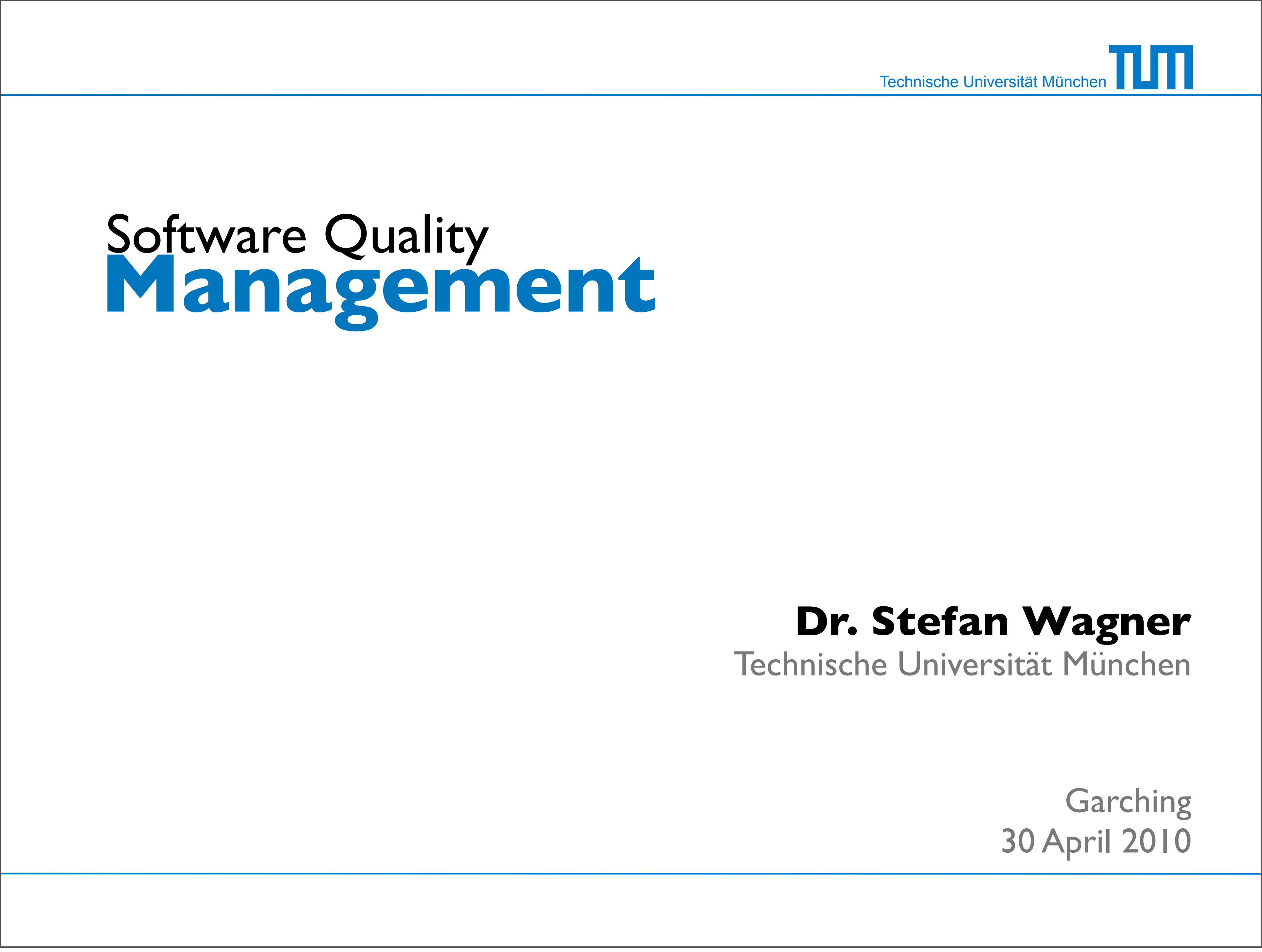


Technische Universität München

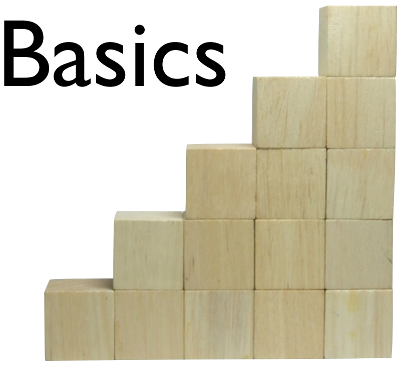
Software Quality Management

Dr. Stefan Wagner
Technische Universität München

Garching
30 April 2010

The question for today is:
What is the difference between functional and quality requirements?

Basics



Product

Quality

Metrics and

Measurement

Process

Quality

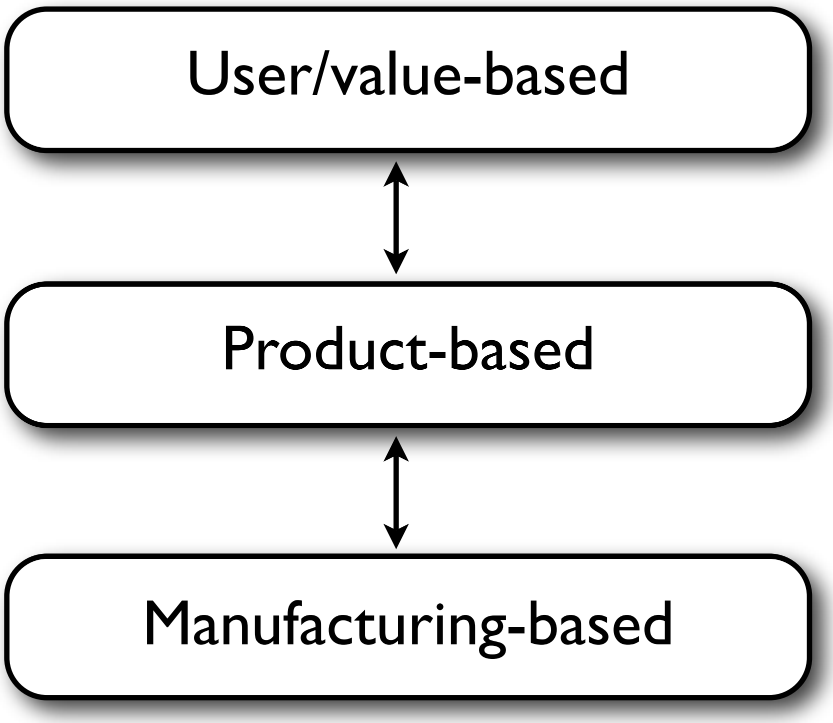
Quality

Management

Certifi-
cation

Short discussion of last week's question

Garvin's quality approaches


User/value-based

Product-based

Manufacturing-based

Garvin, What does product quality really mean?, 1984

In our discussion so far, we mainly considered the stakeholders, i.e., the user/value view. Then we discussed quality attribute and quality models for that, i.e., the product view. Now we come to the manufacturing or process view.

Process quality

Requirements

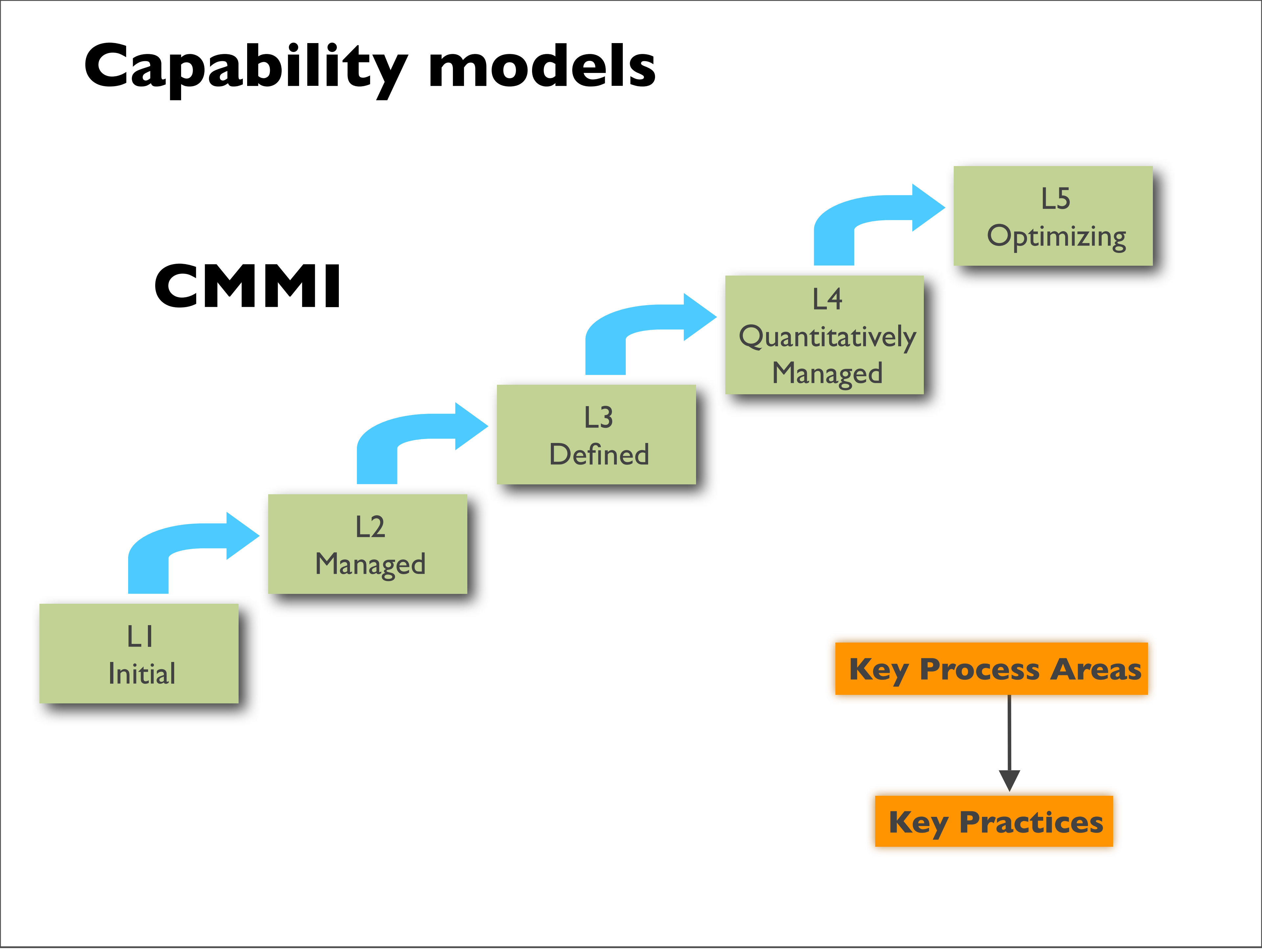
Design

Implementation

Test

Maintenance

Process quality is not directly concerned with the quality of the final product.
It looks at how well the process is done (also how good it is suited to produce a quality product).
The assumption is that a high quality process will produce high quality products.
On average this is true, but not in general.



Capability models

CMMI

L1
Initial

L2
Managed

L3
Defined

L4
Quantitatively Managed

L5
Optimizing

Key Process Areas

Key Practices

Capability or maturity models are a common means to analyse the process quality ("the maturity of the process").

The most prominent examples are SPICE and CMMI.

The levels in CMMI are characterised by:

L1: Ad-hoc project execution

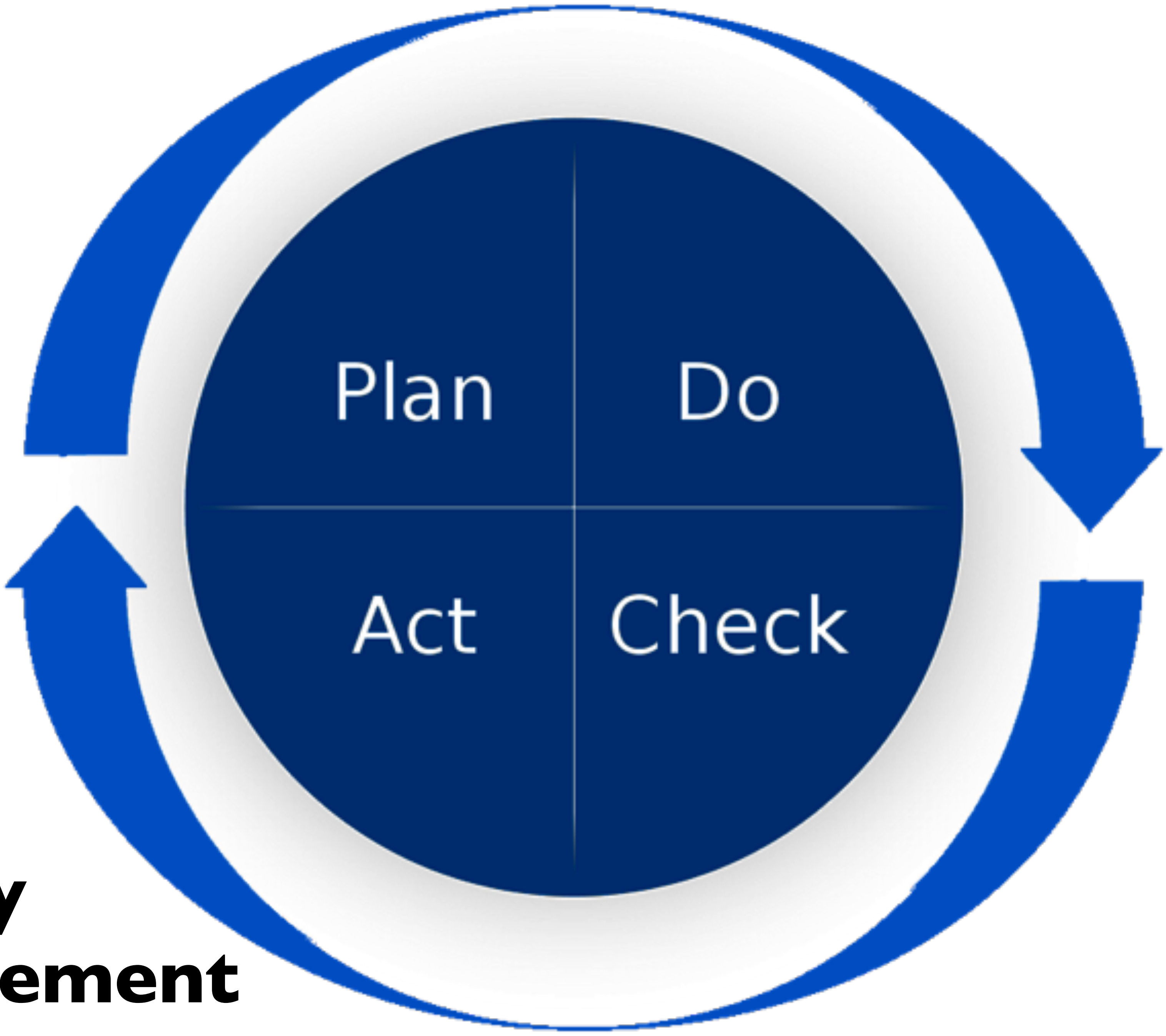
L2: Structured project execution

L3: Company-wide defined processes

L4: Quantitative control

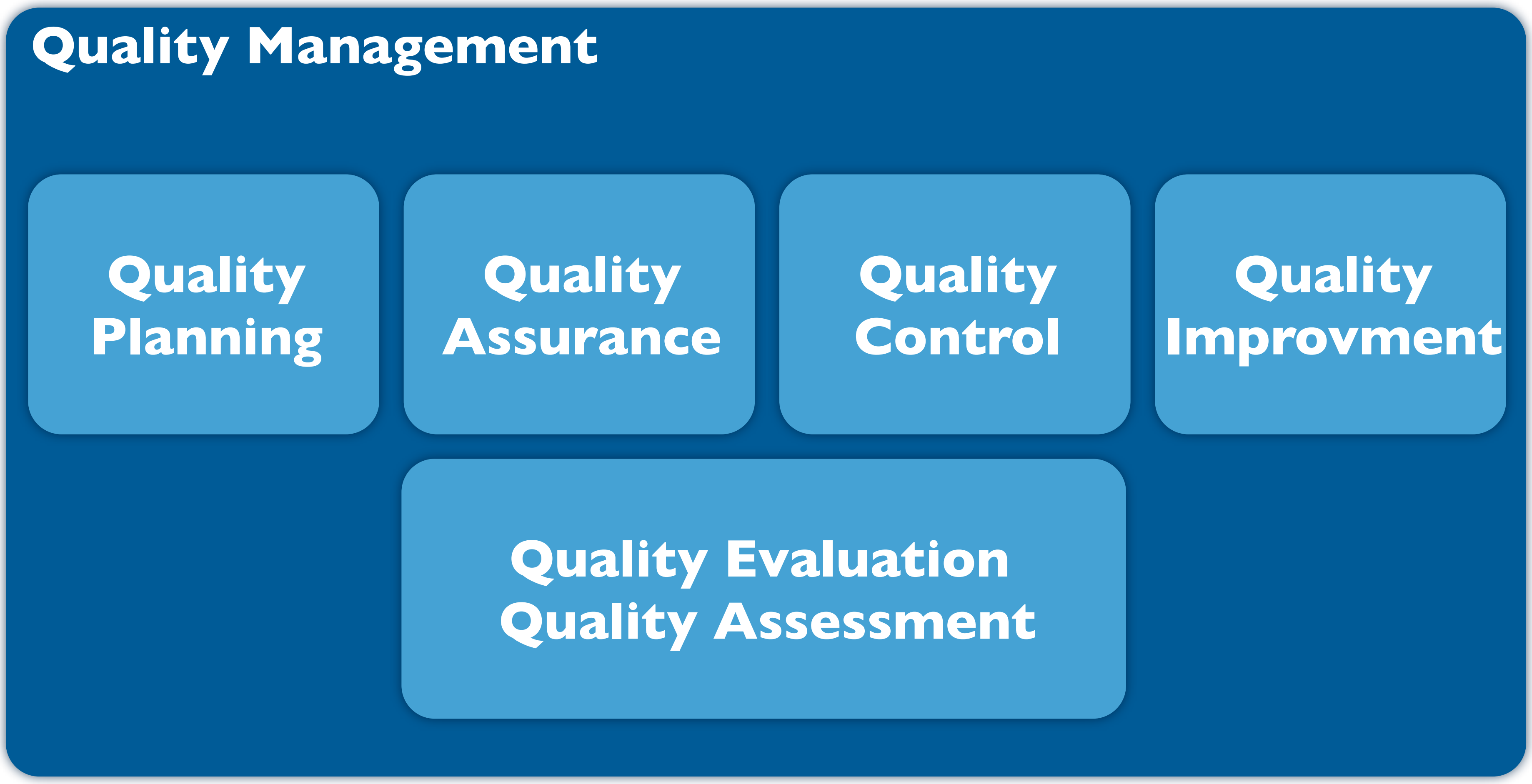
L5: Continuous process optimisation

Quality Management



Deming cycle, **Shewhart cycle**, **PDCA cycle**

Quality management includes all the activities that organizations use to direct, control, and coordinate quality. These activities include formulating a quality policy and setting quality objectives. They also include quality planning, quality control, quality assurance, and quality improvement.



Quality Management

Quality Planning

Quality Assurance

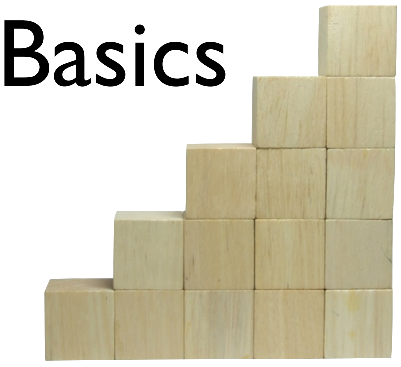
Quality Control

Quality Improvment

Quality Evaluation
Quality Assessment

Quality management contains several other activities.
The terminology differs in the literature. It depends what standard or book you use.
It usually contains
* quality planning that is considered with specifying quality and planning the activities for achieving quality
* quality assurance that are activities that should raise confidence that the quality requirements are achieved
* quality control that checks whether the quality requirements have been achieved
* quality improvment that contains activities for improving anything related to quality
* quality evaluation or assessment that checks the current level of quality

Basics



Product Quality

Metrics and

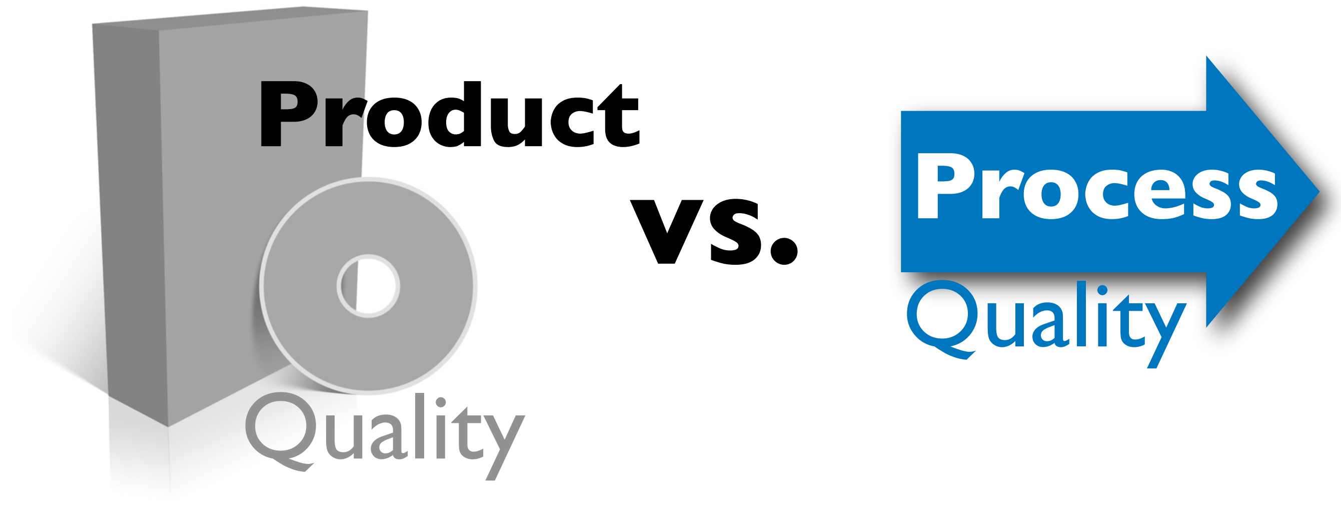
Measurement

Process Quality

Quality

Management

Certification



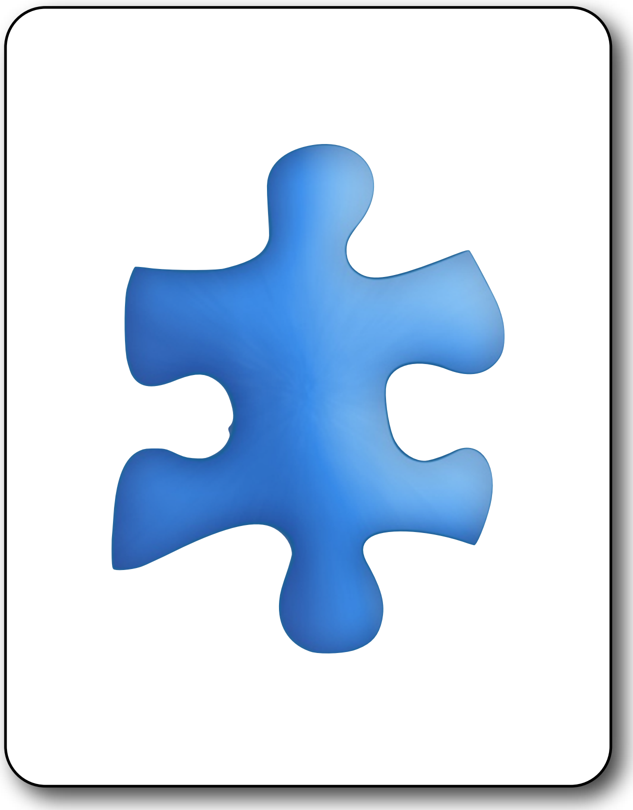
Product Quality **vs.** **Process** Quality

What is the difference?
What is more important?

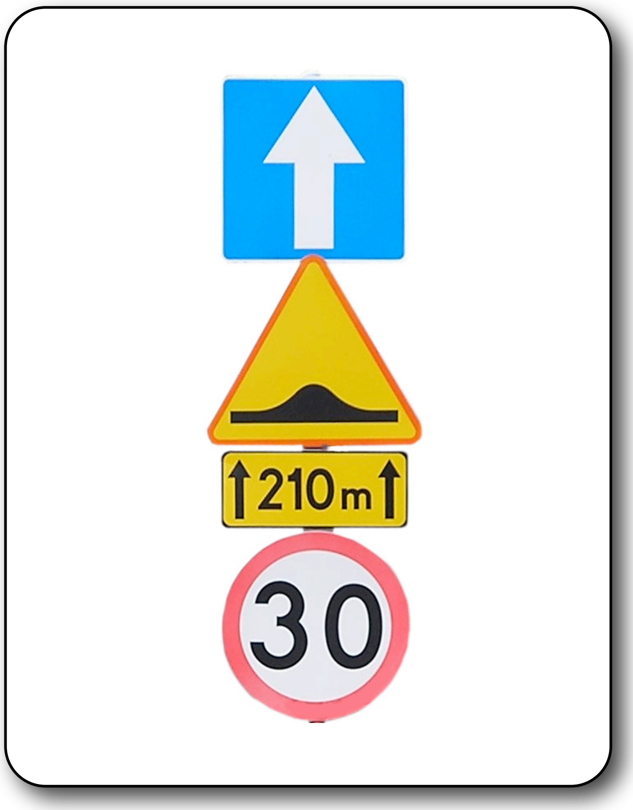
What is the difference between product quality and process quality? What is more important?

Process quality looks at the development process. If the process is good, the quality of the product is supposed to be high.
Product quality looks directly at the product. What properties does the product have that make it of high quality?
In the end, product quality is what matters for the stakeholders. Process quality is an important prerequisite.

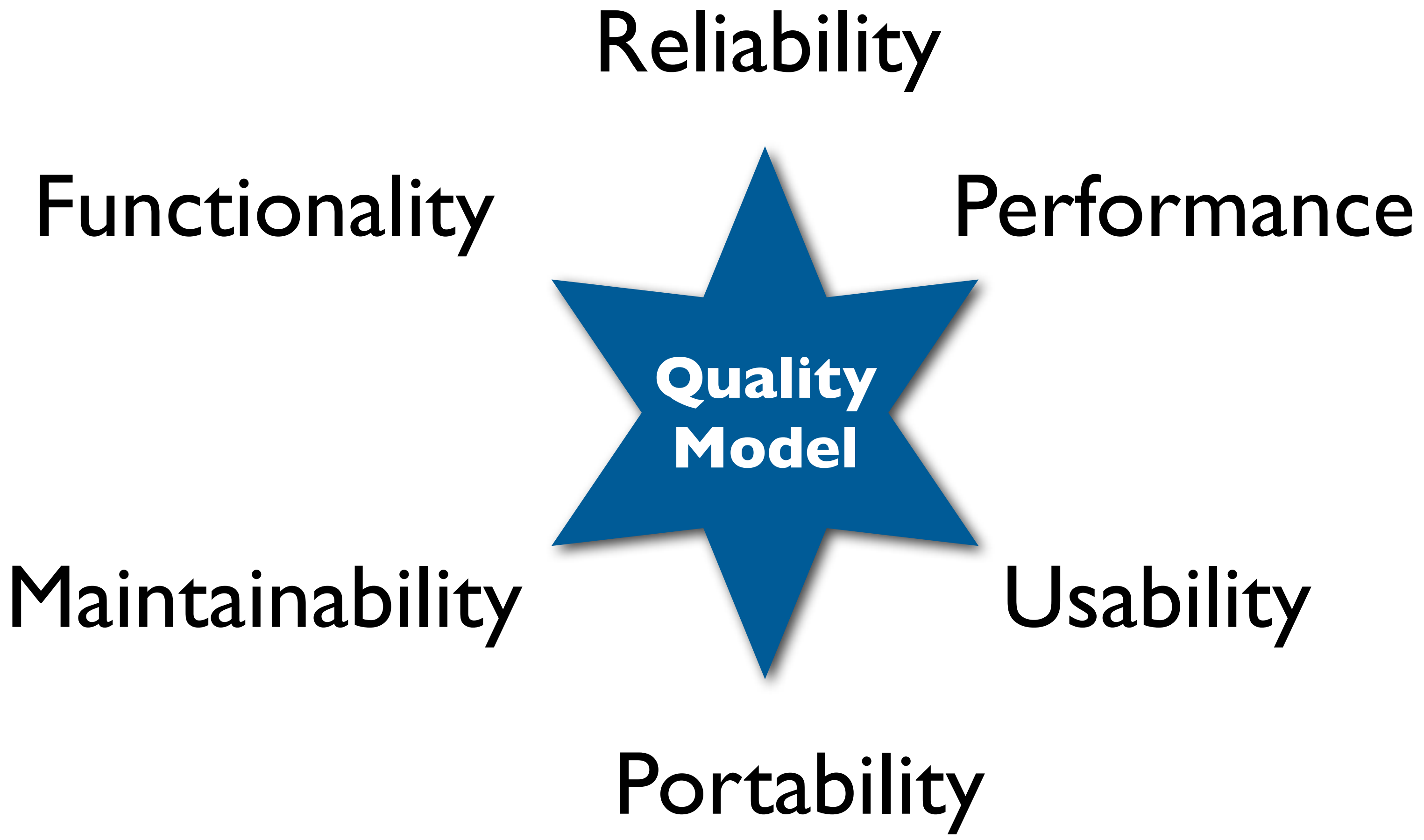
Quality Models



Quality Requirements



Today's lecture will cover quality models and quality requirements.
We start with quality requirements.



Reliability

Functionality

Performance

Quality Model

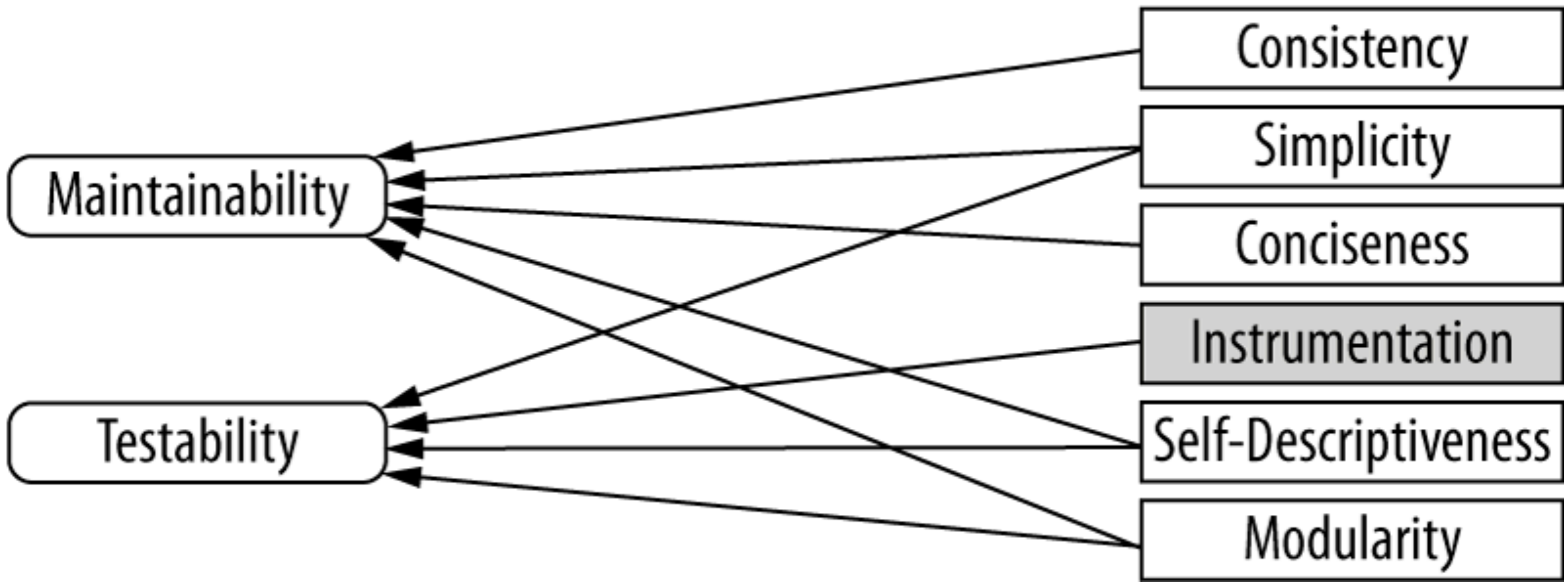
Maintainability

Usability

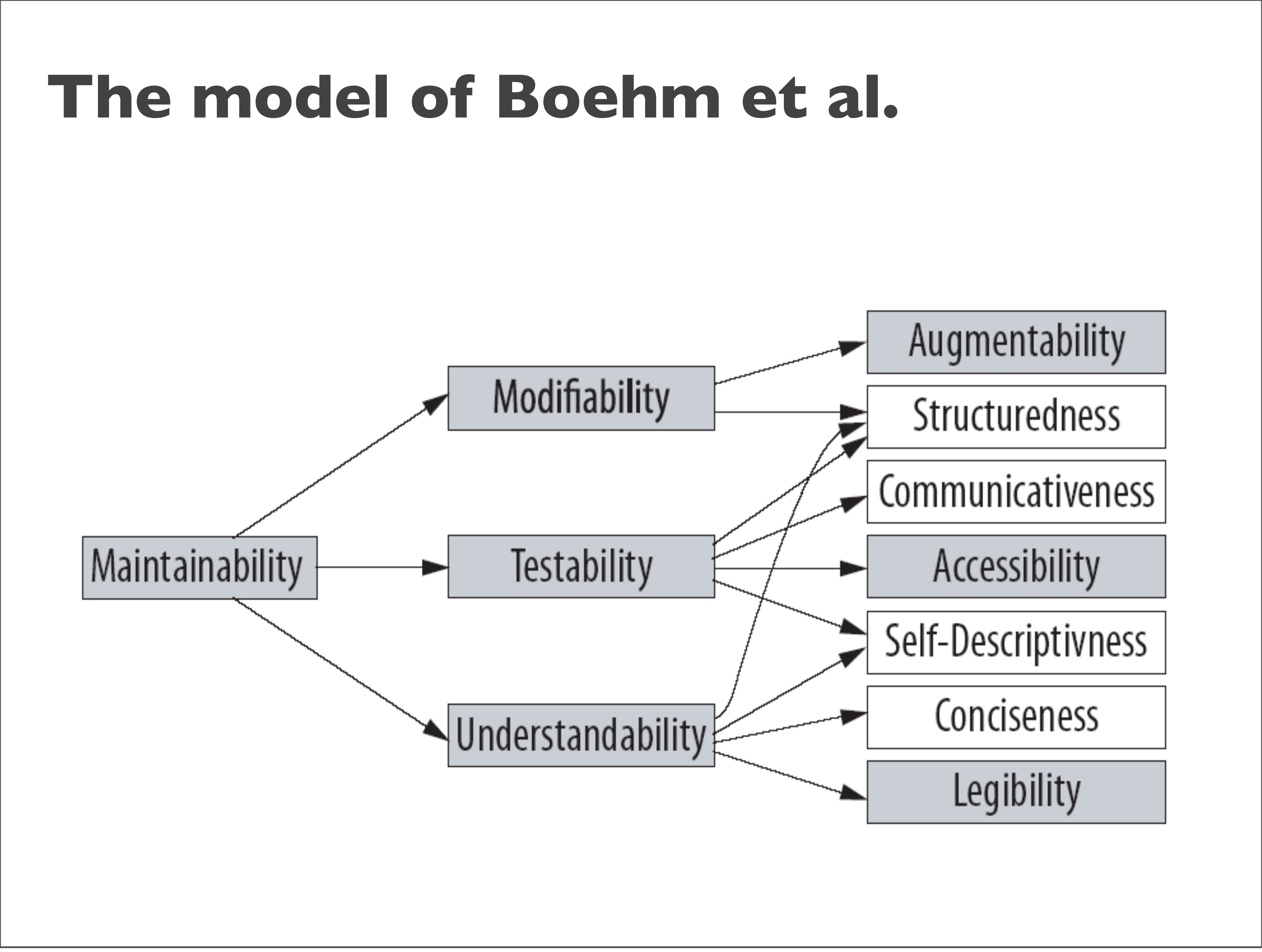
Portability

- We discussed in the 1st lecture what a quality model is.
- Quality model: abstract definition of the important attributes for quality
- Basis for the definition of quality requirements
- Structured quality assessments
- Typically adapted to organisation, project, domain, …
- Standard: ISO 9126

The model of McCall



- McCall proposed one of the first quality models for software.
- He described properties of the software system that have an influence on quality attributes.
- It mixes properties that express a degree (e.g., consistency) and binary properties (e.g., instrumentation).



The model of Boehm et al.

Maintainability

Modifiability

Testability

Understandability

Augmentability

Structuredness

Communicativeness

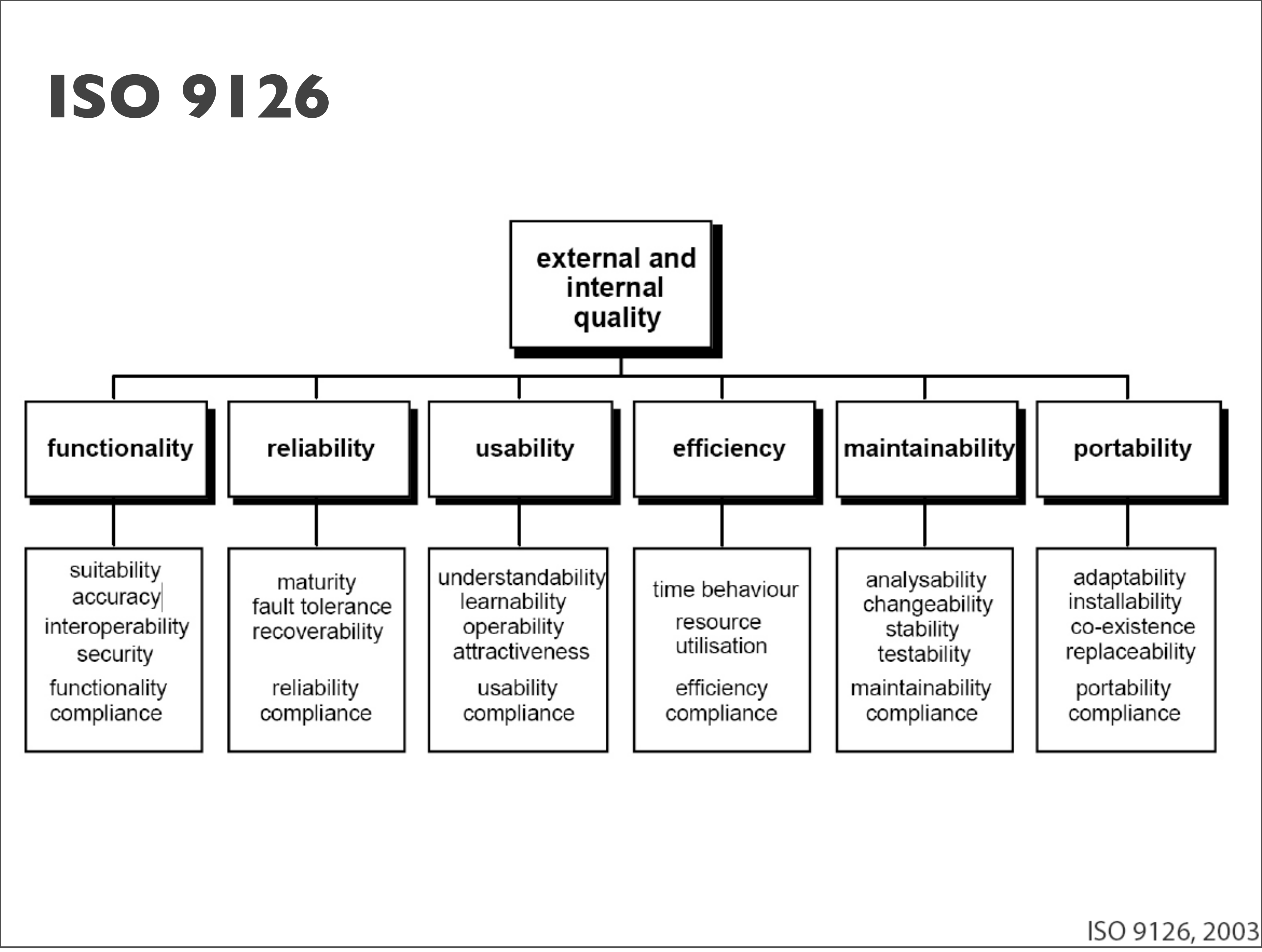
Accessibility

Self-Descriptivness

Conciseness

Legibility

- Boehm et al. proposed a quality model in parallel to McCall.
- Approach
  - Break-down of quality criteria
  - Tree-like structure
- Problem
  - Characteristics vs. activities
  - Ambiguous edge semantic



ISO 9126

external and internal quality

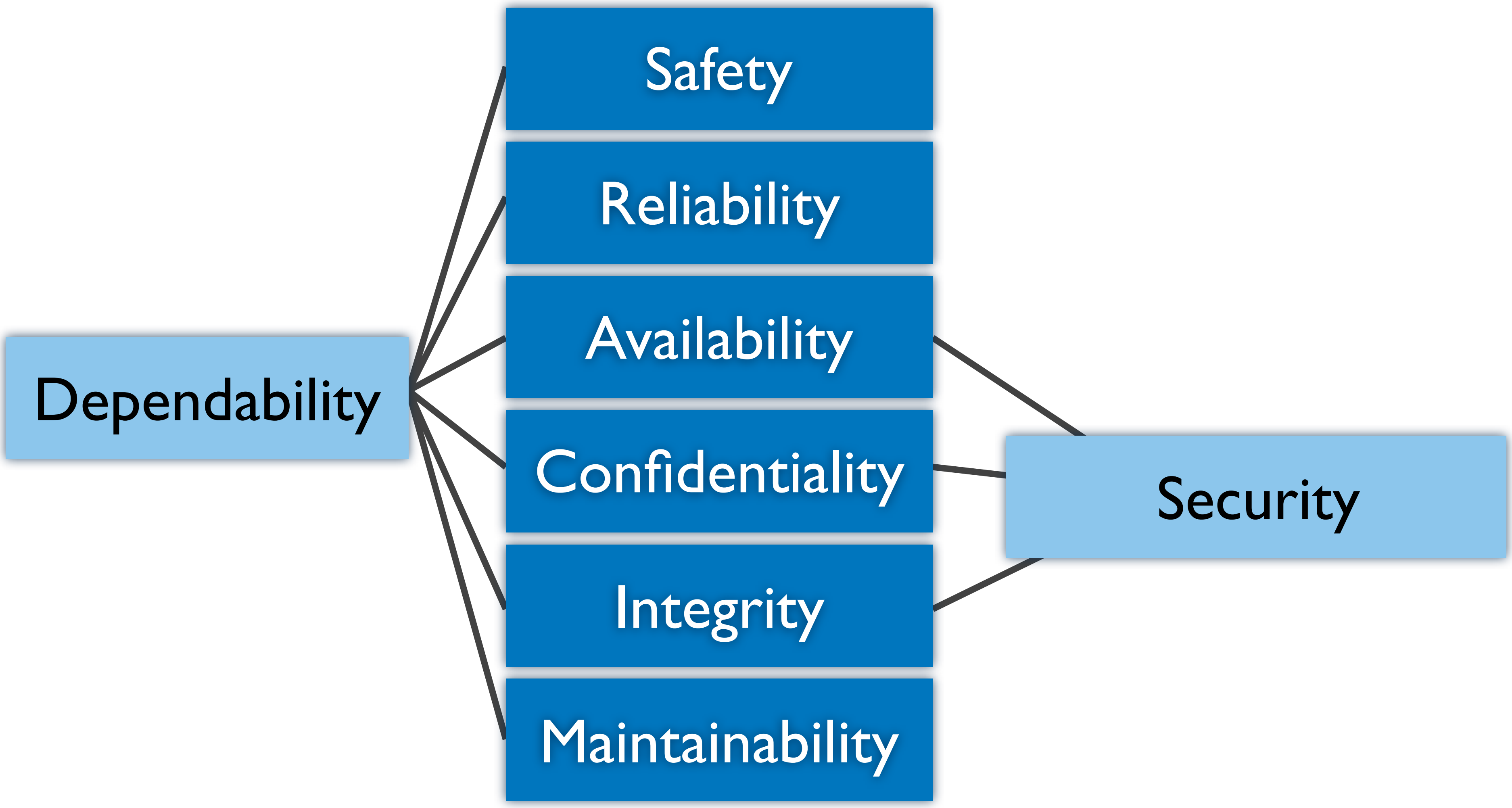
- functionality
  - suitability
  - accuracy
  - interoperability
  - security
  - functionality compliance
- reliability
  - maturity
  - fault tolerance
  - recoverability
  - reliability compliance
- usability
  - understandability
  - learnability
  - operability
  - attractiveness
  - usability compliance
- efficiency
  - time behaviour
  - resource utilisation
  - efficiency compliance
- maintainability
  - analysability
  - changeability
  - stability
  - testability
  - maintainability compliance
- portability
  - adaptability
  - installability
  - co-existence
  - replaceability
  - portability compliance

ISO 9126, 2003

- The works of McCall and Boehm et al. finally resulted in an ISO standard.
- The last version is from 2003.
- Three quality models: quality in use, external quality, internal quality
- Quality in use is decomposed in characteristics such as effectiveness, efficiency, and safety. Quality for the user.
- External and internal quality uses the same characteristics and attributes.
- Differentiation between external and internal quality is not clear.
- Similar problems as with Boehm et al.'s model.
- Gives some measures for the attributes, but not sufficient.

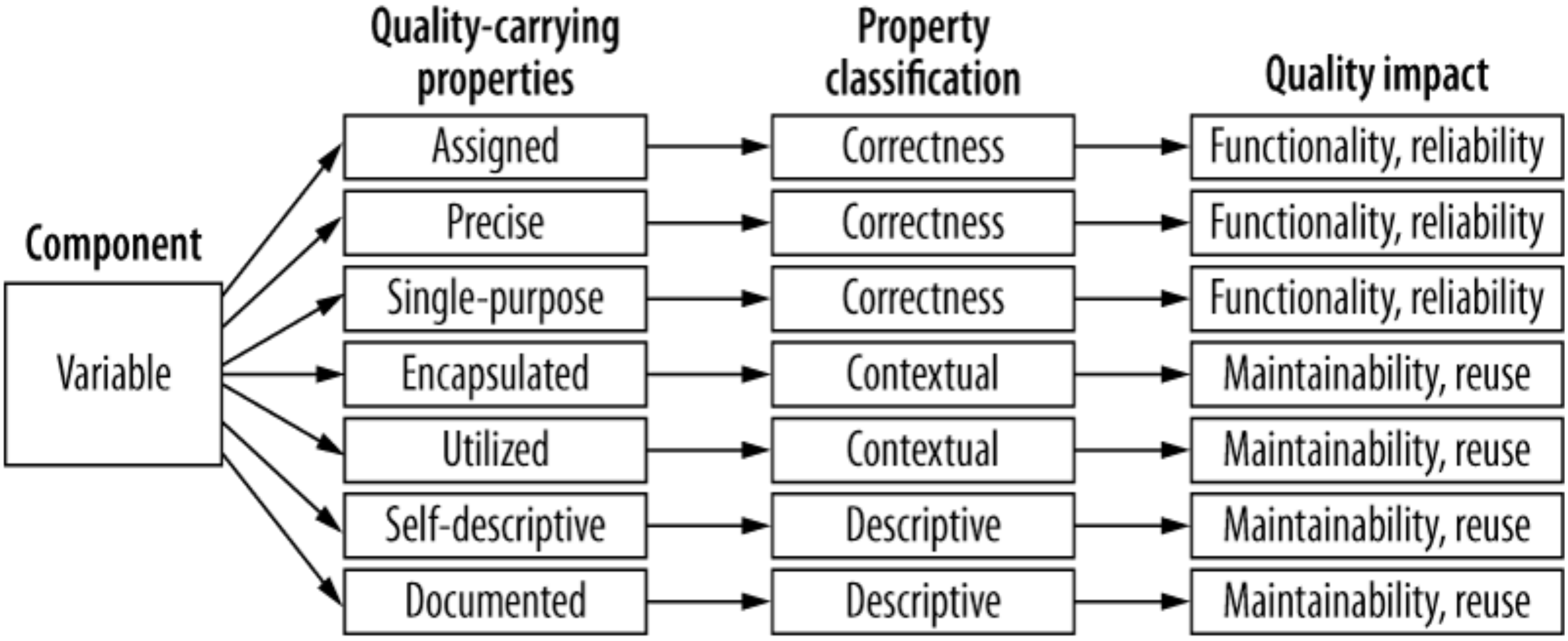
Dependability model

from Avizienis et al.



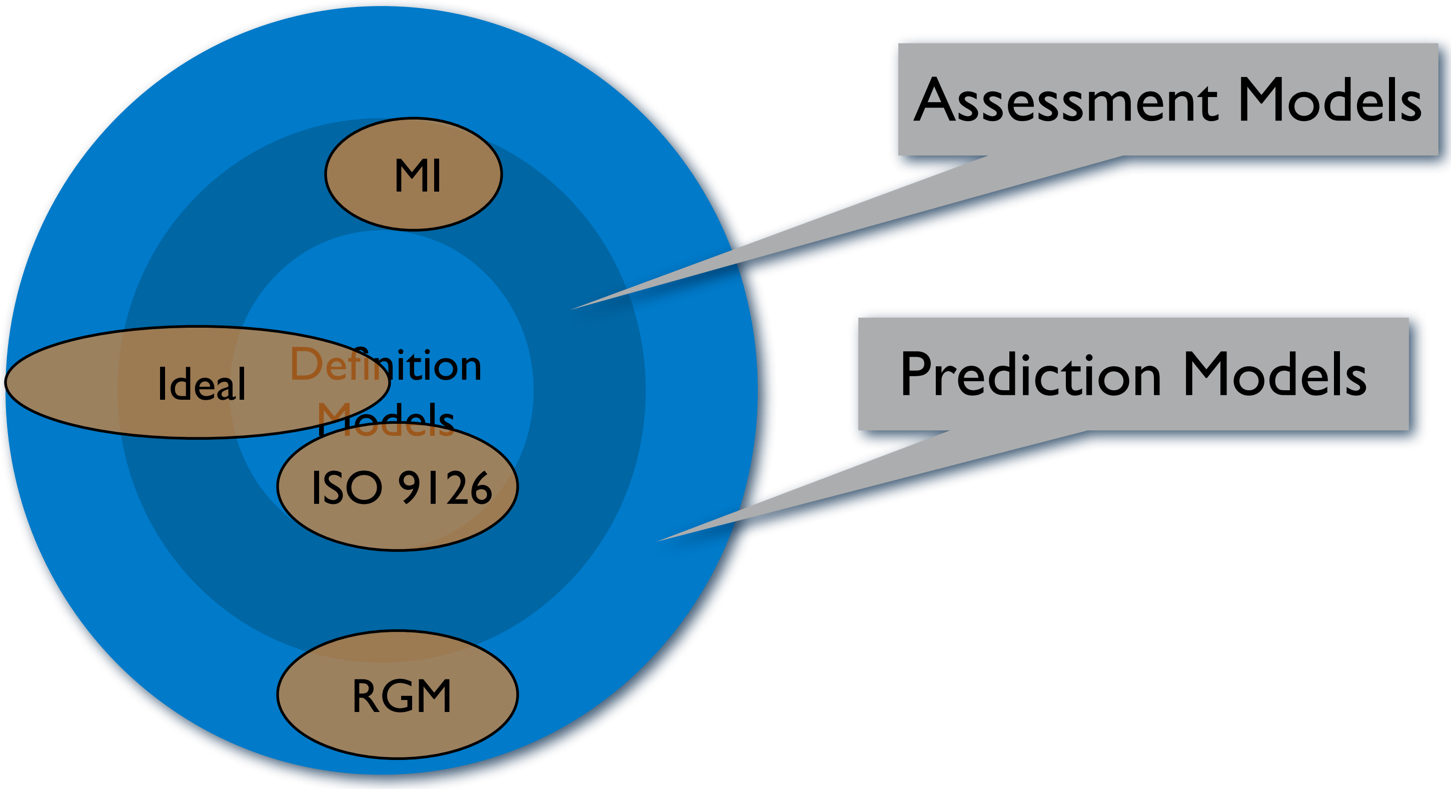
- Dependability is a different view on software quality with an emphasis on the trust one can have that the system will behave as expected, how well the stakeholders can depend on the system.
- The most popular quality model from Avizienis et al. breaks it down similiarly to other quality models.
- Safety, security, reliability, and maintainability are the major quality characteristics.

The model of Dromey



- Dromey added the notion of components to quality models.
- Similarly to McCall, he describes product properties ("quality-carrying properties") that have an impact on a quality attribute ("quality impact").
- He is more specific, however, by specifying explicitly which product entity ("component") is described by the property.
- The properties are also classified ("property classification").

DAP classification
for quality models



Deissenboeck et al., Software quality models: Purposes, usage scenarios, and requirements, 2009

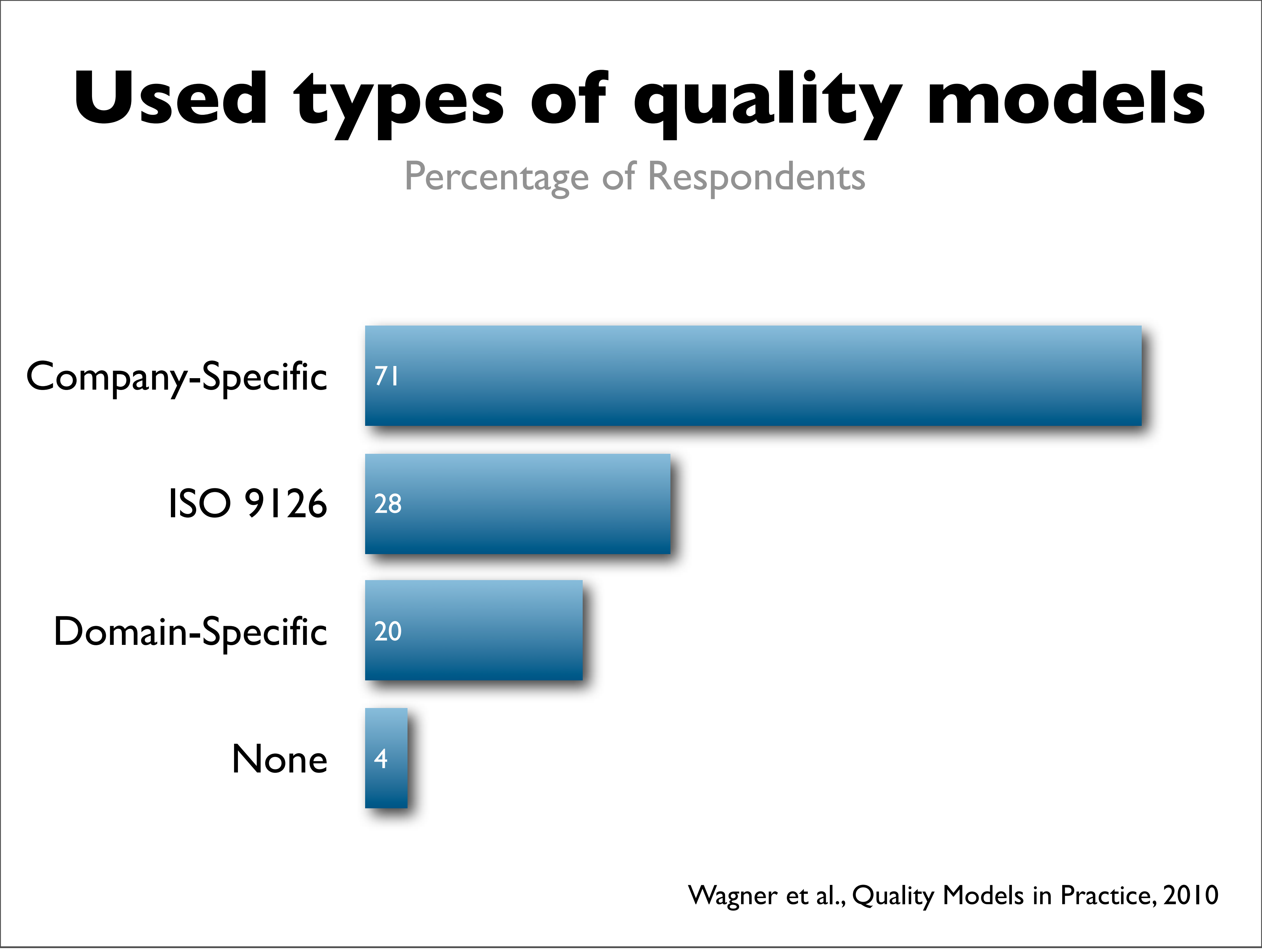
A basic possibility to classify quality models based on their purpose:

- If the model intends only to define quality, it is a definition model.
- If the model is intended to be used in assessing the current quality level, it is an assessment model.
- If the model is intended to predict the future changes in quality, it is a prediction model.

Ideally, a quality model should cover all three classes. It only makes sense to assess quality if I first defined it. Predictions are more complicated and should rely on assessments. Only a definition without support for assessments is not helpful in practice.

MI = Maintainability Index

RGM = Reliability Growth Models



Used types of quality models

Wagner et al., Quality Models in Practice, 2010

More than 70% of the respondents use company-specific quality models.
Only 28% employ the current standard.
Domain-specific standards play a significant role.
Only 4% do not employ any quality model.
Hence, quality models are widely used. Nevertheless, they have problems.

Quality Requirements Reuse

It is important to note that a quality model is mostly a repository for quality requirements.
The requirements need to be specified on a level that allows reuse.
It should also provide means to find – in a structured way – the requirements relevant for a specific project context.

Problems with Quality Models

"The -ilities are good for management talk only."

Wagner et al., Quality models in practice: A preliminary analysis, 2009

- The ISO standard is not considered sufficient by developers.
- Its application in practice requires a huge amount of additional effort.
- Further quotes:
  - „Operationalization (break down to technical attributes) is difficult."
  - „The quality model is not operationalized enough. There is a gap between framework and execution.
- Company-specific models are employed to close this gap.

SAP
Product Standards

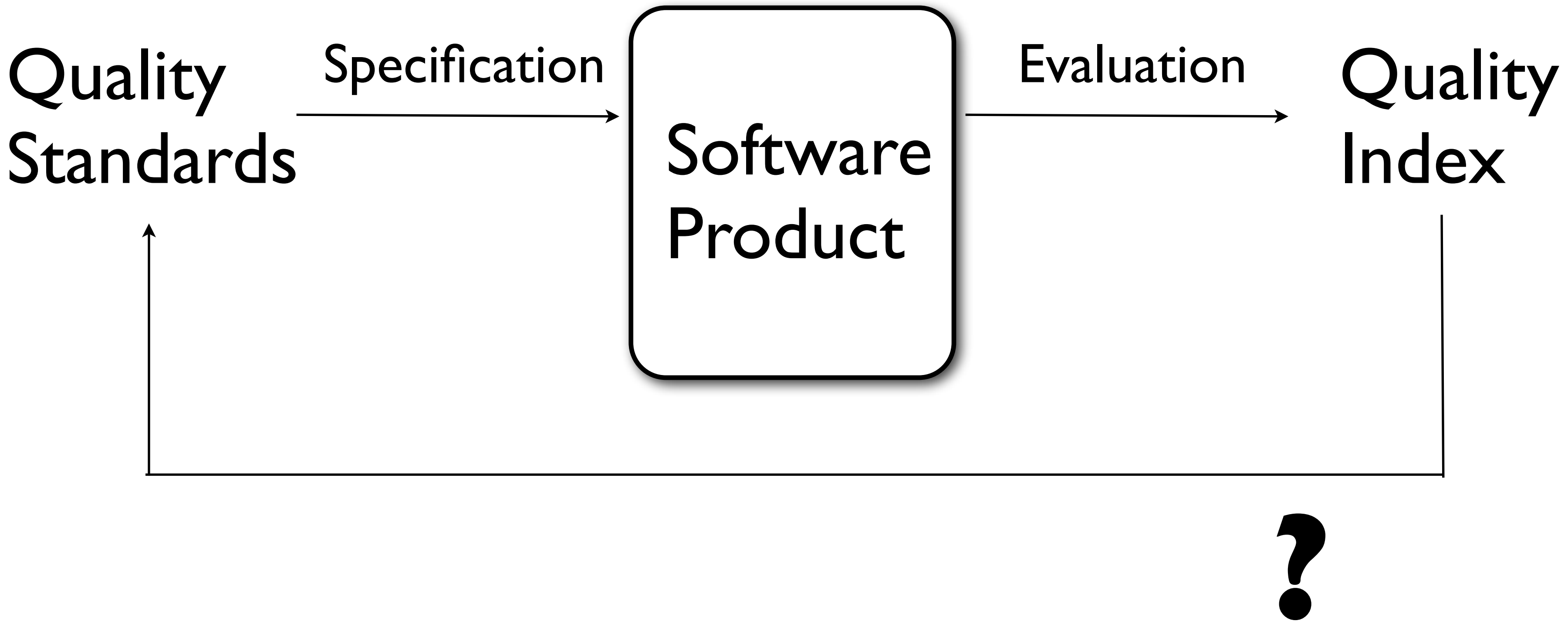
16 x ca. 20 Requirements

Siemens
SPQR Model

several hundred Rules
for C/C++/Java

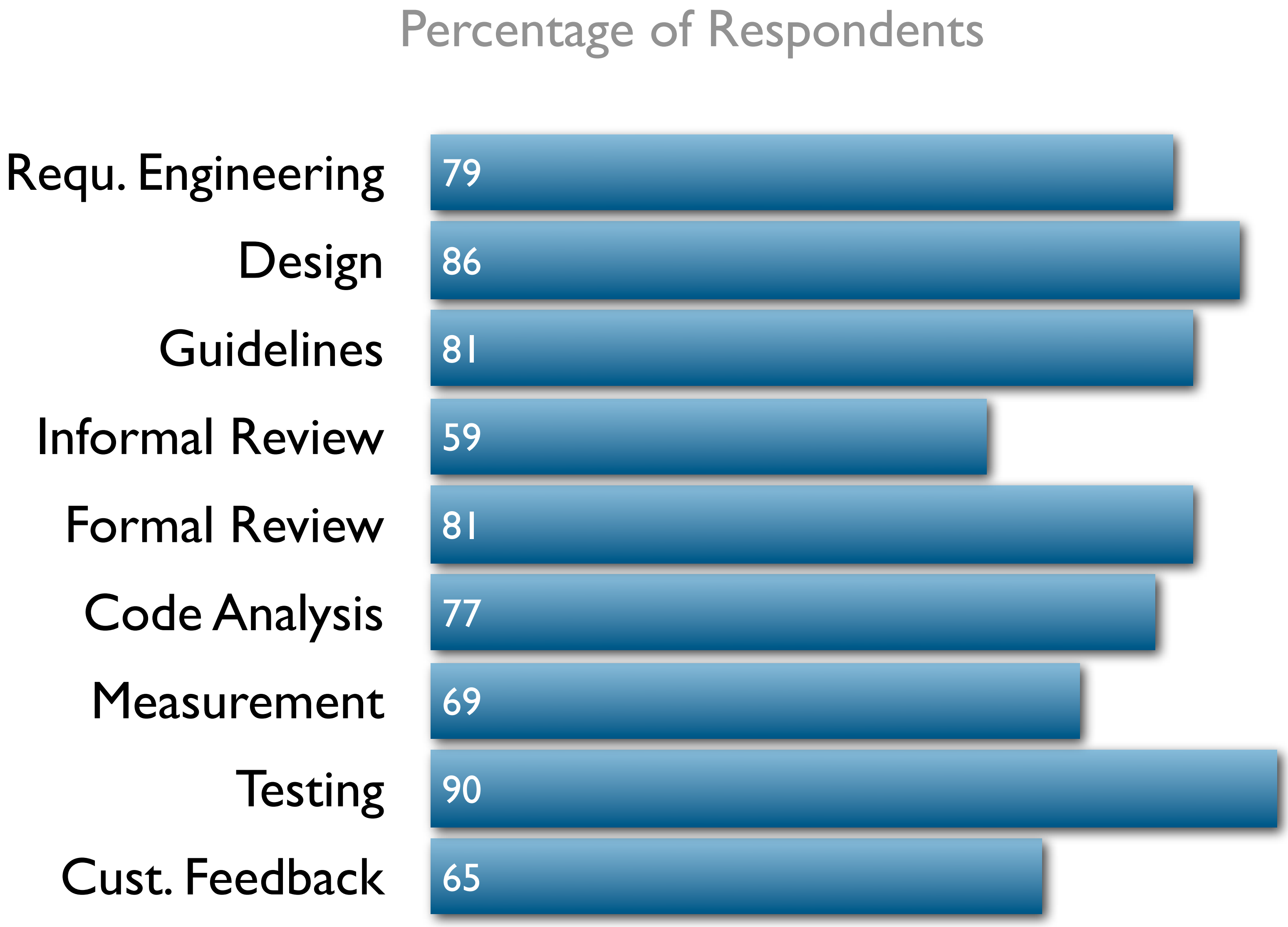
The problem here is that it is company-specific and the company has the complete effort for development and maintenance.
Such models that are applicable in practice become huge.
Furthermore, it lacks a reflection with others. Does the model cover everything relevant?

Separated models



One major problem is that there is rarely one, unified quality model in a company or project.
We often encounter, for example, a requirements model separated from the assessment model.
How and why can developers fullfil the quality index, if it is not part of the requirements?

Where are quality models used?

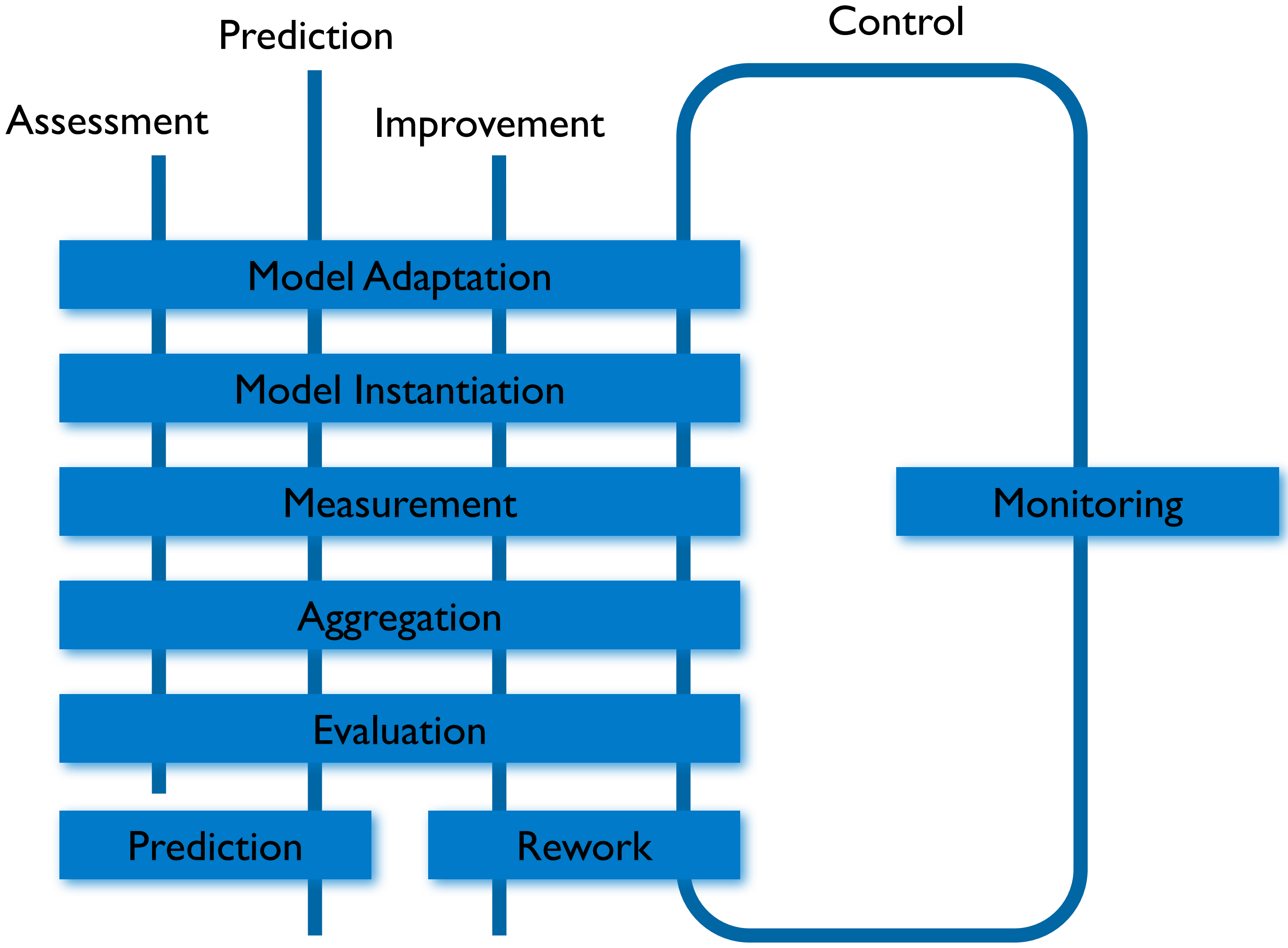


Wagner et al., Quality Models in Practice, 2010

- Quality models play a central role. They are used in almost all phases of software development.
- Models are requirements source: „The requirements mainly come from the company-wide standards.“
- Used for quality assessment: „Quality models play a central role in quality analysis projects. They are used to be able to make a quality statement.“

Group Work (10 Min.):
How are quality models used?
What usage scenarios should be supported by quality models?

Usage scenarios



Deissenboeck et al., Software quality models: Purposes, usage scenarios, and requirements, 2009

The classification in definition, assessment, and prediction is a simplification. The usage scenarios of quality models are more complex.

One way to describe this is with the high-level scenarios **assessment**, **prediction**, **improvement**, and **control**.

The first step is **model adaptation**, in which an existing quality model is changed to fit the product and project (e.g., removed not needed parts such as safety in a non-safety-critical system). It can also include adding additional elements to the model (e.g., because of a new technology that is not yet covered).

The second step is **model instantiation**. This involves mainly to set up the model and corresponding assessment tools for the current product and project. For example, static analysis tools need to be configured, a test environment needs to be set up.

The third step is **measurement**, which consists of performing the needed analyses to get measurement data.

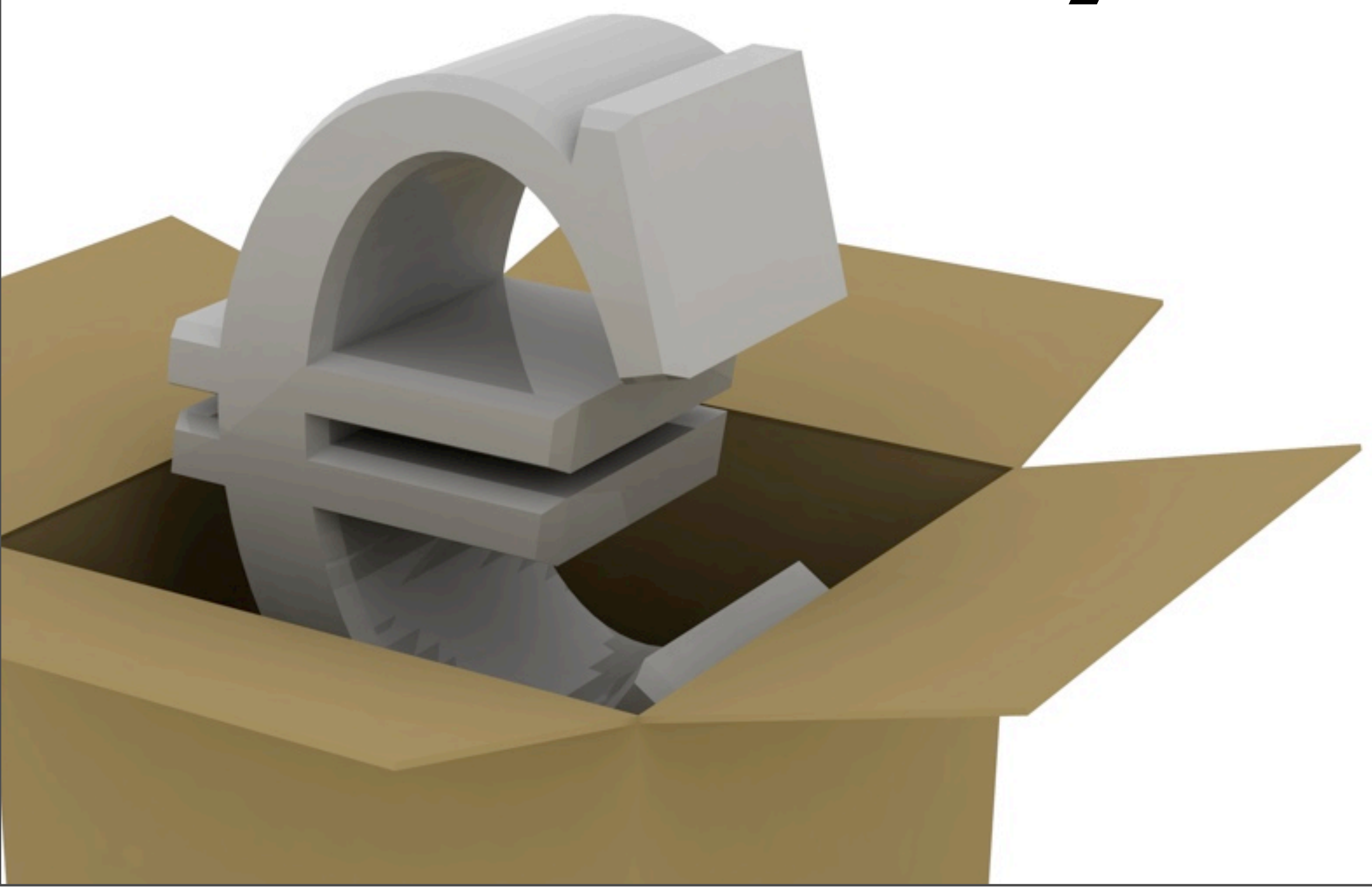
The fourth step, **aggregation**, aggregates these measure to the level we need it for the evaluation.

The fifth step is **evaluation**, which maps the aggregated measures to an evaluation scale (e.g., good/bad, high/medium/low).

Prediction has the additional step **prediction**, which projects the quality level in the future.

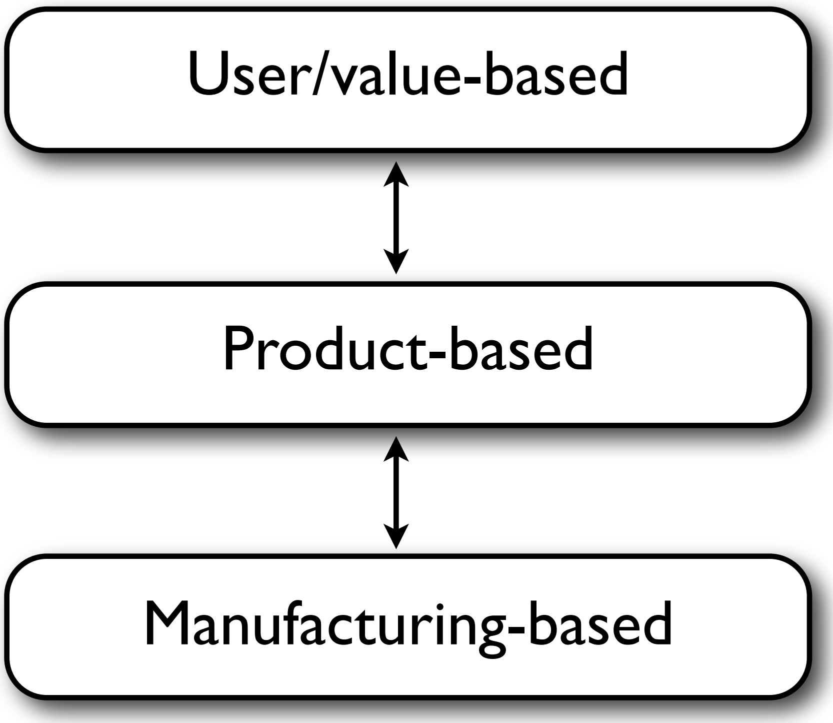
Improvement and control involve **rework**, in which the system is changed to remove quality defects.

Value-Driven
Quality Models



A way to confront problems with quality models are value-driven quality models.
Value-driven refers here that Garvin's value approach drives the structure and usage of the qualty model
They do not resolve all problems but give a modelling philosophy that helps in many cases.

Garvin's quality approaches

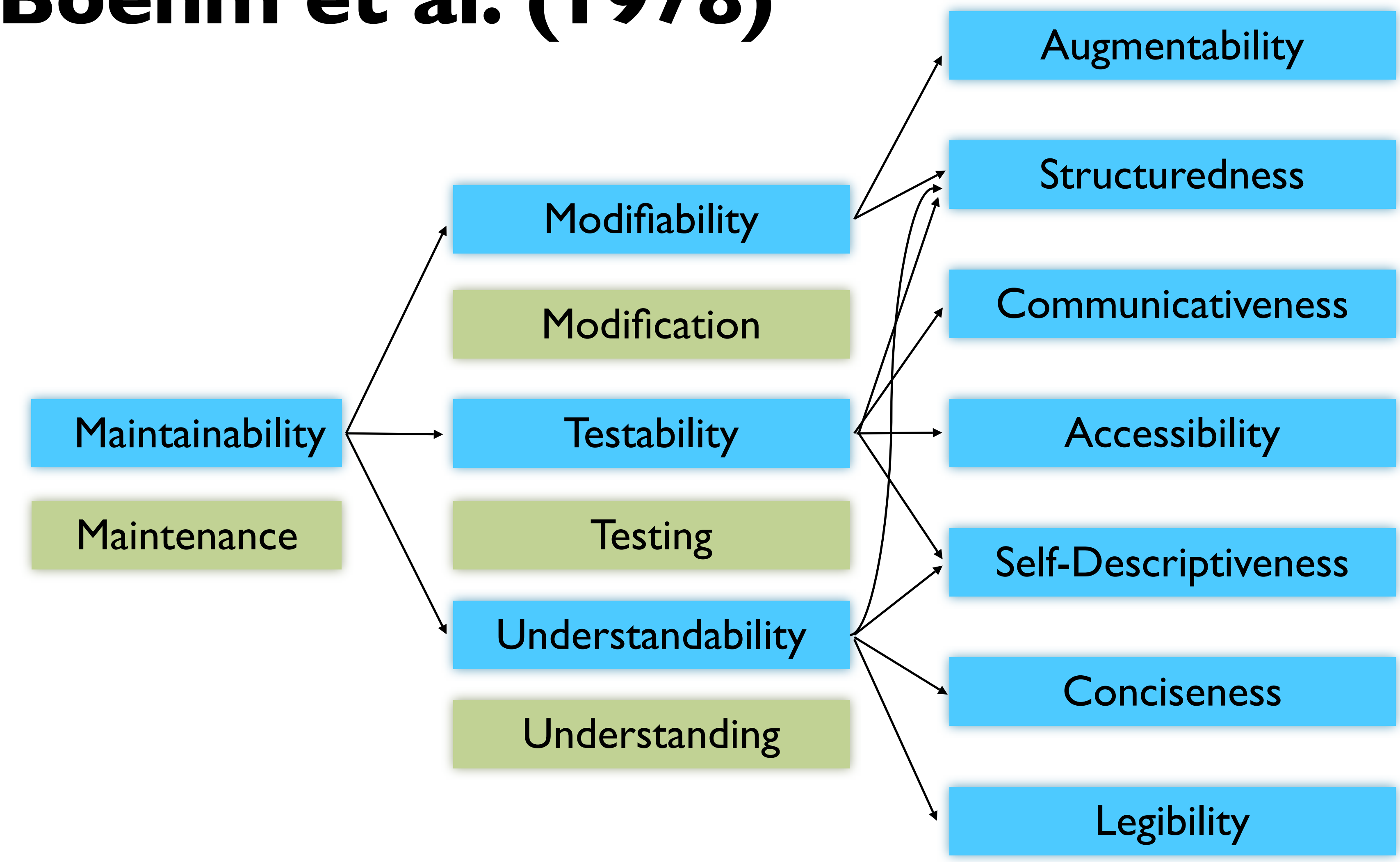


Garvin, What does product quality really mean?, 1984

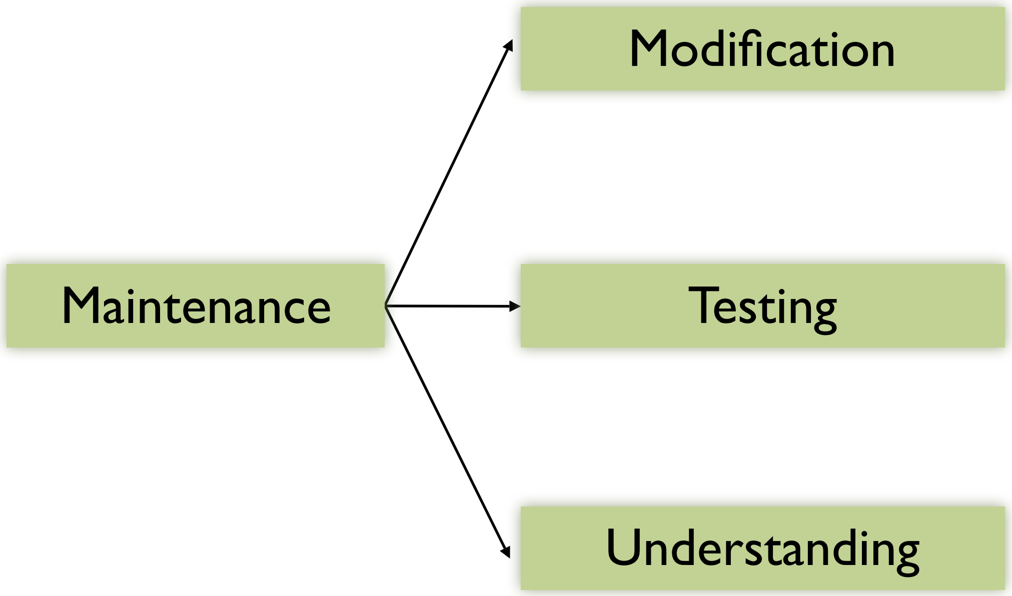
Value-driven quality models cover mostly the user/value approach and product approach. The value for the user is the central means for deriving product attributes that are considered for quality.

Manufacturing/process can also be covered but is not focus.

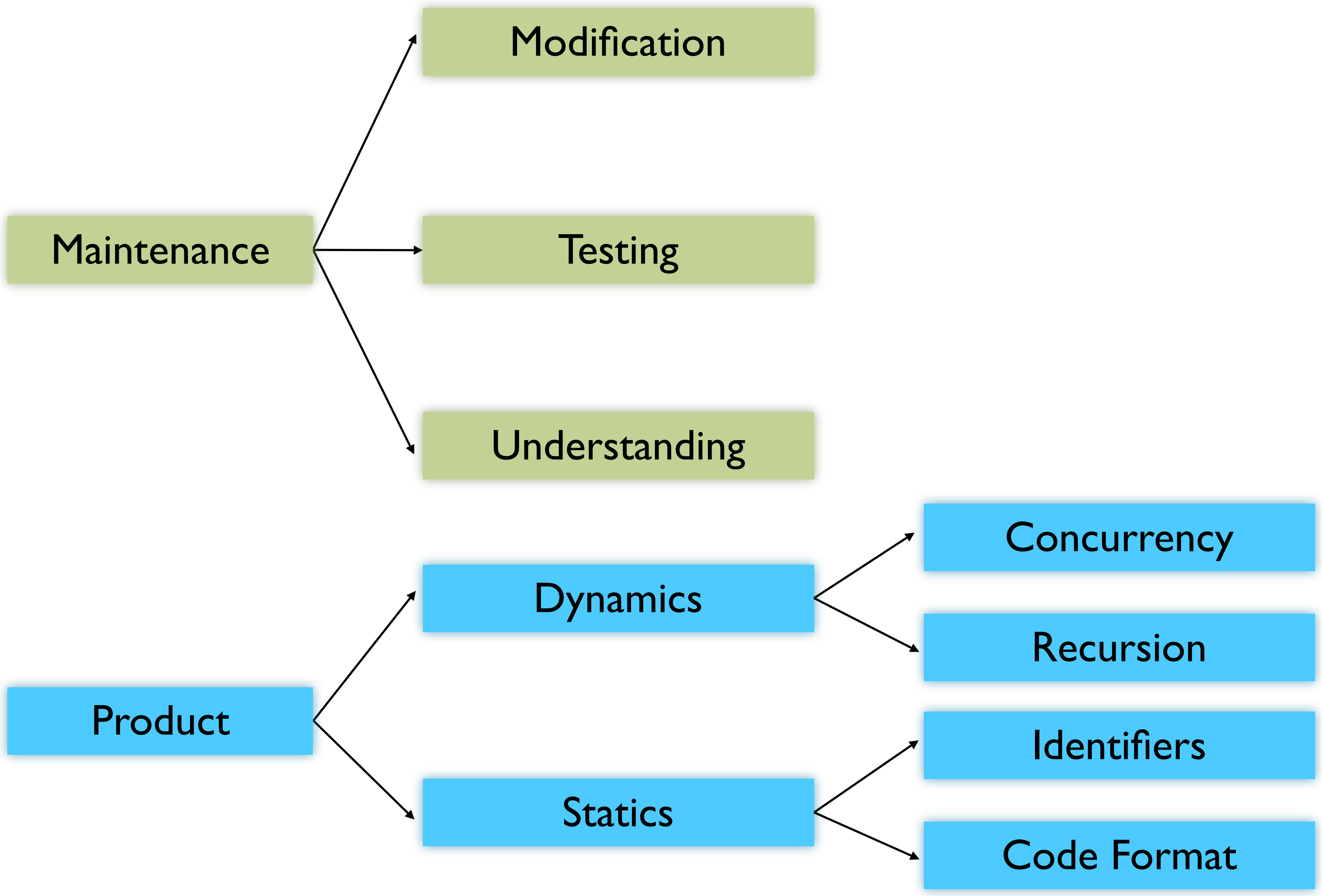
Boehm et al. (1978)



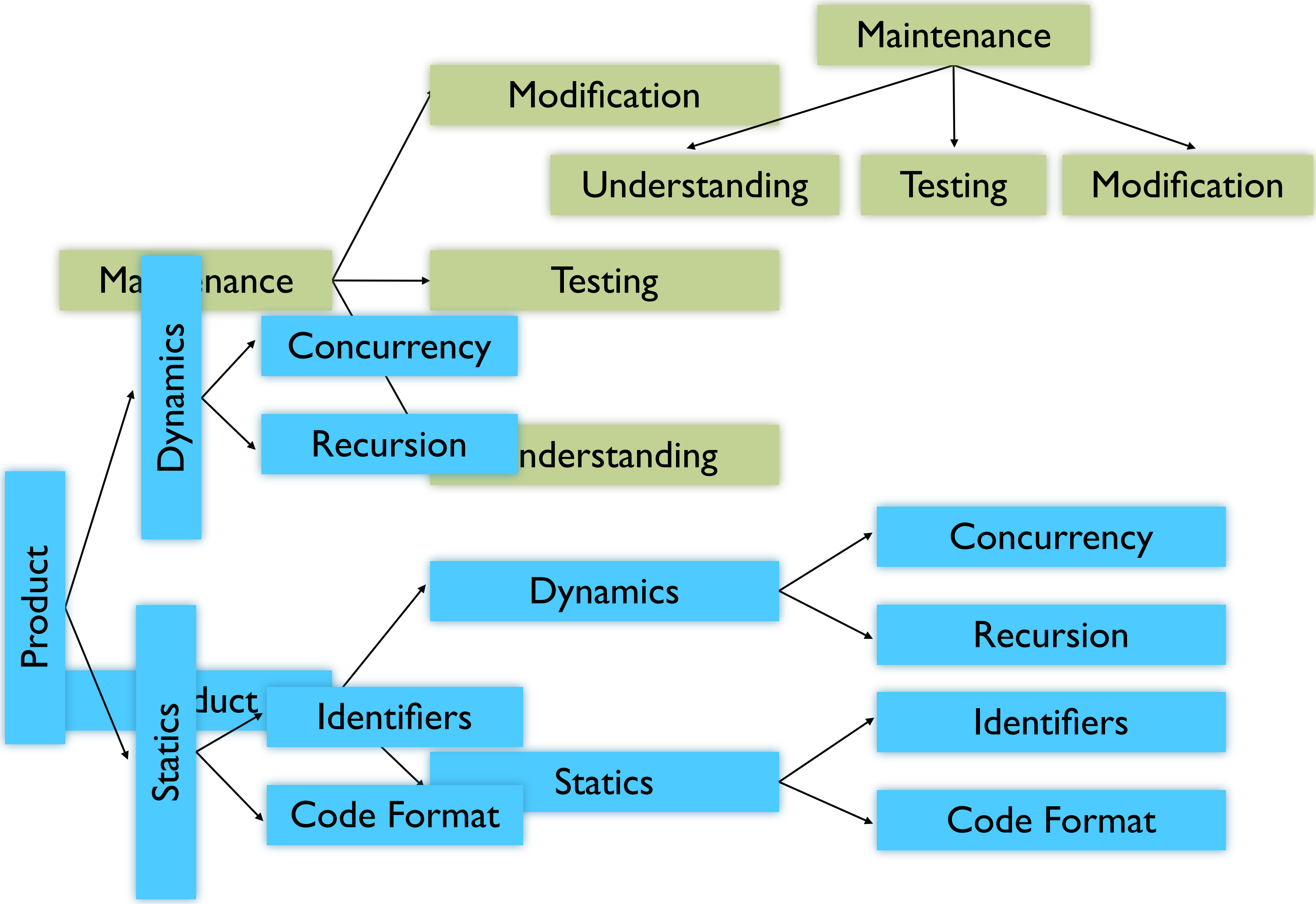
Current quality models mix activities with product properties.



These activities drive the value for the stakeholders.
They can be extracted and described separately.



The product properties are described in addition.



Maintenance

Modification

Understanding

Testing

Modification

Ma nance

Dynamics

Testing

Concurrency

Recursion

nderstanding

Product

Concurrency

Dynamics

Recursion

Statics

duct

Identifiers

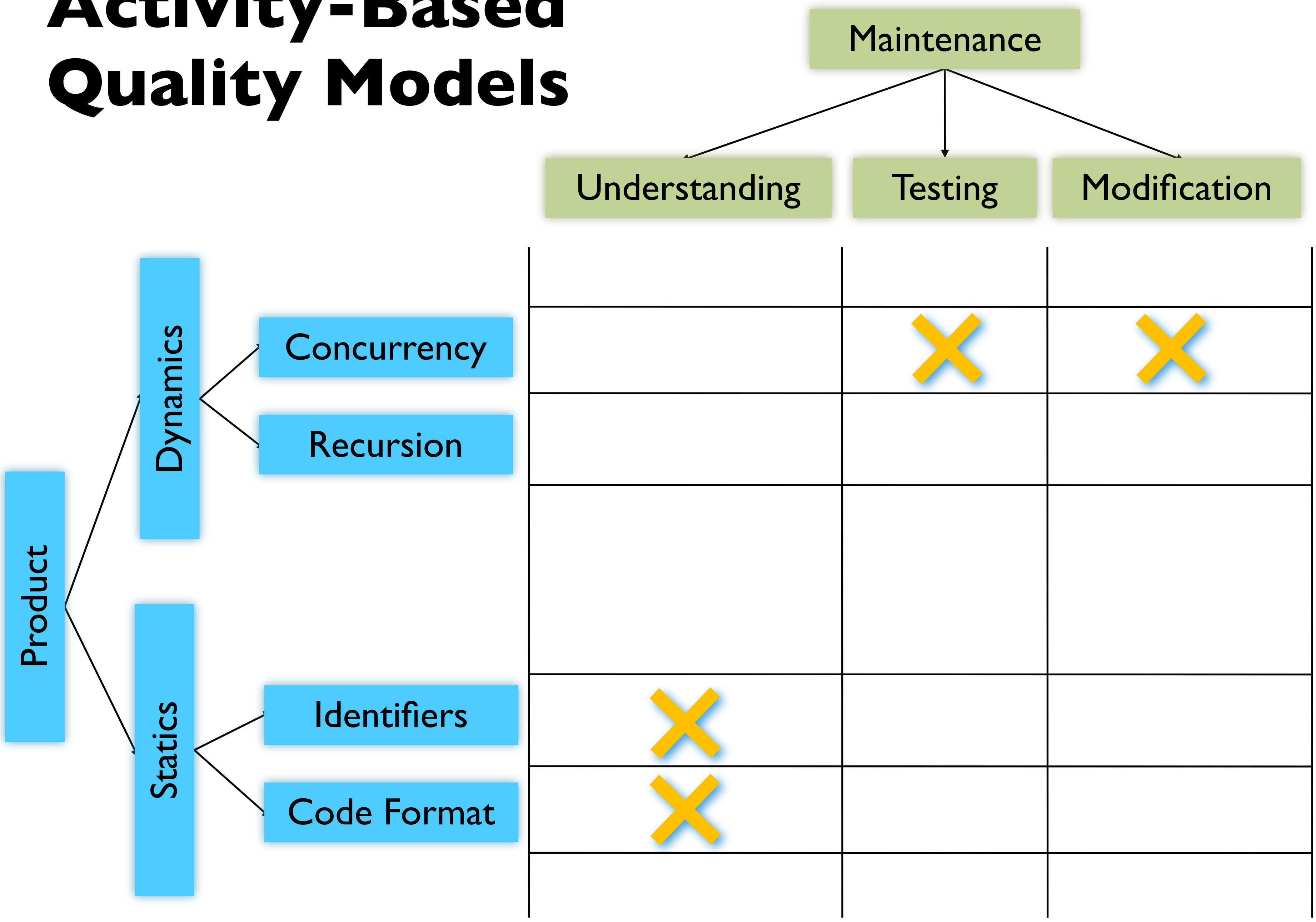
Identifiers

Statics

Code Format

Code Format

Activity-Based Quality Models



	Understanding	Testing	Modification
Concurrency		✕	✕
Recursion			
Identifiers	✕		
Code Format	✕		

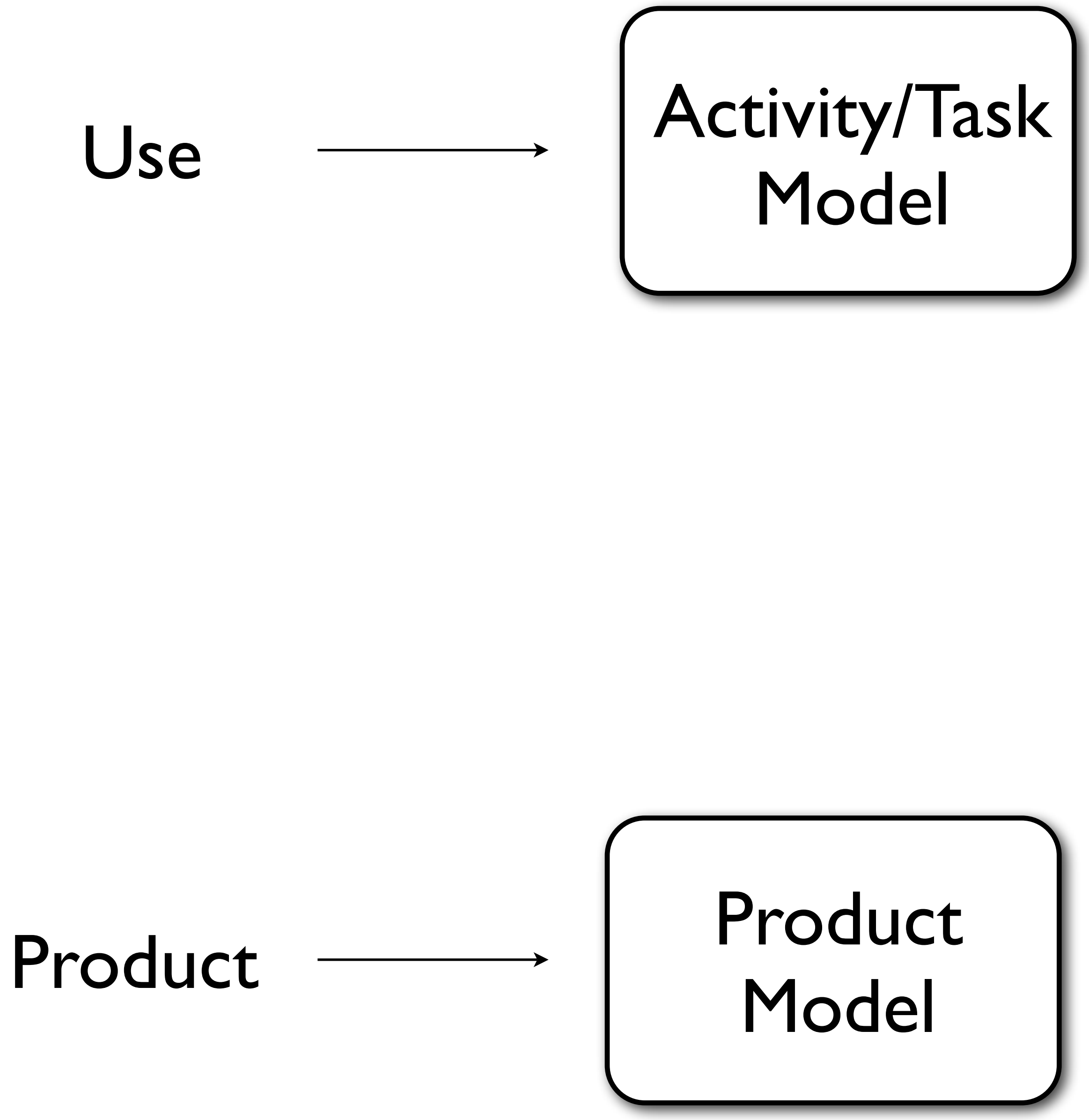
Deissenboeck et al., An activity-based quality model for maintainability, 2007

These two hierarchies (activities and product properties) can then be set into relation.

This is similar to Dromey's model, but he described an impact on quality attributes instead of activities.

Activity–based quality model is a specific instance of a value–driven quality model.

Explicit **Entity Models**



Use → Activity/Task Model

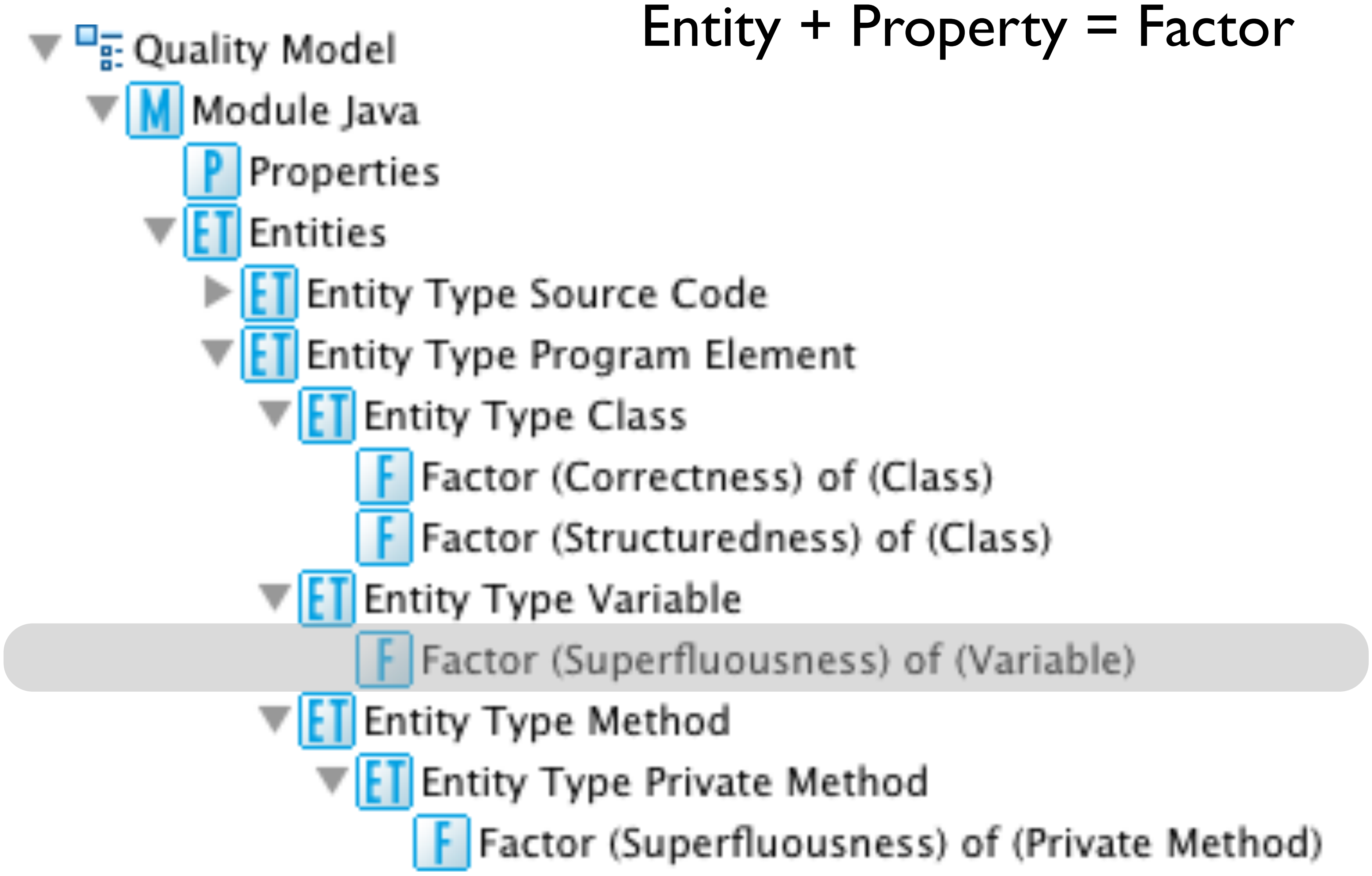
Product → Product Model

Explicit entity models help to clearly describe what is defined and assessed w.r.t. quality. The most important entities are the usage activities/tasks (Use) and the product itself (Product).

Helps in structuring the quality requirements.

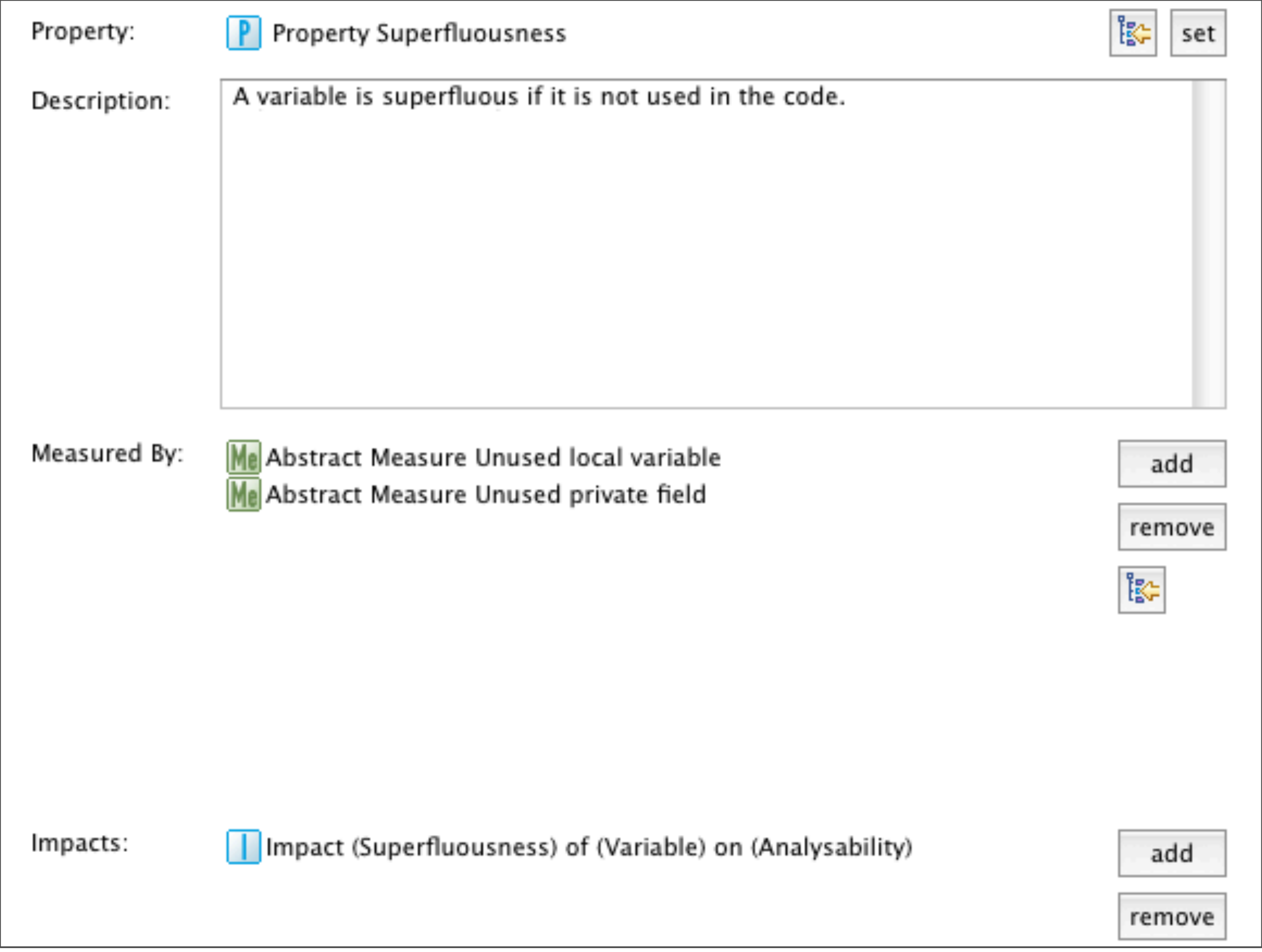
Does also help in the evaluation: what is the concrete thing that I evaluate?

Entity + Property = Factor



- Quality Model
  - Module Java
    - Properties
    - Entities
      - Entity Type Source Code
      - Entity Type Program Element
        - Entity Type Class
          - Factor (Correctness) of (Class)
          - Factor (Structuredness) of (Class)
        - Entity Type Variable
          - Factor (Superfluousness) of (Variable)
        - Entity Type Method
          - Entity Type Private Method
            - Factor (Superfluousness) of (Private Method)

Example how an excerpt of such a quality model could look like
There are entities from the product such as the source code, variable, method, and private method.
These entities are described by properties such as correctness, structuredness, or superfluousness.
The combination of an entity with a property is then called a factor.



Property: P Property Superfluousness

Description: A variable is superfluous if it is not used in the code.

Measured By:
Me Abstract Measure Unused local variable
Me Abstract Measure Unused private field

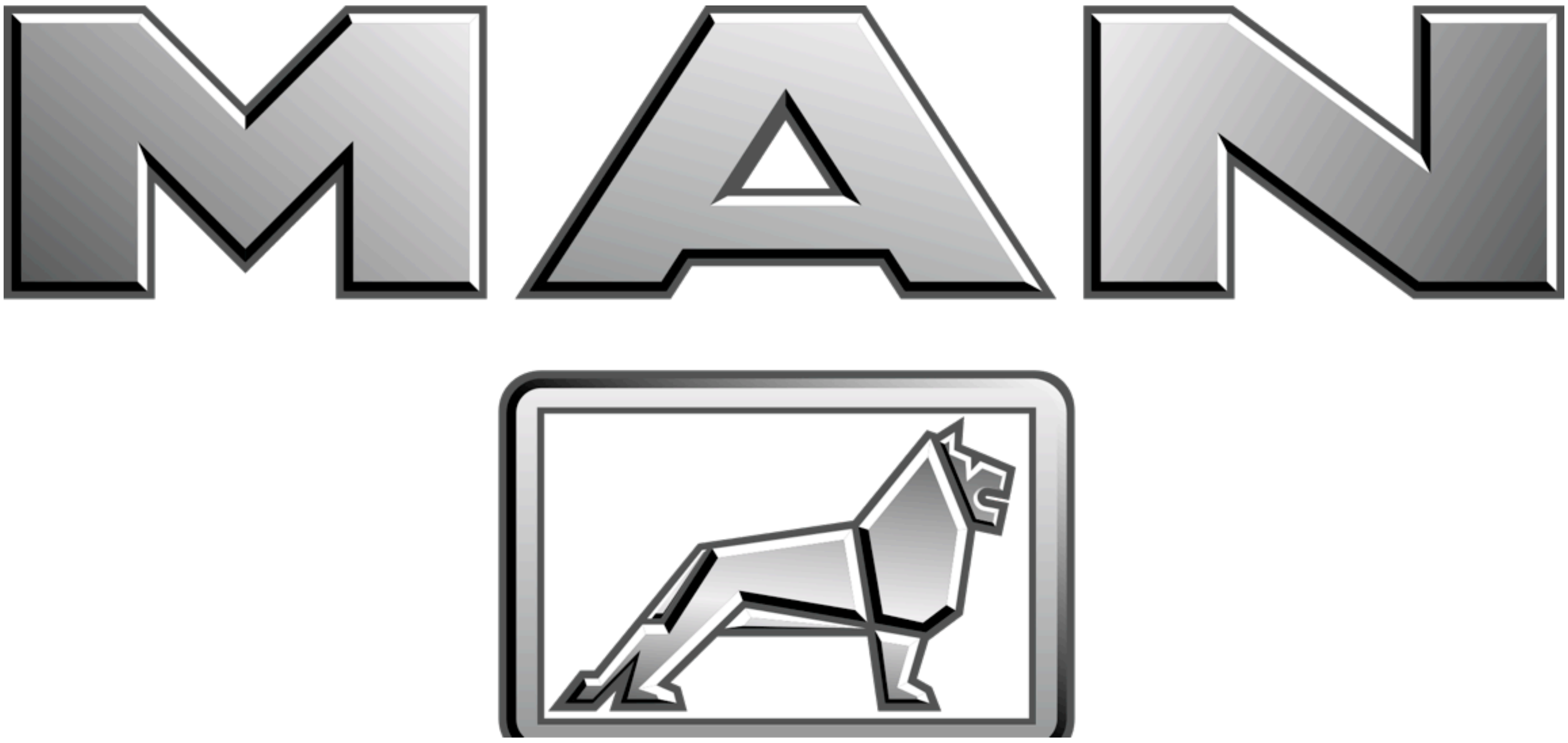
Impacts: I Impact (Superfluousness) of (Variable) on (Analysability)

This shows more details on the factor Superfluousness of Variable.
It specifies its property (superfluousness) and a textual description.
It gives two measures that are indicators for the factor (Unused local variable and Unused private field).
Finally, it shows the impact of the factor on a quality characteristic (or quality aspect). In this case the impact is on analysability.

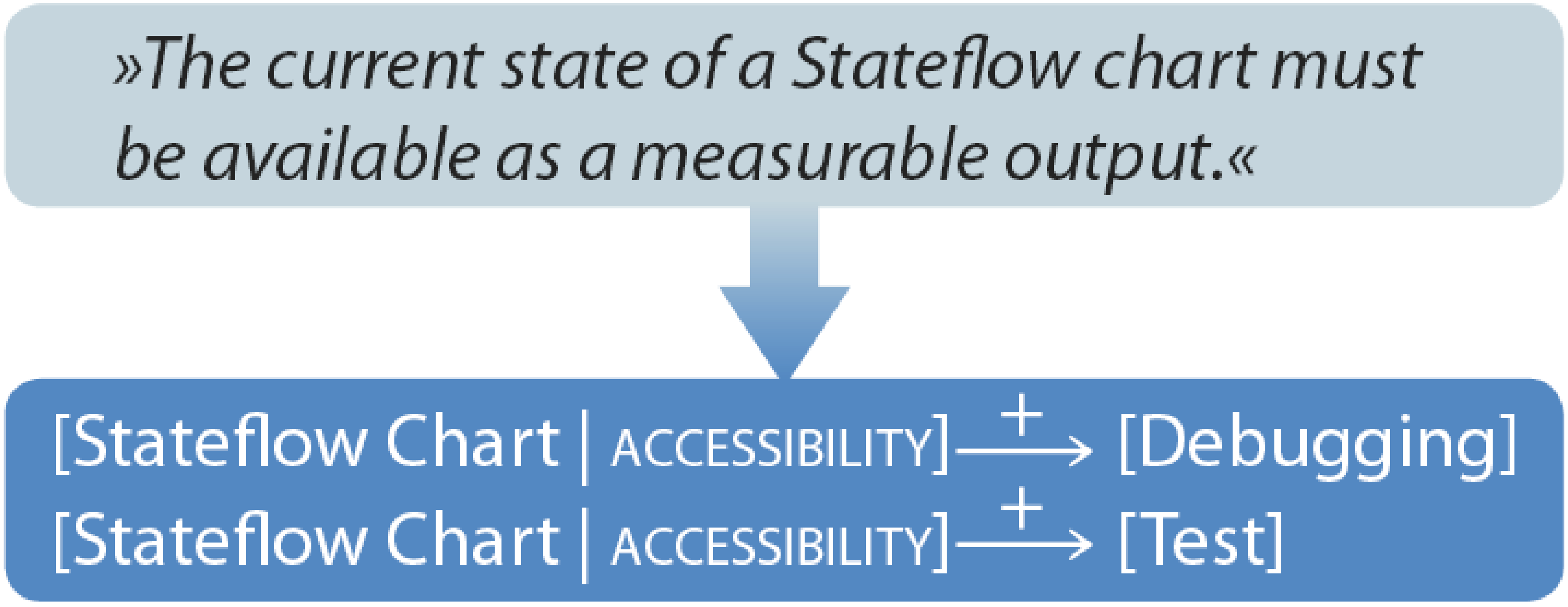
Tools and Checklists

To verify the claims made in the factors (e.g., that the source code is structured), it is essential to operationalise them.
This means that concrete measures and measurement methods are defined.
For everything that can be automated, static and dynamic analysis tools should be used.
There are also tools for aggregating single quality evaluations to a general quality statement.
Everything that cannot be automated should be generated into role- and situation-specific checklists for reviews and inspections.

Example



Large scale case study with MAN Nutzfahrzeuge, which build trucks and buses.
Software development of key components using Matlab Simulink/Stateflow and TargetLink models.
Code generation



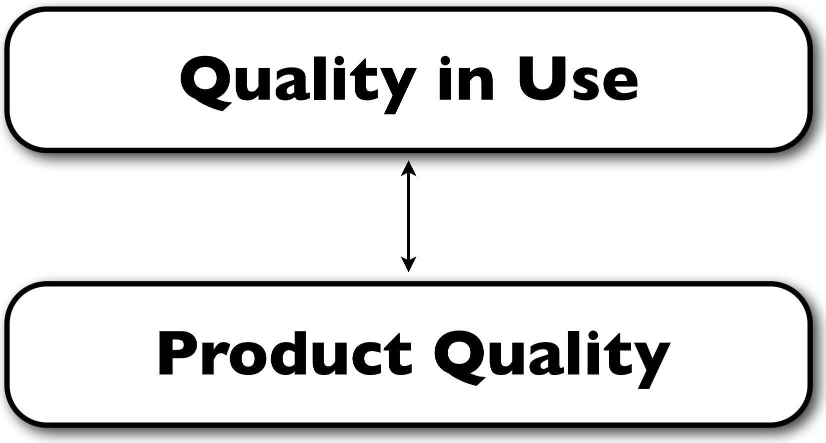
»The current state of a Stateflow chart must be available as a measurable output.«

$$[\text{Stateflow Chart} \mid \text{ACCESSIBILITY}] \xrightarrow{+} [\text{Debugging}]$$

$$[\text{Stateflow Chart} \mid \text{ACCESSIBILITY}] \xrightarrow{+} [\text{Test}]$$

Objective: Quality model for Simulink/Stateflow development at MAN
Sources
* MathWorks documentation
* MAN-internal guideline
* MAAB guideline (MathWorks Automotive Advisory Board)
* dSpace guideline
* Expert know-how
* Studies on model-based development in general

ISO 25010

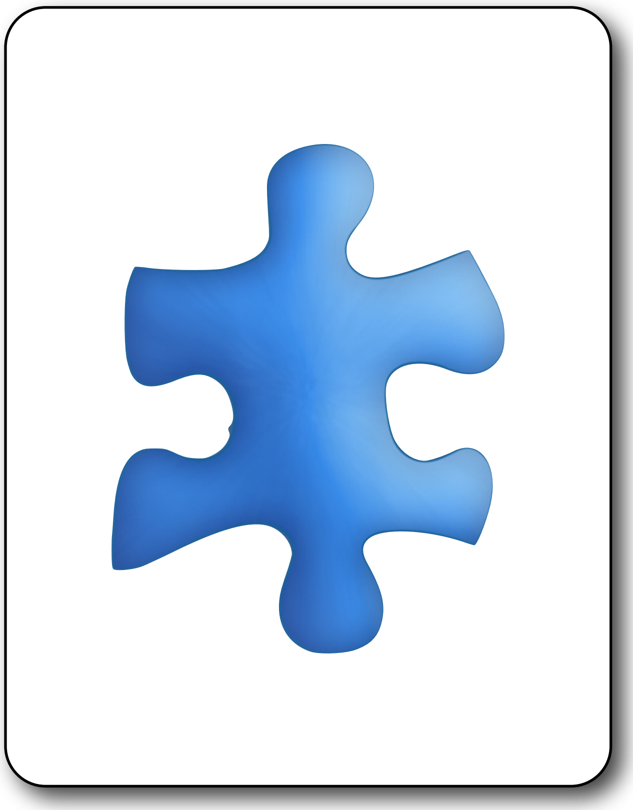

Quality in Use

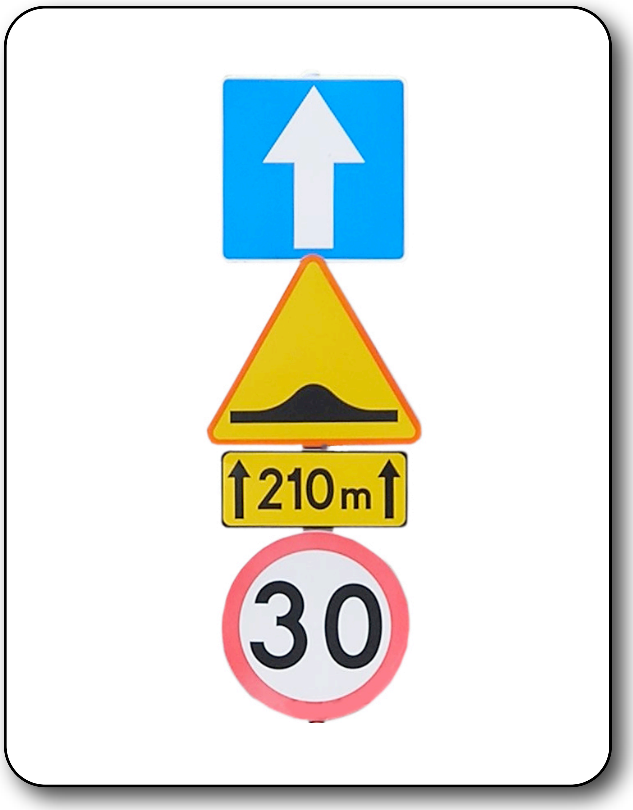
Product Quality

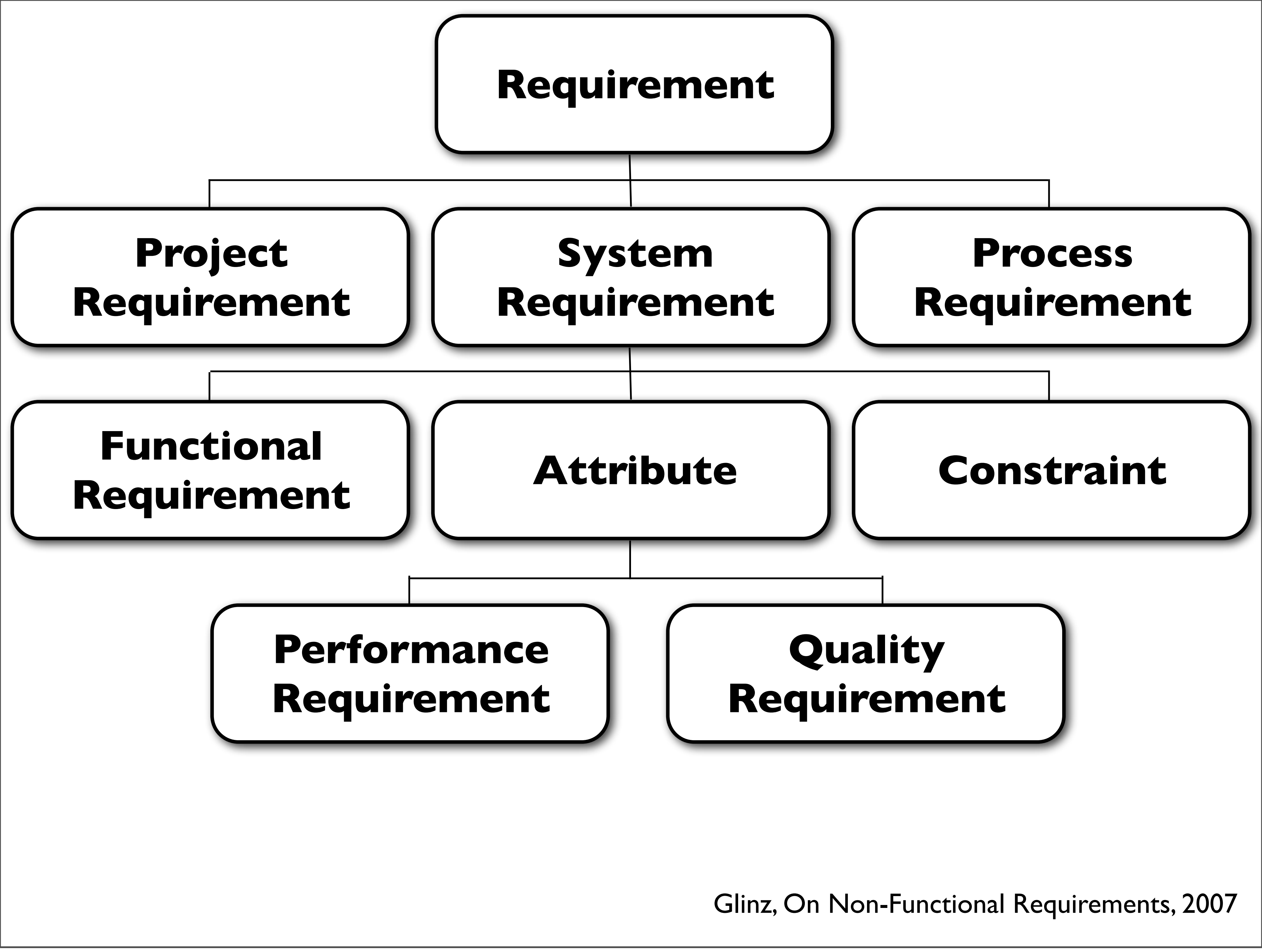
The structure with activities and product properties could also described as:
Quality in Use and Product Quality
Interestingly, this is the current proposal in the ISO 25010. This, however, has not been finalised yet.

Quality Models

Quality Requirements







Requirement

Project Requirement

System Requirement

Process Requirement

Functional Requirement

Attribute

Constraint

Performance Requirement

Quality Requirement

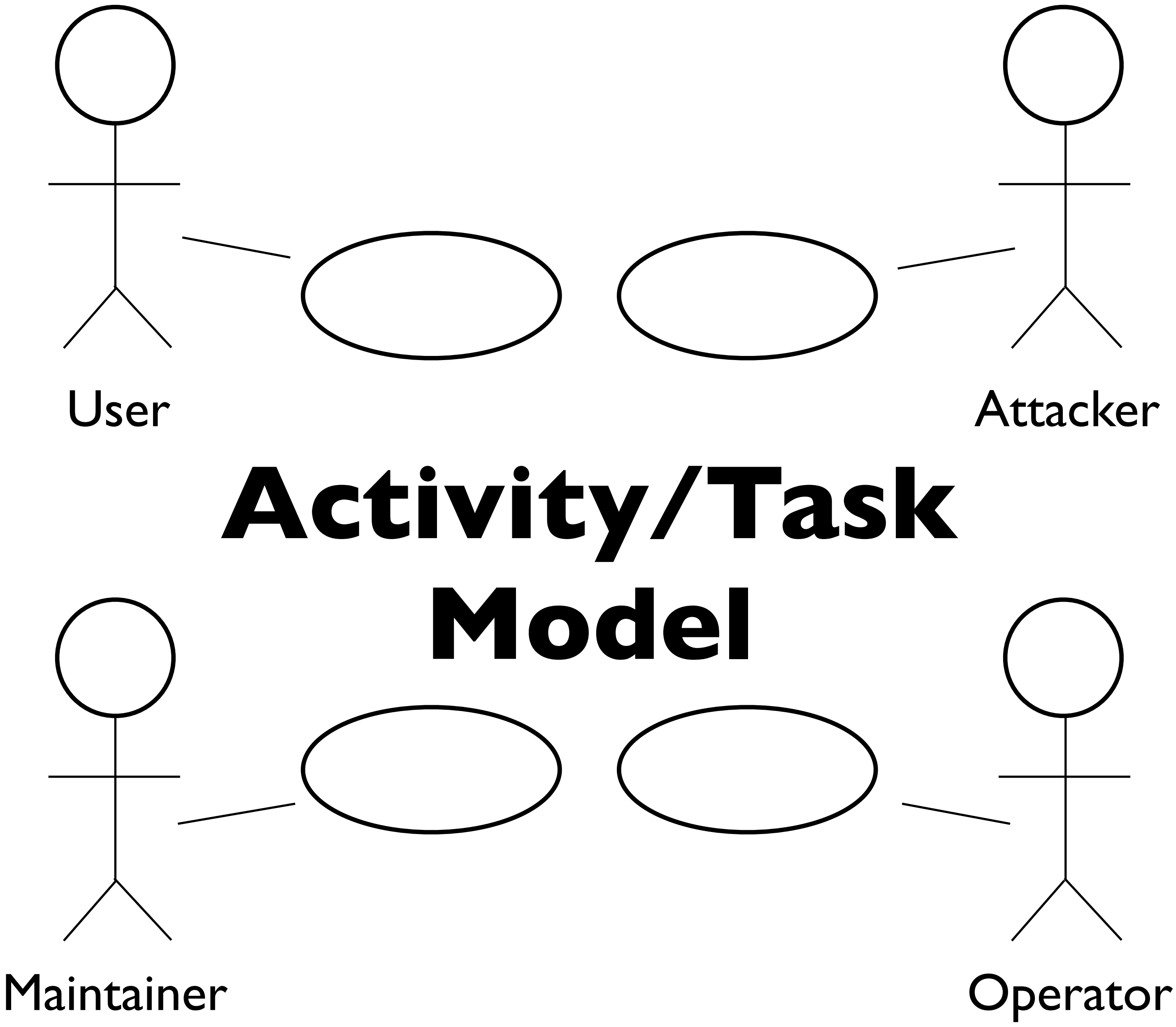
Glinz, On Non-Functional Requirements, 2007

- Part of the „non-functional“ requirements
- Describe not the core functionality but its quality
- Although at a certain level they become functional
- Usually based on a quality model
- Ideally contain
  - the involved stakeholders
  - several layers of abstraction
  - concrete instructions for realisation
  - quantitative measures for checking

How to discuss with stakeholders?

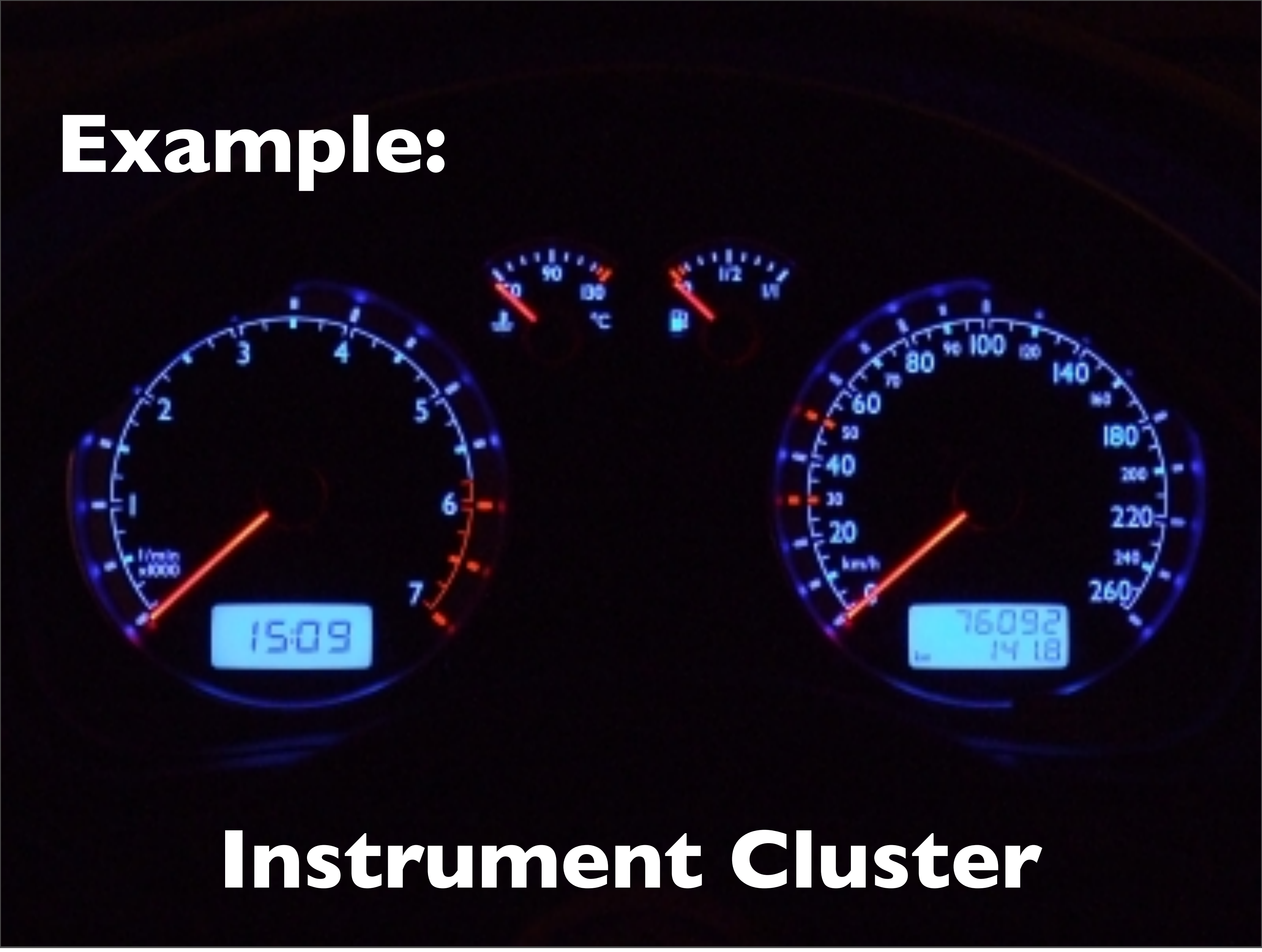
Quality requirements are the major means to discuss quality with the stakeholders.
They hence need to be on a level that is understandable for all stakeholders.
The structure of quality requirments needs to support this.
Again, the tasks and activities a stakeholders perform are something they are familiar with.

Activity/Task Model



In value-driven quality models, there should be explicit activity oder task models. These allow a direct connection to use cases and scenarios. For security, there are also misuse cases that describe an unwanted interaction.
This helps in making requirements more concrete.
It is an abstraction that is comprehensible for many stakeholders. Requirements engineers are also familiar with use cases and scenarios.
Hence, it constitutes a usable method for eliciting and specifying quality requirements.

Example:



Instrument Cluster

Attractive Appearance
The display and behavior of the rev meter shall be attractive, sporty, and agile.

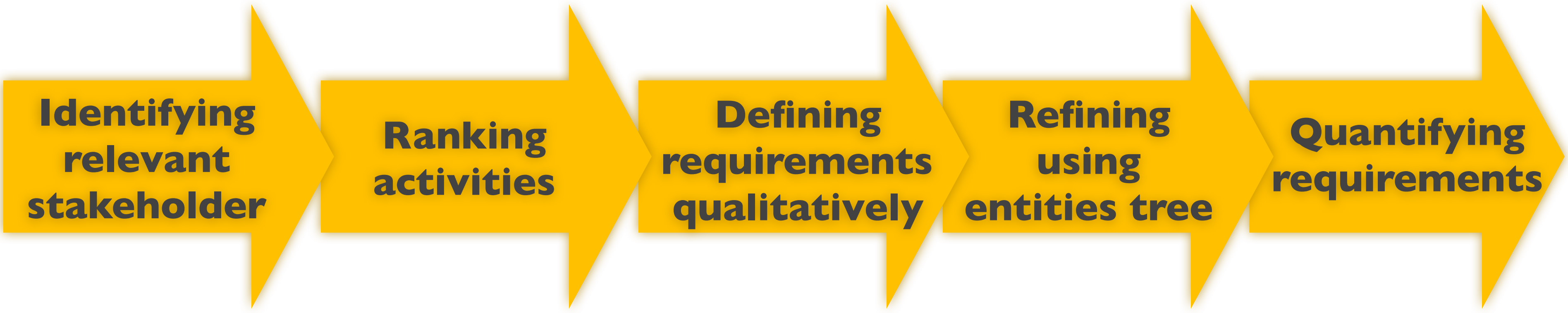
Comfortable view on important status information
The driver shall have a comfortable view on important status information of the car. The driver shall know the reliability of the engine. Furthermore the driver shall know which information is indicated to other road users.

K. Buhr et al. DaimlerChrysler Demonstrator: System Specification Instrument Cluster. ITEA, 2003

These requirements are taken from an instrument cluster specification for a car.
There are three major uses that the requirements (after communication to the stakeholders) are used for:

- Implementation
  - How do I achieve it?
- Derivation of test cases
  - What does „attractive, sporty and agile“ actually mean?
  - What are appropriate test cases?
- Quality measurement
  - What measures are there for comfortable?
  - In what contexts and for which activities?

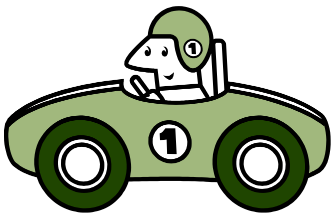
These requirements need to be refined. Currently, they are not measureable and therefore not testable/verifiable.



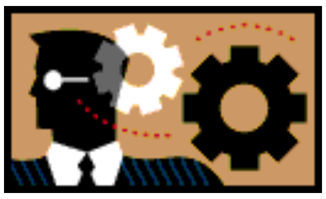
Identifying relevant stakeholder
Ranking activities
Defining requirements qualitatively
Refining using entities tree
Quantifying requirements

Wagner et al., Managing quality requirements using activity-based quality models, 2008

Identifying relevant stakeholder



Driver

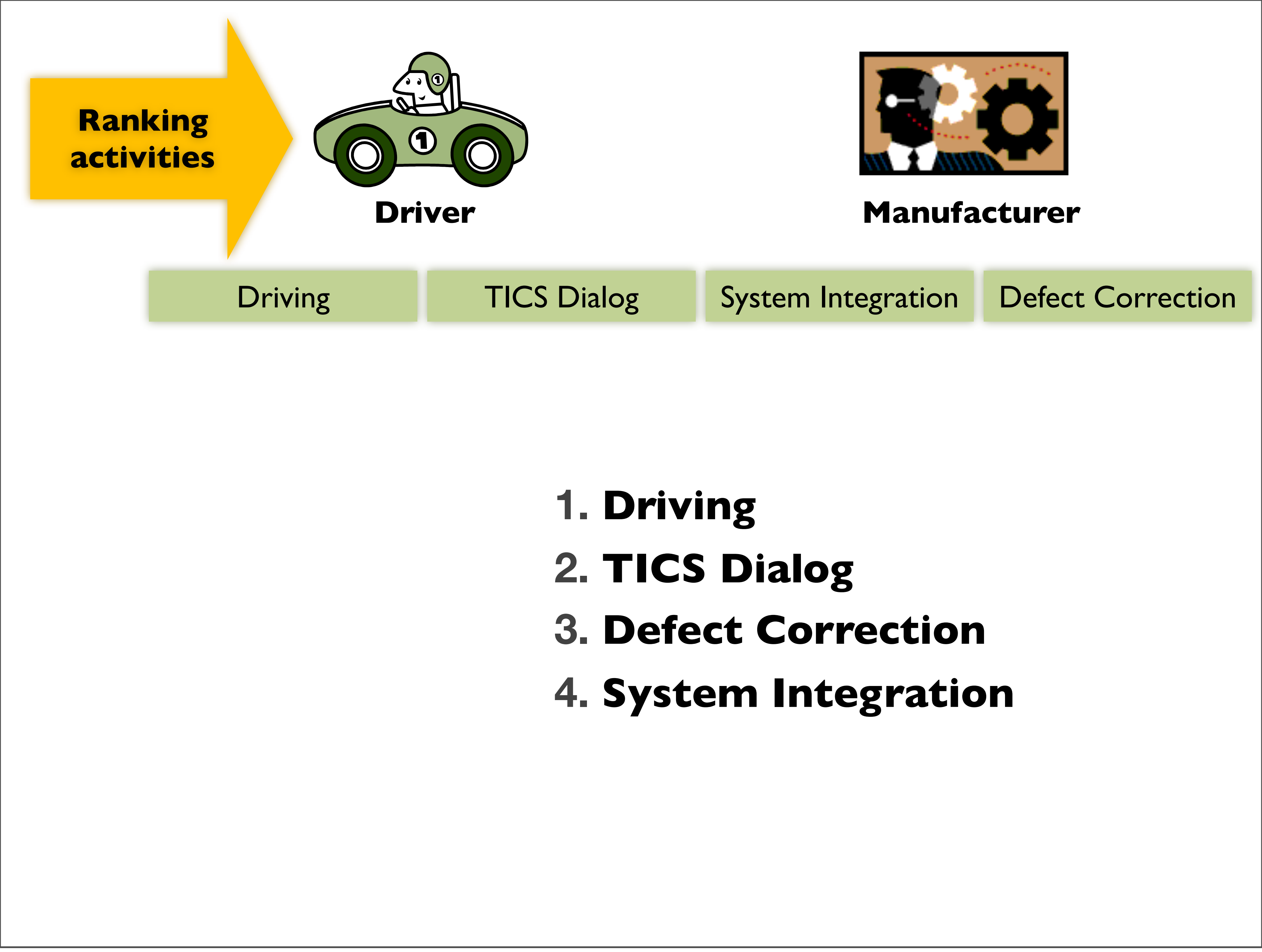


Manufacturer

Driving | TICS Dialog | System Integration | Defect Correction

In the example, we use the stakeholders driver and manufacturer with a sample of their activities.
In reality there would be more.

TICS: transport information and control systems (ISO 15005)



Ranking activities

Driver

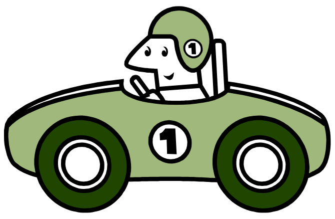
Manufacturer

Driving | TICS Dialog | System Integration | Defect Correction

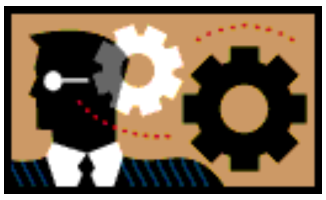
1. **Driving**
2. **TICS Dialog**
3. **Defect Correction**
4. **System Integration**

This is a possible ranking of the activities. The ranking would be important if we find conflicts between the activities.
For example, a property that is good for defect correction could be bad for the TICS dialog. Then we would need to decide how to implement the system.

Defining requirements qualitatively



Driver



Manufacturer

Driving | TICS Dialog | System Integration | Defect Correction

Comfortable

Safe

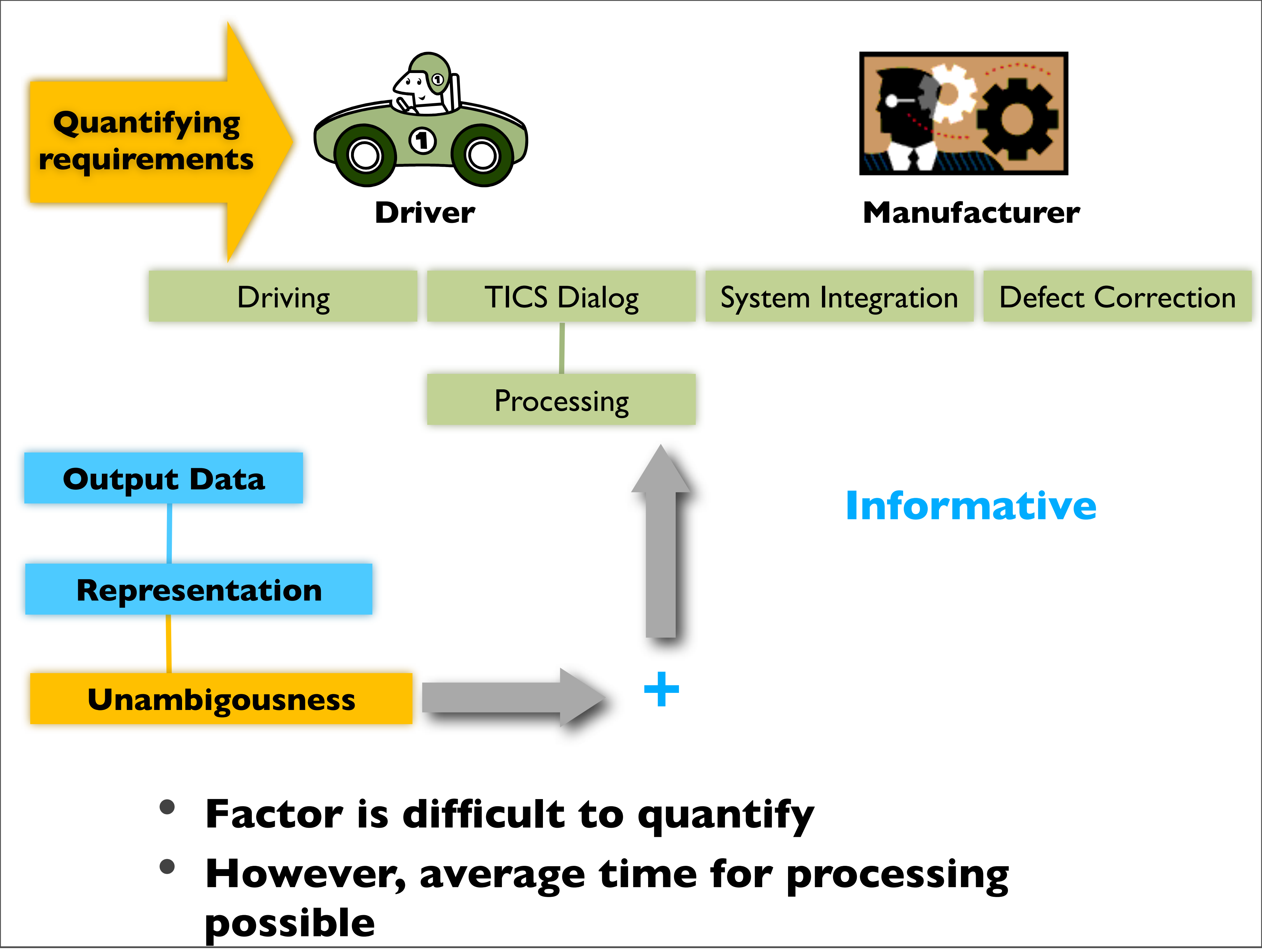
Not distracted

Informative

Intuitive

Reliable

This describes possible qualitative requirements for driving and TICS dialog.



Quantifying requirements

Driver

Manufacturer

Driving

TICS Dialog

System Integration

Defect Correction

Processing

Output Data

Representation

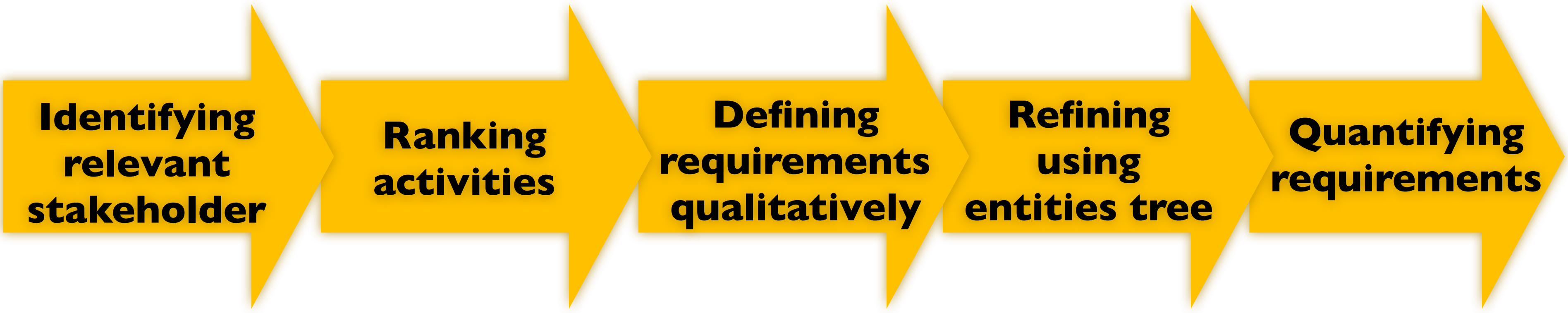
Unambigousness

+

Informative

- **Factor is difficult to quantify**
- **However, average time for processing possible**

We concentrate on the requirement that the TICS dialog should be informative.
To refine this, we use the sub-activity processing, that describes the processing of the information of the dialog by the user.
In the quality model, we find that the unambigousness of the output data representation has a positive influence on this.
Hence, we refine the requirement to this factor.
In this case, the factor is not more easy to quantify. We could, however, quantify the processing time or effort of the user.
The factor would rather become part of a checklist for an usability review.



Identifying relevant stakeholder

Ranking activities

Defining requirements qualitatively

Refining using entities tree

Quantifying requirements

Identifying relevant stakeholder

- Similar to other approaches
- User, customer, operator, developer, maintainer, trainer
- Using quality model to derive corresponding activities

Ranking activities

- List of activities
- Ranked according to importance
- Elaborate and often performed activities
- Expert opinion or similar projects

Defining requirements qualitatively

- How well should these activities be supported?
- Based on ranking of activities
- Some can be *Don‘t care*

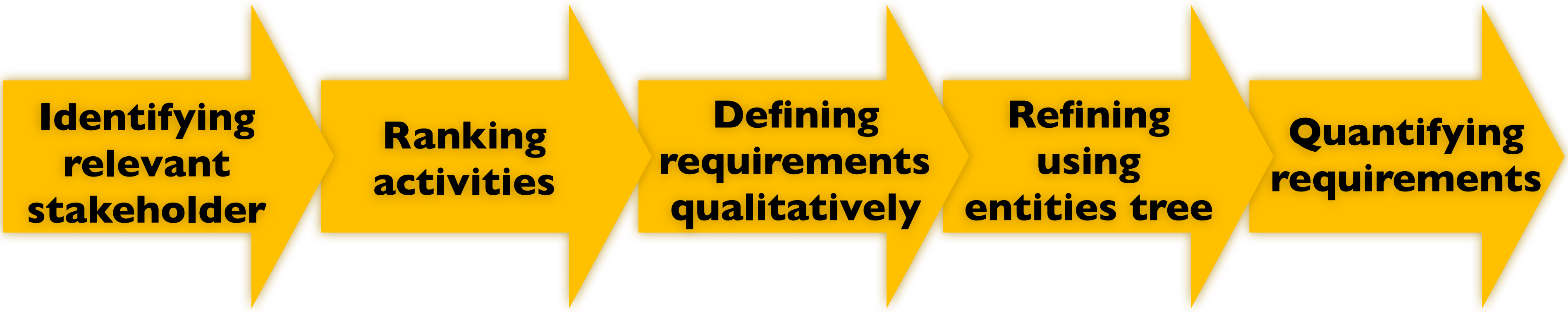
Refining using entities tree

- Follow influences on activities
- Identify important entities and attributes
- Define requirements on those
- Using importance of activities

Quantifying requirements

- Measureable facts in the model
- Define measureable values
- Not for all requirements possible

Quality Requirements Refinement


Identifying relevant stakeholder

Ranking activities

Defining requirements qualitatively

Refining using entities tree

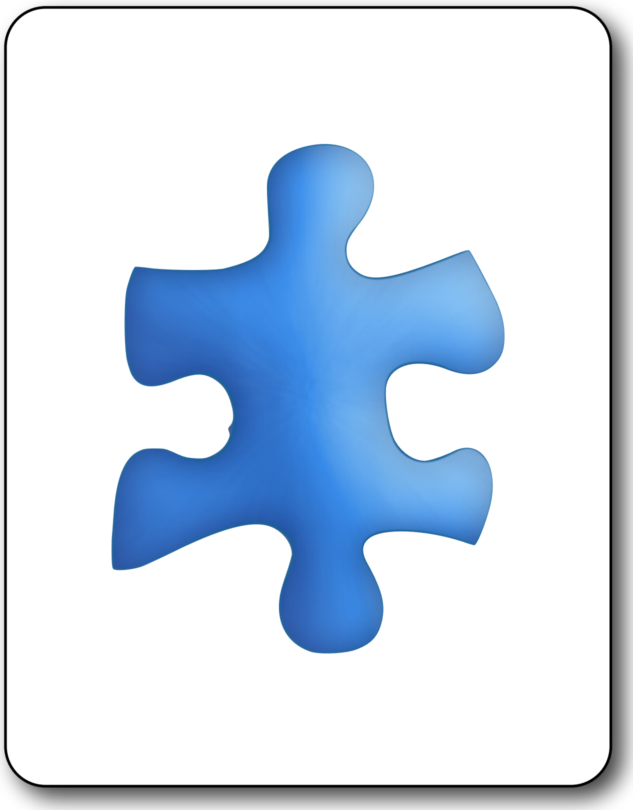
Quantifying requirements

Reuse potential of more than 40%!

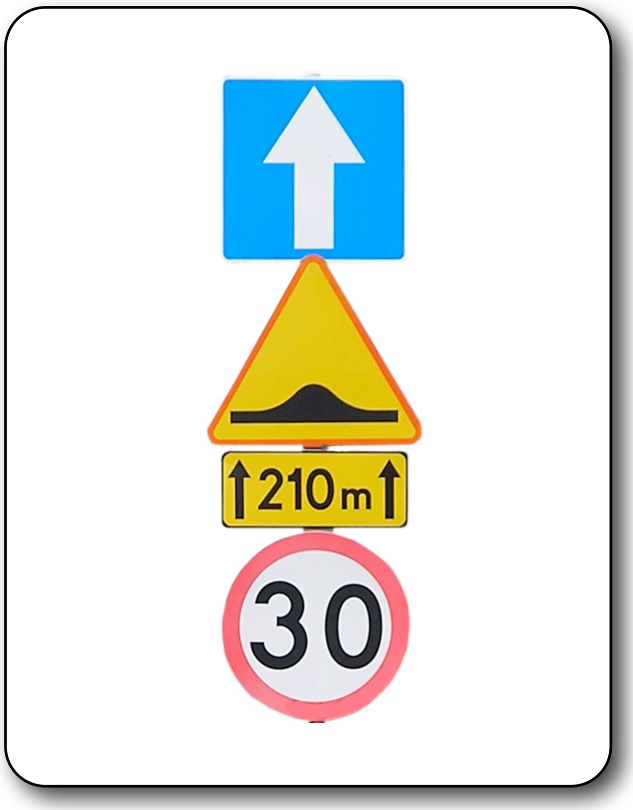
Luckey et al., Reusing security requirements using an extended quality model, 2010

An overview of the specification and refinement process for quality requirements based on value-driven/activity-based quality models.
In an empirical study at Capgemini sd&m we found that using such an approach there is a reuse potential of 47%. This means that almost half of the requirements specified in each project could be reused if it were specified using such a quality model.

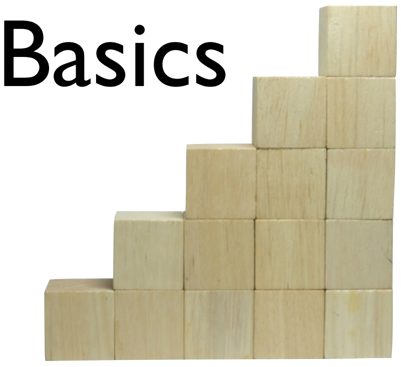
Quality **Models**



Quality **Require-ments**



Basics



Product

Quality

Metrics and

Measurement

Process

Quality

Quality

Management

Certifi-
cation