Comparison of six ways to extend the scope of Cheddar to AADL v2 with Osate

Mickaël Kerboeuf, Alain Plantec, Frank Singhoff
LISyC - University of Brest

Arnaud Schach, Pierre Dissaux
Ellidiss Technologies
Outline

- Motivation
  - Integration of legacy systems

- Principles
  - Using an external data interoperability component
  - Avoiding the synchronization of metamodels

- Technical context
  - Cheddar & Platypus
  - Scope extension to AADL V2
  - Osate

- Interoperability component
  - Global design
  - Cheddar XML Translator

- Conclusions and perspectives
Motivation

Integration of legacy systems

- **Legacy systems**
  - Widely used by different communities
  - Built upon different metamodels
  - Metamodels built with different metamodeling tools

- **Integration**
  - Led by a common language
Principles

Using an external data interoperability component

- Data interoperability
  - Relies on the common language
- External component
  - To keep the legacy systems unchanged
Principles
Avoiding the synchronization of metamodels

- Reusing both metamodels in a common metamodeling environment
  - Enables efficient model transformations
  - Implies
    - A tedious and error-prone translation of one metamodel
    - To check the equivalence between the 2 versions
    - To synchronize the 2 versions
Technical context

Cheddar

- A tool for the performance analysis of real-time applications.
  - Provides a set of standard real-time schedulers and analysis tools.
  - Can be extended with specific schedulers and analysis tools thanks to its dedicated programming language.
- Cheddar specifications can be serialized to a specific XML format.
- Cheddar also handles AADL v1 specifications.
Technical context
Cheddar and Platypus

- Some parts of Cheddar are generated by *Platypus*:
  - A software engineering tool fully integrated inside Squeak
  - Allows metamodel specification and transformation rules definition.
  - Provides a modeling and metamodeling language (Express).

- Specialized version of Platypus dedicated to Cheddar:
  - Involves the metamodel of Cheddar.
  - Allows to extend Cheddar with new schedulers and analysis tools.
Technical context

Scope extension to AADL V2

- Cheddar handles AADL v1 specifications thanks to the Ocarina API it integrates.

- To extend the scope of Cheddar to AADL v2 specifications:
  - Cheddar can be *reengineered* using the new version of the Ocarina API
  - AADL v2 specifications can be *externally* translated into Cheddar XML

- We investigate the second solution
  - The external component can reuse the frontend of *Osate v2*:
    - It avoids the reengineering of Cheddar
    - It offers a natural interface to Cheddar for regular Osate users.
Technical context

Osate

- **Osate v2:**
  - Based on EMF; includes a metamodel of AADL
  - Provides a toolset to parse and print AADL specifications
  - Provides a platform for system integration to develop AADL tools
  - Handles AADL v2 specifications
Interoperability component
Global design

- **External view of Dairy:**

  ![External view diagram](image)

- **White-box view of Dairy:**

  ![White-box view diagram](image)
Interoperability component
Cheddar XML Translator - 1

- Manual Cheddar XML Printer
  - Principle: browsing the AADL instances to manually generate strings of Cheddar XML
  - Enables to quickly develop prototypes.
  - Does not enable to capitalize on the translation rules.

![Workflow diagram]

AADL instances (EMF) → Cheddar XML Printer → Cheddar XML Translator → Cheddar XML

Workflow
Interoperability component
Cheddar XML Translator - 2

- Java generation by Platypus and XSLT
  - Principle:
    - using Platypus-Cheddar to generate java classes for each entities of the Cheddar metamodel.
    - browsing the AADL instances to create Java instances
    - using a java-serialization tool
    - defining translation rules to Cheddar XML with XSLT
  - This approach enables to capitalize on the translation rules.

![Workflow diagram]

AADL instances (EMF) → Java Platypus Translator → Cheddar Data (Java Platypus) → XML → XSLT → Cheddar XML

Workflow → XML Serialization
Interoperability component
Cheddar XML Translator - 3

- Specific Cheddar-Java generation by Platypus
  - Principle: modifying Platypus-Cheddar so that it could generate Cheddar specific Java classes including direct serialization into Cheddar XML.
  - Requires a modification of Platypus-Cheddar.
Interoperability component
Cheddar XML Translator - 4

- EMF translator and XML printer:
  - Principle:
    - transforming the AADL instances into instances of a Cheddar metamodel in EMF (can be performed using ATL or QVT)
    - generating Cheddar XML from these instances (can be performed using tools like EMFText)
  - Require to synchronize two versions of the same metamodel (with EMF and with Express).
Interoperability component
Cheddar XML Translator - 5

- EMF translator and XMI translator
  - Principle: variant of the previous solution:
    - standard XMI serialization applied to Cheddar instances
    - transformation into Cheddar XML using XSLT.

Workflow:
AADL instances (EMF) → EMF translator (manual, or produced by ATL/QVT) → Cheddar Data (EMF) → XMI → XSLT → Cheddar XML

Workflow: XML Serialization

Diagram:
AADL instances (EMF) → EMF translator (manual, or produced by ATL/QVT) → Cheddar Data (EMF) → XMI → XSLT → Cheddar XML
Interoperability component
Cheddar XML Translator - 6

- **STEP printer:**
  - **Principle:**
    - translation of the EMF instances into STEP instances of the Express Cheddar metamodel.
    - automatic generation of Cheddar XML.
  - Platypus-Cheddar must be updated to produce Cheddar XML from STEP instances.

Workflow:
AADL instances (EMF) -> STEP Printer -> STEP instances -> Extended Platypus-Cheddar -> Cheddar XML

XML Serialization
Interoperability component
Cheddar XML Translator - overview

<table>
<thead>
<tr>
<th>Approach</th>
<th>Metamodel transformation</th>
<th>Update of Platypus</th>
<th>Reused tools</th>
<th>Generated tools</th>
<th>Impact of metamodel update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>no</td>
<td>no</td>
<td>none</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>no</td>
<td>XSLT</td>
<td>XML serializer</td>
<td>middle</td>
</tr>
<tr>
<td>3</td>
<td>no</td>
<td>yes</td>
<td>none</td>
<td>XML serializer</td>
<td>low</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>no</td>
<td>none</td>
<td>translator and/or printer</td>
<td>low</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>XSLT</td>
<td>translator and/or XMI serializer</td>
<td>low</td>
</tr>
<tr>
<td>6</td>
<td>no</td>
<td>yes</td>
<td>Platypus</td>
<td>none</td>
<td>low</td>
</tr>
</tbody>
</table>

- 1, 2: do not require any modification of Platypus; enable to quickly develop prototypes.
- 3, 6: minimize the impact of the metamodels’ updates
- 4, 5: likely to be automated at the cost of a complete transformation of the Cheddar metamodel
Conclusions and perspectives

- We plan to develop a first prototype with the first approach.
- We will study in depth the impact of approaches 3 and 6 on Platypus-Cheddar.

Typical tool chain including Dairy:
- AADL model built with:
  - OSATE, Ocarina, Topcased
  - or with any other tool using AADL as a pivot language (e.g. MARTE/UML via AAXL transformation implemented with ATL)
- Transformation into Cheddar for validation.