOdel) Model

2. H/W attributes
   - I. Run-time performance constraints
   - II. Memory constraints
   - III. Volatility of the virtual m/c environment
   - IV. Required turnaround time
3. Personnel attributes
   - I. Analyst capability
   - II. S/w engineer capability
   - III. Applications experience
   - IV. Virtual m/c experience
   - V. Programming Language experience
4. Product attributes
   - I. Use of s/w tools
   - II. Application of S/w engineering methods
   - III. Required development schedule

Attributes value (Table by Boehm)

S/w project	a_a	b_b	c_c	d_d
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Attributes value (Table by Boehm)

S/w project	a_i	b_i
Organic	3.2	1.05
Semidetached	3.0	1.12
Embedded	2.8	1.20

COCOMO (Cont.)

- Each of the 15 attributes is rated on a 6-point scale that ranges from **very low** to **extra high**
- Based on the rating **effort multiplier** is determined from table published by Boehm
- Product of effort multipliers results is an effort adjustment factor (EAF) ranges from 0.9 to 1.4
- Intermediate COCOMO Equation takes the form :

$$E = a_i (KLOC) \exp(b_i) \times EAF$$

COCOMO II

- Hierarchy of estimation models that address the following areas :

  1. **Application composition model** : used during early stage of s/w engg., when prototyping of user interfaces, consideration of s/w and system interaction, assessment of performance and evaluation of technology maturity are paramount.
  2. **Early design stage model** : used once requirements have been stabilized and basic s/w architecture has been established
  3. **Post- architecture stage model** : used during the construction of the s/w

COCOMO II (Cont.)

- Sizing information :
  i. Object points
  ii. Function points
  iii. Lines of source code

- The COCOMO II in application composition modes uses object points

- An indirect s/w measure that is computed using counts of the number of (1) **Screens** (at the user interface), (2) **Reports** and (3) **Components** (likely to be required to build the application)

COCOMO II (Cont.)

- Each object instance (screen or report) is classified into one of three complexity levels

Object type	Complexity weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
Components			10

- **Object point count** is determine by multiplying the original number of object instances by the weighting factor in the table
- Summing to obtain total object point
- When component based development or general s/w reuse is to be applied, the % of reuse is estimated and the object point count is adjusted

COCOMO II (Cont.)

- New Object Point, NOP=(Object Point) X [(100-%reuse)/100]
- Productivity Rate, PROD=NOP/Person-month
- Estimated Effort, E= NOP/PROD
- Productivity rate :

Developer's Experience/Capacity	Very low	Low	Nominal	High	Very High
Environment Maturity/Capacity	Very low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Function Point Analysis

- Its application in non-traditional computing situations
- Searching for a metric that is applicable for a broad range of software environments
- The metric should be technology independent and support the need for estimating, project management, measuring quality and gathering requirements
- A reliable method for measuring the size of computer software. In addition to measuring output
- Function Point Analysis is extremely useful in estimating projects, managing change of scope, measuring productivity, and communicating functional requirements.

Function Point Analysis

- There have been many misconceptions regarding the appropriateness of Function Point Analysis in evaluating emerging environments such as real time embedded code and Object Oriented programming.

- Since function points express the resulting work-product in terms of functionality as seen from the user's perspective, the tools and technologies used to deliver it are independent.

Introduction to Function Point Analysis

- One of the initial design criteria for function points was to provide a mechanism that both software developers and users could utilize to define functional requirements
- One of the primary goals of Function Point Analysis is to evaluate a system's capabilities from a user's point of view
- To achieve this goal, the analysis is based upon the various ways users interact with computerized systems
- From a user's perspective a system assists them in doing their job by providing five (5) basic functions
- Two of these address the data requirements of an end user and are referred to as Data Functions. The remaining three address the user's need to access data and are referred to as Transactional Functions.

The Five Components of Function Points

- **Data Functions**
- Internal Logical Files
- External Interface Files
- **Transactional Functions**
- External Inputs
- External Outputs
- External Inquiries

Internal Logical Files(ILF)

- The first data function allows users to utilize data they are responsible for maintaining
- Logical groupings of data in a system, maintained by an end user, are referred to as Internal Logical Files (ILF).
- For example,
  - A pilot may enter navigational data through a display in the cockpit prior to departure. The data is stored in a file for use and can be modified during the mission. Therefore the pilot is responsible for maintaining the file that contains the navigational information.

External Interface Files(EIF)

- The second Data Function a system provides an end user is also related to logical groupings of data
- In this case the user is not responsible for maintaining the data
- The data resides in another system and is maintained by another user or system
- The user of the system being counted requires this data for reference purposes only
- Groupings of data from another system that are used only for reference purposes are defined as External Interface Files (EIF).
- For example,
  - it may be necessary for a pilot to reference position data

External Interface Files(EIF)

- For example,
  - It may be necessary for a pilot to reference position data from a satellite or ground-based facility during flight. The pilot does not have the responsibility for updating data at these sites but must reference it during the flight

External Input (EI)

- The first Transactional Function allows a user to maintain Internal Logical Files (ILFs) through the ability to add, change and delete the data

- An External Input gives the user the capability to maintain the data in ILF's through adding, changing and deleting its contents

- For example,
  - A pilot can add, change and delete navigational information prior to and during the mission. In this case the pilot is utilizing a transaction referred to as an External Input (EI).

External Output (EO)

- The next Transactional Function gives the user the ability to produce outputs

- For example,
  - A pilot has the ability to separately display ground speed, true air speed and calibrated air speed. The results displayed are derived using data that is maintained and data that is referenced.
  - In function point terminology the resulting display is called an External Output (EO).

External Inquiries (EQ)

- The final capability provided to users through a computerized system addresses the requirement to select and display specific data from files
- To accomplish this a user inputs selection information that is used to retrieve data that meets the specific criteria.
- In this situation there is no manipulation of the data. It is a direct retrieval of information contained on the files
- These transactions are referred to as External Inquiries (EQ).
- For example,
  - if a pilot displays terrain clearance data that was previously set, the resulting output is the direct retrieval of stored information

Adjustment Factors

- Two adjustment factors that need to be considered in Function Point Analysis
  - Functional Complexity
  - Value Adjustment Factor
- Functional Complexity :
  - considers the Functional Complexity for each unique function
  - Functional Complexity is determined based on the combination of data groupings and data elements of a particular function
  - The number of data elements and unique groupings are counted and compared to a complexity matrix that will rate the function as low, average or high complexity

Functional Complexity

- Each of the five functional components (ILF, EIF, EI, EO and EQ) has its own unique complexity matrix
- The following is the complexity matrix for External Outputs

	1-5 DETs	6 - 19 DETs	20+ DETs
0 or 1 FTRs	L	L	A
2 or 3 FTRs	L	A	H
4+ FTRs	A	H	H

Complexity	UFP
L (Low)	4
A (Average)	5
H (High)	7

Example

- Using the examples given above and their appropriate complexity matrices, the function point count for these functions would be:

Example

Function Name	Function Type	Record Element Type	Data Element Type	File Types Referenced	Unadjusted FPs
Navigational data	ILF	3	36	n/a	10
Positional data	EIF	1	3	n/a	5
Navigational data - add	EI	n/a	36	1	4
Navigational data - change	EI	n/a	36	1	4
Navigational data - delete	EI	n/a	3	1	3
Ground speed display	EO	n/a	20	3	7
Air speed display	EO	n/a	20	3	7
Calibrated air speed display	EO	n/a	20	3	7
Terrain clearance display	EQ	n/a	1	1	3
Total unadjusted count					50 **UFPs**

Value Adjustment Factor

- The Unadjusted Function Point count is multiplied by the second adjustment factor called the Value Adjustment Factor
- This factor considers the system's technical and operational characteristics and is calculated by answering 14 questions
- The factors are:
  1. Data Communications
  - The data and control information used in the application are sent or received over communication

  2. Distributed Data Processing

  Distributed data or processing functions are a characteristic of the application within the application boundary.

Value Adjustment Factor

3. Performance

- Application performance objectives, stated or approved by the user, in either response or throughput, influence (or will influence) the design, development, installation and support of the application.

4. Heavily Used Configuration

- A heavily used operational configuration, requiring special design considerations, is a characteristic of the application

Value Adjustment Factor

5. Transaction Rate

- The transaction rate is high and influences the design, development, installation and support.

6. On-line Data Entry

- On-line data entry and control information functions are provided in the application

7. End -User Efficiency

- The on-line functions provided emphasize a design for end-user efficiency

Value Adjustment Factor

8. On-line Update

- The application provides on-line update for the internal logical files.

9. Complex Processing

- Complex processing is a characteristic of the application

10. Reusability

- The application and the code in the application have been specifically designed, developed and supported to be usable in other applications

Value Adjustment Factor

11. Installation Ease

- Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools were provided and tested during the system test phase

12. Operational Ease

- Operational ease is a characteristic of the application. Effective start-up, backup and recovery procedures were provided and tested during the system test phase

Value Adjustment Factor

13. Multiple Sites

- The application has been specifically designed, developed and supported to be installed at multiple sites for multiple organizations

14. Facilitate Change

- The application has been specifically designed, developed and supported to facilitate change

Each of these factors is scored based on their influence on the system being counted. The resulting score will increase or decrease the Unadjusted Function Point count by 35%. This calculation provides us with the Adjusted Function Point count.

An Approach to Counting Function Points

- Function point counting can be accomplished with minimal documentation. However, the accuracy and efficiency of the counting improves with appropriate documentation.

- Examples of appropriate documentation are:
  - Design specifications
  - Display designs
  - Data requirements (Internal and External)
  - Description of user interfaces

Benefits of Function Point Analysis

- Benefits including:
  - improved project estimating;
  - understanding project and maintenance productivity;
  - managing changing project requirements;

  and

  - gathering user requirements

Roger Heller, Vice President Q/P Management Group, Inc., wrote this article. The article was originally published in the STC Crosstalk publication.

Problems

1. Using COCOMO I
   - Determine the Effort to develop of a Software product
   - Determine the Duration to develop of a Software product
   - Determine the Number of People engaged to develop of a Software product.

- Input of your program is Lines of Code and Effort Adjustment Factor.
- Also determine the type of project (i) Organic, (ii) Semi-detached and (iii) Embedded.

Problems

- Using COCOMO II
  - Determine the Object Point to develop of a Software product
  - Determine the New Object Point to develop of a Software product
  - Determine the Effort to develop of a Software product
  - Determine the Number of People engaged to develop Software product, if Duration of development of the software is 5 years.

- If the software consist of 10 Screens, 4 Reports and 15 3GL Components. Assume component based development and 60% reuse is applied.