Component-Based Software Engineering and composition of Quality Attributes

Ivica Crnkovic

What are Quality Attributes

- **A (Software) system perform a function – it provides functions**
  - Functions – **WHAT** the system is doing
- **A (software) system perform a function in a fashion – good, bad….**
- **Quality Attributes – HOW the system is performing the functions**
- **Quality attributes have different names**
  - Non-functional properties
  - Extra-functional properties
  - Properties
  - Ilities.. (reliability, Availability, Mobility, Maintainability,…)

Classification of Quality attributes

- Run-time:
  - ☞Reliability
  - ☞Availability
  - ☞Integrity
  - ☞Performance
  - ☞….
- Life time
  - ☞Maintainability
  - ☞Modifiability
  - ☞Portability
  - ☞Testability
  - ☞Usability
  - ☞Reusability
  - ☞…..

Importance of QA

- **In different domains different QAs are important**
  - Widely used software (office software)
    - ☞Important:
    - ☞Usability, Flexibility, Portability, Interoperability…
  - Safety-critical systems
    - ☞Reliability, Safety, Security,…

More about safety-critical systems…

- **Safety-critical real-time embedded systems**
  - Safety - “Absence of catastrophic consequences on the users and the environment”
  - Property depends on
    - ☞Environment
    - ☞Use of the system
    - ☞Consequently, software is only hazardous in a context
    - ☞It is a system behavior

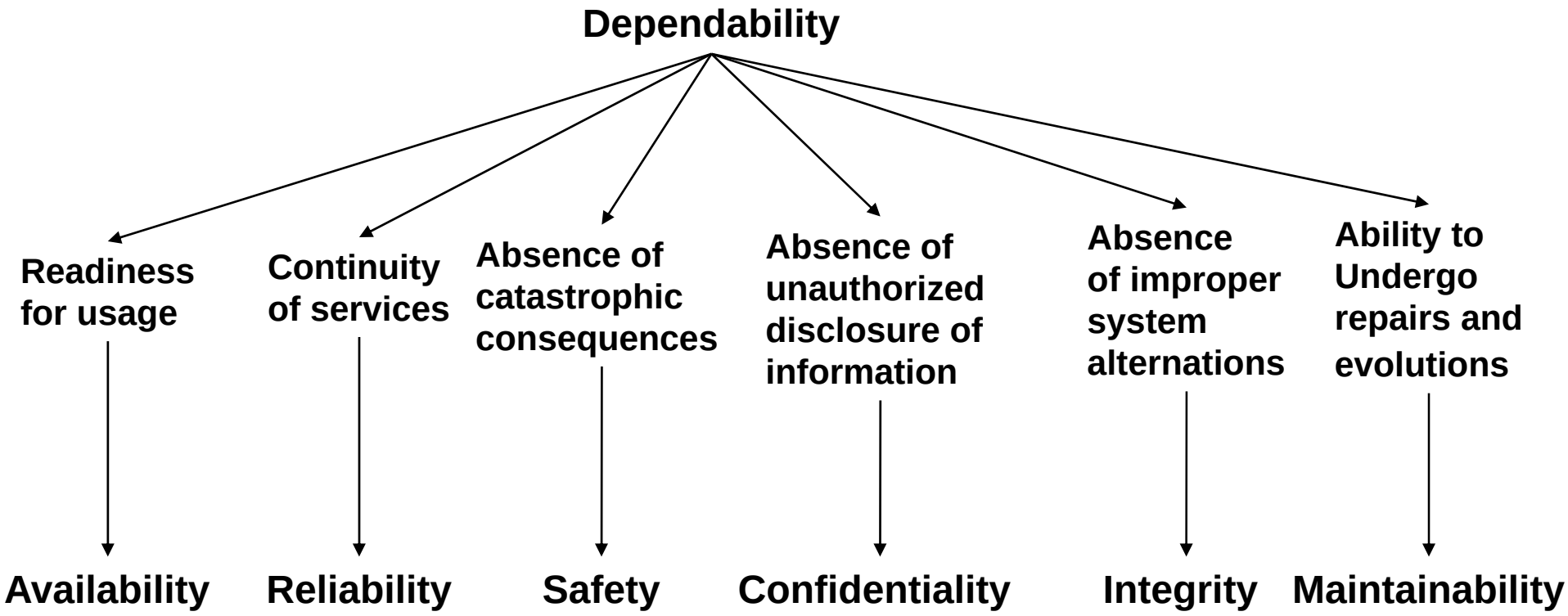
Safety is an attribute of a more general property: Dependability

Dependability

1. **Ability of a system to deliver service that can justifiably be trusted**

Related to

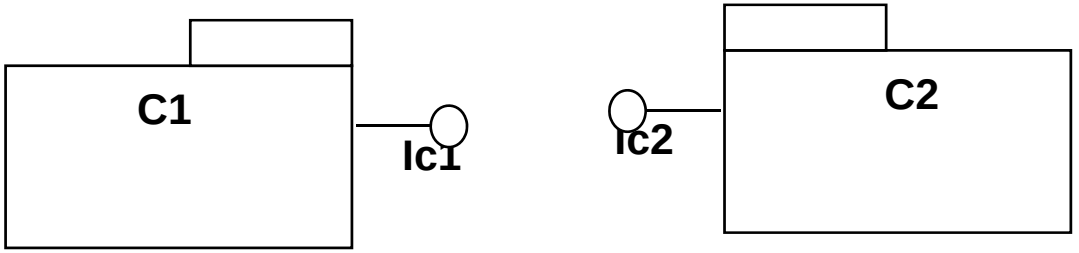
1. **Trustworthiness (assurance that a system will perform as expected)**



Attributes of Dependability

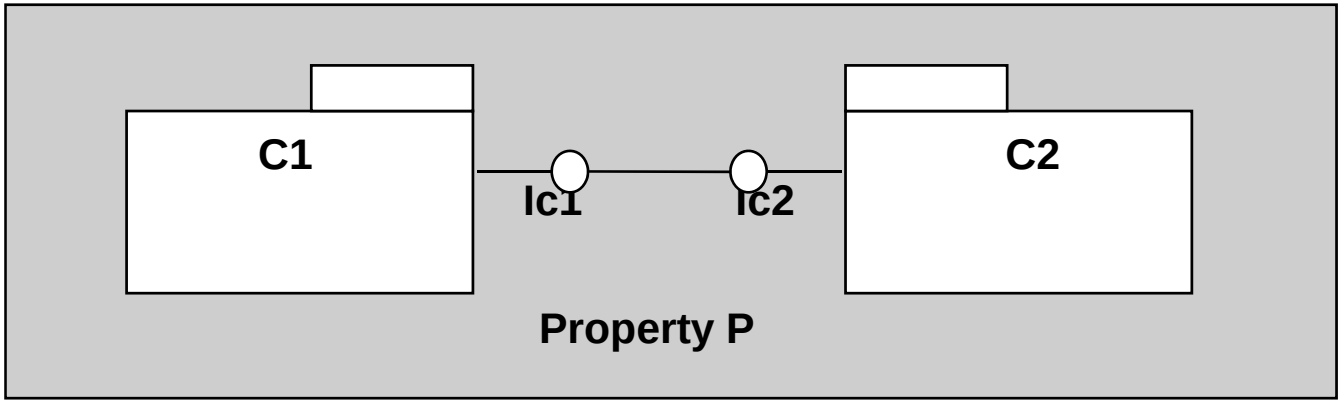
❑ Quality Attributes

The Challenges



Component C1
Property P1 (QA1)

Component C2
Property P2 (AQ2)



Is it possible (i.e. predict) to calculate P from P1 and P2?
Is it possible to predict a quality attribute of a system from quality attributes of components?

Is it possible (i.e. predict) to calculate P from P1 and P2?

Answer:

It depends of the quality attribute (property).

What if it is not possible?

We can test and measure P (hopefully)

But it can be time consuming

It can be very expensive

It can take more efforts to find P than (re)write the code

In some cases it is not worth to se component-based approach

The main question

❑Which quality attributes are composable?

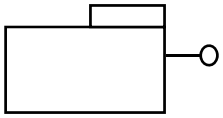
- What are the prerequisites for a predictable composition?
  - ☞The component properties themselves
  - ☞System architecture
  - ☞Particular usage profiles (how the components are used)?
  - ☞The system and the system's environment circumstances?

Idea

- **Classification of quality attributes according to their composability**
  - Related to the question
    - ☞When we develop a components what do we need to know about its usage in the future systems?
    - ☞When developed a system what do we need to know from the components, from the systems and from the system usage?

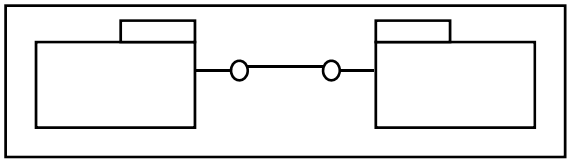
Some definitions

- **Component**



  - Specified by functional interface and a number of properties

- **Assembly**



  - A set of components integrated

- **System**

  - Application
    or a software system

- **Quality attributes vs. properties**
  - A component/system has a property that express a ceraing quality
  - Mostly: Quality attribute = property

Classification of properties

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 properties.* **A property of an assembly which depends on several different properties of the components.**

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

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

- **A)** ***A directly composable property of an assembly is a function of, and only of the same property of the components.***

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

$$A = \{c_i\}$$

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

Example: memory size

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

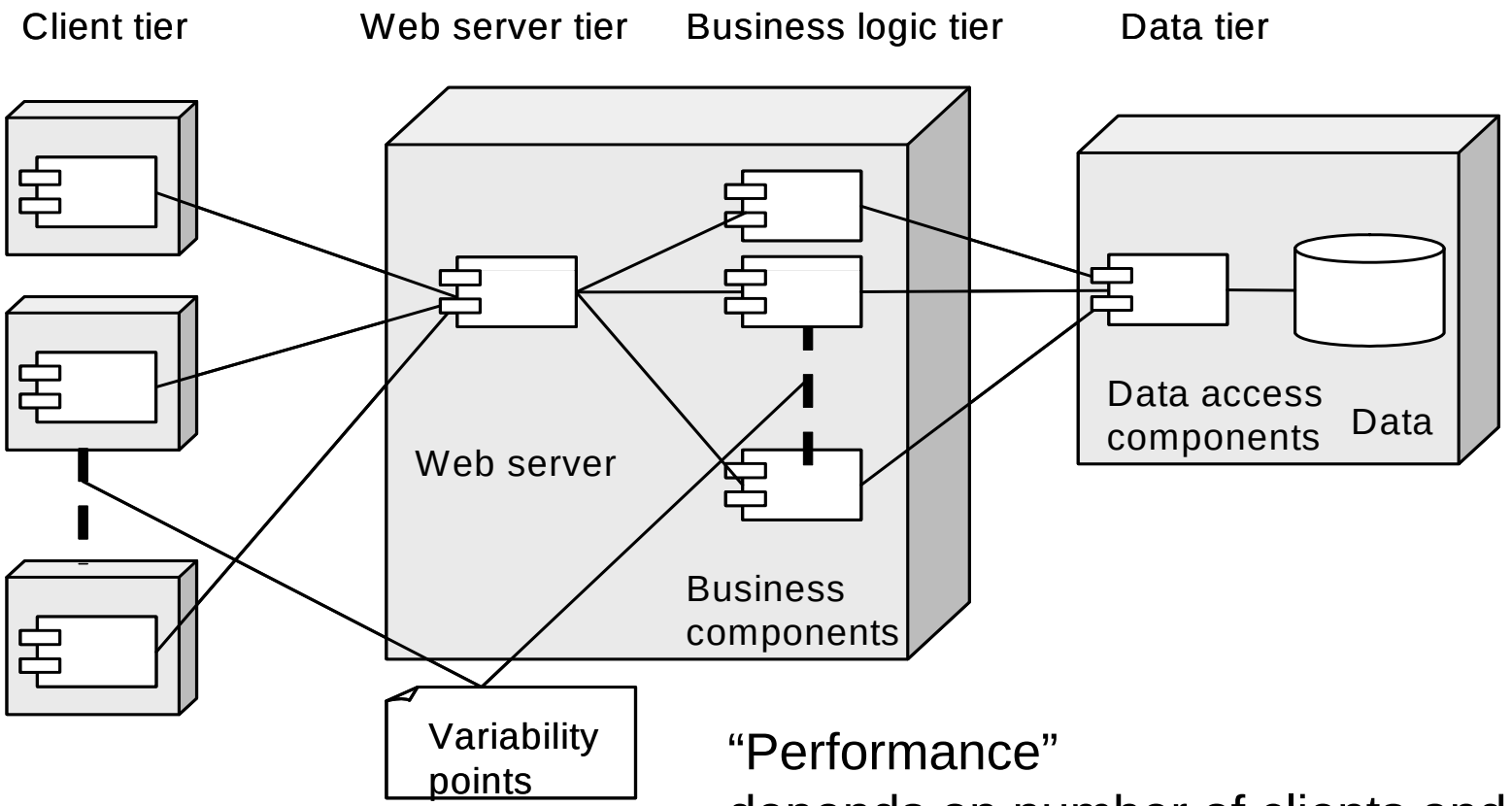
$$M = \text{memory size},\ A = \text{assembly},\ c_i = \text{components}$$

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

$$SA = \text{software architectu reture}, x_k = \text{connection s}$$

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

Architecture-based properties - example



"Performance"
depends on number of clients and concurrent server-components

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

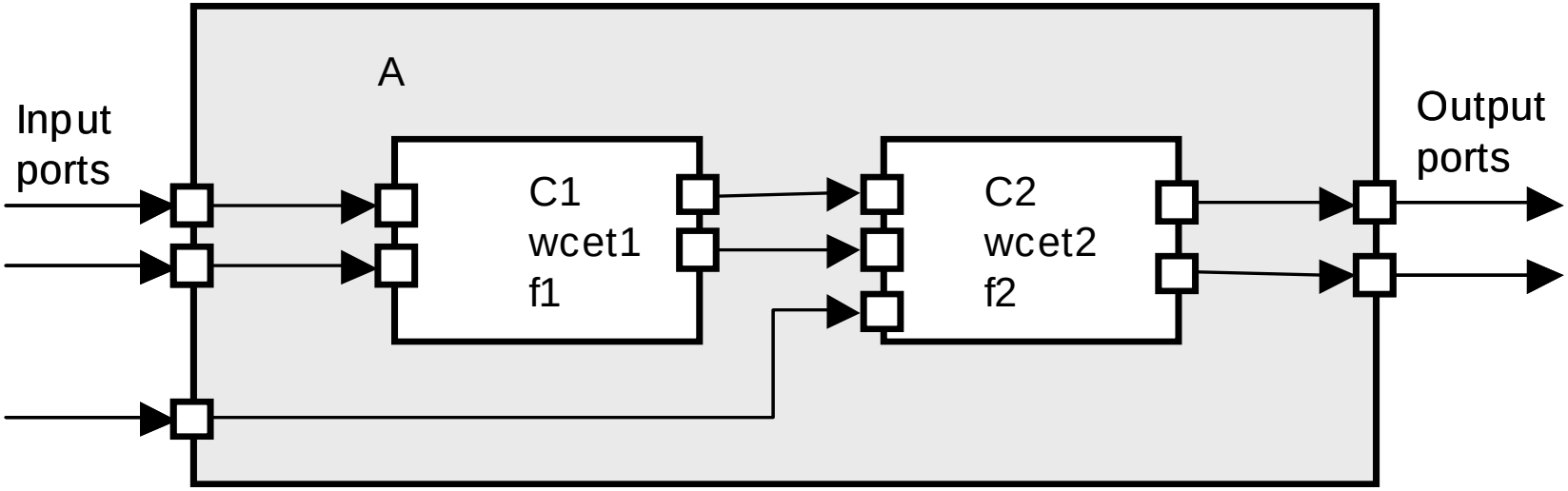
$$P(A) = f(P_1(c_i), P_2(c_i), \cdots, P_k(c_i));$$

$$i, k \in N$$

P = assembly property

$P_1 \ldots P_k$ = component properties

A derived property of an assembly example



Component: (A worse case) execution time
Assembly: Latency (response time)

Christer Notström et al…..

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

- **Usage profile – describes how do we use the system**
  - Used to calculate the reliability

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

P = property for a particular usage profile

U_k = assembly usage profile

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

- ***A System Environment Context property is a property*** **which is determined by other properties and by the state 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 = System usage profile;

E_l = Environmen t context

S = System

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

Example: Safety, security

Summary

- **There are properties it is possible to predict**
  - Directly from the **same properties of the components**
  - From **different properties of the components**
  - From the properties of the components **and the architecture of the system**
  - From properties of the components and the **Usage Profile of the system**
  - From properties of the components and the **Usage Profile of the system and the system environment**

Classification of QAs

Concerns	Quality Attribute	Directly compo-sable	Architecture-related	Derived	Usage-dependent	System environment context
Dependability	Availability		X	XX		
	Confidentiality			XX		
	Integrity			XX		
	Maintainability	X	X			
	Reliability		X		XX	
	Safety				X	XX
	Security				XX	X

Assignment

- Try to classify different properties
- The properties (quality attributes) are grouped in different “concerns”