Goal-Based Framework for Software Measurement

CHAPTER 3

Classifying Software Measure

The first obligation of any software measurement activity is to identify the entities and attributes that we want to measure. Software entities can be classified as follows:

- *Processes:* Software-related activities
- *Products:* Artifacts, deliverables, or documents that result from a *process* activity
- *Resources:* Entities required by a *process* activity

Within Each class of entity, we distinguish between the internal and external attributes of a *product, process,* or *resource*:

- *Internal attributes:* Attributes can be measured purely in terms of *process, product,* or *resource* itself (e.g., size)
- *External attributes:* Measured based on the behaviors of the entities (e.g., quality)

Sample Process Measures/1

Process Entities	Attributes	Possible Measures
development process	elapsed time	Calendar days, working days
	milestones	Calendar dates
	development effort	Staff-hours, days, or months
	phase containment	Percent of total defects found in phase where introduced
	process compliance	Percent of tasks complying with standard procedures or directives
	performance	Number of tests passed divided by number of tests executed
test process	volume	Number of tests scheduled
	progress	Number of tests executed Number of tests passed

Sample Process Measures/2

Process Entities	Attributes	Possible Measures
detailed design	elapsed time	calendar days, working days
	design quality	defect density: number of design defects found in down-stream activities divided by a measure of product size, such as function points or physical source lines of code.
maintenance	cost	dollars per year staff-hours per change request
Change request backlog	size	number of change requests awaiting service estimated effort (staff-hours) for pending requests

Sample Product Measures/1

Product Entities	Attributes	Possible Measures
system	size	number of modules number of bubbles in a data-flow diagram number of function points number of physical source lines of code number of memory bytes or words required (or allocated)
	defect density	defects per KLOC defects per function point
module	length	physical source lines of code logical source statements
	percent reused	ratio of unchanged physical lines to total physical lines, comments and blanks excluded

Sample Product Measures/2

Product Entities	Attributes	Possible Measures
unit	number of linearly independent flowpaths	McCabe's complexity
document	length	number of pages
line of code	statement type	type names
	how produced	name of production method
	programming language	language name
defect	type	type names
	origin	name of activity where introduced
	severity	an ordered set of severity classes
	effort to fix	staff-hours

Sample Resource Measure

Resource Entities	Attributes	Possible Measures
Assigned staff	team size	number of people assigned
	experience	years of domain experience years of programming experience
CASE tools	type	name of type
	Is used?	yes/no (a binary classification)
time	start date, due date	calendar dates
	elapsed time	days
	Execution time	CPU clocks

Goal-Question-Metric Paradigm

The Goal Question Metric (GQM) approach provides a framework for deriving measures from an organization or business goals.

The GQM framework is mainly divided into three major steps

- Goal: List the major goals of the development or maintenance project
- Question: Derived from each goal the question that must be answered to determine whether the goals are being met
- Metrics: Decide what must be measured in order to be able to answer the questions adequately

Measurement Goal

A measurement goal (or sub goal) is a semi-formal representation of a business goal (or sub goal) composed of 3 components:

- A purpose
- A perspective
- A description of the environment and constraints

Template for Goal Definition

The goals and questions are understood in terms of their audience, e.g, a productivity goal for a project manager may be different from that for a department manager or corporate director.

To aid in generating goals, questions, and metrics consider the following templates:

- **Purpose**: To (characterize, evaluate, predict, motivate, etc.) the (process, product, model, metric, etc.) in order to (understand, assess, manage, engineer, learn, improve, etc.) it.
  Example: To evaluate the maintenance process in order to improve it.

- **Perspective**: Examine the (cost, effectiveness, correctness, defects, changes, product measures, etc.) from the viewpoint of the (developer, manager, customer, etc.)
  Example: Examine the cost from the viewpoint of the manager.

- **Environment:** The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc.
  Example: The maintenance staff consists of poorly motivated programmers who have limited access to tools

Active and Passive Measurement Goals

Active measurement goals are directed toward controlling processes or causing changes to products, processes, or resources. These kinds of goals are found in project management and process improvement activities

Passive measurement goals are meant to enable learning or understanding. Passive goals are often accomplished by characterizing objects of interest according to some productivity or quality model.

Passive measurement goals focus on **observing and recording** normal operations or ongoing conditions without interference, providing a more holistic and real-world picture.

Examples

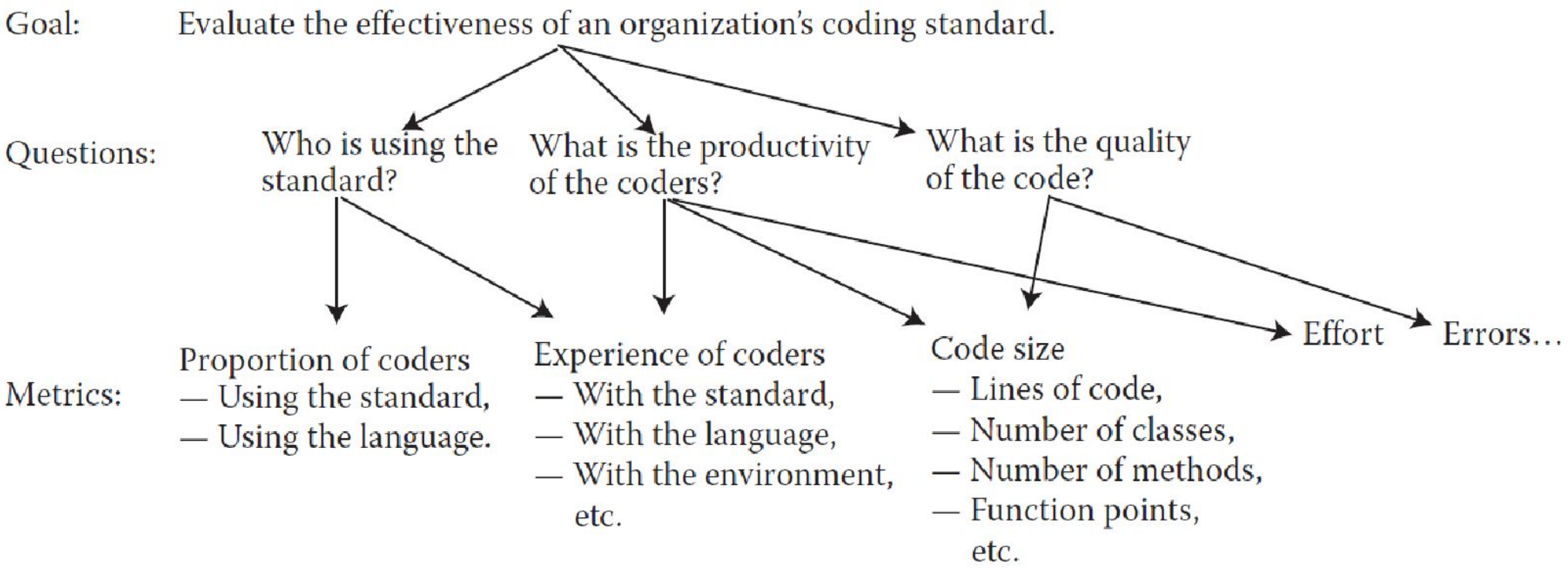
Active Goals

- Meet the scheduled completion date
- Reduce variability
- Improve product reliability
- Improve productivity of the process
- Improve time-to-market
- Reduce employee turnover

Passive Goals

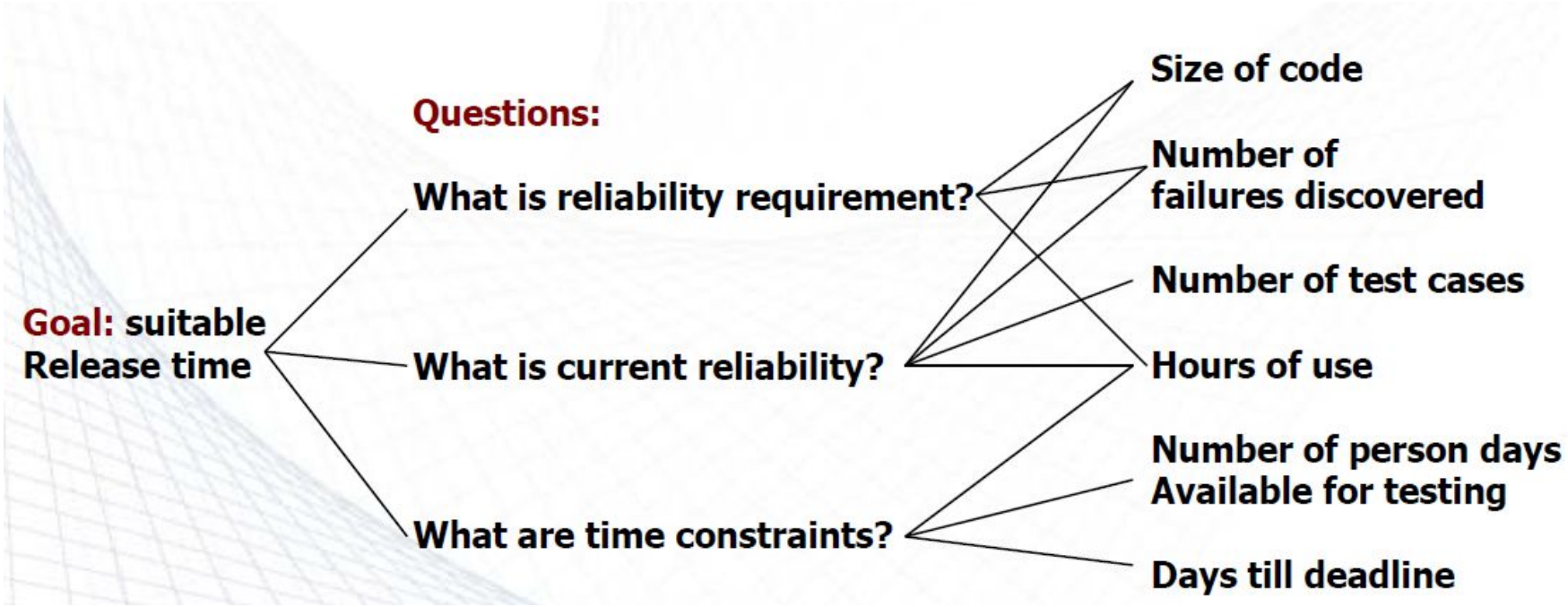
- Understand the current development process
- Identify root causes
- Assess product maintainability
- Identify capabilities and trends, so that we can better predict future performance
- Understand relationships among attributes, so that we can develop models for predicting and estimating

GQM: Example/1



GQM: Example/2

You are a manager of a software development team and you have to decide the release time of your product. Construct a GQM tree related to this goal.



A GQM Case Study Example

Three goals:

- **A:** Maximize customer satisfaction
- **B:** Minimize engineering effort and schedule
- **C:** Minimize defects

Example (Cont..)

GA: Maximize customer satisfaction.

- **QA1:** What are the attributes of customer satisfaction?
  - **MA1:** Functionality, usability, reliability, performance, supportability.
- **QA2:** What are the key indicators of customer satisfaction?
  - **MA2:** Survey, quality function deployment (QFD).
- **QA3:** What aspects result in customer satisfaction?
  - **MA3:** Survey, QFD.
- **QA4:** How satisfied are customers?
  - **MA4:** Survey, interview record, number of customers severely affected by defects.
- **QA5:** How many customers are affected by a problem?
  - **MA5:** Number of duplicate defects by severity

Example (Cont..)

GA: Maximize customer satisfaction.

- **QA6:** How many problems are affecting the customer?
  - **MA6-1:** Incoming defect rate
  - **MA6-2:** Open critical and serious defects
  - **MA6-3:** Defect report/fix ratio
  - **MA6-4:** Post-release defect density
- **QA7:** How long does it take to fix a problem?
  - **MA7-1:** Mean time to acknowledge problem
  - **MA7-2:** Mean time to deliver solution
  - **MA7-3:** Scheduled vs. actual delivery
  - **MA7-4:** Customer expectation of time to fix

Example (Cont..)

GA: Maximize customer satisfaction.

- **QA8:** How does installing a fix affect the customer?
  - **MA8-1:** Time customers operation is down
  - **MA8-2:** Customers effort required during installation
- **QA9:** Where are the bottlenecks?
  - **MA9:** Backlog status, time spent doing different activities

Example (Cont..)

GB: Minimize engineering effort & schedule.

- **QB1:** Where are the worst rework loops in the process?
  - **MB1:** Person-months by product-component-activity.
- **QB2:** What are the total life-cycle maintenance and support costs for the product?
  - **MB2-1:** Person-months by product-component-activity.
  - **MB2-2:** Person-months by corrective, adaptive, perfective maintenance.
- **QB3:** What development methods affect maintenance costs?
  - **MB3:** Pre-release records of methods and post-release costs.

Example (Cont..)

GB: Minimize engineering effort & schedule.

- **QB4:** How maintainable is the product as changes occur?
  - **MB4-1:** Incoming problem rate
  - **MB4-2:** detect density
  - **MB4-3:** Code stability
  - **MB4-4:** Complexity
  - **MB4-5:** Number of modules changed to fix one detect
- **QB5:** What will process monitoring cost and where are the costs distributed?
  - **MB5:** Person-months and costs
- **QB6:** What will maintenance requirements be?
  - **MB6-1:** Code stability, complexity, size
  - **MB6-2:** Pre-release defect density

Example (Cont..)

GB: Minimize engineering effort & schedule.

- **QB7:** How can we predict cycle time, reliability, and effort?
  - **MB7-1:** Calendar time
  - **MB7-2:** Person-month
  - **MB7-3:** Defect density
  - **MB7-4:** Number of detects to fix
  - **MB7-5:** Defect report/fix ratio
  - **MB7-6:** Code stability
  - **MB7-7:** Complexity
  - **MB7-8:** Number of lines to change

Example (Cont..)

GB: Minimize engineering effort & schedule.

- **QB8:** What practices yield best results?
  - **MB8:** Correlations between pre-release practices and customer satisfaction data
- **QB9:** How much do the maintenance phase activities cost?
  - **MB9:** Personal-months and cost
- **QB10:** What are major cost components?
  - **MB10:** Person-months by product-component-activity
- **QB11:** How do costs change over time?
  - **MB11:** Track cost components over entity maintenance life-cycle.

Example (Cont..)

GC: Minimize defects

- **QC1:** What are key Indicators of process health and how are we doing?
  - **MC1:** Release schedule met, trends of defect density, serious and critical detects.
- **QC2:** What are high-leverage opportunities for preventive maintenance?
  - **MC2-1:** Defect categorization
  - **MC2-2:** Code stability
- **QC3:** Are fixes effective with less side effects?
  - **MC3:** Defect report/fix ratio

Example (Cont..)

GC: Minimize defects

- **QC4:** What is the post-release quality of each module?
  - **MC4:** Defect density, critical and serious detects.
- **QC5:** What are we doing right?
  - **MC5-1:** Defect removal efficiency
  - **MC5-2:** Defect report/fix ratio
- **QC6:** How do we know when to release?
  - **MC6-1:** Predicted defect detection and remaining defects.
  - **MC6-2:** Test coverage.

Example (Cont..)

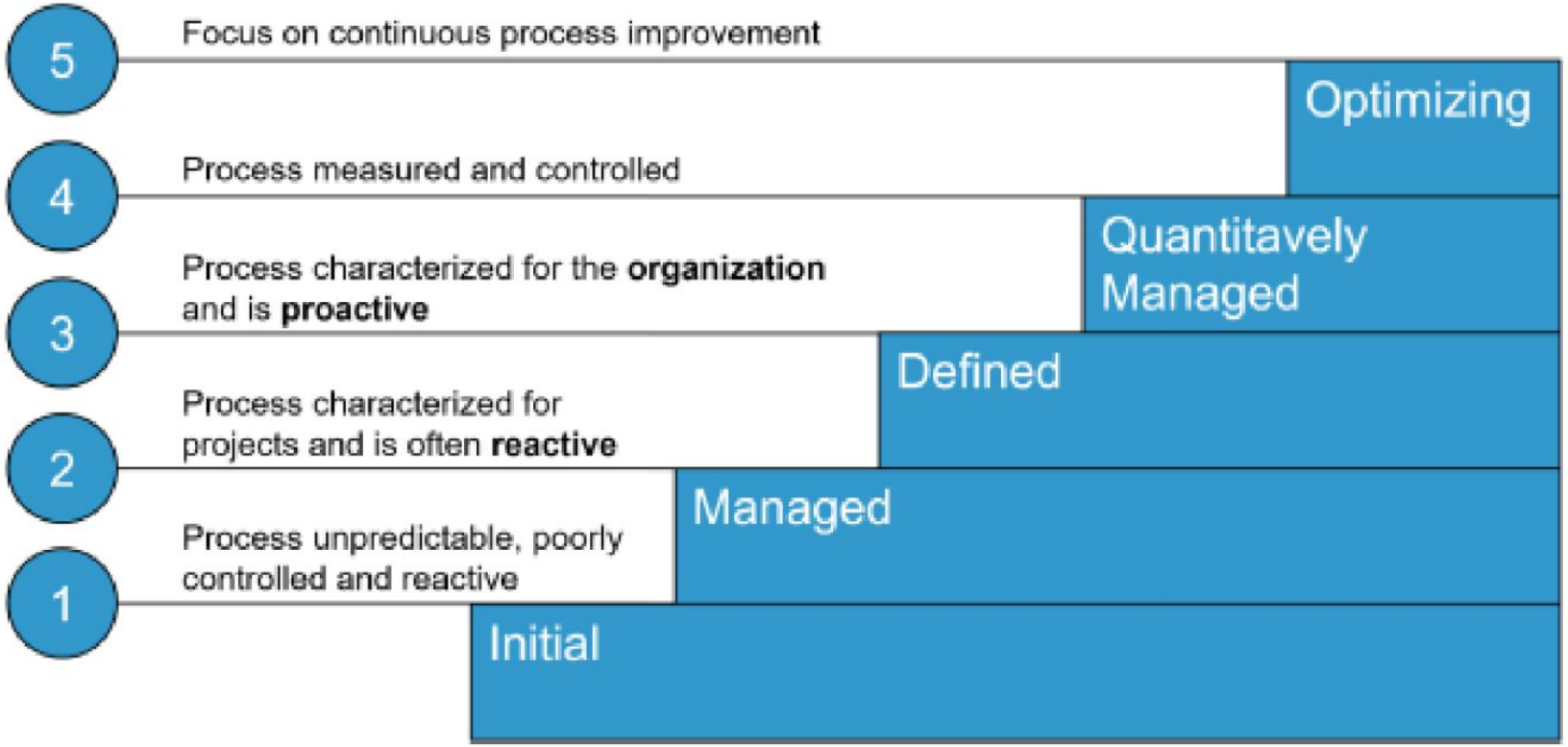
GC: Minimize defects

- **QC7:** How effective is the development process in preventing defects?
  - **MC7:** Post-release detect density
- **QC8:** What can we predict will happen post-release based on pre-release data?
  - **MC8:** Corrections between pre-release complexity, defect density, stability, and customer survey data.
- **QC9:** What defects are getting through and there causes?
  - **MC9:** Detect categorization

Measurement for Process Improvement

- One common goal in the software industry is a process improvement
- Software processes can range from a chaotic and ad hoc, to well-defined and well managed
- A more mature process is more likely to develop software that is reliable, adaptable, delivered on time, and within budget.
- Thus measurement of processes is important to evaluate an organization or company
- The *Capability Maturity Model Integration* (CMMI) is a popular process evaluation technique by the *Software Engineering Institution* (SEI)

Measurement for Process Improvement



REF: *Mario, H. I. R. Z. "An approach supporting integrated modeling and design of complex mechatronics products by the example of automotive applications." (2018).*

Measurement for Process Improvement

- The SEI CMMI distinguishes one level from another in terms of key process activities going on at each level.
- Specific goals, questions, and metrics are developed to assess whether an organization has reached a particular level.

- ***Please see page number 106 in the referenced book.***

1. *Initial*: Level 1 processes are ad hoc and "success depends on the competence and heroics of the people in the organization."
2. *Managed*: Level 2 processes are planned; "the projects employ skilled people … have adequate resources … involve relevant stakeholders; are monitored, controlled, and reviewed …."
3. *Defined*: Level 3 "processes are well characterized and understood, and are described in standards procedures, tools, and methods."
4. *Quantitatively managed*: A Level 4 "organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing projects."
5. *Optimizing*: A Level 5 "organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs" (CMMI Product Team 2010).

Combining GQM with Process Maturity

Suppose you are using the Goal-Question-Metric paradigm to decide what your project should measure. You may have identified at least one of the following **high-level goals:**

- Improving productivity
- Improving quality
- Reducing risk

Within each category, you can represent the goal's satisfaction as a set of subgoals

For example, the goal of improving productivity can be interpreted as several subgoals affecting resources:

- Assuring adequate staff skills
- Assuring adequate managerial skills
- Assuring adequate host software engineering technology

Combining GQM with Process Maturity

if you have chosen improving quality with a sub goal of improving the quality of the requirements, then the related questions might include:

- Is the set of requirements clear and understandable?
- Is the set of requirements testable?
- Is the set of requirements reusable? etc.

Now, suppose you want to answer the question: Is the set of requirements maintainable?

Combining GQM with Process Maturity

- Level 1, then the project is likely to have ill-defined requirements. Measuring requirements characteristics is difficult at this level, so you may choose to count the number of requirements and changes to those requirements to establish a baseline
- Level 2: the requirements are well defined you can collect additional information: the type of each requirement (database requirements, interface requirements, performance requirements, etc.) and the number of changes to each type.
- Level 3: your visibility into the process is improved, and intermediate activities are defined, with entry and exit criteria for each activity. For this level, you can collect a richer type of measurement: e.g. - traceability