Software Quality Metrics

Software quality metrics

- Objectives of quality measurement
- Classification of software quality metrics
- Process metrics
- Product metrics
- Implementation of software quality metrics
- Limitations of software metrics
- The function point method

IEEE definitions of software quality metrics

(1) **A quantitative measure** of the degree to which an item possesses a given quality attribute.

(2) **A function** whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

Main objectives of software quality metrics

1. **Facilitate management control, planning and managerial intervention**.
   Based on:
   - Deviations of actual from planned performance.
   - Deviations of actual timetable and budget performance from planned.
2. **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.

Software quality metrics — Requirements

General requirements

- **Relevant:** related to an attribute of substantial importance
- **Valid:** measures the required attribute
- **Reliable:** **produce similar results when applied undersimilar condition**
- **Comprehensive:** Applicable to a very large variety of implementations and solutions
- **Mutually exclusive:** does not measure attribute measure by the othe metrics

Operative requirements

- **Easy and simple:** data collection performed with minimal resources
- **Does not require independent data collection:** integrated with other project data collection
- **Immune to biased interventions by interested parties:** choice of metrics and adeqate procedures

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.

Classifications of software quality metrics

Classification by phases of software system

- **Process metrics** –related to the software development process
- **Product metrics** –related to software maintenance

Classification by subjects of measurements

- **Quality**
- **Timetable**
- **Effectiveness (of error removal and maintenance services)**
- **Productivity**

Process metrics categories

- **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 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**

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	$ASCE = \frac{WCE}{NCE}$
ADED	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.

Product metrics categories

* **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**.

HD calls 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.

HD 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 failures 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.

Failures of maintenance services 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.

$NYFH \geq NYVitFH \geq NYTFH$.

$1 - TUA \geq VitA \geq FA$

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	$CMaiE = \frac{CMaiYH}{NYF}$

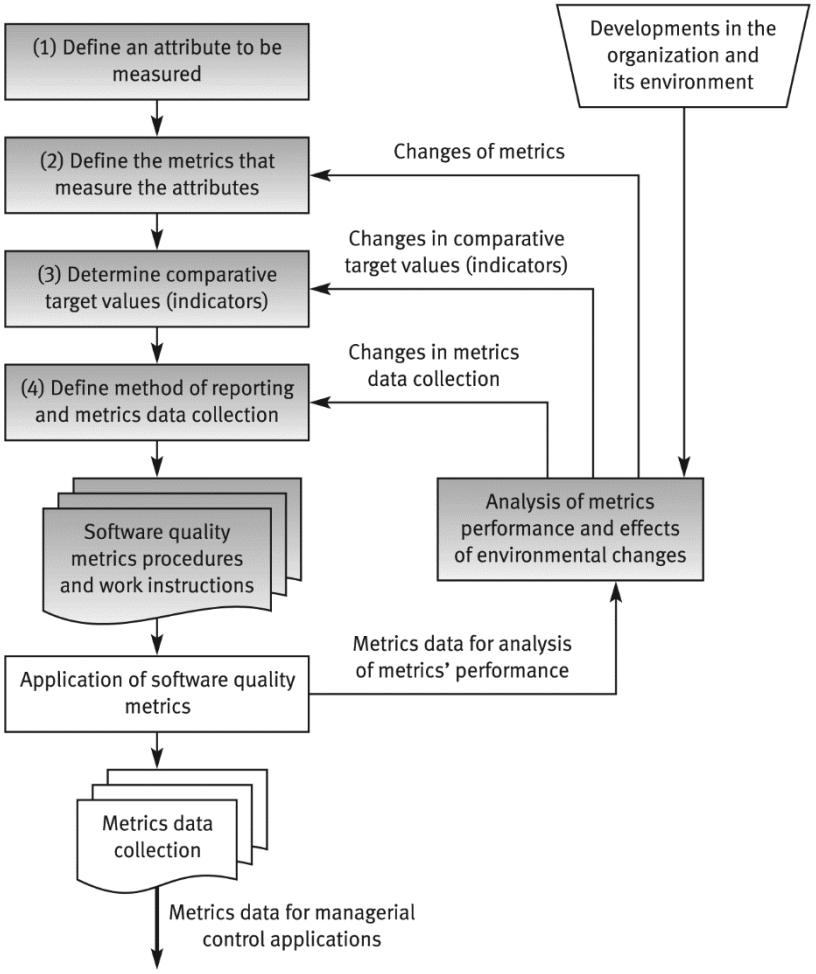
CMaiYH = Total yearly working hours invested in the corrective maintenance of the software system.

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.

The process of defining software quality metrics



Galin, SQA from Theory to Implementation

General limitations 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.

Examples of software metrics that exhibit severe weaknesses

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

Factors affecting parameters used for development process metrics

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

Factors affecting parameters used for product (maintenance) metrics

a. Quality of installed software and its documentation **(NYF, NHYC).**
b. Programming style and volume of documentation comments included in the code be maintained **(KLMC).**
c. Software complexity **(NYF).**
d. Percentage of reused code **(NYF).**
e. 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)

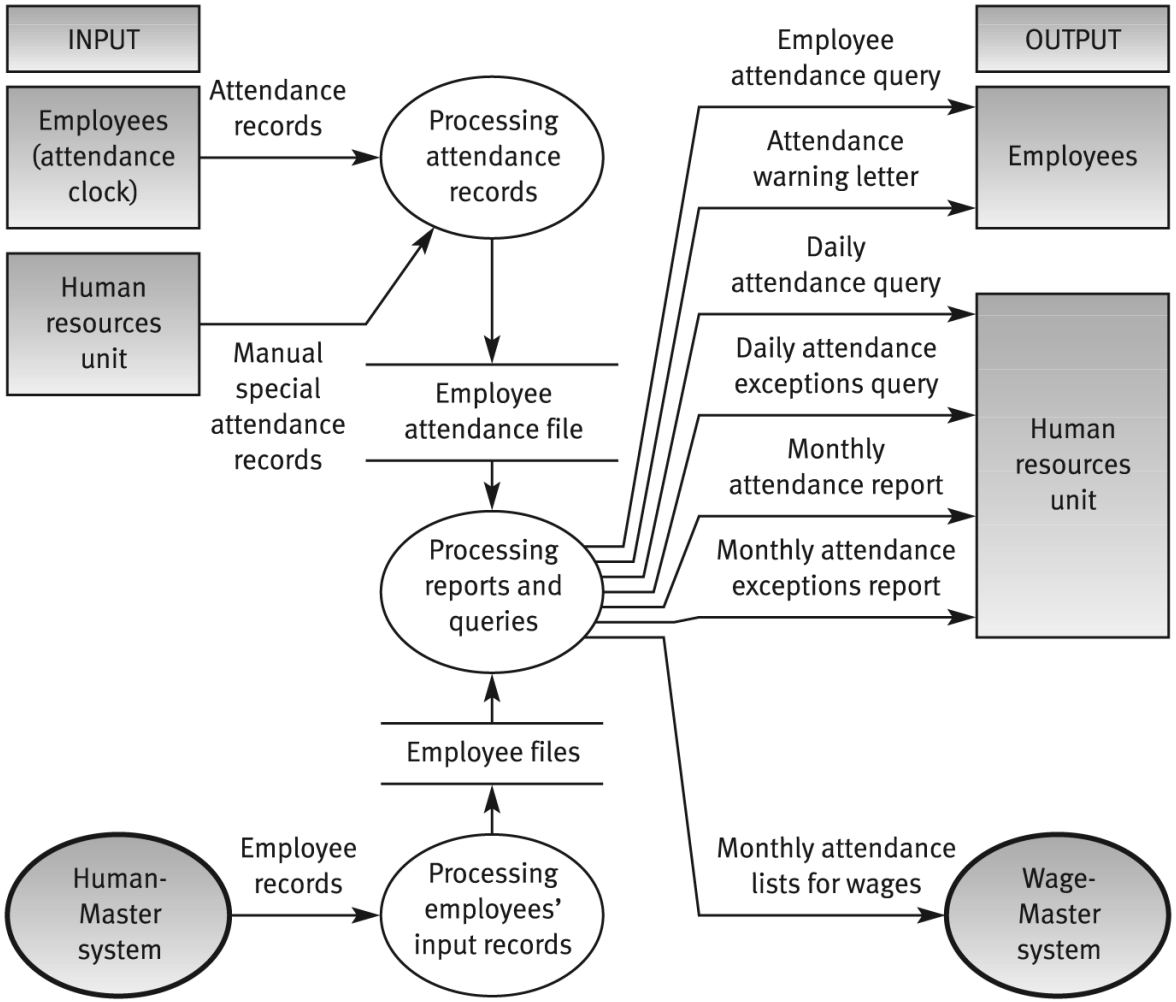
Crude function points (CFP) – calculation form

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

$$FP = CFP \times (0.65 + 0.01 \times RCAF)$$

$$FP = 81 \times (0.65 + 0.01 \times 41) = 85.86$$

The function point method – advantages and disadvantages

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.

Cost of Software Quality

Costs of software quality

1. Objectives of cost of software quality metrics
2. The classic model of cost of software quality
   - Prevention costs
   - Appraisal costs
   - Internal failure costs
   - External failure costs
3. Galin's extended model for cost of software quality
   - Managerial preparation and control costs
   - Managerial failure costs
4. Application of a cost of software quality system
   - Definition of a cost of software quality model
   - Definition of the cost data collection method
   - Implementation of a cost of software quality system
   - Problems in the application of cost of software quality metrics
5. Problems in the application of cost of software quality metrics

Cost of software quality metrics — Objectives

In general – it enables management to achieve economic control over SQA activities and outcomes. The specific objectives are:

* Control organization-initiated costs to prevent and detect software errors.
* Evaluation of the economic damages of software failures as a basis for revising the SQA budget.
* Evaluation of plans to increase or decrease of SQA activities or to invest in SQA infrastructure on the basis of past economic performance.

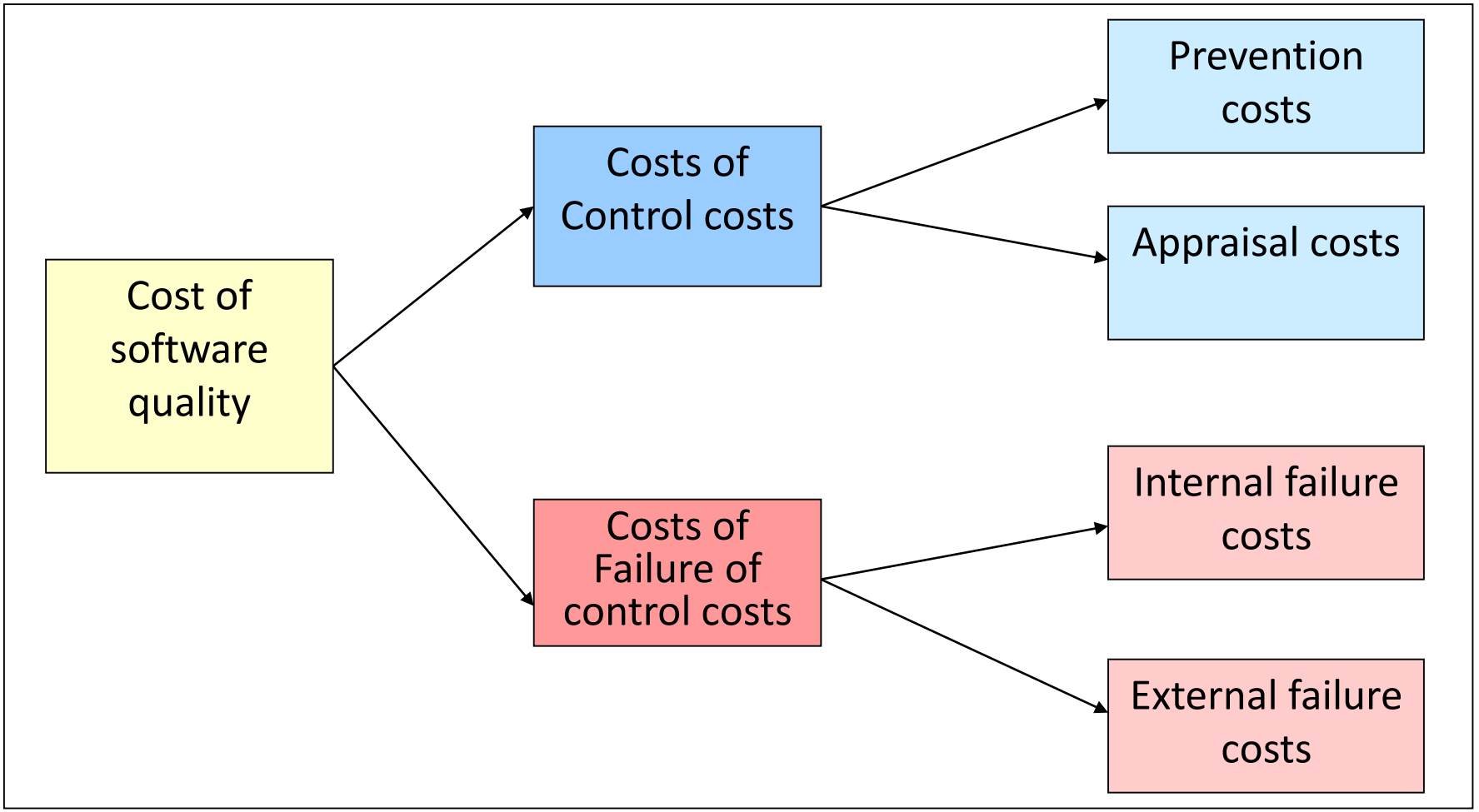
Performance comparisons for Managerial control over SQA costs

* Control Budgeted expenditures (for SQA prevention and appraisal activities).
* Previous year's failure costs
* Previous project's quality costs (control costs and failure costs).
* Other department's quality costs (control costs and failure costs).

Cost metrics for evaluating SQA systems - examples

* Percentage of cost of software quality out of total software development costs.
* Percentage of software failure costs out of total software development costs.
* Percentage of cost of software quality out of total software maintenance costs.
* Percentage of cost of software quality out of total sales of software products and software maintenance.

The classic model of cost of software quality



Prevention costs

a. **Investments in development of SQA infrastructure components**
   * Procedures and work instructions
   * Support devices: templates, checklists etc
   * Software configuration management system
   * Software quality metrics

b. **Regular implementation of SQA preventive activities**:
   * Instruction of new employees in SQA subjects
   * Certification of employees
   * Consultations on SQA issues to team leaders and others

c. **Control of the SQA system through performance of:**
   * Internal quality reviews
   * External quality audits
   * Management quality reviews

Appraisal costs

(a) Costs of reviews:

* Formal design reviews (DRs)
* Peer reviews (inspections and walkthroughs)
* Expert reviews

(b) Costs of software testing:

* Unit, integration and software system tests
* Acceptance tests (carried out by customers)

(c) Costs of assuring quality of external participants

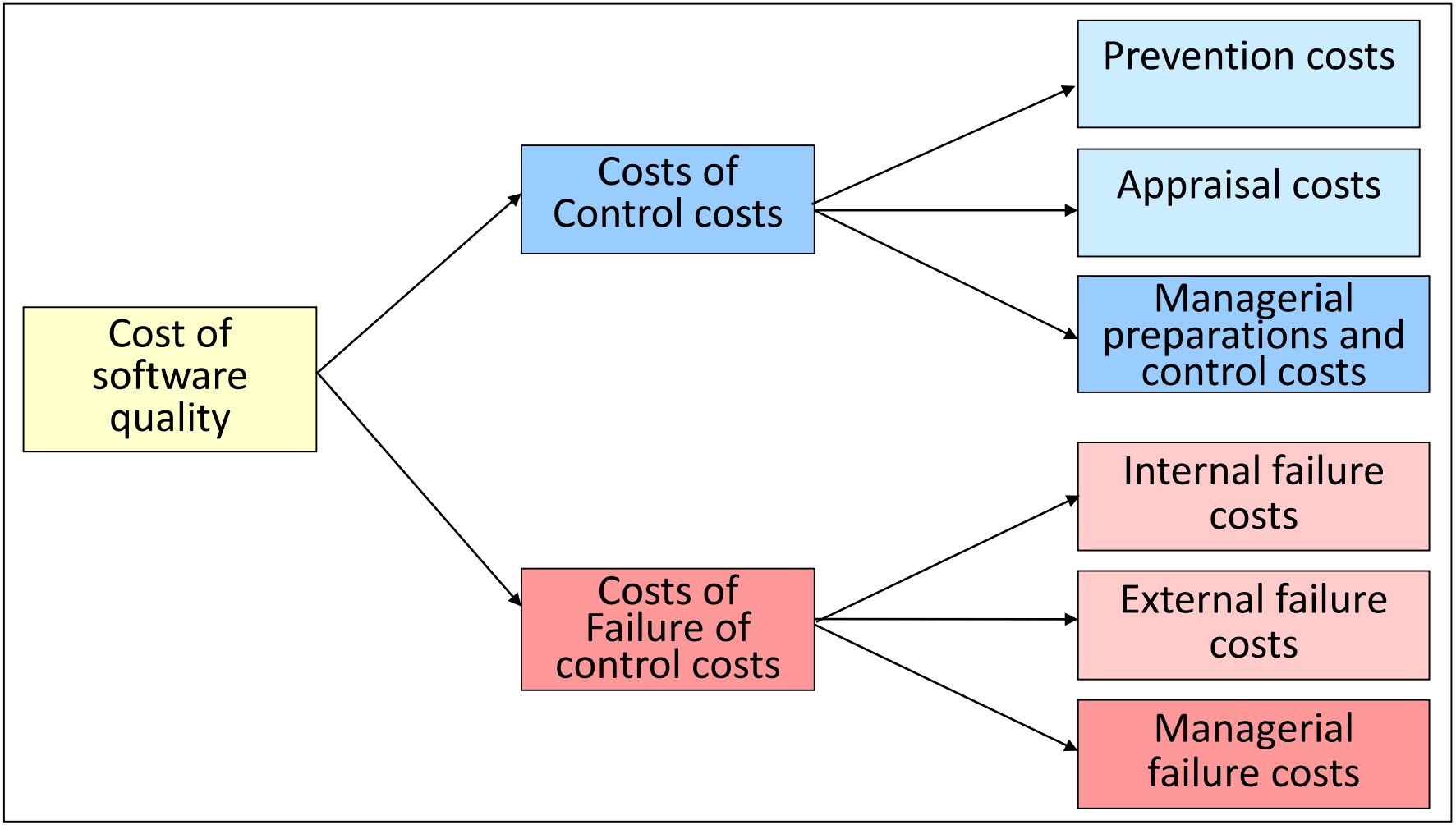
Internal failure costs

* Costs of redesign or design corrections subsequent to design review and test findings
* Costs of re-programming or correcting programs in response to test findings
* Costs of repeated design review and re-testing (regression tests)

External failure costs

- Typical external failure costs cover:
- * Resolution of customer complaints during the warranty period.
- * Correction of software bugs detected during regular operation.
- * Correction of software failures after the warranty period is over even if the correction is not covered by the warranty.
- * Damages paid to customers in case of a severe software failure.
- * Reimbursement of customer's purchase costs.
- * Insurance against customer's claims.
- Typical examples of hidden external failure costs:
- * Reduction of sales to customers that suffered from software failures.
- * Severe reduction of sales motivated by the firm's damaged reputation.
- * Increased investment in sales promotion to counter the effects of past software failures.
- * Reduced prospects to win a tender or, alternatively, the need to under-price to prevent competitors from winning tenders.

Galin's extended mode for cost of software quality



Galin, SQA from Theory to Implementation

Managerial preparation and control costs

* Costs of carrying out contract reviews
* Costs of preparing project plans, including quality plans
* Costs of periodic updating of project and quality plans
* Costs of performing regular progress control
* Costs of performing regular progress control of external participants' contributions to projects

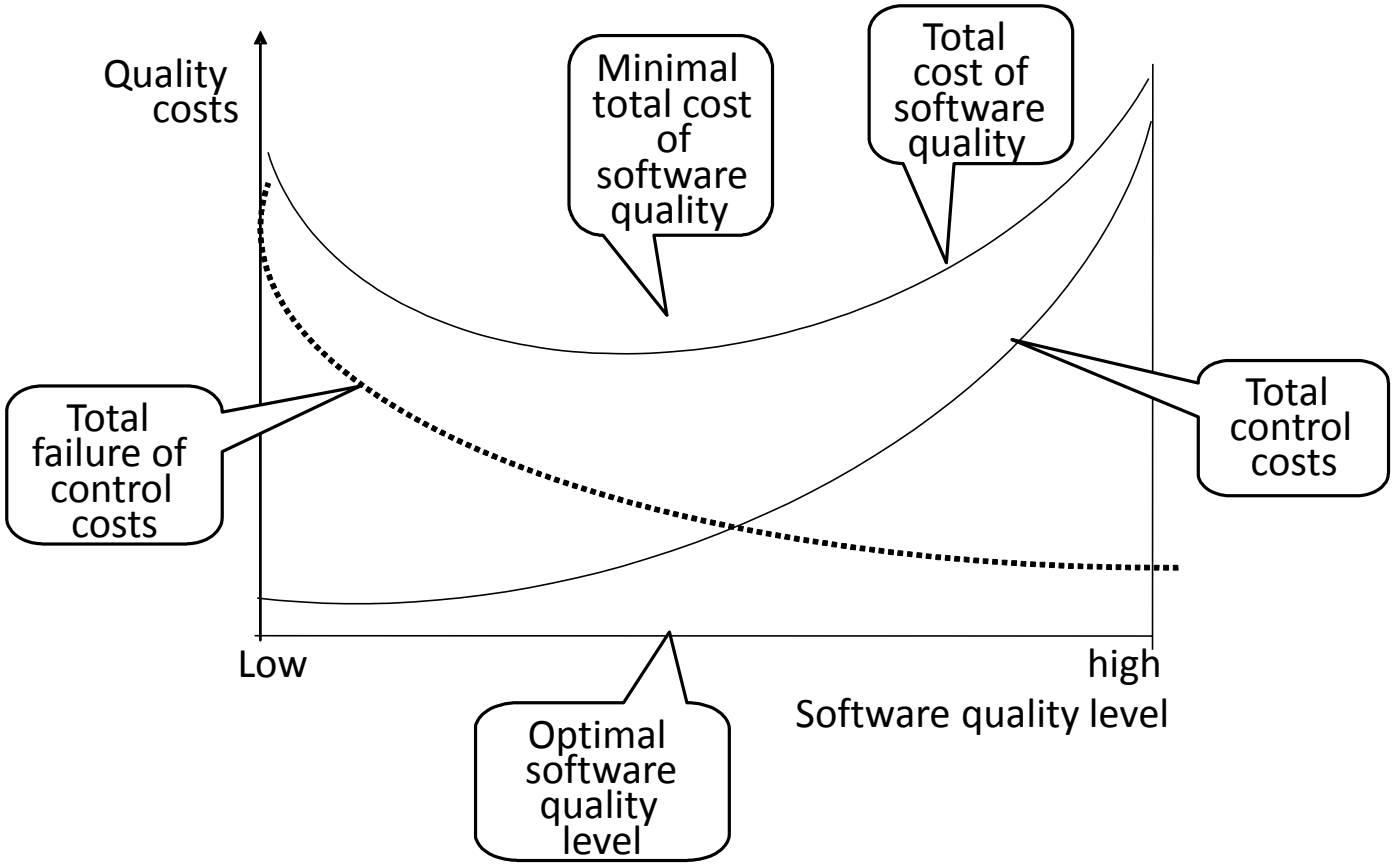
Managerial failure costs

* Unplanned costs for professional and other resources, resulting from underestimation of the resources in the proposals stage.
* Damages paid to customers as compensation for late project completion, a result of the unrealistic schedule in the Company's proposal.
* Damages paid to customers as compensation for late completion of the project, a result of management's failure to recruit team members.
* *Domino effect*: Damages to other projects planned to be performed by the same teams involved in the delayed projects. The domino effect may induce considerable hidden external failure costs.

Application of a cost of software quality system

* Definition of a cost of software quality model and specification of cost items.
* Definition of the method of data collection for each cost item.
* Application of a cost of software quality system, including thorough follow up.
* Actions taken in response to the findings.

Cost of software quality balance by quality level



Galin, SQA from Theory to Implementation

Problems in the application of cost of software quality metrics

General problems

* Inaccurate and/or incomplete identification and classification of quality costs.
* Negligent reporting by team members
* Biased reporting of software costs, especially of “censored” internal and external costs.
* Biased recording of external failure costs - “camouflaged” compensation of customers for failures.

Problems arising when collecting data on managerial costs:

* Contract review and progress control activities are performed in a “part-time mode”. The reporting of time invested is usually inaccurate and often neglected.
* Many participants in these activities are senior staff members who are not required to report use of their time resources.
* Difficulties in determination of responsibility for schedule failures.
* *Payment* of overt and formal *compensation* usually occurs quite some time after the project is completed, and much too late for efficient application of the lessons learned.