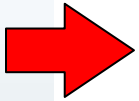
The Software Engineering Institute

Open Source AADL Tool Environment (OSATE)

Peter Feiler
Software Engineering Institute
phf@sei.cmu.edu

Outline



- AADL Tool Strategy
- OSATE Capabilities
- Extensible OSATE Plug-in Architecture
- Meta-model based AADL model processing library
- Annex Extensions & OSATE

The Software Engineering Institute

Key Elements of SAE AADL Standard

- Core AADL language standard
  - Common real-time systems modeling notation across organizations
  - Extensible notation

Language standard approved and published Nov 2004

- AADL Meta model & XMI/XML standard
  - Model interchange & tool interoperability
  - EMF-based model processing API

Passed ballot in Summer 2005

- UML profile for AADL
  - Transition path for UML practitioner community
  - Conceptual model aligned with AADL Meta Model

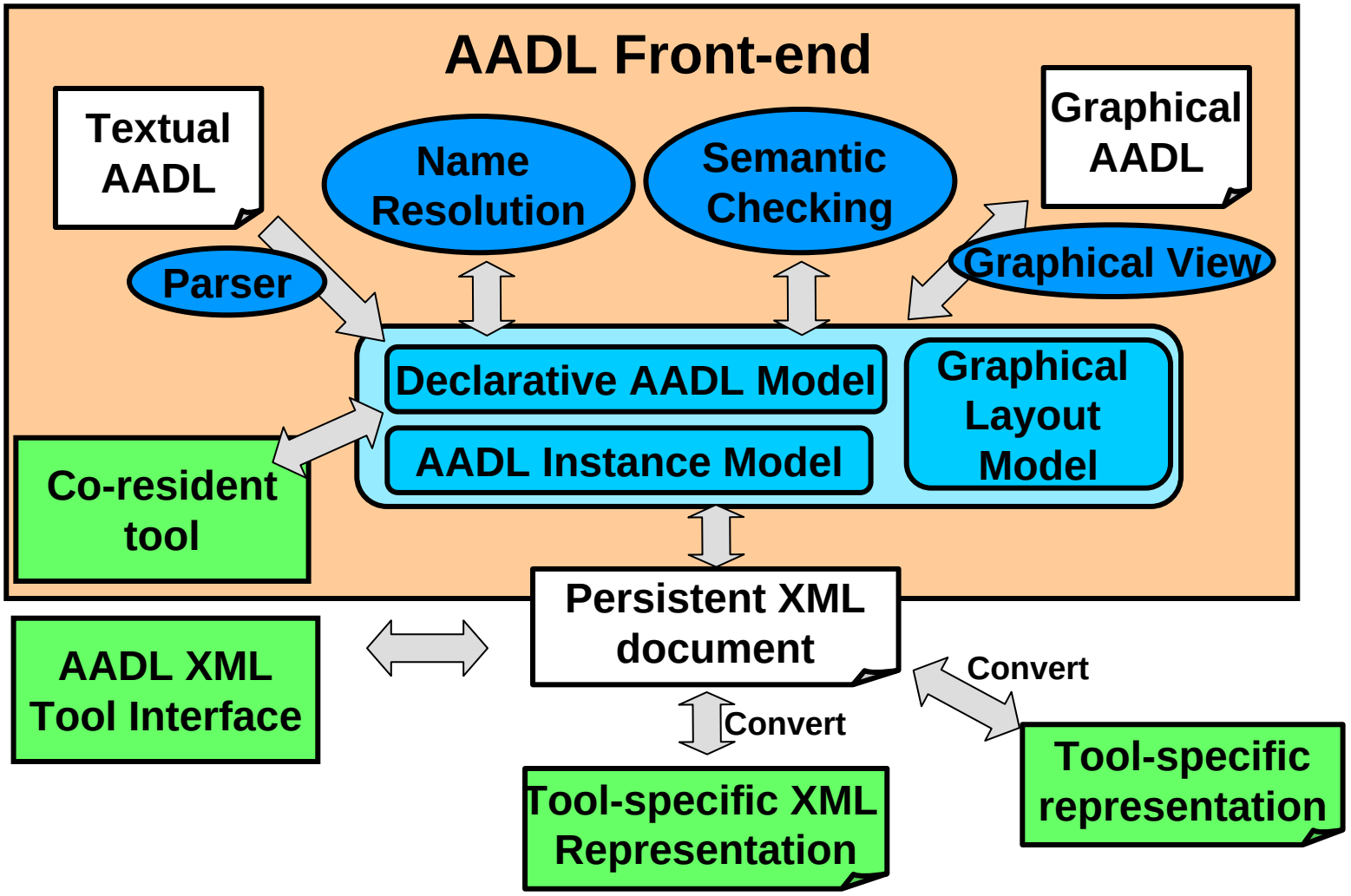
Formal ballot in Falll 2005

The Software Engineering Institute

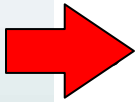
Two-Tier Tool Strategy

- Commercial Tool Support
  - UML tool environment extension based on UML profile (Artisan, Rational, ILogix)
  - Extension to existing modeling environment with AADL export/import (TNI Europe)
  - Analysis tools interfacing via XML (Airbus, Rockwell)
- Open Source AADL Tool Environment (OSATE)
  - Low entry cost solution (no cost Common Public License)
  - Multi-platform support based on Eclipse
  - Vehicle for in-house prototyping of project specific architecture analysis
  - Vehicle for architecture research with access to industrial models & industry exposure to research results

XMI/XML Based Tool Interoperability



Outline



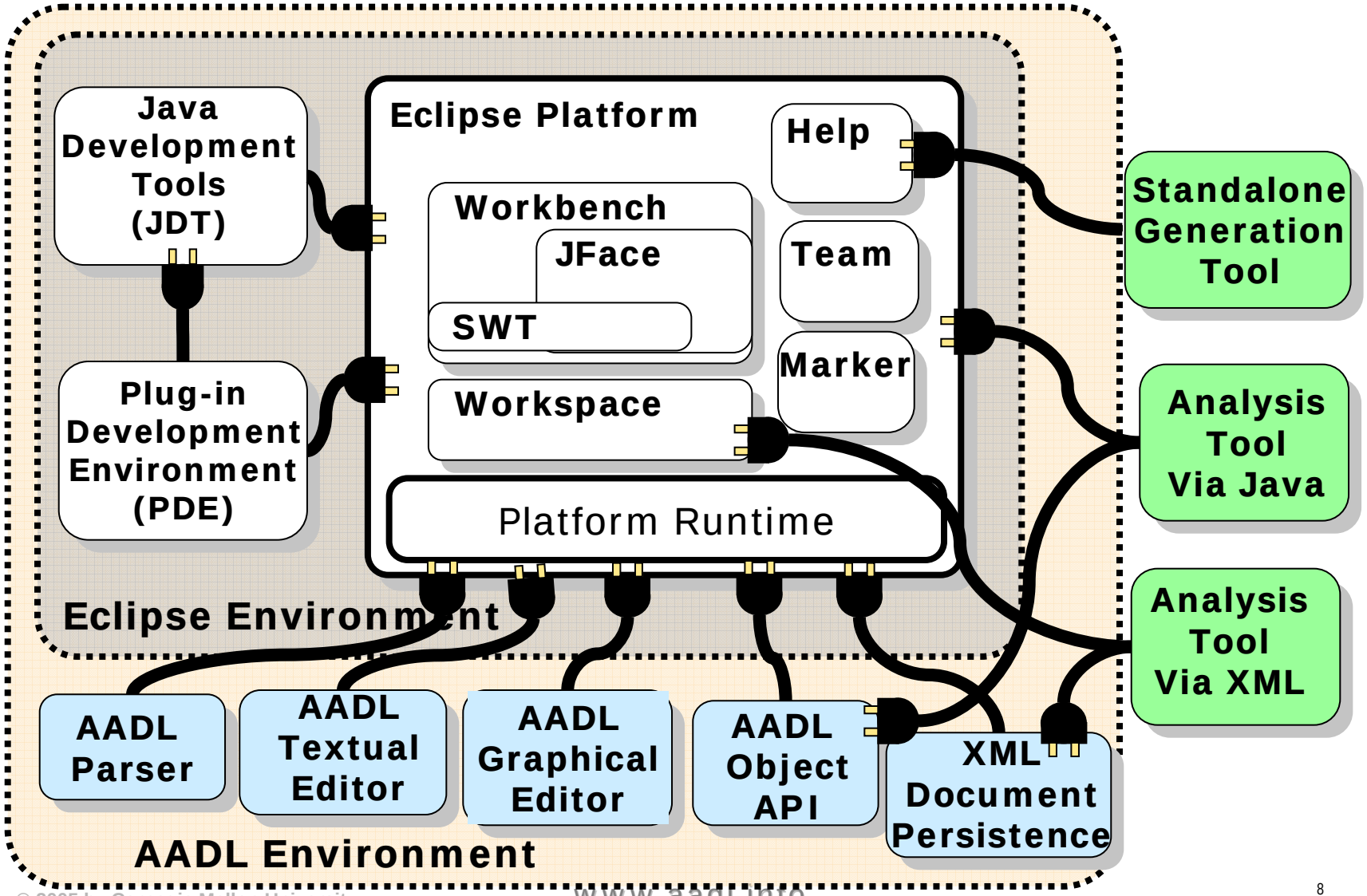
- AADL Tool Strategy
- OSATE Capabilities
- Extensible OSATE Plug-in Architecture
- Meta-model based AADL model processing library
- Annex Extensions & OSATE

OSATE Capabilities

- OSATE Release 1.0 for Eclipse Release 3.0/EMF2.0
  - Parsing & semantic checking of AADL V1.0
  - AADL property viewer
  - Model instantiation
  - Modal system processing
  - Auto-translation AADL text <-> AADL XML
  - Auto-generation of instance models
  - Multi-file support & version control interface
- OSATE Release 1.1 for Eclipse 3.1/EMF 2.1
  - Approved AADL meta model & XML/XMI V1.0
  - Marker-based report generation
  - Extension points for Annex sublanguages
- To come
  - Automation of plug-in processing
  - Graphical editor support

21000 line AADL model in 6 sec

Eclipse Plug-in Architecture



OSATE Frontend Plug-ins

- AADL Model Processing: edu.cmu.sei.aadl.model
- AADL Model Viewing & Edit Cmds: edu.cmu.sei.aadl.model.edit
- AADL Object Editor: edu.cmu.sei.aadl.model.editor
- AADL Text Editor: edu.cmu.sei.aadl.texteditor
- AADL Parser/Semantics: edu.cmu.sei.aadl.model.parser
- AADL Text Generator: edu.cmu.sei.aadl.unparser
- AADL Model Instantiation: edu.cmu.sei.aadl.instance
- OSATE builder: edu.cmu.sei.osate.core
- OSATE UI Support: edu.cmu.sei.osate.ui
- OSATE Project and File Support: edu.cmu.sei.osate.workspace
- AADL Help: edu.cmu.sei.aadl.help

Includes HTML version of standard

The Software Engineering Institute

OSATE Analysis Plug-in Examples

- Safety criticality/security level analysis
  - User-defined properties
  - Analysis on declarative model & instance model
- AADL Model statistics
  - Operates on declarative model library & instance models
- AADL to MetaH translator
  - Sorting, context-sensitive processing
- Distributed Resource Binpacking & Scheduling
  - Interface with Java-based external model
  - Priority inversion checking on AADL model
- Flow latency analysis
  - Spec & instance based flow latency analysis
  - Support for partition-based latency

The Software Engineering Institute

An Extensible Engineering Environment

Embry-Riddle
Reliability Analysis

U.Penn
Hybrid Systems Models

System Verification Manager (CMU)
Simulink/Matlab, Dymola models

AADL Extensions
Annex sublanguages

Architecture Import & Extraction

Concurrency Analysis

Architecture Export
MetaH, *TTA*

MetaH Toolset (Honeywell)
Scheduling analysis
Reliability analysis
Isolation analysis
Runtime system generation

U.Penn
Process Algebra

Object Model Interface
Network model

Model Export Filters
Timing model

TimeWeaver (CMU)
Distributed resource allocation
Multi-platform runtime system generation

TimeWiz Commercial Tool
Scheduling analysis
Execution trace analysis

The Software Engineering Institute

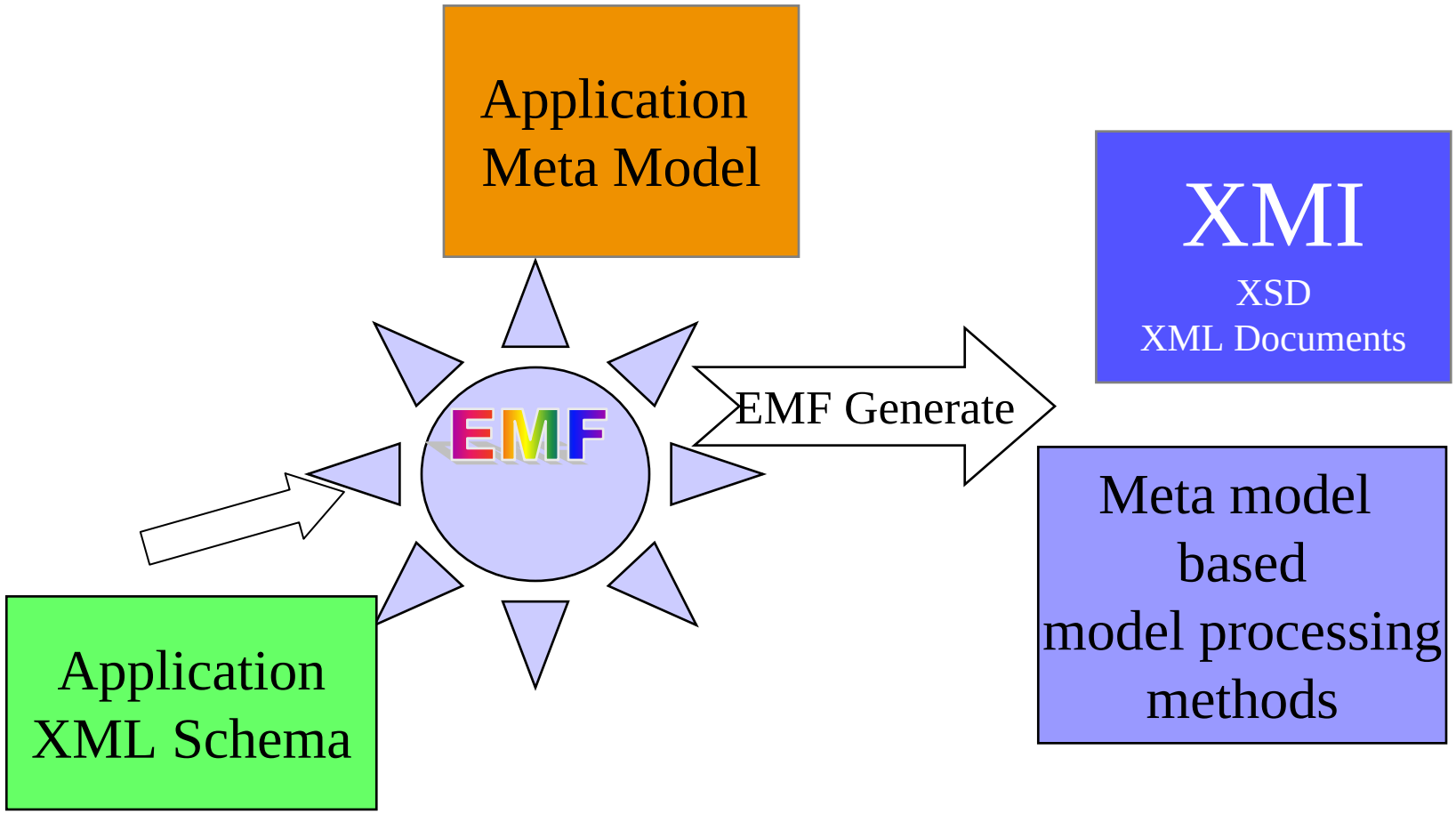
A Research Transition Platform

- In-house prototyping of project specific architecture analysis
  - Rockwell-Collins (Ether-switch workload analysis)
  - Honeywell (Timing & fault tree analysis of alternative avionics architectures)
  - COTRE Airbus Industries (Concurrency behavior verification)
- Turning architecture research into tools
  - Eglin AFB SBIR Phase 2, 21st Century Systems
    - Weapons Plug'n'Play compatibility analysis
  - U.Penn, Fremont Associates STTR phase 2
    - Map hybrid control system language (Charon) into AADL
    - AADL & process algebra
  - EU ASSERT project (Proof-based verification of embedded aerospace architectures)
  - Embry-Riddle (Error model plug-in prototype)
  - U. Illinois (Assumption management)

Outline

- AADL Tool Strategy
- OSATE Capabilities
- Extensible OSATE Plug-in Architecture
- Meta-model based AADL model processing library
- Annex Extensions & OSATE

Leverage of EMF



EMF – Eclipse Modeling Framework

The Software Engineering Institute

AADL Meta Model

- Defined in Eclipse Modeling Framework (EMF)
  - Collection of meta model packages with graphical views
  - Separate from, but close to UML profile of AADL
- XML as persistent storage
  - XMI specification from Ecore meta model
  - Generated XML schema
- In-core AADL model
  - Generated methods for AADL model manipulation
  - Edit history, deep copy, object editor, graphical editor
  - Methods to support
    - AADL extends hierarchy
    - feature “inheritance”
    - property value “inheritance”

The Software Engineering Institute

OSATE Plug-in Extensions

OSATE
XML/XMI AADL, AADL object model API
AADL extension support

EMF
XML/XMI, Metamodel
Change notification
Multi-file support

Eclipse
Platform independence
Extensible help
Task & Problem Mgt
Team support
Plug-in development

AADL Front-end
Text editor
Object editor
Graphical editor
Text<->XML
Semantics

OSATE Extensions
Analysis template
Generation template
AADL Semantic API

External Models
External tools

Model Transformation
Timing analysis (RMA)

Architecture Import
Simulink/Matlab model
Extraction via SVM

Architecture Export
MetaH

Architecture Analysis
Security level
Data stream miss rate
Latency

Architecture Consistency
Required connectivity
Model completeness profiles
Connectivity cycles

Architecture Transform
Conceptual architecture -> Runtime architecture
Rate group optimization
Port group identification

The Software Engineering Institute

Declarative or Instance Model

- Declarative model
  - Analyze complete library of component declarations
  - Identify compositional consistency & constraints
  - Process application domain characteristics
  - Propagation along containment hierarchy
  - Connection & flow analysis
  - Mode specific processing per component implementation
- Instance model
  - Runtime architecture of embedded system
  - Process runtime properties
  - Work with
    - instance hierarchy
    - semantic connections
    - Instance specific property values
    - modal system instances (System Operation Mode)

The Software Engineering Institute

Type of OSATE Plug-ins

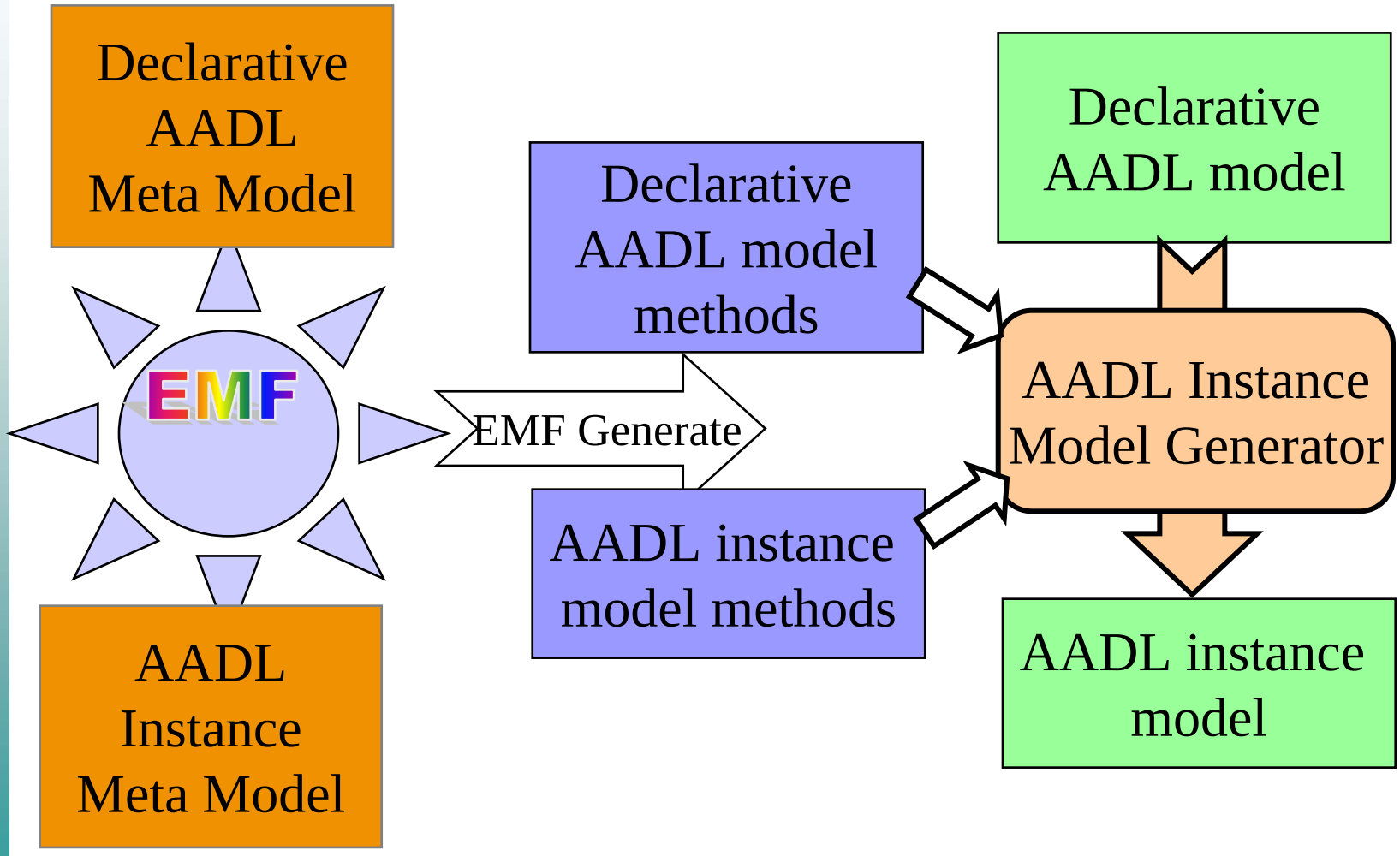
- Model analysis
- Model generation
- Model transformation
- Model extraction
- Source code compliance
- Core language extension

Focus of this session on Model analysis

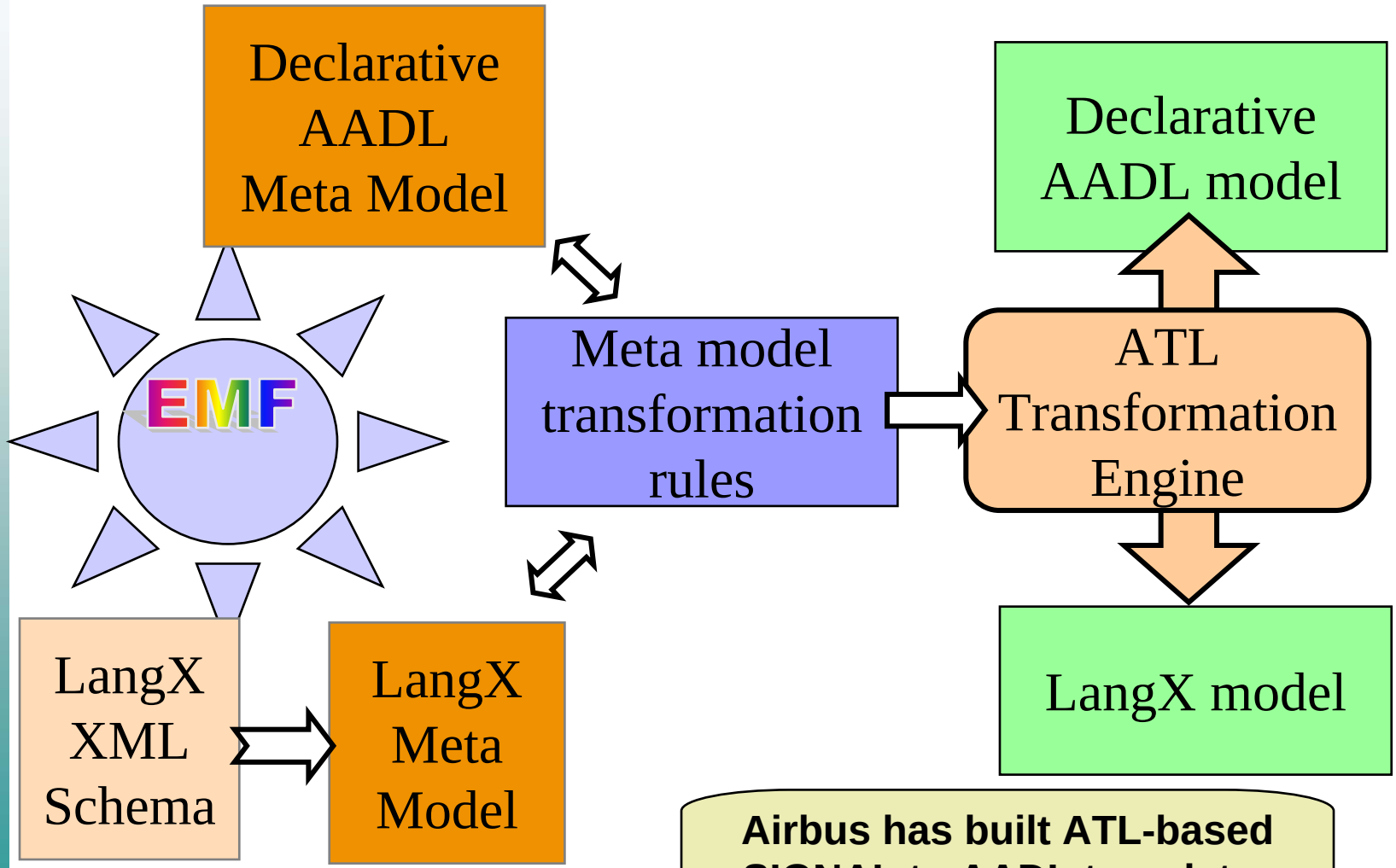
Generators

- Textual AADL
  - Fully preserved AADL model content
- AADL instance model
  - Semantic connections & system operation modes
  - Port group unfolding
- AADL Model transformation
  - Refactoring
  - Restructuring
  - Template instantiation
  - Model optimization
- Model translation
  - Between XML-based representations

EMF-Based Instance Model Generator



ATL-Based Model Translation



OSATE Multi-File Support

- Textual & XML AADL Models
  - Separate file per package
  - Separate file per property set
  - Separate file for AADL specification
- Eclipse Workspace & AADL Name Spaces
  - Eclipse workspace as global name space
  - AADL spec file as anonymous name space
    - Contained packages and property sets are local to spec file
  - Multiple AADL spec files => multiple anonymous name spaces
  - Separate XML file for each instance model
- Need to manage analysis/generation results

The Software Engineering Institute

Execution of OSATE Plug-ins

- Explicit user invocation
  - On specific model objects
  - On all relevant model objects
  - Aaxl<ReadOnly/Modify>Action method
- Automated processing
  - Builder: Resource change propagation & synchronized processing
  - Listener: in-core model object observer notification
- Multi-user support
  - Team synchronization capability in Eclipse
  - Interfacing to version control system

The Software Engineering Institute

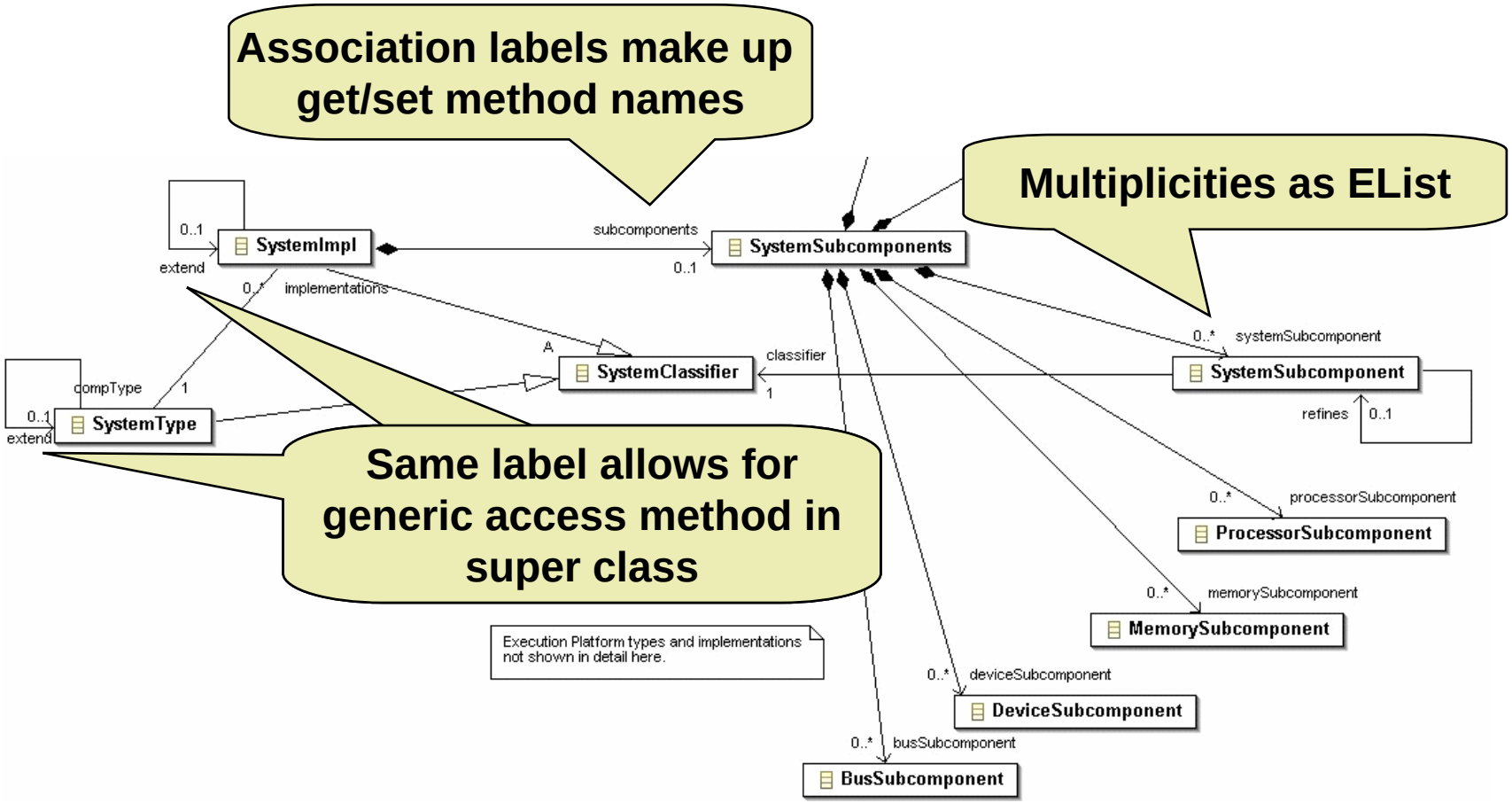
Plug-ins & Standalone Tools

- Standalone tools as Eclipse plug-ins
  - XML file based processing
  - Tool registry in Eclipse
  - Configurable tool invocation
- Analysis capability as OSATE plug-in
  - Incore model analysis
  - Leverage model processing infrastructure
  - Leverage Eclipse IDE
- Repackage plugin as standalone tool
  - Use EMF-based model processing API
  - Use AADL meta-model driven XML processor
  - Batch or interactive tool interface

Outline

- AADL Tool Strategy
- OSATE Capabilities
- Extensible OSATE Plug-in Architecture
- Meta-model based AADL model processing library
- Annex Extensions & OSATE

AADL Meta Model & Access Methods



Access to Model Objects

- Generated access methods
  - For all containment associations
  - For all reference associations
  - Multiplicity associations as ELists
- Generic access methods in abstract classes
- AADL inheritance and access methods
  - Feature inheritance by components
  - *Extend/refine* inheritance by component types and implementations
  - Property value inheritance

Model Processing Support

API in package edu.cmu.sei.aadl.model.util

- Basic model processing support
  - Default processing of visited model objects
  - Filtered processing
  - User-defined actions
  - List-based processing
- Containment hierarchy traversal methods
- Content-driven processing

Basic Model Processing

- Class ForAllAObject
  - Provides containment traversal methods
  - Supports result reporting via Eclipse markers
- Redefinable default processing methods
  - suchThat: model object specific condition (default:true)
  - action: conditionally executed method (default: append visited model object)
  - process: called on every visited model object (default: calls action if suchThat is true)
  - Default implementation returns all conditionally visited model objects

The Software Engineering Institute

Containment Hierarchy Traversal

- Traversing the declarative model
  - Declaration hierarchy
- Traversing the Instance model
  - Corresponds to system component hierarchy

Does not represent system hierarchy

- Hierarchy traversal methods
  - Prefix and postfix order for declarative & instance models
  - Declaration/use order for declarative models
  - Category-specific component instance processing

The Software Engineering Institute

Property Lookup Methods

`PropertyHolder` declares property lookup methods

- **`getSimplePropertyValue(PropertyDeclaration pd)`**
  `getSimplePropertyValue(String ps, String pn)`
  - **For non-modal association of a single-valued property**
  - Returns a **`PropertyValue`** object,
    or **`null`** if association is modal, multi-valued, or not found.
- **`getPropertyValueList(PropertyDeclaration pd)`**
  `getPropertyValueList(String ps, String pn)`
  - **For non-modal association of a multi-valued property**
  - Returns a **`List`** of **`PropertyValue`** objects,
    or **`null`** if association is modal or not found

The Software Engineering Institute

Content-Driven Model Processing

- Limitations of traversal-based switch processing
  - Subclass before superclass processing
  - One parent visit only
  - Prefix only, postfix only
- Need for content-driven processing
  - Different content orderings
  - Optional list processing
  - List element separator & terminator processing
  - Multi-pass processing
  - Property value driven output
  - Context-sensitive output

AADL Text Generator

edu.cmu.sei.aadl.unparser

- Use of class hierarchy
  - Category-specific processing before common processing
  - Component types, implementations, subcomponents, features
- Processing of optional subclauses
  - Subclause processing: Non-existent, empty, non-empty lists
  - Property value lists: list element separators
  - Lists of declarations: list element terminators
  - Automatic indentation
- Context-sensitive processing
  - Property associations as subclause vs. bracketed list
  - Package qualification for non-local references

The Software Engineering Institute

MetaH Text Generator

edu.cmu.sei.aadl.toMetaH

- Multi-pass processing
  - All data types must be declared as port types
  - Data type properties in port type package implementation
  - Data component declarations as port types, property associations, monitors
- Component declaration before use
  - Component type before component implementation
  - Component implementation before subcomponent classifier reference
- Property-based reserved words
  - Thread dispatch protocol
- Content-dependent reserved words
  - System vs. macro reserved word

Outline

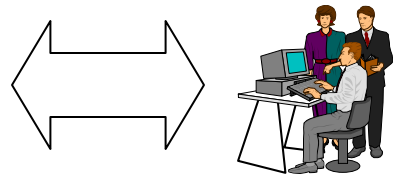
- AADL Tool Strategy
- OSATE Capabilities
- Extensible OSATE Plug-in Architecture
- Meta-model based AADL model processing library
- Annex Extensions & OSATE

AADL Language Extensions

- New properties through property sets
- Sublanguage extension
  - Annex subclauses expressed in an annex-specific sublanguage
- Examples
  - Error Model (Honeywell/Embry-Riddle)
  - Concurrency Behavior (Airbus)
  - Assumption management (U. Illinois)

Reliability Analysis with AADL

OSATE
(with Error Model Annex plug-in)



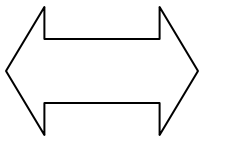
- **create**
- **edit**
- **check syntax**

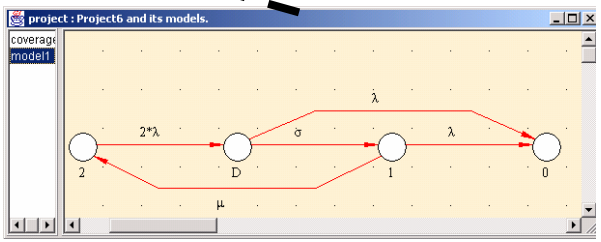
```
processor implementation PPC.M703
annex error {**
   Model => My_Models::Basic.Nominal;
    Fail.Occurrence => poisson 10E-4; **}
end PPC.M703
```

- **transfer**

- **Error models are declared within AADL specifications.**
- **An error model declares a set of error states, transitions and properties that specify error model behavior.**
- **Supports reliability analysis with interfaces to analysis tools (e.g. SHARPE, Mobius).**

SHARPE



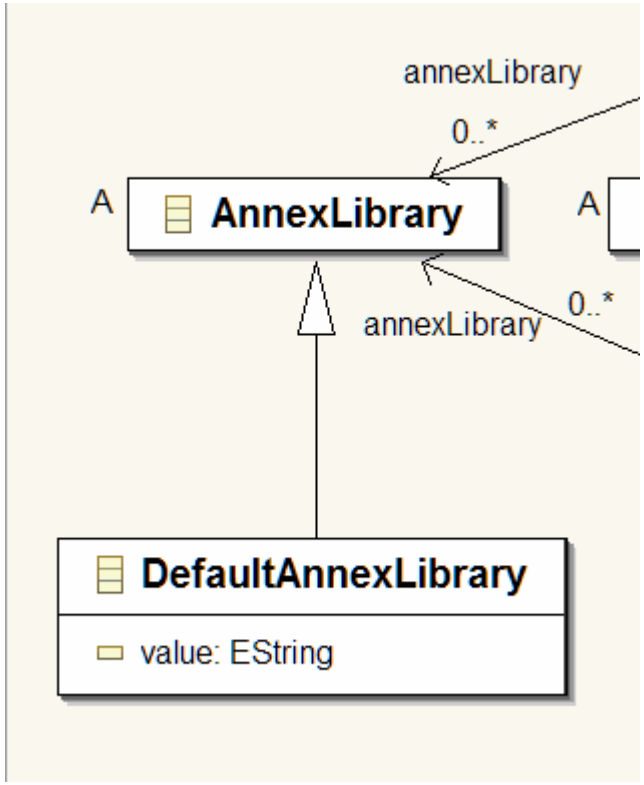


- **analyze**

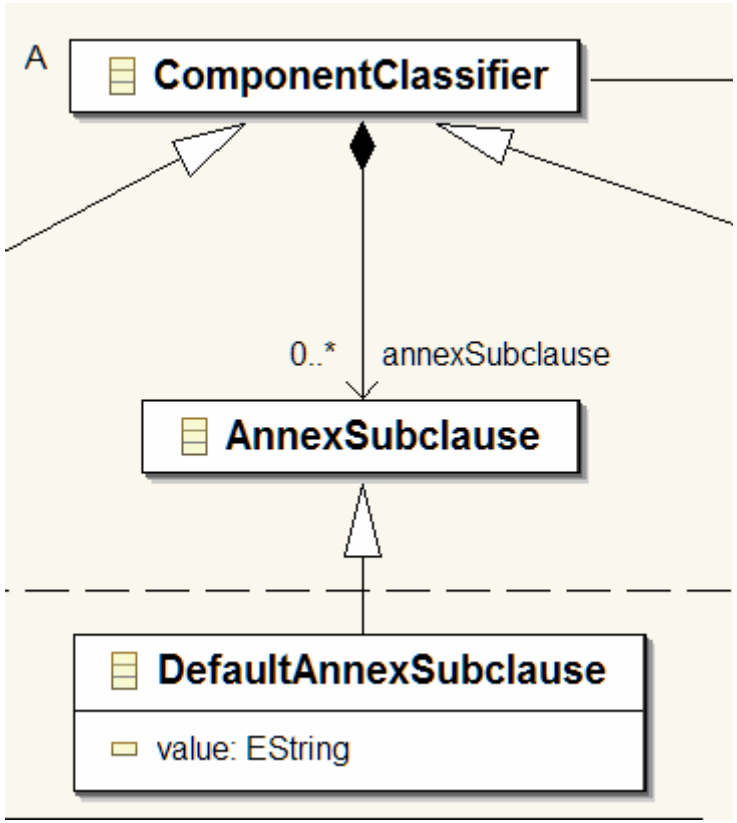
Language Extension Implementation

- Property sets handled by OSATE
- Sublanguage extension
  - AADL meta model provides abstract Annex classes
  - Sublanguage meta model as additional EMF meta model package
    - Subclassing of AADL meta model classes
    - Current limitations to AADL property reuse
  - OSATE extension points
    - AADL parser callout to sublanguage lexer/parser
      - Two-pass & In-place parsing approach
    - AADL Unparser callout to sublanguage

AADL Metamodel



aadlspecpackage.ecd



core.ecd

Summary

- XML/XMI specification provides flexible tool integration strategy
- OSATE is a low-cost entry-point & prototyping platform
- Eclipse plug-in architecture facilitates analysis & generation plug-in development
- Reusable AADL model processing library through EMF-based AADL meta model