INF5181: Process Improvement and Agile Methods in Systems Development

Lecture 06:
SPI & Measurement

UNIVERSITETET I OSLO

Dr. Dietmar Pfahl

email: dietmarp@ifi.uio.no

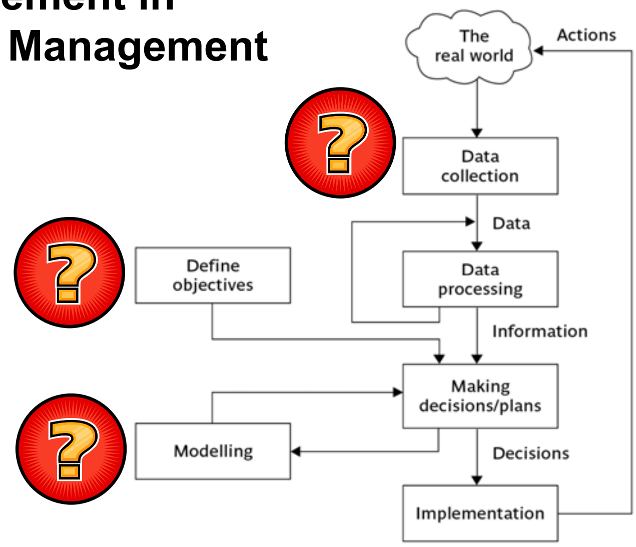
Fall 2011

Structure of Lecture 06

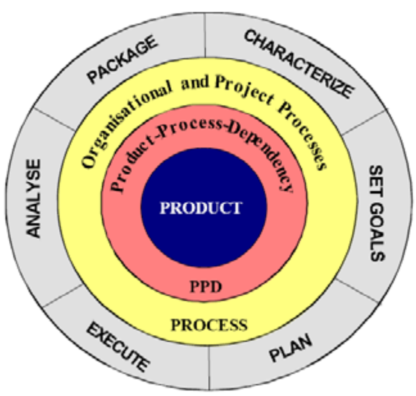
- Hour 1:
  - Introduction & Motivation ←
  - SW Measurement: Why – What – How?
- Hour 2:
  - GQM Process
  - Example Measurement Program
- Hour 3:
  - Question/answer session about project
  - Exercise

UNIVERSITETET I OSLO

Measurement in Project Management



Measurement in SPI ?



The PROFES improvement cycle

PROFES PHASES	PROFES STEPS
CHARACTERIZE	1. VERIFY COMMITMENT
	2. IDENTIFY PRODUCT QUALITY NEEDS
	3. DETERMINE CURRENT PRODUCT QUALITY
	4. DETERMINE CURRENT PROCESS CAPABILITY
SET GOALS	5. SET PRODUCT IMPROVEMENT GOALS
	6. DETERMINE NECESSARY PROCESS CHANGES
PLAN	7. DESCRIBE PROCESS CHANGES
	8. SET METRICS FOR THE PROCESSES AND PRODUCT
	9. PREPARE IMPROVEMENT IMPLEMENTATION
EXECUTE	10. IMPLEMENT AN MONITOR IMPROVEMENTS
ANALYSE	11. EVALUATE RESULTS
PACKAGE	12. UPDATE EXPERIENCE BASE

Phases and steps of the PROFES improvement methodology

Measurement – What & How?

- What to measure?
  - Product
    - Quality
    - Cost
  - Process
    - Capability / Maturity
    - Time
    - Effort
  - Resources
    - Quality
    - Cost
- How to measure?
  - Standards/Frameworks
    - Product quality → ISO 9126
    - Process capability / maturity → ISO 15504 (SPICE) / CMMI
  - GQM (Goal / Question / Metric)

Definition of "(Software) Quality"

- ISO 8402-1986:
  The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs
- ISO 9126-1991:
  The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs

Entity

- ISO 9000-2005:
  Degree to which a set of inherent characteristics fulfills requirements

ISO 9126 software quality characteristics

functionality	does it satisfy user needs?
reliability	can the software maintain its level of performance?
usability	how easy is it to use?
efficiency	relates to the physical resources used during execution
maintainability	relates to the effort needed to make changes to the software
portability	how easy can it be moved to a new environment?

Sub-characteristics of Functionality

- Suitability
- Accuracy
- Interoperability
  - Ability of software to interact with other software components
- Functionality compliance
  - Degree to which software adheres to application-related standards or legal requirements e.g audit
- Security
  - Control of access to the system

Sub-characteristics of Reliability

- Maturity
  - Frequency of failure due to faults - the more the software has been used, the more faults will have been removed
- Fault-tolerance
- Recoverability
  - Note that this is distinguished from ‘security’ - see above
- Reliability compliance
  - Complies with standards relating to reliability

Sub-characteristics of Usability

- Understandability
  - Easy to understand?
- Learnability
  - Easy to learn?
- Operability
  - Easy to use?
- Attractiveness – this is a recent addition
- Usability compliance
  - Compliance with relevant standards

Sub-characteristics of Efficiency

- Time behaviour
  - E.g. response time
- Resource utilization
  - E.g. memory usage
- Efficiency compliance
  - Compliance with relevant standards

Sub-characteristics of Maintainability

- Analysability
  - E. g., ease with which the cause of a failure can be found
- Changeability
  - How easy is software to change?
- Stability
  - Low risk of modification having unexpected effects
- Testability
  - How easy to test?
- Compliance (to standards affecting maintainability)

Sub-characteristics of Portability

- Adaptability
- Installability
- Co-existence
  - Capability of co-existing with other independent software products
- Replaceability
  - Factors giving ‘upwards’ compatibility - ‘downwards’ compatibility is excluded
- Portability conformance
  - Adherence to standards that support portability

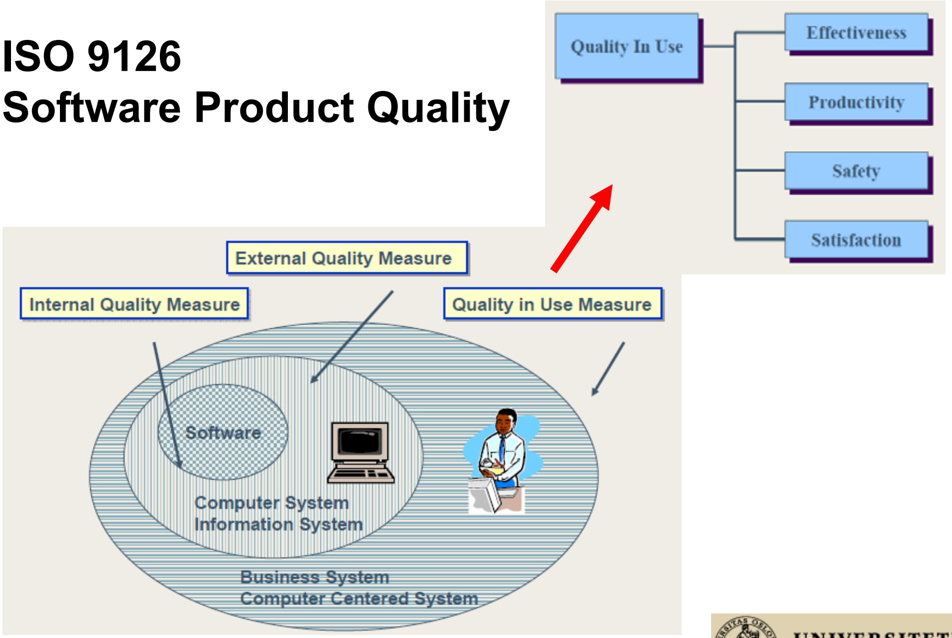
Quality Model: ISO 9126

1 : n relation between Characteristics and Attributes (Sub-Characteristics)

Characteristics	Attributes		
Functionality	Suitability	Interoperability	Accuracy
	Security	Compliance	
Reliability	Maturity	Recoverability	Fault Tolerance
	Compliance		
Usability	Understandability	Learnability	Operability
	Attractiveness	Compliance	
Efficiency	Time Behaviour	Resource Behaviour	Compliance
Maintainability	Analyzability	Stability	Changeability
	Testability	Compliance	
Portability	Adaptability	Installability	Co-existence
	Replaceability	Compliance	

UNIVERSITETET I OSLO

ISO 9126
Software Product Quality



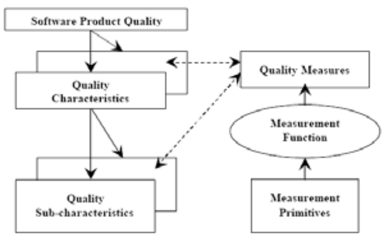
UNIVERSITETET I OSLO

ISO 9126 – Future Developments

- A new series of standards is currently under development.
- Name: Software Product Quality Requirements and Evaluation (SQuaRE - ISO 25000).
- This series of standards will replace the current ISO 9126 (and ISO 14598) series of standards.
  - Note: the new standard will replace the word ”metric” by “measure”

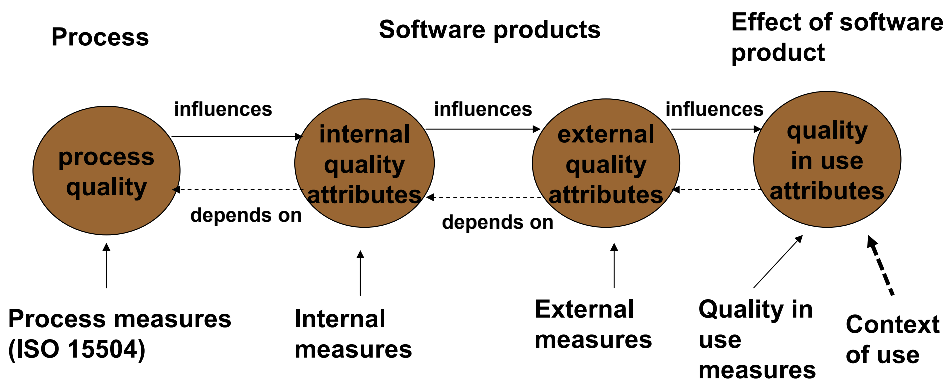
Table 2 WG6 recommended set of Quality Measures

Quality Group Name	Quality Measure Name
Internal Quality Measures	Functional Adequacy
	Precision
	Restartability
	Physical Accessibility
External Quality Measures	Computational Accuracy
	Access Controllability
	Operational Consistency
	Installation Flexibility
Quality in Use Measures	Task Completion
	Productive Proportion
	Discretionary Usage



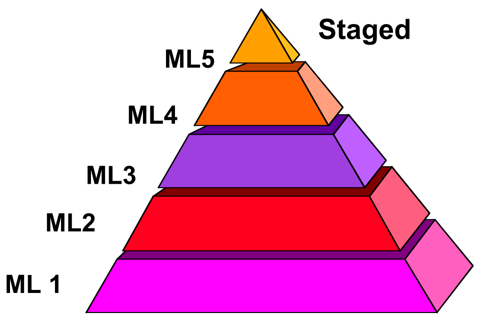
UNIVERSITETET I OSLO

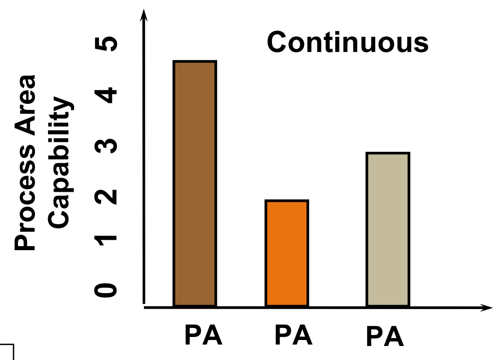
Software Process & Product Quality



UNIVERSITETET I OSLO

Software Process Assessment CMMI / ISO 15504 (SPICE)





Defines 5 maturity levels (MLs); in order to achieve a maturity level all process areas associated to this level, plus all process areas associated with levels below must have a certain minimal capability.

A maturity profile is established based on the capabilities of individual process areas

UNIVERSITETET I OSLO

CMMI Levels and Process Areas (staged)

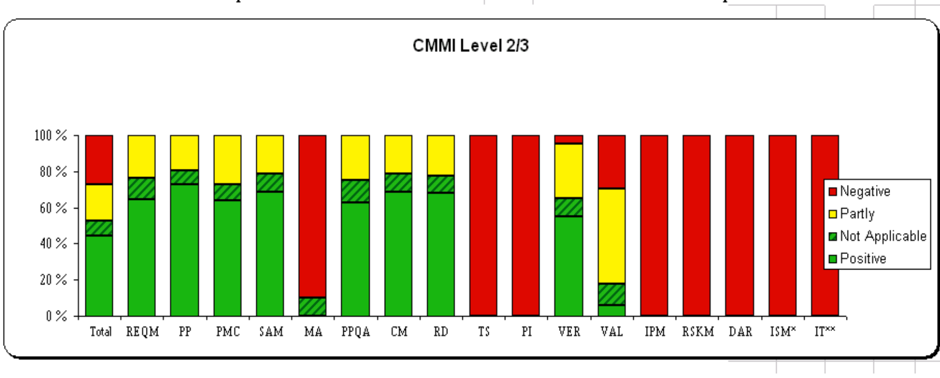
Level	Process Areas
5 Optimizing	Causal Analysis and Resolution Organizational Innovation and Deployment
4 Quantitatively Managed	Quantitative Project Management Organizational Process Performance
3 Defined	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Risk Management Integrated Project Management (for IPPD*) Integrated Teaming* Integrated Supplier Management** Decision Analysis and Resolution Organizational Environment for Integration*
2 Managed	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management
1 Performed	

* Integrated Product/Process Development (IPPD) – add-on to the Engineering processes
** Acquisition – add-on to the Engineering processes

UNIVERSITETET I OSLO

CMM Assessment Results (continuous)

Total CMMI Complience: 63 %
Tailoring: Use of Not Applicable: 8 %
CMMI Level 2 complience: 77 %
CMMI Level 3 complience: 22 %



UNIVERSITETET I OSLO

Structure of Lecture 06

- Hour 1:
  – Introduction & Motivation
  – SW Measurement: Why – What – How?
- Hour 2:
  – GQM Process
  – Example Measurement Program
- Hour 3:
  – Question/answer session about project
  – Exercise

SW Measurement: Who benefits?

- Managers
  – What does each process cost?
  – How productive is development?
  – How good is the product (code, design)?
  – Will the user be satisfied with the product?
  – How can we improve?
- Engineers
  – Are the requirements testable?
  – Have we found all (severe) defects?
  – Have we met our product or process goals?
  – What can we predict about our software product in the future?

Measurement and Measure

Measurement:

- Measurement is the process through which values are assigned to attributes of entities of the real world.

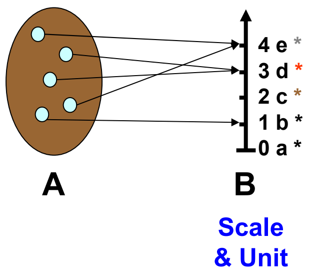
Measure:

- A measure is the result of the measurement process, so it is the assignment of a value to an entity with the goal of characterizing a specified attribute.

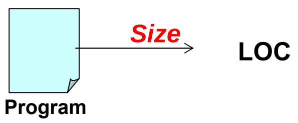
Source: Sandro Morasca, “Software Measurement”, in “Handbook of Software Engineering and Knowledge Engineering - Volume 1: Fundamentals” (refereed book), pp. 239 - 276, Knowledge Systems Institute, Skokie, IL, USA, 2001, ISBN: 981-02- 4973-X.

UNIVERSITETET I OSLO

Measure ~~(Metric)~~



- **Measure:**
  - **Let A be a set of empirical (physical) objects**
  - **Let B be a set of formal objects, such as numbers (or symbols)**
  - **A *measure m* is a mapping from A to B, i.e., m: A → B**



Note:

- **this is neither (exactly) the definition of the mathematical measure, i.e., μ: σ(A) → [0, ∞), with σ(A) is the σ-algebra of A**
- **nor of the mathematical metric, i.e., d: A × A → B with d(x, y) ≥ 0, d(x, y) = 0 if and only if x = y, d(x, y) = d(y, x), and d(x, z) ≤ d(x, y) + d(y, z).**

UNIVERSITETET I OSLO

Measurement: Characterization

- **Relevant objects (entities) may be described, identified, categorized, ordered, and compared in terms of their key properties (attributes)**

- **Measurement is a means of assessing these properties:**
  - **with known reliability**
  - **with known systematic bias (→ validity), if any**
  - **efficiently**
  - **in a manner that is useful for decision-making**

Measurement Scale Types

Scale Type	Characterization	Example (generic)	Example (SE)
Nominal	Divides the set of objects into categories, with no particular ordering among them	Labeling, classification	Name of programming language, name of defect type
Ordinal	Divides the set of entities into categories that are ordered	Preference, ranking, difficulty	Ranking of failures (as measure of failure severity)
Interval	Comparing the differences between values is meaningful	Calendar time, temperature (Fahrenheit, Reaumur, Celsius)	Beginning and end date of activities (as measures of time distance)
Ratio	There is a meaningful "zero" value, and ratios between values are meaningful	Length, weight, time intervals, absolute temperature (Kelvin)	Lines of code (as measure of attribute "Program length/size")
Absolute	There are no meaningful transformations of values other than identity	Object count	Count (as measure of attribute "Number of lines of code")

Measurement Scale Types – cont'd

Scale Type	Admissible Transformation	Indicators of Central Tendency
Nominal	Bijection (one-to-one mapping)	Mode
Ordinal	Monotonically increasing transformation	Mode + Median
Interval	Positive linear transformation $M' = a\,M + b$ ($a>0$)	Mode + Median + Arithmetic Mean
Ratio	Proportionality $M' = a\,M$ ($a>0$)	Mode + Median + Arithmetic Mean + Geometric Mean
Absolute	Identity $M' \equiv M$	Mode + Median + Arithmetic Mean + Geometric Mean

The classification of scales has an important impact on their practical use, in particular on the statistical techniques and indices that can be used.

Example: Indicator of central tendency of a distribution of values ("Location").

Mode = most frequent value of distribution
Median = the value such that not more than 50% of the values of the distribution are less than the median and not more than 50% of the values of the distribution are greater than the median

Scale types and meaningful measurement

- Scales are defined through their admissible transformations
- Scales (and their admissible transformations) help us decide
  - whether a statement involving measures is meaningful
  - what type of statistical analyses we can apply
- Definition of Meaningfulness:

  A statement S with measurement values (i.e., measures m_1, …, m_n) is meaningful iff its truth of falsity value is invariant under admissible transformations Tr.

  iff: "if and only if"

  $Tr(S[m_1, \dots, m_n])$ is true iff $S[Tr(m_1), \dots, Tr(m_n)]$ is true

Software Measurement Challenge

- **Measuring physical properties:**

entity	attribute	unit	scale (type)	value
Human	Height	cm	ratio	178

- **Measuring non-physical properties:**

entity	attribute	unit	scale (type)	value
Human	Intelligence/IQ	index	ordinal	135
Program	Modifiability	?	?	?

- **Software properties are usually non-physical:**
  - **size, complexity, functionality, reliability, maturity, portability, flexibility, maintainability, correctness, testability, coupling, coherence, interoperability, …**

Base vs. Derived Measures

- A measure is base if it directly characterizes an empirical property and does not require the prior measurement of some property
- Derived measure: uses one or more base measures of one or more attributes to measure, indirectly, another supposedly related attribute.
  - Requires first the measurement of two or more attributes
  - Then it combines them using a mathematical model of some kind, according to the laws imposed by the empirical model.

Base Measure: measurement method, scale (or: range), scale type, unit

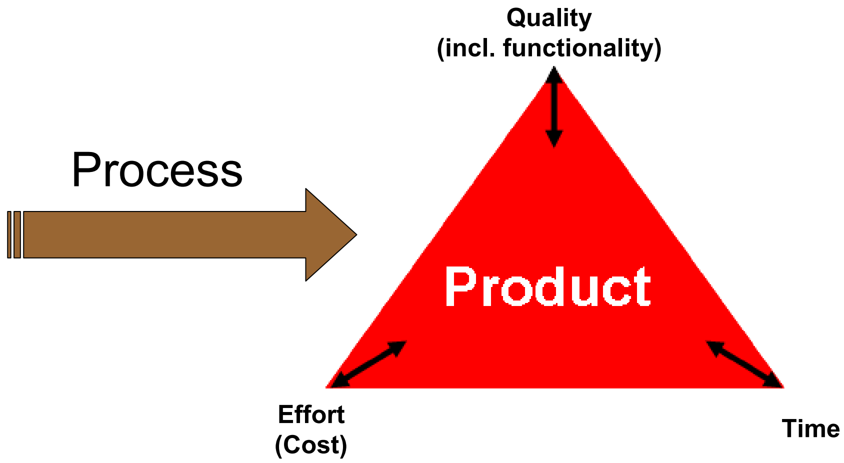
Base Measures	1. Project X Lines of code	2. Project X Hours of effort
Measurement Method	1. Count semicolons in Project X code	2. Add timecard entries together for Project X
Type of Measurement Method	1. Objective	2. Objective
Scale	1. Integers from zero to infinity	2. Real numbers from zero to infinity
Type of Scale	1. Ratio	2. Ratio
Unit of Measurement	1. Line	2. Hour

Derived Measures

- Examples:
  - Productivity
  - Defect density

- Scale of an indirect measure M will generally be the weakest of the scale types of the direct measures $M_1, \ldots, M_n$

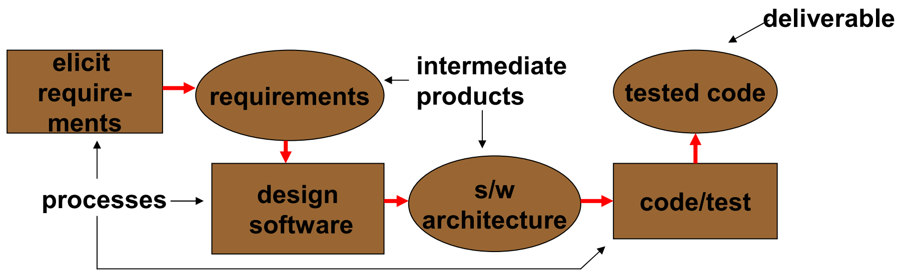
What to Measure?



UNIVERSITETET I OSLO

ISO 12207 development life cycle

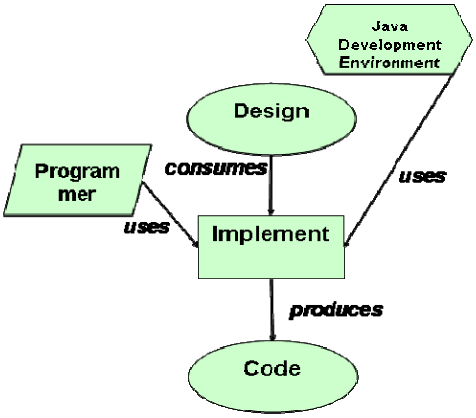
- A development life cycle defines the sequence of processes and activities that will produce the software deliverable and the intermediate products that will pass between the processes/activities.



UNIVERSITETET I OSLO

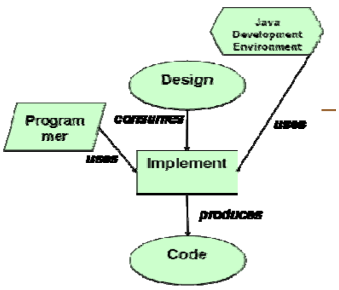
Measurable entities in a process model

- An entity can represent any of the following:
  - Process/Activity: any activity (or set of activities) related to software development and/or maintenance (e.g., requirements analysis, design, testing) – these can be defined at different levels of granularity
  - Product/Artifact: any artifact produced or changed during software development and/or maintenance (e.g., source code, software design documents)
  - Resources: people, time, money, hardware or software needed to perform the processes



Attribute

- An attribute is a feature or property of an entity
  - e.g., blood pressure of a person, cost of a journey, duration of the software specification process
- There are two general types of attributes:
  - Internal attributes can be measured based on the entity itself (→ static)
    - e.g., entity: code, internal attribute: size, modularity, coupling
  - External attributes can be measured only with respect to how the entity relates to its environment (behavior, usage → dynamic)
    - e.g., entity: code, external attribute: reliability, maintainability



Examples of Software Product Attributes

- Size
  - Length, Complexity, Functionality
- Modularity
- Cohesion
- Coupling
- Quality
- Cost

- Quality (→ ISO 9126)
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

Examples of Software Process and Resource Attributes

- Process Efficiency:
  - How fast, how much effort, how much quantity/quality per time or effort unit?
- Process Effectiveness:
  - Do we get the quantity/quality we want?
- Process Maturity:
  - CMMI level
- People/Organisation-related:
  - Skills, knowledge, learning, motivation
- Method/Technique/Tool-related:
  - Effectiveness, Efficiency, Learnability, Cost

UNIVERSITETET I OSLO

Cost (Effort) Measurement

- Effort consumption in the project
  - Includes overtime, excludes non-project related activities like department meetings etc.
  - How to distinguish productive time from unproductive time?
  - How to distinguish defect correction, change management and “pure development"?
  - Allocation of effort over phases / increments / activities?
- Necessary training costs
  - Close competence gap to be able to do the project
- Tool costs
  - Pure purchase and possible license costs
  - (Tool) Training costs
  - Learning curve costs?
- NB: To be able to investigate cost-effectiveness, cost/effort data must be related to amount of produced output/value (→ productivity)

Time Measurement

- Time-to-market is often considered as very important
  - How do you define "time-to-market"?
  - How do you monitor this parameter?
- Time (and its measurement) must be precisely defined!
  - Number of work hours or days, number of calendar days, weeks, months … ???
  - Requires that projects/increments/processes/phases/activities have clearly defined start and end times

Objective vs. Subjective Measurement

- **Objective Measurement**
  - Usually the measurement process can be automated
  - (Almost) no random measurement error, i.e., the process is perfectly reliable
- **Subjective Measurement**
  - Human involvement in the measurement process
  - If we repeat the measurement of the same object(s) several times, we might not get exactly the same measured value every time, i.e., the measurement process is not perfectly reliable

Objective vs. Subjective Measurement (cont'd)

Examples:

- **Subjective Measurement**
  - Classification of defects into severity classes
  - Function Points (when counted manually)
  - Software Process Assessments
- **Objective Measurement**
  - Lines of Code
  - Cyclomatic Complexity
  - Memory Size
  - Test Coverage

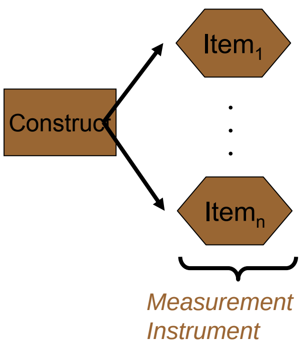
To which category belong …
- Effort ?
- Time ?
- Defect Count ?

Remarks on Subjective Measures

- Well developed subjective measures have proven to be useful
  - e.g., to select suppliers, to identify skill gaps, to assign priorities (e.g., for requirements, defects, etc.)
- It is possible to have objective and subjective measures for the same attribute
  - e.g., measures of code size: LOC and Function Points

- Rule of Thumb:
  - If an objective measure is available, then it is preferable

Basic Concepts in Subjective Measurement

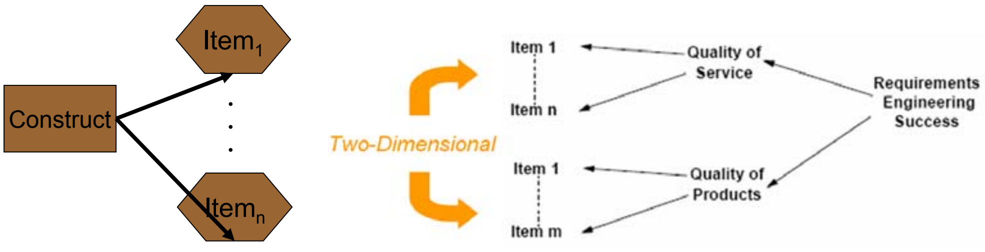


- **Construct:** A conceptual object that cannot be directly observed and therefore cannot be directly measured (i.e., we estimate the quantity we are interested in rather than directly measure it); for example:
  - User Satisfaction
  - Competence of a Software Engineer
  - Efficiency of a Process
  - Maturity of an Organization

- **Item:** A subjective measurement scale that is used to measure a construct
  - A question on a questionnaire is an item

Dimensionality of Constructs

- Constructs can be one-dimensional or multi-dimensional
- If a construct is multidimensional, then each dimension covers a different and distinct aspect of the construct
  - e.g., the different dimensions of customer satisfaction



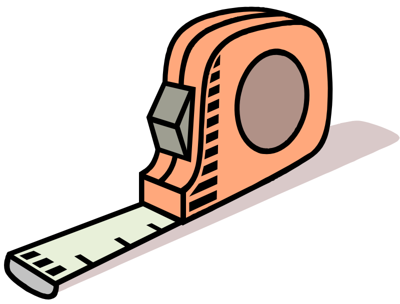
One-Dimensional

Procedures for Subjective Measurement

- Subjective Measures usually entail a well-defined Measurement Procedure that precisely describes:
  - How to collect the data (usually via questionnaires on paper or online)
  - How to conduct interviews
  - How to review documents (software artifacts)
  - In which order to assess the dimensions/items of the instrument, etc.
- Examples: ISO9000 Audit, CMMI/SPICE Assessment, Function Points

Commonly Used Subjective Measurement Scales

- Likert-Type Scale
  - Evaluation-Type
  - Frequency-Type
  - Agreement-Type
- Semantic Differential Scale



UNIVERSITETET I OSLO

Likert Type Scales

- Evaluation-type

  Example:
  - "Familiarity with and comprehension of the software development environment"
    - ❑ Little
    - ❑ Unsatisfactory
    - ❑ Satisfactory
    - ❑ Excellent

- Frequency-type

  Example:
  - "Customers provide information to the project team about the requirements"
    - ❑ Never
    - ❑ Rarely
    - ❑ Occasionally
    - ❑ Most of the time

- Agreement-type

  Example:
  - "The tasks supported by the software at the customer site change frequently"
    - ❑ Strongly Agree
    - ❑ Agree
    - ❑ Disagree
    - ❑ Strongly Disagree

UNIVERSITETET I OSLO

Semantic Differential Scale

- Items which include semantic opposites
- Example:
  - Processing of requests for changes to existing systems: the manner, method, and required time with which the MIS staff responds to user requests for changes in existing computer-based information systems or services.

Slow □ □ □ □ □ □ □ Fast

Timely □ □ □ □ □ □ □ Untimely

UNIVERSITETET I OSLO

Assigning numbers to scale responses

- Likert-Type Scales:

  ❑ Strongly Agree → 1
  ❑ Agree → 2
  ❑ Disagree → 3
  ❑ Strongly Disagree → 4

- Ordinal Scale
- But:
  - Often the distances between the four response categories are approximately (conceptually) equidistant and thus are treated like approximate interval scales.

- Semantic Differential Scale:

  Slow □ □ □ □ □ □ □ Fast
  1 2 3 4 5 6 7

- Ordinal scale, but again, often treated as interval scales

UNIVERSITETET I OSLO

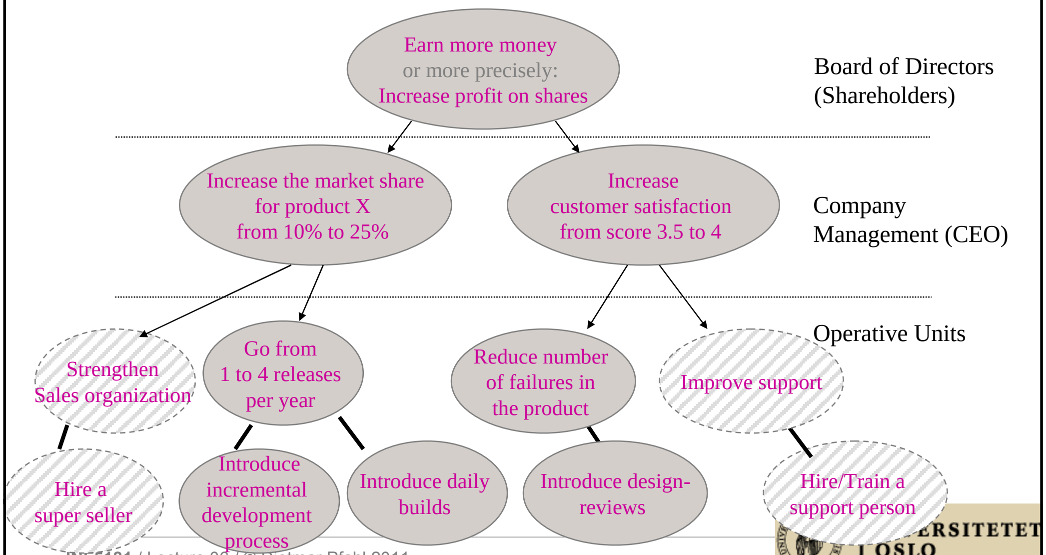
Structure of Lecture 06

- Hour 1:
  - Introduction & Motivation
  - SW Measurement: Why – What – How?
- Hour 2:
  - GQM Process
  - Example Measurement Program
- Hour 3:
  - Question/answer session about project
  - Exercise

How to define a Measurement Program?

- GQM = Goal / Question / Metric (Measure)

Hierarchy of Goals



Business Focus on Quality

Typical Quality-related Goals

- Reduce number of failures in field (i.e., at customer's site)
  - by reducing number of faults in product
  - by abolishing error triggers
  - → has product, process, and people aspects
- Characterise quality
  - → this is often the starting point
  - (see process-related example on next slide)

Typical changes in focus of interest:

- Introduce/alter verification techniques (e.g., inspections) or validation techniques (e.g., new test techniques)
  - to detect more defects (earlier)
- Establish/reorganize quality management
  - to improve defect data collection, storage, analysis, and maintenance
- Introduce better design techniques
  - to reduce possibilities of committing errors
  - to improve readability/testability of artefacts
- Intensify training
  - to reduce the probability of committing errors

Business Focus on Cost … and Time

Typical Cost-related Goals
- Identify cost divers
- Decrease effort
  – by increasing productivity

Typical changes in focus of interest
- New methods (e.g., perspective based reading)
- Design for reuse
- Introduce component-based development (COTS)
- Outsourcing

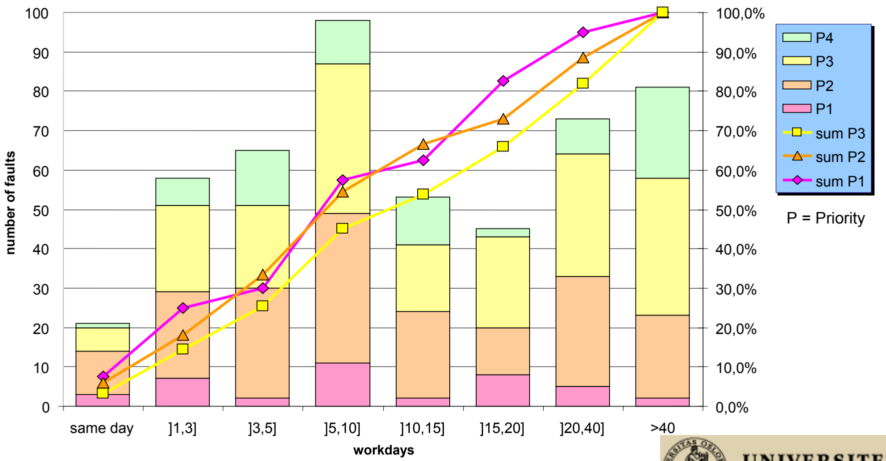
Typical Time-related Goals
- Reduce Time to Market
  – by increasing efficiency

Typical changes in focus of interest
- Product-line development
- Parallel development (concurrent engineering)
- Evaluation of new methods, tools or techniques

UNIVERSITETET I OSLO

Business Focus on Time – Example

How long does it take until defects are removed? (Real World Example)



UNIVERSITETET I OSLO

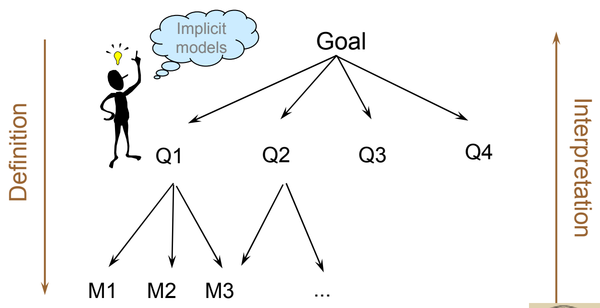
GQM Principles

1. **Goal-Driven:** Define measurement goals (systematically).
2. **Documented:** Document measurement goals and their refinement explicitly.
3. **People-Oriented:** Actively involve all participants during the entire measurement program.
4. **Context-Sensitive:** Consider context/environment when defining measurement goals.
5. **Top-Down:** Refine goals top-down into measures via questions.
6. **Bottom-Up:** Analyze and interpret the collected data bottom-up in the context of the goal.
7. **Sustained:** Measurement is part of a systematic and continuous software quality improvement process.

GQM Core Elements

GQM has three elements:

- Goals
- Questions (and associated Models)
- Measures



INF5181 / Lecture 06 / © Dietmar Pfahl 2011 UNIVERSITETET I OSLO

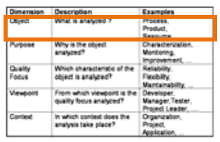
GQM Core Elements: Goals

- GQM goal (or: Measurement Goals) are derived from business or improvement goals
- A GQM goal defines
  - which object is measured,
  - for which purpose,
  - with respect to which quality focus (aspect),
  - from which viewpoint,
  - and in which context (environment).

GQM Goal Template

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement, ...
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager,Tester, Project Leader, ...
Context	In which context does the analysis take place?	Organization, Project, Application, ...

GQM Goal – Object



- **Products**:
  - artifacts (documents) produced during system life cycle phases (e.g., specification, design, programs, test suites)
- **Processes**:
  - software related activities (e.g., specifying, designing, coding, testing, inspecting)
- **Resources**:
  - “items” used by processes in order to produce their outputs (e.g., people, hardware, software, office space)

GQM Goal – Purpose

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager, Tester, Project Leader, ...
Context	In which context does the analysis take place?	Organization, Project, Application, ...

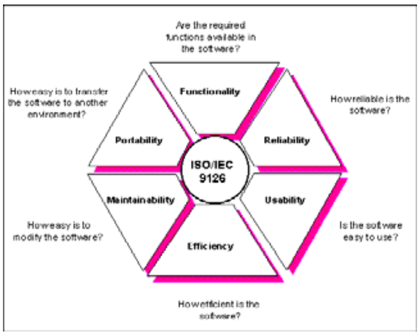
- **Characterization:**
  - aims at forming a snapshot of the current state/performance of the software development processes and products
- **Monitoring:**
  - aims at following the trends/evolution of the performance/state of processes and products
- **Evaluation:**
  - aims at comparing and assessing the quality of products and the efficiency/effectiveness of processes
- **Prediction:**
  - aims at identifying relationships between various process and product factors and using these relationships to predict relevant external attributes of products and processes
- **Control and Change:**
  - aim at identifying causal relationships that influence the state/performance of processes and products
    - Control consists in influencing the course of a project in order to alleviate risks.
    - Change implies modifying the process from project to project in order to improve quality or productivity.
    - Change requires a finer grained understanding of the phenomena under study than control.

UNIVERSITETET I OSLO

GQM Goal – Quality Focus

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager, Tester, Project Leader, ...
Context	In which context does the analysis take place?	Organization, Project, Application, ...

- **Cost**
- **Time-to-Market**
- **Efficiency**
- **Effectiveness**
- **Correctness**
- **Reliability**
- **Reusability**
- **Usability**
- **Maintainability**
- **…**



Quality focus might be aligned to to standards (e.g. ISO 9126)

UNIVERSITETET I OSLO

GQM Goal – Viewpoint

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement,
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager, Tester, Project Leader
Context	In which context does the analysis take place?	Organization, Project, Application,

- **Software Users**
  - interested in the quality and value of the software products
- **Senior Managers**
  - interested in overall understanding, control and improvement across projects in the business unit
- **Project Managers**
  - interested in understanding, control and improvement of the specific software projects they manage
- **Software Engineers**
  - interested in understanding, control and improvement of the specific software project activities and quality of work products in which they are involved
- **Software Process Engineers / Quality Assurance Team**
  - interested in a cross section of what the four previous audiences are interested in

Defines the stakeholder(s) interested in the measurement results.

UNIVERSITETET I OSLO

GQM Goal – Context

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement,
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager, Tester, Project Leader
Context	In which context does the analysis take place?	Organization, Project, Application,

Defines the environment in which the measurement project takes place.

Is important for
- *assessing generalisability (external validity)*
- *future re-use of plans, measurements, and models*

- **Organization**
  - Company, Business Unit, Department, Project, etc.
- **Type of Product**
  - Business Application, MIS, Embedded System, etc.
- **Product Domain**
  - Telecommunication, Transportation Systems, Commerce (banks, insurance companies), medical health care systems, etc.
- **Other**
  - Development history
  - Organizational maturity
  - Platforms / Technologies used, etc. …

UNIVERSITETET I OSLO

GQM Goal – Example

Analyze	test process
for the purpose of	characterization
with respect to (quality aspect)	effectiveness
from the viewpoint of the	test team
in the environment of	project X, organization Y.

UNIVERSITETET I OSLO

GQM Question – Examples

Dimension	Description	Examples
Object	What is analyzed?	Process, **Test** Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Improvement, ...
Quality Focus	Which characteristic of the object is analyzed?	Effectiveness, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager, Tester, Project Leader, ...
Context	In which context does the analysis take place?	Organization, Project, Application, ...

- Goal: Analyze the *test process* for the purpose of *characterization* with respect to (quality aspect) *effectiveness* from the viewpoint of the test team in the environment of project X, organization Y.
- Question 1: How many failures are detected during testing?
- Question 2: When are failures detected (time)?
- Question 3: What types of failures are detected?
- Question 4: How much testing effort is spent?
- Question 5: Which test techniques/tools are applied?
- Etc.

UNIVERSITETET I OSLO

GQM Questions & Models

- *Questions:*
  - Specify verbally the information required to achieve the goal
- *Models:*
  - Specify formally (and make operational) the information required to achieve the goal
  - Type of model depends on goal purpose
  - Models are sometimes called Indicators

Dimension	Description	Examples
Object	What is analyzed ?	Process, Product, Resource
Purpose	Why is the object analyzed?	Characterization, Monitoring, Effectiveness,..
Quality Focus	Which characteristic of the object is analyzed?	Reliability, Flexibility, Maintainability, ...
Viewpoint	From which viewpoint is the quality focus analyzed?	Developer, Manager,Tester, Project Leader, ...
Context	In which context does the analysis take place?	Organization, Project,

UNIVERSITETET I OSLO

GQM: Model Type ↔ Purpose

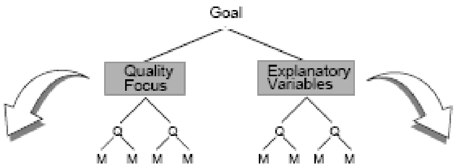
Goal	Purpose	Model Type	Formula
Dimension: Object, Purpose, Quality Focus, Viewpoint, Context	Characterization	Descriptive	$\delta = f(x_1, \dots, x_n)$
	Monitoring	Descriptive	
	Evaluation	Evaluation	$\varepsilon = g(x_1, \dots, x_n)$ $\varepsilon \in \{\varepsilon_1, \dots, \varepsilon_m\}$
	Prediction	Predictive	$\hat{\upsilon} = h(x_1, \dots, x_n)$ $\hat{p}(\upsilon) = h'(x_1, \dots, x_n)$
	Control and Change	Predictive	

UNIVERSITETET I OSLO

GQM Question Categories

- Goal: Analyze the *test process* for the purpose of *characterization* with respect to (quality aspect) *effectiveness* from the viewpoint of the test team in the environment of project X, organization Y.

- In order to help formulate appropriate questions, the goal can be refined into two aspects:
  - Quality focus variables: Characterize quality focus defined by the GQM goal
  - Explanatory variables (or: variation factors): specify parameters that may have an impact on the quality focus: e.g., experience of testers, used test techniques/tools
- Questions may be generated for each of the two aspects



- Question 1: How many failures are detected during testing?
- Question 2: When are failures detected (time)?
- Question 3: How are failures distributed wrt. criticality?

- Question 4: How much testing effort is spent?
- Question 5: Which test techniques/tools are applied?

GQM Measures – Example

- Q3: What is the distribution of failures reported during test by criticality?

  Model refines to …

  –M1.1: Failure count …

  –M3.1: Criticality classification
  - object: reported failure
  - attribute: criticality
  - scale/range: [critical, uncritical, other]
  - scale type: nominal
  - unit: criticality class

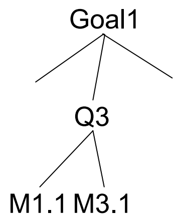
- Q6: How experienced are the development team members?

  Model refines to …

  –…

  –M6.1: Experience classification
  - object: development team member
  - attribute: experience
  - scale/range: [inexperienced, low (< 5 modules developed), medium (5-10 modules developed), high experience (> 10 modules developed)]
  - scale type: ordinal
  - unit: experience class

Developing the GQM Hierarchy

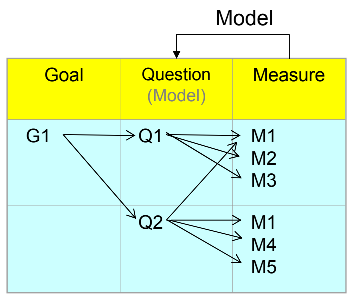


Example GQM Hierarchy (incomplete):

- Question 3: What is the distribution of failures by criticality?
- Model: D = F(x, y) = x[y]/x[all], x = Measure 1.1, y = Measure 3.1, where D: distribution of # failures per criticality class
- Measure 1.1: Failure count (ST: absolute; U: n/a; S: positive integer; O: product version 1.0)
  - *Hypothesis: 120 failures*
- Measure 3.1: Failure criticality (ST: nominal; U: n/a; S: {critical = complete breakdown of system, uncritical = unable to perform one or more of the functions F1, ..., F6, other}, O: failure report)
  - *Hypothesis: 5% critical failures, 15% major failures, 80% minor failures*

GQM Plan

- The models and measures are identified by answering "What kind of information do we need in order to answer the questions?"
- The GQM-tree is documented in tabular form
- Each measure is defined by:
  - Name, ID
  - Scale, unit, etc.
  - Hypotheses



Measurement Plan – Example

- Table for tracing Measurement Plan entries to GQM Plan, Project Plan and Data Collection Forms

Goal-ID	Metric-ID	Metric-Name	Data Creation Event	Data Col. Time	Data Col. Resource	Data Provider	Data Collector	Form-Id
…	…	…	…	…	…	…	…	…
Goal 1	M1.1	Failure count	Failure Report Summary	Test COMPLETE	TOOL: Failure Management System	Tester	QA Manager	Form X
Goal 1	M1.2	Failure criticality	Failure Report	Test report COMPLETE	TOOL: Failure Management System	Tester	QA Manager	Form X
…	…	…	…	…	…	…	…	…
Goal 1	M4.1	Dev. team experience	Project team assignment	Project START	HUMAN: Interview or Questionnaire	Team member	Project Manager	Form Y
Goal 1	M5.1	Document count	CM system report	Test COMPLETE	TOOL: CM system	Developer / Tester	Project Manager	Form Z
Goal 1	M5.2	Document type	Document complete	Test COMPLETE	TOOL: CM system	Developer / Tester	Project Manager	Form Z

GQM Plan | Project Plan | Data Collection Forms

Data Collection – Example Questionnaire

Defect Report Form

Project: ________________

Date: ________________

Name: ________________

Please fill in one report form for each defect you detect.

Defect number: ________________

How much time did you spend to isolate the defect?

_____ h _____ min

How much time did you spend to correct the defect?

_____ h _____ min

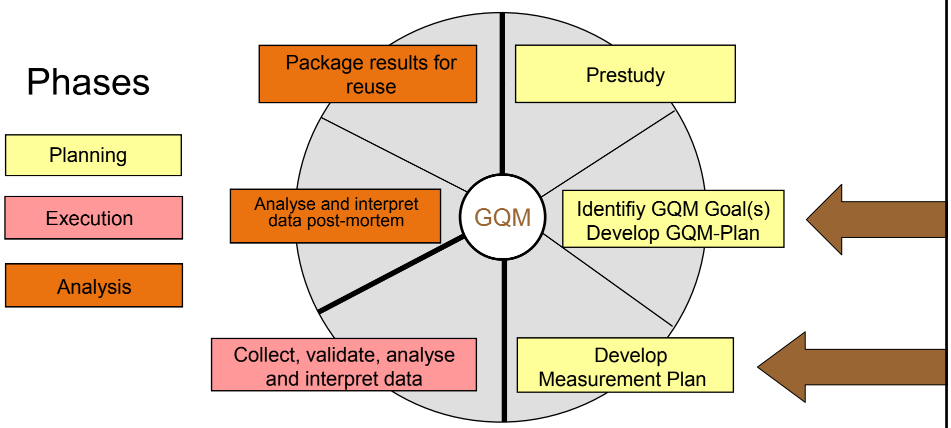
What is the defect type?

- ☐ Calculation
- ☐ Interface
- ☐ Control Flow
- ☐ Other

When was the defect injected?

- ☐ Requirements specification
- ☐ Design
- ☐ Coding
- ☐ Unknown

The GQM Process



Structure of Lecture 06

- Hour 1:
  - Introduction & Motivation
  - SW Measurement: Why – What – How?
- Hour 2:
  - GQM Process
  - Example Measurement Program
- Hour 3:
  - Question/answer session about project
  - Exercise

Software Metrics Initiative at Motorola [Das92]

Why?

- Engineers and managers wanted to better understand the software development process and be able to determine necessary changes to improve productivity, quality, and cycle time.

How?

- Definition of software processes
- Focusing on continuous process and product improvement
- Setting quantitative goals
- Controlling the achievement of goals

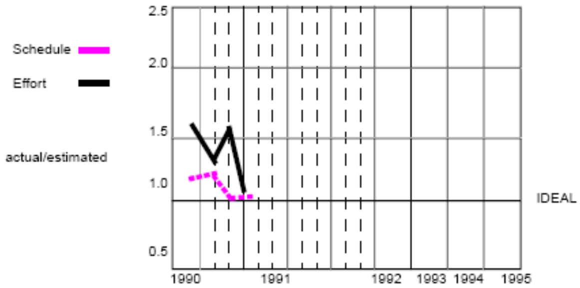
→ Measurement became an integral part of the software development process

Improvement Goals

- Goal 1: Improve project planning
- Goal 2: Increase defect containment
  → ability to detect and correct defects as soon as they are injected
- Goal 3: Increase software reliability
- Goal 4: Decrease software defect density
- Goal 5: Improve customer service
- Goal 6: Reduce cost of non-conformance
- Goal 7: Increase software productivity

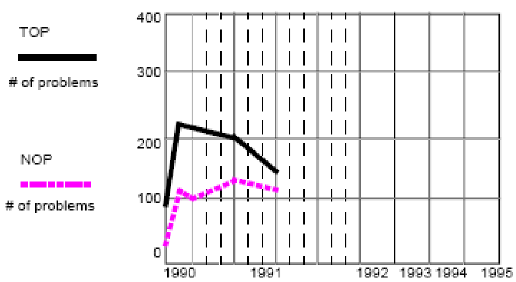
Goal 1: Improve Project Planning

- Question 1.1: How accurate are the estimates of the actual project schedule (duration)?
  - Metric 1.1: Schedule Estimation Accuracy (actual project duration/estimated project duration)
- Question 1.2: How accurate are the estimates of the actual project effort?
  - Metric 1.2: Effort Estimation
    effort)



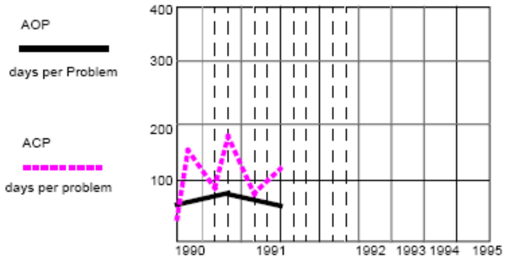
Goal 5: Improve Customer Service

- Question 5.1: What is the number of new problems that were opened during the month?
  - Metric 5.1: New Open Problems (NOP = number of new post-release problems that remain open at the end of the month)
- Question 5.2: What is the total number of open problems at the end of the month?
  - Metric 5.2: Total Open Problem
    that remain open at the end of t



Goal 5: Improve Customer Service (cont'd)

- Question 5.3: What is the mean age of open problems at the end of the month?
- Metric 5.3: (Mean) Age of Open Problems (AOP = total time post-release problems remaining open at end of month have been open / TOP)
- Question 5.4: What is the mean age of problems that were closed during the month?
- Metric 5.4: (Mean) Age of Closed Problems (ACP = total time post-release problems closed within the month were open / number of post-release problems closed within the month)



UNIVERSITETET I OSLO

Use of Metrics for In-Process Project Control

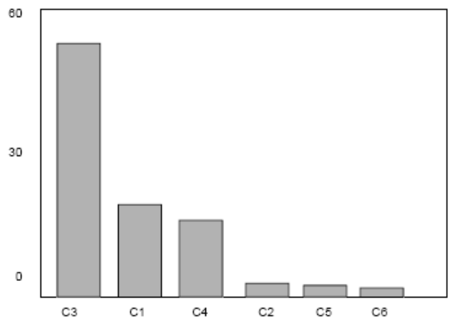
- The charts shown on the previous slides are examples of the so-called "10-up software metrics charts". These can be used for in-process control.
- More detailed data for in-process control includes:
  - Tracking of Life-Cycle Phase / Schedule Progress
  - Cost/Earned Value Tracking
  - Tracking of Impact of Requirements Changes on the project
  - Tracking of Design Progress
  - Fault-Type Tracking
  - Remaining Defects Estimates (e.g., using an assumed Rayleigh curve distribution for fault detection rate)
  - Effectiveness of Reviews (Design, Code)
  - Tracking the fixing of defects per priority/severity class, …

UNIVERSITETET I OSLO

Fault Type Tracking

Purpose:

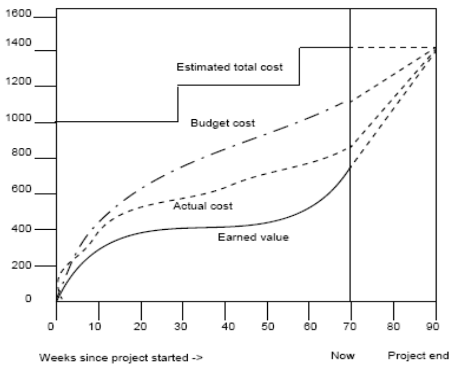
- Understanding (and communicating) the nature of code faults (and possibly their root causes) in order to prevent programmers from injecting similar faults in the future



Cause categories:
C1 - Incorrect or missing initialization of a variable
C2 - Incorrect interface; call of an operation with the wrong parameters.
C3 - Logic problem, the control flow is wrong, the computation of a value is wrong.
C4 - Error handling problem, exception handled incorrectly, the operation has no recovery mechanism when an incorrect input is encountered.
C5 - The definition of a variable is incorrect, the fields of records are incorrectly defined.
C6 - Other

UNIVERSITETET I OSLO

Cost/Earned Value Tracking of the Project



Purpose: to allow manager to track in-process the following cost-related quantities (and update the project plan if necessary):

- Estimated total cost of the project
- Budgeted cumulative cost of the project
- Actual cumulative cost of the project
- Earned value of the project (the sum of the budgeted cost for the activities already completed by the project)

↪ summary of the actual progress of the project and how this relates to the project budget/cost.

UNIVERSITETET I OSLO

Lessons Learnt

- Necessary prerequisites: infrastructure (cost accounting, configuration management, problem reporting), documented process
- Start with a small set of metrics addressing important improvement areas; then evolve over time
- Initial charts were used for in-process control and feedback (→ immediate impact of measurement)
- Data analysis should be done by engineers and managers, not by external experts (= facilitators of the measurement program)
- The code review package deployed by the Metrics Working Group was heavily used (67% of software engineers and managers)
- Metrics can only show problems and trigger corrective action; only if action is implemented benefits can be achieved

➔ Measurement is not the goal. The goal is improvement through measurement, analysis and feedback.

Cost of Measurement at Motorola

- Cost for meetings:
  - Metrics Working Group meetings ~ 8 participants (twice a quarter)
  - Metric User Group meetings (→ feedback sessions) ~ 15 participants (quarterly)
- Additional cost for data collection (incl. providing necessary tools), analysis and meeting preparation (~1% of total project resources)

Data Collection in Agile Projects?

Structure of Lecture 06

- Hour 1:
  - Introduction & Motivation
  - SW Measurement: Why – What – How?
- Hour 2:
  - GQM Process
  - Example Measurement Program
- Hour 3:
  - Question/answer session about project
  - Exercise

Structure of Lecture 06

- Hour 1:
  - Introduction & Motivation
  - SW Measurement: Why – What – How?
- Hour 2:
  - GQM Process
  - Example Measurement Program
- Hour 3:
  - Question/answer session about project
  - Exercise ←

UNIVERSITETET
I OSLO

Next Lecture

- Topic:
  - Problem Solving and Improvement - by Individuals and in Groups
- For you to do:
  - Complete project report (draft)
  - Submit not later than 13:30 by e-mail to dietmarp@ifi.uio.no
  - Only PDF format will be accepted
  - Submission is mandatory!

UNIVERSITETET
I OSLO