UECS3383 Software Quality Assurance

Lecture 05 – Utilization of Metrics to Software Quality Assurance (SAQ)

Components of Software Quality Assurance System Overview

- Pre-project components
- Software project life cycle components
- Infrastructure components for error prevention and improvements
- **Management SQA components**
- SQA standards, system certification and assessment components
- Organizing for SQA – the human components
- Considerations guiding construction of organization's SQA system

Components of SQA System Overview

- Pre-Project SQA components
  - **Contract reviews**
  - **Development and quality plans**
- Software Project Life Cycle components
  - **Reviews**
  - **Expert opinions**
  - **Software testing**
  - **Software maintenance components**
  - **Assurance of the quality of external participants' work**

Components of SQA System Overview

- Infrastructure components for prevention and improvement
  - **Procedures and work instruction**
  - **Templates and checklists**
  - **Staff training, retraining and certification**
  - **Preventive and corrective actions**
  - **Configuration management**
  - **Documentation control**
- Management SQA components
  - **Project progress control**
  - **Software quality metrics**
  - **Software quality costs**

Basic Measurement Theory

- Development and Maintenance phase:
  - Measurement of the various aspect of software quality is considered to be an effective tool for the support and control activities and the initiation of process improvements during the development and maintenance of process improvement
  - Measurement apply to the functional quality, productivity and organizational aspects of the project
  - Metrics of SQ:
    - Quality of software development and maintenance activities
    - Development teams' productivity
    - Helpdesk and maintenance teams' productivity
    - Software faults density
    - Schedule deviations

Software Quality Costs

- Quality costs incurred by software development and application are:
  - Cost of control (prevention cost, appraisal cost, managerial preparation and control cost)
  - Cost of failure (internal failure costs, external failure costs, managerial failure costs)
- Management is interested in the total sum of quality cost. More willing to exhibit greater readiness to allocate funds to profitable proposals to improve application of existing SQA system components and further development of new components

Software Quality Metrics

- IEEE definition:
  - A **quantitative measure** of the degree to which an item possesses a given quality attribute
  - A **function** whose input are software data and whose output is a single numerical value that can be interpreted as the degree to which the software posses a given quality attribute

Main Objectives of SQ Metrics

- **Facilitate management control, planning and managerial intervention**.
  Based on:
  - Deviations of actual from planned performance.
  - Deviations of actual timetable and budget performance from planned.

- **Identify situations for development or maintenance process improvement (preventive or corrective actions).**
  - Based on : Accumulation of metrics information regarding the performance of teams, units, etc.

Elements of Quality Definitions

Quality factors	Predictable	Measurable
Defect Level	Yes	Yes
Defect Origins	Yes	Yes
Defect Severity	Yes	Yes
Defect Removal Efficiency	Yes	Yes
Product Complexity	Yes	Yes
Project Reliability	Yes	Yes
Project Maintainability	Yes	Yes
Project schedules	Yes	Yes
Project budgets	Yes	Yes
Portability	Yes	Yes
Conformance to requirements	No	Yes
User Satisfaction	No	Yes
Fitness for Use	No	Yes
Robustness	No	No

Software Quality Metrics

- **Quality**: The totality of features and characteristics of a product, process or service that bear on its ability to satisfy stated or implied needs.
- The expected quality features and characteristics of a software product are commonly referred to as **Quality Attributes**.
- Quality attributes are **used early in the development process** to identify user quality requirements. Each system has specific and unique quality needs, which are a function of the purpose of the application.
- Software quality attributes are assessed by **using Quality Control** Techniques and when possible, appropriate Software Measurements or **Metrics**.

SQ Metrics - Requirements

- **General requirements**
  - Relevant
  - Valid
  - Reliable
  - Comprehensive
  - Mutually exclusive
- **Operative requirements**
  - Easy and simple
  - Does not require independent data collection
  - Immune to biased interventions by interested parties

SQ Metrics – Requirements (Lecture)

Software Quality Metrics – Requirements	
General requirements	
Relevant	Related to an attribute of substantial importance
Valid	Measures the required attribute
Reliable	Produces similar results when applied under similar conditions
Comprehensive	Applicable to a large variety of implementations and situations
Mutually exclusive	Does not measure attributes measured by other metrics
Operative requirements	
Easy and simple	The implementation of the metrics data collection is simple and is performed with minimal resources
Does not require independent data collection	Metrics data collection is integrated with other project data collection systems; employee attendance, wages, cost accounting, etc. in addition to its efficiency aspects, this requirement contributes to coordination of all information systems serving the organization
Immune to biased interventions by interested parties	In order to escape the expected results of the analysis of the metrics, it is expected that interested persons will try to change the data and, by doing so, improve their record. Such actions obviously bias the relevant metrics. Immunity (total or at least partial) is achieved mainly by choice of metrics and adequate procedures.

Classification of Software Metrics

- **Product Metrics**
  - Describe the characteristics of the product such as size, complexity, design features, performance, and quality level
- **Process Metrics**
  - can be used to improve software development and maintenance
  - Examples include the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process
- **Project Metrics**
  - describe the project characteristics and execution
  - Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity. Some metrics belong to multiple categories. For example, the inprocess quality metrics of a project are both process metrics and project metrics.

Classification of Software Quality Metrics

- **Classification by phases of software system**
  - **Process metrics** – metrics related to the software development process
  - **Product metrics** – metrics related to software maintenance
- **Classification by subjects of measurements**
  - Quality
  - Timetable
  - Effectiveness (of error removal and maintenance services)
  - Productivity

Software Size (Volume) Measures

- **KLOC —** classic metric that measures the size of software by thousands of code lines.
- **Number of function points (NFP) —** a measure of the development resources (human resources) required to develop a program, based on the functionality specified for the software system.

Measurement: Error Counted Measures

- Number of Code Errors (NCE) vs. Weighted Number of Code Errors (WCE)

	Calculation of NCE	Calculation of WCE	
Error severity class	**Number of Errors**	**Relative Weight**	**Weighted Errors**
a	b	c	D = b x c
low severity	42	1	42
medium severity	17	3	51
high severity	11	9	99
Total	**70**	**---**	**192**
NCE	**70**	**---**	**---**
WCE		**---**	**192**

Category of Process Metrics

- Software process quality metrics
  - Error density metrics
  - Error severity metrics
- Software process timetable metrics
- Software process error removal effectiveness metrics
- Software process productivity metrics

Error Density Metrics

Code	Name	Calculation formula
CED	**Code Error Density**	$CED = \frac{NCE}{KLOC}$
DED	**Development Error Density**	$DED = \frac{NDE}{KLOC}$
WCED	**Weighted Code Error Density**	$WCDE = \frac{WCE}{KLOC}$
WDED	**Weighted Development Error Density**	$WDED = \frac{WDE}{KLOC}$
WCEF	**Weighted Code Errors per Function Point**	$WCEF = \frac{WCE}{NFP}$
WDEF	**Weighted Development Errors per Function Point**	$WDEF = \frac{WDE}{NFP}$

NCE = The number of code errors detected by code inspections and testing.
NDE = total number of development (design and code) errors) detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.

Error Severity Metrics

Code	Name	Calculation formula
ASCE	Average Severity of Code Errors	$ASCE = \frac{WCE}{NCE}$
ASDE	Average Severity of Development Errors	$ASDE = \frac{WDE}{NDE}$

NCE = The number of code errors detected by code inspections and testing.
NDE = total number of development (design and code) errors detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.

Software Process Timetable Metrics

Code	Name	Calculation formula
TTO	**Time Table Observance**	$TTO = \frac{MSOT}{MS}$
ADMC	**Average Delay of Milestone Completion**	$ADMC = \frac{TCDAM}{MS}$

MSOT = Milestones completed on time.
MS = Total number of milestones.
TCDAM = Total Completion Delays (days, weeks, etc.) for all milestones.

Error Removal Effectiveness Metrics

Code	Name	Calculation formula
DERE	Development Errors Removal Effectiveness	$DERE = \frac{NDE}{NDE + NYF}$
DWERE	Development Weighted Errors Removal Effectiveness	$DWERE = \frac{WDE}{WDE+WYF}$

NDE = total number of development (design and code) errors) detected in the development process.
WCE = weighted total code errors detected by code inspections and testing.
WDE = total weighted development (design and code) errors detected in development process.
NYF = number software failures detected during a year of maintenance service.
WYF = weighted number of software failures detected during a year of maintenance service.

Process Productivity Metrics

Code	Name	Calculation formula
DevP	Development Productivity	$DevP = \frac{DevH}{KLOC}$
FDevP	Function point Development Productivity	$FDevP = \frac{DevH}{NFP}$
CRe	Code Reuse	$Cre = \frac{ReKLOC}{KLOC}$
DocRe	Documentation Reuse	$DocRe = \frac{ReDoc}{NDoc}$

DevH = Total working hours invested in the development of the software system.
ReKLOC = Number of thousands of reused lines of code.
ReDoc = Number of reused pages of documentation.
NDoc = Number of pages of documentation.

Category of Product Metrics

- **HD quality metrics:**
  - HD calls density metrics - measured by the number of calls.
  - HD calls severity metrics - the severity of the HD issues raised.
  - HD success metrics – the level of success in responding to HD calls.
- **HD productivity metrics.**
- **HD effectiveness metrics.**
- **Corrective maintenance quality metrics.**
  - Software system failures density metrics
  - Software system failures severity metrics
  - Failures of maintenance services metrics
  - Software system availability metrics
- **Corrective maintenance productivity and effectiveness metrics**.

Helpdesk (HD) Density Metrics

Code	Name	Calculation Formula
HDD	HD calls density	$HDD = \frac{NHYC}{KLMC}$
WHDD	Weighted HD calls density	$WHYC = \frac{WHYC}{KLMC}$
WHDF	Weighted HD calls per function point	$WHDF = \frac{WHYC}{NMFP}$

NHYC = the number of HD calls during a year of service.
KLMC = Thousands of lines of maintained software code.
WHYC = weighted HD calls received during one year of service.
NMFP = number of function points to be maintained.

Severity of HD Calls Metrics

Code	Name	Calculation Formula
ASHC	Average severity of HD calls	$ASHC = \frac{WHYC}{NHYC}$

NHYC = the number of HD calls during a year of service.

WHYC = weighted HD calls received during one year of service.

Helpdesk Success Metrics

Code	Name	Calculation Formula
HDS	HD service success	$HDS = \frac{NHYOT}{NHYC}$

NHYNOT = Number of yearly HD calls completed on time during one year of service.
NHYC = the number of HD calls during a year of service.

HD Productivity and Effectiveness Metrics

Code	Name	Calculation Formula
HDP	HD Productivity	$HDP = \frac{HDYH}{KLNC}$
FHDP	Function Point HD Productivity	$FHDP = \frac{HDYH}{NMFP}$
HDE	HD effectiveness	$HDE = \frac{HDYH}{NHYC}$

HDYH = Total yearly working hours invested in HD servicing of the software system.
KLMC = Thousands of lines of maintained software code.
NMFP = number of function points to be maintained.
NHYC = the number of HD calls during a year of service.

Software System Failure Density Metrics

Code	Name	Calculation Formula
SSFD	Software System Failure Density	$SSFD = \frac{NYF}{KLMC}$
WSSFD	Weighted Software System Failure Density	$WFFFD = \frac{WYF}{KLMC}$
WSSFF	Weighted Software System Failures per Function point	$WSSFF = \frac{WYF}{NMFP}$

NYF = number of software failures detected during a year of maintenance service.
WYF = weighted number of yearly software failures detected during one year of maintenance service.
NMFP = number of function points designated for the maintained software.
KLMC = Thousands of lines of maintained software code.

Software System Failure Severity Metrics

Code	Name	Calculation Formula
ASSSF	Average Severity of Software System Failures	$ASSSF = \frac{WYF}{NYF}$

NYF = number of software failures detected during a year of maintenance service.
WYF = weighted number of yearly software failures detected during one year.

Failure of Maintenance Service Metrics

Code	Name	Calculation Formula
MRepF	Maintenance Repeated repair Failure metric -	$MRepF = \frac{RepYF}{NYF}$

NYF = number of software failures detected during a year of maintenance service.
RepYF = Number of repeated software failure calls (service failures).

Software System Availability Metrics

Code	Name	Calculation Formula
FA	Full Availability	$FA = \frac{NYSerH - NYFH}{NYSerH}$
VitA	Vital Availability	$VitA = \frac{NYSerH - NYVitFH}{NYSerH}$
TUA	Total Unavailability	$TUA = \frac{NYTFH}{NYSerH}$

NYSerH = Number of hours software system is in service during one year.
NYFH = Number of hours where at least one function is unavailable (failed) during one year, including total failure of the software system.
NYVitFH = Number of hours when at least one vital function is unavailable (failed) during one year, including total failure of the software system.
NYTFH = Number of hours of total failure (all system functions failed) during one year.
NYFH ≥ NYVitFH ≥ NYTFH.
1 – TUA ≥ VitA ≥FA

Software Corrective Maintenance Productivity and Effectiveness Metrics

Code	Name	Calculation Formula
CMaiP	Corrective Maintenance Productivity	$CMaiP = \frac{CMaiYH}{KLMC}$
FCMP	Function point Corrective Maintenance Productivity	$FCMP = \frac{CMaiYH}{NMFP}$
CMaiE	Corrective Maintenance Effectiveness	$CMaiE = \frac{CMaiYH}{NYF}$

CMaiYH = Total yearly working hours invested in the corrective maintenance of the software system.
NYF = number of software failures detected during a year of maintenance service.
NMFP = number of function points designated for the maintained software.
KLMC = Thousands of lines of maintained software code.

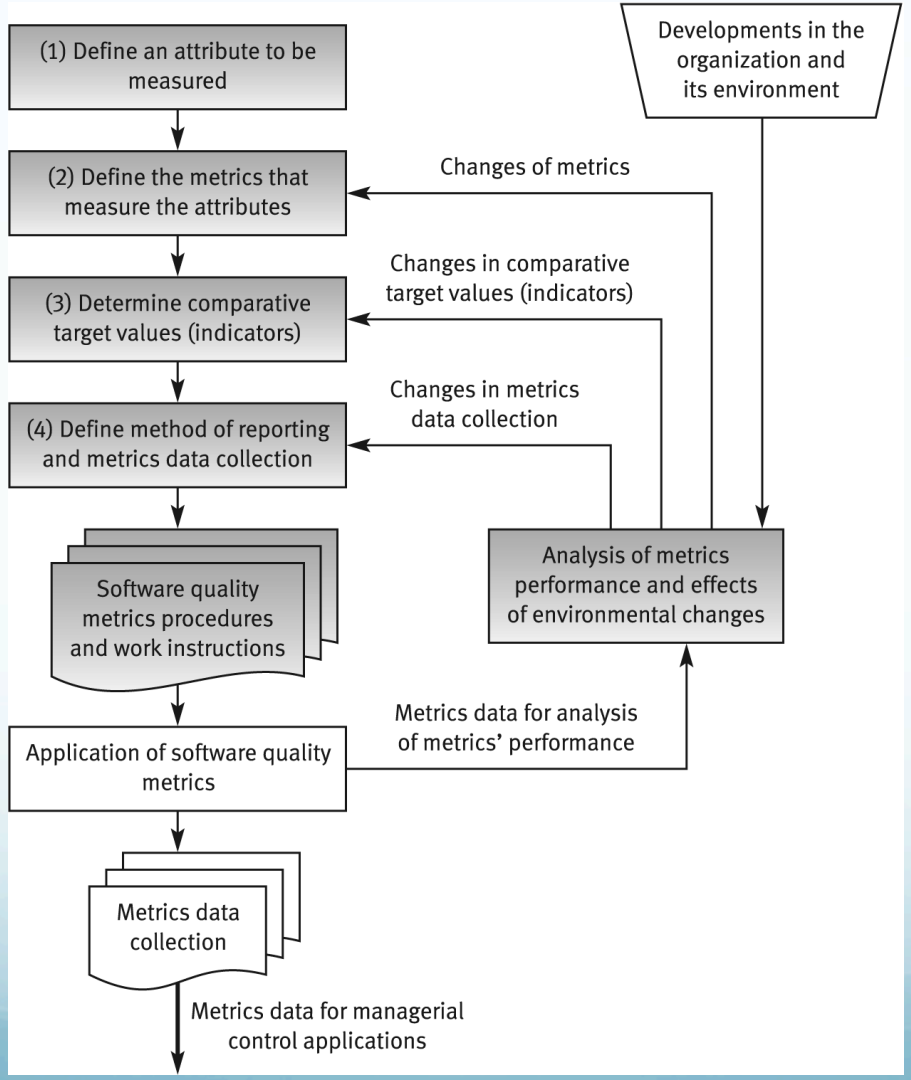
Typical Quality Attributes

- **Usability:** effort required to learn, operate and use a program.
- **Safety:** ability of the system to operate without catastrophic failure.
- **Fault-tolerance:** ability of the system to continue operating after some system faults have manifested themselves.
- **Security:** ability of the system to protect itself against accidental or deliberate intrusion.

Typical Quality Attributes

- **Portability:** effort required to transfer a program from one computing environment or platform to another.
- **Reusability:** ability and effort required to reuse a program or part of a program in other applications.
- **Interoperability:** effort required interconnecting or relating two different applications, running possibly in different computing environments
- **Maintainability:** effort required modifying, updating, evolving, or repairing a program during its operation.

The Process of Defining Software Quality Metrics



General Limitation of Quality Metrics

- Budget constraints in allocating the necessary resources.
- Human factors, especially opposition of employees to evaluation of their activities.
- Validity Uncertainty regarding the data's, partial and biased reporting.

Software Metrics that Exhibit Severe Weaknesses

- Parameters used in development process metrics:
  KLOC, NDE, NCE.
- Parameters used in product (maintenance) metrics:
  KLMC, NHYC, NYF.

KLOC — classic metric that measures the size of software by thousands of code lines
NCE = The number of code errors detected by code inspections and testing.
NDE = total number of development (design and code) errors) detected in the development process.
KLMC = Thousands of lines of maintained software code.
NHYC = the number of HD calls during a year of service.
NYF = number of software failures detected during a year of maintenance service.

Factors Affecting Parameters Used for Development Process Metrics

- Programming style **(KLOC).**
- Volume of documentation comments **(KLOC).**
- Software complexity **(KLOC, NCE).**
- Percentage of reused code **(NDE, NCE).**
- Professionalism and thoroughness of design review and software testing teams: affects the number of defects detected **(NCE).**
- Reporting style of the review and testing results: concise reports vs. comprehensive reports **(NDE, NCE).**

Factors Affecting Parameters Used for Product (Maintenance) Metrics

- Quality of installed software and its documentation **(NYF, NHYC).**
- Programming style and volume of documentation comments included in the code be maintained **(KLMC).**
- Software complexity **(NYF).**
- Percentage of reused code **(NYF).**
- Number of installations, size of the user population and level of applications in use: **(NHYC, NYF).**

The Function Point Method

The function point estimation process:

- **Stage 1**: Compute crude function points **(CFP).**
- **Stage 2**: Compute the relative complexity adjustment factor **(RCAF)** for the project. RCAF varies between 0 and 70.
- **Stage 3**: Compute the number of function points **(FP):**

  FP = CFP x (0.65 + 0.01 x RCAF)

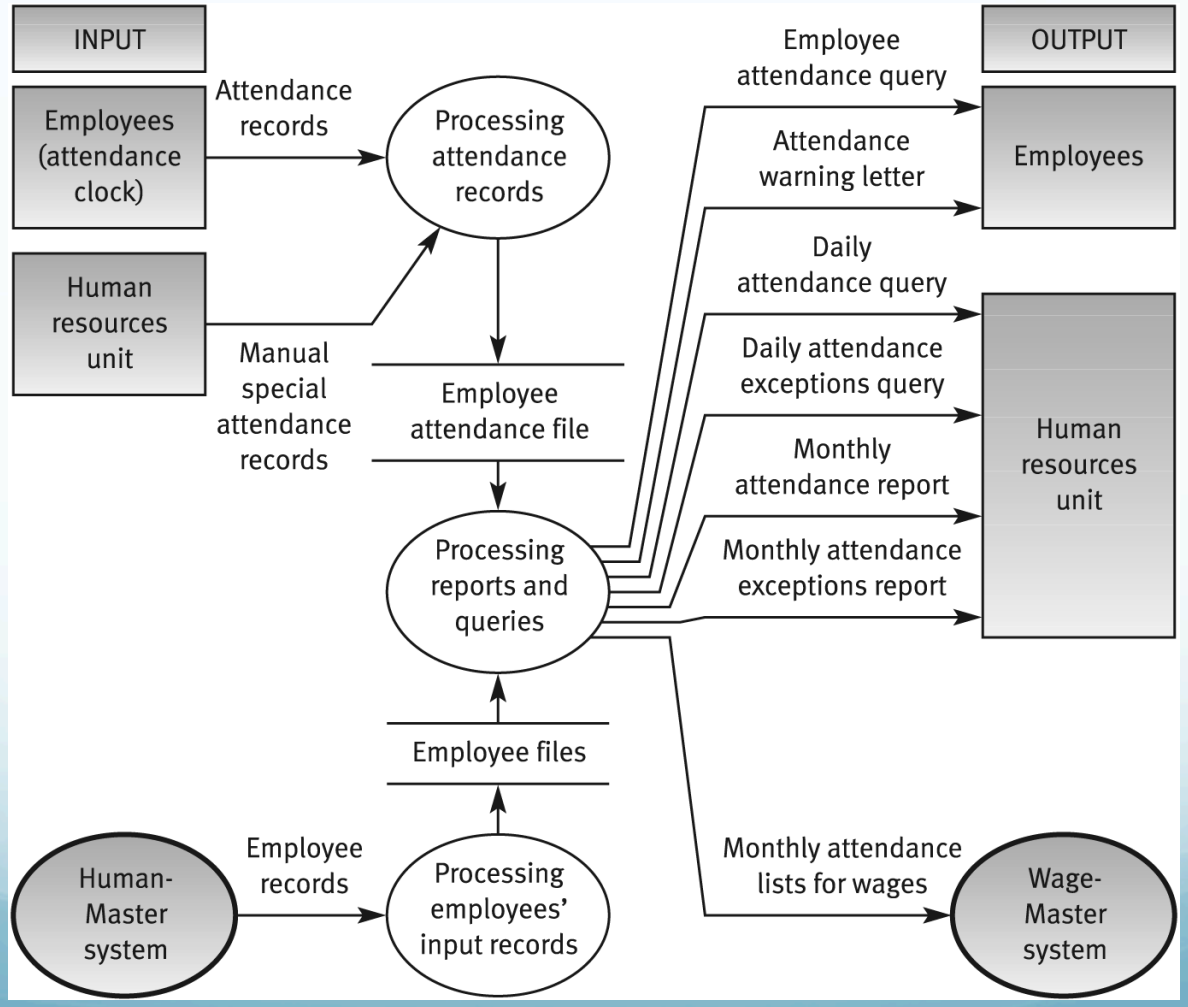
Crude Function Points (CFP) – Calculation form

Software system components	Complexity level									Total CFP
	Simple			average			complex			
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points	
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH	
User inputs		3			4			6		
User outputs		4			5			7		
User online queries		3			4			6		
Logical files		7			10			15		
External interfaces		5			7			10		
Total CFP										

Relative Complexity Adjustment Factor (RCAF) - Form

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5
	Total = RCAF	

The ATTEND MASTER – Data Flow Diagram



The ATTEND MASTER – CFP Calculation Form

Software system components	Complexity level									Total CFP
	Simple			average			complex			
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points	
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH	
User inputs	1	3	3	---	4	---	1	6	6	9
User outputs	---	4	---	2	5	10	1	7	7	17
User online queries	1	3	3	1	4	4	1	6	6	13
Logical files	1	7	7	---	10	---	1	15	15	22
External interfaces	---	5	---	---	7	---	2	10	20	20
Total CFP										81

The ATTEND MASTER – RCAF Calculation form

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 **5**
2	Requirement for data communication	**0** 1 2 3 4 5
3	Extent of distributed processing	**0** 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 **5**
5	Expected operational environment	**0** 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 **4** 5
7	Extent of multi-screen or multi-operation online data input	0 1 **2** 3 4 5
8	Extent of online updating of master files	0 1 **2** 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 **4** 5
10	Extent of complex data processing	0 1 2 3 **4** 5
11	Extent that currently developed code can be designed for reuse	0 1 2 **3** 4 5
12	Extent of conversion and installation included in the design	0 1 **2** 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 **5**
14	Extent of change and focus on ease of use	0 1 2 3 4 **5**
	Total = RCAF	41

The ATTEND MASTER – Function Points Calculation

- **FP = CFP x (0.65 + 0.01 x RCAF)**

- **FP = 81 x (0.65 + 0.01 x 41) = 85.86**

The Function Point Method

- **Main Advantages**
  - Estimates can be prepared at the pre-project stage.
  - Based on requirement specification documents (not specific dependent on development tools or programming languages), the method's reliability is relatively high

- **Main Disadvantages**
  - FP results depend on the counting instruction manual.
  - Estimates based on detailed requirements specifications, which are not always available.
  - The entire process requires an experienced function point team and substantial resources.
  - The evaluations required result in subjective results.
  - Successful applications are related to data processing. The method cannot yet be universally applied