SOFTWARE ENGINEERING

Week 4

Software Project Management

Yard. Doç. Dr. A. Cüneyd TANTUĞ
Yard. Doç. Dr. Tolga OVATMAN
Istanbul Technical University
Computer Engineering Department

Agenda

1. Project Management Concepts
2. Project Planning
3. Estimation
4. Process Metrics
5. Quality Management

1. Project Management Concepts
2. Estimation
3. Planning
4. Management

Project Management Concepts

4.1

Software Project Management

- Concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

Success Criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer's expectations.
- Maintain a happy and well-functioning development team.

4P's

1. People
the most important element of a successful project

2. Product
the software to be built

3. Process
the set of framework activities and software engineering tasks to get the job done

4. Project
all work required to make the product a reality

Software Project Management

- What to Estimate?
  - Effort
  - Time
- What to Plan?
  - People
  - Tasks
  - Time
- How to manage?
  - Risk
  - Quality - What to Measure?
    - Process Metrics
    - OO Metrics
  - Deliveries

1. Project Management Concepts
2. Estimation
3. Planning
4. Management

Estimation

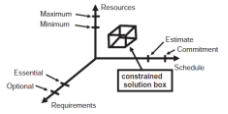
4.2

Estimation

- Organizations need to make software effort and cost estimates.
- Estimation of resources, cost, and schedule for a software engineering effort requires
  - experience
  - access to good historical information (metrics
  - the courage to commit to quantitative predictions when qualitative information is all that exists
- Estimation carries inherent risk and this risk leads to uncertainty.
- There is no simple way to make an accurate estimate of the effort required to develop a software system.
- Initial estimates are usually based on inadequate information about user requirements.

Estimation

- In theory anything is negotiable, in practice
  - Time: Usually customer has a strict deadline
  - Cost: The budget of the project is almost fixed due to competition
  - Quality: There is a certain amount of quality induced according to the type of the project.

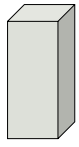


- Man-month= Man-month is a hypothetical unit of work representing the work done by one person in one month
- Brooks's law: Adding manpower to a late software project makes it later.

Estimation

Coding

40-50%



Testing

30-40%

WRONG!!

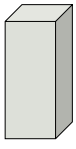
Analysis and Design

15-20%

Estimation

Analysis and Design

40-50%

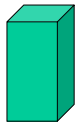


Coding

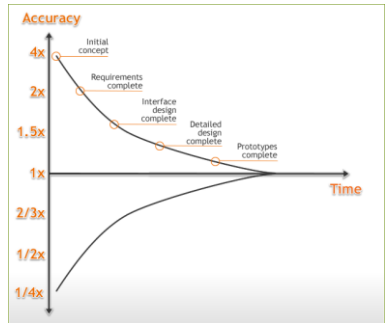
15-20%

Testing

30-40%



Estimation



Estimation Techniques

1. Estimation by analogy
   The cost of a new project is estimated by analogy with similar completed projects.
2. Expert judgement
   Several experts on the proposed software application domain are consulted. They each estimate the project cost. The estimation process iterates until an agreed estimate is reached.
3. Algorithmic cost modelling
   A model based on historical cost information that relates some software metric (usually its size) to the project cost is used. COCOMO is an example of algorithmic cost modelling.
4. Parkinson's Law
   Parkinson's Law states that work expands to fill the time available. The cost is determined by available resources rather than by objective assessment.
5. Pricing to win
   The estimated effort depends on the customer's budget and not on the software functionality.

Algorithmic Cost Modelling

- Cost (i.e. Effort) is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers:

  Effort = A * (Size)^E * EAF

  where
  - A is an organisation-dependent constant,
  - E reflects the disproportionate effort for large projects,
  - EAF (Effort Adjustment Factor) 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, E and EAF.

Function Points

- Function Point is a measurement estimation method for a software from the functionality perspective.
  - Functionality as viewed from the user's perspective.
- This estimation method is based on a empirical model which is mainly independent of programming language.
- Function Points can be used for several purposes:
  - Estimating the FP of a new planned software.
  - Assessing the retail value of an existing software.
- Developed by Allan J. Albrecht, and standardized by the "International Function Point User Group". (www.ifpug.org)

FP — Project Types

Project Type	FP Purpose
Development Project	Measures the functions that will be provided to the users in a new application.
Enhancement Project	Measures the modifications to an existing application.
Application Assesment	Measures the functionality provided to users in an existing application.

FP — Counting Steps

1. Count Data Functions and Transactional Functions
2. Calculate Unadjusted Function Point
3. Calculate Value Adjustment Factor (VAF)
4. Calculate Adjusted Function Point

FP - Equations

$$VAF = \left(\sum_{i=1}^{14} GSC_i * 0.01 \right) + 0.65$$

Adjusted FP = (Unadjusted FP) * VAF

VAF: Value Adjustment Factor

GSC: General System Characteristic factors

FP - Value Adjustment Factor (VAF)

- VAF consists of 14 General System Characteristics (GSC) such as data communications, response times, end user efficiency, multiple sites, flexibility,etc.
- Each GSC can be an integer value between 0 and 5.
- Min VAF = (14*0)*0.01 + 0.65 = 0.65
- Max VAF = (14*5)*0.01 + 0.65 = 1.35
- The overall effect of VAF can vary in range from 0.65 (when all GSCs are low) to 1.35 (when all GSCs are high), so its overall adjustment effect could be **± 35 %.**
- We will study the GSCs and VAF later.

FP – Calculating Unadjusted Function Points

Type of Component	Complexity of Components			
	Low	Average	High	Total
EI	☐ x 3	☐ x 4	☐ x 6	= ☐
EO	☐ x 4	☐ x 5	☐ x 7	= ☐
EQ	☐ x 3	☐ x 4	☐ x 6	= ☐
ILF	☐ x 7	☐ x10	☐ x15	= ☐
EIF	☐ x 5	☐ x 7	☐ x10	= ☐

Unadjusted Function Points = ☐

FP – Standard Functions

- In counting FPs there are five standard "functions" that you count.

Data Functions:

- Internal Logical Files (ILF)
- External Interface Files (EIF)

Transactional Functions:

- External Inputs (EI)
- External Outputs (EO)
- External Inquiries (EQ)

FP – Data Functions

- **ILF:**
  - Files controlled by the program.
  - Each data file (or database table) is counted.
  - Examples
    - ILF refers to logical group of data files maintained by the application such as **Employee file.**
    - Note the inside application data is updated and not any external data.
- **EIF:**
  - Files controlled by other programs.
  - All machine readable interfaces (import/export data file) that are used to transmit information to another system are counted.
  - Examples
    - EIF refers to logical group of data referenced but not maintained internally such as an **Currency file.**

FP - Transactional Functions

EI:

- Each user input (screens, forms, dialog boxes, controls etc.) that provides distinct data to the software is counted.
- Individual data items within a data-entry screen are not counted separately.
- Inputs should be distinguished from inquiries, which are counted separately.

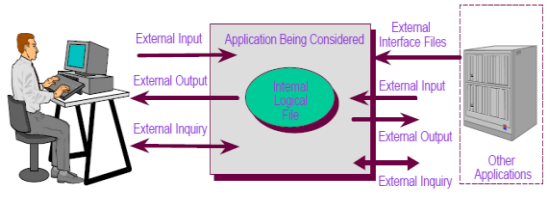
EO:

- Each user output that provides information to the user is counted.
- In this context, output refers to reports, screens, graphs, error messages, etc.
- Individual data items within a report are not counted separately.

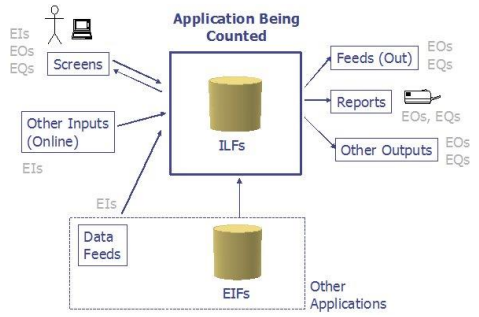
EQ:

- An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output.
- Each distinct inquiry is counted.

FP – System Example 1



FP – System Example 2



FP – Logical File (ILF) example

- ILF refers to logical group of data files maintained by the application such as **Employee file**.
- Note the inside application data is updated and not any external data.

FP – External Interface File (EIF) example

- EIF refers to logical group of data referenced but not maintained internally such as an **Currency file.**

FP – External Input (EI) examples

- The basic EI is from **user screens (for data entry / editing).**
- Users should have interface through which they can maintain the data files in ILF through **Add, Delete, Update** menu selections (transactional functions).
  - Passing control data such as **menu selections** into the application is considered as EI.

FP – External Output (EO) examples

- EO usually refers to **reports** which contain **derived data** from the internal files (ILF) such as calculated totals**.**
  - Formatted data sent out of application with added value.
  - For example, list of students and the calculated grade average in a class.
- EO can also refer to **screens** which contain the followings:
  - Displaying derived data from the internal files (ILF).
  - Prefilling a listbox with **hardcoded data** in the program.
- Outputs sent to external systems are EO.
  - For example, your application generates CSV (comma separated values) files.
  - These files are then used by some external application to update the external application tables (EIF).

FP – External Query (EQ) examples

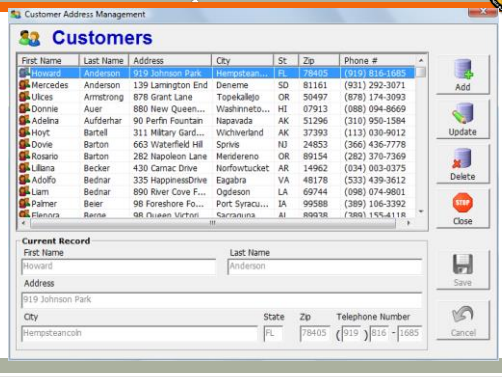
- EQ functions will be mainly **reports** which **do not contain derived data**.
- Formatted data sent out of application without added value. (For example, only the list of students in a class)
- Reports may have input criteria, so that can be another EQ.
- Also **search screens** are EQ.
- Prefilling a listbox from ILF is considered as EQ, because this is an **implied inquiry**.
- Note EQ functions don't update any ILF or EIF. They only fetch data for display.

FP Example : "Customer" Application

- In this simple example, we will evaluate a Customer GUI (Graphical User Interface) application.
- The database has only one table, which contains customer information.
- The GUI program will allow the user add, delete, and update the records.

31

FP Example :"Customer" Screen



32

FP Example : Counting the Functions (1)

- There is 1 Internal Logical File
  1) Customer table

Customer : Table

	Field Name	Data Type
🔑	CustID	Number
	LastName	Text
	FirstName	Text
	Address	Text
	City	Text
	State	Text
	Zip	Text
	PhoneNumber	Text

33

FP Example : Counting the Functions (2)

- There are 3 External Inputs
  1) "Customer selection" area in the screen for selecting a customer
  2) "Current record" area in the screen for entering / editing customer data
  3) All transaction buttons (ADD, UPDATE, DELETE, etc.)

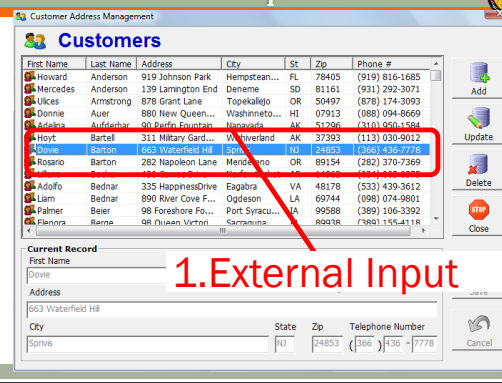
34

FP Example : Counting the Functions (3)

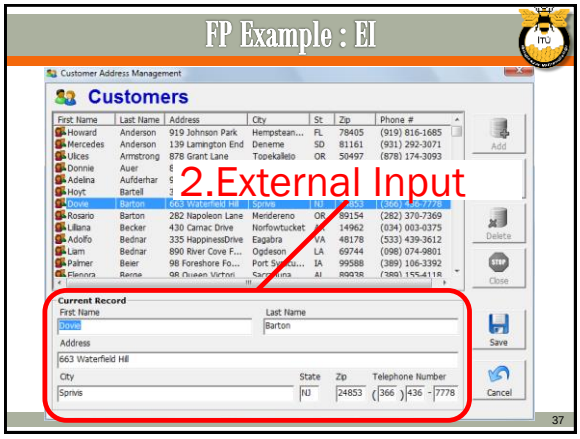
- There are 2 External Outputs
  1) The output listbox of all customer names and addresses (always read only)
  2) "Current record" area in the screen for displaying currently selected customer data (read only in display mode)
- There is 1 External Query
  1) The confirmation dialog boxes for "Delete" and "Save" buttons

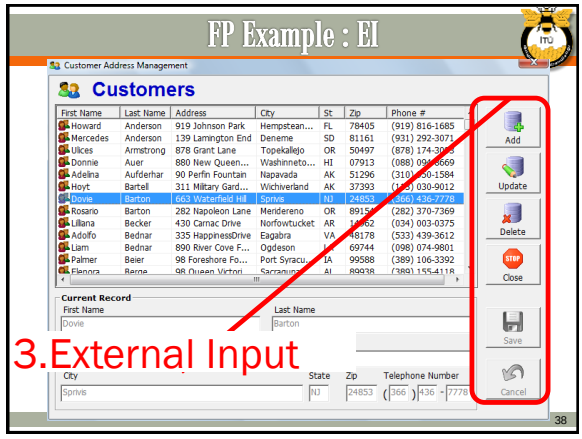
35

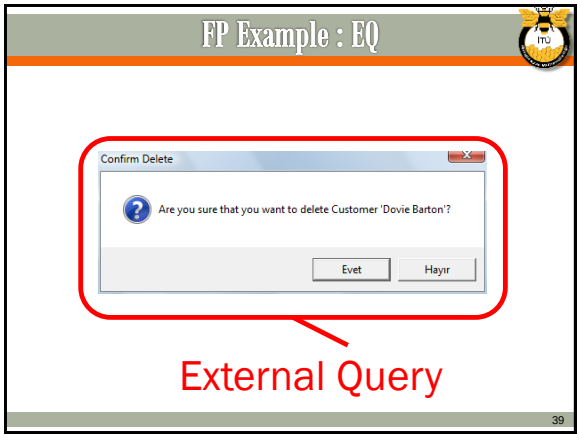
FP Example : EI

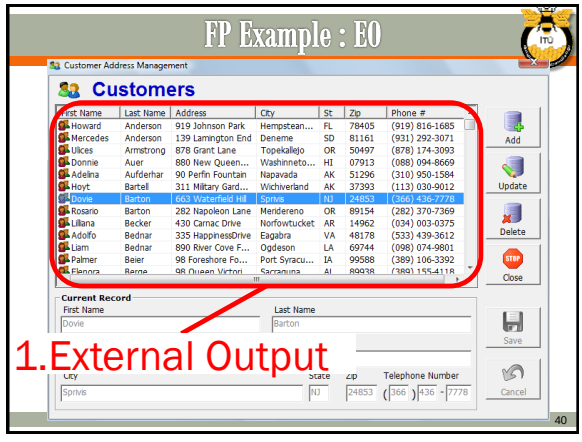


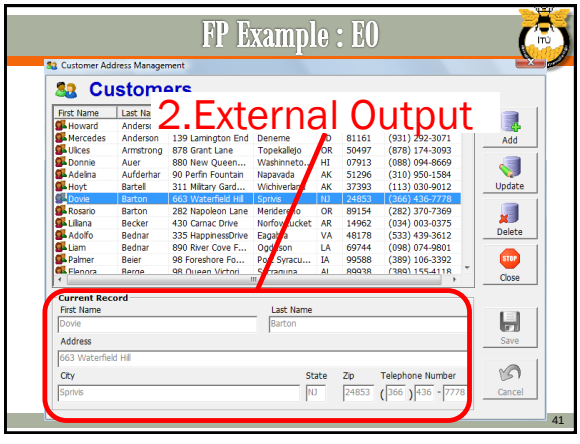
36











Unadjusted Function Points

Type of Component	Complexity of Components			
	Low	Average	High	Total
EI		3 x4		= 12
EO	2 x4			= 8
EQ	1 x3			= 3
ILF	1 x7			= 7

Unadjusted Function Points = 30

42

FP - General System Characteristics (GSC)

- This is a very important part in Function Points.
- GSC factors can affect the software a lot and also the cost of it.
- GSC gives us something called as VAF (Value Added Factor).
- There are 14 GSC points considered to come out with VAF.
- Each GSC can have a rating between 0 and 5.

43

FP - General System Characteristics

1. Data Communication
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. Online data entry
7. End user efficiency
8. Online update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

Description	Rating
No influence	0
Incidental	1
Moderate	2
Average	3
Significant	4
Essential	5

44

FP - GSC Definitions (1)

1. **Data communications:** How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2. **Distributed data processing:** How are distributed data and processing functions handled?
3. **Performance:** Did the user require response time or throughput?
4. **Heavily used configuration:** How heavily used is the current hardware platform where the application will be executed?
5. **Transaction rate:** How frequently are transactions executed; daily, weekly, monthly, etc.?
6. **On-Line data entry:** What percentage of the information is entered On-Line?
7. **End-user efficiency:** Was the application designed for end-user efficiency?

45

FP - GSC Definitions (2)

8. **On-Line update:** How many ILFs are updated by On-Line transaction?
9. **Complex processing:** Does the application have extensive logical or mathematical processing?
10. **Reusability:** Was the application developed to meet one or many user's needs?
11. **Installation ease:** How difficult is conversion and installation?
12. **Operational ease:** How effective and/or automated are start-up, back up, and recovery procedures?
13. **Multiple sites:** Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14. **Facilitate change:** Was the application specifically designed, developed, and supported to facilitate change?

46

FP - GSCs for "Customer" Example

GSC	Value (0-5)
1. Data communications	0
2. Distributed data processing	0
3. Performance	3
4. Heavily used configuration	3
5. Transaction rate	4
6. On-Line data entry	5
7. End-user efficiency	4
8. On-Line update	1
9. Complex processing	0
10. Reusability	0
11. Installation ease	0
12. Operational ease	2
13. Multiple sites	0
14. Facilitate change	0
Total =	22

47

FP - Adjusted FP for "Customer" Example

VAF = 0.65 + (Sum of all GSC factors) * 0.01)
= 0.65 + (22 * 0.01)
= 0.87

Adjusted FP = VAF * (Total Unadjusted FP)
= 0.87 * 30
≈ 26

48

FP - Approximate Estimation of Size and Effort

- Assume the program will be implemented in Visual Basic, which has a standard rate of 50 LOC/FP.
  - The project will be approximately 50 * 26 ≈ 1300 lines of code.

- Assume a programmer works for 5 FP per day.
  - The project will take approximately 26 / 5 ≈ 5 days.

FP - Actual Lines of Code in Visual Basic

Module Name	Total Lines	Effective Coded Lines
frmCustMaint.frm	989	400
modGeneral.bas	65	60
modValidate.bas	246	240
TOTALS =	**1300**	**700**

Estimation for Agile Projects

- Each user scenario (a mini-use-case) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- Each task is estimated separately. Note: estimation can be based on historical data, an empirical model, or "experience."
  - Alternatively, the 'volume' of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The effort estimates for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.