SOFTWARE ENGINEERING

Week 3
Software Project Management and Planning

Prof. Dr. Tolga OVATMAN Assoc. Prof. Dr. Ayse TOSUN
Istanbul Technical University
Computer Engineering Department

Agenda

1. Project Planning
2. Risk Management
3. Process Metrics
4. Quality Management

Project Management Concepts

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
  - Deliverables

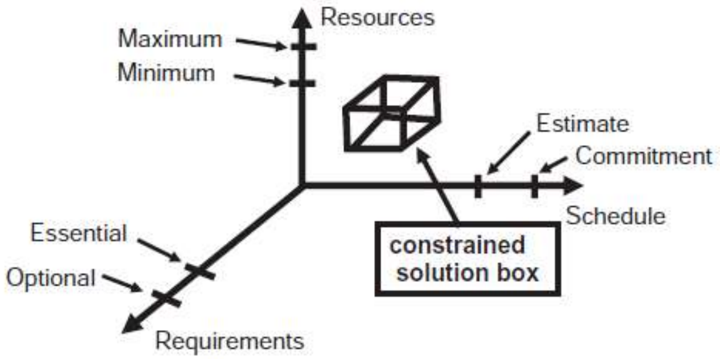
Estimation

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, previous projects)
  - 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

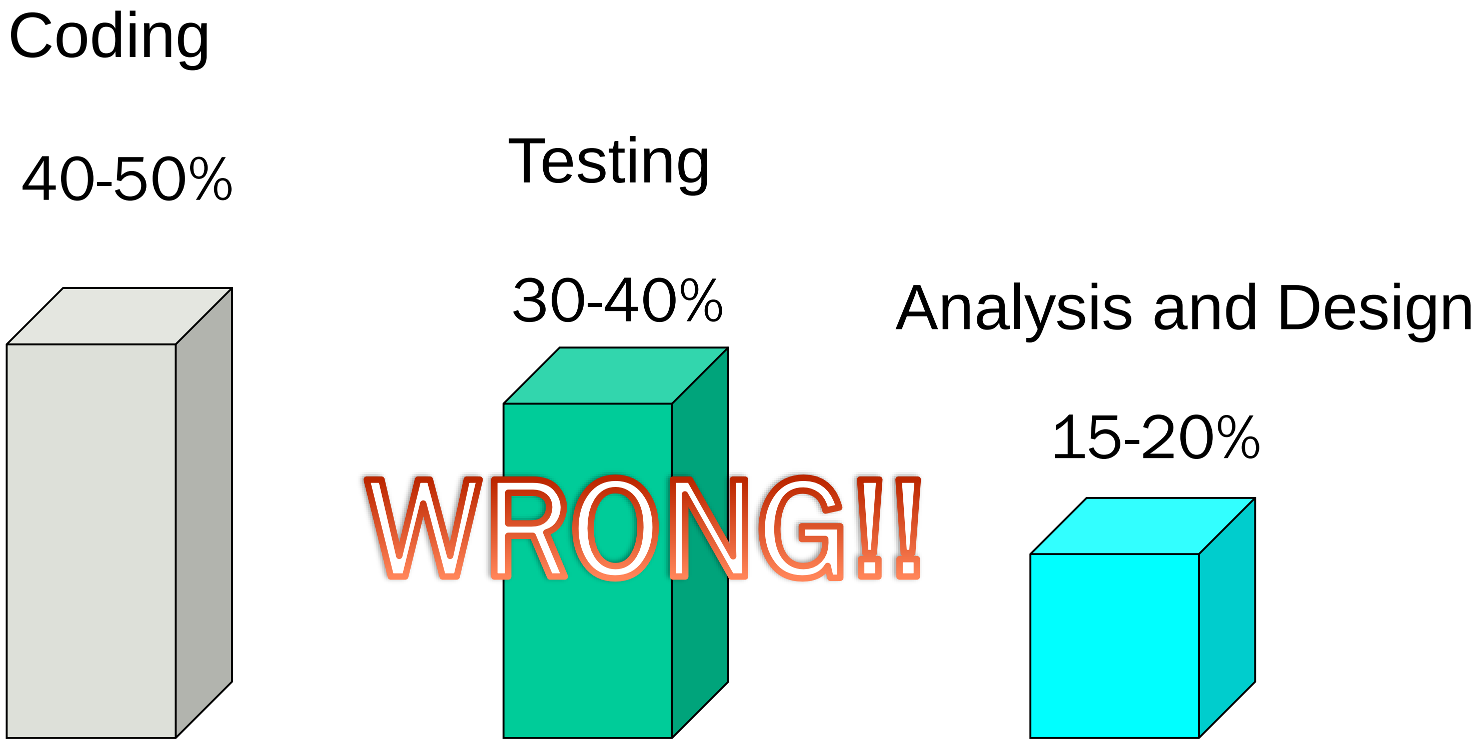
- In theory anything is negotiable, in practice
  - Time: Usually customer has a strict deadline
  - Cost: The budget of the project is almost fixed due to competition
  - Quality: There is a certain amount of quality induced according to the type of the project.



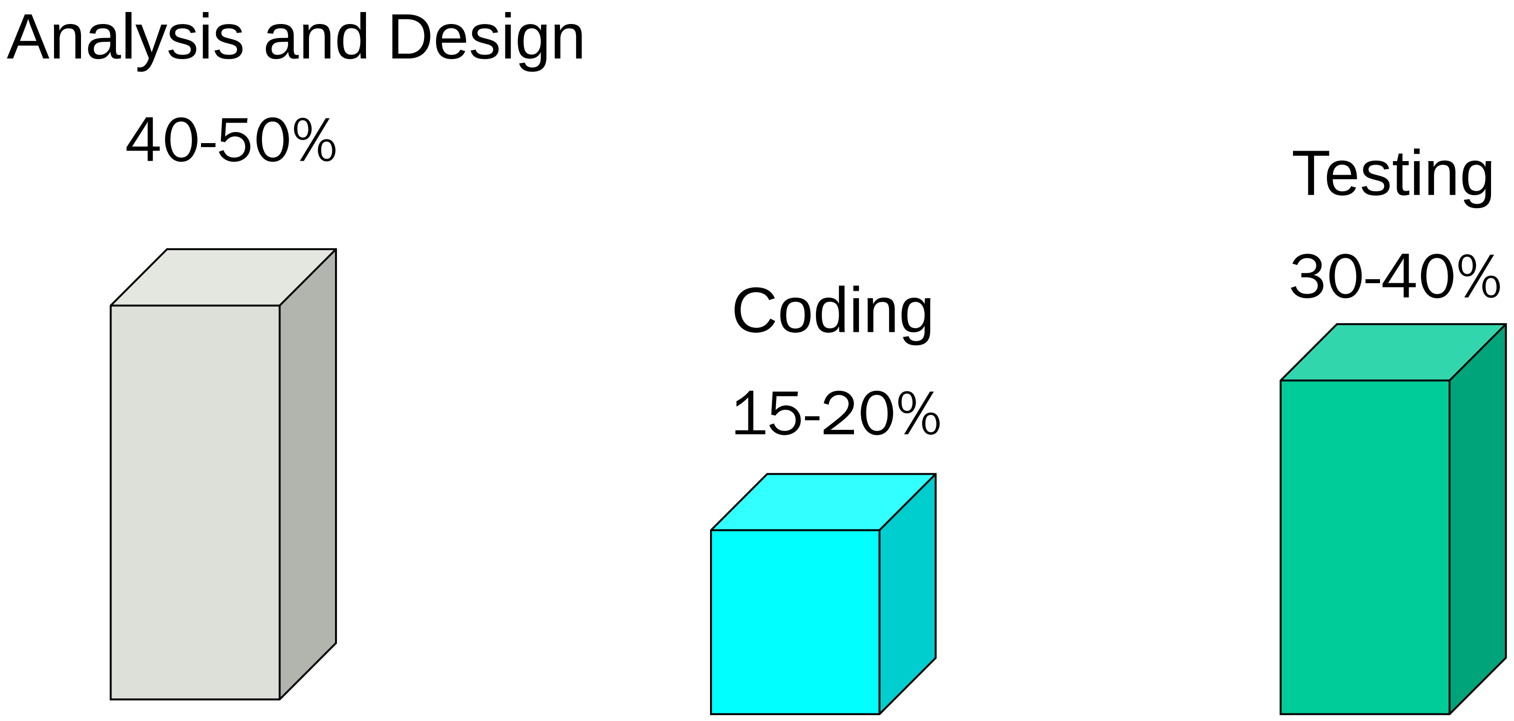
- Man-month= Man-month is a hypothetical unit of work representing the work done by one person in one month

- Brooks's law: Adding manpower to a late software project makes it later.

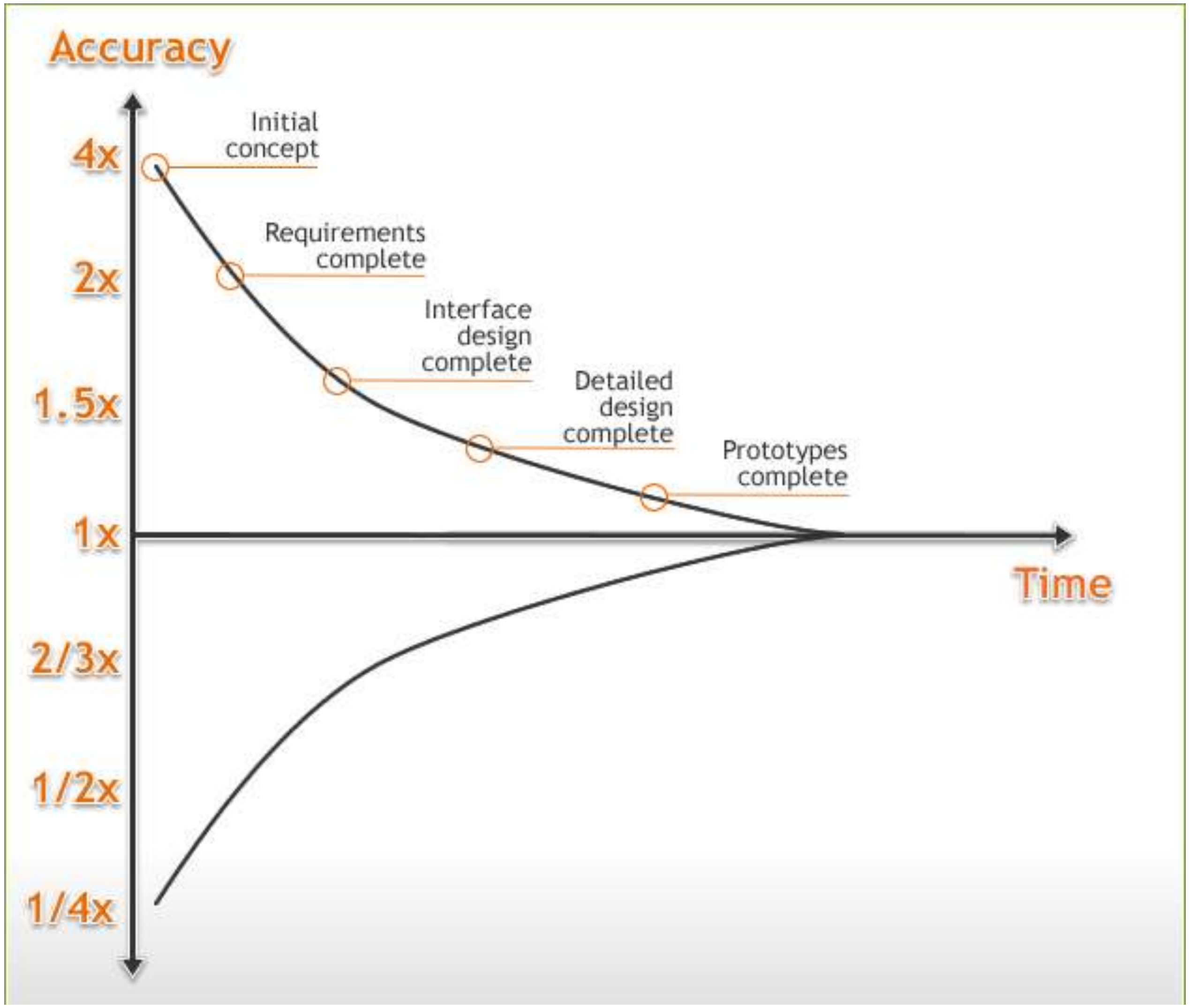
Estimation



Estimation



Estimation


Accuracy
4x
2x
1.5x
1x
2/3x
1/2x
1/4x
Initial concept
Requirements complete
Interface design complete
Detailed design complete
Prototypes complete
Time

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:

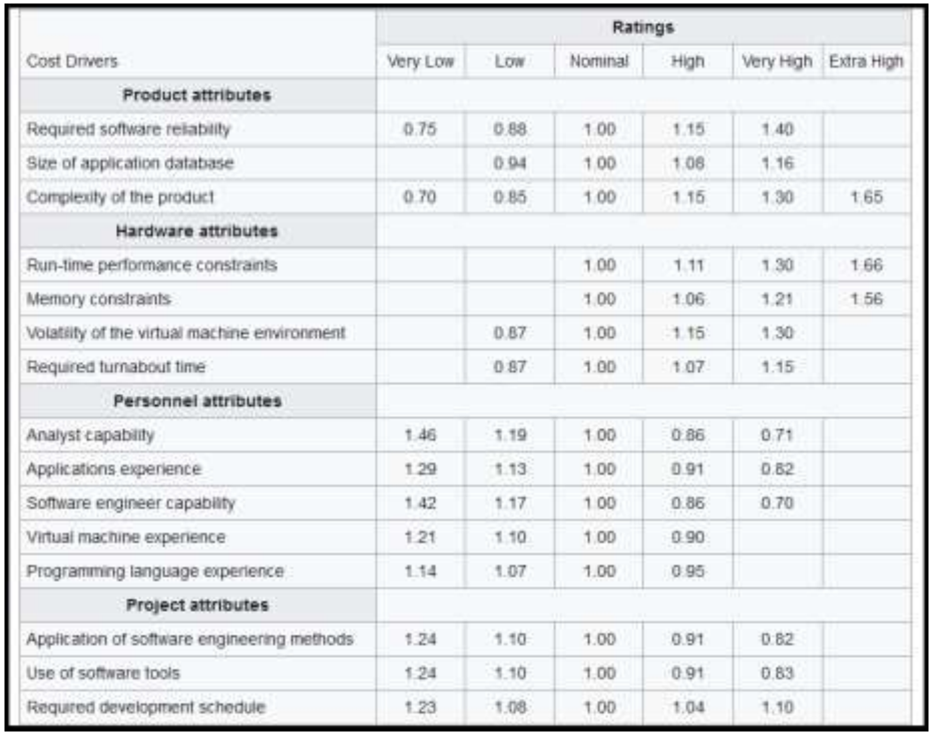
  Effort = A * (Size)^E * EAF

  (COCOMO model by Prof. Barry Boehm)

  - 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**.
- Majority of the models are similar in terms of attributes used, but they define different values for A, E and EAF.

- Effort Adjustment Factors in COCOMO Intermediate Model (also called cost drivers)
- Product attributes
  - Required software reliability extent
  - Size of application database
  - Complexity of the product
- Hardware attributes
  - Run-time performance constraints
  - Memory constraints
  - Volatility of the virtual machine environment
  - Required turnabout time
- Personnel attributes
  - Analyst capability
  - Software engineering capability
  - Applications experience
  - Virtual machine experience
  - Programming language experience
- Project attributes
  - Use of software tools
  - Application of software engineering methods
  - Required development schedule



	Ratings					
Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

From COCOMO II, Boehm 2000.

Product of the values of these factors are 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 an 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

$$\boxed{\mathrm{VAF} = \left(\sum_{i=1}^{14} \mathrm{GSC}_i \; * \; 0.01 \right) + 0.65}$$

$$\boxed{\text{Adjusted FP} = (\text{Unadjusted FP}) * \text{VAF}}$$

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: (data into the application)

- 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.

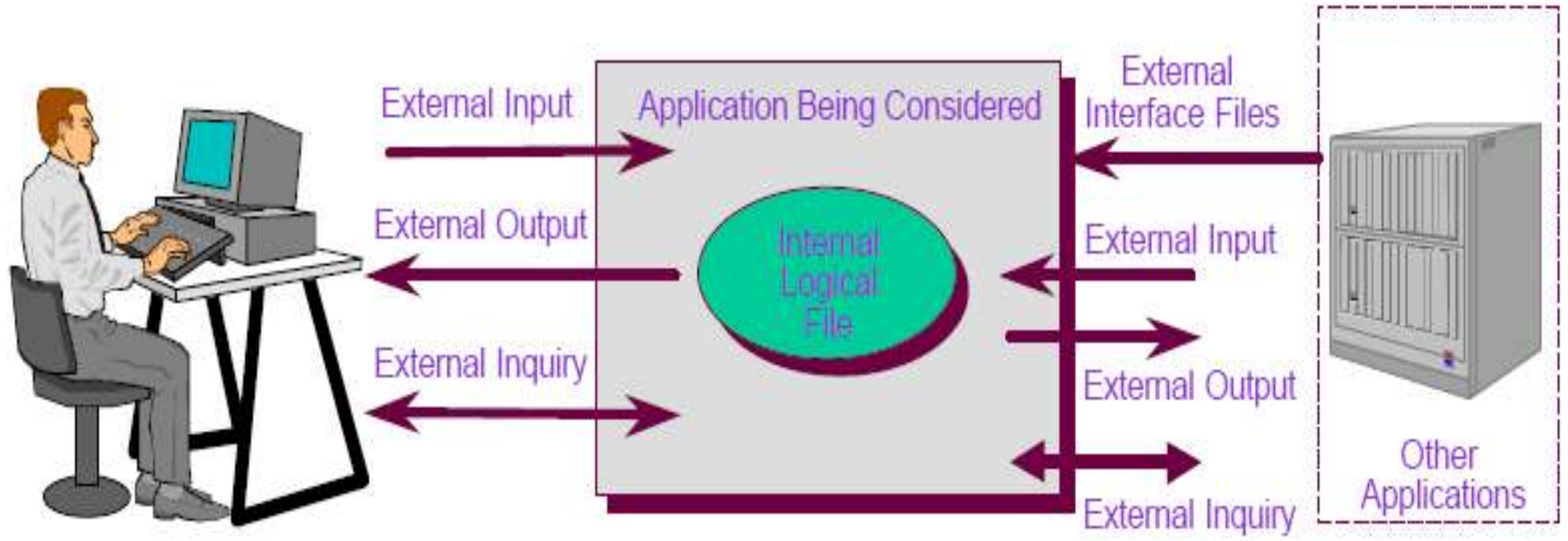
EO: (data out of the system)

- 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.

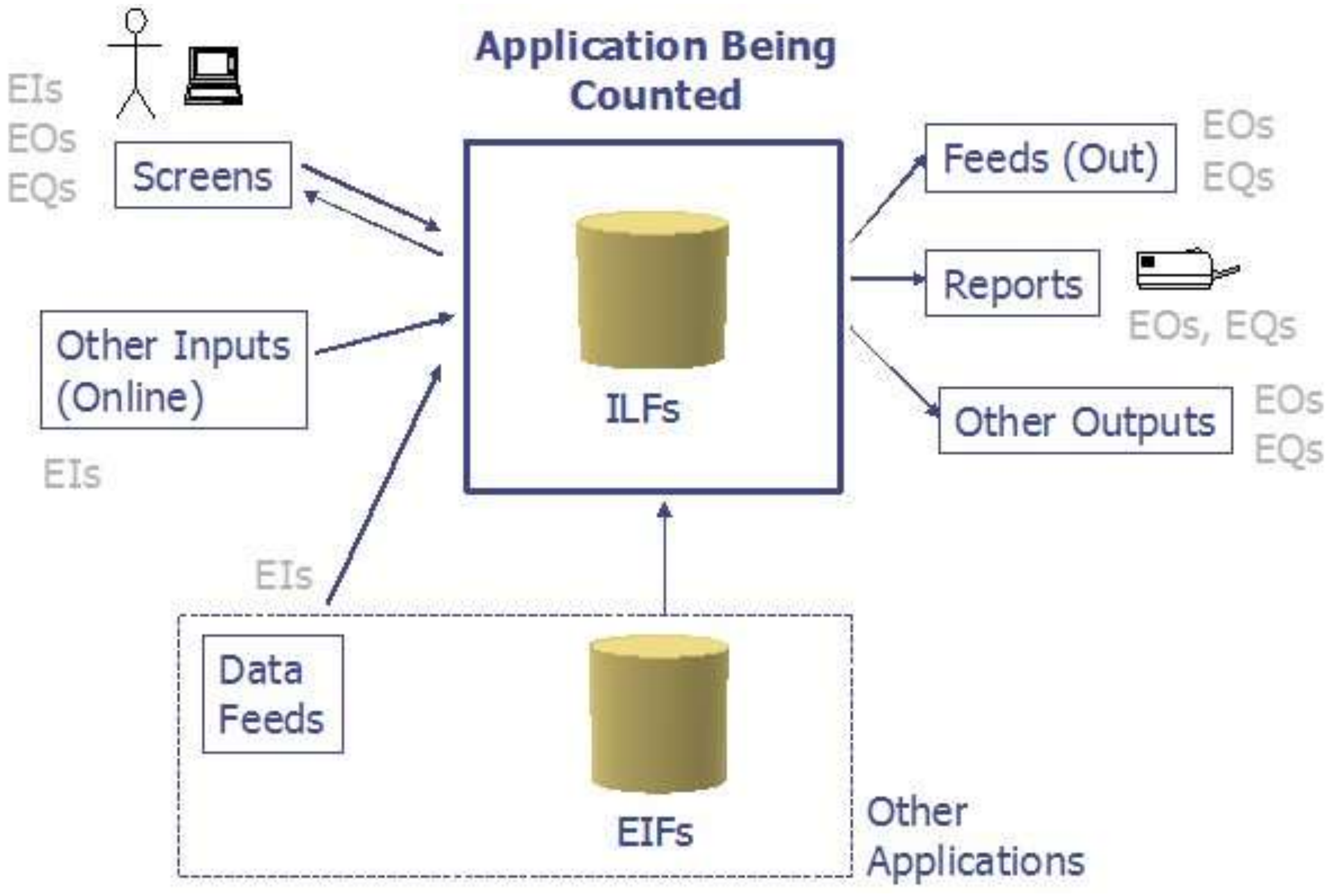
EQ: (data retrieval)

- 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.

- Note that data inside the application 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.

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.

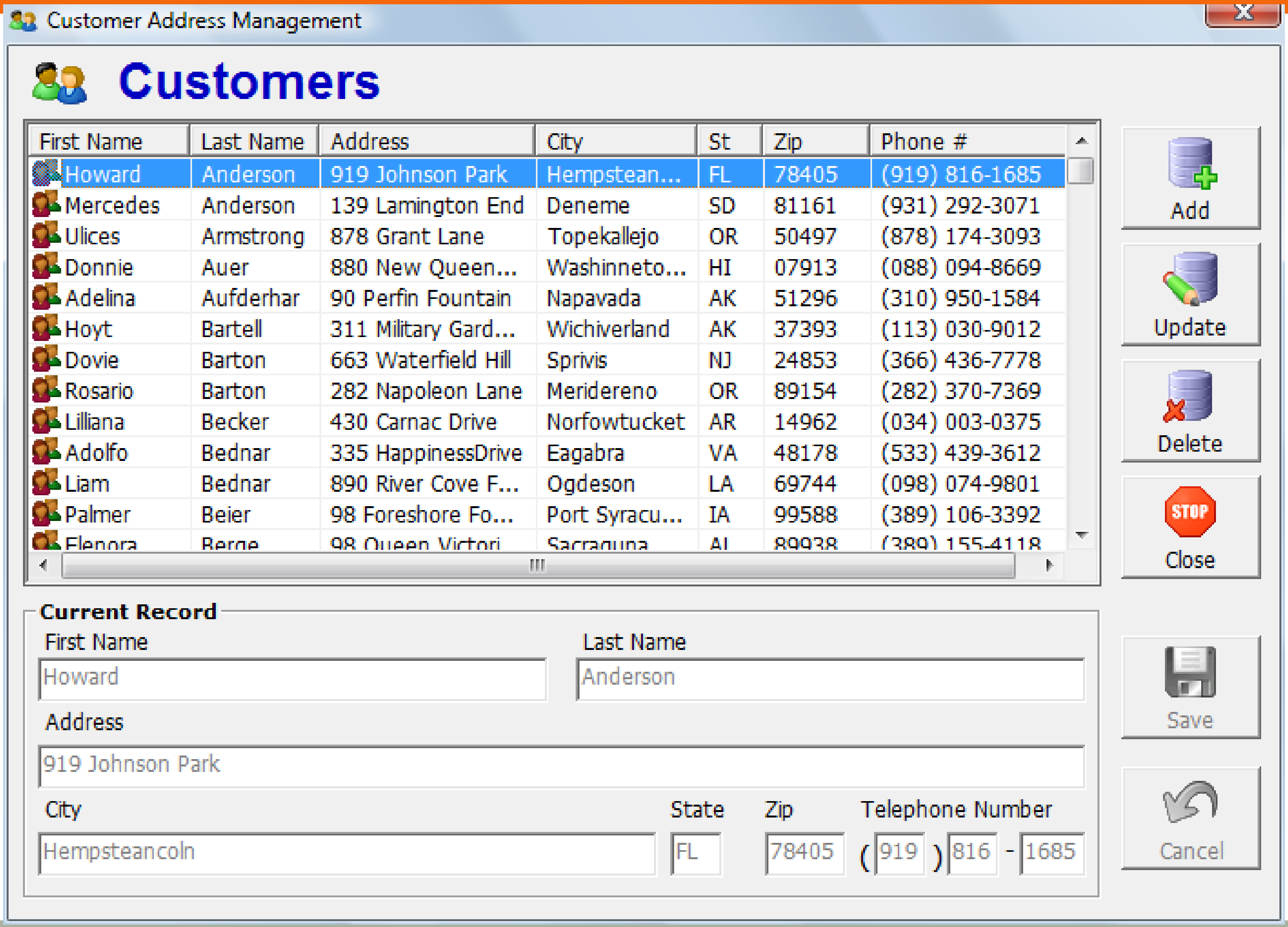
- Prefilling a listbox from ILF is considered as EQ, because this is an **implied inquiry**.

- 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.

FP Example :"Customer" Screen



Customer Address Management

Customers

First Name	Last Name	Address	City	St	Zip	Phone #
Howard	Anderson	919 Johnson Park	Hempstean...	FL	78405	(919) 816-1685
Mercedes	Anderson	139 Lamington End	Deneme	SD	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-3093
Donnie	Auer	880 New Queen...	Washinneto...	HI	07913	(088) 094-8669
Adelina	Aufderhar	90 Perfin Fountain	Napavada	AK	51296	(310) 950-1584
Hoyt	Bartell	311 Military Gard...	Wichiverland	AK	37393	(113) 030-9012
Dovie	Barton	663 Waterfield Hill	Sprivis	NJ	24853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meridereno	OR	89154	(282) 370-7369
Lilliana	Becker	430 Carnac Drive	Norfowtucket	AR	14962	(034) 003-0375
Adolfo	Bednar	335 HappinessDrive	Eagabra	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogdeson	LA	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Port Syracu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	Sacraguna	AL	89938	(389) 155-4118

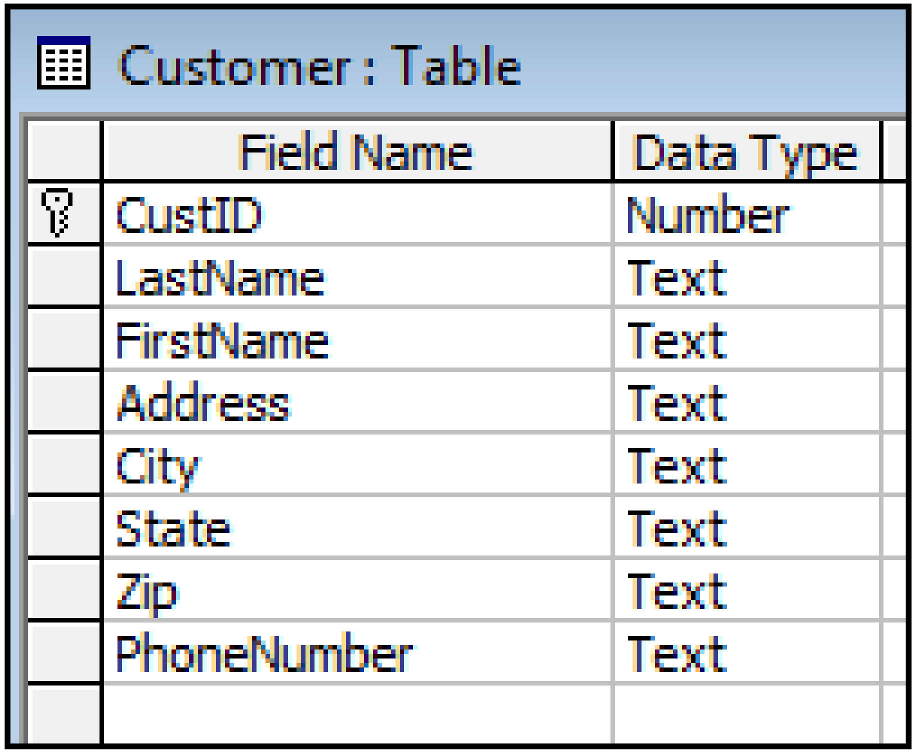
Current Record

First Name: Howard
Last Name: Anderson
Address: 919 Johnson Park
City: Hempsteancoln
State: FL
Zip: 78405
Telephone Number: (919) 816 - 1685

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

FP Example : Counting the Functions (2)

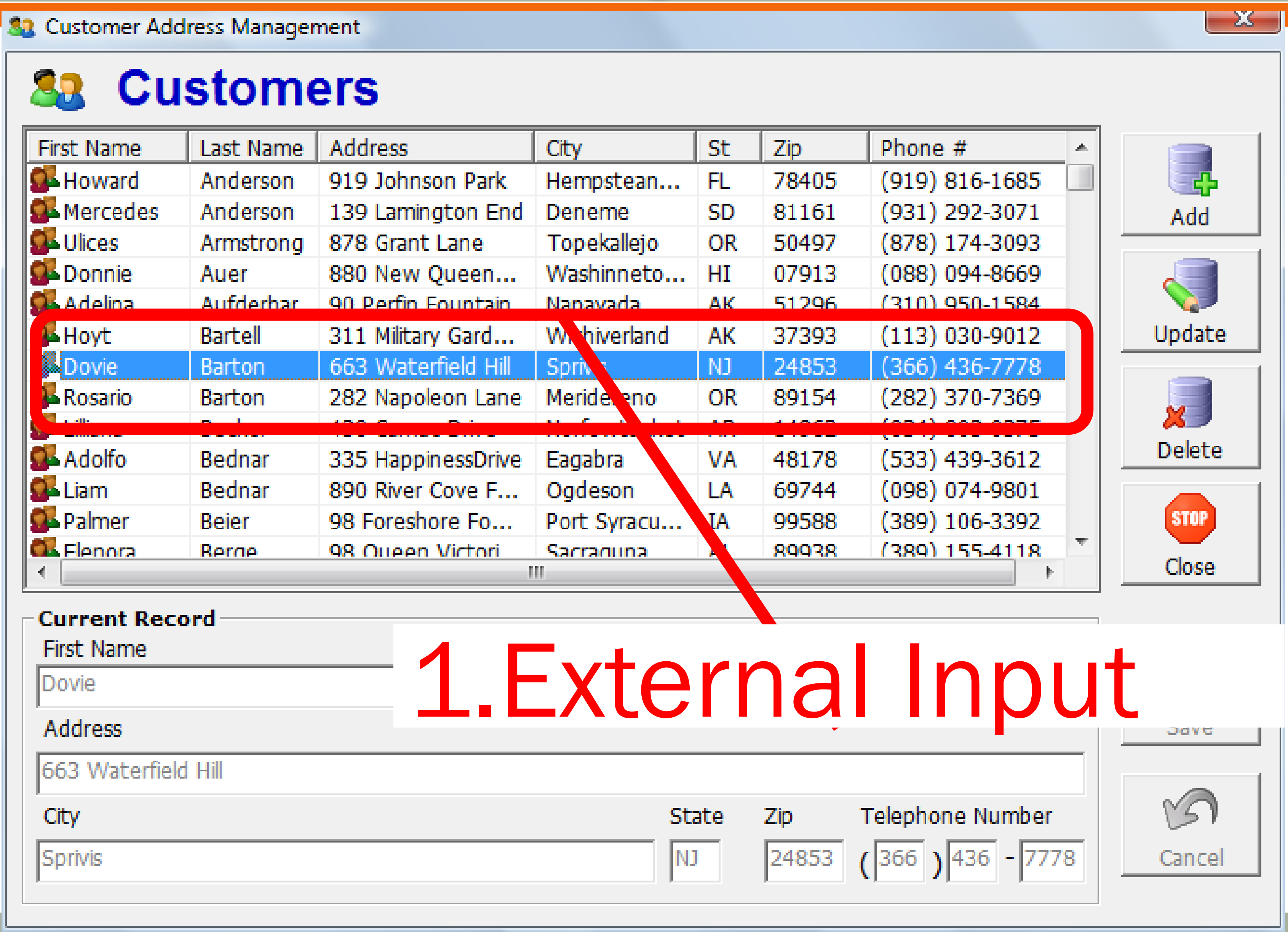
- There are 3 External Inputs
  1) “Customer selection” area in the screen for selecting a customer
  2) “Current record” area in the screen for entering / editing customer data
  3) All transaction buttons
     (ADD, UPDATE, etc.)

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

FP Example : EI



First Name	Last Name	Address	City	St	Zip	Phone #
Howard	Anderson	919 Johnson Park	Hempstean...	FL	78405	(919) 816-1685
Mercedes	Anderson	139 Lamington End	Deneme	SD	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-3093
Donnie	Auer	880 New Queen...	Washinneto...	HI	07913	(088) 094-8669
Adelina	Aufderhar	90 Perfin Fountain	Napavada	AK	51296	(310) 950-1584
Hoyt	Bartell	311 Military Gard...	W[illegible]hiverland	AK	37393	(113) 030-9012
Dovie	Barton	663 Waterfield Hill	Spriv[illegible]	NJ	24853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meride[illegible]eno	OR	89154	(282) 370-7369
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Adolfo	Bednar	335 HappinessDrive	Eagabra	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogdeson	LA	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Port Syracu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	Sacraguna	[illegible]	89938	(389) 155-4118

Current Record

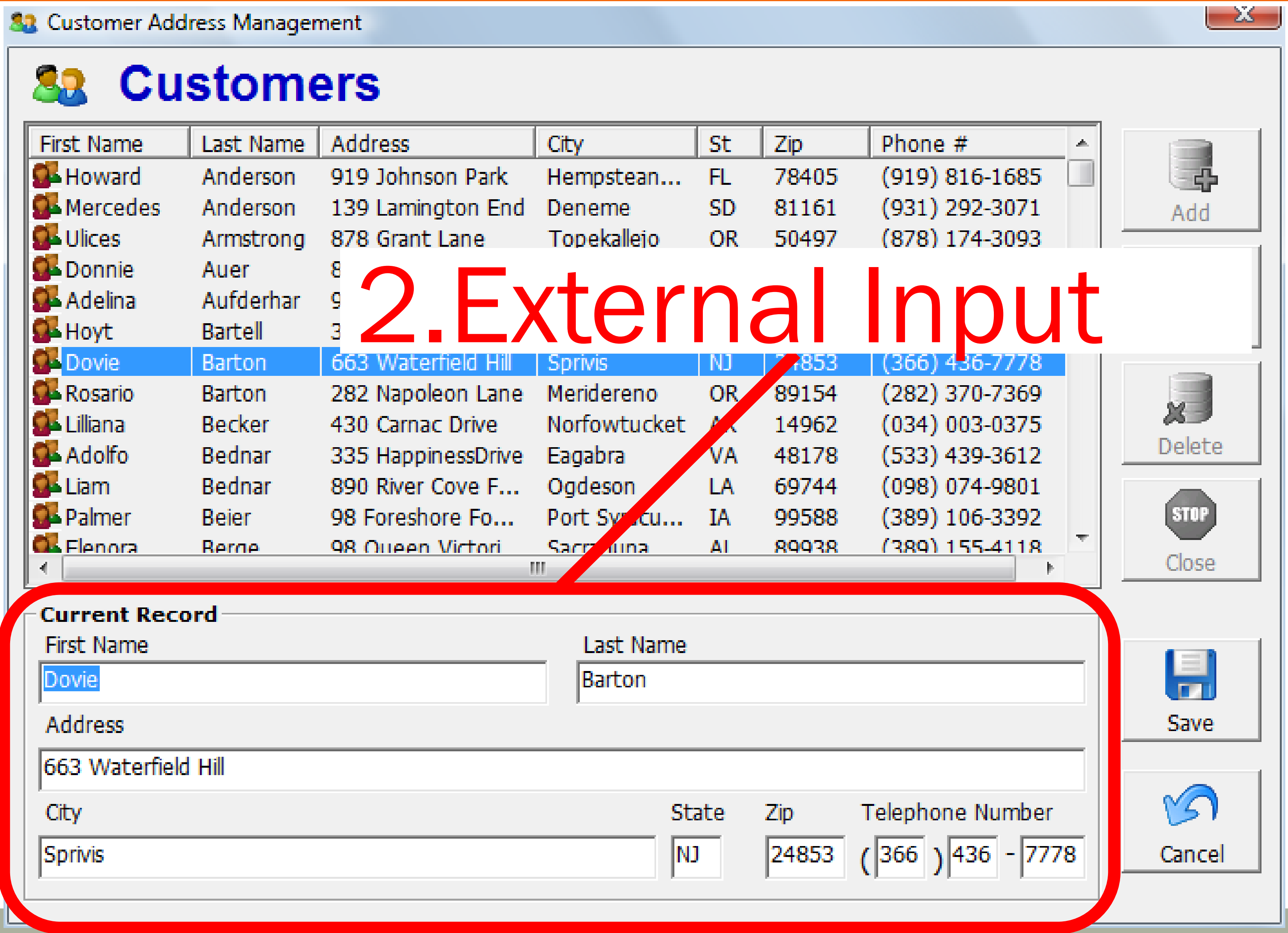
First Name: Dovie

Address: 663 Waterfield Hill

City: Sprivis | State: NJ | Zip: 24853 | Telephone Number: (366) 436 - 7778

1.External Input

FP Example : EI



Customer Address Management

Customers

First Name	Last Name	Address	City	St	Zip	Phone #
Howard	Anderson	919 Johnson Park	Hempstean...	FL	78405	(919) 816-1685
Mercedes	Anderson	139 Lamington End	Deneme	SD	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-3093
Donnie	Auer	8				
Adelina	Aufderhar	9				
Hoyt	Bartell	3				
Dovie	Barton	663 Waterfield Hill	Sprivis	NJ	[illegible]853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meridereno	OR	89154	(282) 370-7369
Lilliana	Becker	430 Carnac Drive	Norfowtucket	[illegible]	14962	(034) 003-0375
Adolfo	Bednar	335 HappinessDrive	Eagabra	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogdeson	LA	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Port Sy[illegible]cu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	Sacr[illegible]luna	AL	89938	(389) 155-4118

Add | Delete | Close

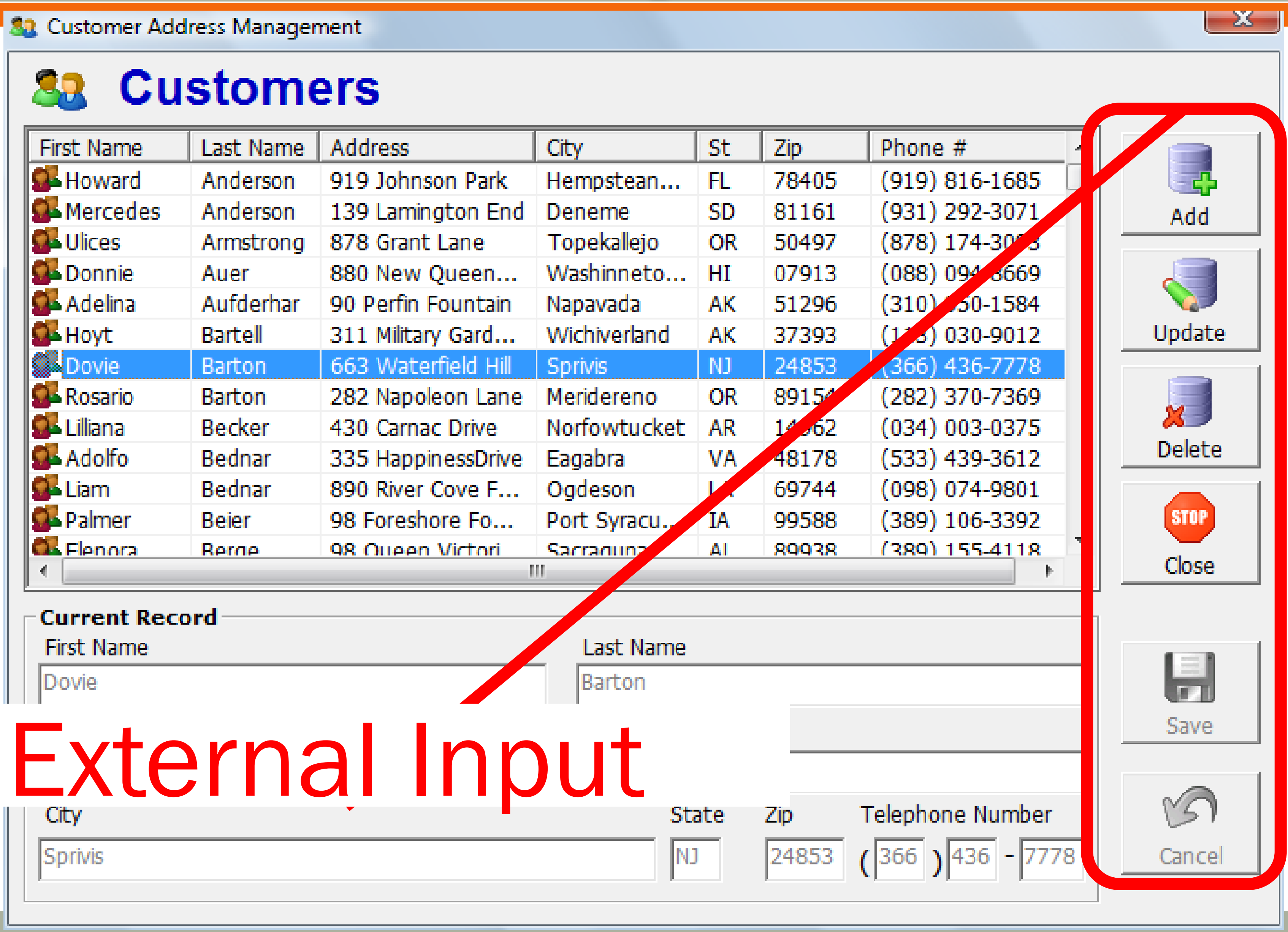
Current Record

First Name	Dovie
Last Name	Barton
Address	663 Waterfield Hill
City	Sprivis
State	NJ
Zip	24853
Telephone Number	(366) 436 - 7778

Save | Cancel

2.External Input

FP Example : EI



Customer Address Management

Customers

First Name	Last Name	Address	City	St	Zip	Phone #
Howard	Anderson	919 Johnson Park	Hempstean...	FL	78405	(919) 816-1685
Mercedes	Anderson	139 Lamington End	Deneme	SD	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-30[illegible]3
Donnie	Auer	880 New Queen...	Washinneto...	HI	07913	(088) 09[illegible]-8669
Adelina	Aufderhar	90 Perfin Fountain	Napavada	AK	51296	(310) [illegible]50-1584
Hoyt	Bartell	311 Military Gard...	Wichiverland	AK	37393	(1[illegible]3) 030-9012
Dovie	Barton	663 Waterfield Hill	Sprivis	NJ	24853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meridereno	OR	8915[illegible]	(282) 370-7369
Lilliana	Becker	430 Carnac Drive	Norfowtucket	AR	1[illegible]62	(034) 003-0375
Adolfo	Bednar	335 HappinessDrive	Eagabra	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogdeson	[illegible]	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Port Syracu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	Sacraquin[illegible]	AL	89938	(389) 155-4118

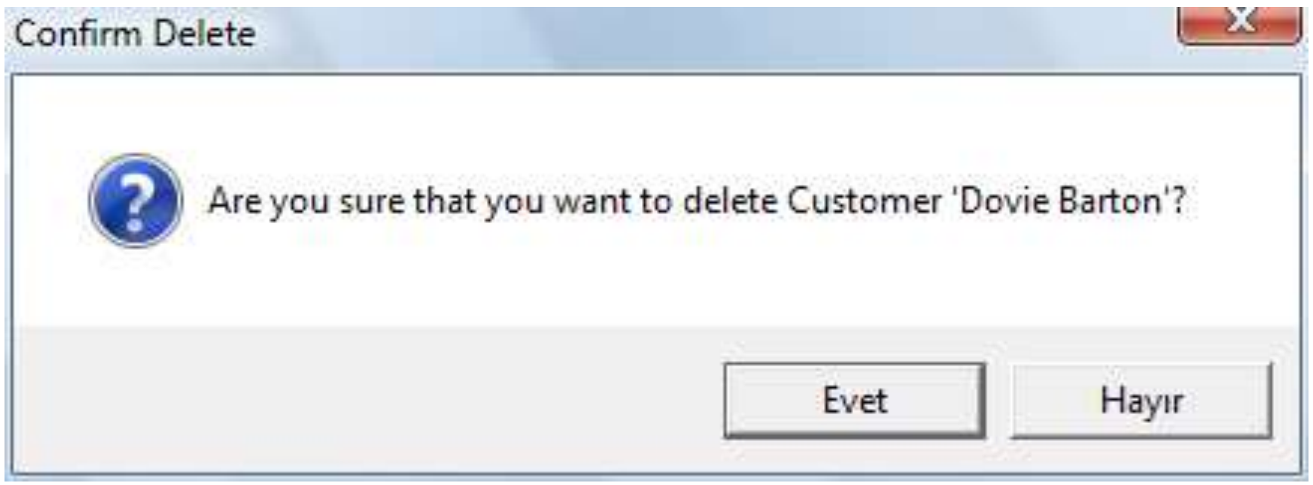
Current Record

First Name	Last Name
Dovie	Barton

City	State	Zip	Telephone Number
Sprivis	NJ	24853	(366) 436 - 7778

3.External Input

FP Example : EQ

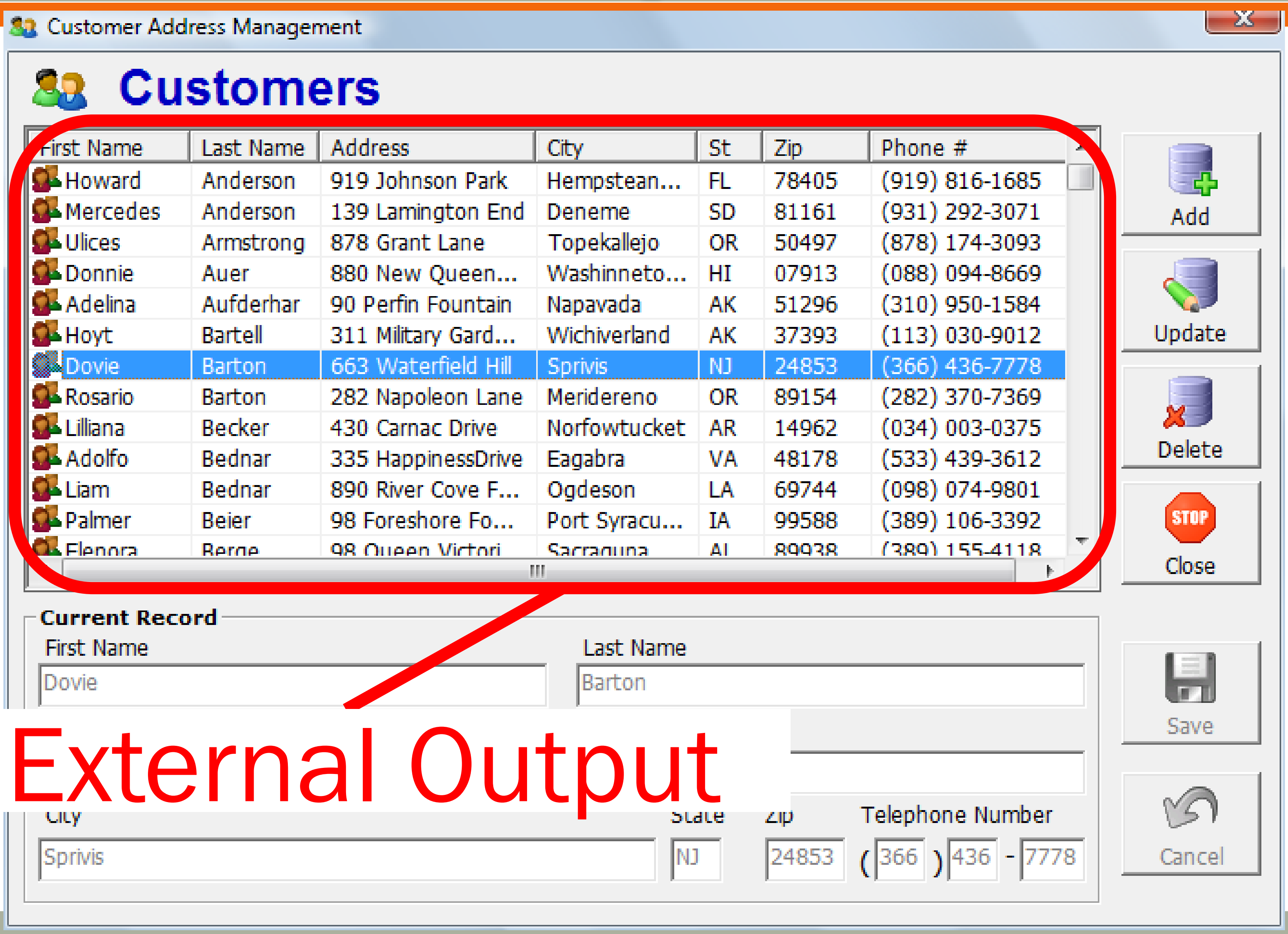


Confirm Delete

Are you sure that you want to delete Customer 'Dovie Barton'?

External Query

FP Example : EO



Customer Address Management

Customers

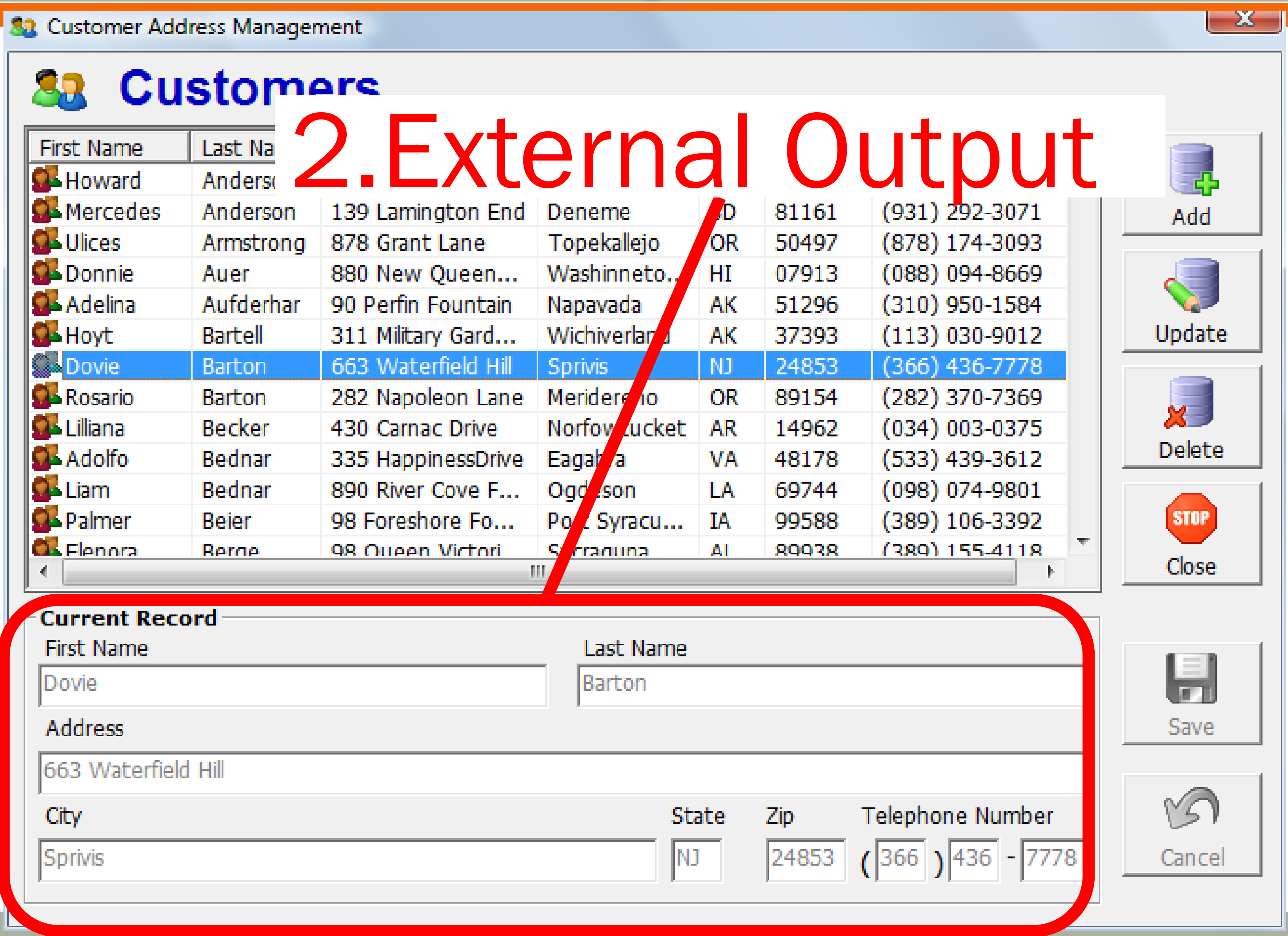
First Name	Last Name	Address	City	St	Zip	Phone #
Howard	Anderson	919 Johnson Park	Hempstean...	FL	78405	(919) 816-1685
Mercedes	Anderson	139 Lamington End	Deneme	SD	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-3093
Donnie	Auer	880 New Queen...	Washinneto...	HI	07913	(088) 094-8669
Adelina	Aufderhar	90 Perfin Fountain	Napavada	AK	51296	(310) 950-1584
Hoyt	Bartell	311 Military Gard...	Wichiverland	AK	37393	(113) 030-9012
Dovie	Barton	663 Waterfield Hill	Sprivis	NJ	24853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meridereno	OR	89154	(282) 370-7369
Lilliana	Becker	430 Carnac Drive	Norfowtucket	AR	14962	(034) 003-0375
Adolfo	Bednar	335 HappinessDrive	Eagabra	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogdeson	LA	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Port Syracu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	Sacraguna	AL	89938	(389) 155-4118

Current Record

First Name	Dovie
Last Name	Barton
City	Sprivis
State	NJ
Zip	24853
Telephone Number	(366) 436 - 7778

1.External Output

FP Example : EO



Customer Address Management

Customers

2.External Output

First Name	Last Na					
Howard	Anders					
Mercedes	Anderson	139 Lamington End	Deneme	[illegible]D	81161	(931) 292-3071
Ulices	Armstrong	878 Grant Lane	Topekallejo	OR	50497	(878) 174-3093
Donnie	Auer	880 New Queen...	Washinneto...	HI	07913	(088) 094-8669
Adelina	Aufderhar	90 Perfin Fountain	Napavada	AK	51296	(310) 950-1584
Hoyt	Bartell	311 Military Gard...	Wichiverland	AK	37393	(113) 030-9012
Dovie	Barton	663 Waterfield Hill	Sprivis	NJ	24853	(366) 436-7778
Rosario	Barton	282 Napoleon Lane	Meridere[illegible]io	OR	89154	(282) 370-7369
Lilliana	Becker	430 Carnac Drive	Norfow[illegible]ucket	AR	14962	(034) 003-0375
Adolfo	Bednar	335 HappinessDrive	Eaga[illegible]a	VA	48178	(533) 439-3612
Liam	Bednar	890 River Cove F...	Ogd[illegible]son	LA	69744	(098) 074-9801
Palmer	Beier	98 Foreshore Fo...	Po[illegible] Syracu...	IA	99588	(389) 106-3392
Elenora	Berge	98 Queen Victori	S[illegible]raguna	AL	89938	(389) 155-4118

Current Record

First Name: Dovie
Last Name: Barton
Address: 663 Waterfield Hill
City: Sprivis
State: NJ
Zip: 24853
Telephone Number: (366) 436 - 7778

Unadjusted Function Points

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

Unadjusted Function Points = $\boxed{30}$

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.

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

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?

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?

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

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

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.

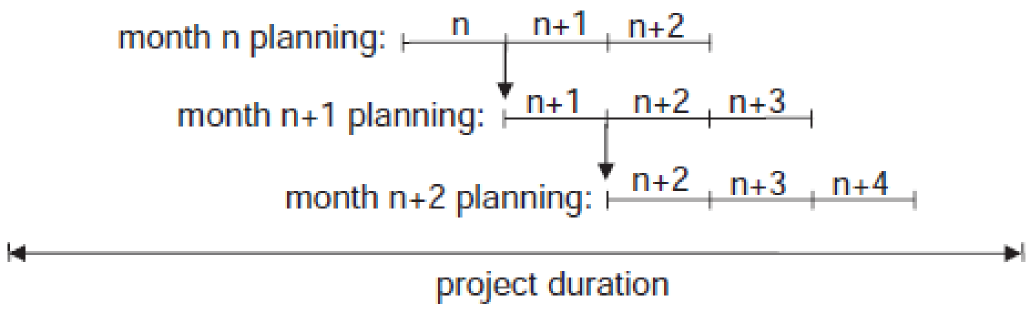
- Assume a programmer works for 5 FP per day.
  - The project will take approximately 26 / 5 ≈ 5 days.

Planning

ꕥ 3 ꕥ

Project Planning

- Planning techniques, includes the following activities
  - Identify and organize resources (e.g. team, hardware) required for the project
  - Develop a work breakdown structure (WBS) and identify packages for the tasks in the work breakdown structure(WBS)
  - Define a schedule of objectively measurable milestones organized as a GANTT chart
  - Prepare a schedule network and identify the critical path(s)
- Planning activities should be continuously held during the project as a rolling wave



Team Planning

ᘏ 3.1 ᘓ

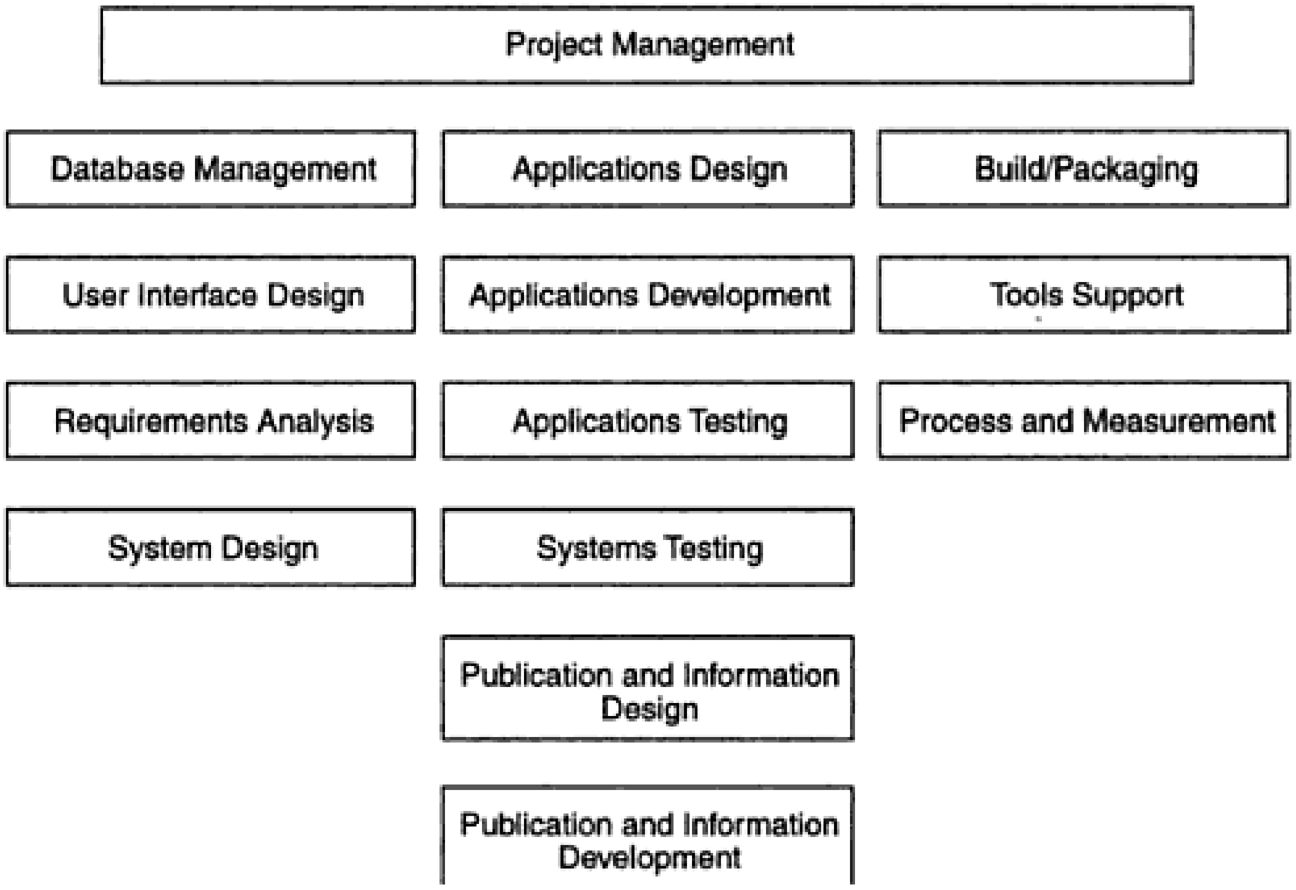
Software Teams

- *The following factors must be considered when selecting a software project team structure*
  - the difficulty of the problem to be solved
  - the size of the resultant program(s) in lines of code or function points
  - the time that the team will stay together (team lifetime)
  - the degree to which the problem can be modularized
  - the required quality and reliability of the system to be built
  - the rigidity of the delivery date
  - the degree of sociability (communication) required for the project

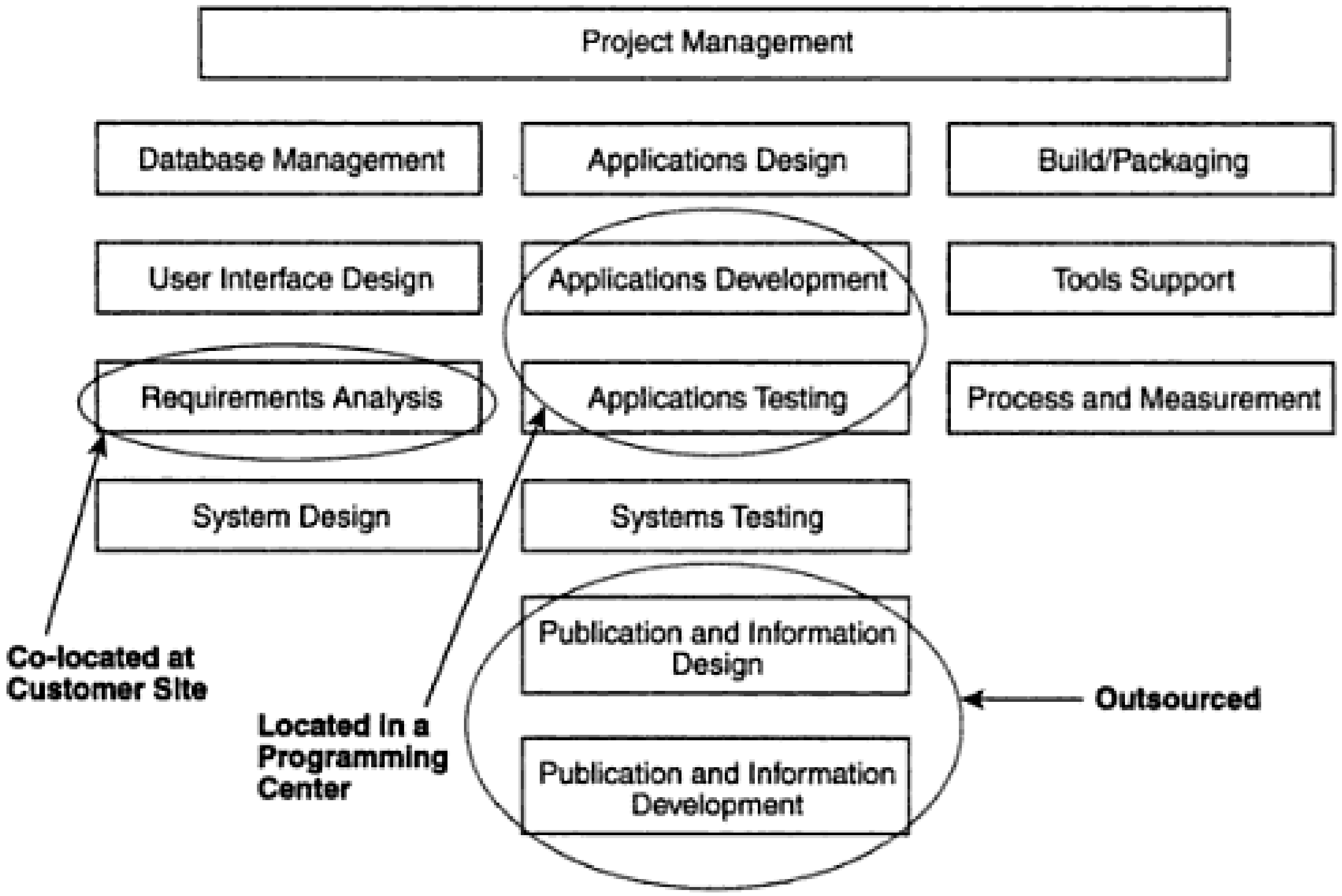
Team Organizations

- There exist two main types of organizational structures:
  - Hierarchical organization: All the people associated with a project are grouped into functional departments that report directly within the vertical line of command.
  - Matrix organization: People are grouped based on the functions they perform.
- Also based on the location of the project personnel, teams may be:
  - Centralized
  - Distributed
    - Globally distributed
  - Mixed
- Small software engineering groups are usually organised informally without a rigid structure.
- For large projects, there may be a hierarchical structure where different groups are responsible for different sub-projects.

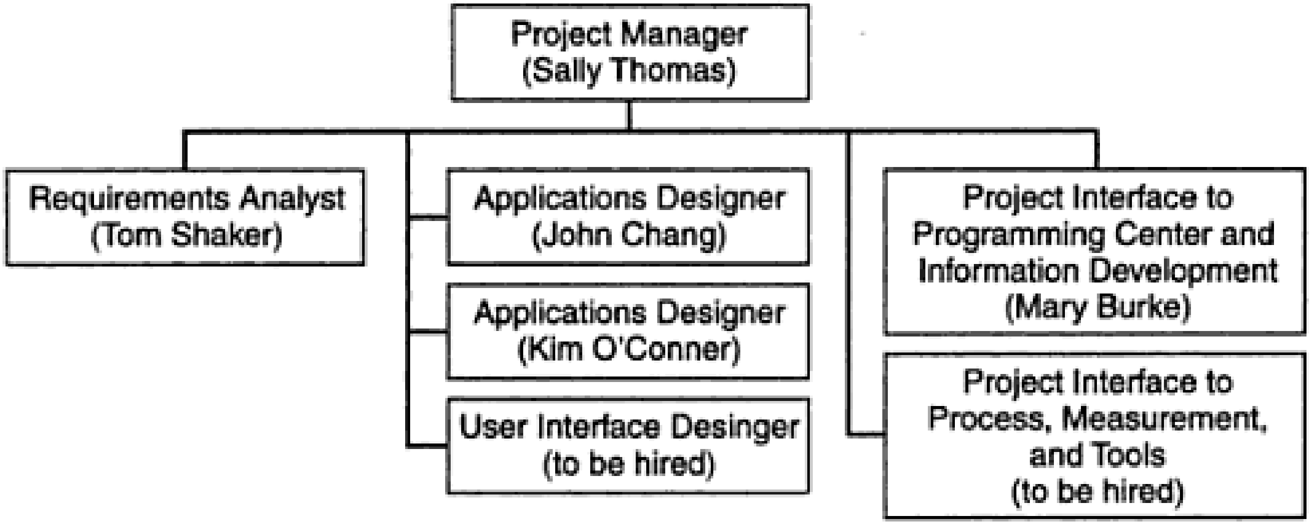
Team Organizations



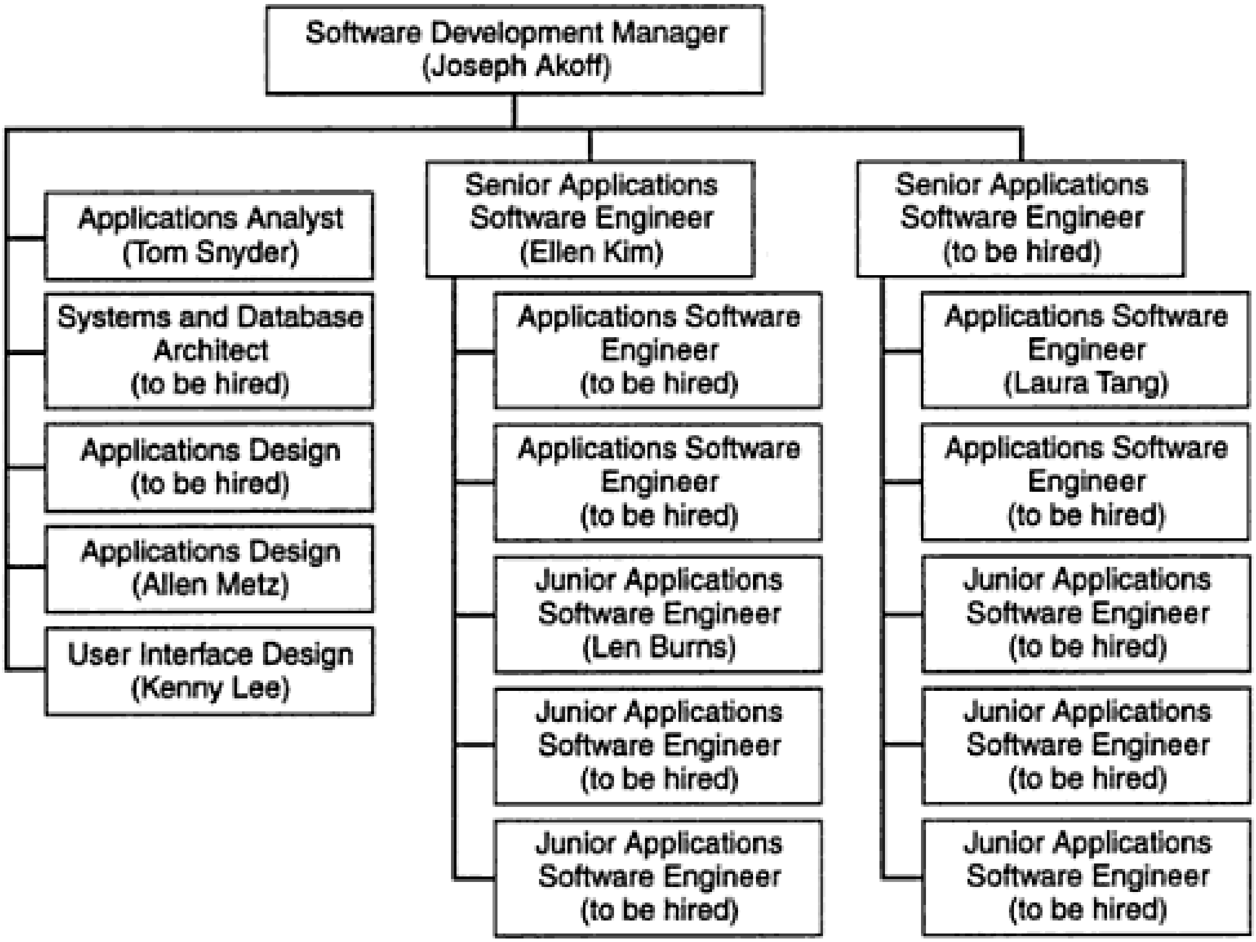
An Example Team Organization



An Example Team Organization


Project Manager (Sally Thomas)
Requirements Analyst (Tom Shaker)
Applications Designer (John Chang)
Applications Designer (Kim O'Conner)
User Interface Desinger (to be hired)
Project Interface to Programming Center and Information Development (Mary Burke)
Project Interface to Process, Measurement, and Tools (to be hired)

Team Organizations


Software Development Manager (Joseph Akoff)
Applications Analyst (Tom Snyder)
Systems and Database Architect (to be hired)
Applications Design (to be hired)
Applications Design (Allen Metz)
User Interface Design (Kenny Lee)
Senior Applications Software Engineer (Ellen Kim)
Applications Software Engineer (to be hired)
Applications Software Engineer (to be hired)
Junior Applications Software Engineer (Len Burns)
Junior Applications Software Engineer (to be hired)
Junior Applications Software Engineer (to be hired)
Senior Applications Software Engineer (to be hired)
Applications Software Engineer (Laura Tang)
Applications Software Engineer (to be hired)
Junior Applications Software Engineer (to be hired)
Junior Applications Software Engineer (to be hired)
Junior Applications Software Engineer (to be hired)

Agile Teams

Agile teams have the following characteristics

- Co-location
- Engaged with Customers
- Self-Organizing
- Accountable and Empowered
- Cross-Functional

Tips for Forming Your Agile Team

- Look for Generalists
- People Who Are Comfortable with Ambiguity
- Team Players Who Can Check Their Egos at the Door

Task Planning

ꕥ 3.2 ꕥ

Task Planning

- In order to work your way down in decomposition towards atomic tasks, you need to come up with an architectural decomposition.
- Architectural design of software is concerned with specifying the software modules, their interrelationships, and their connections to the environment of the software.
- We will examine Software Architectures in detail later. But let's work on a trivial ATM software example to illustrate how we obtain WBS.

Task Planning

Some prioritized requirements for ATM software

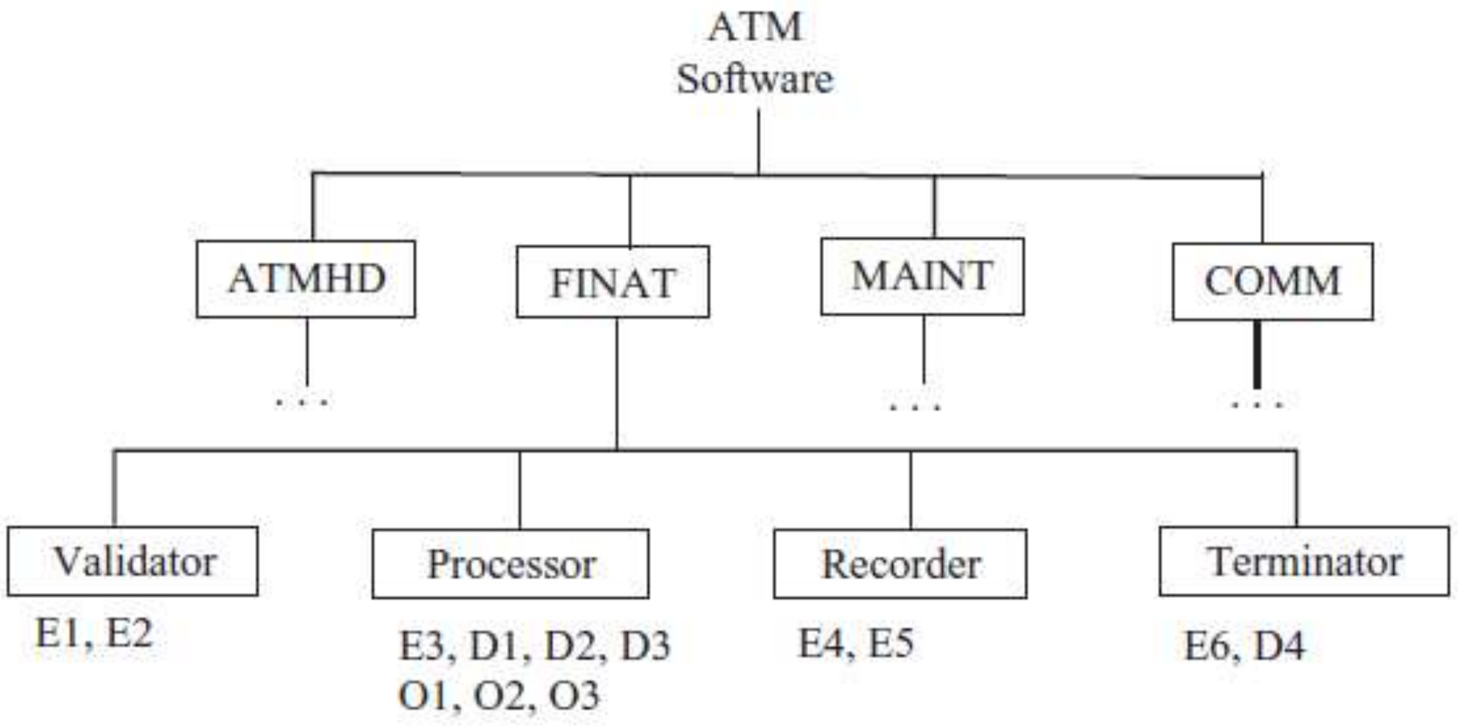
- Essential requirements
  - E1 Financial transactions shall be authorized by an ATM card and a password
  - E2 Financial transactions shall be terminated if a customer fails to enter the correct password « settable » times
  - E3 Financial transaction shall allow quick cash withdrawals
  - E4 Financial transaction shall provide a printed receipt for each transaction
  - E5 The ATM shall retain the information listed in the requirements specification, for each customer transaction
  - E6 Financial transaction shall process Terminate requests from customers

Task Planning

Some prioritized requirements for ATM software

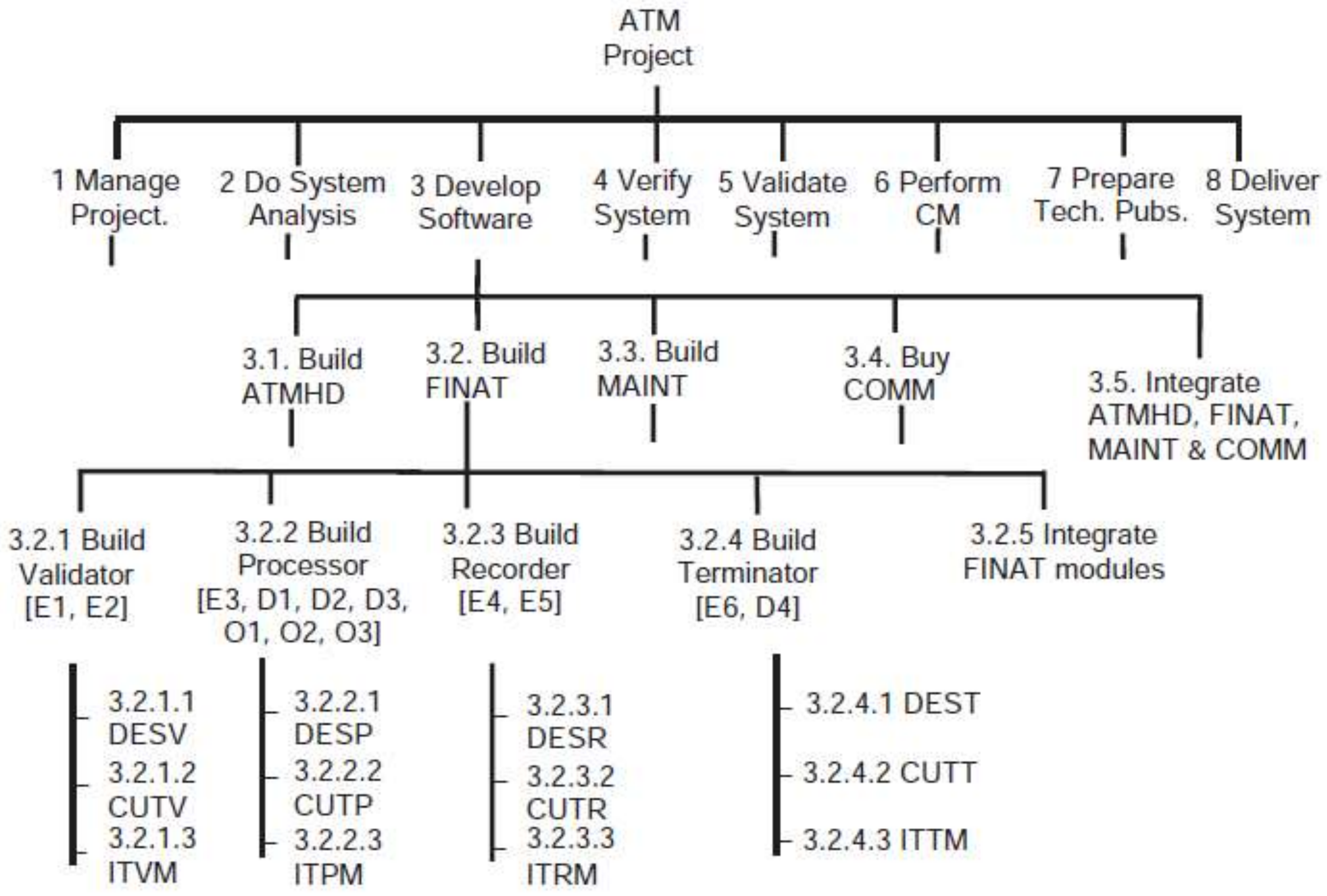
- Desirable requirements
  - D1 Financial transaction should accommodate balance inquiry transactions
  - D2 Financial transaction should accommodate standard withdrawal transactions
  - D3 Financial transaction should accommodate deposit transactions
  - D4 Customers should be allowed to conduct multiple transactions per session
- Optional requirements
  - O1 Financial transaction could support debit card transactions
  - O2 Financial transaction could support payment of utility bills
  - O3 Financial transaction could allow customers to purchase postage stamps which will be disbursed by the ATM hardware

Task Planning



ATMHD: Hardware Drivers
FINAT: Financial Transactions
MAINT: Maintenance and Diagnostics
COMM: Communications Package

Task Planning - WBS



DESx: detailed design of module x; CUTx: coding & unit testing x; ITxC: integrating and testing of x

Task Planning – Task Extraction

- Following WBS construction, tasks should be extracted to build a time plan and assign to resources(team). Tasks can also be assigned story points and can be used in estimation and tracking for agility.

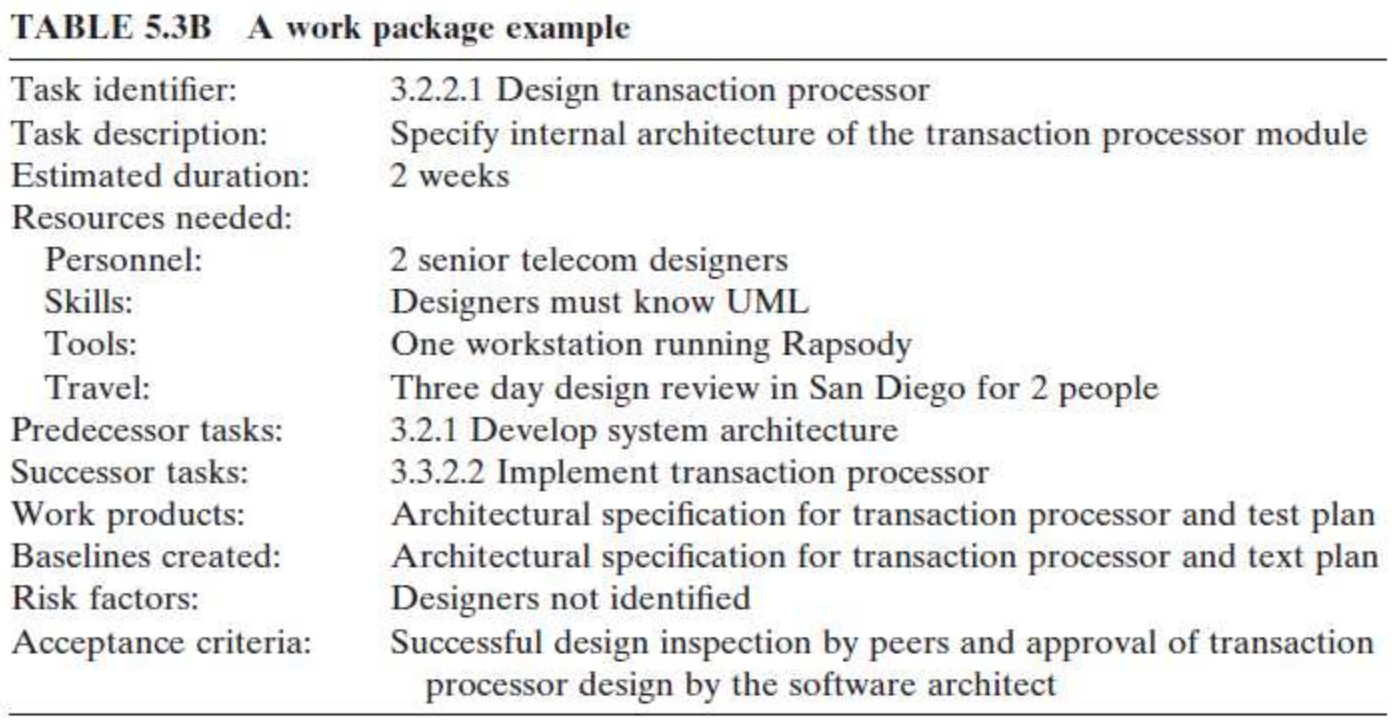


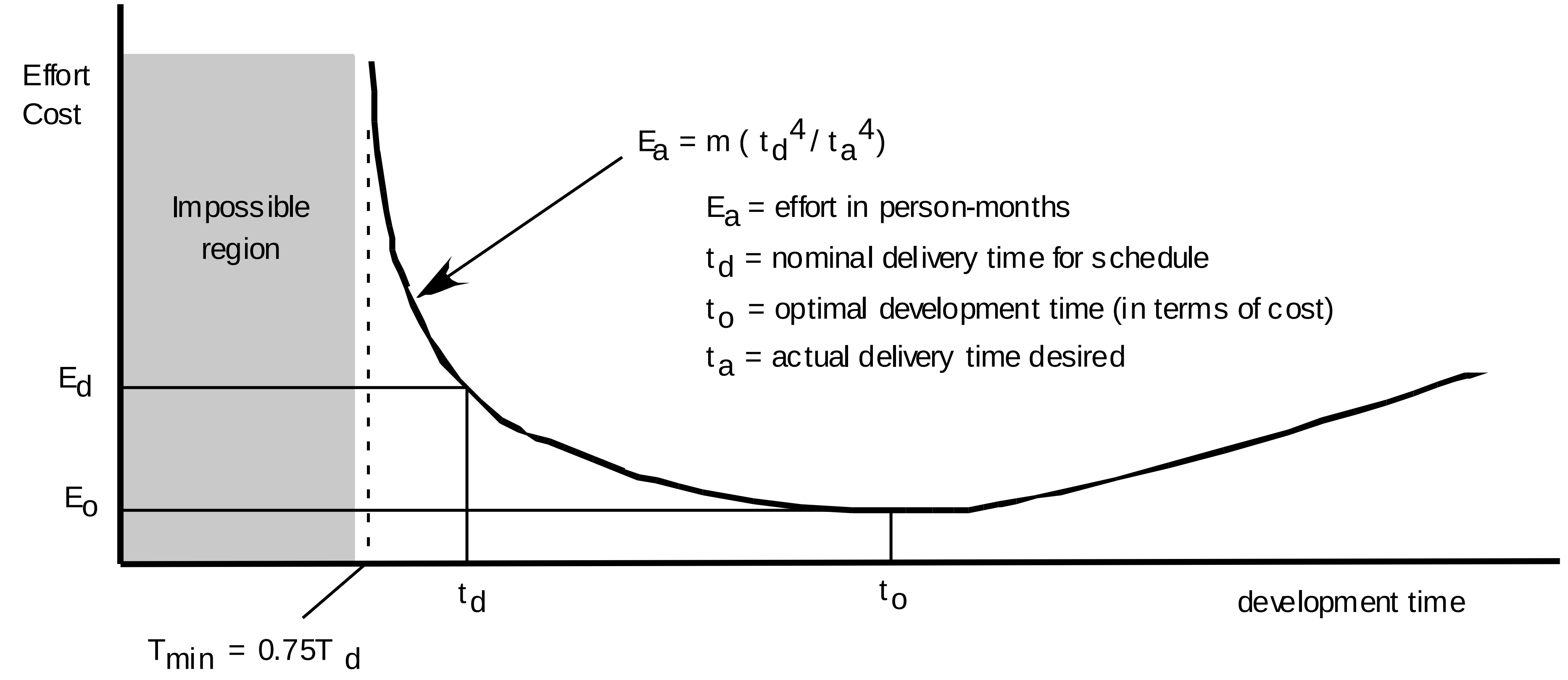
TABLE 5.3B A work package example

Task identifier:	3.2.2.1 Design transaction processor
Task description:	Specify internal architecture of the transaction processor module
Estimated duration:	2 weeks
Resources needed:	
Personnel:	2 senior telecom designers
Skills:	Designers must know UML
Tools:	One workstation running Rapsody
Travel:	Three day design review in San Diego for 2 people
Predecessor tasks:	3.2.1 Develop system architecture
Successor tasks:	3.3.2.2 Implement transaction processor
Work products:	Architectural specification for transaction processor and test plan
Baselines created:	Architectural specification for transaction processor and text plan
Risk factors:	Designers not identified
Acceptance criteria:	Successful design inspection by peers and approval of transaction processor design by the software architect

Time Planning

ꕥ 3.3 ꕥ

Effort and Delivery Time



Putnam–Norden–Rayleigh (PNR) Curve

For Scheduling

- **PERT Chart:**
  - Shows the relationships based on tasks or activities
  - Defines tasks that can be done concurrently or not and critical path

- **Gantt Chart:**
  - Shows schedule (timeline) information for each task as a bar chart

PERT Chart

- Project Evaluation and Review Technique

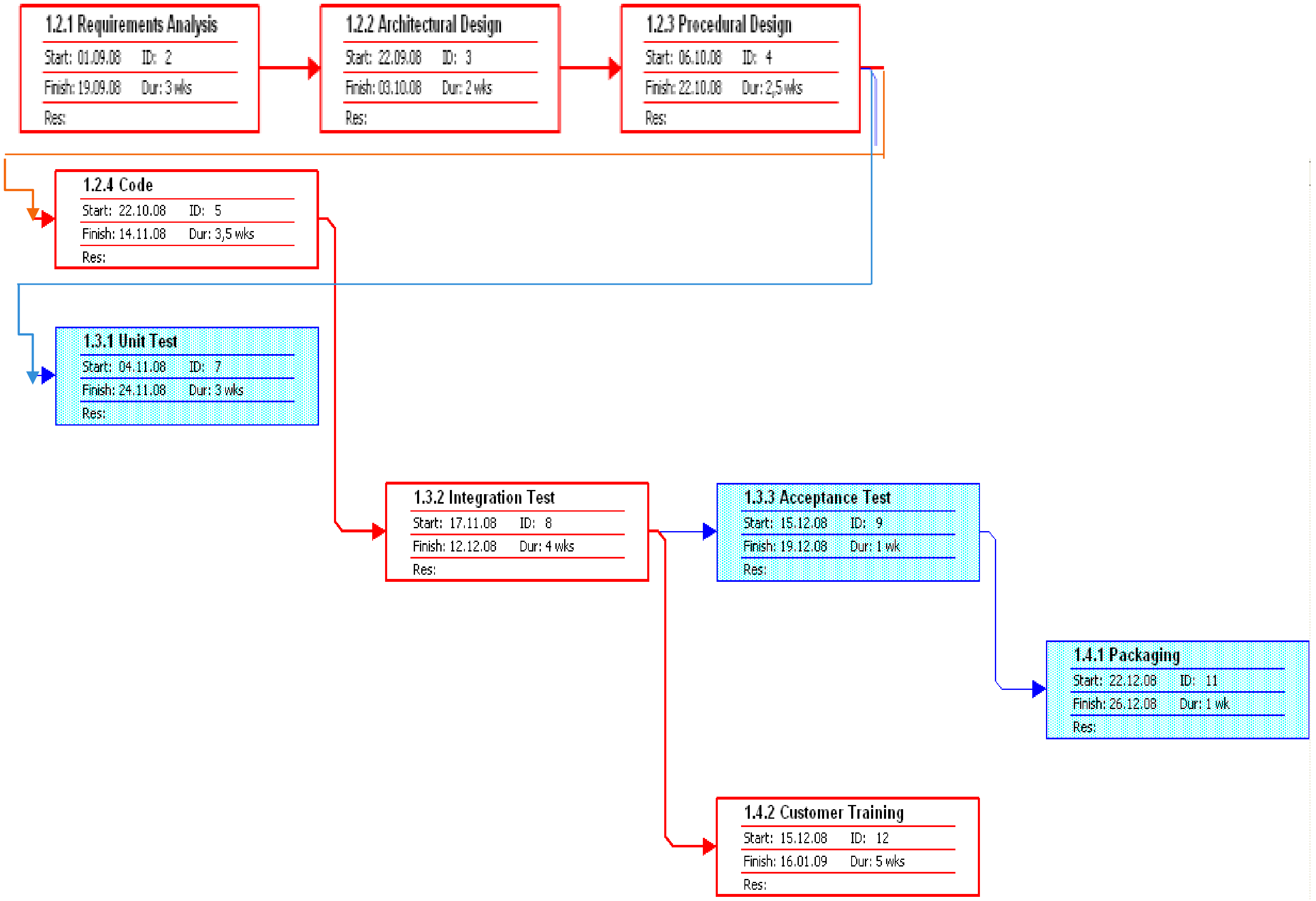
- Also known as Task Network Diagram
  - **Nodes:** activities / tasks and estimated duration
  - **Edges:** dependencies

- **Critical path:** Longest path from start to finish.
  Any slippage on the critical path will cause project delay.

PERT Example : Tasks and Durations

TASKS	WEEKS
1.2 Software Development	**11**
1.2.1 Requirements Analysis	3
1.2.2 Architectural Design	2
1.2.3 Procedural Design	2.5
1.2.4 Code	3.5
1.3 Testing	**7**
1.3.1 Unit Test	3
1.3.2 Integration Test	4
1.3.3 Acceptance Test	1
1.4 Operations	**5**
1.4.1 Packaging	1
1.4.2 Customer Training	5

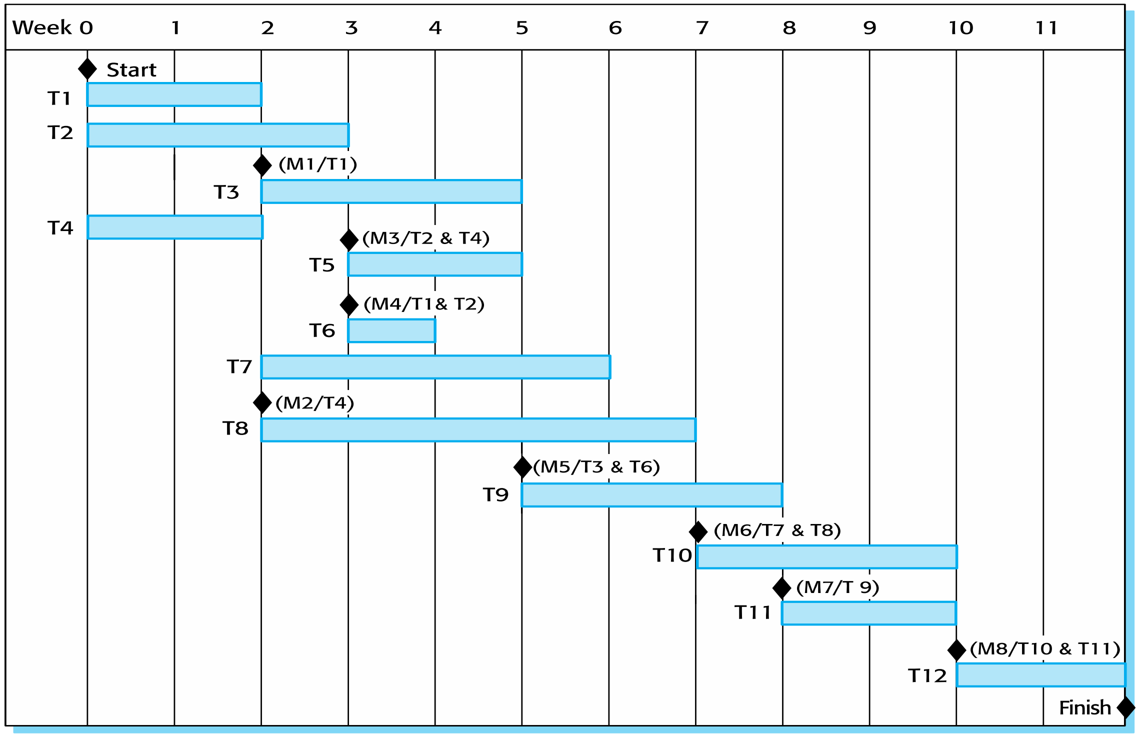
PERT Chart



1.2.1 Requirements Analysis
Start: 01.09.08 ID: 2
Finish: 19.09.08 Dur: 3 wks
Res:
1.2.2 Architectural Design
Start: 22.09.08 ID: 3
Finish: 03.10.08 Dur: 2 wks
Res:
1.2.3 Procedural Design
Start: 06.10.08 ID: 4
Finish: 22.10.08 Dur: 2,5 wks
Res:
1.2.4 Code
Start: 22.10.08 ID: 5
Finish: 14.11.08 Dur: 3,5 wks
Res:
1.3.1 Unit Test
Start: 04.11.08 ID: 7
Finish: 24.11.08 Dur: 3 wks
Res:
1.3.2 Integration Test
Start: 17.11.08 ID: 8
Finish: 12.12.08 Dur: 4 wks
Res:
1.3.3 Acceptance Test
Start: 15.12.08 ID: 9
Finish: 19.12.08 Dur: 1 wk
Res:
1.4.1 Packaging
Start: 22.12.08 ID: 11
Finish: 26.12.08 Dur: 1 wk
Res:
1.4.2 Customer Training
Start: 15.12.08 ID: 12
Finish: 16.01.09 Dur: 5 wks
Res:

GANTT Chart

- Also known as timeline chart
- Shows high level view of whole project
  - Horizontal bars show duration of tasks
  - Triangles show milestones
  - Vertical lines show dependencies



An example

- Let’s assume we identify the following tasks

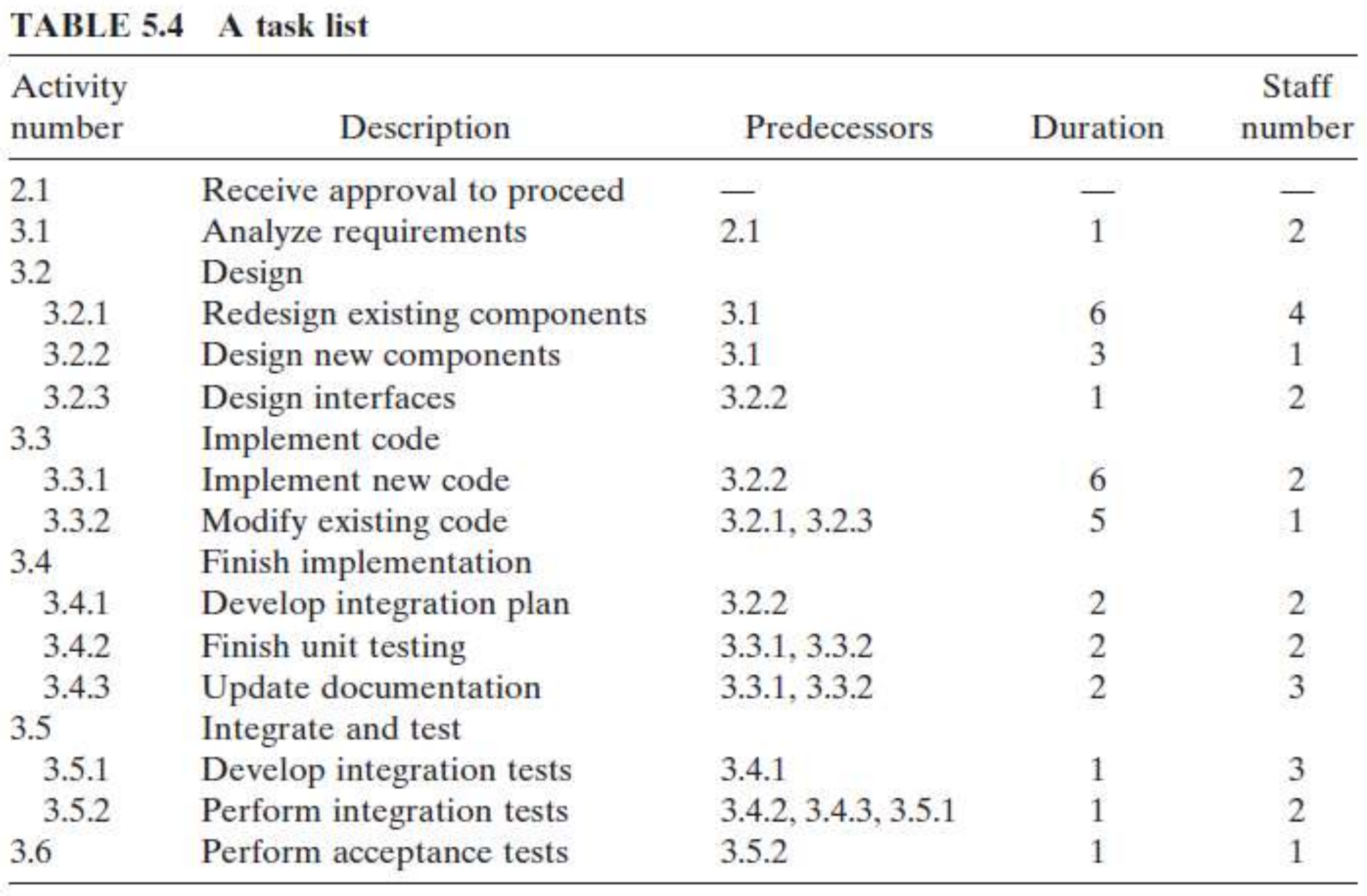


TABLE 5.4 A task list

Activity number	Description	Predecessors	Duration	Staff number
2.1	Receive approval to proceed	—	—	—
3.1	Analyze requirements	2.1	1	2
3.2	Design			
3.2.1	Redesign existing components	3.1	6	4
3.2.2	Design new components	3.1	3	1
3.2.3	Design interfaces	3.2.2	1	2
3.3	Implement code			
3.3.1	Implement new code	3.2.2	6	2
3.3.2	Modify existing code	3.2.1, 3.2.3	5	1
3.4	Finish implementation			
3.4.1	Develop integration plan	3.2.2	2	2
3.4.2	Finish unit testing	3.3.1, 3.3.2	2	2
3.4.3	Update documentation	3.3.1, 3.3.2	2	3
3.5	Integrate and test			
3.5.1	Develop integration tests	3.4.1	1	3
3.5.2	Perform integration tests	3.4.2, 3.4.3, 3.5.1	1	2
3.6	Perform acceptance tests	3.5.2	1	1

An example

- Following milestones can be extracted

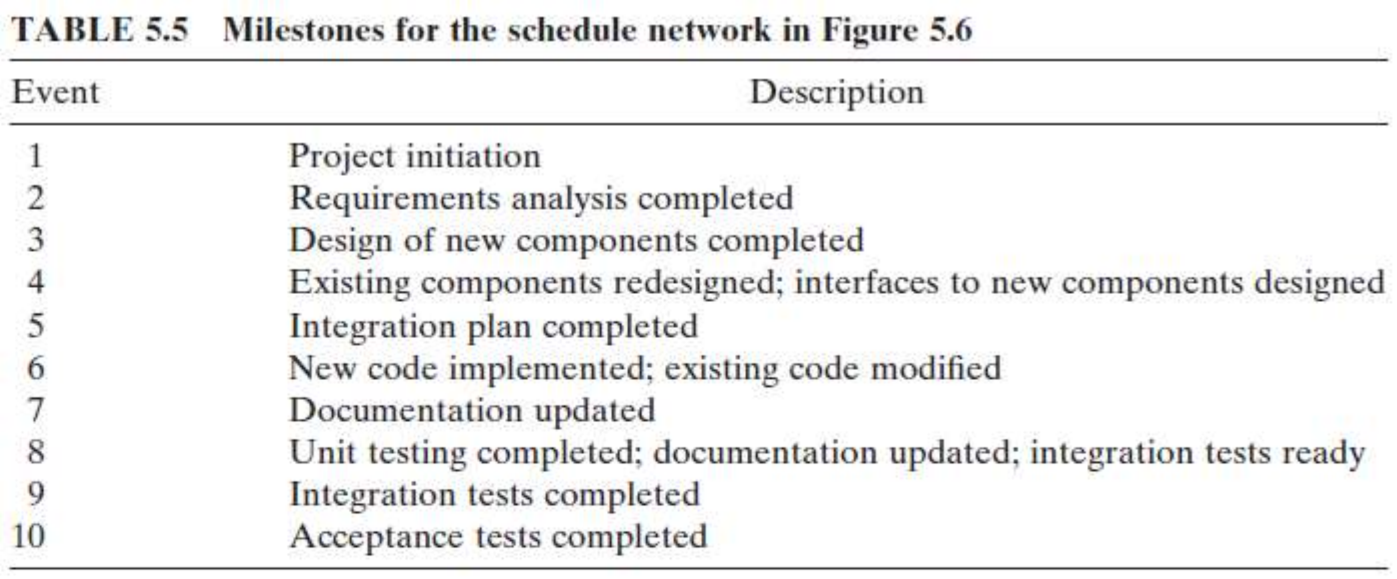


TABLE 5.5 Milestones for the schedule network in Figure 5.6

Event	Description
1	Project initiation
2	Requirements analysis completed
3	Design of new components completed
4	Existing components redesigned; interfaces to new components designed
5	Integration plan completed
6	New code implemented; existing code modified
7	Documentation updated
8	Unit testing completed; documentation updated; integration tests ready
9	Integration tests completed
10	Acceptance tests completed

An example

- Sometimes a task network is constructed to identify ***critical paths***.

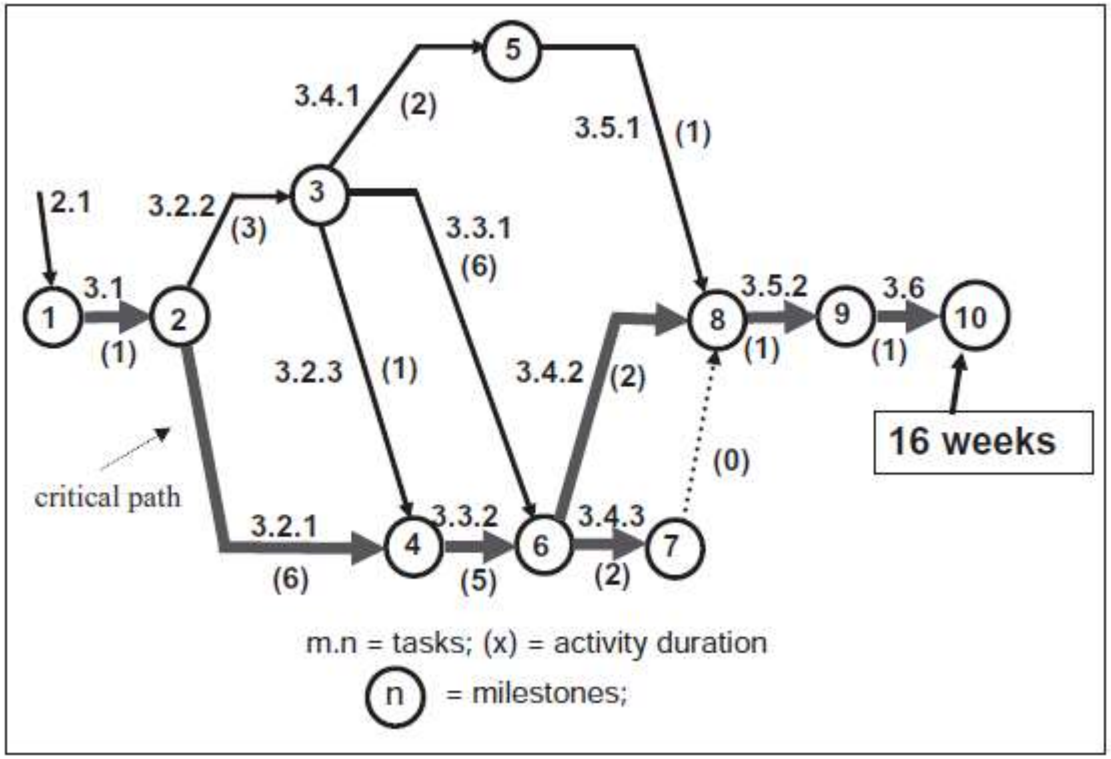
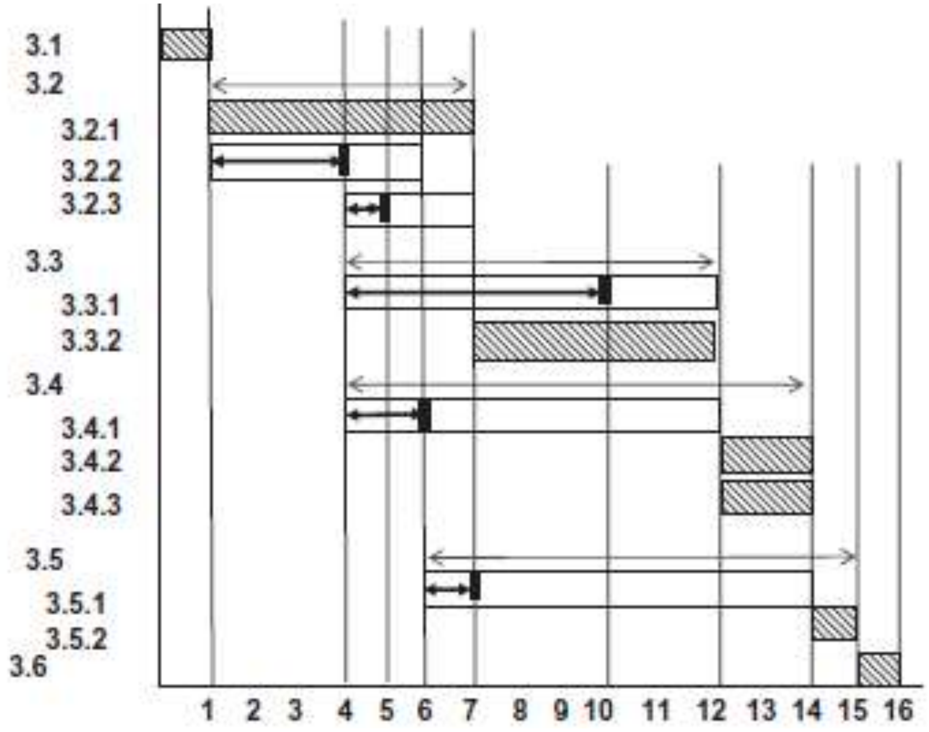


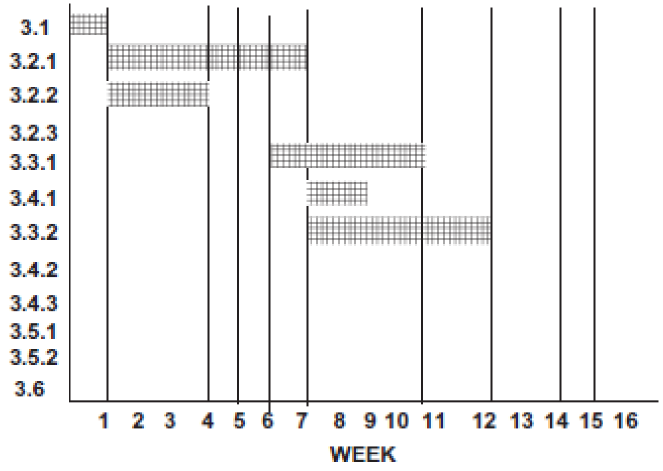
FIGURE 5.6 A critical-path activity network

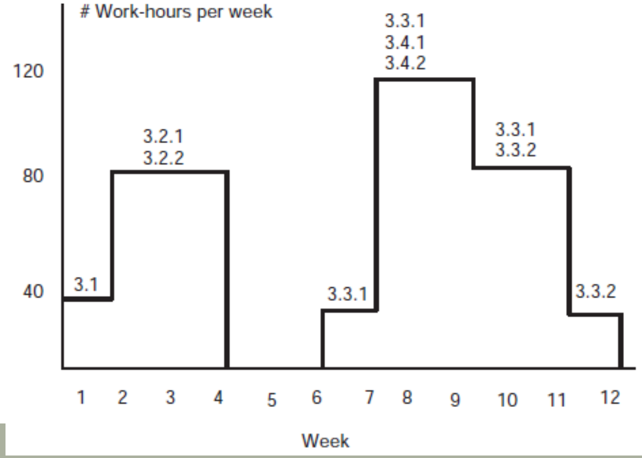
An example

- A final remark is to identify dependencies and pay attention to workloads



Resource-Gantt Chart for a developer





Project Plans

- In a plan-driven development project, a project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work.
- Plan sections
  - Introduction
  - Project organization
  - Risk analysis
  - Hardware and software resource requirements
  - Work breakdown
  - Project schedule
  - Monitoring and reporting mechanisms

A Typical Software Project Plan Document

1. Introduction
   A. Purpose of Plan
   B. Project Scope and Objectives
2. Project Estimates
   A. Historical Data Used for Estimates
   B. Estimation Techniques
   C. Estimates of Effort, Cost, Duration
3. Risks Management Strategy
   A. Risk Table
   B. Discussion of Risks to be Managed
   C. RMMM Plan
4. Schedule
   A. Project Work Breakdown Structure
   B. Task Network
   C. Timeline Chart
   D. Resource Table
5. Project Resources
   A. People
   B. Hardware and Software
   C. Special Resources
6. Staff Organization
   A. Team Structure
   B. Management Reporting
7. Tracking and Control Mechanisms
   A. Quality Assurance and Control
   B. Change Management and Control
8. Appendices

Risk Management

ℵ 4.1 ℶ

Risk Estimation

- *Risk projection*, also called *risk estimation,* attempts to rate each risk in two ways
  - the likelihood or probability that the risk is real
  - the consequences of the problems associated with the risk, should it occur.
- There are four risk projection steps:
  - establish a scale that reflects the perceived likelihood of a risk
  - define/describe consequences of a risk
  - estimate the impact of a risk on the project and the product,
  - note the overall accuracy of the risk projection so that there will be no misunderstandings.

Risk Table

Risks	Category	Probability %	Impact	Remedy Plan
Size estimate may be significantly low	PS	60	2	
Larger number of users than planned	PS	30	3	
Less reuse than planned	PS	70	2	
End users resist system	BU	40	3	
Delivery deadline will be tightened	BU	50	2	
Funding will be lost	CU	40	1	
Customer will change requirements	PS	80	2	
Technology will not meet expectations	TE	30	1	
Lack of training on tools	TE	60	3	
Staff inexperienced	ST	30	2	
Staff turnover will be high	ST	60	2	
…….				

- **Sort the table by probability and impact**
- **Cut-off low probability risks**

Examples of Different Risk Types

Risk type	Possible risks
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)

Risk Impact

- The overall *risk exposure,* RE, is determined using the following relationship

$$RE = P \times C$$

where

- P is the probability of occurrence for a risk
- C is the cost to the project should the risk occur

Risk Impact Example

- **Risk identification.** Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability.** 80% (likely).
- **Risk impact.** 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure.** RE = 0.80 x 25,200 ~ $20,200.

Risk Mitigation, Monitoring, Management

- Risk Mitigation,Monitoring,Management (RMMM)
- Mitigation : how can we avoid the risk?
- Monitoring : what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- Management :what contingency plans do we have if the risk becomes a reality?

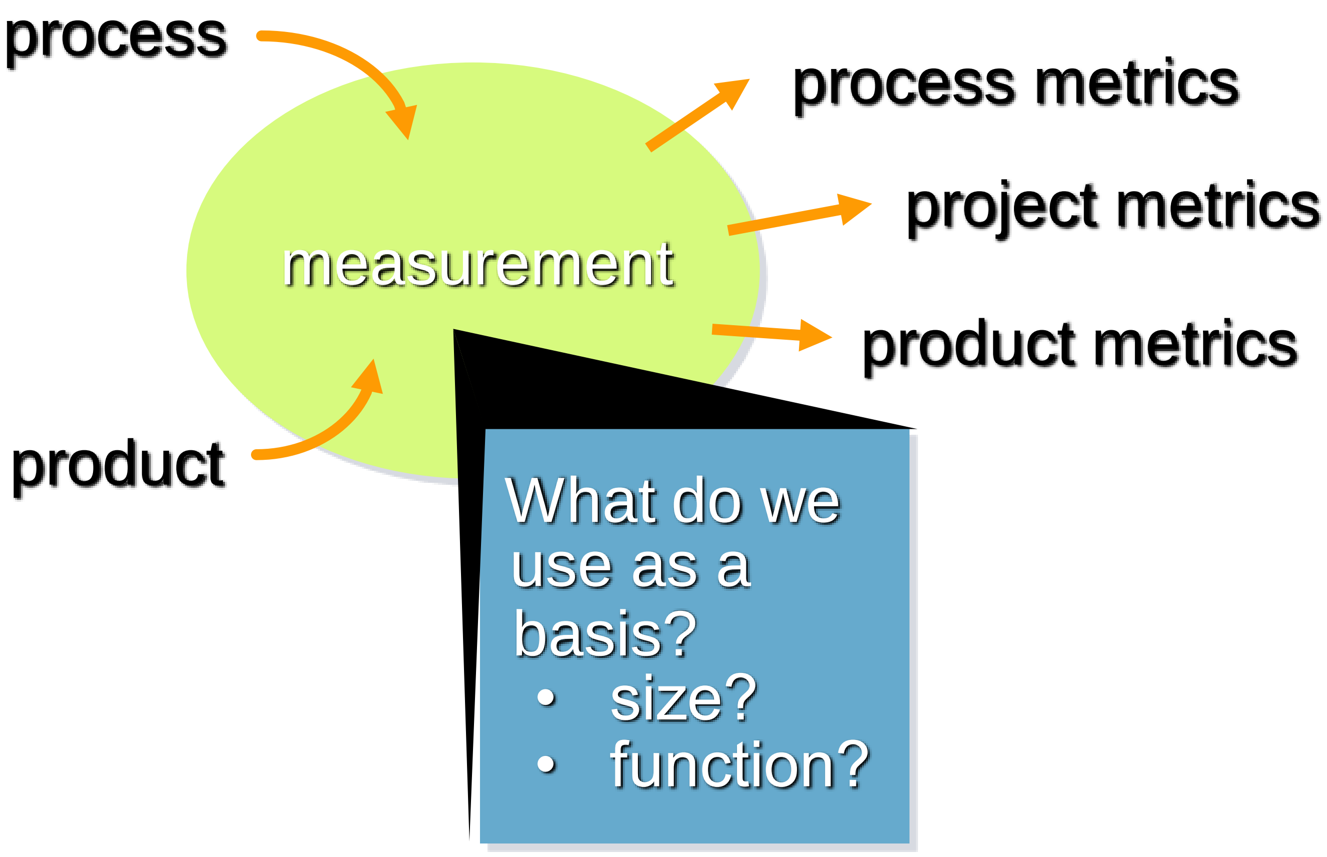
Examples of common risks

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

Quality Management and Metrics

4.2

Why Do We Measure?



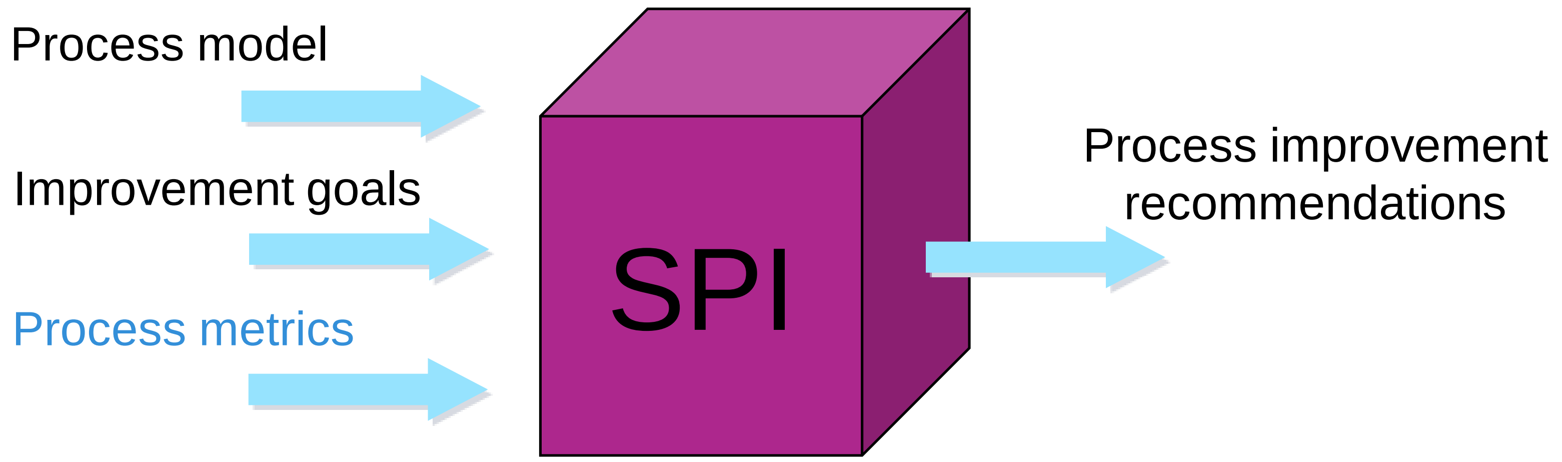
What do we use as a basis?

- size?
- function?

Why Do We Measure?

- assess the status of an ongoing project
- track potential risks
- uncover problem areas before they go “critical,”
- adjust work flow or tasks,
- evaluate the project team’s ability to control quality of software work products.

Software Process Improvement



Typical Project Metrics

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks

Examples:

- total FP estimation (Function Points)
- total LOC estimation (Lines of Code)
- FP per person-month
- LOC per person-month
- errors or defects per KLOC (thousand LOC)
- pages of documentation per KLOC

Object-Oriented Metrics

- Number of scenario scripts (use-cases)
- Number of support classes (required to implement the system but are not immediately related to the problem domain)
- Average number of support classes per key class (analysis class)
- Number of subsystems (an aggregation of classes that support a function that is visible to the end-user of a system)

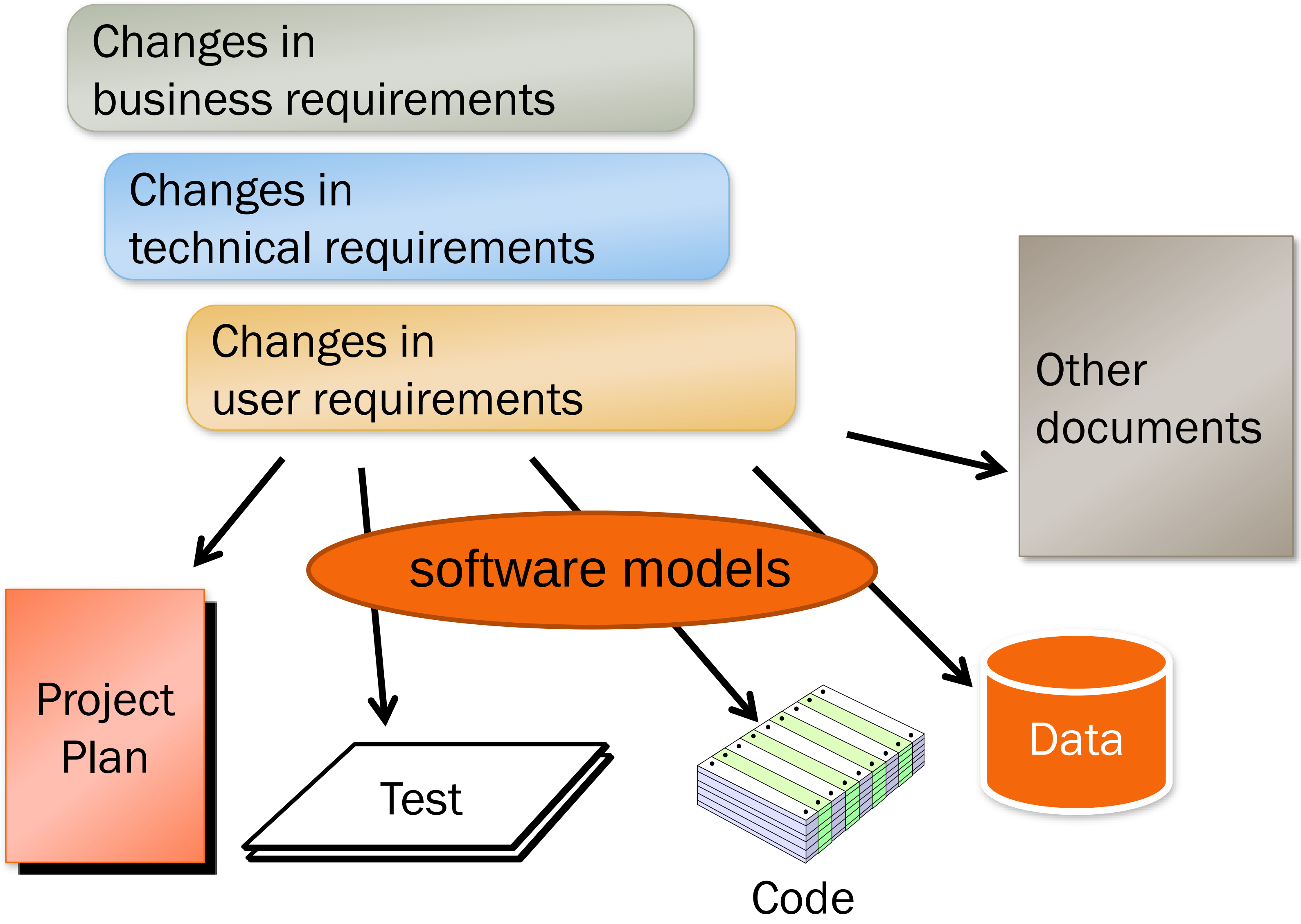
Web Project Metrics

- Number of static Web pages (the end-user has no control over the content displayed on the page)
- Number of dynamic Web pages (end-user actions result in customized content displayed on the page)
- Number of internal page links (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

Change Management

4.3

What are changes?


Changes in business requirements
Changes in technical requirements
Changes in user requirements
Other documents
software models
Project Plan
Test
Code
Data

Terminology

Term	Explanation
Configuration control	The process of ensuring that versions of systems and components are recorded and maintained so that changes are managed and all versions of components are identified and stored for the lifetime of the system.
Configuration item or software configuration item (SCI)	Anything associated with a software project (design, code, test data, document, etc.) that has been placed under configuration control. There are often different versions of a configuration item. Configuration items have a unique name.
Version	An instance of a configuration item that differs, in some way, from other instances of that item. Versions always have a unique identifier, which is often composed of the configuration item name plus a version number.
Baseline	A baseline is a collection of component versions that make up a system. Baselines are controlled, which means that the versions of the components making up the system cannot be changed. This means that it should always be possible to recreate a baseline from its constituent components.
Codeline	A codeline is a set of versions of a software component and other configuration items on which that component depends.

Terminology - II

Term	Explanation
Mainline	A sequence of baselines representing different versions of a system.
Release	A version of a system that has been released to customers (or other users in an organization) for use.
Workspace	A private work area where software can be modified without affecting other developers who may be using or modifying that software.
Branching	The creation of a new codeline from a version in an existing codeline. The new codeline and the existing codeline may then develop independently.
Merging	The creation of a new version of a software component by merging separate versions in different codelines. These codelines may have been created by a previous branch of one of the codelines involved.
System building	The creation of an executable system version by compiling and linking the appropriate versions of the components and libraries making up the system.

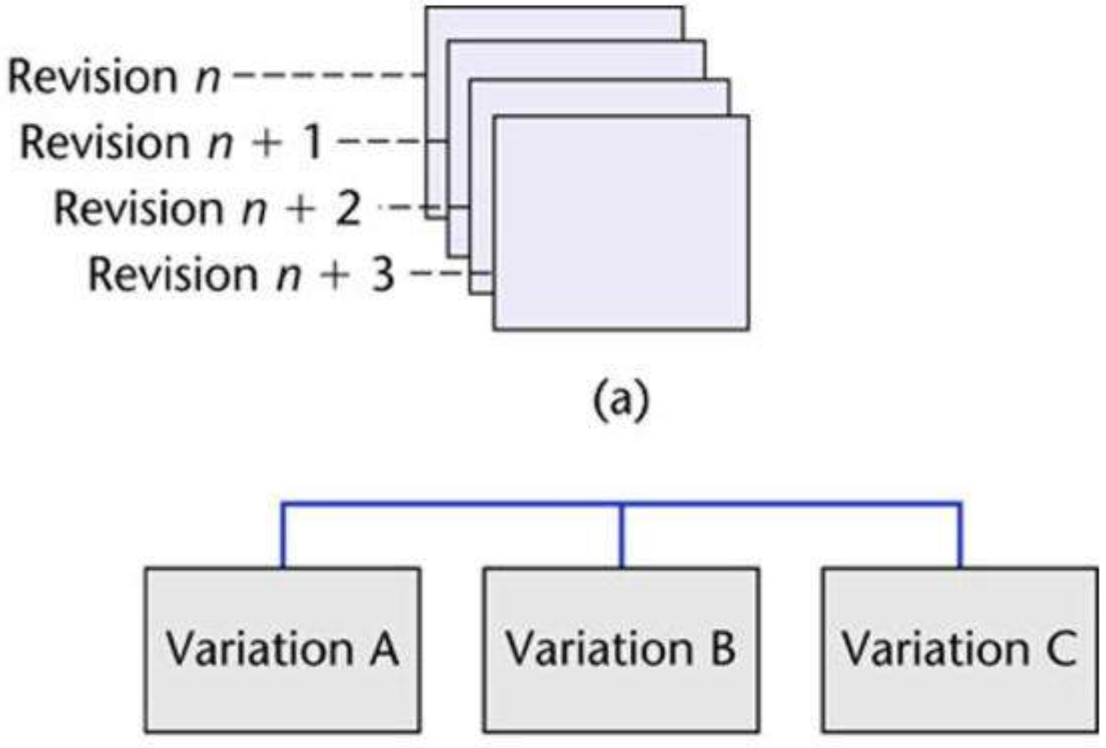
Versioning

- During maintenance, at all times there are at least two versions of the product:the old version, and the new version

- There are two types of versions: *revisions* and *variations*
- Revisions:
  - A version to fix a fault in the artifact
  - We cannot throw away an incorrect version
    - The new version may be no better
    - Some sites may not install the new version
  - Perfective and adaptive maintenance also result in revisions

Variations

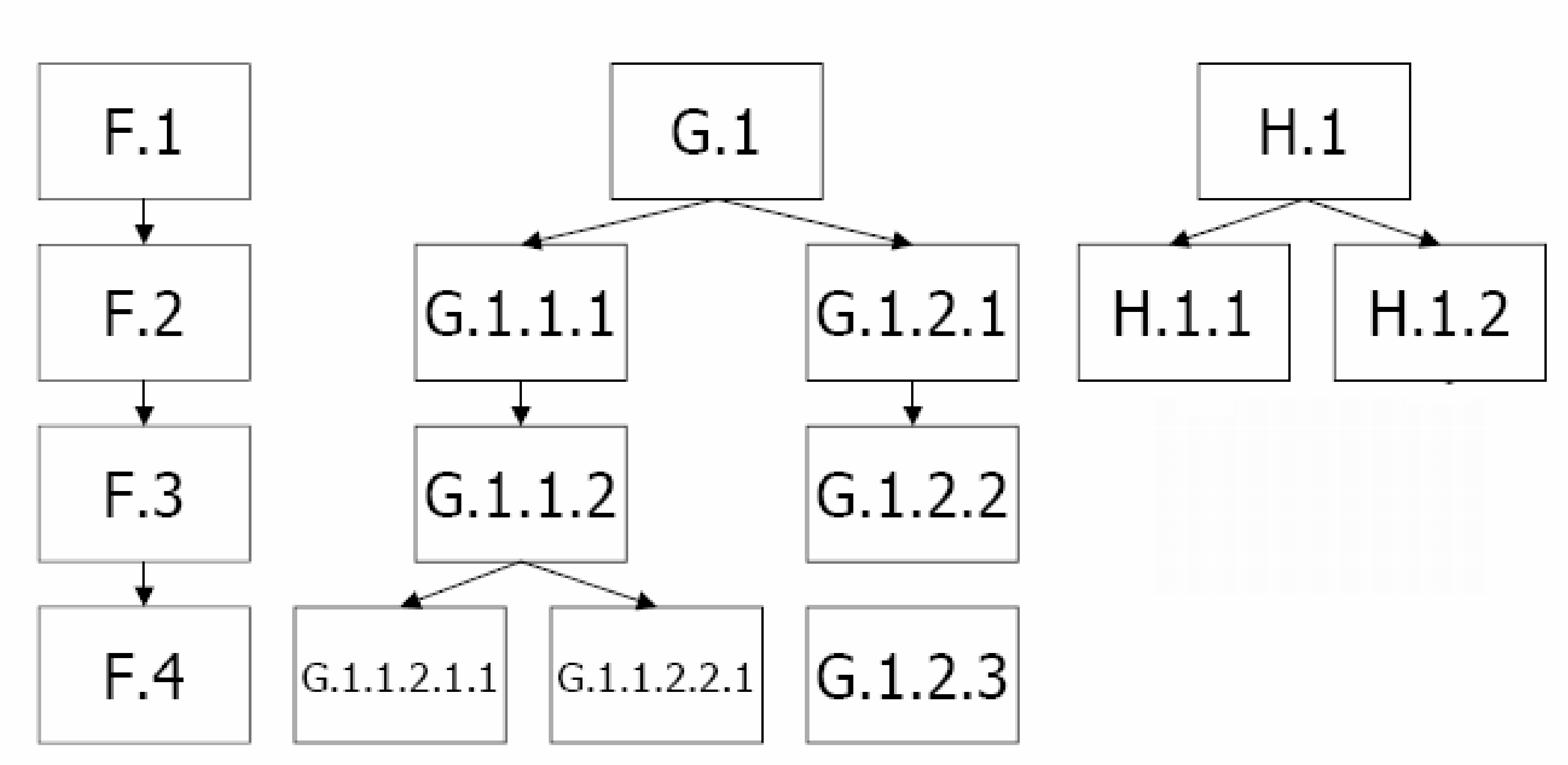
- **Variations:**
  - A variation is a version for a different operating system
  - Variations are designed to coexist in parallel



(a)

(b)

Baseline Versioning



Configuration-Control Tools

- Configuration-Control Tools
  - Gradle
  - Maven
  - Build.NET
- Versioning Tools
  - CVS
  - SVN
  - GIT
- Other commercial tools
  - TFS (Microsoft)
  - ClearCase (IBM Rational Software)

Wrap-up

This week we present

- Project Estimation: How to forecast about the project before the project goes underway
- Project Planning: What kind of activities should be perform to organize the project resources-activities in a plan-driven way
- Project Management: How can we measure-monitor-assess various indicators of project progress