An Agile MDA Approach for Service-Oriented Components

S. Motogna, I. Lazăr, B. Pârv, I. Czibula

Department of Computer Science
Babeș Bolyai University
Cluj-Napoca, Romania
ComDeValCo

- **Component Definition, Validation and Composition** framework

- Includes:
  - Action language
    - UML structured activities
    - Graphical notation
  - Dynamic execution environment: iComponent
The context

Agile MDA approach:
• MDA: OMG, 2003
  – PIM, PSM, transformations
  – heavy-weight processes
• Agile – Mellor, 2005
  – light-weight process
  – test-first
  – immediate execution
• Executable UML
  – Non-standard action languages: Mellor&Balcer, 2002;

Service-orientation:
Approach:

1. Decompose \(\Rightarrow\) a collection of interacting services.
2. Define components implementing application services.
3. Define composite components for execution - as service specifications

Platform dependent:
• SCA, 2007; iPOJO, 2008;
Platform independent:
• iComponent, 2008
Our approach

• Any UML case tool can be used to construct the models
• conform to fUML specification for executable models
  ➢ consists of applying the following steps in the specified order:
  1. the model is described on different layers:
  2. for simple components proceed with test-first component development.

S. Motogna et al. - Agile MDA...
Model description layers

Services

Structure

Deployment

S. Motogna et al. - Agile MDA...
Service Model

- Defined by the system analyst
- Describes the services provided by the system.
- Corresponding modules may include any data type:
  - classes,
  - interfaces, or
  - components.
Structural Model

• typically defined by the system architect,

• indicates component instances that will implement the services ⇒ decomposition in simple and composite components.

• the internal structure of a component uses instances of other components, connected through ports;

• Specify the provided and required ports

• To select a certain service implementation satisfying some criteria – attributes:
  • **property** – provides -to export service properties
  • **filter** – requires – to filter required services
  • InstanceSpecification objects indicate which components should be created

S. Motogna et al. - Agile MDA...
Structural Model and component instances
Deployment Model – Modules dependencies
Deployment Model – Domain and nodes

S. Motogna et al. - Agile MDA...
Deployment Model – Component instances

S. Motogna et al. - Agile MDA...
Agile MDA process

1. Add a test
2. Run the test
3. Add production code
4. Run the test
if (productCode = self.productCode)
    return (1-discountPercentage/100)* price;
else
    return price;

assert 90 = pricingStrategy.adjustPrice("3",100);
assert 100 = pricingStrategy.adjustPrice("1",100);
## Related work

<table>
<thead>
<tr>
<th>iComponent</th>
<th>iPOJO</th>
<th>SCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>-</td>
<td>Domain</td>
</tr>
<tr>
<td>Node</td>
<td>-</td>
<td>Node</td>
</tr>
<tr>
<td>DynamicExecutionEnvironment</td>
<td>OSGi implementation</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>Bundle</td>
<td>Contribution</td>
</tr>
<tr>
<td>Component</td>
<td>Component</td>
<td>Component</td>
</tr>
<tr>
<td>Composite component</td>
<td>Composite</td>
<td>Composite</td>
</tr>
<tr>
<td>provides</td>
<td>provides</td>
<td>Service</td>
</tr>
<tr>
<td>requires</td>
<td>requires</td>
<td>Reference</td>
</tr>
<tr>
<td>validate and invalidate</td>
<td>validate and invalidate</td>
<td>-</td>
</tr>
<tr>
<td>controller</td>
<td>controller</td>
<td>-</td>
</tr>
<tr>
<td>config</td>
<td>Property</td>
<td>Property</td>
</tr>
</tbody>
</table>

S. Motogna et al. - Agile MDA...
Conclusions

• Agile MDA development approach for the development of service-oriented components.
• Distributed service architecture
• Separation between business logic and non-functional aspects
• Platform independence
• Rapid development - graphical