LMP:
Logic Model Processing

Prolog Day Symposium
Paris, 10 Nov 2022

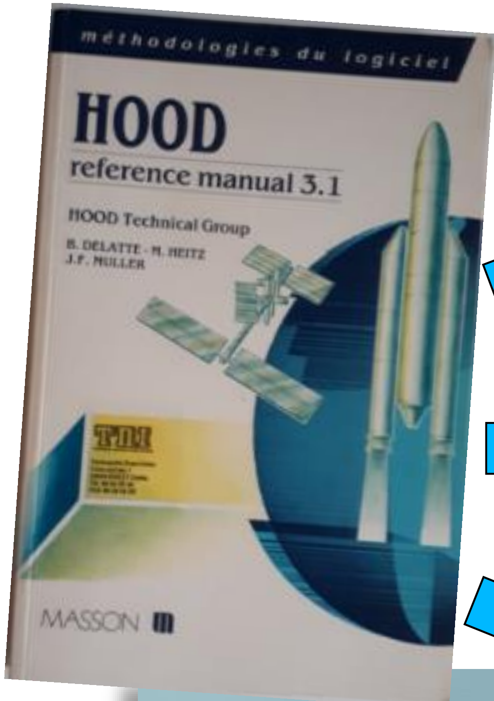
P. Dissaux
Ellidiss Technologies, 24, quai de la douane,
29200 Brest, Brittany, France

Ellidiss
Technologies
www.ellidiss.com

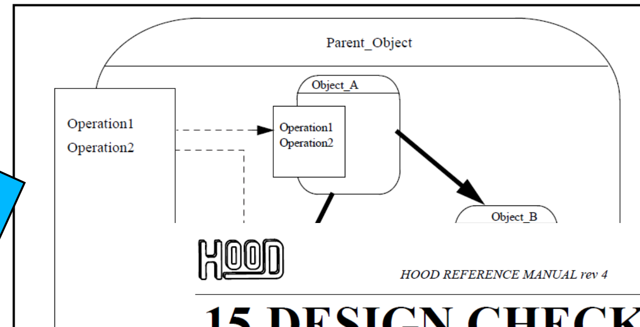
Ellidiss
Technologies
www.ellidiss.com

LMP Genesis
~ 1995

HOOD
for critical software design



DESIGN



VERIFICATION

15 DESIGN CHECKING, SCOPING, VISIBILITY AND HOOD RULES

A HOOD model is a representation of a System to Design in terms of objects, classes, generics and relationships within a system configuration. In order to ensure consistency of these representations, HOOD defines visibility and scoping rules.

CODE GENERATION

The term module is used to designed object, class and generic.

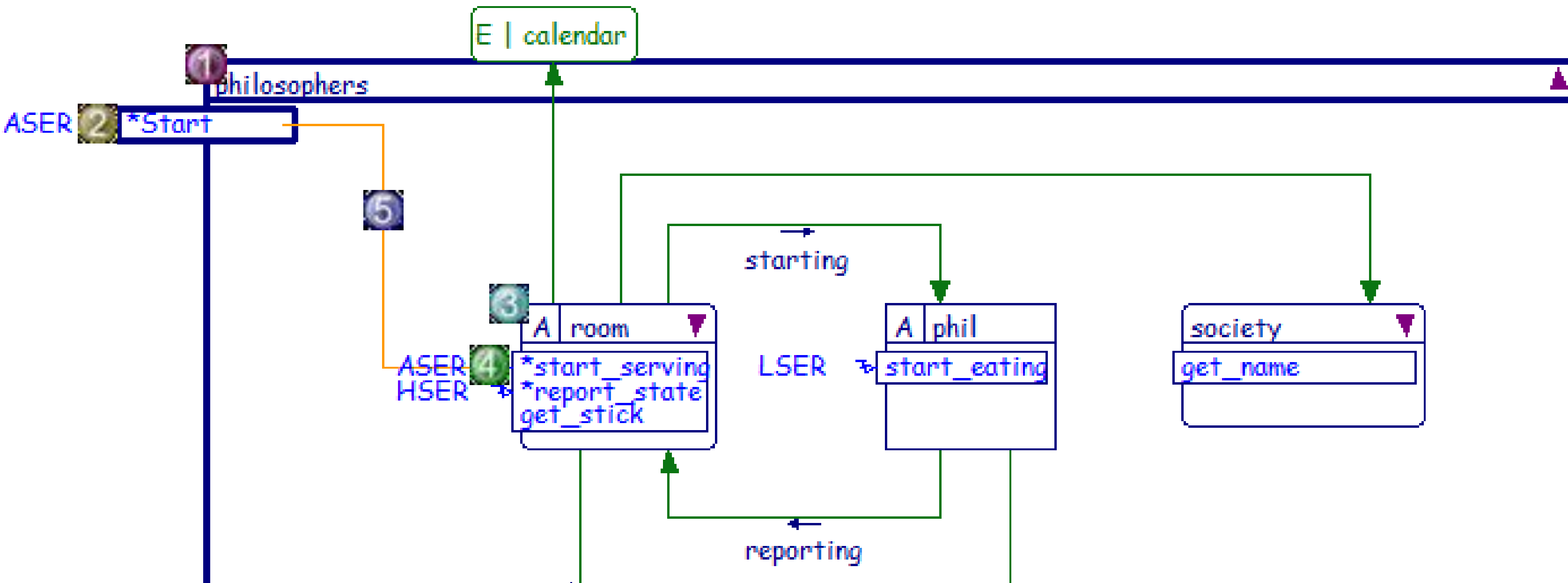
AG-1. A module is mapped into a package named as the object.

Optional rules shall specify the conditions for the generation of library units, nested units, subunits, (private) child units or protected units. By default child objects are mapped into library units and operations into associated bodies.

AG-2. A provided operation of a terminal module is mapped into the declaration of a subprogram specification in the package specification and in a subprogram body in the package body associated with the object.

Optional rules may specify the condition for the generation of subunits[17]. By default they are nested in the package body associated to the object.

From Design to Facts




```
…
1 isObject('philosophers','PASSIVE','SYSTEM_CONFIGURATION').
2 isProvided('Start','OPERATION','philosophers’).
…
3 isObject('room','ACTIVE','philosophers’).
4 isProvided('start_serving','OPERATION','room’).
5 isImplementedBy('Start','OPERATION','philosophers’,
                 'start_serving','room',1).
…
```

Design Rules

16.2 INCLUDE RELATIONSHIP

The include relationship describes modular breakdown of a MODULE into other MODULEs.

I-1 A PARENT is a MODULE that INCLUDEs at least one other MODULE.

I-2 A CHILD is a MODULE which is INCLUDEd by a PARENT.

I-3 When a PARENT is broken down into CHILDren, they shall provide together the same functionality.

I-4 *A CHILD shall not have more than one PARENT.*

I-5 *A MODULE shall not INCLUDE itself.*

I-6 Each OPERATION provided by a PARENT shall be IMPLEMENTED_BY one OPERATION provided by one of its CHILDren.

Ellidiss Technologies
www.ellidiss.com

From Design Rules To Prolog Rules

I-6 Each OPERATION provided by a PARENT shall be IMPLEMENTED_BY one OPERATION provided by one of its CHILDren.

```
/* HRM 4 rule I6:
Each OPERATION provided by a PARENT shall be IMPLEMENTED_BY one
OPERATION provided by one of its CHILDren. */

errI6(X,U) :- isProvided(U,'OPERATION',X),
   not(isImplementedBy(U,'OPERATION',X,_,_)).

checkI :- isNotTerm(X), errI6(X,U),
   write('ERROR : Operation : '), write(U),
   write(' provided by not terminal object : '), write(X),
   write(' should be implemented by a child object ! (I6)'), nl.
checkI :- printlf('---> rule I6 checked...’).

/* utility rules */
isTerm(Y) :- hoodObject(Y,_,_), not(hoodObject(_,_,Y)).
isNotTerm(Y) :-  hoodObject(Y,_,_), isParent(Y).
isParent(Y) :- hoodObject(_,_,Y), !.
```

Ellidiss Technologies
www.ellidiss.com

LMP realizations for HOOD

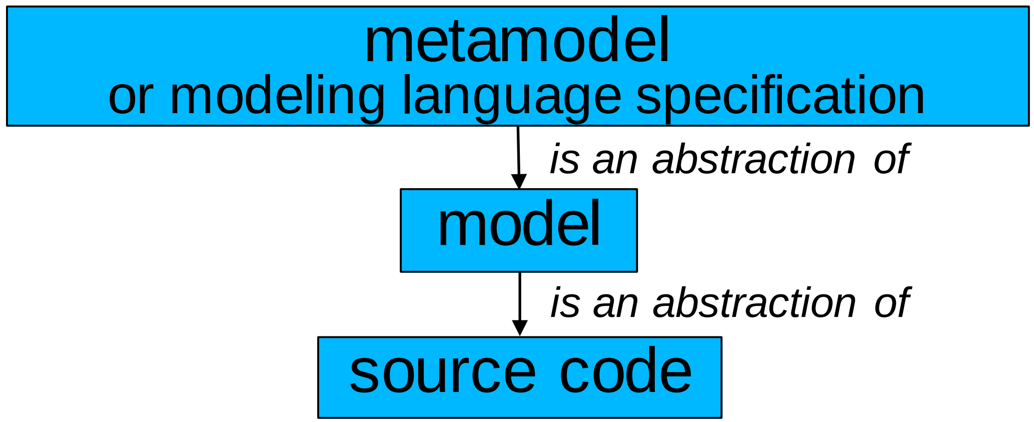
Airbus 340/380/350

Rules Base	Tool	DO 178 Qualification (verification tools)
HOOD to Ada	Stood	
HOOD to C/Asm	Stood	
HOOD Checker	Stood	Yes
HOOD to Document	Stood	
HOOD to Frama-C	Stood	Yes
HOOD to Caveat	Stood	Yes

Ellidiss Technologies
www.ellidiss.com

LMP for Model Driven Engineering

Software Model Driven Engineering



Ex. HOOD

Ex. Ada

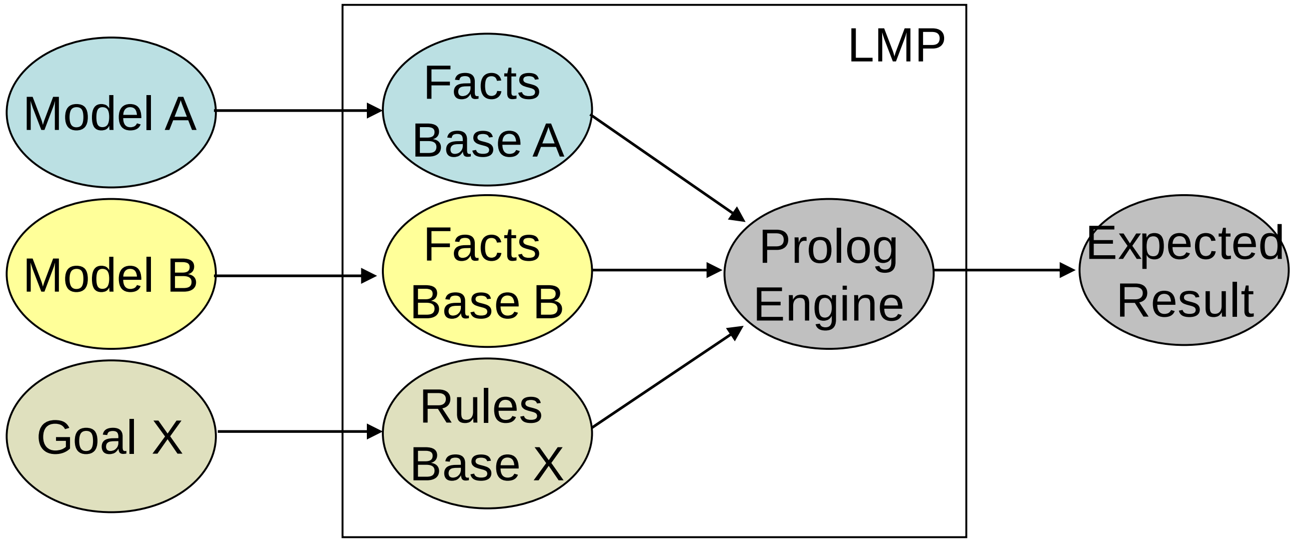
Logic Model Processing applied to Model Driven Engineering

- Observe: model exploration, queries and views
- Verify static properties: rules checkers
- Perform model transformations:
  - Refinement along the development life-cycle (e.g. system to software)
  - Verification: transform a descriptive model into a verification model
  - Source code generation (model to code)
  - Source code reverse engineering (code to model)
  - Documentation generation

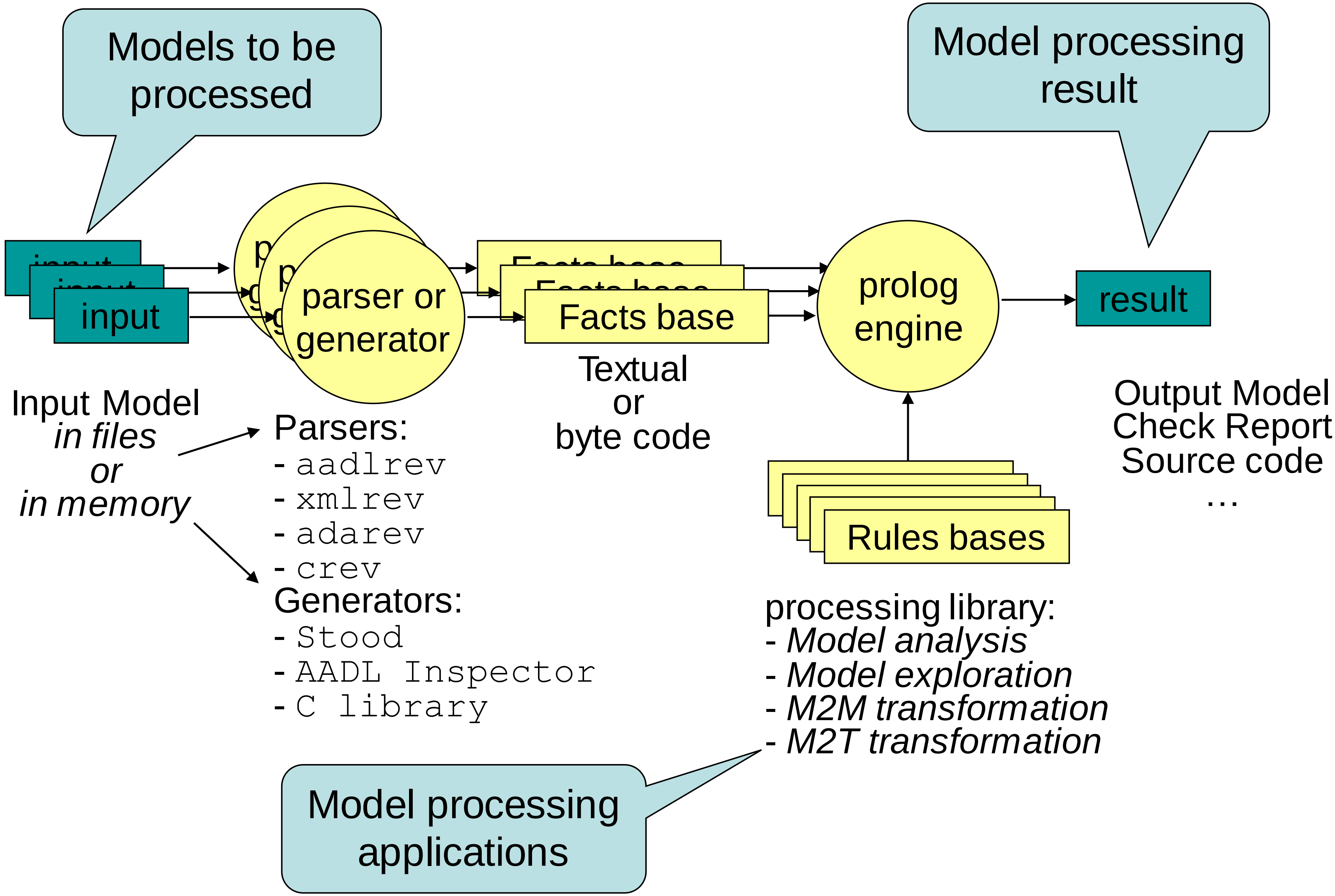
LMP: a generic model processing approach

LMP development and runtime process:

- Any Modeling (or Programing) Language can be represented by a set of Prolog Fact Definitions
- Any Model can be represented by a populated Prolog Facts Base
- Any Model processing action (queries, constraints, transformations, …) can be represented by a Prolog Rules Base
- Prolog facts and rules base can be merged together to get the expected result



LMP runtime process



Ellidiss Technologies
www.ellidiss.com

LMP parsers for tag-based models (XMI/XML) e.g. ReqIF

Fragment of the meta-model: XML Schema Definition (XSD)

```
<xsd:schema ... >
  <xsd:complexType name="SPEC-OBJECT">
    ...
    <xsd:attribute name="DESC" type="xsd:string" use="optional"/>
    <xsd:attribute name="IDENTIFIER" type="xsd:ID" use="required"/>
    <xsd:attribute name="LAST-CHANGE" type="xsd:dateTime" use="required"/>
    <xsd:attribute name="LONG-NAME" type="xsd:string" use="optional"/>
  </xsd:complexType>
  ...
</xsd:schema>
```

Corresponding prolog Fact Definition:

```
isSpecObject(Desc,Identifier,LastChange,LongName).
```

Ellidiss Technologies
www.ellidiss.com

LMP parsers for token based models
e.g. AADL

Fragment of the meta-model: Backus-Naur Form (BNF)

```
…
component_type ::=
        component_category component_identifier
        { property }*
        end component_identifier;
property ::=
        property_name => property_value;
…
```

Corresponding prolog Facts definition:

```
isComponentType(ComponentCategory,ComponentIdentifier).
isProperty(ComponentIdentifier,PropertyName,PropertyValue).
```

Ellidiss Technologies
www.ellidiss.com

LMP: Merging and Processing

1. Merge together the two facts bases:

model A (requirements)

```
isSpecObject('','','','Temp_Lower_Bound').
isSpecObject('','','','Temp_Upper_Bound').
...
isComponentType('Thread','Thermostat').
isProperty('Thermostat','Coverage','Temp_Lower_Bound').
isProperty('Thermostat','Coverage','Temp_Upper_Bound').
```

model B (design)

2. Add the rules base:
check requirements coverage:

```
checkCoverage :-
  isSpecObject(_,_,_,R),
  isComponentType(_,C),
  not(isProperty(C,'Coverage',R)),
  writeErrorMessage(R).
```

rule to check

3. Run the Prolog engine

Ellidiss Technologies
www.ellidiss.com

LMP realizations for AADL(1)

LMP rules	category
AADL semantic rules	model checker
AADL instance builder	model exploration
AADL ARINC 653 rules	model checker
UML MARTE to AADL	model transformation
SysML to AADL	model transformation
Capella to AADL	model transformation
FACE(2) to AADL	model transformation
AADL to Cheddar	model verification (timing analysis)
AADL to Marzhin	model verification (simulation)
AADL to OpenPSA	model verification (safety analysis)
AADL printer	model unparser
LAMP checker	online model processing

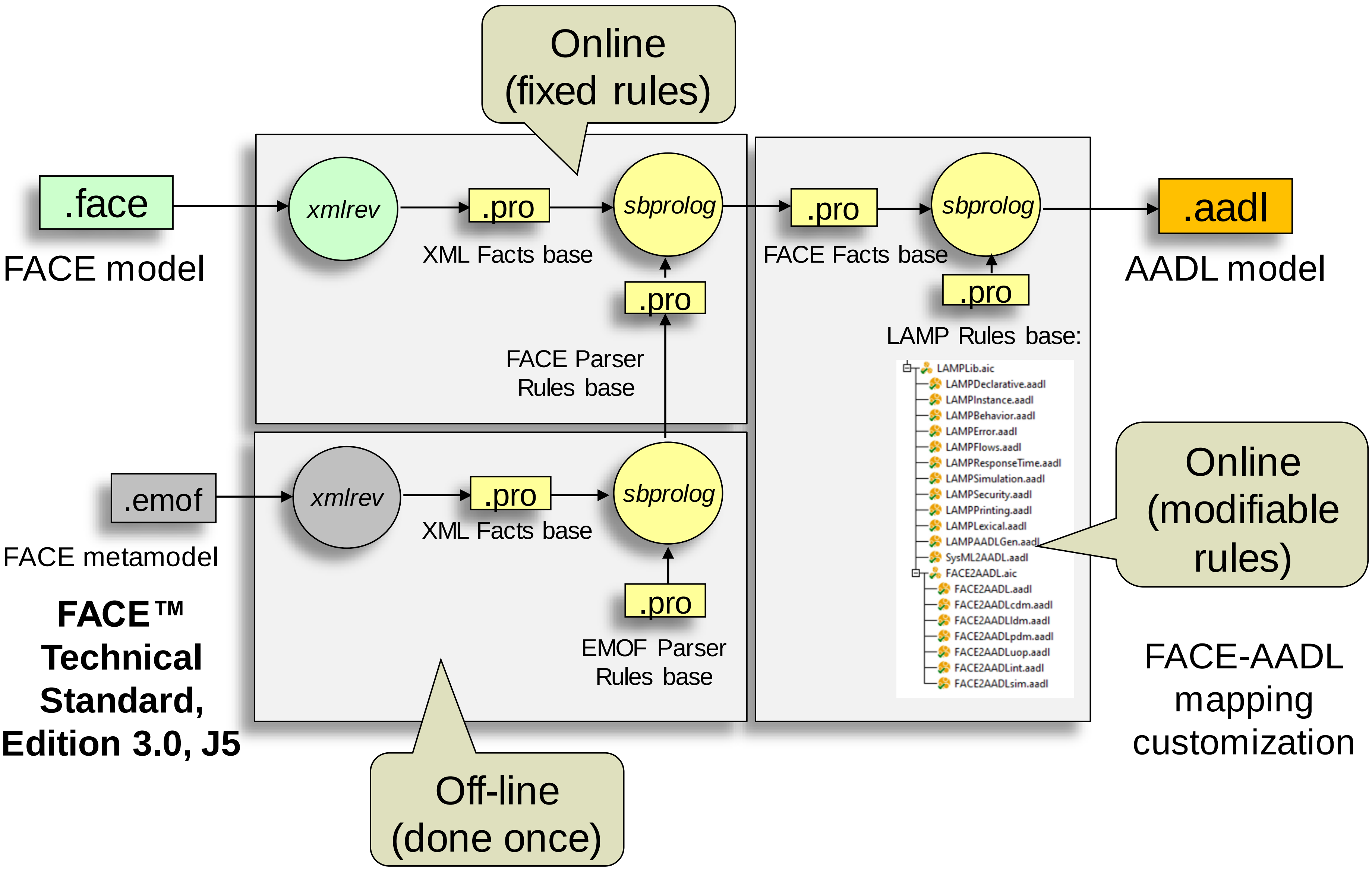
(1) Architecture Analysis and Design Language (AADL) is a modeling standard of the SAE (AS-5506)
(2) FACE is a trademark of the Open Group

Ellidiss Technologies
www.ellidiss.com

LAMP: On-line LMP programs for AADL

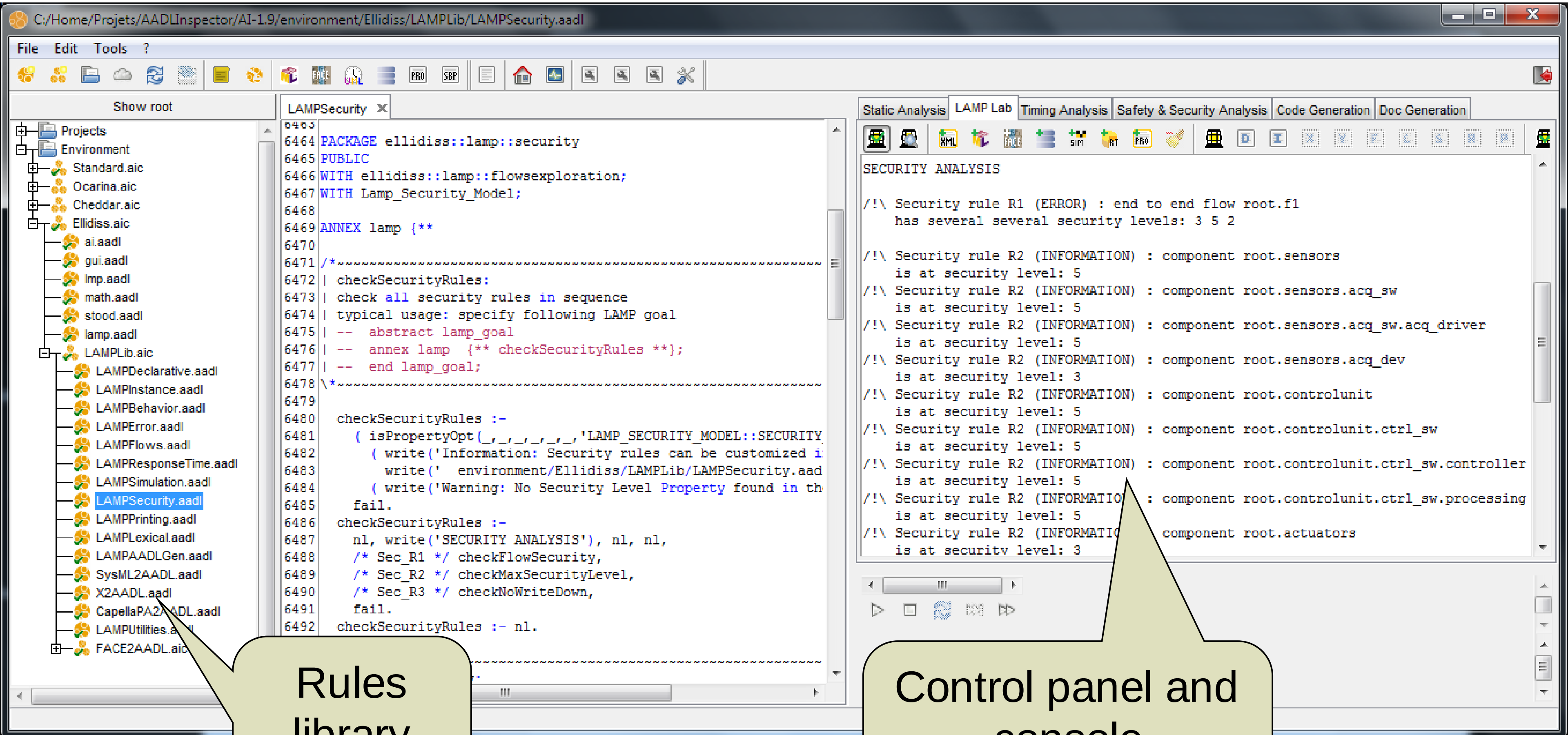
```
package Ellidiss::ERTS2020::paper26::e1
public

abstract A
-- a LAMP goal at component level
   annex LAMP {**
   /* standard prolog syntax */
   checkSecurityRules
   **};
end A;

-- LAMP rules at package level
annex LAMP {**
/* standard prolog syntax */
checkSecurityRules :-
  …
**};

end Ellidiss::ERTS2020::paper26::e1;
```

User-customizable Model Transformations



Ellidiss Technologies
www.ellidiss.com

Tool integration: AADL Inspector



Rules library LAMPLib

Control panel and console LAMPLab

Ellidiss Technologies
www.ellidiss.com

Benefits of the LMP approach

- **Generic** solution for:
  - Model API
  - Model constraints
  - Model transformations
  - Model exploration and architectural reasoning
- **Standard** prolog language (ISO/IEC 13211-1)
- **Independent**: compatible with the main meta-modelling formats (BNF, XSD, MOF, Ecore)
- **Interpreted**: supports both off-line and online processing (e.g. LAMP)
- **Declarative**: fits well with multi-steps incremental development processes
- **Modular**:
  - separate fact and rules bases
  - rules bases transitivity: e.g. SysML to code : SysML to AADL and AADL to code
- **Formal** (boolean logic): appropriate for tool qualification
- **Flexible**:
  - Supports heterogeneous models
  - Supports incomplete models (subsets)
  - Supports erroneous models (debugging)
- **Industrial** return of experience of many off-line processing tools:
  - Airbus: LMP applied for the verification of DO-178 certified projects (A380, A350)
  - European Space Agency: used in the TASTE tool-chain
  - Honeywell: architecture reasoning
  - Ellidiss: AADL Inspector model adaptors and Stood code generators
  - Commercial support
- **Sustainable** project started almost 30 years ago and still relevant and efficient

Ellidiss Technologies
www.ellidiss.com

Conclusion

- The Logic Model Processing (LMP in short) approach has been gradually elaborated during the last three decades.
- These realizations are guided by requirements coming from the aerospace industry.
- It mostly consists of the use of existing Prolog language technology in a particular application domain (implementation of Model Driven Engineering development tools).
- Ellidiss Technologies embeds LMP inside its own commercial products and R&D developments to efficiently implement most of the required « model processing » features.
- Model Driven Engineering is more and more applied in various domains of application (System Engineering) and contributes to the digitalization of the industry. LMP feedback shows that Prolog can play a significant role in this area.