UML 2.0 Components

Matej Polák
Goals

- analyze component model provided by UML 2.0
- demonstrate UML 2.0 component model capabilities on Sofa and Fractal component languages
- discuss support of UML 2.0 in various UML modelling tools
- write a plugin that generates Sofa and Fractal source code
1. view: UML = “nice diagrams”

2. view: background
   - 4 layer metamodel
   - class hierarchy
   - packages
     • kernel
     • components
     • ...
Composite structures

• internal structures
  – structure of elements within an instance of a classifier

• ports
  – communication point of a classifier

• collaborations
  – explains how a system works
Composite structures

• structured classes
  – class may have a port and an internal structure

• Example diagram:
  – menu
  – toolbars
  – main window
Components

- **component**
  - modular part of a system

- **connector**
  - link between two or more components

- **realization**
  - defines a Classifier that implements a Component
Components - diagrams

- component with a provided and a required interface
- component with port
- realization
Component provisions & requirements

1. direct provided & required interfaces
   - not named provisions/requirements

2. provided & required interfaces via ports
   - named provisions/requirements
Port features

- type
- multiple interfaces
- multiplicity
- isService
  - optional
  - mandatory
  
  ![Component diagram]

  no exact meaning
Component attributes & methods

- component inherits from a class
  ⇒ methods
  ⇒ attributes
  ⇒ generalization

- example
Subcomponents

1. as a part (from StructuredClassifier)
   - multiplicity
   - reference

2. PackageableElement
   - instance of a classifier
   - type, class, component, interface
   - package
Connectors

• enables communication between instances
• between interfaces, ports or mixed
• one to many, many to one
• the target must be a *signature compatible subset* of operations of the source
Connector kinds

- **assembly:**
  - required $\rightarrow$ provided interface/port
- **delegate:**
  - required $\rightarrow$ required
  - provided $\rightarrow$ provided
Component realization

• directly by a component itself
  – using component methods
  – component must have an *Implementation* relationship to the realizing interface

• by its realizing classifiers
  – via Realization Dependency
Component inheritance

- multiple inheritance
- base
  - class
  - component
  - interface
- required & provided interfaces and ports are be inherited
UML Profiles

- extension mechanism to:
  - add semantics
  - different notation for existing elements
  - add constrains
  - modify basic data types

- uses
  - stereotypes
  - tagged values
Sofa

• problems
  1. two kinds of components (Frame, Architecture)
  2. subcomponents and properties
  3. named interfaces
  4. three kinds of connectors (assembly, delegate, subsume)
  5. inheritance
  6. realization of a primitive architecture and behavior protocol

• solution:
  – UML profile
Sofa – 1\textsuperscript{st} problem

- two kinds of components (Frame, Architecture)
- solution
  1. extend UML model
  2. UML Profiles: using \texttt{<<SOFAArchitecture>>} and \texttt{<<SOFAFrame>>} stereotypes
     - tagged value
       - system (True/False)
     - Realization between Frame and Architecture
Sofa – 2\textsuperscript{nd} problem

- subcomponents
  - component instance of a Frame
  - property with a Frame classifier
    - multiplicity

- properties = attributes
- methods are ignored
Sofa – 3rd problem

- named interface
- solutions
  1. direct interfaces with dynamically generated names
  2. named ports
     - one interface per port
     - no type
Sofa – 4\textsuperscript{th} problem

- three kinds of connectors
- solution
  - assembly
  - subsume
  - delegate

= delegate
Sofa – 5th problem

- inheritance
- solutions
  1. extend Sofa to support inheritance (Oplušil: Inheritance in ADL)
  2. solve inheritance at a source code generation level
Sofa – 6th problem

• realization of primitive architecture
  1. via realizing classifier
  2. automatically generated

• behavior protocol
  1. Port state machines (Mencl: Specifying Component Behavior with Port State Machines)
  2. via tagged value SOFABehaviorProtocol
Fractal

- problems
  1. named interfaces, cardinality & contingency of interfaces
  2. subcomponents, shared subcomponents, embedded definition
  3. realization and behavior protocol

- solution:
  - UML profile
Fractal – 1\textsuperscript{st} problem

- named interfaces
  1. direct interfaces with dynamically generated names
  2. named ports
- cardinality
  - multiplicity of a port
- contingency
  - isService attribute of a port
Fractal – 2\textsuperscript{nd} problem

- subcomponents
  - component instance
  - part with type of a component
- shared subcomponents
  - part reference
- embedded definitions
  - only a component
Fractal - 3rd problem

• realization
  – via realizing classifier

• behavior protocol
  1. Port state machines (Mencl: Specifying Component Behavior with Port State Machines)
  2. via tagged value FractalBehaviorProtocol
Example
module BankDemo{
    frame Supervisor {
        provides:
            ISupervisorAccess supervisor;
        requires:
            IDatastoreAccess ds;
    };
    frame DataStore {
        provides:
            IDatastoreAccess ds;
    };
    frame Teller {
        provides:
            ITellerInterface teller;
        requires:
            IDatastoreAccess ds;
    };
    frame Bank {
        property long NoT;
        provides:
            ITellerInterface tellers[1..NoT];
        ISupervisorAccess supervisor;
    };
    architecture ::BankDemo::ASupervisor implements ::BankDemo::Supervisor primitive;
    architecture ::BankDemo::ADataStore implements ::BankDemo::DataStore primitive;
    architecture ::BankDemo::ATeller implements ::BankDemo::Teller primitive;
    architecture ::BankDemo::ABank implements ::BankDemo::Bank
    {
        property long NoT;
        inst ::BankDemo::Teller t[1..NoT];
        inst ::BankDemo::Supervisor s;
        inst ::BankDemo::DataStore ds;
        delegate tellers[1..NoT] to t:teller;
        delegate supervisor to s:supervisor;
        bind t:ds to ds:ds;
        bind s:ds to ds:ds;
    }
}
Conclusion

• UML 2.0
  ± huge, universal specification
  – sometimes ambiguous, inconsistencies
  + extensible

• Component model
  ± very lax
  – sometimes ambiguous
Conclusion

• Common (Sofa & Fractal)
  – UML component model fits well, some restrictions and functionalities had to be added
    • parts & instances as a subcomponents
    • ports as named interfaces or generated names
    • attributes = properties
    • behavior protocol

• Sofa
  – added two kinds of components: Frame and Architecture
  – added inheritance to a Sofa model

• Fractal
  – support of cardinality & contingency
  – shared subcomponents and embedded definition
Thanks for your attention