UML 2.0 Components and Fractal: An Analysis

Vladimir Mencl\textsuperscript{1,2} and Matej Polak\textsuperscript{2}

\textsuperscript{(1)}United Nations University
International Institute for Software Technology
UNU-IIST
http://www.iist.unu.edu/

\textsuperscript{(2)}Charles University, Prague
Distributed System Research Group
http://nenya.ms.mff.cuni.cz/
Outline

• Motivations & objectives
• UML 2.0 components: a gentle overview
  ▪ interfaces, ports, and connectors
  ▪ subcomponents
• Fractal components with UML 2.0
  ▪ evaluation and mapping
• Implementation
  ▪ modeling tool plug-in
• Conclusion
Motivations & Objectives

• Advanced component models exist
• UML 2.0 now has Components with
  ▪ provided and required interfaces
  ▪ hierarchy

**Note**: UML 1.x Components were deployment/packaging units only

• Objectives
  ▪ Analyze UML 2.0 — “does it fit the needs of advanced component models” ?
    • including “extra features” — collection interfaces, “collection subcomponents”....
  ▪ Propose a mapping for Fractal
UML 2.0 Overview: Key Features

- Component: now “as we now it”
  - hierarchy / nested components
  - provided and required interfaces

- Key concepts:
  - StructuredClassifier
    - functionality decomposed into parts
  - EncapsulatedClassifier
    - communication through Ports
  - Port
    - has provided and required Interfaces
    - has multiplicity (=> collection interfaces)
  - Component
    - combines these features
UML 2.0: Metamodel — relevant parts
Subcomponents

• Two ways to model subcomponents:

• Containment
  ▪ *Component* is a *Namespace*
  ▪ may own a *Component, InstanceSpecification, Class, Interface*
  ▪ owned Component is a type definition only
    => must be accompanied with an *InstanceSpecification*
  ▪ ... no multiplicities

• As *parts*
  ▪ *Component* is a *StructuredClassifier*
    • may own parts
      ▪ part: has type, multiplicity
Figure: Subcomponents, Connectors ...
Connectors ~ bindings in Fractal —

- connect provisions and requirements
  - precisely: a ConnectableElement
  - Port and Property are a ConnectableElement, Interface is not!
    - raised in UML2.0 FTF issues 7247-7251 ... postponed
- link to part (subComponent) via `partWithPort`

- Technical problems: connector can’t be linked to:
  - Interfaces
  - InstanceSpecification (nor the Ports/Interfaces of the subcomponent it represents)

- Two types of connectors
  - delegate — “vertical”)
  - assembly — “horizontal”

- can be mapped to Fractal bindings and SOFA (3 kinds)
Components: Attributes & Methods

- **Component** is a specialization of **Class**

- may have attributes and methods
  - attributes — configuration parameters (“attributes, properties”) of components
  - methods declarations:
    - operations directly provided in Component
    - concept used in Corba CCM
Components: Realization, Inheritance

Realization:

“How the component type will be implemented”

- implement directly (owned methods)
- realizing classifier
  - point to an implementing class

- a Component may inherit
  - from another Component
    - Component (type) inheritance
  - from a Class, Interface
    - exact meaning not given
    - method and attribute declarations — as if specified for the Component
    - method implementations — implicit realization
Summary

• abstractions match the needs of Fractal
  ▪ component types
  ▪ interfaces
  ▪ subcomponents
  ▪ bindings
  ▪ attributes
  ▪ component implementation

▪ It is possibly to have a Fractal mapping ...
  ... but it is necessary to propose one!
▪ Goal: cover all Fractal ADL constructs
Fractal Mapping

• Stereotype <<FractalComponent>>
  ▪ identify components designed for Fractal
  ▪ store tagged values

• Mapping specifics
  ▪ we consider Java implementations of Fractal
  ▪ and assume interface signatures and content descriptors are FQ Java class names
Fractal Mapping: Interfaces

- Interfaces: both options considered.
  - Direct Interfaces:
    - Automatically assigned unique names.
    - Mandatory interface with single cardinality.
  - Interfaces via Ports:
    - Only one interface per port.
    - Position of interface client/server.
    - Port multiplicity determines cardinality + contingency.
  - The UML Interface determining the type is mapped to a Java interface.
Fractal Mapping: Subcomponents

- Fractal specific feature: embedded subcomponent definition
  - Nested UML Component definition similar
    - but defines only component type
  - Only with InstanceSpecification has the desired meaning
    - **Mapping**: pair Component+InstanceSpecification
- Subcomponent (<component definition="..."
  - either as part or InstanceSpecification
- InstanceSpecification with path expression => shared component
- Connectors: both assembly and delegate map to bindings
Fractal Mapping (cont.)

• Attributes => component attributes
  - AttributeController interface generated as a part of mapping
    - type restrictions:
      UML must use primitive types only.

• Generalization =>
  - component inheritance
  - content class inherits from a base class + a number of interfaces

• Realization => content class selected, otherwise generated
Fractal Mapping: Behavior Specifications

• Initial simple approach:
  ▪ tagged value BehaviorProtocol

• A Component may own a Behavior

• Suitable metaclass OpaqueBehavior
  ▪ Attributes body and language can be mapped to new Fractal ADL Behavior element
  ▪ Can accommodate recent as well as earlier behavior specification mechanisms for Fractal
Fractal Mapping: Implementation

- Mapping implemented as a plug-in for Enterprise Architect
  - Generates Fractal ADL + Java source code
    - Java Interfaces, Attribute Controller, skeleton of content class.
  - Possible future extension:
    - Reverse engineer Fractal ADL
      ...and possibly also runtime Fractal representation
    - Additional Generator for Fractlet (Fractal implementation based on Java 1.5 annotations)
    - Generate initialization code instead of ADL
Evaluation: UML vs. Fractal

- Fractal covered
  - Fractal ADL can be modeled in UML
  - partly due to flexibility of UML
  - only missing piece: component arguments

- UML not covered
  - Not everything syntactically correct in UML
    - has a meaning in Fractal
    - is legal in Fractal
    - makes sense to map
  - Part of our mapping: a number of constraints for UML models to be compliant with the mapping

- Decisions need to be made in mappings
  - e.g., Corba CM Profile
Other Component Models: SOFA

• Two levels of component specification
  ▪ frame, architecture
  => two stereotypes
    • <<SOFAFrame>>, <<SOFAArchitecture>>
  ▪ architecture linked to frame via a <<realize>>
    dependency
  ▪ only an architecture may contain subcomponents, and their type must be a frame

• Other minor differences
  ▪ SOFA allows constant definitions in frames
    • mapped as readOnly attributes with an initialization value
  ▪ Multiplicities on Ports mapped as arrays of Interfaces
  ▪ Behavior: Only Behavior Protocols supported in SOFA
Conclusion

• UML 2.0: a lot is underspecified or unspecified

• Some flexibility intentional
  ▪ Left up to profile or tool developers.
    • e.g., meaning of “A Component inherits from a Class”.

• Some issues not handled — metamodel does not permit some needed constructs.
  ▪ e.g., link a **Connector** to ports of a subcomponent specified as an **InstanceSpecification**
  ▪ Tools use proprietary metamodel modifications.
    “Hacks” => Negative impact on model interchangeability

• UML very rich, a selection of constructs mapped.
Future work:

• Implementation
  ▪ Various extensions possible.
• Propose fixes for UML
  ▪ new model element ComponentInstance
• Look at additional component models.

• References