A Classification Framework for Component Models

Ivica Crnkovic, Séverine Sentilles, Aneta Vulgarakis, Michel Chaudron

PROGRESS

A national Swedish Strategic Research Centre

MÄLARDALEN UNIVERSITY

MRTC
MÄLARDALEN REAL-TIME RESEARCH CENTRE

SCIENCE • TECHNOLOGY • MEDICINE • Swedish Foundation for Strategic Research

What is component?

- The component case
  - Many definitions
  - Some acknowledge ones:
    - *software component is a unit of composition with contractually specified interfaces and context dependencies only. A software component can be deployed independently and is subject to composition by third parties.*

Szyperski

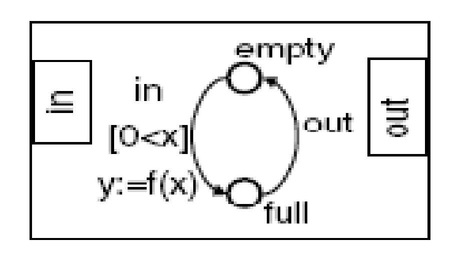
- *A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard*

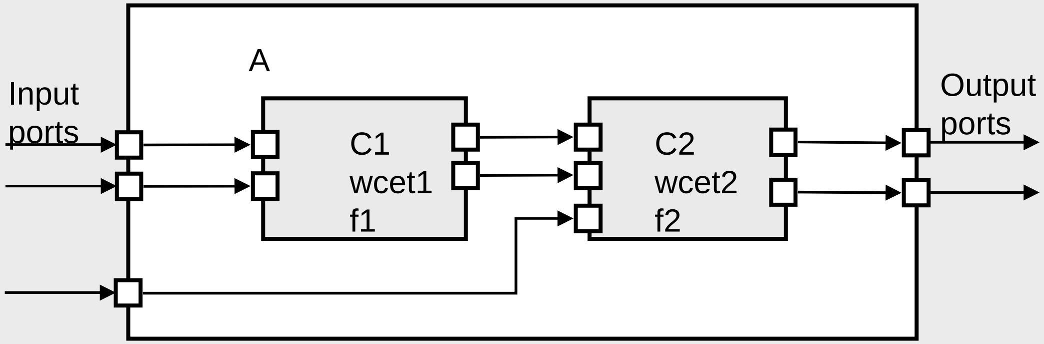
Heineman and Councill

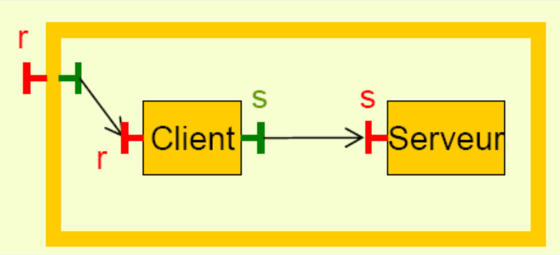
  - Intuitive perception may be quite different at different levels (model, implementation, run-time)

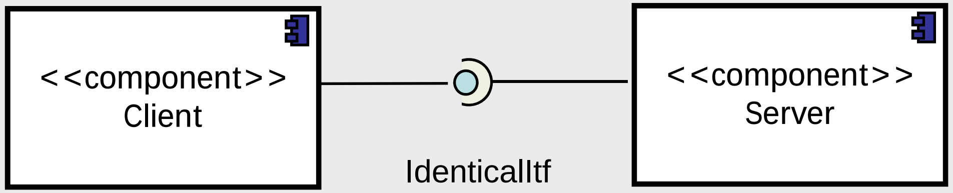
Different solutions

- A plethora of CB models (with many different characteristics)









AUTOSAR
BIP
COMDES
CCA
Corba CM
EJBFractal
KOALA
KobrA

MS COM
OpenCom
OSGi PIN
PECOS
ROBOCOP
RUBUS
SaveCCM
SOFA 2.0

Questions

- What is common to component models?

- It is possible to identify common principles and common features?

- Is it possible to utilize/instantiate these principles for particular component models in particular domains?

- Increase the understanding of the basic concepts of component models
- Easier differentiate component models according to several properties

Goal

- Propose a classification framework for component models
  - Defining categories
  - Grouping characteristics
  - Illustrating its use by providing a survey of a number of component models
  - (Analysis of the data resulting)

Definitions:
Software Component – Component Model

Definition:

- A *Software Component* is a software building block that conforms to a component model.
- A *Component Model* defines standards for
  - (i) properties that individual components must satisfy and
  - (ii) methods, and possibly mechanisms, for composing components.

Classification

- How to describe (i) Commonalities, (ii) Differences?
- Different approaches
  - Specification of Meta model
  - List of characteristics
  - Identification of categories and their characteristics

- **Component Specification**

  $C = <\{Interfaces\}, \{Properties\}>$

- **Component Composition:**

  $C = C_1 \oplus C_2$

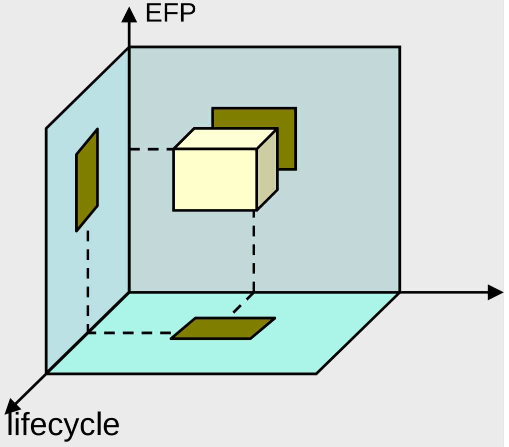
  Interaction (Interface composition): $I(C) = I(C_1) \oplus I(C_2)$

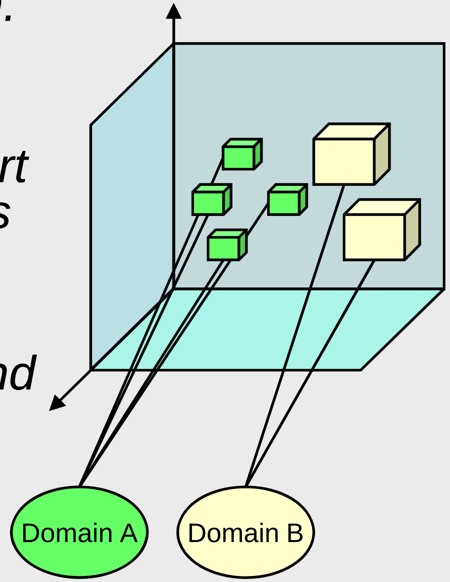
  Property composition: $P_i(C) = P_i(C_1) \oplus P_i(C_2)$

- **Component Lifecycle**

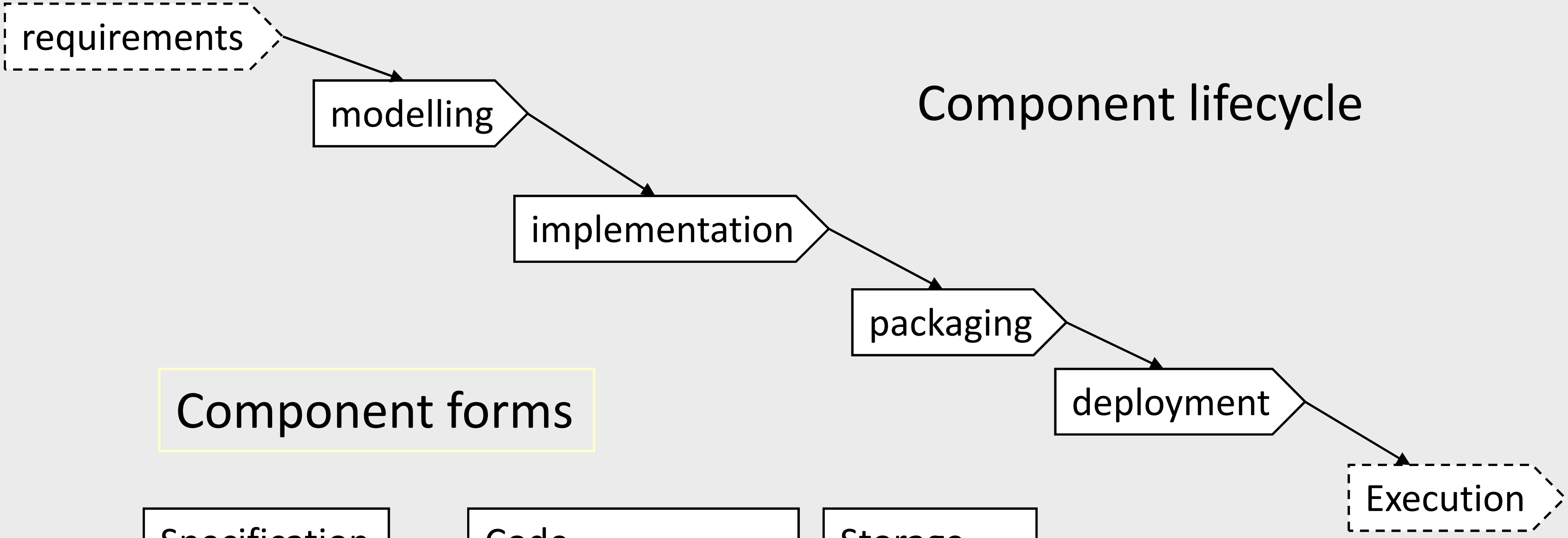
The Classification Framework - Categories

- **Lifecycle.** *The lifecycle dimension identifies the support provided (explicitly or implicitly) by the component model, in certain points of a lifecycle of components or component-based systems*.
- **Constructs**. *The constructs dimension identifies (i) the component interface used for the interaction with other components and external environment, and (ii) the means of component binding and communication.*
- **Extra-Functional Properties**. *The extra-functional properties dimension identifies specifications and support that includes the provision of property values and means for their composition*.
- **Domains**. *This dimension shows in which application and business domains component models are used.*





Component lifecycle



Component forms

Specification
- Interface
- Models
- Meta data

Code
- Source code
- Executable code
- Executable models

Storage
- Repository
- Package
- Meta data

Installed Files

Executable code

Lifecycle category

Different stages of a component lifecycle

- ***Modelling***. The component models provide support for the modelling and the design of component-based systems and components.

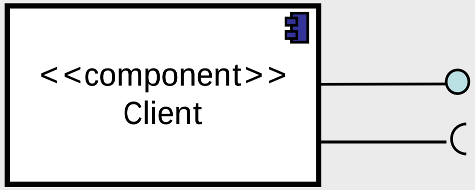
- ***Implementation****.* The component model provides support for generating and maintaining code. The implementation can stop with the provision of the source code, or can continue up to the generation of a binary (executable) code.

- ***Storage & Packaging****.* Since components can be developed separately from systems, there is a need for their storage and packaging – either for the repository or for a distribution

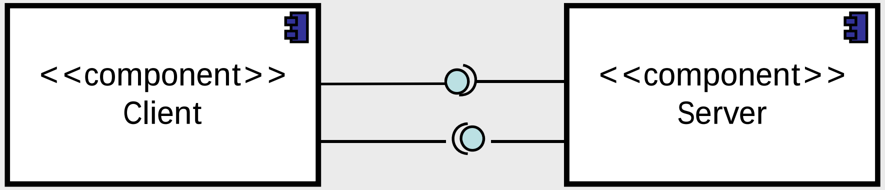
- ***Deployment & Execution****.* At a certain point of time, a component is integrated into a system. This activity happens at different points of development or maintenance phase.

Constructs

- Specification of
  - Interface



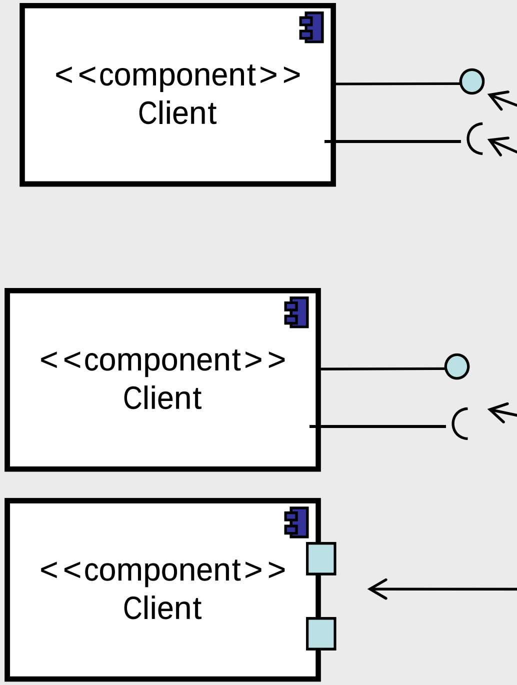
  - Composition (interaction)



Constructs – Interface Specification

Categories

- Levels
  - - Syntactic
    -Semantic
    - Behavioral
- Specification language
- Distinquish
  - Provide
  - Require
- Interface type
  - Operation-based
  - Port-based

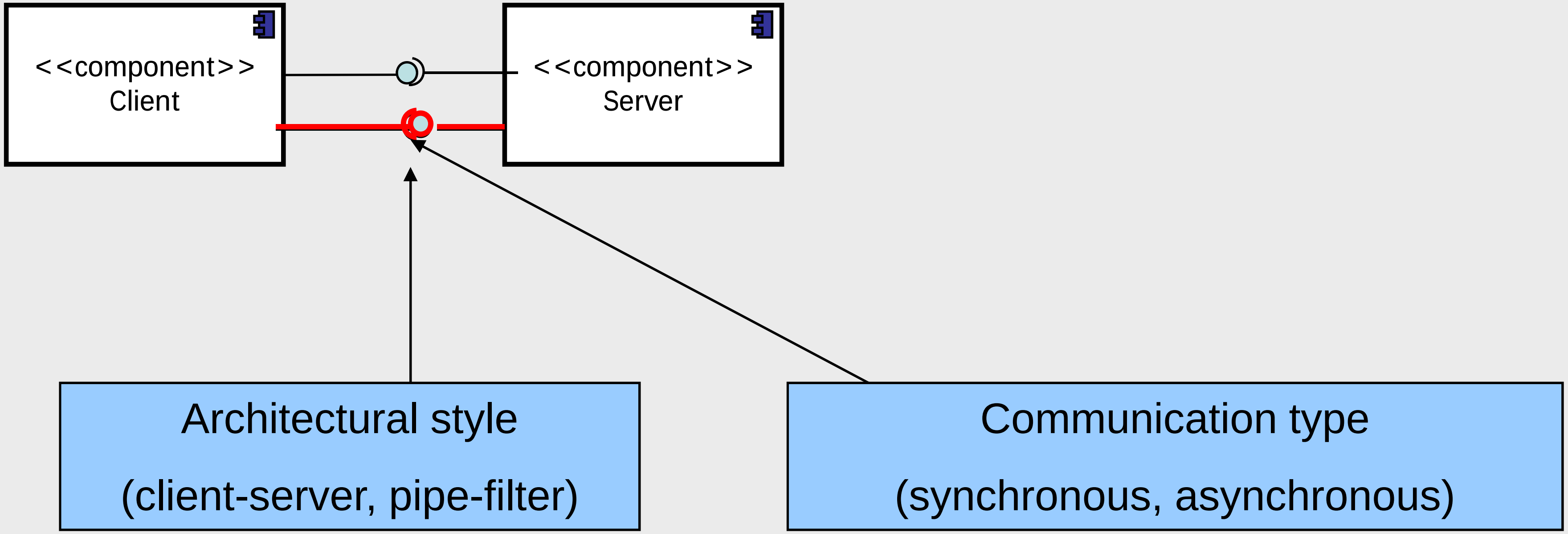


<<component>>
Client

<<component>>
Client

<<component>>
Client

Constructs – compositions (I)



<<component>>
Client

<<component>>
Server

Architectural style

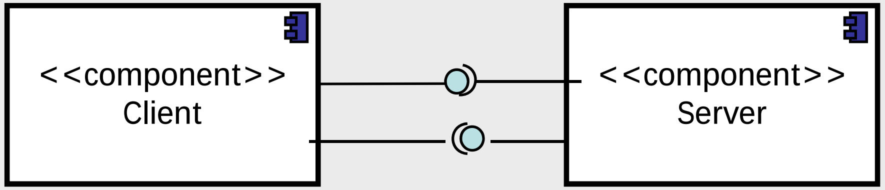
(client-server, pipe-filter)

Communication type

(synchronous, asynchronous)

Constructs compositions (II)

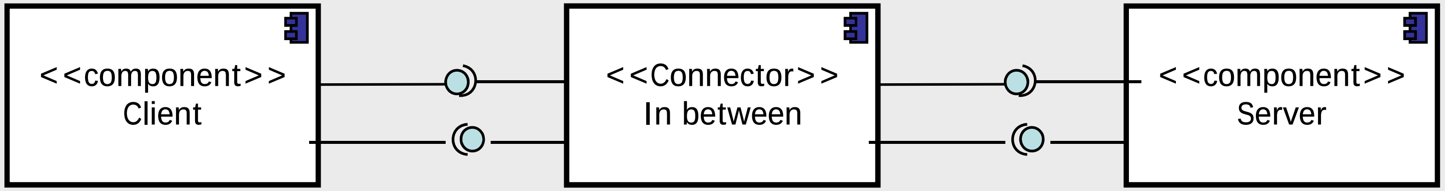
Endogenous



<<component>>
Client

<<component>>
Server

Exogenous



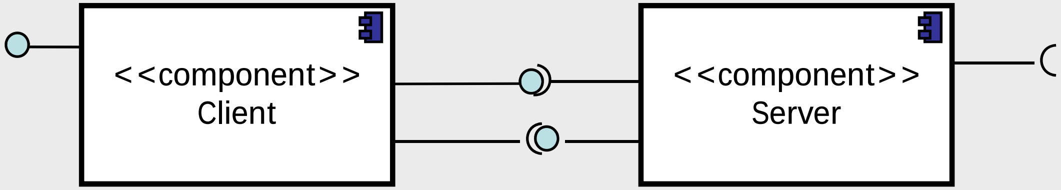
<<component>>
Client

<<Connector>>
In between

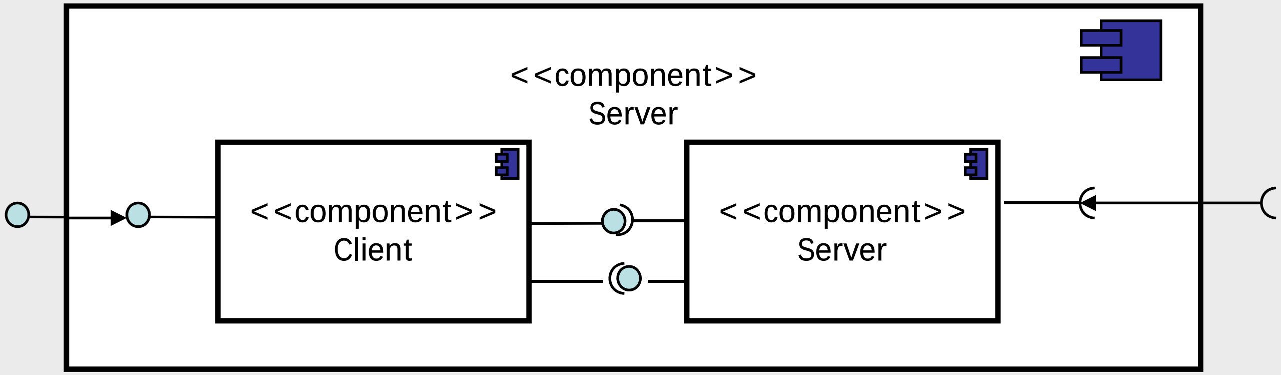
<<component>>
Server

Constructs compositions (III)

Composition



Horizontal



Vertical

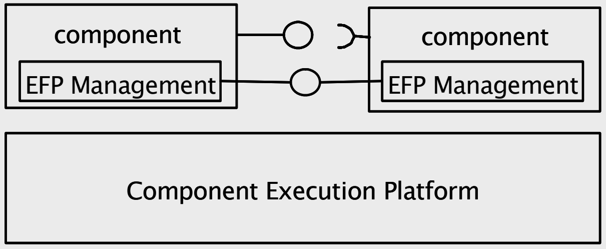
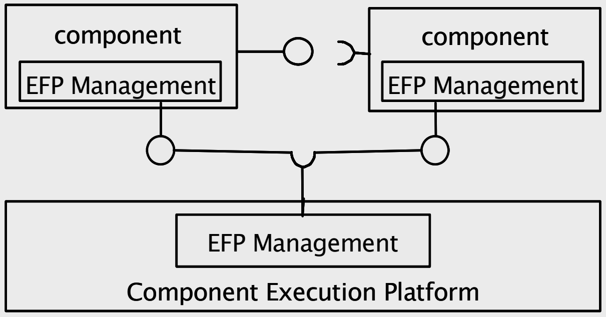
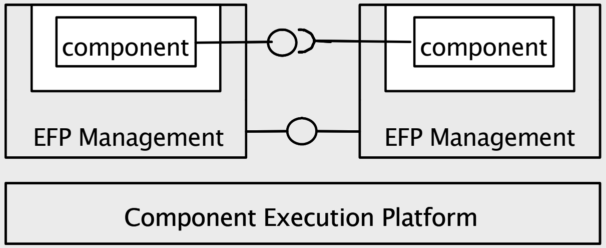
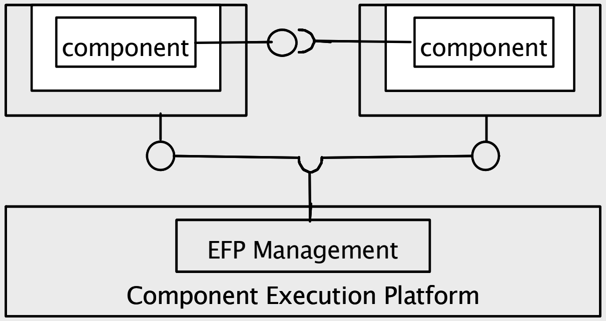
Constructs classification

- ***Interface***
  - operation-based/port-based
  - provides/requires
  - The interface level (syntactic, semantic, behaviour)
  - distinctive features
- ***Connections***
  - Architectural Style
  - Communication type (synchronous/asynchronous)
  - Binding type
    - Endogenous, Exogenous
    - Vertical, horisontal

Extra-Functional Properties

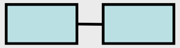
- Management of extra-functional properties
  - Does a component provide any support for extra-functional properties?
  - What are the mechanisms?
  - Which properties are managed?
- Composition of extra-functional properties
  - P(C1 o C2) = P(C1) o P(C2)
  - What kind of composition is supported?
  - Which properties?

Management of EFP

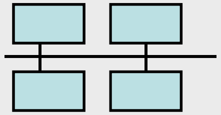
Endogenous EFP management	A  component EFP Management component EFP Management Component Execution Platform	B  component EFP Management component EFP Management EFP Management Component Execution Platform
Exogenous EFP management	C  component EFP Management component EFP Management Component Execution Platform	D  component component EFP Management Component Execution Platform
	EFP Managed per collaboration	EFP Managed systemwide

EPF – composition types (I)

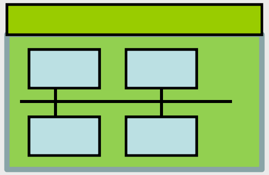
1. *Directly composable properties*.



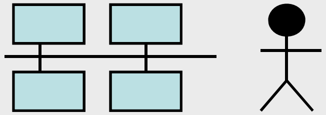
2. *Architecture-related properties*



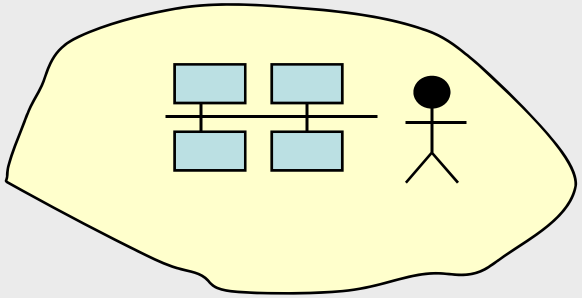
3. *Derived properties*.



4. *Usage-depended properties.*



5. *System environment context properties.*



EPF – composition types (II)

1. *Directly composable properties*. A property of an assembly is a function of, and only of, the same property of the components involved.
   - **P(A) = f(P(C1),…P(Ci),…,P(Cn))**

2. *Architecture-related properties.* A property of an assembly is a function of the same property of the components and of the software architecture.
   - **P(A) = f(SA, …P(Ci)…), i=1…n**
   - **SA = software architecture**

EPF – composition types (III)

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

- **P(A) = f(SA, …Pi(Cj)…), i=1…m, j=1…n**
- **Pi = component properties**
- **Cj = components**

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

- **P(A,U) = f(SA, …Pi(Cj,U)…), i=1…m, j=1…n**
- **U = Usage profile**

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

- **P(S,U,X) = f(SA, …Pi(Cj,U,X)…), i=1…m, j=1…n**
- **S= system, X = system context**

Domains

Applications and business domain of the Component Models

- ***General-purpose:***
  - Basic mechanisms for the production and the composition of components
  - Provide no guidance, nor support for any specific architecture.

- ***Specialised:***
  - Specific application domains
    (i.e. consumer electronics, automotive, …)

- ***Generative:***
  - Instantiation of particular component models
  - Provide common principles and some common parts of technologies (for example modelling)
  - Other parts are specific (for example different implementations)

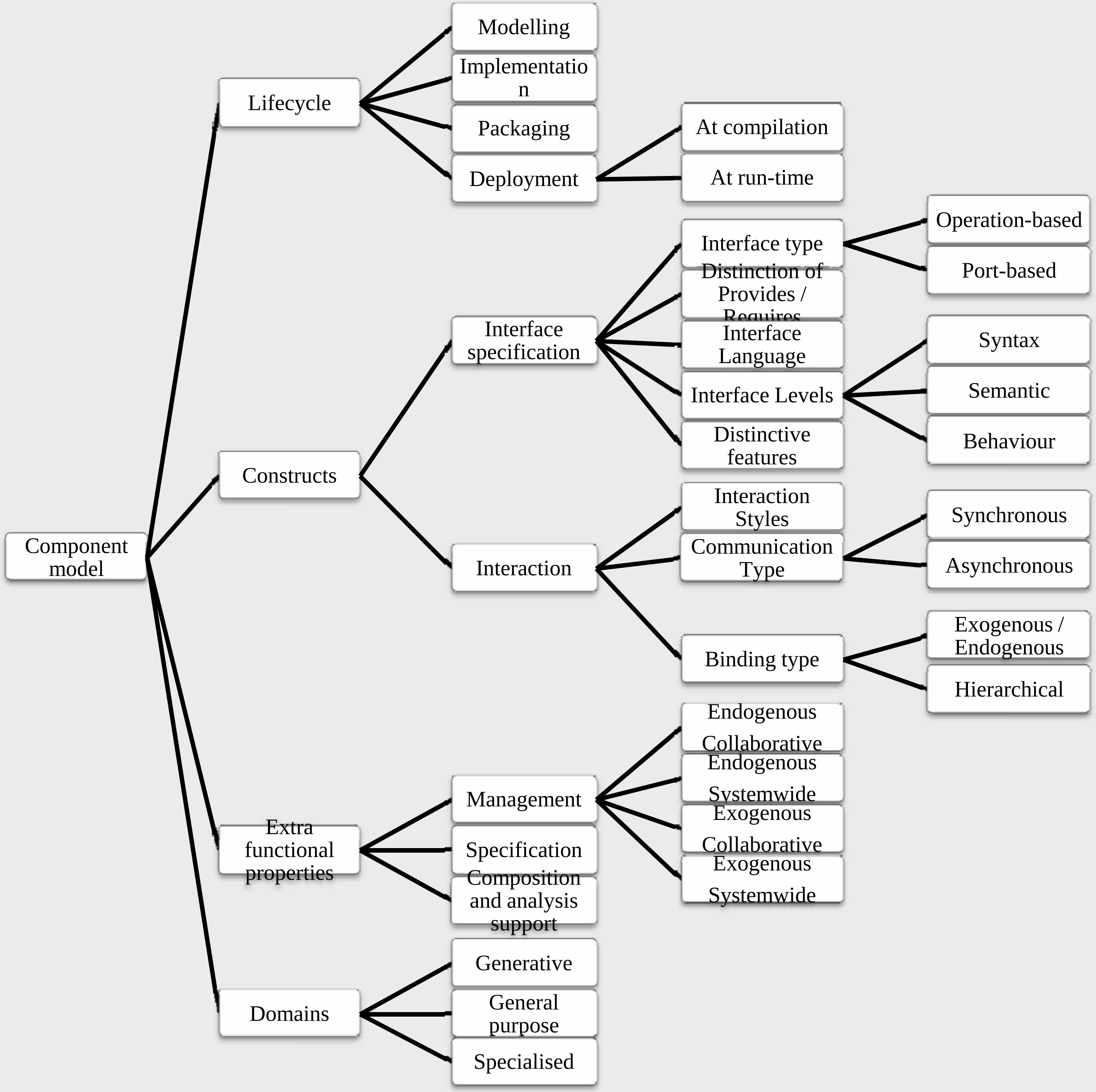

Component model
Lifecycle
Modelling
Implementatio n
Packaging
Deployment
At compilation
At run-time
Constructs
Interface specification
Interface type
Operation-based
Port-based
Distinction of Provides / Requires
Interface Language
Interface Levels
Syntax
Semantic
Behaviour
Distinctive features
Interaction
Interaction Styles
Communication Type
Synchronous
Asynchronous
Binding type
Exogenous / Endogenous
Hierarchical
Extra functional properties
Management
Endogenous Collaborative
Endogenous Systemwide
Exogenous Collaborative
Exogenous Systemwide
Specification
Composition and analysis support
Domains
Generative
General purpose
Specialised

Illustration of the Classification Framework use

- Survey of 20 component models
- Selection of documentation for each component model
  - Satisfies criteria
  - Disponibility the definition (Interfaces, composition)
  - Some points in the table have been subject our interpretation.

Chosen component models

- AUTOSAR
- BIP
- COMDES
- Common Component Architecture (CCA)
- CompoNETS
- CORBA Component Model (CCM)
- The Entreprise JavaBeans (EJB
- Fractal
- The K-Component Model
- KobrA
- Koala
- PIN
- Microsoft Component Object Model (COM)
- OpenCOM
- The Open Services Gateway Initiative (OSGi)
- Palladio
- Pin
- Robocop
- Rubus
- SaveCCM

Lifecycle table

Component Models	Modelling	Implementation	Packaging	Deployment
AUTOSAR	N/A	C	Non-formal specification of container	At compilation
BIP	A 3-layered representation: behavior, interaction, and priority	BIP Language	N/A	At compilation
BlueArX	N/A	C	N/A	At compilation
CCM	N/A	Language independent	Deployment Unit archive (JARs, DLLs)	At run-time
COMDES II	ADL-like language	C	N/A	At compilation
CompoNETS	Behavour modeling (Petri Nets)	Language independent	Deployment Unit archive (JARs, DLLs)	At run-time
EJB	N/A	Java	EJB-Jar files	At run-time
Fractal	ADL-like language (Fractal ADL, Fractal IDL), Annotations (Fractlet)	Java (in Julia, Aokell) C/C++ (in Think) .Net lang. (in FracNet)	File system based repository	At run-time
KOALA	ADL-like languages (IDL,CDL and DDL)	C	File system based repository	At compilation
KobrA	UML Profile	Language independent	N/A	N/A
IEC 61131	Function Block Diagram (FBD) Ladder Diagram (LD) Sequential Function Chart (SFC)	Structured Text (ST) Instruction List (IL)	N/A	At compilation
IEC 61499	Function Block Diagram (FBD)	Language independent	N/A	At compilation
JavaBeans	N/A	Java	Jar packages	At compilation
MS COM	N/A	OO languages	DLL	At compilation and at run-time
OpenCOM	N/A	OO languages	DLL	At run-time
OSGi	N/A	Java	Jar-files (bundles)	At run-time and at compilation
Palladio	UML profile	Java	N/A	At run-time
PECOS	ADL-like language (CoCo)	C++ and Java	Jar packages or DLL	At compilation
Pin	ADL-like language (CCL)	C	DLL	At compilation
ProCom	ADL-like language, timed automata	C	File system based repository	At compilation
ROBOCOP	ADL-like language, resource management model	C and C++	Structures in zip files	At compilation and at run-time
RUBUS	Rubus Design Language	C	File system based repository	At compilation
SaveCCM	ADL-like (SaveComp), timed automata	C	File system based repository	At compilation
SOFA 2.0	Meta-model based specification language	Java	Repository	At run-time

Lifecycle table

Component Models	Modelling	Implementation	Packaging	Deployment
AUTOSAR	N/A	C	N/A	At compilation
BIP	A 3-layered representation: behavior, interaction and priority	Source code, implementation in BIP language	N/A	At compilation
CCM	Abstract model:OMG-IDL, Programming model: CIDL	Language independent.	Deployment Unit archive (JARs, DLLs)	At run-time
Fractal	ADL-like language (Fractal ADL, Fractal IDL), Annotations (Fractlet)	Julia, Aokell(Java) Think(C/C++) FracNet(.Net)	File system based repository	At run-time
KOALA	ADL-like languages (IDL,CDL and DDL)	C	File system based repository	At compilation
EJB	N/A	Java binary code	EJB-Jar files	At run-time

Constructs table - Interface

Component Models	Interface type	Distinction of Provides / Requires	Distinctive features	Interface Language	Interface Levels (Syntactic, Semantic, Behaviour)
AUTOSAR	Operation-based Port-based	Yes	AUTOSAR Interface*	C header files	Syntactic
BIP	Port-based	No	Complete interfaces, Incomplete interfaces	BIP Language	Syntactic Semantic Behaviour
BlueArX	Port-based	Yes	N/A	C	Syntactic
CCM	Operation-based Port-based	Yes	Facets and receptacles Event sinks and event sources	CORBA IDL, CIDL	Syntactic
COMDES II	Port-based	Yes	N/A	C header files State charts diagrams	Syntactic Behaviour
CompoNETS	Operation-based Port-based	Yes	Facets and receptacles Event sinks and event sources	CORBA IDL, CIDL, Petri nets	Syntactic Behaviour
EJB	Operation-based	No	N/A	Java Programming Language + Annotations	Syntactic
Fractal	Operation-based	Yes	Component Interface, Control Interface	IDL, Fractal ADL, or Java or C, Behavioural Protocol	Syntactic Behaviour
KOALA	Operation-based	Yes	Diversity Interface, Optional Interface	IDL, CDL	Syntactic

Constructs table - interaction

COMPONENT MODELS	INTERACTION STYLES	COMMUNICATION TYPE	BINDING TYPE	
			EXOGENOUS	HIERARCHICAL
AUTOSAR	Request response, Messages passing	Synchronous, Asynchronous	No	Delegation
BIP	Triggering Rendez-vous, Broadcast	Synchronous, Asynchronous	No	Delegation
BlueArX	Pipe&filter	Synchronous	No	Delegation
CCM	Request response, Triggering	Synchronous, Asynchronous	No	No
COMDES II	Pipe&filter	Synchronous	No	No
CompoNETS	Request response	Synchronous, Asynchronous	No	No
EJB	Request response	Synchronous, Asynchronous	No	No
Fractal	Multiple interaction styles	Synchronous, Asynchronous	Yes	Delegation, Aggregation
KOALA	Request response	Synchronous	No	Delegation, Aggregation

EFP

Component Models	Management of EFP	Properties specification	Composition and analysis support
BlueArX	Endogenous per collaboration (A)	Resource usage, Timing properties	N/A
EJB 3.0	Exogenous system wide (D)	N/A	N/A
Fractal	Exogenous per collaboration (C)	Ability to add properties (by adding “property” controllers)	N/A
KOALA	Endogenous system wide (B)	Resource usage	Compile time checks of resources
KobrA	Endogenous per collaboration (A)	N/A	N/A
Palladio	Endogenous system wide (B)	Performance properties specification	Performance properties
PECOS	Endogenous system wide (B)	Timing properties, generic specification of other properties	N/A
Pin	Exogenous system wide (D)	Analytic interface, timing properties	Different EFP composition theories, example latency
ProCom	Endogenous system wide (B)	Timing and resources	Timing and resources at design and compile time
Robocop	Endogenous system wide (B)	Memory consumption, Timing properties, reliability, ability to add other properties	Memory consumption and timing properties at deployment
Rubus	Endogenous system wide (B)	Timing	Timing properties at design time
SaveCCM	Endogenous system wide (B)	Timing properties, generic specification of other properties	Timing properties at design time
SOFA 2.0	Endogenous system wide (B)	Behavioural (protocols)	Composition at design

Domains

Domain	AUTOSAR	BIP	BlueArX	CCM	COMDES II	CompoNETS	EJB	Fractal	KOALA	KobrA	IEC 61131	IEC 61499	JavaBeasns	MS COM	OpenCOM	OSGi	Palladio	PECOS	Pin
General-purpose				x		x	x	x		x			x	x	x		x		x
Specialised	x	x	x		x				x		x	x				x		x	
Generative								x											

Conclusion

- From the results we can recognize some recurrent patterns such as
  - general-purpose component models utilize client-server style
  - Specialized domains (mostly embedded systems) pipe & filter is the predominate style.
  - Composition of extra-functional properties is rather scarce.
  - Behaviour & Semantic rarely supported
  - Almost never repository
- Further work
  - Refinement of the model (detailed and more formalised classification)
  - Inclusion of additional component models
  - Analysis per domain
  - Pattern for specific groups of models