Classification of quality attributes for predictability in component-based systems

Ivica Crnkovic
Mälardalen University, Sweden
Department of Computer Science and Engineering
www.idt.mdh.se/~icc, ivica.crnkovic@mdh.se

Magnus Larsson
ABB Research, Sweden

Component-based approach

- Building systems from (existing) components
- Component development is separated from system development process
- A combination of a bottom-up and top-down approach
- Many explicit and implicit assumptions
  - Architectural styles (middleware, deployment,..)

Why component-based approach?

- Primary a concern of business and life-cycle factors
  - Costs, Time-to-market
  - Flexibility
  - Understandability, maintainability
  - Reuse of already existing software
- Higher abstraction level for *functional* properties
- To less degree a concern of non-functional properties
  - The requirements that must be fulfilled *also* with this approach
  - Sometimes more difficult to achieve
  - Might be a reason that component-based approach is less (or not) feasible

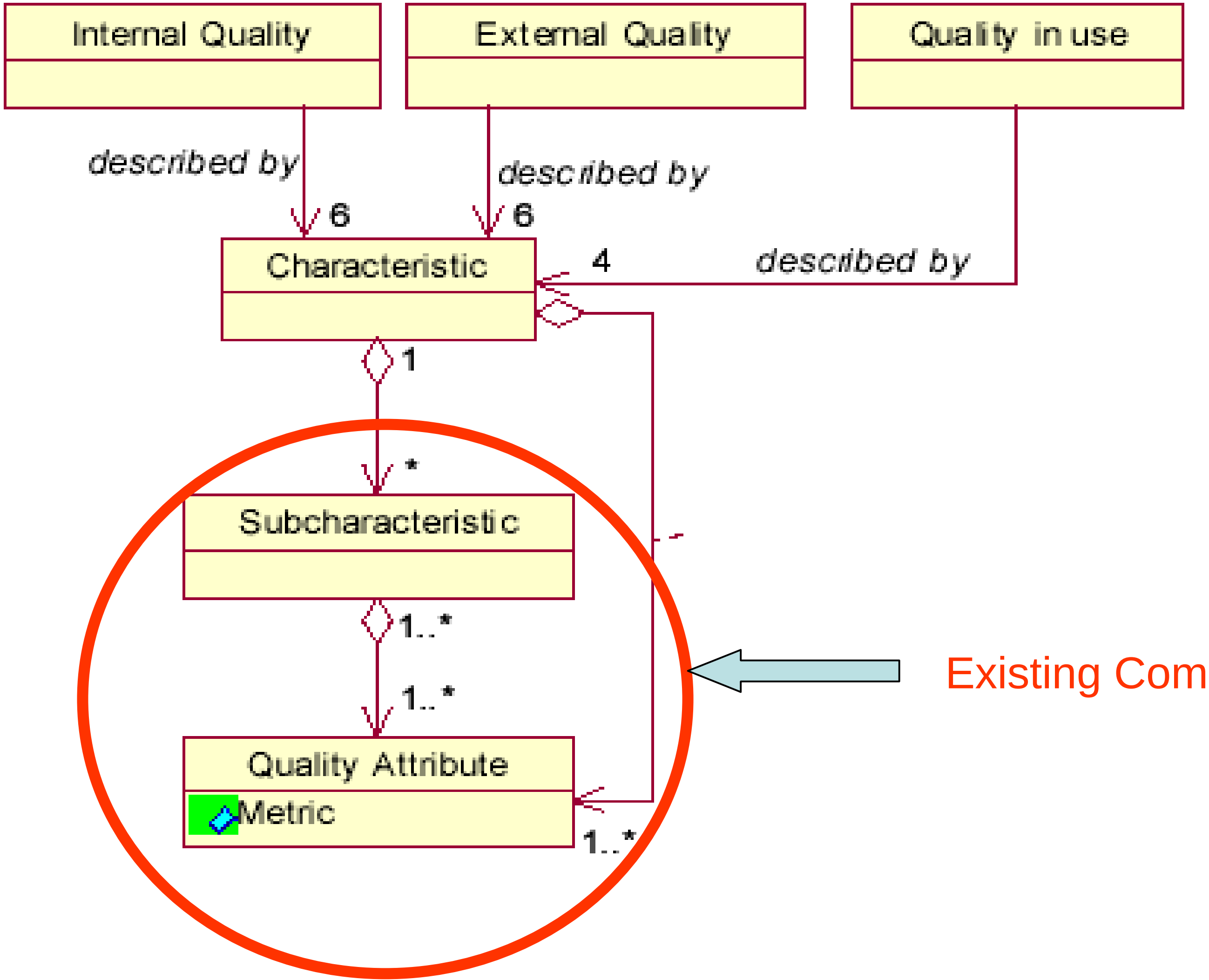
The main question(s)

- Is component-based approach appropriate for building (dependable) systems?
  - Yes
  - No
  - Irrelevant
- To which extent components (and not only architecture) determine the properties of a system?
  - (Remember: you are not developing components that will meet your requirements, you are adopting existing components)

Predictable behavior of assemblies- important questions

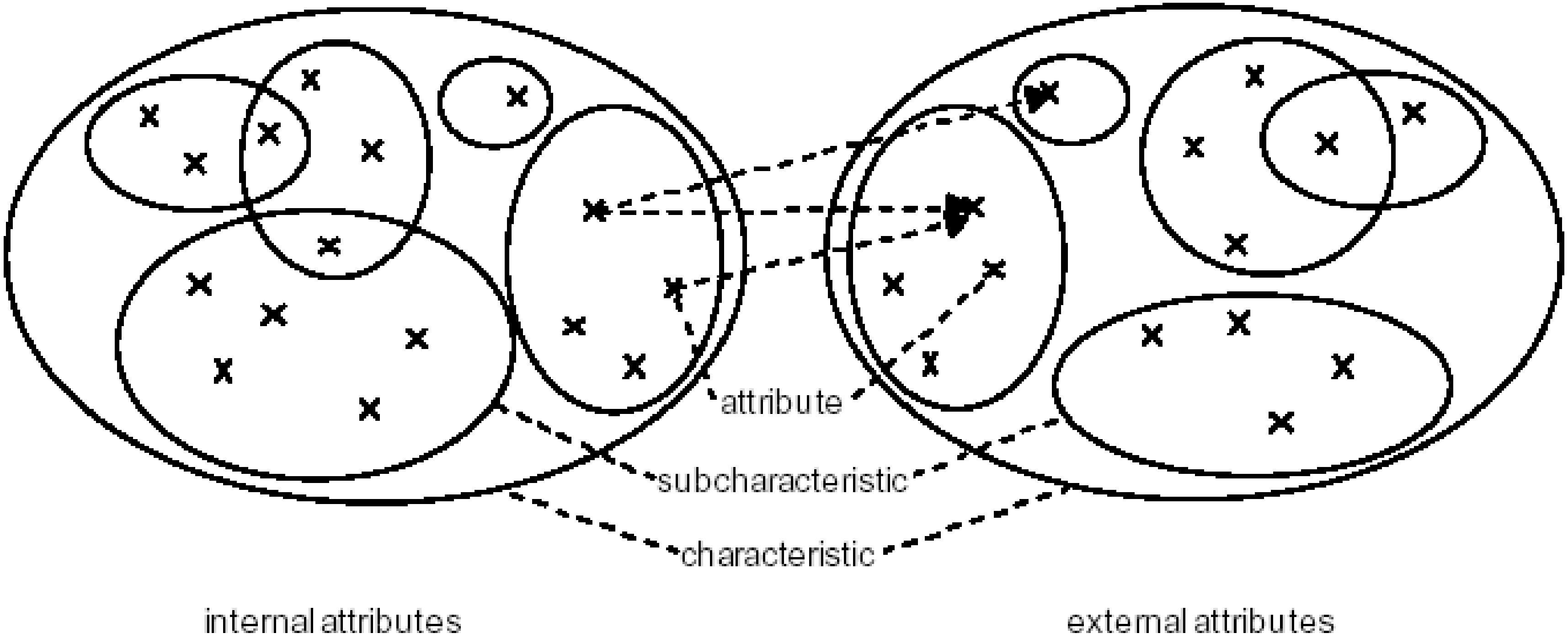
- Given the system quality attributes required**,** which properties are required of the components concerned?
- Given a set of component properties, which system properties are predictable?
- How can system quality attributes be accurately predicted, from the properties of components which are determined with a certain (in)accuracy?
- To which extent, and under which constraints are the emerging system properties (i.e. the system properties non-existent on the component level) determined by the component properties?

General Concepts of the ISO/IEC 9126-1



Existing Components

Quality characteristics, sub-characteristics and attributes



Problem Statement

- Composability problem
  - Which properties are composable? Which properties are justifiable composable?
  - Can we classify attributes (properties) according to COMPOSITION PREDICTABILITY (i.e. ability to predict properties of component assemblies BEFORE the assemblies are created and being performed)?
  - (what must be known/specified to achieve a certain level of predictability?)

Classification

1. *Directly composable properties.*
   A property of an assembly which is a function of, and only of the same property of the components involved.

2. *Architecture-related properties*.
   A property of an assembly which is a function of the same property of the components and of the software architecture.

3. *Derived (emerging) properties.*
   A property of an assembly which is result on several different properties of the components and software architecture.

4. *Usage-depended properties.*
   A property of an assembly which is determined by its usage profile.

5. *System context properties.*
   A property which is determined by other properties and by the state of the system environment*.*

1. Definition: *A directly composable property of an assembly is a function of, and only of the same property of the components.*

$$P = \text{property}, A = \text{assembly}, c = \text{component}$$

$$A = \{c_i\}$$

$$P(A) = f(P(c_i)); i \in N$$

- Consequence: to derive (predict) a assembly property it is not necessary to know anything about the system(s)

Example

- “Physical characteristics”
  - Static memory

$$M(A)=\sum_{i=1}^{n} M(c_i)$$

$$M = \text{memory size},\ A = \text{assembly},\ c_i = \text{components}$$

  - (the “function” can be much more complicated)
  - (the functions are determined by different factors, for example technologies, or design decisions)

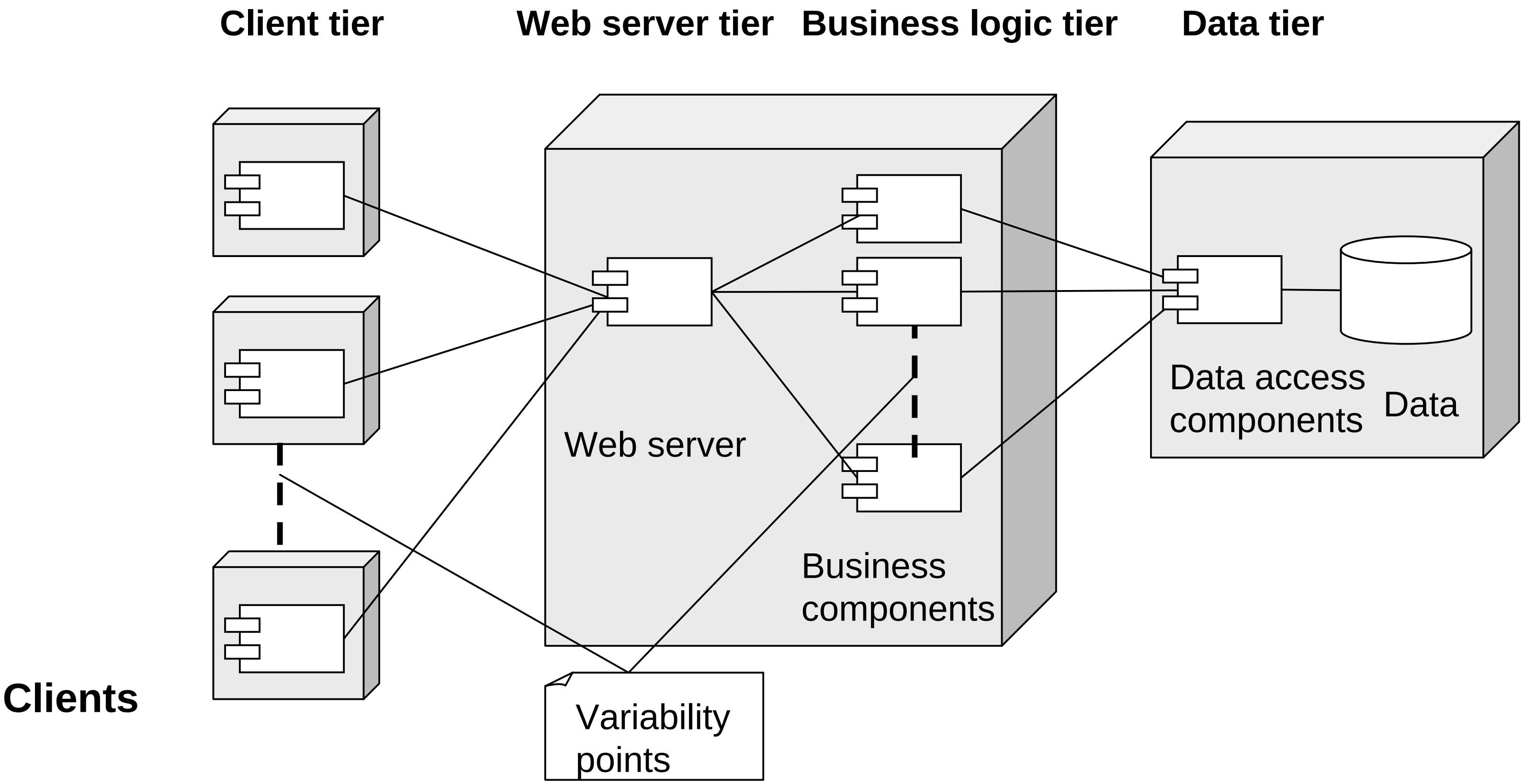
2. Definition: *An architecture-related property of an assembly is a function of the same property of the components and of the software architecture.*

$SA =$ software architectureture, $x_k =$ connections

$$P(A) = f(P(c_i), SA(c_i, x_k)); \quad i, k \in N$$

- *Consequence: System/assembly architecture must be known*
  - *Ok when building systems of particular class*

Example (J2EE or .NET distributed systems)



$$T/N = ax + b\frac{x}{y} + cy$$

T/N = execution time per transaction

x = number of clients; y = number of components

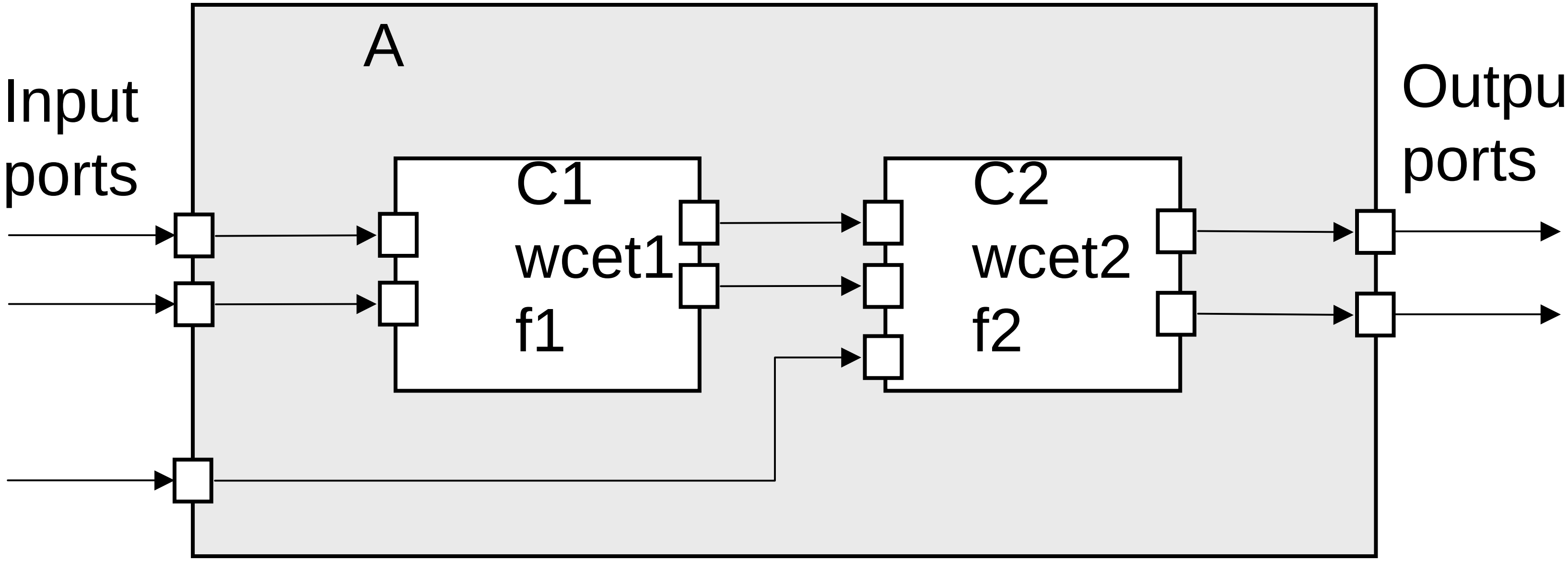
a, b, c = proportional factors for a particular implementation

3. Definition: *A derived property of an assembly is a property that depends on several different properties of the components*.

$$P(A) = f(P_1(c_i), P_2(c_i), \cdots, P_k(c_i));$$
$$i, k \in N$$
$$P = \text{assembly property}$$
$$P_1 \ldots P_k = \text{component properties}$$

- – Consequence: we must know different properties and their relations (might be quite complex)

Example



end-to-end deadline is a function of different component properties, such as **worst case execution time** (WCET) and **execution period.**

4. Definition: *A Usage-dependent property of an assembly is a property which is determined by its usage profile.*

$$P\,(A, U_k) = f\,(P(c_i, U'_{i,k}));\ \ i,k \in N$$

P = property for a particular usage profile

U_k = assembly usage profile

$U'_{i,k}$ = component usage profile

Consequence: It is not enough to know which system will be built. It must be known how the system will be used

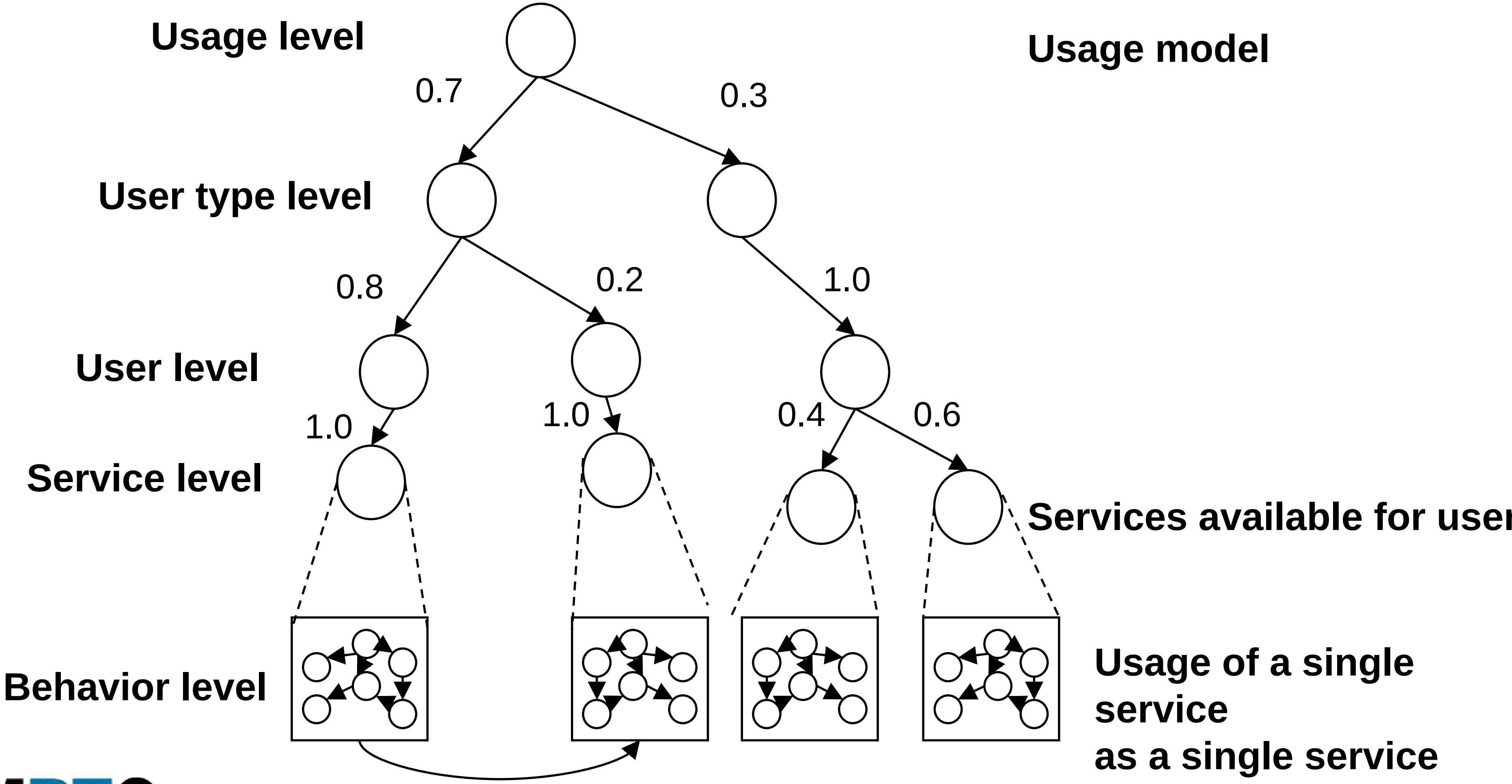
Example: Reliability

- Mean-time between failure
- How to calculate?
- The process
  - Define usage model
  - Define the usage profile
    - On the system level and component level
  - Define the test cases
  - Execution of test cases

Usage modeling and usage profile

- Intended to model external view of the use of components
- Use of Markov chains (FSM + probability of transition between states)
  - Problem – for complex systems Markov chains become very large
  - Attempt to solve the complexity by introduction of State Hierarchy Model [Claes Wohlin & Per Runesson 1994]

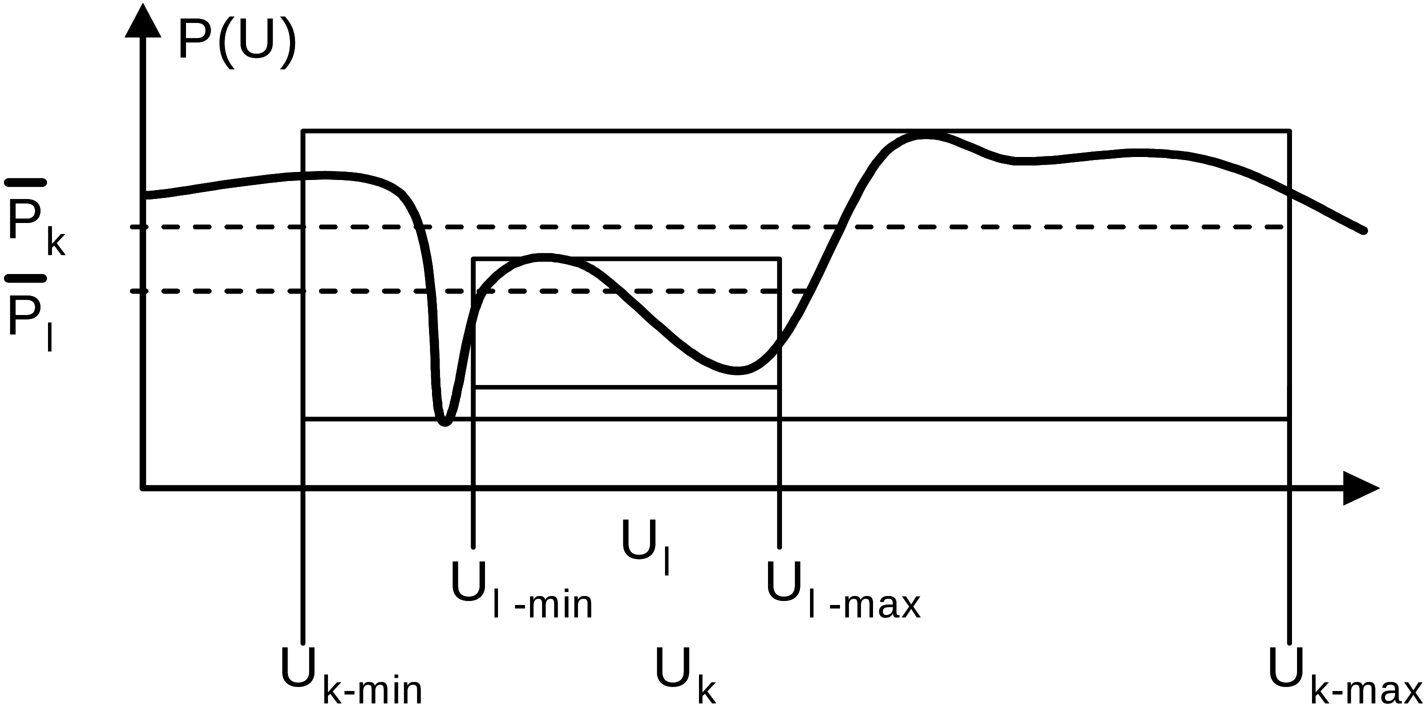
Usage profile – probabilities of usage



Reuse problem:

mapping system usage profile to component usage profile

When the known (measured) properties values can be reused?



5. Definition: *A System Environment Context property is a property which is determined by other properties and by context of the system environment.*

$$P_k(S,U_k,E_l) = f(P_k(c_i,U'_{i,k}),E_l); \quad i,k,l \in N$$

$$U_k = \text{System usage profile;}$$

$$E_l = \text{Environment context}$$

$$S = \text{System}$$

$$U'_{i,k} = \text{Component usage profile}$$

– **Consequence:** ***It is not sufficient to know the systems and their usage, it is necessary to know particular systems and the context in which they are being performed***

Example

- safety property
  - related to the potential catastrophe
  - the same behavior may have different safety concerns even for the same usage profile.

Survey of properties

Similar to ISO 9126-1 model (characteristics and subcharacteristics):

- Quality attributes grouped in Concerns
- About 50 different quality attributes (taken from different references)

Classification process – an inquiry:

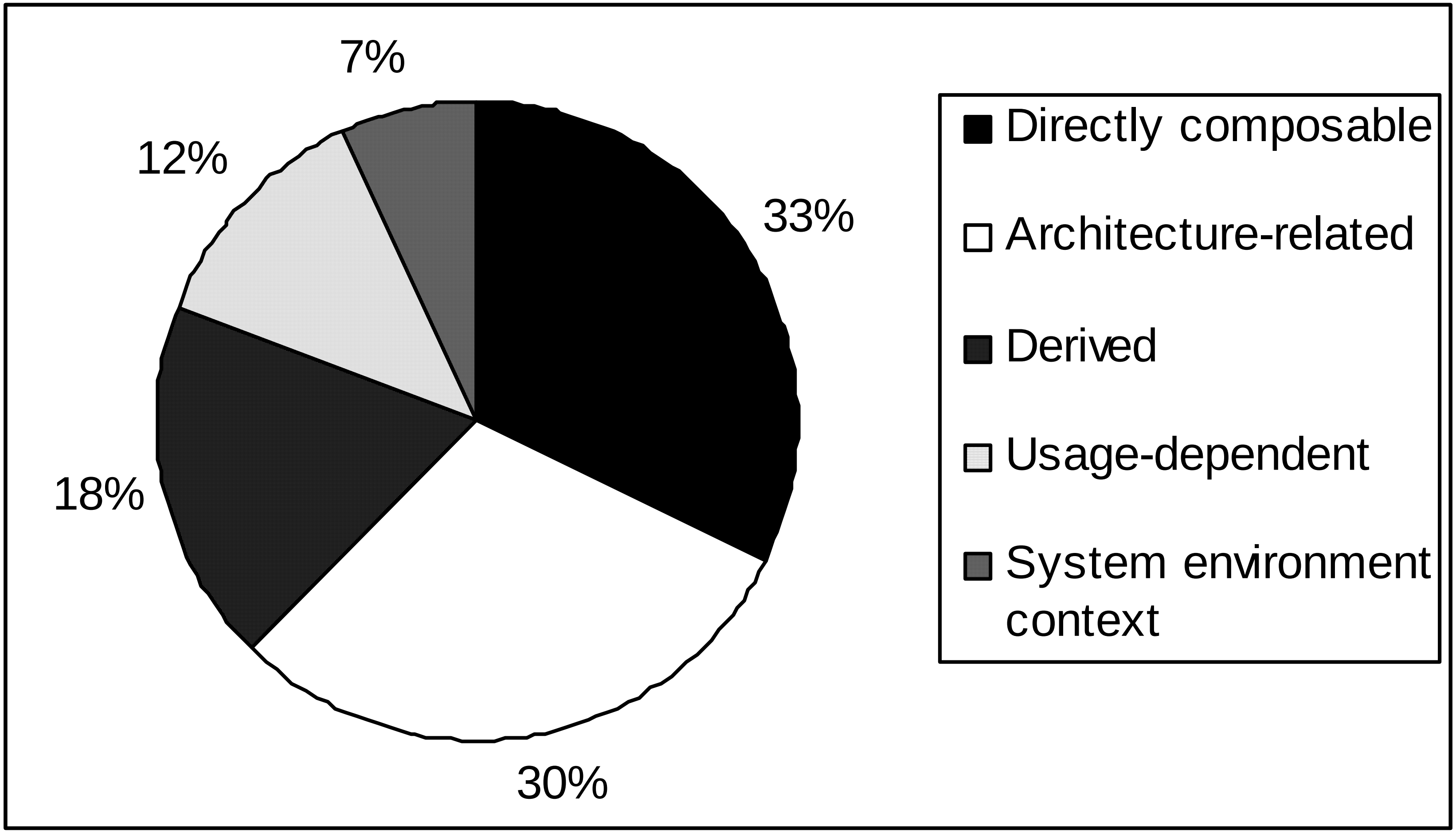
- Short description of the classification
- A definition of every quality attribute
- About 30 researchers (mostly from SA community) asked to classify the quality attributes

Survey questions

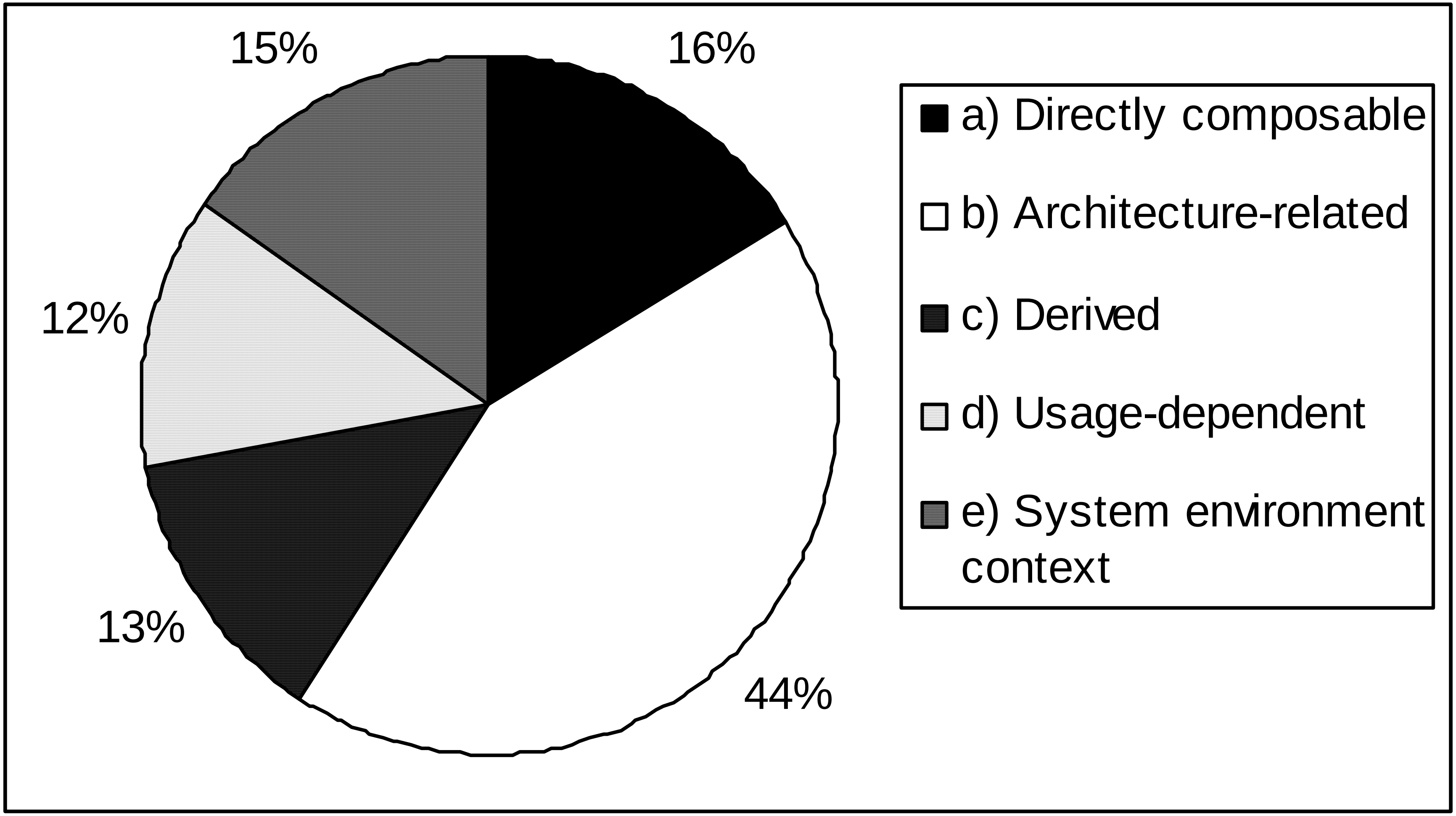
- Directly composable attributes - Is it possible to analyze this assembly property given the same property of the components involved?
- Architecture Related attributes - Is it possible to analyze this assembly property given the assembly software architecture and the same property of the components involved?
- Derived attributes - Is it possible to analyze this assembly property from several different component properties of the components involved?
- Usage-dependent attributes - Is it necessary to know the usage profile of the assembly to analyze this property ?
- System environment context dependent attributes - Is it necessary to have system environment information to analyze this property ?

Survey

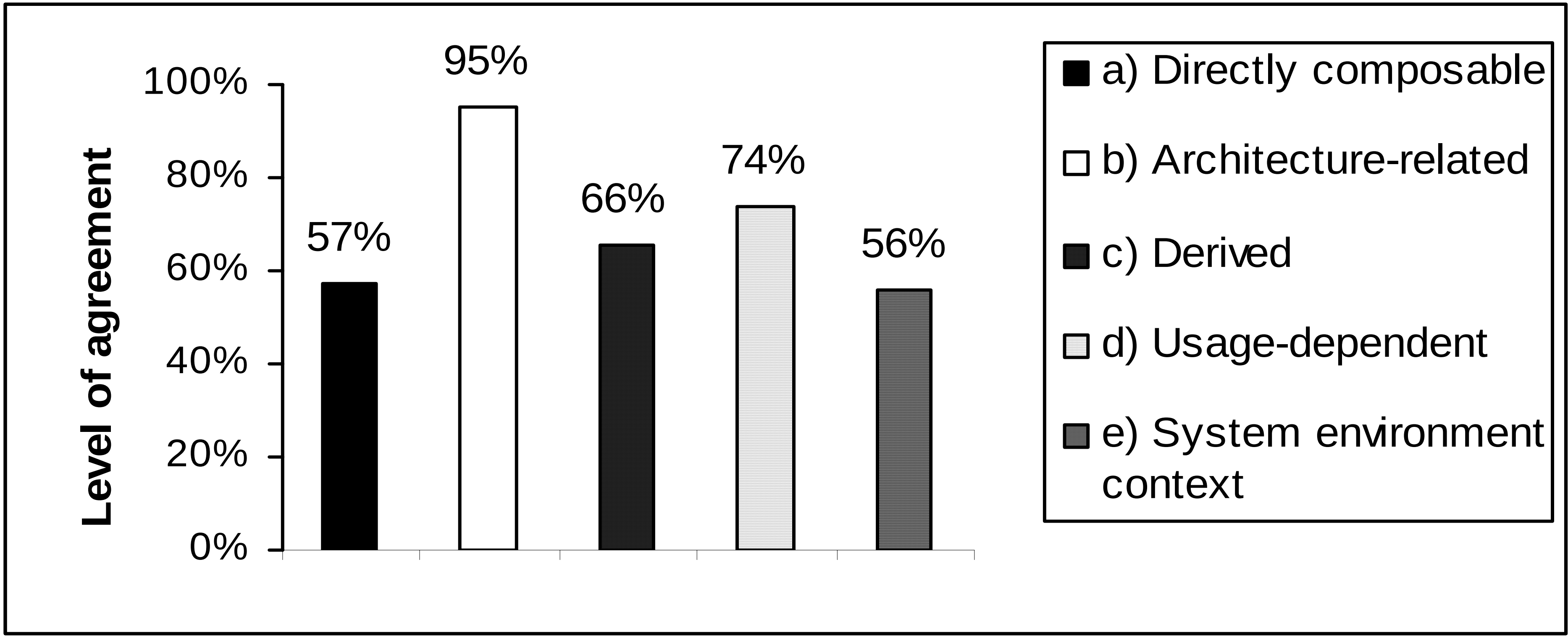
Results


7%
12%
33%
18%
30%
Directly composable
Architecture-related
Derived
Usage-dependent
System environment context

Survey


15%
16%
12%
13%
44%
■ a) Directly composable
□ b) Architecture-related
■ c) Derived
□ d) Usage-dependent
■ e) System environment context

Level of agreement of the participants for the classification



Dependability

- Using Laprie definition:
  - Attributes: Reliability, Availability, Safety, Confidentiality, Integrity, (Maintainability)
- Reliability – Usage-dependent attribute
- Availability – Usage-dependent
- Safety – system context
- Confidentiality, Integrity – not measurable and not composable
- Maintainability –not composable

Dependability and composability

- Difficult to predict dependability from the composition of the properties
- Increased possibility with different restrictions
  - – In architectural solutions
  - – Usage profiles
  - – ….

Conclusion

- “Return of investment” for component-based approach depends also on predictability and assurance of quality attributes
- Different engineering/application domains focus on different quality attributes
- In some domains (or for particular aspects) component-based approach include more problems – this should be related to the benefits.

- On-going work: study of
  - Vehicular systems (in particular automotive industry)
  - Robotics
    - and feasibility of a component-based approach

Results