Project Management

Lecture – 8

Outline of the talk

- Software project planning – an introduction.
- Function point analysis
- Numerical problems based on FPAs.
- Software project and process metrics.

Software Project Planning

In order to conduct a successful software project, we must understand:

- Scope of work to be done
- The risk to be incurred
- The resources required
- The task to be accomplished
- The cost to be expended
- The schedule to be followed

Software Project Planning

Software planning begins before technical work starts, continues as the software evolves from concept to reality, and culminates only when the software is retired.

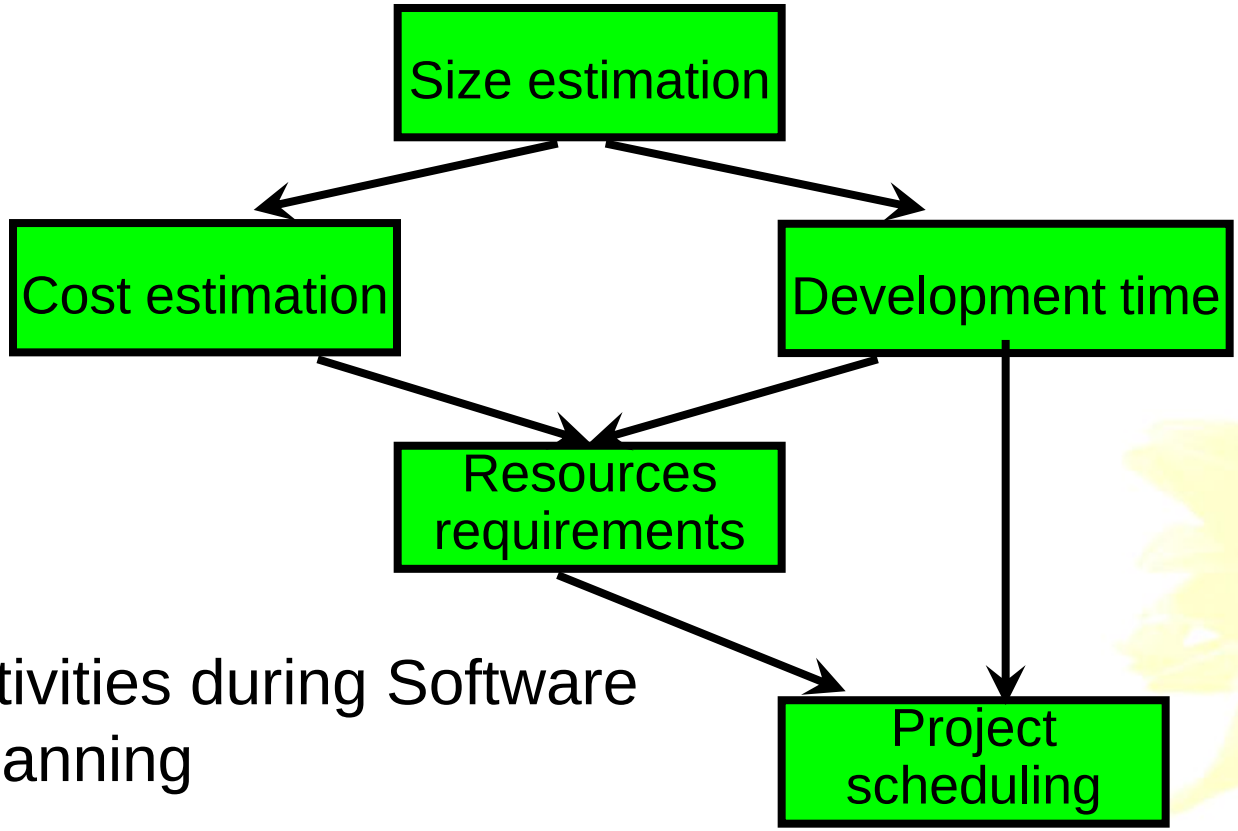


Fig. 1: Activities during Software Project Planning

Size Estimation

Lines of Code (LOC)

If LOC is simply a count of the number of lines then figure shown below contains 18 LOC .

When comments and blank lines are ignored, the program in figure 2 shown below contains 17 LOC.

Fig. 2: Function for sorting an array

```
1.  int. sort (int x[ ], int n)
2.  {
3.  int i, j, save, im1;
4.  /*This function sorts array x in ascending order */
5.  If (n<2) return 1;
6.  for (i=2; i<=n; i++)
7.  {
8.  im1=i-1;
9.  for (j=1; j<=im; j++)
10. if (x[i] < x[j])
11. {
12. Save = x[i];
13. x[i] = x[j];
14. x[j] = save;
15. }
16. }
17. return 0;
18. }
```

Software Project Planning

•Furthermore, if the main interest is the size of the program for specific functionality, it may be reasonable to include executable statements.

• The only executable statements in figure shown above are in lines 5-17 leading to a count of 13. The differences in the counts are 18 to 17 to 13.

•One can easily see the potential for major discrepancies for large programs with many comments or programs written in language that allow a large number of descriptive but non-executable statement. Conte has defined lines of code as:

Software Project Planning

"A line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line. This specifically includes all lines containing program header, declaration, and executable and non-executable statements".

This is the predominant definition for lines of code used by researchers. By this definition, figure shown above has 17 LOC.

Function Count

Alan Albrecht while working for IBM, recognized the problem in size measurement in the 1970s, and developed a technique (which he called Function Point Analysis), which appeared to be a solution to the size measurement problem.

Software Project Planning

The principle of Albrecht's function point analysis (FPA) is that a system is decomposed into functional units.

- Inputs : information entering the system
- Outputs : information leaving the system
- Enquiries : requests for instant access to information
- Internal logical files : information held within the system
- External interface files : information held by other system that is used by the system being analyzed.

Software Project Planning

The FPA functional units are shown in figure given below:

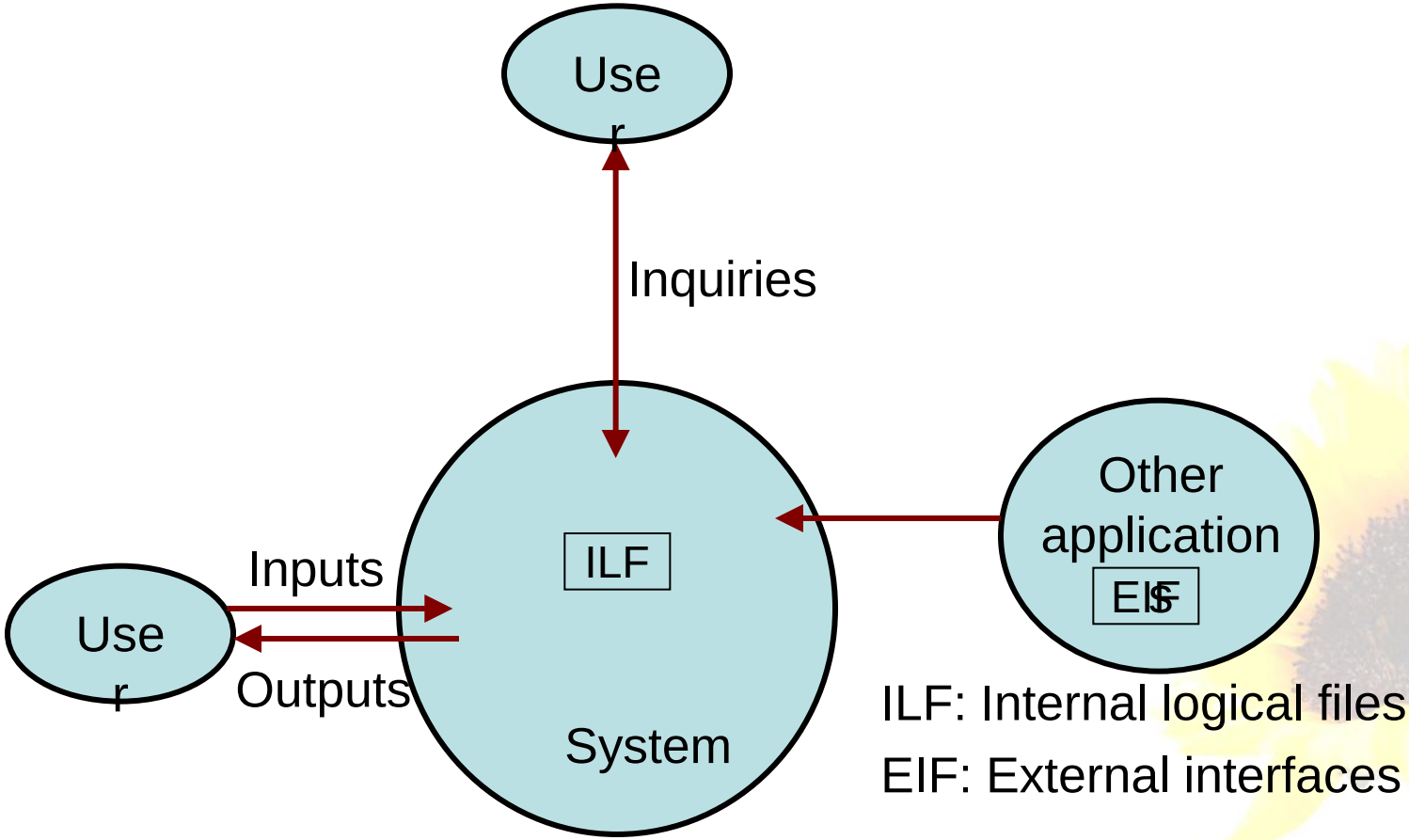


Fig. 3: FPAs functional units System

Software Project Planning

The five functional units are divided in two categories:

(i) Data function types

- Internal Logical Files (ILF): A user identifiable group of logical related data or control information maintained within the system.
- External Interface files (EIF): A user identifiable group of logically related data or control information referenced by the system, but maintained within another system. This means that EIF counted for one system, may be an ILF in another system.

(ii) Transactional function types

- External Input (EI): An EI processes data or control information that comes from outside the system. The EI is an elementary process, which is the smallest unit of activity that is meaningful to the end user in the business.
- External Output (EO): An EO is an elementary process that generate data or control information to be sent outside the system.
- External Inquiry (EQ): An EQ is an elementary process that is made up to an input-output combination that results in data retrieval.

Software Project Planning

Special features

- Function point approach is independent of the language, tools, or methodologies used for implementation; i.e. they do not take into consideration programming languages, data base management systems, processing hardware or any other data base technology.
- Function points can be estimated from requirement specification or design specification, thus making it possible to estimate development efforts in early phases of development.

Software Project Planning

- Function points are directly linked to the statement of requirements; any change of requirements can easily be followed by a re-estimate.
- Function points are based on the system user's external view of the system, non-technical users of the software system have a better understanding of what function points are measuring.

Counting function points

Functional Units	Weighting factors		
	Low	Average	High
External Inputs (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiries (EQ)	3	4	6
External logical files (ILF)	7	10	15
External Interface files (EIF)	5	7	10

Table 1 : Functional units with weighting factors

Software Project Planning

Table 2: UFP calculation table

Functional Units	Count Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)		Low x 3 =		
		Average x 4 =		
		High x 6 =		
External Outputs (EOs)		Low x 4 =		
		Average x 5 =		
		High x 7 =		
External Inquiries (EQs)		Low x 3 =		
		Average x 4 =		
		High x 6 =		
External logical Files (ILFs)		Low x 7 =		
		Average x 10 =		
		High x 15 =		
External Interface Files (EIFs)		Low x 5 =		
		Average x 7 =		
		High x 10 =		
Total Unadjusted Function Point Count				

Software Project Planning

The weighting factors are identified for all functional units and multiplied with the functional units accordingly. The procedure for the calculation of Unadjusted Function Point (UFP) is given in table shown above.

Software Project Planning

The procedure for the calculation of UFP in mathematical form is given below:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

Where i indicate the row and j indicates the column of Table 1

W_{ij} : It is the entry of the i^{th} row and j^{th} column of the table 1

Zij : It is the count of the number of functional units of Type *i* that have been classified as having the complexity corresponding to column *j*.

Software Project Planning

Organizations that use function point methods develop a criterion for determining whether a particular entry is Low, Average or High. Nonetheless, the determination of complexity is somewhat subjective.

$$FP = UFP * CAF$$

Where CAF is complexity adjustment factor and is equal to $[0.65 + 0.01 \times \Sigma F_i]$. The F_i (i=1 to 14) are the degree of influence and are based on responses to questions noted in table 3.

Software Project Planning

Table 3 : Computing function points.

Rate each factor on a scale of 0 to 5.

Number of factors considered (F_i)

1. Does the system require reliable backup and recovery ?
2. Is data communication required ?
3. Are there distributed processing functions ?
4. Is performance critical ?
5. Will the system run in an existing heavily utilized operational environment ?
6. Does the system require on line data entry ?
7. Does the on line data entry require the input transaction to be built over multiple screens or operations ?
8. Are the master files updated on line ?
9. Is the inputs, outputs, files, or inquiries complex ?
10. Is the internal processing complex ?
11. Is the code designed to be reusable ?
12. Are conversion and installation included in the design ?
13. Is the system designed for multiple installations in different organizations ?
14. Is the application designed to facilitate change and ease of use by the user ?

Software Project Planning

Functions points may compute the following important metrics:

Productivity	=	FP / persons-months
Quality	=	Defects / FP
Cost	=	Rupees / FP
Documentation	=	Pages of documentation per FP

These metrics are controversial and are not universally acceptable. There are standards issued by the International Functions Point User Group (IFPUG, covering the Albrecht method) and the United Kingdom Function Point User Group (UFPGU, covering the MK11 method). An ISO standard for function point method is also being developed.

Software Project Planning

Example: 4.1

Consider a project with the following functional units:

Number of user inputs = 50

Number of user outputs = 40

Number of user enquiries = 35

Number of user files = 06

Number of external interfaces = 04

Assume all complexity adjustment factors and weighting factors are average. Compute the function points for the project.

Solution

We know

$$UFP = \sum_{i=1}^{5}\sum_{J=1}^{3} Z_{ij} w_{ij}$$

UFP = 50 x 4 + 40 x 5 + 35 x 4 + 6 x 10 + 4 x 7

= 200 + 200 + 140 + 60 + 28 = 628

CAF = $(0.65 + 0.01 \Sigma F_i)$

= (0.65 + 0.01 (14 x 3)) = 0.65 + 0.42 = 1.07

FP = UFP x CAF

= 628 x 1.07 = 672

Example:4.2

An application has the following:

10 low external inputs, 12 high external outputs, 20 low internal logical files, 15 high external interface files, 12 average external inquiries, and a value of complexity adjustment factor of 1.10.

What are the unadjusted and adjusted function point counts ?

Solution

Unadjusted function point counts may be calculated using as:

$$UFP = \sum_{i=1}^{5} \sum_{J=1}^{3} Z_{ij} w_{ij}$$

= 10 x 3 + 12 x 7 + 20 x 7 + 15 + 10 + 12 x 4

= 30 + 84 +140 + 150 + 48

= 452

FP = UFP x CAF

= 452 x 1.10 = 497.2.

Software Project Planning

Example: 4.3

Consider a project with the following parameters.

(*i*) External Inputs:

(a)10 with low complexity

(b)15 with average complexity

(c)17 with high complexity

(*ii*) External Outputs:

(a)6 with low complexity

(b)13 with high complexity

(*iii*) External Inquiries:

(a) 3 with low complexity

(b) 4 with average complexity

(c) 2 high complexity

Software Project Planning

(*iv*) Internal logical files:

(a)2 with average complexity

(b)1 with high complexity

(v) External Interface files:

(a)9 with low complexity

In addition to above, system requires

i. Significant data communication

ii. Performance is very critical

iii. Designed code may be moderately reusable

iv. System is not designed for multiple installation in different organizations.

Other complexity adjustment factors are treated as average. Compute the function points for the project.

Software Project Planning

Solution: Unadjusted function points may be counted using table 2

Functional Units	Count	Complexity		Complexity Totals	Functional Unit Totals
External Inputs (EIs)	10	Low x 3	=	30	
	15	Average x 4	=	60	
	17	High x 6	=	102	192
External Outputs (EOs)	6	Low x 4	=	24	
	0	Average x 5	=	0	
	13	High x 7	=	91	115
External Inquiries (EQs)	3	Low x 3	=	9	
	4	Average x 4	=	16	
	2	High x 6	=	12	37
External logical Files (ILFs)	0	Low x 7	=	0	
	2	Average x 10	=	20	
	1	High x 15	=	15	35
External Interface Files (EIFs)	9	Low x 5	=	45	
	0	Average x 7	=	0	
	0	High x 10	=	0	45
Total Unadjusted Function Point Count					424

Software Project Planning

$$\sum_{i=1}^{14} F_i = 3+4+3+5+3+3+3+3+3+3+2+3+0+3=41$$

CAF = $(0.65 + 0.01 \times \Sigma F_i)$

= $(0.65 + 0.01 \times 41)$

= 1.06

FP = UFP x CAF

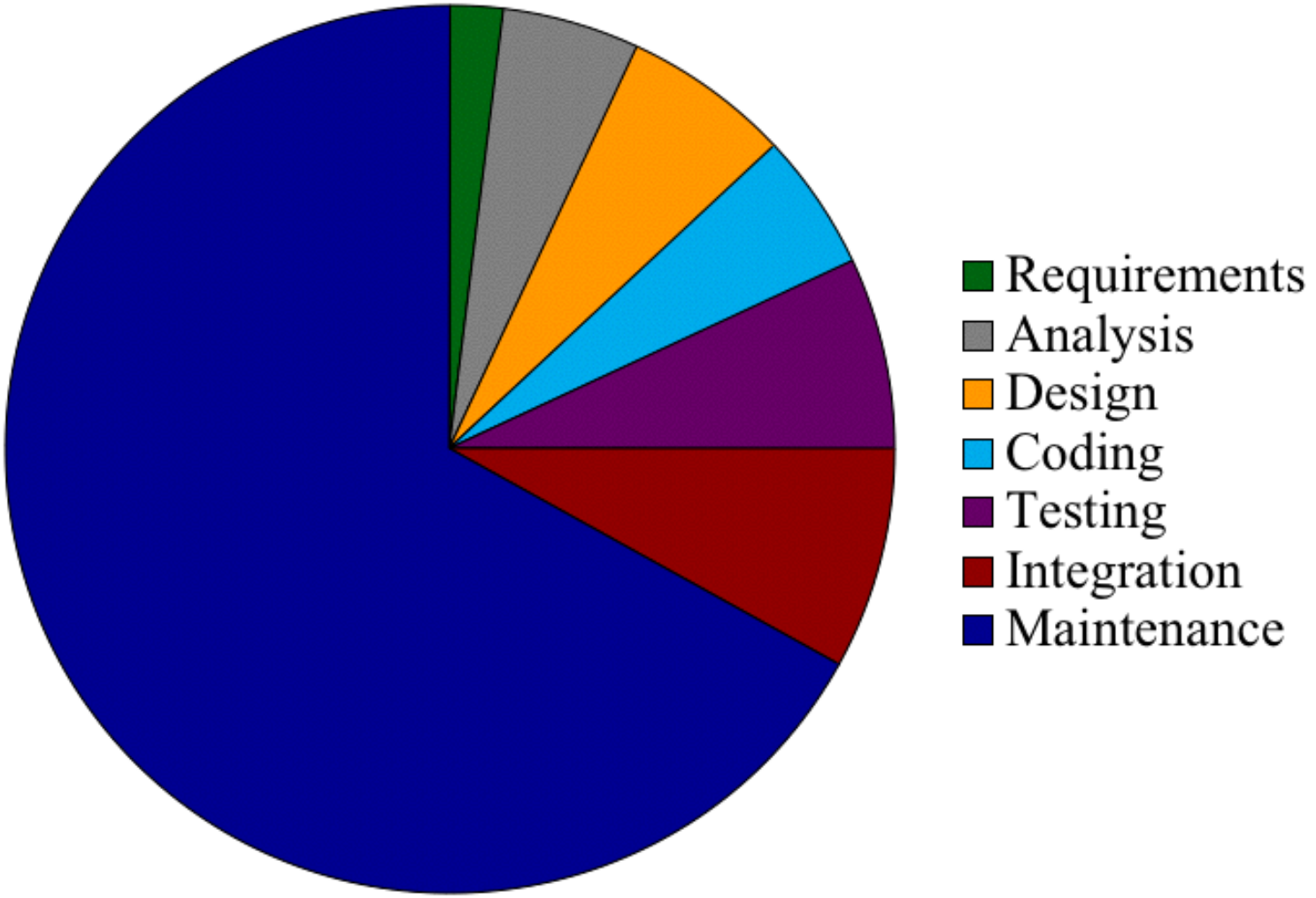
= 424 x 1.06

= 449.44

Hence FP = 449

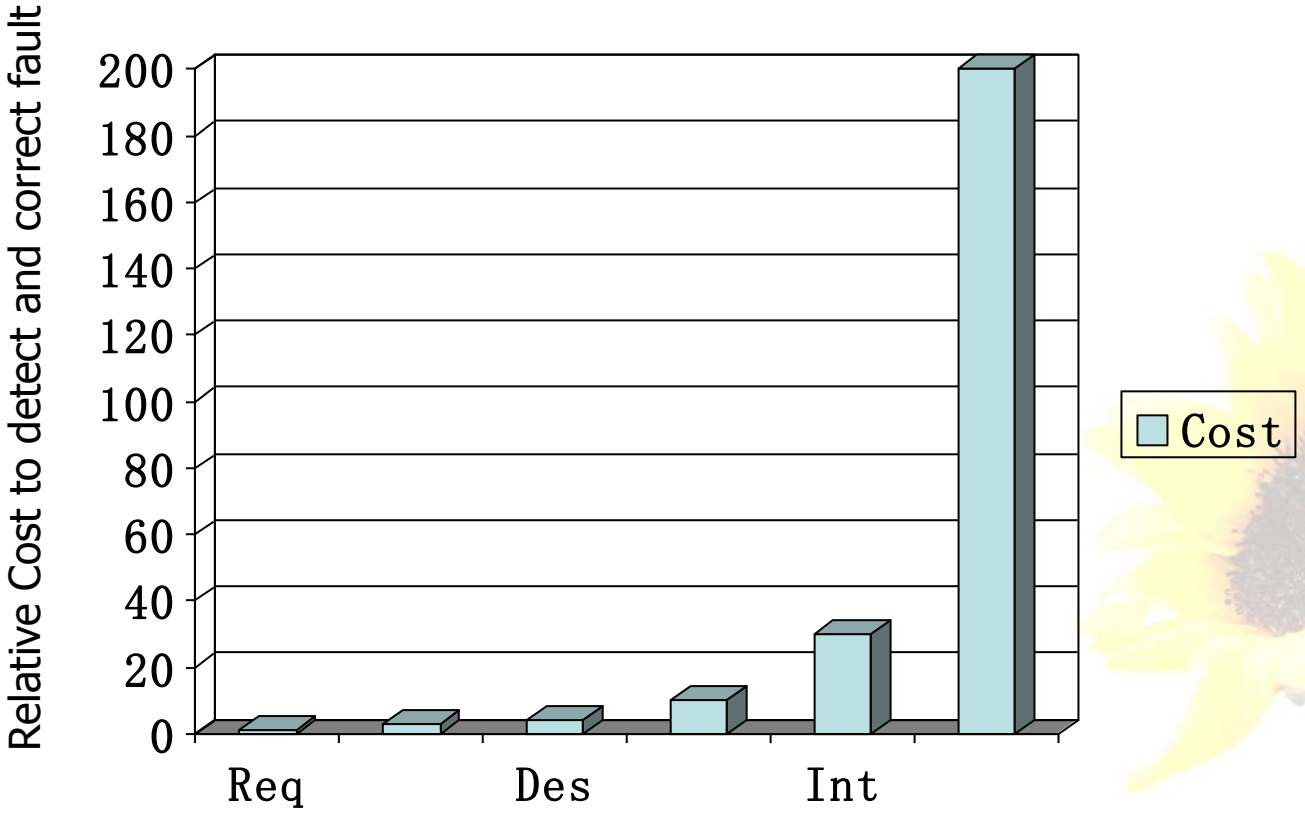
Software Project Planning

Relative Cost of Software Phases



Software Project Planning

Cost to Detect and Fix Faults



Software Project Planning

Cost Estimation

A number of estimation techniques have been developed and are having following attributes in common :

- Project scope must be established in advance
- Software metrics are used as a basis from which estimates are made
- The project is broken into small pieces which are estimated individually

To achieve reliable cost and schedule estimates, a number of options arise:

- Delay estimation until late in project
- Use simple decomposition techniques to generate project cost and schedule estimates
- Develop empirical models for estimation
- Acquire one or more automated estimation tools

Software Process and Project Metrics

- Outline:
  - In the Software Metrics Domain:
    - product metrics
    - project metrics
    - process metrics
  - Software Measurement
    - size-oriented metrics
    - function-oriented metrics
  - Metrics for Software Quality

Measure, Metrics, and Indicator

- **Measure** -- Provides a quantitative indication of the extent, amount, dimensions, capacity, or size of some product or process attribute.
- **Metrics** -- A quantitative measure of the degree to which a system, component, or process possesses a given attribute.
- **Software Metrics** -- refers to a broad range of measurements for computer software.
- **Indicator** -- a metric or combination of metrics that provide insight into the software process, a software project, or the product itself.

In the Process and Project Domains

- Process Indicator
- - enable insight into the efficiency of an existing process
  - to assess the current work status
  - Goal -- to lead to long-term software process improvement
- Project Indicator
  - assess the status of an ongoing project
  - track potential risks
  - uncover problem areas before they “go critical”
  - evaluate the project team’s ability to control the product quality

Measurement

- What to measure?
  - errors uncovered before release
  - defects delivered to and reported by end users
  - work products delivered
  - human effort expended
  - calendar time expended
  - schedule conformance
- At what level of aggregation?
  - By team?
  - Individual?
  - Project?

Project Metrics

- Software Project Measures Are Tactical
  - used by a project manager and a software team
  - to adapt project work flow and technical activities
- The Intent of Project Metrics Is Twofold
  - to minimize the development schedule to avoid delays and mitigate potential problems and risks
  - to assess project quality on an ongoing basis and modify the technical approach to improvement quality
- Production Rates
  - pages of documentation
  - review hours
  - function points
  - delivered source lines
  - errors uncovered during SW engineering

Software Metrics

- Direct measures
  - Cost and effort applied (in SEing process)
  - Lines of code(LOC) produced
  - Execution speed
  - CPU utilization
  - Memory size
  - Defects reported over certain period of time
- Indirect Measures
  - Functionality, quality, complexity, efficiency, reliability, maintainability.