



**Carnegie Mellon
Software Engineering Institute**

Pittsburgh, PA 15213-3890

AADL – Domain Specific Language and UML Profile

Lutz Wrage

**Sponsored by the U.S. Department of Defense
© 2006 by Carnegie Mellon University**



Carnegie Mellon
Software Engineering Institute

Agenda

AADL Domain Specific Language

AADL UML Profile

Translation DSL \leftrightarrow UML Profile



DSL Approach

AADL implemented as a Domain Specific Language (DSL), e.g., in the Open Source AADL Tool Environment (OSATE)

AADL implemented directly via a meta-model

Implementation using Eclipse, and the Eclipse Modeling Framework (EMF)

Meta-model defined in Ecore





EMF and Ecore

Intro

- Ecore: modeling notation; classes, inheritance, attributes, references, (operations)
- EMF: class libraries, code generation
- Used in many Eclipse projects
- No OCL

EMF

- Integration with Eclipse
- Generation of code to manipulate AADL models
- Generation of a model editor
- XML model exchange



AADL Meta-model 1

Design goals

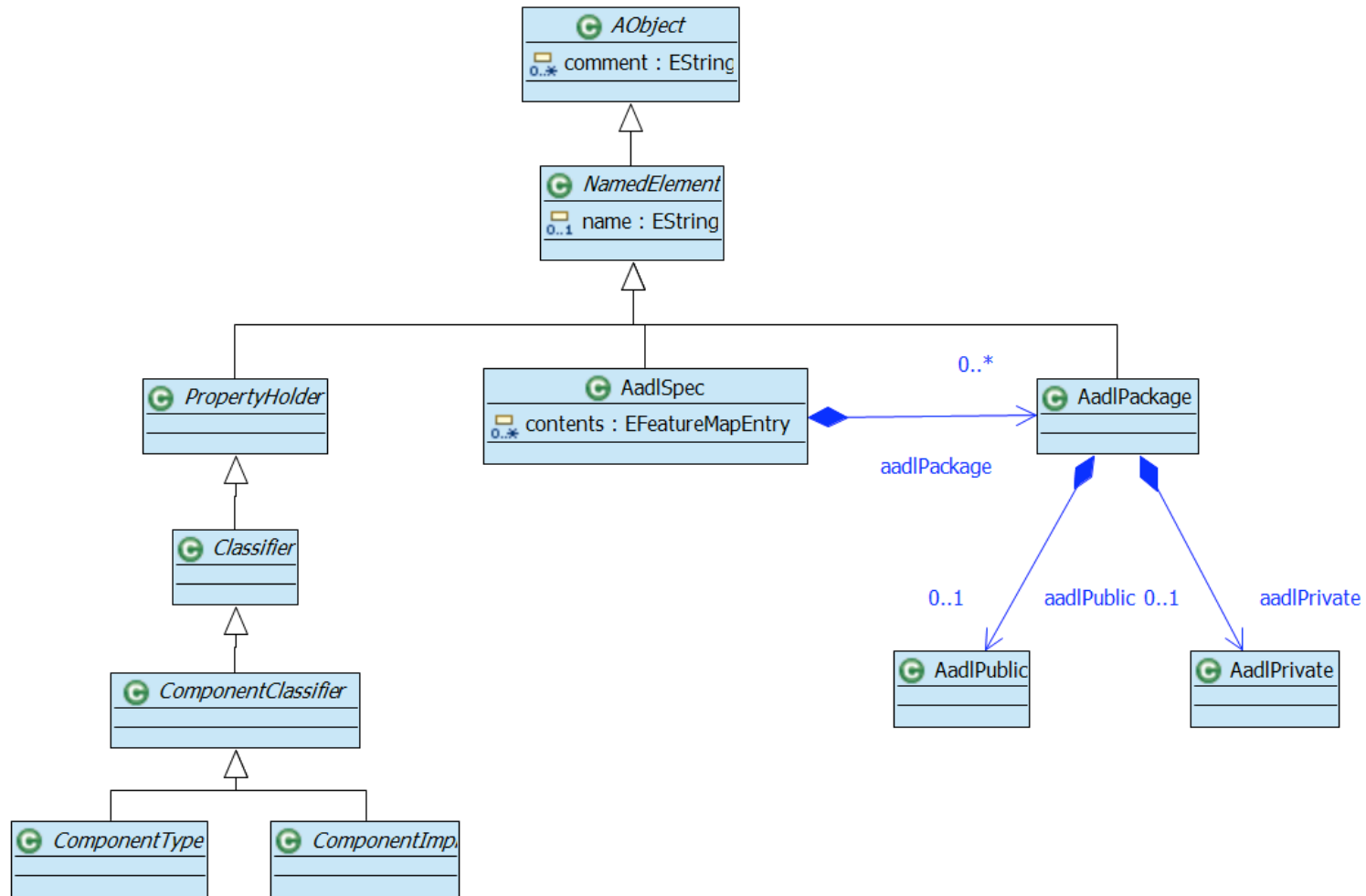
- Syntactic constraints explicit in the meta-model
- Reconstruction of textual AADL from model
- “Readable” XMI serialization

7 Ecore packages

- Core – Basic concepts, mostly abstract classes
- Components – Component types, instances, subcomp.
- Features – Ports, access, subprograms, etc.
- Connections
- Flows
- Properties
- (Instance)

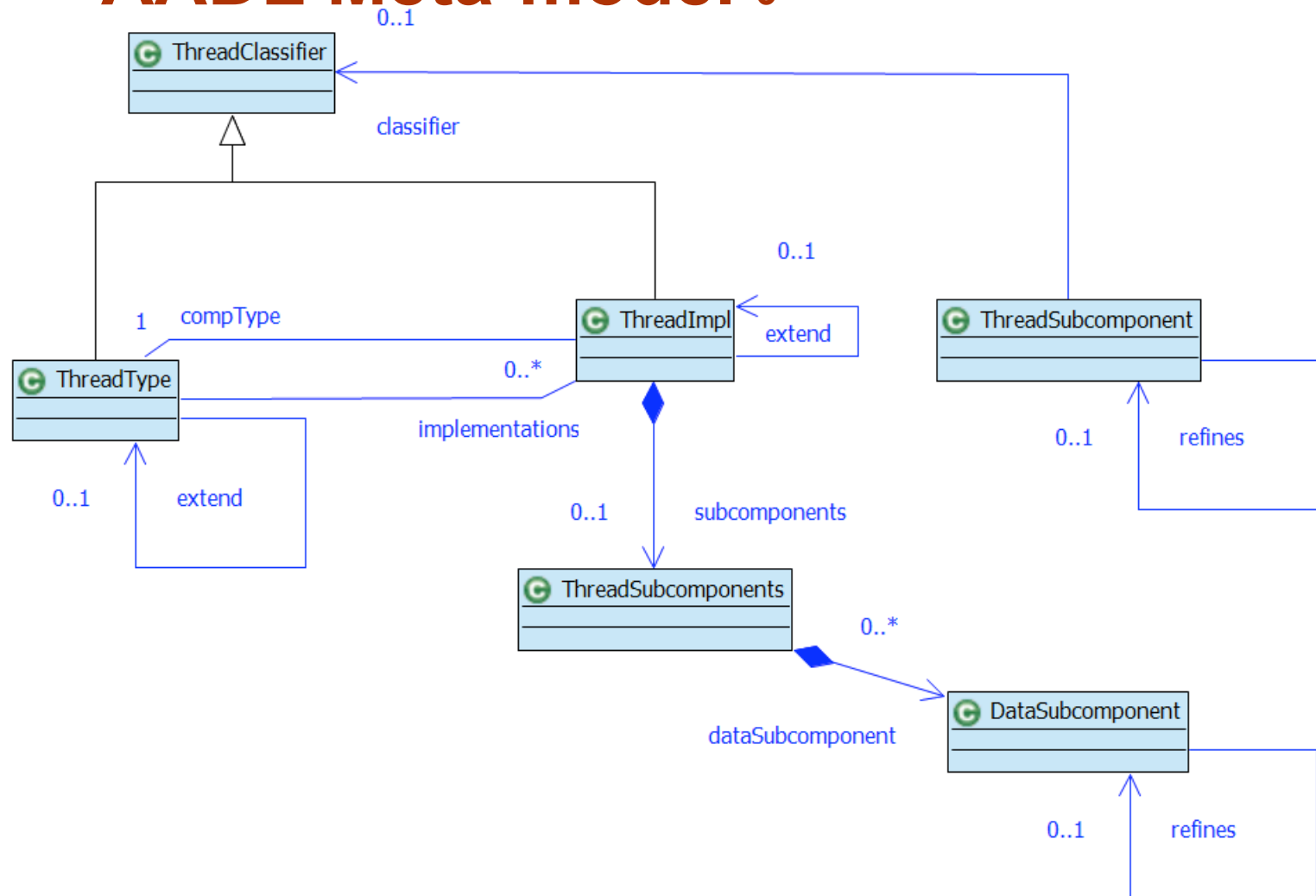


AADL Meta-model 2





AADL Meta-model 3





AADL Example

```
system S
end S;

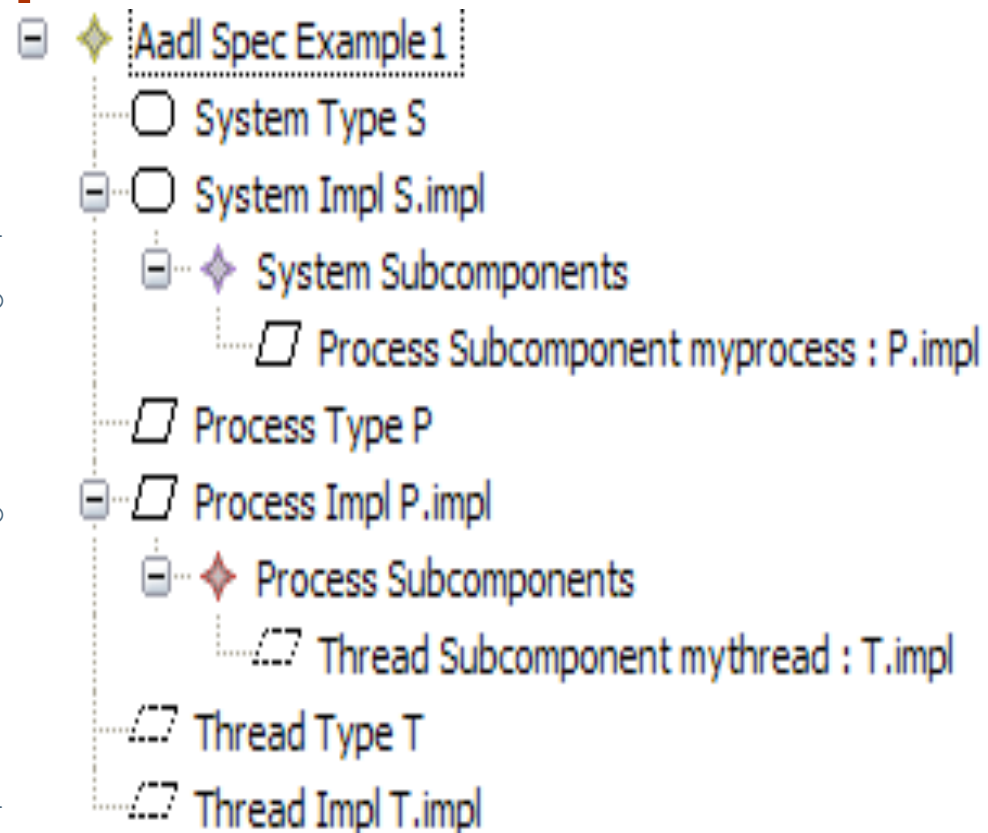
system implementation S.impl
  subcomponents
    myprocess: process P.impl
  end S.impl;

process P
end P;

process implementation P.impl
  subcomponents
    mythread: thread T.impl;
  end P.impl;

thread T
end T;

thread implementation T.impl
end T.impl;
```





Carnegie Mellon
Software Engineering Institute

DSL Summary

Benefits

- Precise semantics of model elements
- Relatively easy to implement analyses

Potential drawbacks

- Specialized approach needs specialized tools
 - Integration into development process
 - Interoperability with UML tools already in use



AADL UML Profile

A set of

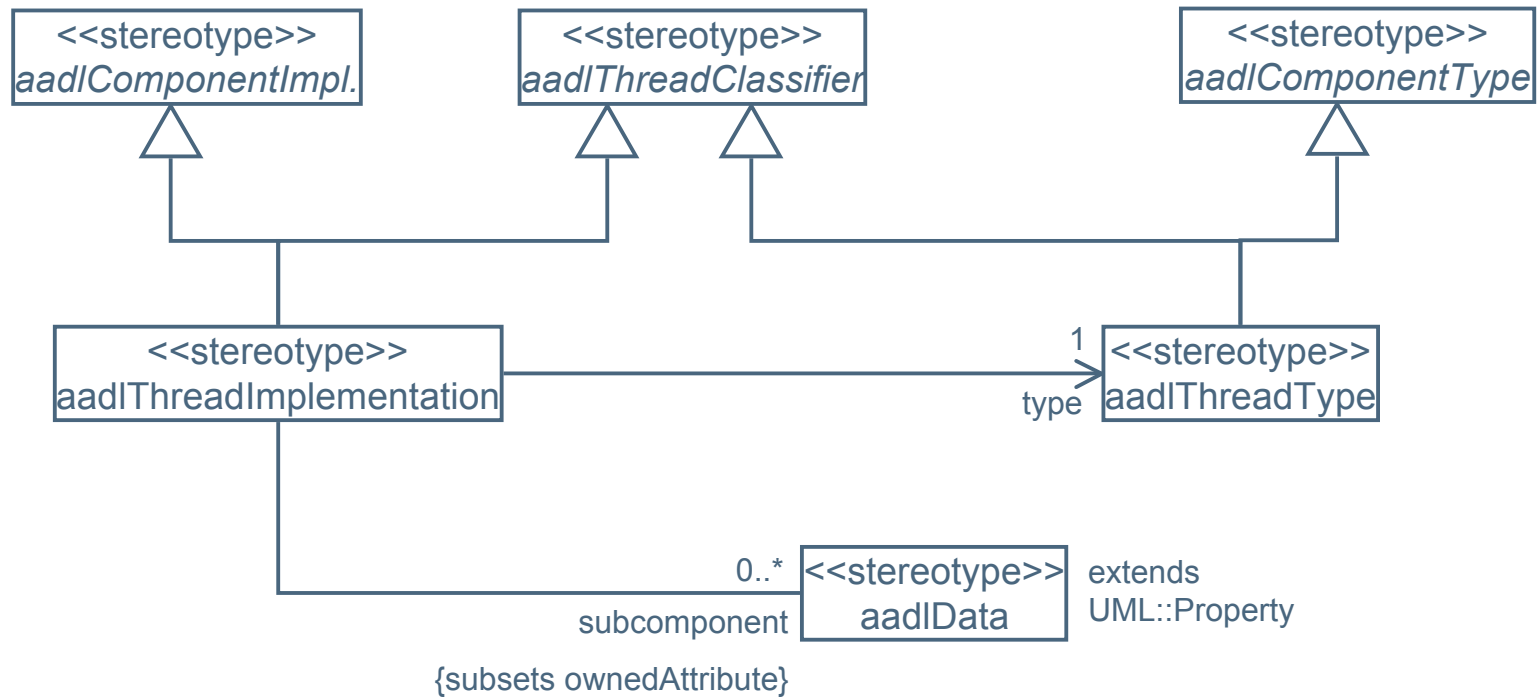
- Stereotypes
 - Abstract stereotypes
Conceptual entities: namespace, component type & implementation, feature, etc.
 - Concrete stereotypes
AADL entities: system type & implementation, data port, data connections, etc.
- Constraints
 - Ensure consistency
 - Limit composition options
- Stereotype properties
 - AADL standard properties



UML Profile: Elements

AADL Concept	UML Profile
Component Type	Stereotype <i>aadlThreadType</i> , ... extend UML Class
Feature	Stereotype <i>aadlDataPort</i> , ... extend UML Property
Component Implementation	Stereotype <i>aadlThreadImplementation</i> , ... extend UML Class
Subcomponent	Stereotype <i>aadlThread</i> , ... extend UML Property
Package, Property Set	Stereotype <i>aadlPackage</i> , <i>aadlPropertySet</i> extend UML Package
Connection	Stereotype <i>aadlDataConnection</i> , ... extend UML Association

UML Profile: Constraints





UML Profile Example

```
system S  
end S;
```

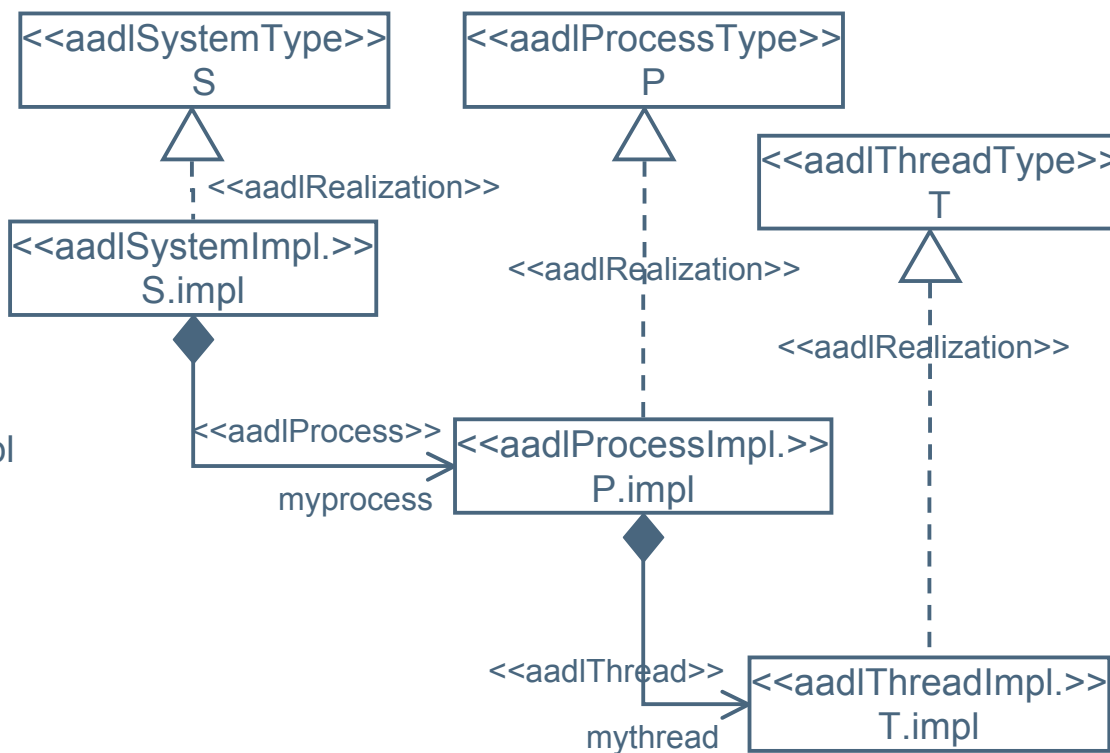
```
system implementation S.impl  
subcomponents  
  myprocess: process P.impl;  
end S.impl;
```

```
process P  
end P;
```

```
process implementation P.impl  
subcomponents  
  mythread: thread T.impl;  
end P.impl;
```

```
thread T  
end T;
```

```
thread implementation T.impl  
end T.impl;
```





DSL – UML Profile Translation

Bridge the two approaches by translating between

- AADL models conforming to the AADL meta-model
- AADL models conforming to the AADL UML profile

Enables

- Use of mature UML tools for model development
- Use of AADL specific analysis techniques based on AADL meta-model

Goals

- **Enable commercial tool vendors to quickly support AADL**
- **Speed up transition of AADL into industry**



Processes

1. Create AADL model using UML tool
2. Translate to AADL meta-model
3. Run analyses, e.g., in OSATE
4. Translate to UML Profile
5. Refine UML model

UML tool adds

- Version control / configuration management
- Reporting
- Integration with other tools, e.g., requirement management



Technology

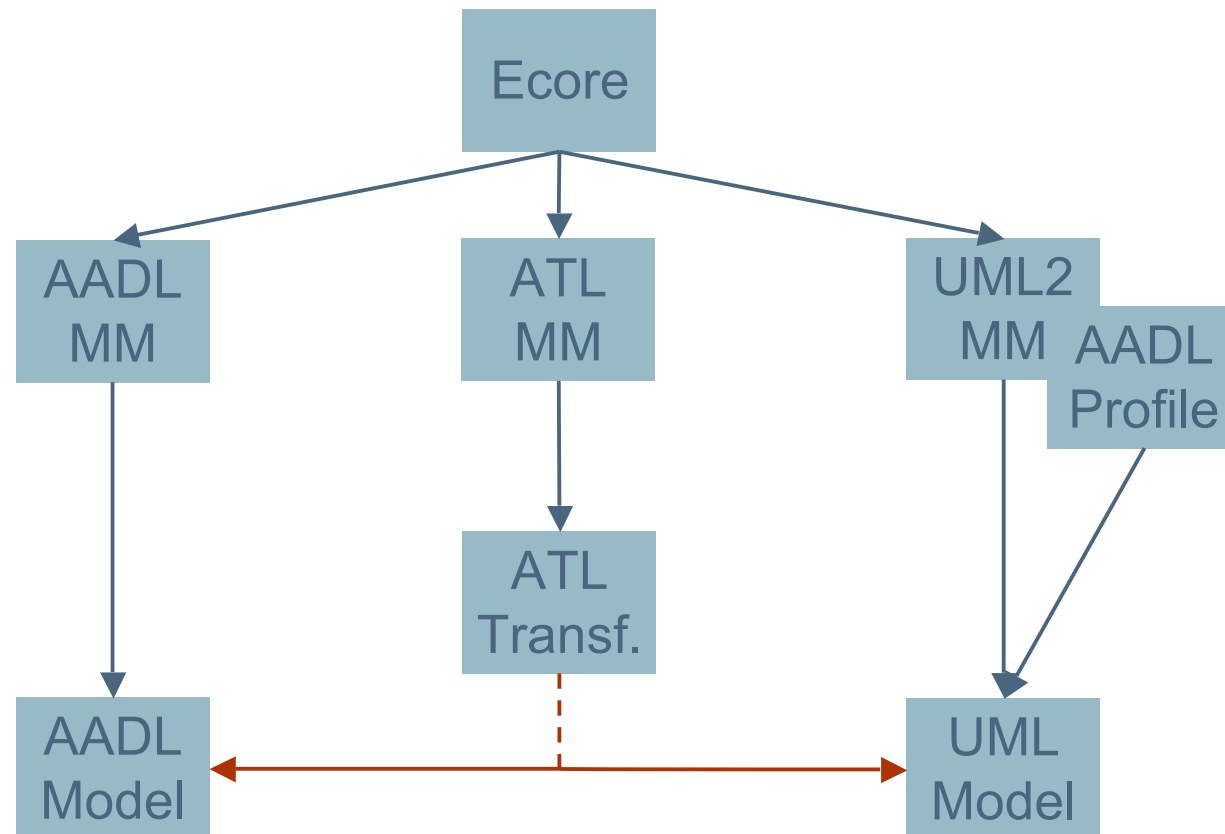
Current implementation based on

- Eclipse
- EMF
- UML2
- Atlas Transformation Language (ATL)

ATL

- Subproject of Eclipse GMT project
- Developed at University of Nantes, France
- Rule-based transformation of Ecore models
 - Declarative rules match model elements
 - Imperative constructs handle UML profiles

Using ATL





ATL Example

From AADL to UML Profile transformation

```
rule SystemType {  
  from s : AADL!SystemType  
  to t : UML!Class (  
    name <- s.name  
  )  
  do {  
    thisModule.applyStereotype(  
      t, "aadlSystemType");  
  }  
}
```



Translation Status

Draft version of UML Profile

Proof of concept implementation of AADL to UML

Plan

- Complete translation in parallel with profile definition
- Make translation available for free as open source
- Based on our reference implementation develop tool-specific versions, e.g., for RSA or Rhapsody



Carnegie Mellon
Software Engineering Institute

Links

AADL Web site: www.aadl.info

OSATE download: www.aadl.info/download

Software Engineering Institute : www.sei.cmu.edu