CHAPTER TWO

Software Metric

- Measures
- Metric
- Indicators
- Software Management
- Metric for software quality
- Statistical Quality Control
- Metric for small Organizations

Software Metrics

- Measure : A measure provides a **quantitative indication** of the extent, amount, dimension, capacity , or size of some attribute of a product or process
- Metric : Metric as " a **quantitative measure** of the **degree to** which a system, component, or process possesses a given attribute.
- Indicator: An indicator is a metric or condition of the metrics that provides **insight** into the software process, a software project, or the product itself.

Software Metrics

- Metrics strongly support software project **management activities**
- They relate to the four functions of management as follows:
- **Planning**
- **Organizing**
- **Controlling**
- **Improving**

Size Estimation Metrics

- Size of a program is not the number **of bytes** that the source code occupies
- It is **not the size** of the executable code
- It is an indicator of the effort and time required to develop the program.
- Size of program indicates development complexity
- Estimating the problem size is fundamental to estimating **the effort, time, and cost** of planned software.

Software Measurement

- **Direct measure** and **indirect measure.**
- Direct measures of the software include how many lines of **code (LOC) produced**, **execution speed, memory size, and defects reported**.

- Indirect measures include functionality, quality, complexity, efficiency, reliability, and maintainability of the software.

Lines of Code

- The simplest among all metrics available to estimate project size
- Project size estimated by counting the number of source instructions
- Lines used for commenting, header lines ignored
- To find LOC at the beginning of a project divide module into sub modules and so on until size of each module can be predicted

Disadvantages of LOC

- Gives a numerical value of problem size that vary widely with individual coding style

```
If( x>y )
  then x++;
else
  y++;
```

```
x > y ? x++ : y++;
```

- Effort needed for analysis, design , coding, testing etc (not just coding)

Disadvantages of LOC

- # Larger Code size → Better Quality?
- # Logical Complexity?

  Complex Logic -> More Effort

  Simple Logic -> Less Effort

```
while(i<4) {
  printf("testing");
}
```

```
printf("testing");
printf("testing");
printf("testing");
printf("testing");
```

Disadvantage of LOC

- Accurate computation of LOC only after project completion!!

Lines of code

- **1. Cost per line of code** = Labor rate / Productivity
- **2. Estimated project cost** = Estimated line of code*Cost per line of code
- **3. Estimate labor effort** = Estimated line of code / Productivity

Example Of LOC

- Estimated line of code = 33,200
- Productivity = 620 LOC/PM
- Labor Rate = $ 8000/PM
- Cost per line of code = ?
- Estimated Project cost = ?
- Estimate Labor Effort = ?

Example of LOC

- Cost per line of code = $8000/620

  = $13

- Estimated project cost = 33,200 * 13

  = $431,600

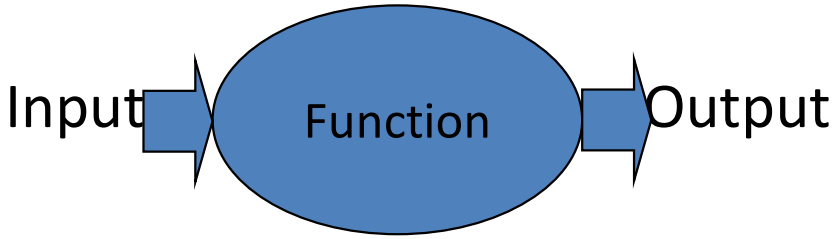
- Estimated Labor effort = LOC / productivity

  = 33,200/620

  = 54 PM

Function Point Metric

- Size of software product computed directly **from problem specification**
- Size of software = number of different functions/ features it supports



- Many features  Larger size
- Apart from that size depends on
  - number of files
  - number of interfaces
  - number of enquiries

Functional Point

Size of Function Point (FP)= Weighted sum of these five problem characteristics

1.**Number of inputs**: Data items input by user

(Group of user inputs taken together)

» Employee
- Name
- Age
- Sex
- Address

Account
- Account Name
- Account Number
- Account Open Date

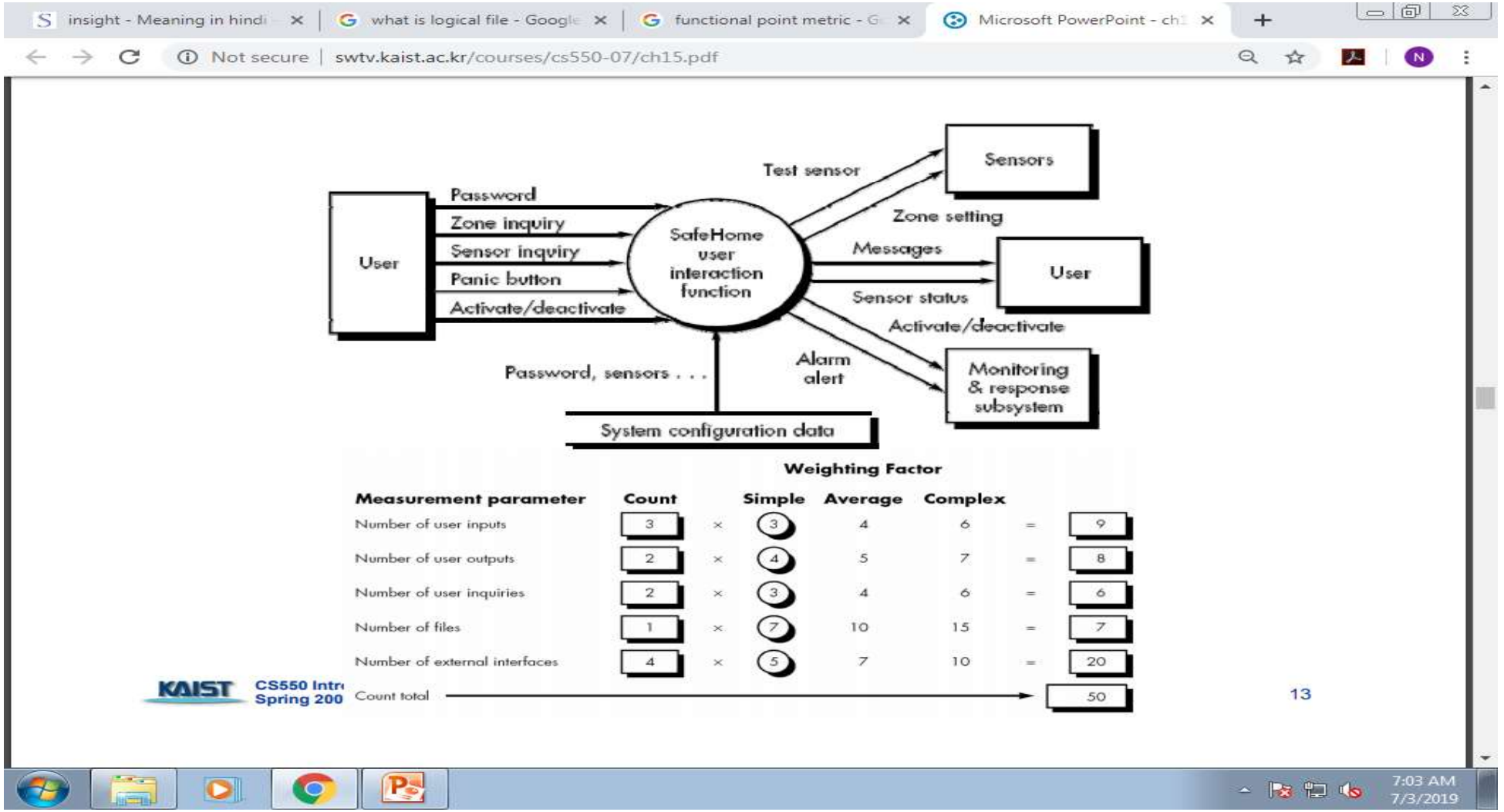
Functional Point Metric

2. **Number of Outputs**: Reports, Screen outputs, Error Messages
3. **Number of inquiries**: Interactive queries made by users
4. **Number of Files**: Logical files e.g. data structures, physical files
5. **Number of interfaces**: Interfaces for exchanging information e.g. disk, tapes, communication links

Functional Point

Measurement parameter	Count		Weighting factor Simple	Average	Complex		
Number of user inputs		x	3	4	6	=	
Number of user outputs		x	4	5	7	=	
Number of user inquiries		x	3	4	6	=	
Number of files		x	7	10	15	=	
Number of external interfaces		x	5	7	10	=	
Count total							

Functional Point Metric



Functional point

Information domain value	Opt.	Likely	Pess.	Est. count	Weight	FP count
Number of external inputs	20	24	30	24	4	97
Number of external outputs	12	15	22	16	5	78
Number of external inquiries	16	22	28	22	5	88
Number of internal logical files	4	4	5	4	10	42
Number of external interface files	2	2	3	2	7	15
Count total						*320*

Factor	Value
Backup and recovery	4
Data communications	2
Distributed processing	0
Performance critical	4
Existing operating environment	3
Online data entry	4
Input transaction over multiple screens	5
Master files updated online	3
Information domain values complex	5
Internal processing complex	5
Code designed for reuse	4
Conversion/installation in design	3
Multiple installations	5
Application designed for change	5
Value adjustment factor	**1.17**

Function point

Limitation of function point metric

- The weight of items on metric is fixed which may not sufficient for all cases.

Function Point

To compute function points (FP), the following relationship is used:

- **FP = count total * [0.65 + 0.01 *∑(Fi)]**
- The Fi (i 1 to 14) are value adjustment factors (VAF) based on responses to the following questions

Function Point

- **These 14 questions are scaled to 0 to 5 where,**

- *0 = No influence or No importance, 1= Incidental, 2= moderate, 3= Average, 4 = Significant, 5 = Essential*

Function Point

1. Does the system require **reliable backup and recovery**?
2. Are specialized data communications required to transfer information to or from the application?
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 online data entry?
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated online?

Function Point

9. Are 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?

Function Point

- Example : With the given data for an online shopping site developed by ABC software developers
- Number of user input = 98
- Numbers of user Output = 51
- Number of User Inquires = 47
- Number of External Interfaces = 32
- Number of Logical Files = 61
- Assuming that the complexity of the given website development is average, compute the function point if the productivity of the ABC S/W developers is 35 FP/PM and their salary structure is Rs. 1500 per month on average, estimate total cost of the software .

Function Point

- Formula for Function Point
- **Cost per FP** = labor rate/productivity
- **Estimated project cost** = *estimated fP* cost per Fp*
- **Estimated Labor effort** = *Estimated FP/Productivity*

Function Point

- FP = 672
- Organizational productivity = 6.5 FP/pm
- Labor Rate per month = $8000
- Cost per FP = $8000/6.5 = $1230
- Effort = FP/Productivity = 672/6.5 =103 PM
- Total project cost = (672*1230) = $826560

Functional Point

- Expected Value for estimate variable size (s) = ?
- Make optimistic , most likely, pessimistic , estimate for each item, then compute expected value .

$$S = \frac{S_{opt} + 4S_m + S_{pess}}{6}$$

Functional Point

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
- Assuming all complexity adjustment factors and weighing factors as average, the function points for the project will be;

Functional Point

- Characteristics for weights are

0 = No influence, 1= Incidental, 2= moderate,3= Average, 4 = Significant, 5 = Essential

Functional Point

- **Function point (FP)= UFP x VAF**

Where ,UFP = Unadjusted function point ,

VAF =Value Adjustment Factor

FP

- $FP_{estimated}$ = 672
- Organizational productivity = 6.5 FP/PM
- Labor Rate = $8000 /PM
- Cost per FP = $8000/6.5 = $1230
- Effort = $Fp_{estimated/}$Productivity = 672/6.5 =103 PM
- Total project cost = (672*1230) = $826560

FP Problem (Class Work)

- Assuming that the complexity of the NCIT MIS software development is average, compute the function point for it with the given data:

  Number of User Input : 95

  Number of User Output : 55

  Number of Inquiries : 4

  Number of logical Files : 66

  Number of External Interfaces:27

- If the productivity of the software developers is 30 FP/PM and their salary structure is RS.18000 per month on average, estimate the total cost of the software.

Q. No 1(b)

Given. Number of user inputs = 28
Number of user outputs = 44
Number of user inquiries = 7
Number of files = 3
Number of external interfaces = 2
Effort = 37 P-M
Technical document = 360 pages
user document = 129 pages
cost = Rs 8000 P-M

Information domain	count	weighing factor	FP count
no. of user inputs	28	4	112
no of user outputs	44	5	220
no of user inquiries	7	4	28
no of files	3	10	30
no of external interface	2	7	14
			404

Raw FP (count total) = 404

$\Sigma fi = 4+1+1+3+5+5+4+4+3+3+2+3+4+5$
$= 47$

Function point (FP) : Raw FP $\times [0.65 + (0.01 \times \Sigma fi)]$
$= 404 \times [0.65 + (0.01 \times 47)]$
$= 452.48$
~ 453

Productivity = $\frac{FP}{effort}$ = $\frac{453}{37}$ = 12.24/P-M

Total pages of documentation = 360 + 129
= 489

Documentation = $\frac{\text{Pages of documentation}}{FP}$

= $\frac{489}{453}$

= 1.0794

Cost per function = $\frac{\text{cost}}{\text{productivity}}$

= $\frac{8000}{12.24}$

= Rs 653.59 per function

=

Categories of Metrics

- Product Metrics
- Process Metrics
- Project Metrics

Characteristics of Metrics

- Product Metrics : Product metric describes the characteristics of the product, such as **size**, **complexity, performance, efficiency.**

Characteristics of Metrics

Process Metrics: Process metric describe **effectiveness and quality** of the process. E.g

- **Effort** required in the process
- **Time** to produce the product
- Number of **defects** found during testing

Categories of Metrics

Project Metrics: Project metrics describe the project characteristics and execution. E.g

- Number of software developer
- Staffing pattern over the life cycle of the software
- Cost and schedule
- Productivity

Attribute of Effective Software Metrics

1. Simple and computable
2. Empirically and Intuitively Persuasive (satisfy Engineers' intuitive)
3. Consistent and Objective
4. Consistent in the use of units and dimensions
5. Programming Language Independent
6. An Effective Mechanism for High Quality Feedback

Metric for Software Quality

- Software quality can be measured through the software Engineering process, **before release** to customer and **after release** to the customer
- The main goal of software engineering is to produce a high-quality system.
- A good software Engineer and good software engineering must measure if **high quality** is to be realized .

Metric for software quality

Measuring Quality

- Indicators to measure the quality
  - Correctness
  - Maintainability
  - Integrity (loyally functioning or not)
  - Usability (user friendly or not)

Defect Removal Efficiency

- A quality metric that provides **benefit at both the project and process** level is defect removal efficiency (DRE)
- DRE is computed as

DRE = E/E+D

Where E = Errors found before delivery of the software

D = Defects found after the delivery

Metric for small Organization

Organization might select the following set of easily collected measures:

- Time (hours or days) elapsed from the time a request is made until evaluation is complete, t_{queue}.
- Effort (person-hours) to perform the evaluation, W_{eval}.
- Time (hours or days) elapsed from completion of evaluation to assignment of

Metric for small Organization

- comprehensive software metrics programs
- software organizations of all sizes measure
- use the resultant metrics to help improve their local software.

Metric for small Organization

- change order to personnel, *t*eval.
- Effort (person-hours) required to make the change, *W*change.
- Time required (hours or days) to make the change, *t*change.
- Errors uncovered during work to make change, *E*change.
- Defects uncovered after change is released to the customer base, *D*change.

Metric for Small organization

- Defect removal efficiency

$$DRE = \frac{E_{change}}{E_{change} + D_{change}}$$

Project Estimation Technique

- Empirical estimation technique
- Heuristic Technique
- Analytic estimation technique

Project Estimation Technique

- **Empirical Estimation** : making an educated guess of the project using past experience e.g Delphi and Expert judges
- Heuristic Techniques: Project parameter can be modeled by the mathematical expression.
- Analytic Estimation : Like Heuristic Technique but supports scientific facts .

COCOMO II MODEL

- The constructive cost model (COCOMO) is an algorithmic software cost estimation model developed by Barry Boehm
- The model uses a basic regression formula, with some parameters that are derived from historical project data and current project characteristic

COCOMO II MODEL

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
- **Application composition model**: Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
- **Early design stage model**: Used once requirements have been stabilized and basic software architecture has been established.
- **Post-architecture-stage model**: Used during the construction of the software.

Basic COCOMO-II Model

- COCOMO applies to three classes of software projects:
  - **Organic:** Developing well understood application programs, small experienced team
  - **Semi Detached**: mix of experienced and non-experienced team
  - **Embedded**: strongly coupled to computer hardware

- Basic COCOMO
  - Effort = a (KLOC)b PM
  - Time = c (Effort)d Months
  - Number of people required = (Effort applied) / (Development time)

Example: The size of organic software is estimated to be 32,000 LOC. The average salary for software engineering is Rs. 15000/- per month. What will be effort and time for the completion of the project?

Solution:

- **Effort applied** = 2.4 x $(32)^{1.05}$ PM = 91.33 PM (Since: 32000 LOC = 32KLOC)
- Time = 2.5 x $(91.33)^{0.38}$ Month = 13.899 Months
- **Cost** = Time x Average salary per month = 13.899 x 15000 = Rs. 208480.85
- **People required** = (Effort applied) / (development time) = 6.57 = 7 persons

Software Risk

- Anticipated unfavorable event
- When risk turns to reality it hampers successful and timely completion of project

Risk

Basic Characteristics of Risks:

Uncertainty: **Risk** may or may not h**appen**

- *Loss*: **If risk becomes a reality** *losses* **will occur**

RISK

categories of risks are:

a. **Project Risk:**
b. **Technical Risk** :
c. **Business Risk** :
d. **Known Risk:**
e. **Predictable Risk :**
f. **Unpredictable Risk** :

Project Risk

- ***Project risks*** threaten the project plan.
- If project risks become real, it is likely that the project schedule will slip and that costs will increase.
- Project risks identify potential budgetary, schedule, personnel (staffing and organization), re- source, stakeholder, and requirements problems and their impact on a software project.

Technical Risk

- Technical risks identify potential design, implementation, interface, verification, and maintenance problems.
- In addition, specification ambiguity, technical uncertainty, technical obsolescence.

Business Risk

Top Five **Business risk** are

- **Market risk**: system that no one really wants
- **Strategic risk**: product no longer fits into the overall business strategy
- **Sales risk**: the sales force doesn't understand how to sell the product
- **Management risk:** losing the support of senior management

- **Budget risks:** losing budgetary or personnel commitment

Known Risk

- *Known risks* are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).

Predictable Risk

- *Predictable risks* are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer).

Unpredictable

- *Unpredictable risks* are the joker in the deck.
- They can and do occur, but they are extremely difficult to identify in advance.

RISK

- **Reactive and Proactive Risk Strategy**

Proactive Approach

- Being proactive means that you identify risks before they happen and figure out ways to avoid or alleviate the risk.
- Proactive risk management seeks to reduce the risk potential of the hazard or even better prevent the threat altogether.

Reactive Approach

- The reactive approach is stressful and costly as the management makes the decisions as the events unfold.

RISK MANAGEMENT PROCESS

- **Risk identification**

– Identify project, product and business risks

- **Risk analysis**

– Assess the likelihood and consequences of these risks

- **Risk planning**

– Draw up plans to avoid or minimize the effects of the risk

- **Risk monitoring**

– Monitor the risks throughout the project

Risk Identification

- The different categories of risk (project, technical, business, known, predictable, and unpredictable) can be further divided as:

a) Generic risks:

b) Product-specific risks:

Generic Risk

- ***Generic risks*** are a potential threat to every software project

Product Specific

- ***Product-specific risks*** can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the software that is to be built.
- To identify product-specific risks, the project plan and the software statement of scope are examined, and an answer to the following question is developed: “What special characteristics of this product may threaten our project plan?”
- Method for identifying risks is to create a risk item checklist.

Risk Identification

One method for identifying risks is to create a **risk item checklist**; we need to identify the following area from where risk will appear:

- *P**roduct size***—risks associated with the overall size of the software to be built or modified.
- ***Business impact***- risks associated with constraints imposed by management or the marketplace.
- ***Process definition***—risks associated with the degree to which the software process has been defined and is followed by the development organization.
- ***Development environment***—risks associated with the availability and quality of the tools to be used to build the produce
- ***Technology to be built***—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
- ***Staff size and experience***—risks associated with the overall technical and project experience of the software engineers who will do the work.

Risk assessment

- Rank risk in terms of their damage causing potential

- **The overall risk exposure, RE**

$$\text{Risk Exposure (RE)} = r \mathbf{X} c$$

- **Here**, r **is** probability of occurrence **for a risk, and c is** cost to project **should risk occur.**

CASE STUDY

For example, assume that software team defines a project risk in as follows:

- **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

CASE STUDY

- 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.

Class Work

- **Risk Identification**: Only 60 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**: 70%
- **Risk Impact**: 80 reusable software components were planned. If only 60 percent can be used other components would have to be developed from. Since the average component is 350 LOC and local data indicate that the software engineering cost for each LOC is RS. 20.00

RISK ANALYSIS

- **Assess probability and effect of each risk**
- **Probability may be very low, low, moderate, high or very high**
- **Risk effects might be catastrophic, serious, tolerable or insignificant**

RISK PLANNING

- **Consider each risk and develop a strategy to manage that risk**
- **Reactive vs. proactive approach**
- **Avoidance strategies**
  – The probability that the risk will arise is reduced
- **Minimization strategies**
  – The impact of the risk on the project or product will be reduced
- **Contingency plans**
  – If risk arises, contingency plans are to deal with that risk
- **Risk mitigation, risk transfer**

RISK MONITORING

- **Assess each identified risks regularly to decide whether it is becoming less or more probable**
- **Also assess whether the effects of the risk have changed**
- **Each key risk should be discussed at management progress meetings**

Assessing Overall Project Risk

- The following questions have been derived from risk data obtained by surveying experienced software project managers in different parts of the world.
- The questions are ordered by their relative importance to the success of a project.

Questions

- **Have top software and customer managers formally committed to support the project?**
- **Are end users enthusiastically committed to the project and the system/ product to be built?**
- **Are requirements fully understood by the software engineering team and its customers?**
- **Have customers been involved fully in the definition of requirements?**
- **Do end users have realistic expectations?**
- **Is the project scope stable?**
- **Does the software engineering team have the right mix of skills?**
- **Are project requirements stable?**
- **Does the project team have experience with the technology to be implementation**

Questions

- **Is the number of people on the project team adequate to do the job?**
- **Do all customer agree on the importance of the project and on the requirements for the system/product to be built?**

Risk Components and Drivers

Risk components are defined in the following manner:

- ***Performance risk***—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- ***Cost risk***—the degree of uncertainty that the project budget will be maintained.
- ***Support risk***—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- ***Schedule risk***—the degree of uncertainty that the project schedule will be maintained, and that the product will be delivered on time.

- The impact of each risk driver on the risk component is divided into one of four impact categories—negligible, marginal, critical, or catastrophic

Risk Projection

Risk projection, also called ***risk estimation,*** attempts to rate each risk in two ways—

(1) the l**ikelihood or probability** that the risk is real

(2) the **consequences** of the problems associated with the risk, should it occur.

Risk Projection

Risk projection steps:

1. Establish a scale that reflects the perceived likelihood of a risk.
2. Delineate the consequences of the risk.
3. Estimate the impact of the risk on the project and the product.
4. Assess the overall accuracy of the risk projection so that there will be no misunderstandings.

Risk Table

- A risk table provides you with a simple technique for risk projection.

- You begin by listing all risks in the first column of the table.
- This can be accomplished with the help of the risk item checklists
- Each risk is categorized in the second column
- The table is sorted by probability and by impact.
- High-probability, high-impact risks percolate to the top of the table, and low-probability risks drop to the bottom.

Risk Projection and Risk Table

Risks	Category	Probability	Impact	RMMM
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	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover will be high	ST	60%	2	
Σ				
Σ				
Σ				

Impact values:
1—catastrophic
2—critical
3—marginal
4—negligible

Risk Category:
PS = Project size risk
BU = Business risk
TE = Technology risk
DE = Development risk
ST = Stakeholder risk
CU =Customer risk

Fig. RISK TABLE

Risk Table and Projection

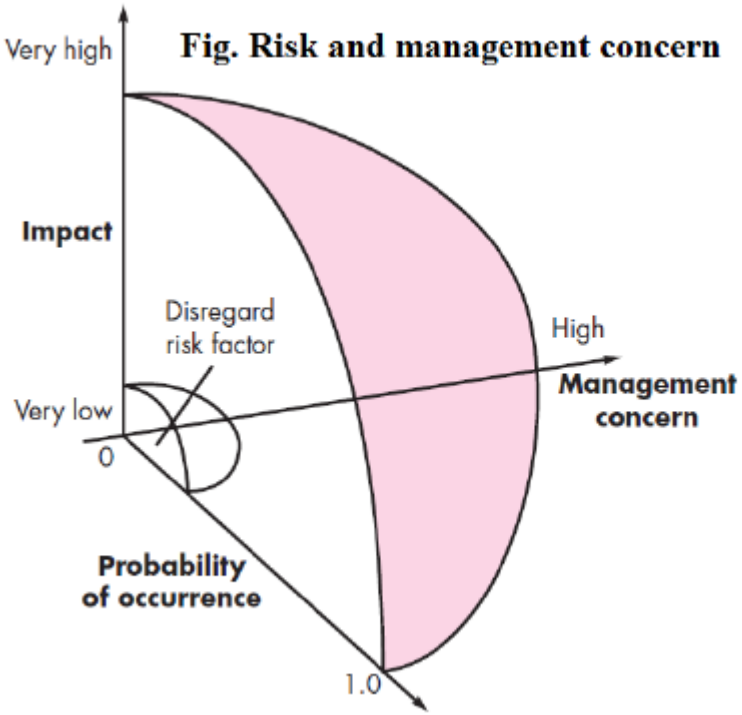


Fig. Risk and management concern

Risk Table and Projection (Management Concern)

- A risk factor that has a high impact, but a very low probability of occurrence should not absorb a significant amount of management time.
- However, high-impact risks with moderate to high probability and low-impact risks with high probability should be carried forward into the risk analysis steps that follow.
- All risks that lie above the cutoff line should be managed.
- The column labeled RMMM contains a pointer into a *risk mitigation, monitoring, and management plan* or, alternatively, a collection of risk information sheets developed for all risks that lie above the cutoff.

Risk Refinement

- During early stages of project planning, a risk may be stated quite generally.
- As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.
- One way to do this is to represent the risk in ***condition-transition-consequence*** **(CTC) format.**
- Given that <condition> then there is concern that (possibly) <consequence>.

RISK REFINEMENT

- **Given that all reusable software components must conform to specific design standards and that some do not conform, then there is concern that (possibly) only 70 percent of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30 percent of components.**
- This general condition can be refined in the following manner:
- **Subcondition 1.** Certain reusable components were developed by a third party with no knowledge of internal design standards.
- **Subcondition 2.** The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
- **Subcondition 3.** Certain reusable components have been implemented in a language that is not supported on the target environment.
- The consequences associated with these refined subconditions remain the same (i.e., 30 percent of software components must be custom engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

RMMM

- Risk **Mitigation, Monitoring, and Management**
- Assessment helps to develop strategy
- An effective strategy must consider three issues: **risk avoidance**, **risk monitoring**, **and risk management and contingency planning**.

RMMM

- All of the risk analysis activities presented to this point have a single goal—to assist the project team in developing a strategy for dealing with risk.
- An effective strategy must consider three issues: risk avoidance, risk monitoring, and risk management and contingency planning.
- If a software team adopts a proactive approach to risk, avoidance is always the best strategy. This is achieved by developing a plan for *risk mitigation*

RMMM

For example, assume that high staff turnover is noted as a project risk

- To mitigate this risk, you would develop a strategy for reducing turnover.

The possible steps to be taken are:

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market).
- Mitigate those causes that are under your control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.

RMMM

- Define work product standards and establish mechanisms to be sure that all models and documents are developed in a timely manner.
- Conduct peer reviews of all work (so that more than one person is “up to speed”).
- Assign a backup staff member for every critical technologist.

Risk Information Sheet

Risk information sheet			
Risk ID: P02-4-32	Date: 5/9/09	Prob: 80%	Impact: high
Description: 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.			
Refinement/context: Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards. Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components. Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.			
Mitigation/monitoring: 1. Contact third party to determine conformance with design standards. 2. Press for interface standards completion; consider component structure when deciding on interface protocol. 3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.			
Management/contingency plan/trigger: *RE* computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly. Trigger: Mitigation steps unproductive as of 7/1/09.			
Current status: 5/12/09: Mitigation steps initiated.			
Originator: D. Gagne		Assigned: B. Laster	