Dedal-CDL: Modeling First-class Architectural Changes in Dedal

Huaxi (Yulin) Zhang, Christelle Urtado, Sylvain Vauttier, Lei Zhang, Marianne Huchard and Bernard Coulette

1 Dept. Math / Info, Université Toulouse 2, Toulouse, France
2 LGI2P / Ecole des Mines d’Alès, Nîmes, France
3 LIRMM, UMR 5506, CNRS and Univ. Montpellier 2, Montpellier, France
4 Research Center of Automation, Northeastern University, Shenyang, China
Context

Architecture-centric component-based development needs a three-level ADL
Huaxi (Yulin) Zhang, Christelle Urtado, and Sylvain Vauttier
@ ECSA 2010

- Component-based software design by reuse
- The three level Dedal ADL

Motivation : architecture evolution management

Modeling changes

Versioning architectures

Comparison to state-of-the-art ADLs

Conclusion and perspectives
Component-based software design by reuse

The Dedal three levels ADL
The three levels for component description in Dedal

**Specification**
(abstract & partial types)

**Configuration**
(concrete classes)

**Assembly**
(specific instances)
Insights on Dedal’s syntax

component_role Session
required_interfaces BikeOprs; CourseOprs; AccountOprs
provided_interfaces Account; Bike
component_behavior . . .

component_type BasketType
required_interfaces BikeOprs; CourseOprs; AccountOprs; CampusOprs; AccessoryOprs
provided_interfaces Account; Bike
component_behavior . . .

component_class Basket
implements BasketType
content fr.ema.locaBike.Basket
attributes string company; string currency
versionID 1.0
Insights on Dedal’s syntax

**component_instance** BasketLocaBike

**instance_of** Basket (1.0)

**initiation_state**
- company="LocaBike"
- concurrency="euro"

**component_class** Basket

**implements** BasketType

**content** fr.ema.locabike.Basket

**attributes**
- string company
- string currency

**versionID** 1.0

**component_role** Session

**required_interfaces**
- BikeOprs
- CourseOprs
- AccountOprs

**provided_interfaces**
- Account
- Bike

**component_behavior** ...
Motivation: architecture evolution management 1/2

![Architecture Evolution Diagram](image-url)
Changes can be initiated elsewhere.

Avoid mismatches: architecture pendency, architecture erosion and architecture drift.

Change impact must be propagated appropriately to maintain all architecture descriptions and the software coherent at each time (co-evolution).

Versioning should be proposed to track architectural changes and soundly integrated with change management.

Proposal: model changes as first class elements.
## Modeling changes

### Change characteristics and a Dedal-CDL change description example

<table>
<thead>
<tr>
<th>Change Characteristics</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>time of change</td>
<td>static, dynamic</td>
</tr>
<tr>
<td>anticipation</td>
<td>anticipated, unanticipated</td>
</tr>
<tr>
<td>affected level</td>
<td>specification, configuration, assembly</td>
</tr>
<tr>
<td>change operation</td>
<td>addition, removal, substitution, modification</td>
</tr>
<tr>
<td>subject of change</td>
<td>elements of architectures in three levels</td>
</tr>
<tr>
<td>change purpose</td>
<td>corrective, perfective, adaptive</td>
</tr>
<tr>
<td>nature of change</td>
<td>given, generated, propagated</td>
</tr>
</tbody>
</table>

```
change additionStationData
time dynamic
anticipation unanticipated
level configuration
operation addition
subject component_class is StationData
purpose perfective
nature given
```
Versioning architecture descriptions

**Captions:**
+ : add  — : remove  × : replace  ¶ : modify
→ : derivation

Diagram showing versioning relationships between BRSSpec (1.0) and BRSSpec (2.0), BRSCConfig (1.0) and BRSCConfig (2.0), and BRSAss (1.0) and BRSAss (2.0). Relationships marked with `<<realizes>>`, `<<instantiates>>`, and `architectureBehavior`. Additional relationships marked with `+GIS`, `+connection3`, and `+StationData`, `+StationDataC1`.
<table>
<thead>
<tr>
<th>Characteristics of change</th>
<th>C2</th>
<th>Darwin</th>
<th>Dynamic Wright</th>
<th>SOFA2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of change</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Anticipation</td>
<td>Unanticipated</td>
<td>Anticipated, Unanticipated</td>
<td>Anticipated</td>
<td>Anticipated</td>
</tr>
<tr>
<td>Change type</td>
<td>Structural</td>
<td>Structural</td>
<td>Structural</td>
<td>Structural</td>
</tr>
<tr>
<td>Change purpose</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Level of change</td>
<td>Configuration</td>
<td>Configuration</td>
<td>Configuration</td>
<td>Configuration</td>
</tr>
<tr>
<td>Change operation</td>
<td>Addition, removal</td>
<td>Addition, removal</td>
<td>Addition, removal</td>
<td>Addition, removal</td>
</tr>
<tr>
<td>Subject of change</td>
<td>Components, connectors, connections</td>
<td>Components, connections</td>
<td>Components, connections</td>
<td>Components, connections, interfaces of composite components</td>
</tr>
<tr>
<td>Version model</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>State-based</td>
</tr>
<tr>
<td>Characteristics of change</td>
<td>xADL2.0</td>
<td>MAE</td>
<td>Dedal</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>------------------------------</td>
<td></td>
</tr>
<tr>
<td>Time of change</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td>Dynamic</td>
<td></td>
</tr>
<tr>
<td>Anticipation</td>
<td>Unanticipated</td>
<td>Unanticipated</td>
<td>Anticipated, Unanticipated</td>
<td></td>
</tr>
<tr>
<td>Change type</td>
<td>Structural</td>
<td>Semantical</td>
<td>Structural, Semantical</td>
<td></td>
</tr>
<tr>
<td>Change purpose</td>
<td>—</td>
<td>Perfective</td>
<td>Perfective, corrective</td>
<td></td>
</tr>
<tr>
<td>Level of change</td>
<td>Configuration</td>
<td>Configuration</td>
<td>Specification, configuration, assembly</td>
<td></td>
</tr>
<tr>
<td>Change operation</td>
<td>Addition, removal</td>
<td>Substitution</td>
<td>Addition, removal, substitution</td>
<td></td>
</tr>
<tr>
<td>Subject of change</td>
<td>Components, connectors, connections</td>
<td>Components</td>
<td>Components, connectors in three levels</td>
<td></td>
</tr>
<tr>
<td>Version model</td>
<td>—</td>
<td>Change-based</td>
<td>Change-based</td>
<td></td>
</tr>
</tbody>
</table>
**Conclusion: contribution**

**Contribution**

- **Changes are first class**
  - described by the characteristics we found in state-of-the-art works

- **Architecture description / running software co-evolution principle is described**
  - consistency checking, change propagation, evolution test, versioning

- **Software prototype: (small) ad-hoc proof of concept**
  - over the Fractal component model
Conclusion: perspectives

- **Further investigations**
  - Produce the change information (automatically / manually)
    - learn from / reuse previous changes
  - Use the change information to adequately / rigorously manage change
    - formalization of relations among architecture descriptions (various semantics can be explored)
    - formalization of coherence maintenance rules (no more ad hoc implementation but transformations)

- **Need for an improved state of the art**
  - deeper / more recent

- **Develop an eclipse plugin**
  - Integration of the architectural change management tool to the system development tool for an improved control
Dedal-CDL: Modeling First-class Architectural Changes in Dedal

Huaxi (Yulin) Zhang, Christelle Urtado, Sylvain Vauttier, Lei Zhang, Marianne Huchard and Bernard Coulette

We welcome any suggestion.