

Managing Quality Requirements Using Activity-Based Quality Models

Stefan Wagner, Florian Deissenboeck, Sebastian Winter
Technische Universität München, Germany

An Instrument Cluster Specification

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.



Scenarios of Specification Use

- 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?

→ **Refinement**

→ **No measures → not checkable**

Problem Statement

- David A. Garvin, 1984: „Quality is a complex and multifaceted concept. It is also the source of great confusion...“
- Holds also for quality requirements
- Often not handled satisfyingly in practice
- Also neglected in research
- Results in
 - vague quality requirement definitions
 - large step to implementation
 - difficult quality assurance

→ **Need for a structured approach for quality requirements management**

Outline

- Quality Requirements
- Activity-Based Quality Models
- Proposed Approach
- Case Study
- Conclusions

Quality Requirements

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

Quality Models

- Abstract definition of the attributes important for quality
- Basis of quality requirements
- Standard: ISO 9126
- Problems
 - Too abstract for operational use
 - Ambiguous decomposition
 - No customisation mechanisms
 - Relation to costs?



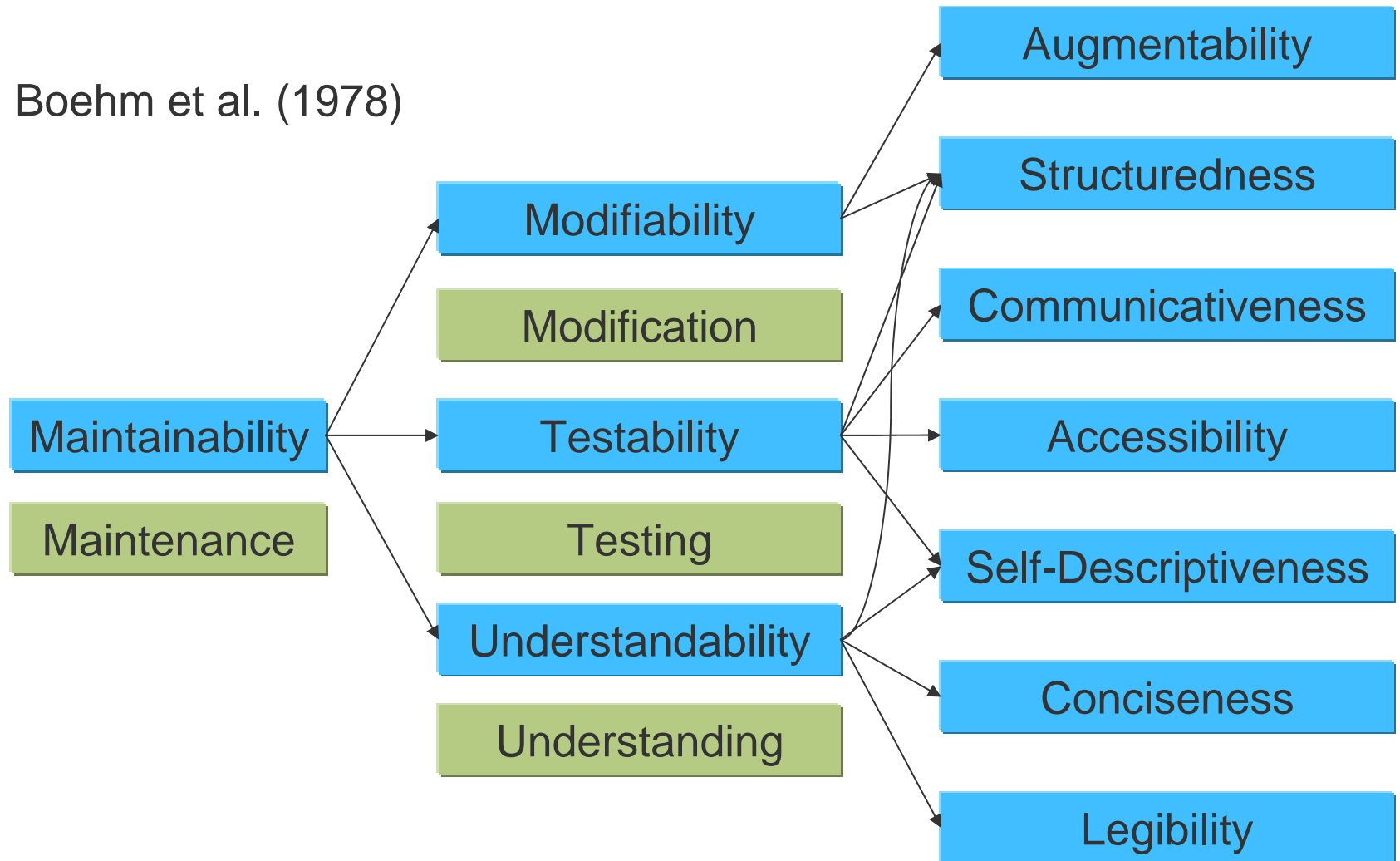
Sketch of Proposed Approach

General theme: Only specify what you can measure!

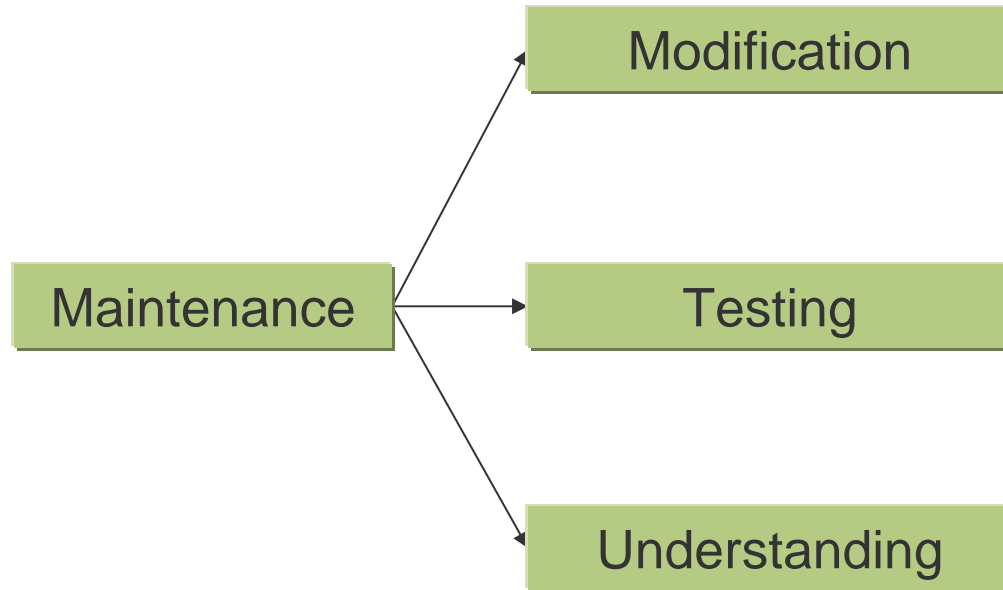
- We need a clear connection to the stakeholders
 - Most stakeholders perform some kind of activity with the system
 - Human activities cause the most costs in software
 - Quality is how good the system supports these activities
- **A quality model based on activities**
- **Use that model for requirements management**

Activity-Based Quality Models

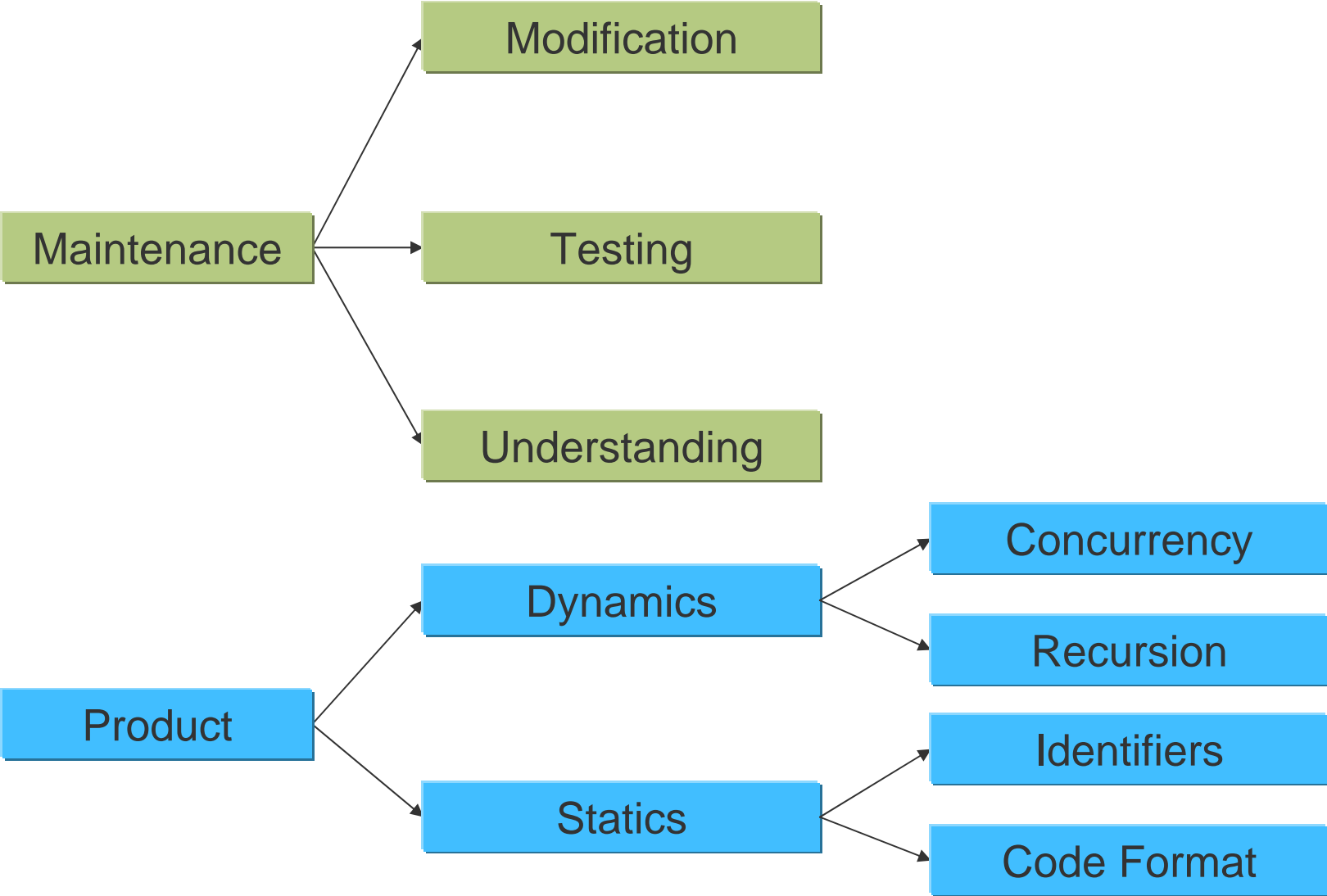
Boehm et al. (1978)



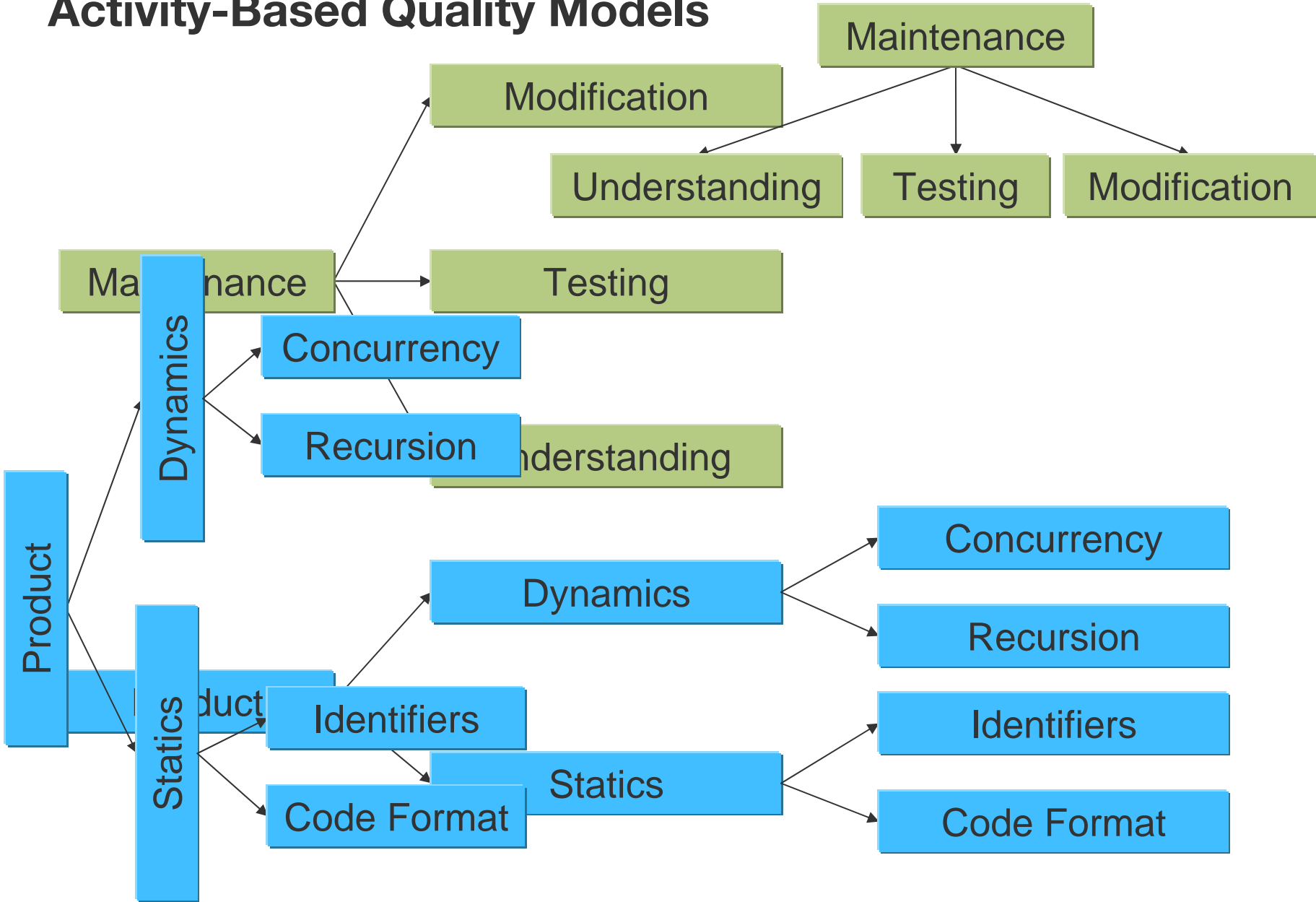
Activity-Based Quality Models



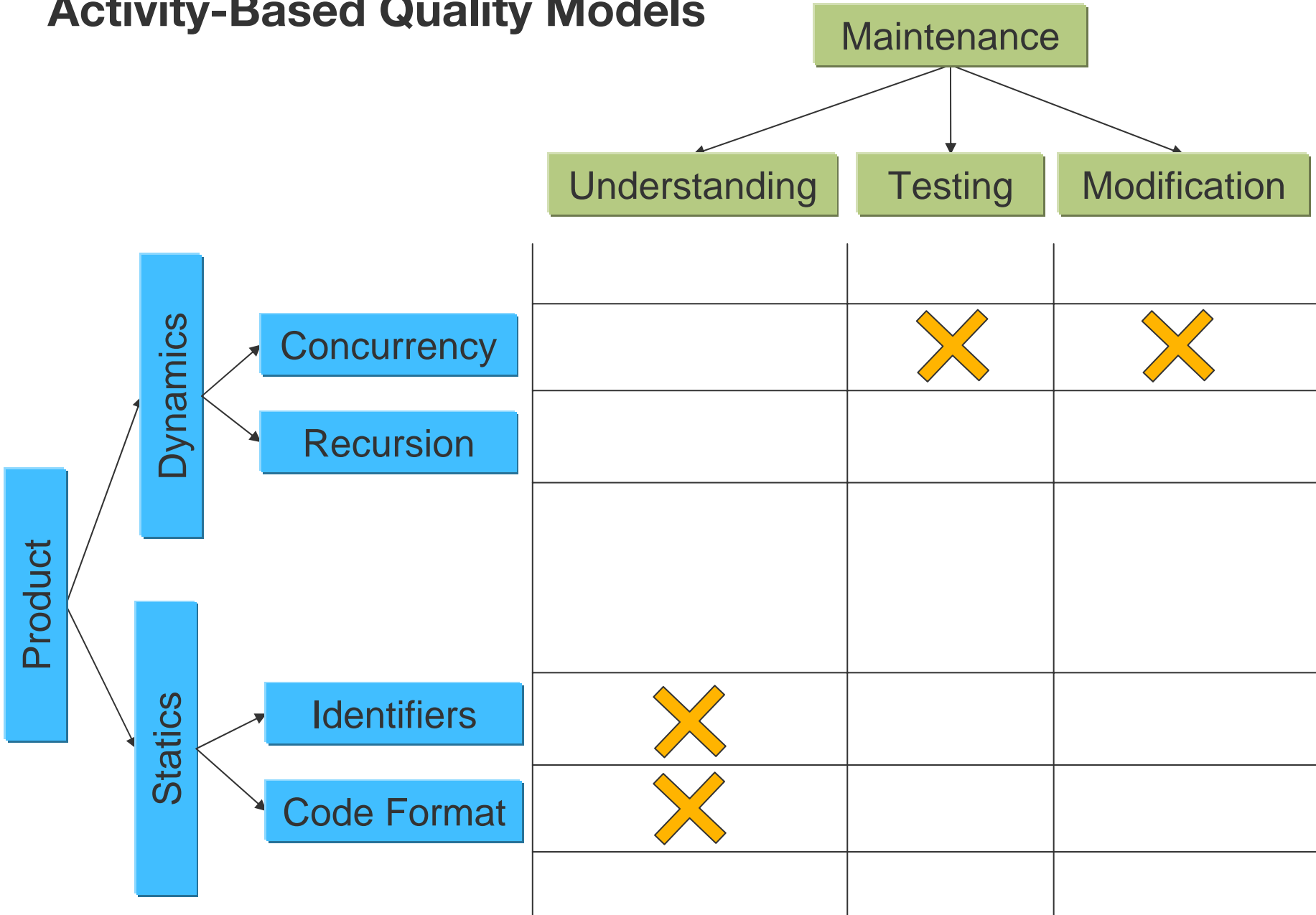
Activity-Based Quality Models



Activity-Based Quality Models



Activity-Based Quality Models



Quality Requirements Management



```
graph LR; A[Identifying relevant stakeholder] --> B[Ranking activities]; B --> C[Defining requirements qualitatively]; C --> D[Refining using entities tree]; D --> E[Quantifying requirements];
```

**Identifying
relevant
stakeholder**

**Ranking
activities**

**Defining
requirements
qualitatively**

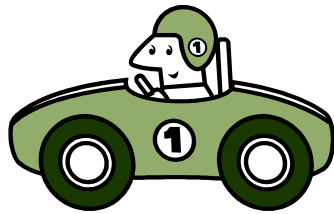
**Refining
using
entities tree**

**Quantifying
requirements**

Case Study



Identifying
relevant
stakeholder



Driver



Manufacturer

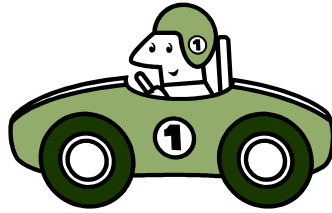
Driving

TICS Dialog

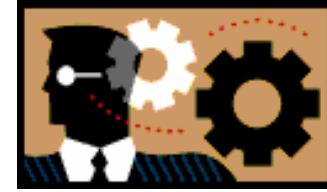
System
Integration

Defect
Correction

**Ranking
activities**



Driver



Manufacturer

Driving

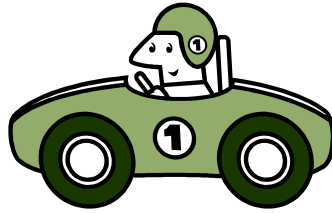
TICS Dialog

System
Integration

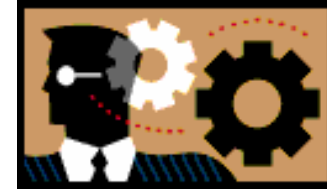
Defect
Correction

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

Defining requirements qualitatively



Driver



Manufacturer

Driving

TICS Dialog

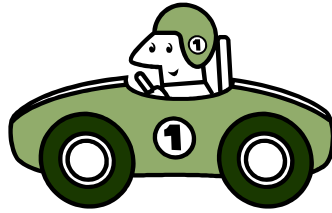
System
Integration

Defect
Correction

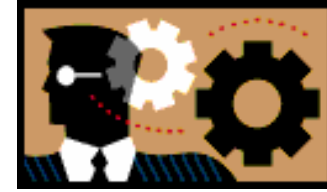
- **Comfortable**
- **Safe**
- **Not distracted**

- **Informative**
- **Intuitive**
- **Reliable**

Quantifying requirements



Driver



Manufacturer

Driving

TICS Dialog

System
Integration

Defect
Correction

Processing

Output Data

Representation

Unambiguousness

• Informative

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

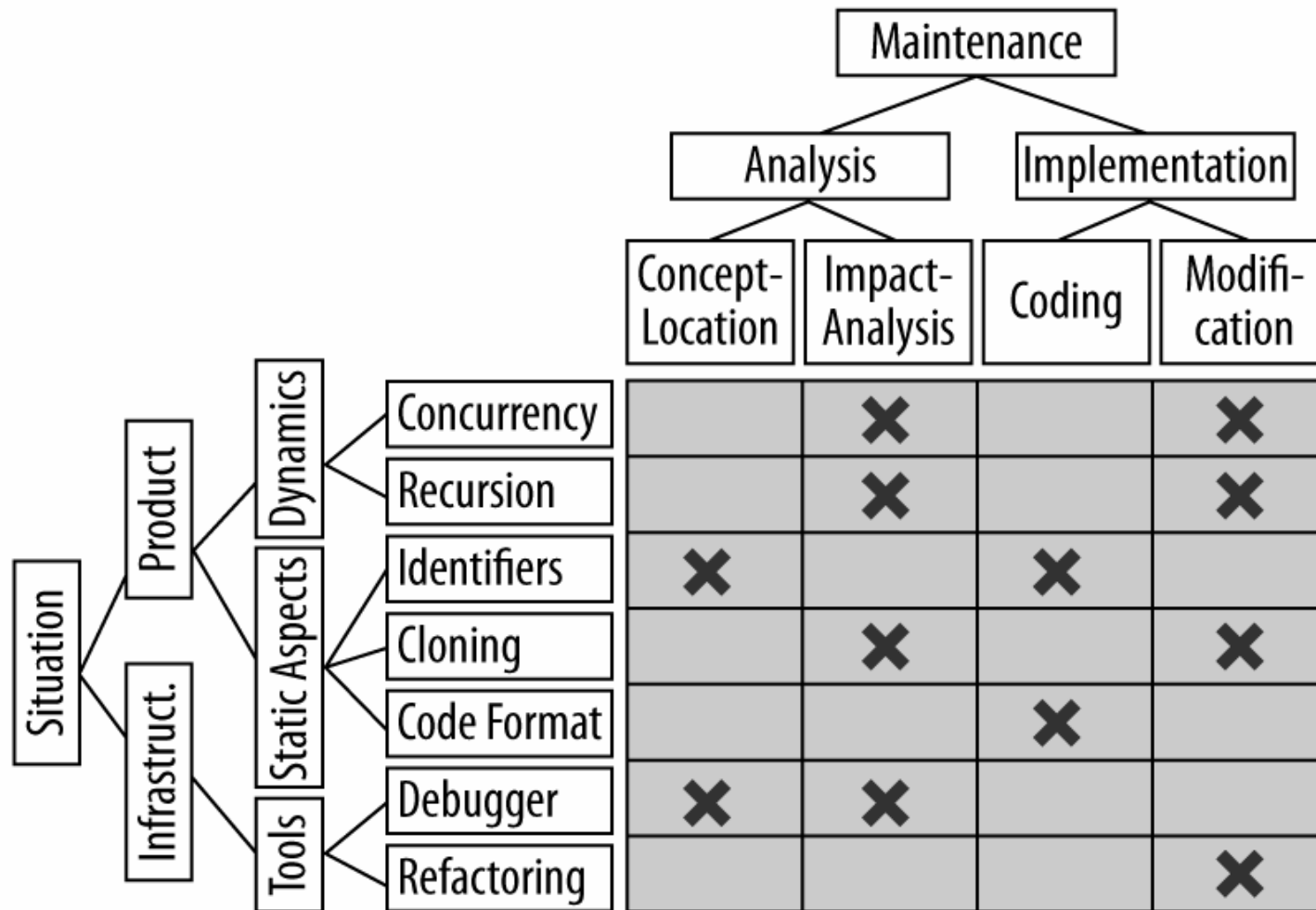
Conclusions

- Managing requirements based on activity-based quality models
- Relation to the stakeholder
- Refinement using entities tree
- Support for quantification
- Future work
 - Incorporation of use cases and architecture
 - Case studies with emphasis on different quality attributes


Related Work

- Cysneiros, do Prado Leite: non-functional requirements
- Mylopoulos, Chung, Nixon: concentration on RE process
- Ebert: classification in **user-oriented** and **developer-oriented** requirements
- UMD by Basili et al.: modelling of **issues** that should be avoided
- Doerr et al: usage of standard quality models for non-functional requirements

Activity-Based Quality Models



Quality Requirements Management I



```
graph LR; A[Identifying relevant stakeholder] --> B[Ranking activities]; B --> C[Defining requirements qualitatively]; C --> D[Refining using entities tree]; D --> E[Quantifying requirements];
```


Identifying relevant stakeholder

Ranking activities

Defining requirements qualitatively

Refining using entities tree

Quantifying requirements



Identifying relevant stakeholder


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

Quality Requirements Management II



**Ranking
activities**

- List of activities
- Ranked according to importance
- Elaborate and often performed activities
- Expert opinion or similar projects
- How well should these activities be supported?
- Based on ranking of activities
- Some can be *Don't care*



**Defining
requirements
qualitatively**

Quality Requirements Management III



**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