Ellidiss Technologies

www.ellidiss.fr

LAMP
Tutorial

Pierre Dissaux, AADL committee, Paris, 26 June 2019

Ellidiss Technologies
www.ellidiss.fr

Logic Model Processing for AADL

LMP: Logic Model Processing

- Prolog API for modelling and programming languages
  - converts language entities into prolog facts
  - implicit accessors and iterators
  - enables formal processing rules in prolog
  - available methodology and tools (LMP Designer)
- Usage:
  - existing implementation for XML/XMI, AADL, Ada, C
  - used to develop processing plugins for Stood, AADL Inspector, TASTE
  - rules checkers, model transformations, code generators
  - ex: design rules checker for A380/A350, DO178 qualified

LAMP: using LMP as an AADL annex sub-language

- Use an ISO standard language (prolog)
- Declarative style and formal semantics
- Exhaustive coverage of AADL specifications (core + annexes)
- Inherits LMP experience (libraries, know-how)
- Can do all what Resolute can do, and probably much more.

LMP
main principles

1. Input data (model elements) are translated into prolog facts:

```
PACKAGE P PUBLIC THREAD t FEATURES i : IN DATA PORT d; END t; END p;
```

```
isPackage ('p','PUBLIC',1).
isComponentType('p','PUBLIC','t','THREAD','NIL',1).
isFeature('PORT','p','t','i','IN','DATA','d','NIL','NIL',1).
isComponentTypeEnd('p','PUBLIC','t','t',1).
isPackageEnd('p','p',1).
```

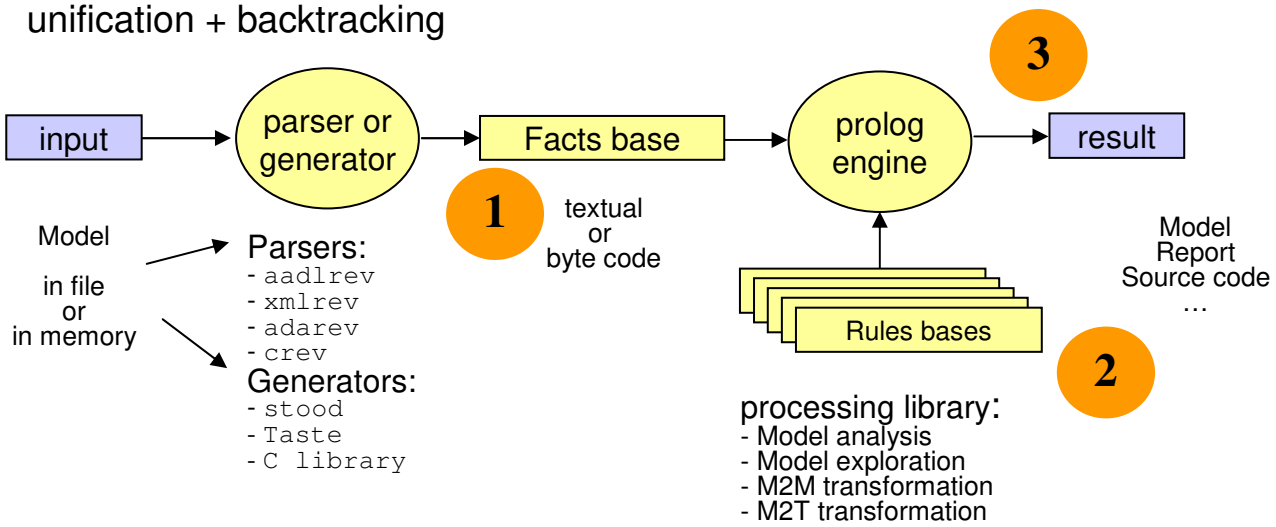
2. Processing rules and libraries are defined with standard prolog syntax:

```
logical AND: comma
logical OR: semicolon
logical NOT: not() operator
```

```
isComponentType(_,_,T,'THREAD',_,_), write(T).
```

3. Execution follows standard prolog semantics:

unification + backtracking



Inside LAMP

LAMP AADL Annex subclause:

- Syntax: `ANNEX LAMP {** /* standard prolog syntax */ **};`
- LAMP user defined libraries in AADL Packages
- LAMP user defined local rules in AADL Components
- No new langage to define and maintain
- Direct access to the LMP low level API (all AADL model elements)
- Can also work on incorrect models (debugging)

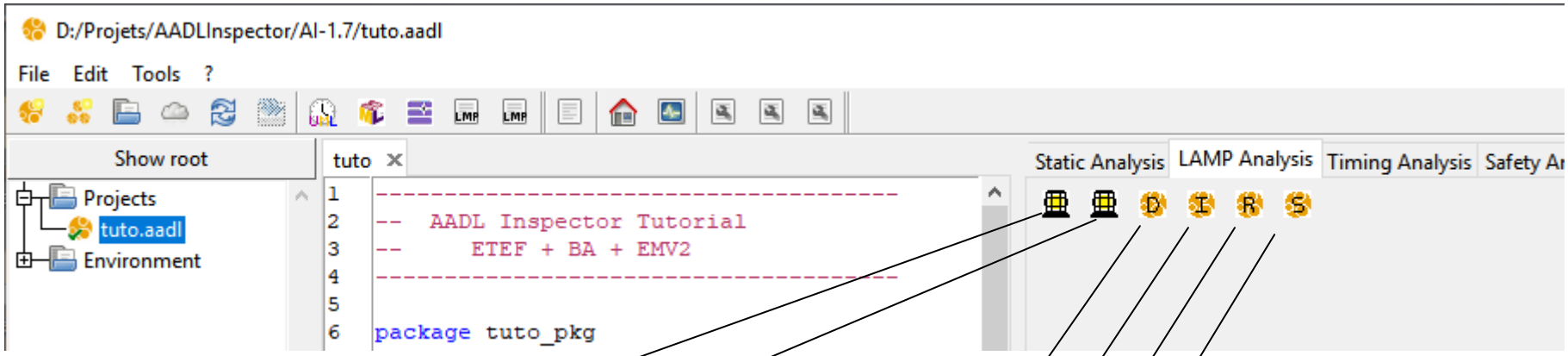
LAMP standard library: LAMPLib.aadl:

- High level API to the AADL declarative model
- High level API to the AADL instance model
- High level API to the Behavior and Error annexes
- API to analysis results (e.g. simulation traces)
- Utility rules (printing, …)

LAMP support inside AADL Inspector

- New LAMP analysis plugin
- LAMPLib is pre-loaded within the AADL "environment"
- Available in A.I. 1.7 (http://www.ellidiss.com/downloads)

Using LAMP



run LAMP checker

run simulation then run LAMP checker

facts:

show low level API (declarative model)

show instance model (part of)

show computed response times

show simulation events

Declarative model facts

```
isVersion('AADL-2.2','aadlrev2.13','(c)Ellidiss Technologies','27Aug2018').

isPackage('tuto_pkg','PUBLIC',7).
isImportDeclaration('tuto_pkg','PUBLIC','ellidiss::math::int',8).
isAliasDeclaration('tuto_pkg','PUBLIC','ALL','NIL','ellidiss::math::int',9).

isComponentType('tuto_pkg','PUBLIC','tuto','SYSTEM','NIL',14).
isAnnex('tuto_pkg','tuto','NIL','unnamed_A1','LAMP','   write('hello'), nl ',15).
isComponentTypeEnd('tuto_pkg','PUBLIC','tuto','tuto',16).

isComponentImplementation('tuto_pkg','PUBLIC','tuto','i','SYSTEM','NIL','NIL',19).
isSubcomponent('tuto_pkg','tuto','i','de1','DEVICE','se','NIL','NIL',21).
isConnection('PORT','tuto_pkg','tuto','i','c1','de1.o','->','sw1.i','NIL',32).
isFlowImplementation('END TO END','tuto_pkg','tuto','i','f',
          'de1.f->c1->sw1.f->c2->sw2.f->c3->sw3.f->c4->de2.f','NIL',38).
isProperty('NIL','APPLIES TO','tuto_pkg','tuto','i','hw1','scheduling_protocol',
          '(Rate_Monotonic_Protocol)',41).

isEMV2Version('EMV2-1.0','aadlrev2.13','(c)Ellidiss Technologies','27Aug2018').
isEMV2UseBehavior('tuto_pkg','tuto','i','unnamed_A1','errorlibrary::failstop',55).
isEMV2CompositeStateElem('tuto_pkg','tuto','i','unnamed_A1','t','$1','de1.failstop','NIL',58).
isEMV2CompositeStateExpr('tuto_pkg','tuto','i','unnamed_A1','t','$3','$1','OR','$2',58).
isEMV2CompositeState('tuto_pkg','tuto','i','unnamed_A1','t',
          'de1.failstop OR de2.failstop OR hw1.failstop OR hw2.failstop OR hw3.failstop OR
           sw1.failstop OR com.failstop OR sw2.failstop OR sw3.failstop','failstop','NIL',58).
isEMV2Property('NIL','APPLIES TO','tuto_pkg','tuto','i','unnamed_A1','NIL','failure',
          'emv2::occurrencedistribution','[probabilityvalue=>1.0e-6;distribution=>poisson;]',62)
isRecordField('tuto_pkg','tuto','i','failure','emv2::occurrencedistribution',1,
          'probabilityvalue','1.0e-6',62).
isRecordField('tuto_pkg','tuto','i','failure','emv2::occurrencedistribution',1,
          'distribution','poisson',62).
isAnnex('tuto_pkg','tuto','i','unnamed_A1','EMV2','',63).

isComponentImplementationEnd('tuto_pkg','PUBLIC','tuto','i','tuto','i',64).
```

more at: http://www.ellidiss.fr/public/wiki/wiki/aadlparser

Instance model facts

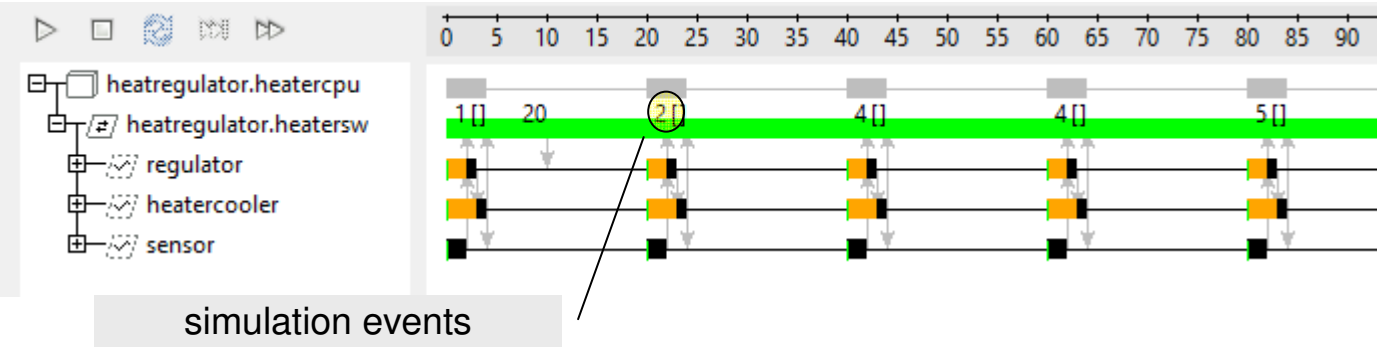
```
isAADLRoot('tuto_pkg','tuto','i','tuto_pkg::tuto.i','root',19).

isAADLInstance('SYSTEM','NIL','NIL','NIL','tuto_pkg::tuto.i','tuto_pkg','tuto','i',
          'NIL','root','SELF',19).
isAADLInstance('DEVICE','tuto_pkg','tuto','i','de1','tuto_pkg','se','NIL',
          'root','root.de1','SELF',21).
isAADLInstance('PROCESSOR','tuto_pkg','tuto','i','hw1','tuto_pkg','hw','NIL',
          'root','root.hw1','SELF',23).
isAADLInstance('BUS','tuto_pkg','tuto','i','com','tuto_pkg','com','NIL',
          'root','root.com','SELF',26).
isAADLInstance('PROCESS','tuto_pkg','tuto','i','sw1','tuto_pkg','sw','i',
          'root','root.sw1','SELF',27).
isAADLInstance('THREAD','tuto_pkg','sw','i','th1','tuto_pkg','t','i',
          'root.sw1','root.sw1.th1','SELF',142).

isAADLProcBinding('PROCESS','root.sw1','root.hw1').
isAADLProcBinding('THREAD','root.sw1.th1','root.hw1').
isAADLProcBinding('THREAD','root.sw1.th2','root.hw1').
isAADLProcBinding('PROCESS','root.sw2','root.hw2').
isAADLProcBinding('THREAD','root.sw2.th1','root.hw2').
isAADLProcBinding('THREAD','root.sw2.th2','root.hw2').
isAADLProcBinding('PROCESS','root.sw3','root.hw3').
isAADLProcBinding('THREAD','root.sw3.th1','root.hw3').
isAADLProcBinding('THREAD','root.sw3.th2','root.hw3').

isAADLBusBinding('CONNECTION','root.c1','root.com').
isAADLBusBinding('CONNECTION','root.c2','root.com').
isAADLBusBinding('CONNECTION','root.c3','root.com').
isAADLBusBinding('CONNECTION','root.c4','root.com').
```

Response time analysis facts

	Deadline	Computed	Max Cheddar	Max Marzhin	Avg Cheddar	Avg Marzhin	Min Cheddar	Min Marzhin
hw1		8.00 %		8.91 %				
sw1								
th1	50	4.00000	2	2	2.00	2.00	2	2
th2	50	2.00000	4	4	4.00	4.00	4	4
hw2		8.00 %		10.06 %				
sw2								
th1	50	4.00000	7	2	7.00	2.00	7	2
th2	50	2.00000	9	4	9.00	4.00	9	4
hw3		8.00 %		9.76 %				
sw3								
th1	50	4.00000	12	2	12.00	2.00	12	2
th2	50	2.00000	14	4	14.00	4.00	14	4
com		4.00 %		5.00 %				
VirtualLink								
c2		1.00000	5	1	5.00	1.00	5	1
c3		1.00000	10	1	10.00	1.00	10	1

```
isWorstCaseResponseTime('root.hw1','root.sw1.th1','4.00000','').

isMaxCheddarResponseTime('root.hw1','root.sw1.th1','2','').
isAvgCheddarResponseTime('root.hw1','root.sw1.th1','2.00','').
isMinCheddarResponseTime('root.hw1','root.sw1.th1','2','').

isMaxMarzhinResponseTime('root.hw1','root.sw1.th1','2','').
isMinMarzhinResponseTime('root.hw1','root.sw1.th1','2','').
isAvgMarzhinResponseTime('root.hw1','root.sw1.th1','2.00','').

isMaxMarzhinComputation('root.hw1','root.sw1.th1','2','').
isMinMarzhinComputation('root.hw1','root.sw1.th1','2','').
isAvgMarzhinComputation('root.hw1','root.sw1.th1','2.00','').

isCheddarUtilization('root.hw1','NIL','0.08000','').
isMarzhinUtilization('root.hw1','NIL','0.0891089108910891','').
```

Simulation events facts




```
isTrace('20190625-095124','tuto_pkg::tuto.i','NIL').

isTraceEvent('20190625-095124','0','PARTITION','root.hw1','root.sw1','RUNNING',
          'NIL','NIL','NIL').
isTraceEvent('20190625-095124','0','THREAD','root.hw1','root.sw1.th1','THREAD_STATE_RUNNING',
          'NIL','NIL','TRUE').
isTraceEvent('20190625-095124','0','THREAD','root.hw1','root.sw1.th2','THREAD_STATE_SUSPENDED',
          'NIL','NIL','FALSE').
isTraceEvent('20190625-095124','0','PROCESSOR','root.hw1','NIL','OCCUPIED',
          'NIL','NIL','NIL').
isTraceEvent('20190625-095124','0','CHANNEL','root.com.com_channel','NIL','RUNNING',
          'NIL','NIL','NIL').
isTraceEvent('20190625-095124','0','BUS','root.com','NIL','AVAILABLE',
          'NIL','NIL','NIL').
isTraceEvent('20190625-095124','1','OUT DATA PORT','root.hw1','root.sw1.th1.o','NIL',
          '1','NIL','NIL').
isTraceEvent('20190625-095124','2','THREAD','root.hw1','root.sw1.th1','THREAD_STATE_SUSPENDED',
          'NIL','NIL','FALSE').
isTraceEvent('20190625-095124','2','THREAD','root.hw1','root.sw1.th2','THREAD_STATE_RUNNING',
          'NIL','NIL','TRUE').
isTraceEvent('20190625-095124','2','IN DATA PORT','root.hw1','root.sw1.th2.i','NIL',
          '0','NIL','NIL').
isTraceEvent('20190625-095124','3','OUT DATA PORT','root.hw1','root.sw1.th2.o','NIL',
          '2','NIL','NIL').
```

LAMP Std Lib (1/4)

Show root

- Projects
  - tuto.aadl
- Environment
  - Standard.aic
  - Ocarina.aic
  - Ellidiss.aic
    - ai.aadl
    - gui.aadl
    - lmp.aadl
    - math.aadl
    - stood.aadl
    - LAMPLib.aic
      - LAMPDeclarative.aadl
      - LAMPInstance.aadl
      - LAMPBehavior.aadl
      - LAMPError.aadl
      - LAMPFlows.aadl
      - LAMPAnalysis.aadl
      - LAMPPrinting.aadl

LAMPAnalysis

```
3765 ---------------------------------------------
3766 -- LAMP standard library v1.0
3767 -- (c) Ellidiss Technologies 2018
3768 -- Author: Pierre Dissaux
3769 ---------------------------------------------
3770 -- ellidiss::lamp::analysis
3771 -- Examples of LAMP static analysis rules
3772 ---------------------------------------------
3773
3774 PACKAGE ellidiss::lamp::analysis
3775 PUBLIC
3776
3777 ANNEX lamp {**
3778
3779 /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*\
3780 | getMaxResponseTime(Id,Duration,Method):
3781 | returns the computed worst case response time for given
3782 | thread or  bus message instance Id, using the specified
3783 | computation method.
3784 | - Id       (+): thread or message instance identifier (e.g. 'roo
3785 | - Duration (-): integer number (related to the reference time ur
3786 | - Method   (+): 1 uses Cheddar theoritical tests
3787 |                 2 uses Cheddar simulation
3788 |                 3 uses Marzhin simulation
3789 \*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
3790
3791   getMaxResponseTime(Id,Duration,1) :-
3792     isWorstCaseResponseTime(_,Id,D,_),
3793     floatToInt(D,Duration).
3794
3795   getMaxResponseTime(Id,Duration,2) :-
3796     isMaxCheddarResponseTime(_,Id,Duration,_).
3797
3798   getMaxResponseTime(Id,Duration,3) :-
3799     isMaxMarzhinResponseTime(_,Id,Duration,_).
3800
```

LAMP Std Lib (2/4)

high lever core language accessors

returns all the features of a classifier (component type and ancestors)

returns all the subcomponents of a classifier (component implementation and ancestors)

returns all the properties of a classifier (comp. type and impl. and ancestors.)

```
package Ellidiss::LAMP::Declarative public
annex LAMP {**
  getClassFeatures(Class,Name,Categ,FClass) :- …
  getClassSubcomponents(Class,Name,Categ,SClass) :- …
  getLocalProperties(Class,Name,Val,Owner) :- …
  … **};
end Ellidiss::LAMP::Declarative;
```

returns the root classifier (component implementation)

returns all the component instances

returns all the feature instances

returns all the properties for the given instance element

```
package Ellidiss::LAMP::Instance public
annex LAMP {**
  getRoot(Class) :-
  getInstances(Id,Categ,Class) :- … /* components */
  getInstances(Id,Categ,Class) :- … /* features */
  getProperties(Id,Class,Prop,Val) :- …
  … **};
end Ellidiss::LAMP::Instance;
```

LAMP Std Lib (2/4)

high level annexes accessors

```
package Ellidiss::LAMP::BA2 public
annex LAMP {**
  getBAVariables(Class,Name,VClass,Value) :- …
  getBAProperties(Class,Variable,Property,Val) :- …
  getBAStates(Class,Name,Initial,Complete,Final) :- …
  getBATransitions(Class,Name,Source,Destination,Priority) :- …
  getBADispatch(Class,Condition) :- …
  getBACondition(Class,Transition,Condition) :- …
  getBAComputation(Class,Duration) :- …
  …  **};
end Ellidiss::LAMP::BA2;
```

```
package Ellidiss::LAMP::EMV2 public
annex LAMP {**
  getErrorTypes(Class,Name,Ancestor) :- …
  getErrorEvents(Class,Name,Kind,TypeSet,Condition) :- …
  getErrorStates(Class,Name,TypeSet,Kind) :- …
  getErrorTransitions(Class,Name,Source,TypeSet,
    Condition,Target,TypeInst) :- …
  getErrorTransitionBranches(Class,Name,Target,TypeInst,
    Probability):- …
  … **};
end Ellidiss::LAMP::EMV2;
```

LAMP Std Lib (4/4)

utilities

```
/* strings */
splitName(Dotted,Last,Others) :- …
splitFirst(Dotted,First,Others) :- …
concat(First,Second,…,Last,All) :- …
memezTra(String1,string2) :- …
toLower(String1,string2) :- …
toUpper(String1,string2) :- …
```

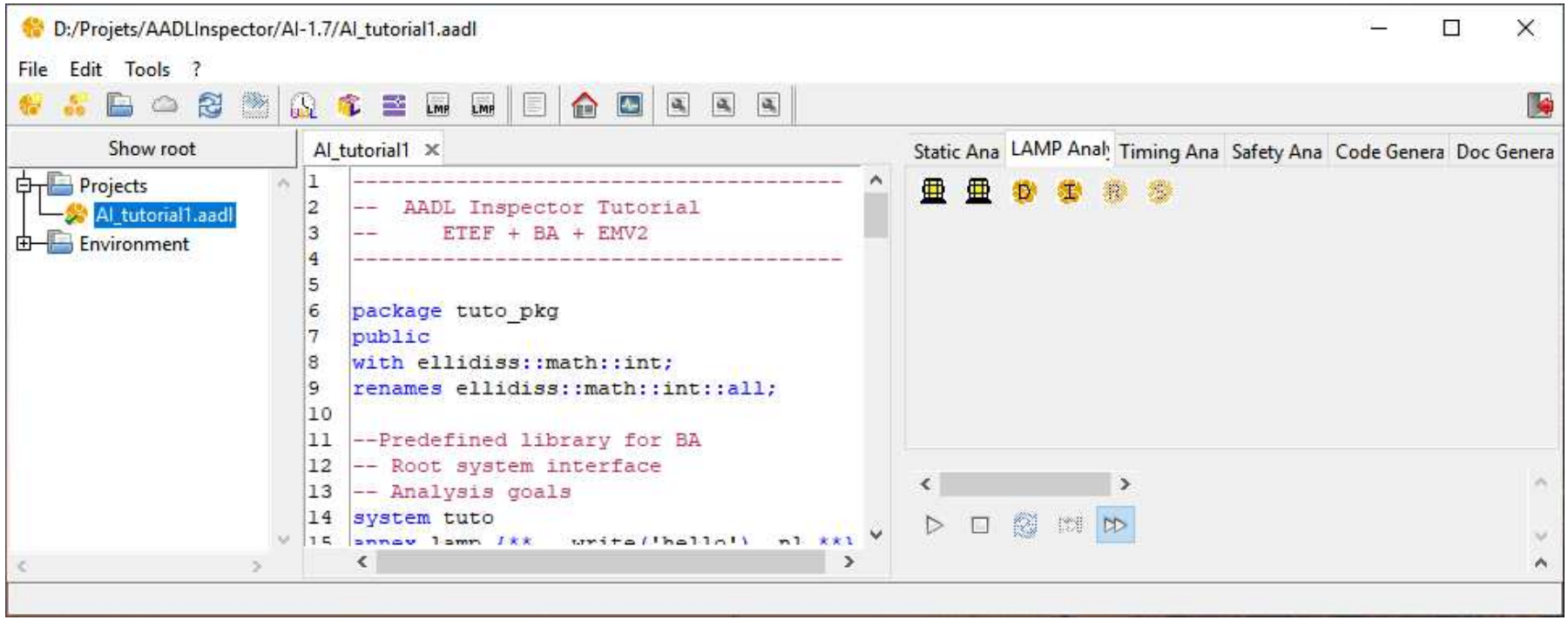
```
/* files */
insertFile(Filename,Option) :- …
pathName(Path,Dir,File) :- …
nl :- …
sp :- …
dot :- …
sc :- …
indent(Int) :- …
```

```
/* numbers */
strToNum(String,Num) :- …
intToList(Int,List) :- …
intToStr(Int,String) :- …
floatToInt(Float,Int) :- …
```

LAMP tutorial

Preparation:

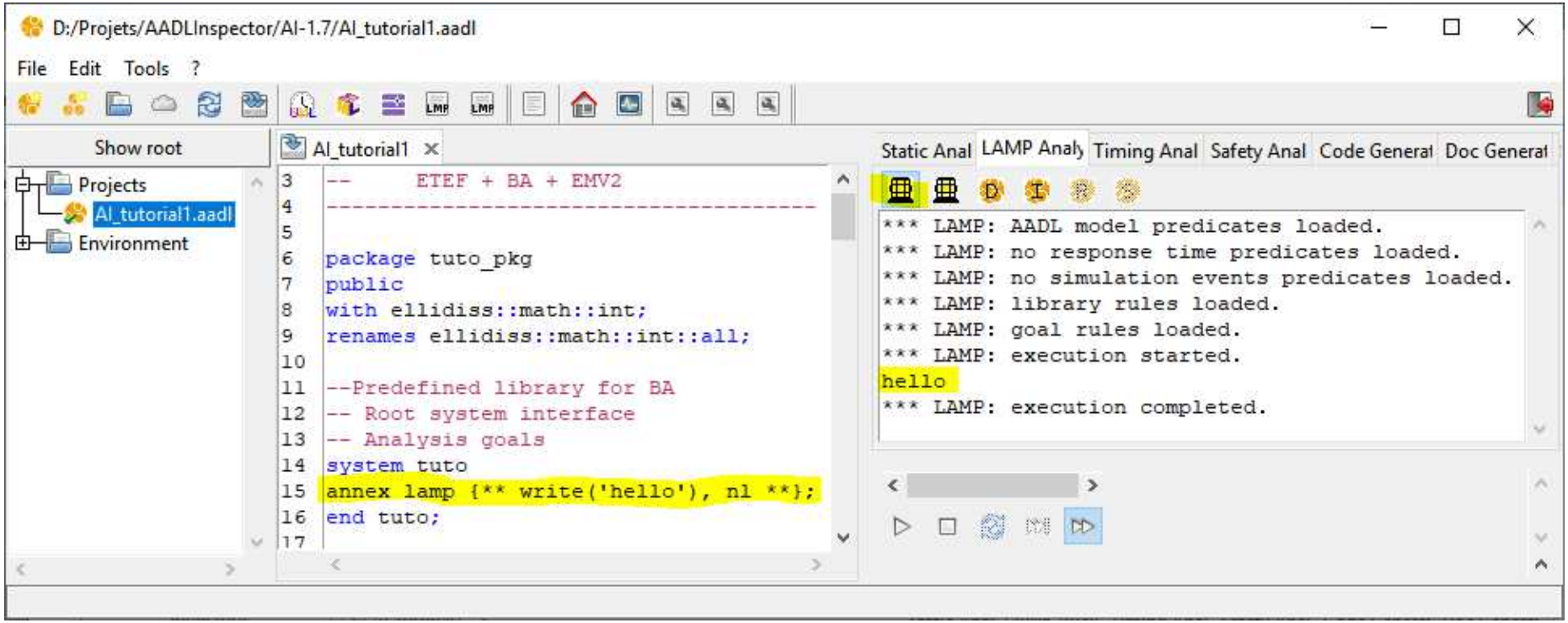
- Install AADL Inspector 1.7
- Add tutorial use case: AI_tutorial1.aadl
- Launch AADL Inspector
- Load AI_tutorial1.aadl
- Select LAMP analysis tab



LAMP exercice 1

First step:

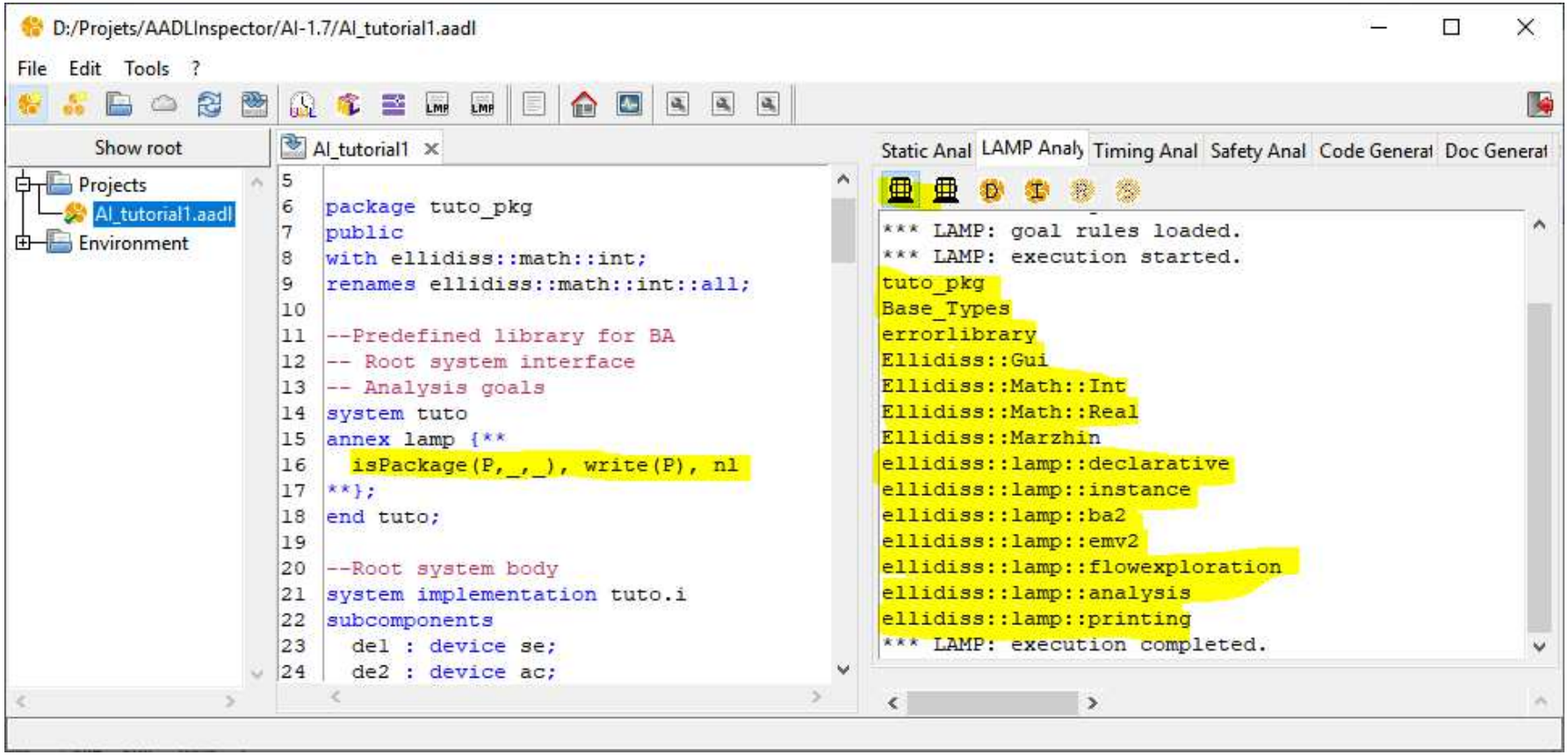
- Add a LAMP annex subclause
- Insert a simple "hello world" printing rule
- Launch LAMP checker
- Show the result



LAMP exercice 2

Queries:

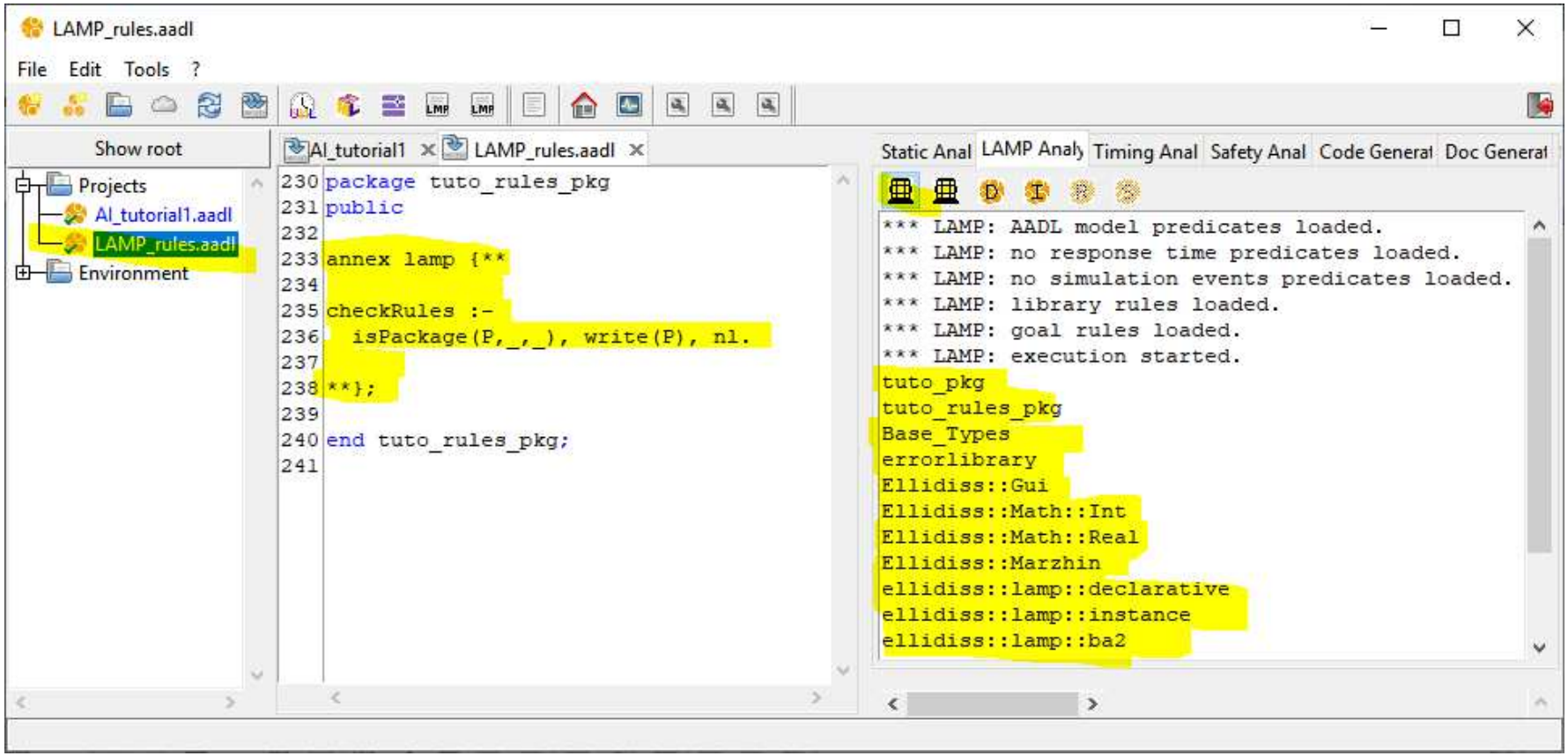
- What are the visible AADL packages ?
- Change the LAMP annex contents
- Launch LAMP checker
- Show the result



LAMP exercice 3

Rules libraries:

- Create a new AADL file declaring a new Package
- Add a new LAMP annex library sub-clause with query of ex. 2
- Change original LAMP goal in main model: `{** checkRules **}`
- Launch LAMP checker
- Show the result



LAMP exercice 4

Search properties:

- Write rules to get all AADL property associations within the project
- Launch LAMP checker
- Show the result

LAMP_rules × LAMPDeclarative × LAMPInstance × LA

```
2055 package tuto_rules_pkg
2056 public
2057
2058 annex lamp {**
2059
2060 checkRules :-
2061   coreProperties,
2062   printDashLine,
2063   emv2Properties,
2064   printDashLine,
2065   instanceProperties.
2066
2067 coreProperties :-
2068   isProperty(_,_,P,T,I,E,Q,V,_),
2069   setClassifier(P,T,I,C),
2070   printProperty(C,E,Q,V),
2071   fail.
2072 coreProperties.
2073
```

Static Analysis | LAMP Analysis | Timing Analysis | Safety Analysis | Code Generation | Doc Generation | Scripts

```
*** LAMP: execution started.
scheduling_protocol => (Rate_Monotonic_Protocol) applies to hw1 in: tuto_pkg::tuto.i
scheduling_protocol => (Rate_Monotonic_Protocol) applies to hw2 in: tuto_pkg::tuto.i
scheduling_protocol => (Rate_Monotonic_Protocol) applies to hw3 in: tuto_pkg::tuto.i
actual_processor_binding => (REFERENCE(hw1)) applies to sw1 in: tuto_pkg::tuto.i
actual_processor_binding => (REFERENCE(hw2)) applies to sw2 in: tuto_pkg::tuto.i
actual_processor_binding => (REFERENCE(hw3)) applies to sw3 in: tuto_pkg::tuto.i
actual_connection_binding => (REFERENCE(com)) applies to c1 in: tuto_pkg::tuto.i
actual_connection_binding => (REFERENCE(com)) applies to c2 in: tuto_pkg::tuto.i
actual_connection_binding => (REFERENCE(com)) applies to c3 in: tuto_pkg::tuto.i
actual_connection_binding => (REFERENCE(com)) applies to c4 in: tuto_pkg::tuto.i
timing => immediate applies to c1 in: tuto_pkg::tuto.i
timing => immediate applies to c2 in: tuto_pkg::tuto.i
timing => immediate applies to c3 in: tuto_pkg::tuto.i
timing => immediate applies to c4 in: tuto_pkg::tuto.i
timing => IMMEDIATE applies to c2 in: tuto_pkg::sw.i
dispatch_protocol => Periodic applies to th1 in: tuto_pkg::sw.i
```