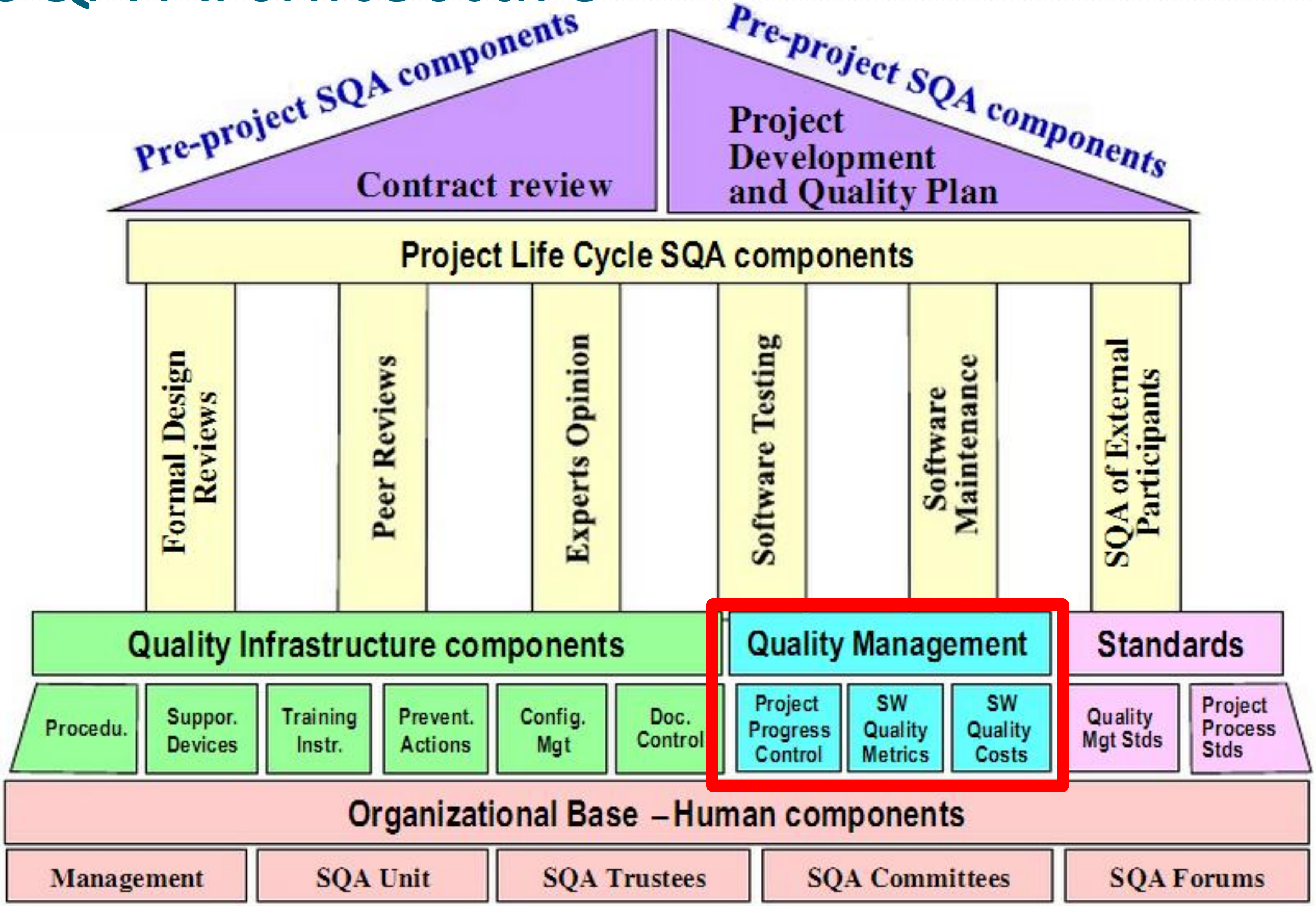
Management components of software quality

1 Overview	2 Life cycle components	3 Infrastructure components	4 Management components	5 Standards and Organizing
6 Static tesing	7 Dynamic testing	8 Test management	9 Tools	

SQA Architecture



Learning objectives

- Explain the **objectives** of project progress control, software quality metrics, costs of software quality measurements
- Explain the components of **project progress control**
- Classify software quality **metrics**
- Compare the **classic model** to the **extended cost model** of software quality

References

- Galin (2004). *Software Quality Assurance from theory to implementation.* Pearson Education Limited
- Ian Sommerville (2011). *Software engineering*. Ninth Edition. Addison-Wesley

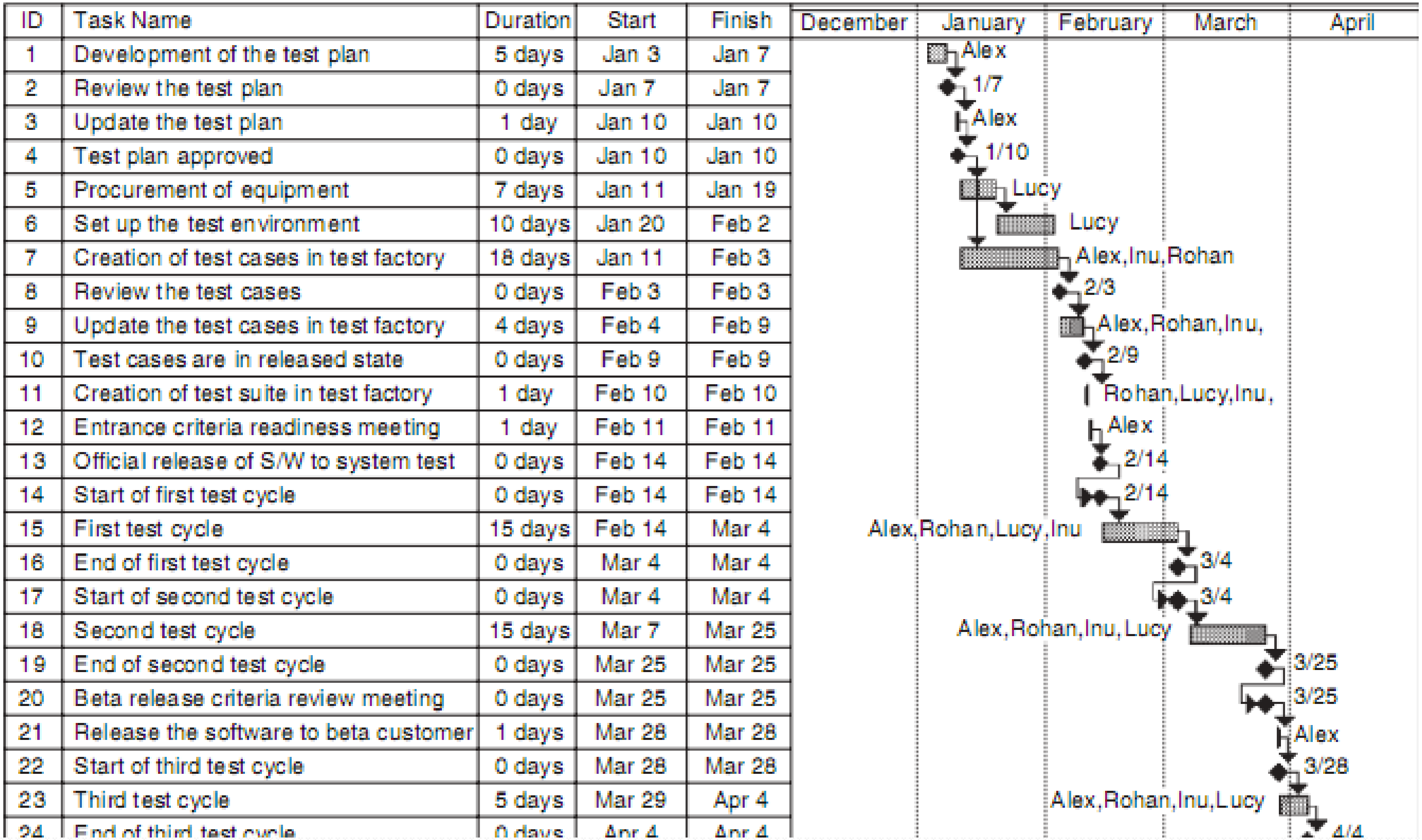
Contents

- Project progress control
- Software quality metrics
- Software quality costs

Project progress control

- Objective
  - immediate: early detection of irregular events
  - long-term: initiation of corrective actions
- The main components
  - **risk** management activities
  - project **schedule** control
  - project **resource** control
  - project **budget** control

Project progress control



ID	Task Name	Duration	Start	Finish
1	Development of the test plan	5 days	Jan 3	Jan 7
2	Review the test plan	0 days	Jan 7	Jan 7
3	Update the test plan	1 day	Jan 10	Jan 10
4	Test plan approved	0 days	Jan 10	Jan 10
5	Procurement of equipment	7 days	Jan 11	Jan 19
6	Set up the test environment	10 days	Jan 20	Feb 2
7	Creation of test cases in test factory	18 days	Jan 11	Feb 3
8	Review the test cases	0 days	Feb 3	Feb 3
9	Update the test cases in test factory	4 days	Feb 4	Feb 9
10	Test cases are in released state	0 days	Feb 9	Feb 9
11	Creation of test suite in test factory	1 day	Feb 10	Feb 10
12	Entrance criteria readiness meeting	1 day	Feb 11	Feb 11
13	Official release of S/W to system test	0 days	Feb 14	Feb 14
14	Start of first test cycle	0 days	Feb 14	Feb 14
15	First test cycle	15 days	Feb 14	Mar 4
16	End of first test cycle	0 days	Mar 4	Mar 4
17	Start of second test cycle	0 days	Mar 4	Mar 4
18	Second test cycle	15 days	Mar 7	Mar 25
19	End of second test cycle	0 days	Mar 25	Mar 25
20	Beta release criteria review meeting	0 days	Mar 25	Mar 25
21	Release the software to beta customer	1 days	Mar 28	Mar 28
22	Start of third test cycle	0 days	Mar 28	Mar 28
23	Third test cycle	5 days	Mar 29	Apr 4
24	End of third test cycle	0 days	Apr 4	Apr 4

Gantt chart for FR–ATM service interworking test project

Control of risk management activities

- Refers to the software development risk items identified in the preproject stage, listed in contract review and project plan documents, together with other risk items
- Systematic risk management activities required:
  - **periodic assessment** about the state of the software risk items
  - based on this reports the project managers are expected to intervene and help arrive at a solution in the more extreme cases

Project schedule control

- Deals with the project's compliance with its approved and contracted **timetables**
- Based mainly on **milestones** in addition to periodic reports
  - milestones set in contracts, especially dates for delivery, receive special emphasis
- Control activities should be focused on **critical delays** (which may effect final completion of the project)
- Management interventions:
  - allocation of additional resources
  - renegotiating the schedule with the customer

Project resource control

- Main control items:
  - �uman resources
  - special development and testing equipment (real-time systems; firmware)
- Control is based on periodic reports of resources used
- True extent of derivations can only be assessed from the point of view of the project progress
- Internal composition of the resource also counts (percentage of senior staff involved, …)

Project budget control

- Based on comparison of actual with scheduled costs
- The main budget items to be controlled
  - human resources
  - development and testing facilities
  - purchase of COTS software
  - purchase of hardware
  - payments to subcontractors
- How to control?
  - based on the milestone reports and other periodic reports

Progress control of internal projects and external participants

- Problem: In practice project control provides only a limited view of the progress of internal software development and an even **more limited view** on the progress made by external project participants
- More significant efforts are required in order to achieve acceptable levels of control for an external project participant due to the **more complex communication and coordination**
- Project progress control of external participants must focus mainly on **the project's schedule and the risks** identified in planned project activities

Project progress control implementation

- Allocation of responsibilities for
  - **person** or **management unit** for progress control
  - **frequency of progress <u>reports</u> required** from each of the unit levels and administrative level
  - situations requiring the project leader to **<u>report</u> immediately** to management
  - situations requiring lower-level management to **<u>report</u> immediately** to upper-level management
- Management audits of project progress

  (1) how well progress reports are transmitted by project leaders and by lower to upper-level management

  (2) specific management control activities to be initiated

Computerized project progress control

- Required for non trivial projects
- Automation can reduce costs considerably
- Examples of services
  - control of risk management activities
  - project schedule control
  - project resource control
  - project budget control

Contents

- Project progress control
- **Software quality metrics**
- Software quality costs

Software quality metrics

- Definition and objectives
- Classification
- Process metrics
- Product metrics
- Implementation of software quality metrics
- Limitations of software metrics

Software quality metrics

- Definition (IEEE, 1993)
  - A **quantitative measure** of the degree to which a system, component, or process possesses a given attribute
- Main objectives
  - to facilitate management **control** as well as **planning** and **execution** of the appropriate managerial interventions
  - to identify situations for development or maintenance process **improvement** (preventive or corrective actions)

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 Volume – Errors Counted

- Software Volume Measures: use **KLOC** or **Function Points**
  - *KLOC* – classic metric that measures the size of software by thousands of code lines
  - *NFP (*Number of Function Points) – a measure of the development resources (human resources) required to develop a program, based on the functionality specified for the software system

- Errors Counted Measures: relate to the **number of errors** or the **weighted number of errors**

Example: Error Counted Measures

	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**

Number of code errors (NCE) vs. weighted number of code errors (WCE)

Process metrics Categories

1. Software process **quality** metrics
    - error density metrics
    - error severity metrics
2. Software process **timetable** metrics
3. Software process **error removal effectiveness** metrics
4. Software process **productivity** metrics

Process metrics

1. Quality metrics: Error density metrics

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

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.

Process metrics
1. Quality metrics: Error density metrics

- Example:

Measures and metrics	Calculation of CED (Code Error Density)	Calculation of WCED (Weighted Code Error Density)
NCE	70	--
WCE	--	192
KLOC	40	40
CED (NCE/KLOC)	1.75	--
WCED (WCE/KLOC)	--	4.8

Process metrics
1. Quality metrics: Error density metrics

- The concept of indicator
  - A software development department may apply two alternative metrics for calculation of code error density: CED and WCED
  - The unit has to determine indicators for unacceptable software quality:
    - CED > 2 and WCED > 4

Process metrics

1. Quality metrics: Error severity metrics

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

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

Process metrics
2. 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.

Process metrics

3. 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 metrics

4. Productivity metrics

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

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

- Refer to the software system’s operational phase
- Customer services are of two main types:
  - Help desk services (HD)
    - **software support** by instructing customers regarding the method of application of the software and solution for customer implementation problems
    - HD metrics are **based on all customer calls**
  - Corrective maintenance services
    - **correction of software failures** identified by customers/users or detected by the customer service team prior to their discovery be the customer
    - corrective maintenance metrics are **based on failure reports**

Product metrics Categories

1. 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
2. HD productivity and effectiveness metrics
3. Corrective maintenance quality metrics
   - software system failures density metrics
   - software system failures severity metrics
   - failures of maintenance services metrics
   - software system availability metrics
4. Corrective maintenance productivity and effectiveness metrics

Product metrics

1. HD quality 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.

Product metrics

1. HD quality metrics: Severity of HD calls metrics

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

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

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

Product metrics
1. HD quality metrics: HD success metrics

- The capacity to solve problems raised by customer calls within the time determined in the service contract

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.

Product metrics

2. 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.

Product metrics
3. Corrective maintenance quality metrics

- Software system failures density metrics

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

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.

Product metrics

3. Corrective maintenance quality metrics

- 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.

Product metrics
3. Corrective maintenance quality metrics

- Failures of maintenance services metrics

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

RepYF = Number of repeated software failure calls (service failures).

NYF = number of software failures detected during a year of maintenance service.

Product metrics
3. Corrective maintenance quality metrics

- Software system availability metrics

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

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.

$NYFH \geq NYVitFH \geq NYTFH$.

$1 - TUA \geq VitA \geq FA$

Product metrics

4. Software corrective maintenance productivity and effectiveness metrics

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

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.

Contents

- Project progress control
- Software quality metrics
- **Software quality costs**

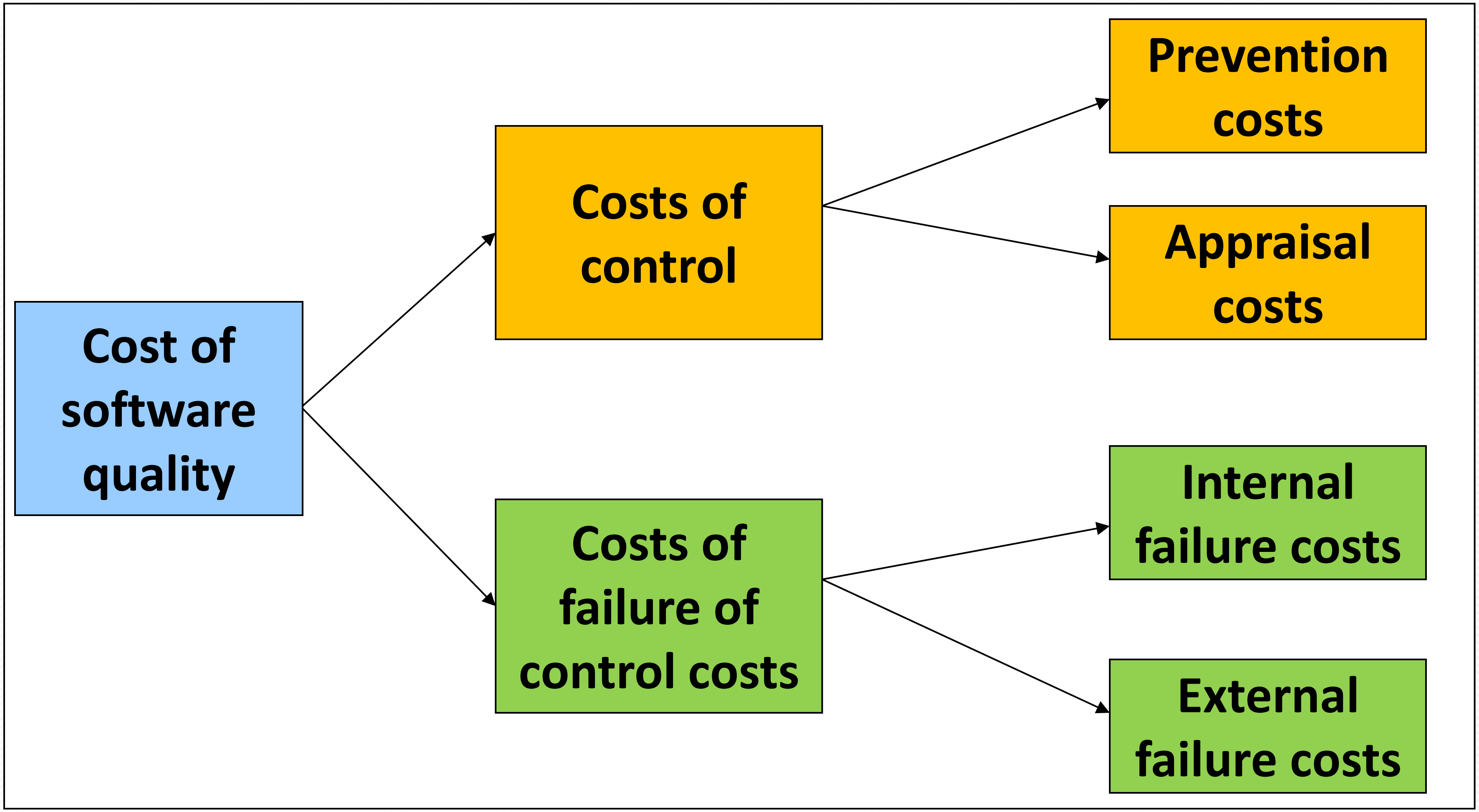
Costs of software quality

- Objectives of cost of software quality metrics
- The classic model
- An extended model
- Application of a cost of software quality system
- Problems in the application of cost of software quality metrics

Objectives of cost of software quality

- **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 SQA activities or to invest in a new or updated SQA infrastructure on the basis of past economic performance

Classic model of cost of software quality



Costs of control
Prevention costs

- Investments in development of new or improved SQA **infrastructure**
  - procedures and work instructions
  - support devices: templates, checklists etc
  - software configuration management system
  - software quality metrics
- 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
- Control of the SQA system through performance of:
  - internal quality reviews
  - external quality audits by customers and SQA system certification organizations
  - management quality reviews

Costs of control
Appraisal costs

- Costs of reviews:
  - formal design reviews (DRs)
  - peer reviews (inspections and walkthroughs)
  - expert reviews
- Costs of software testing:
  - unit, integration and software system tests
  - acceptance tests (carried out by customers)
- Costs of assuring quality of external participants
  - subcontractors, suppliers of COTS software systems and reusable software modules, customers

Costs of failure of control costs
Internal failure costs

- Represent the costs of **error correction** subsequent to **formal examinations** of the software during its development, prior to the system's installation at the customer's site
  - 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)

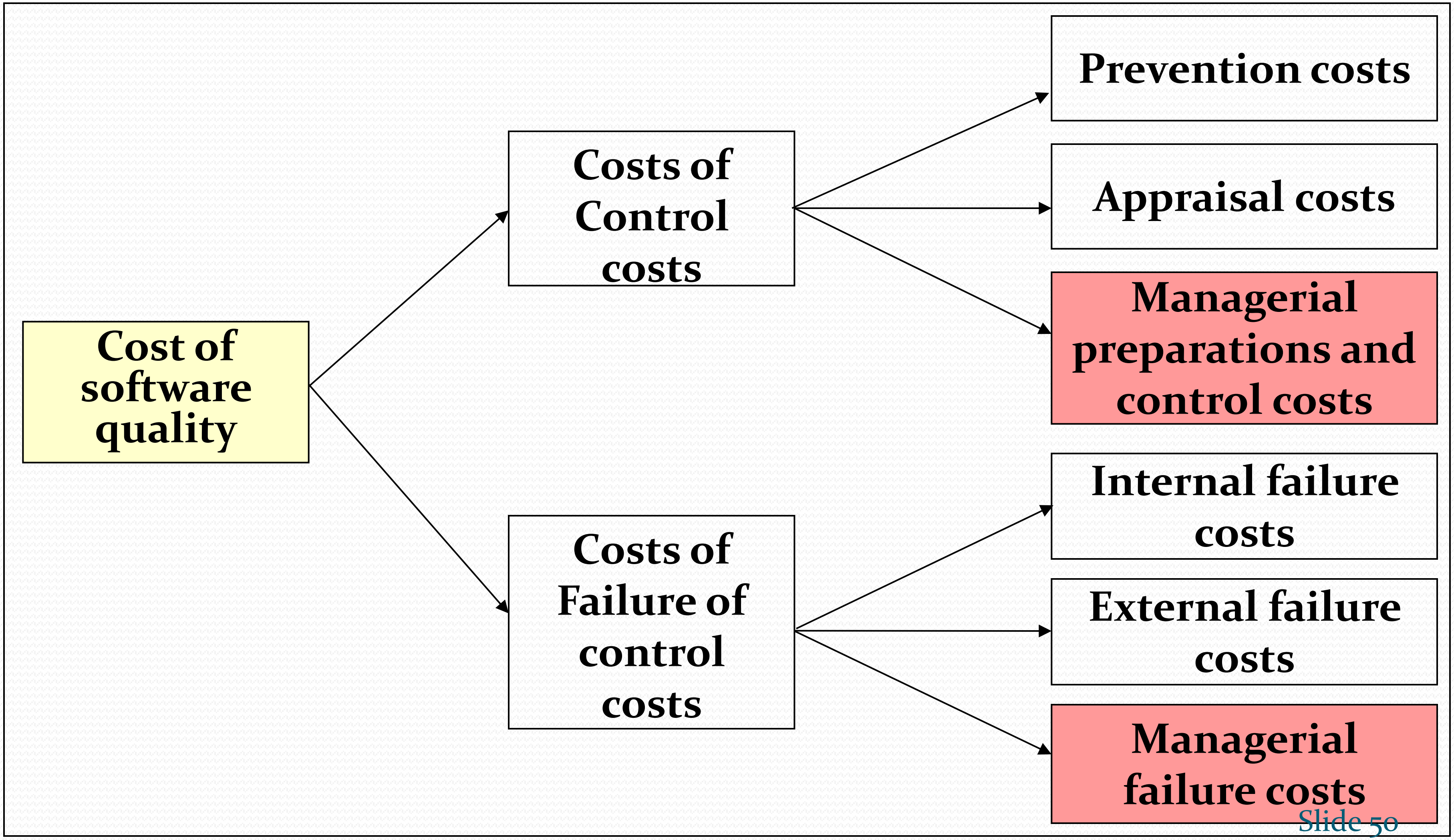
Costs of failure of control costs External failure costs

- Entail the costs of **correcting failures** detected by **customers** or **maintenance teams** after the software system has been installed at customer sites
- Typical external failure costs (“overt” cost)
  - resolution of customer complaints
  - correction of software bugs
  - correction of software failures after the warranty period
  - damages paid to customers
  - reimbursement of customer's purchase costs
  - insurance against customer's claims
  - …

Costs of failure of control costs External failure costs (cont’d)

- 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

Galin’s extended model for cost of software quality



Galin's extended model
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

Galin’s extended model
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
- **Implementation** of a cost of software quality system, including thorough follow up
- Actions taken in **response** to the findings

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