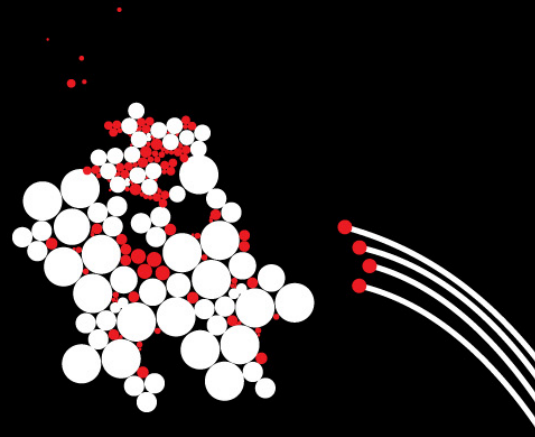
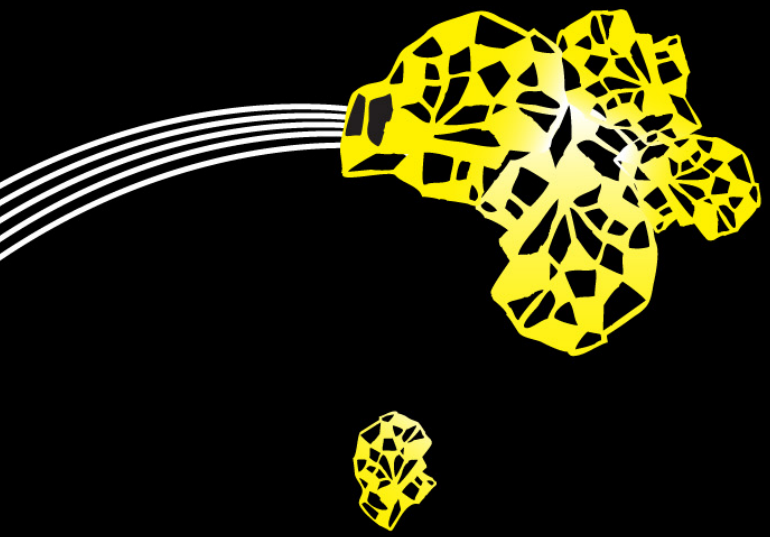
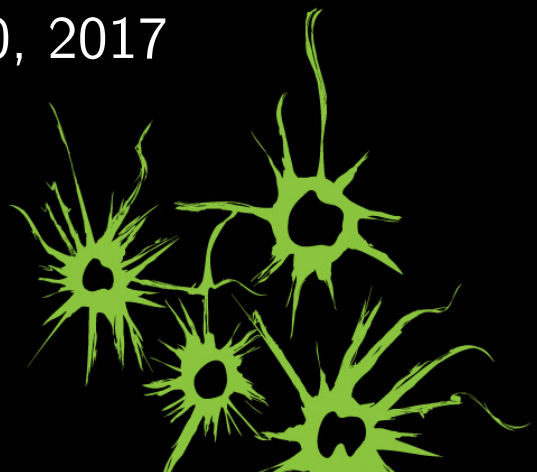
UNIVERSITY OF TWENTE.



Model Transformations applied to Process Calculi



Djurre van der Wal
Data61 (CSIRO), University of Twente
November 20, 2017



Process algebra

Process algebra
= Algebraic tool for describing agent interactions at a high level

Process algebra

Process algebra
= Algebraic tool for describing agent interactions at a high level

General-purpose	**Domain-specific**
Versatile. Available modeling tools. Large user-base.	Easy to learn and use. Domain rules apply automatically. No distracting supportive elements.

Process algebra

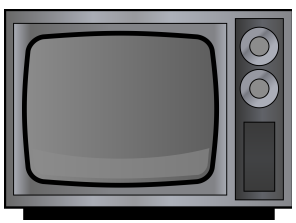
Process algebra
= Algebraic tool for describing agent interactions at a high level

General-purpose	**Domain-specific**
Versatile. Available modeling tools. Large user-base.	Easy to learn and use. Domain rules apply automatically. No distracting supportive elements.

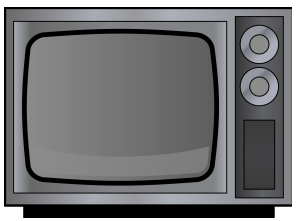
mCRL2 mini Common Representation Language 2
AWN Algebra for Wireless Networks

mCRL2 by example

```
Volume(v: Integer) =
    decVolume . Volume(v > 0 ? v - 1 : 0)
  + incVolume . Volume(v < 30 ? v + 1 : 30);
```

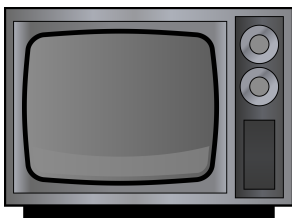


mCRL2 by example



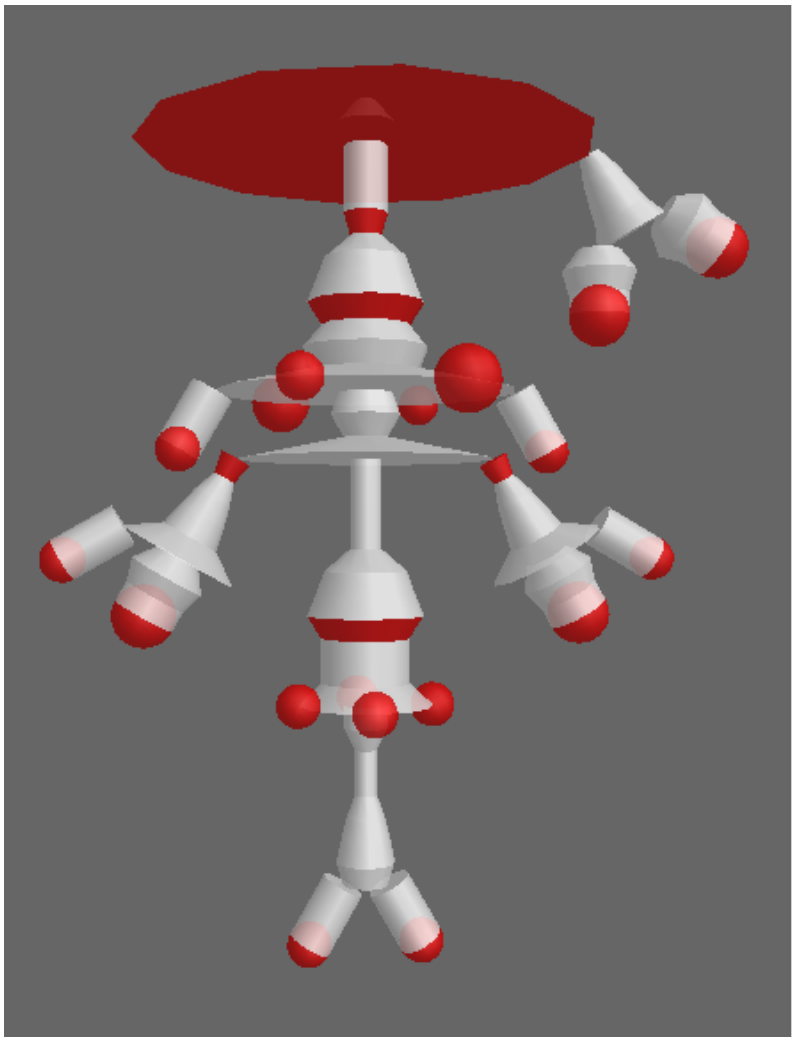
```
Volume(v: Integer) =
    decVolume . Volume(v > 0 ? v - 1 : 0)
  + incVolume . Volume(v < 30 ? v + 1 : 30);

Channel(c: Integer) =
  sum n1:{0..9} . press(n1) .
    sum n2:{0..9} . press(n2) .
      Channel(n1 * 10 + n2);
```

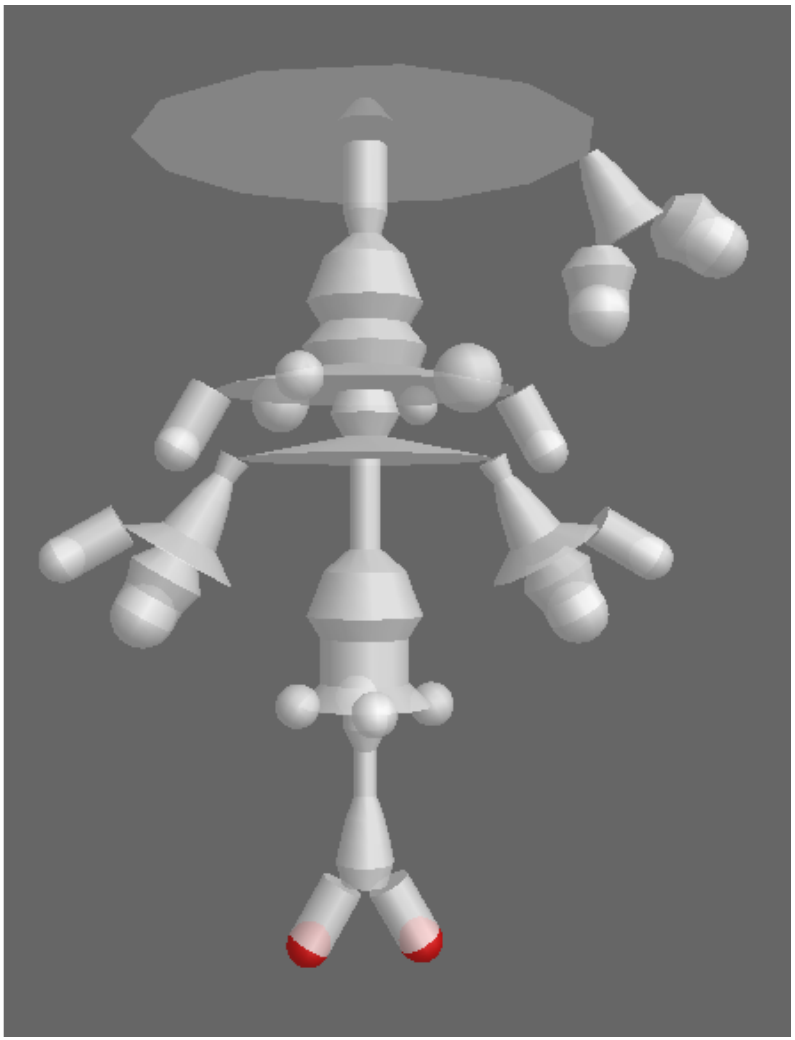
mCRL2 by example



```
Volume(v: Integer) =
    decVolume . Volume(v > 0 ? v - 1 : 0)
  + incVolume . Volume(v < 30 ? v + 1 : 30);

Channel(c: Integer) =
  sum n1:{0..9} . press(n1) .
    sum n2:{0..9} . press(n2) .
      Channel(n1 * 10 + n2);

TV = Channel(1) || Volume(15);
```

mCRL2 by example





AWN by example

```
RadioTower(m: Integer) =
  broadcast(Message(m)) . RadioTower(m + 1);
```

AWN by example

```
RadioTower(m: Integer) =
  broadcast(Message(m)) . RadioTower(m + 1);

Radio(last: Message) = receive(m) . Radio(m);
```

AWN by example

```
RadioTower(m: Integer) =
  broadcast(Message(m)) . RadioTower(m + 1);

Radio(last: Message) = receive(m) . Radio(m);

RadioNetwork = [ 1 : RadioTower(1) : { 2, 4 } ||
                 2 : Radio(null) : {} ||
                 3 : Radio(null) : {} ||
                 4 : Radio(null) : {} ];
```

Main goal

Make an automated translation from AWN to mCRL2.

From AWN to mCRL2

What would a typical implementation look like?

From AWN to mCRL2

What would a typical implementation look like?

AWN text

From AWN to mCRL2

What would a typical implementation look like?

AWN text $\rightarrow$ AWN data

From AWN to mCRL2

What would a typical implementation look like?

AWN text $\rightarrow$ AWN data $\rightarrow$ mCRL2 data

From AWN to mCRL2

What would a typical implementation look like?

AWN text $\rightarrow$ **AWN data** $\rightarrow$ **mCRL2 data** $\rightarrow$ **mCRL2 text**

From AWN to mCRL2

What would a typical implementation look like?

AWN text $\rightarrow$ **AWN data** $\rightarrow$ **mCRL2 data** $\rightarrow$ **mCRL2 text**

Model-driven engineering (=MDE):

- Distribute implementation over *(meta-)models*.
- Transform model to model as required.
- Use DSLs specialized for the type of transformation.
- Benefit from peripheral tools.

From AWN to mCRL2

What would a typical implementation look like?

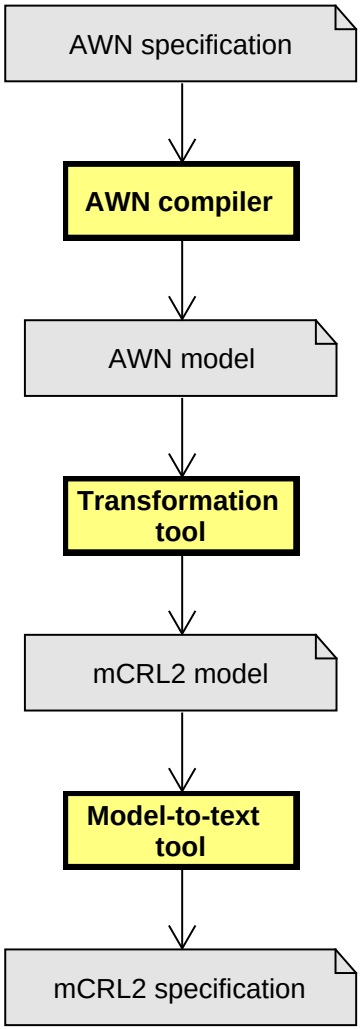
AWN text $\rightarrow$ **AWN data** $\rightarrow$ **mCRL2 data** $\rightarrow$ **mCRL2 text**

Model-driven engineering (=MDE):

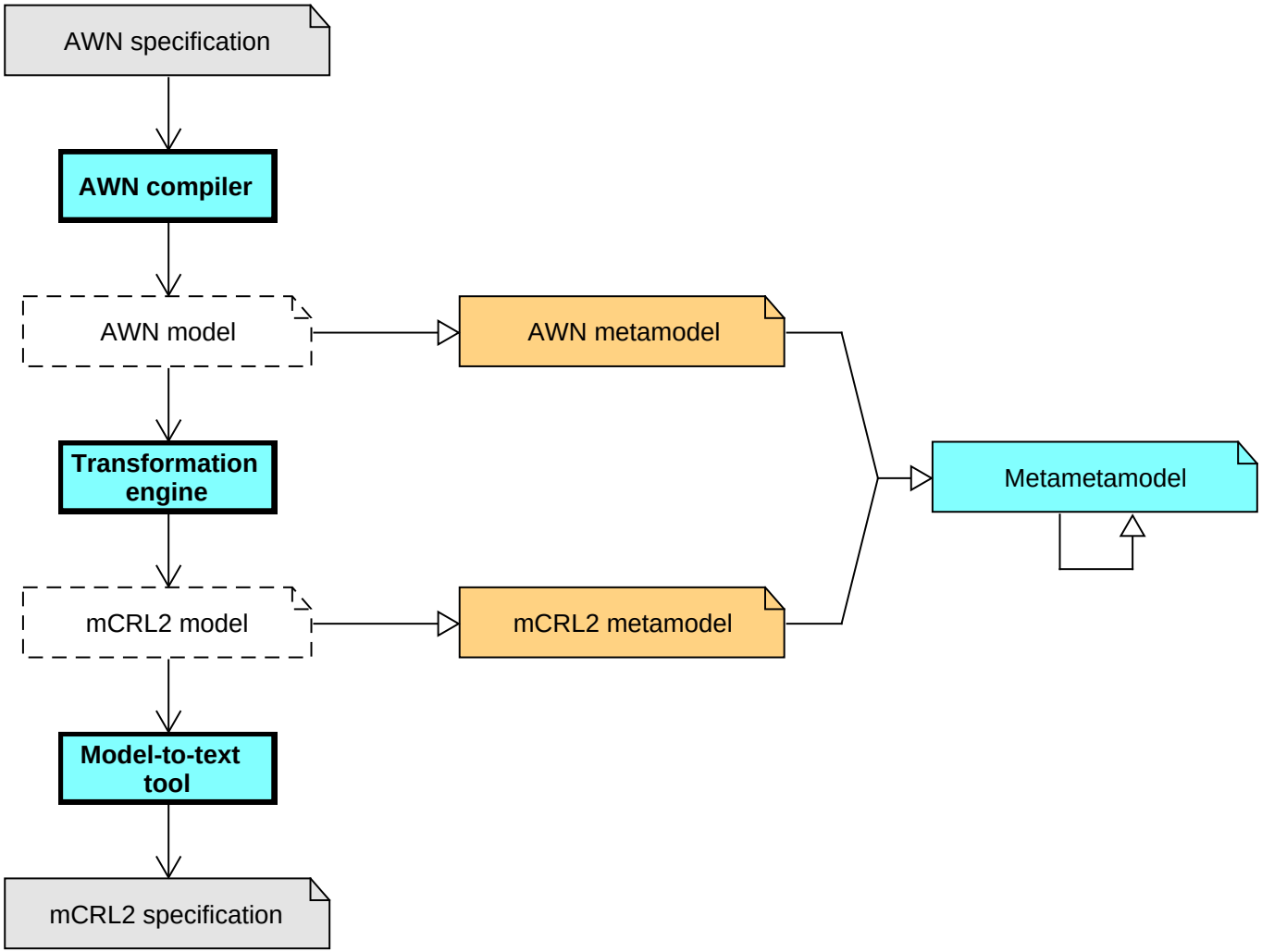
- Distribute implementation over *(meta-)models*.
- Transform model to model as required.
- Use DSLs specialized for the type of transformation.
- Benefit from peripheral tools.

Model
= Bunch of related objects organized according to a metamodel

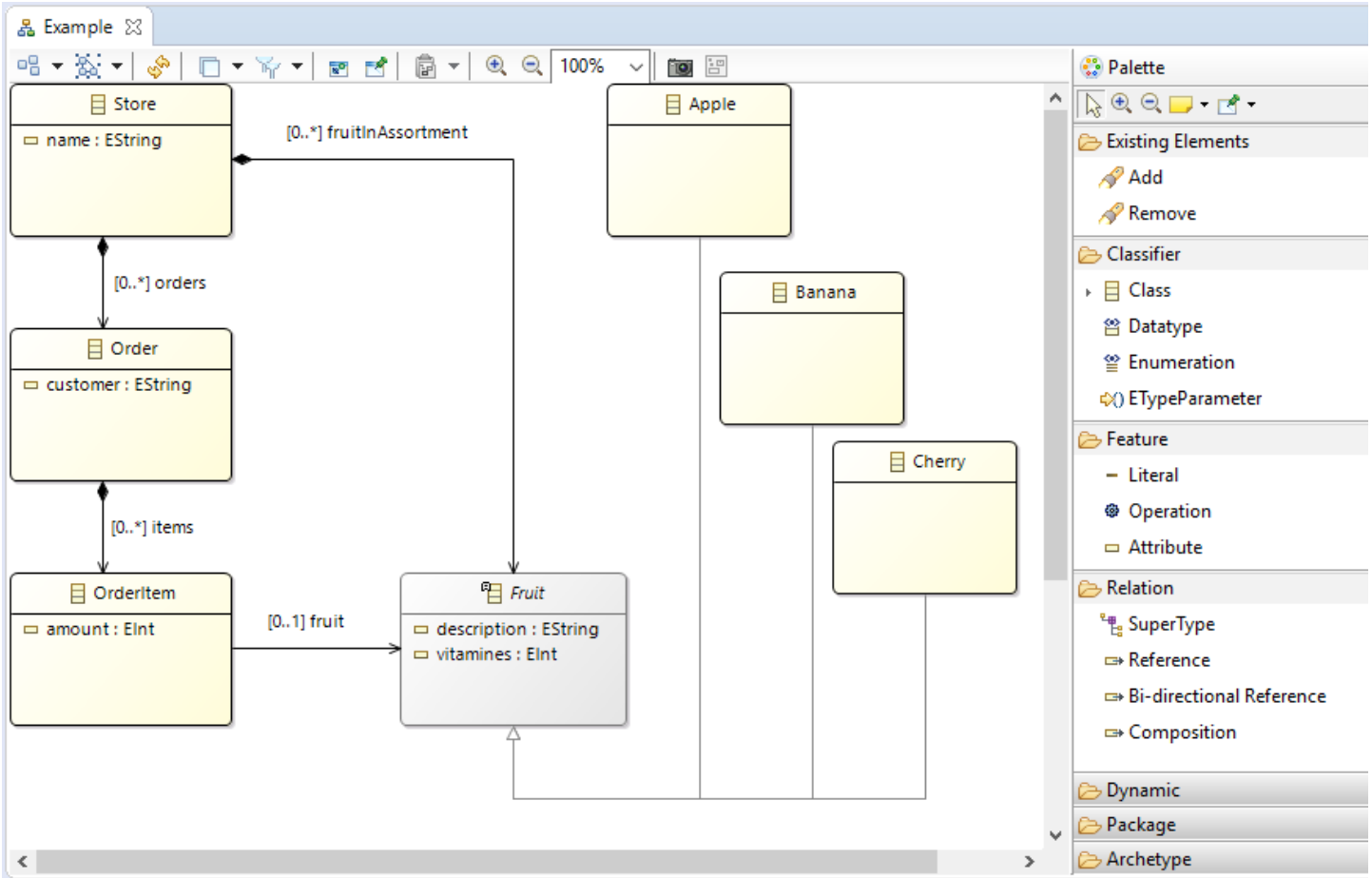
From AWN to mCRL2 using MDE



From AWN to mCRL2 using MDE



Edit metamodels

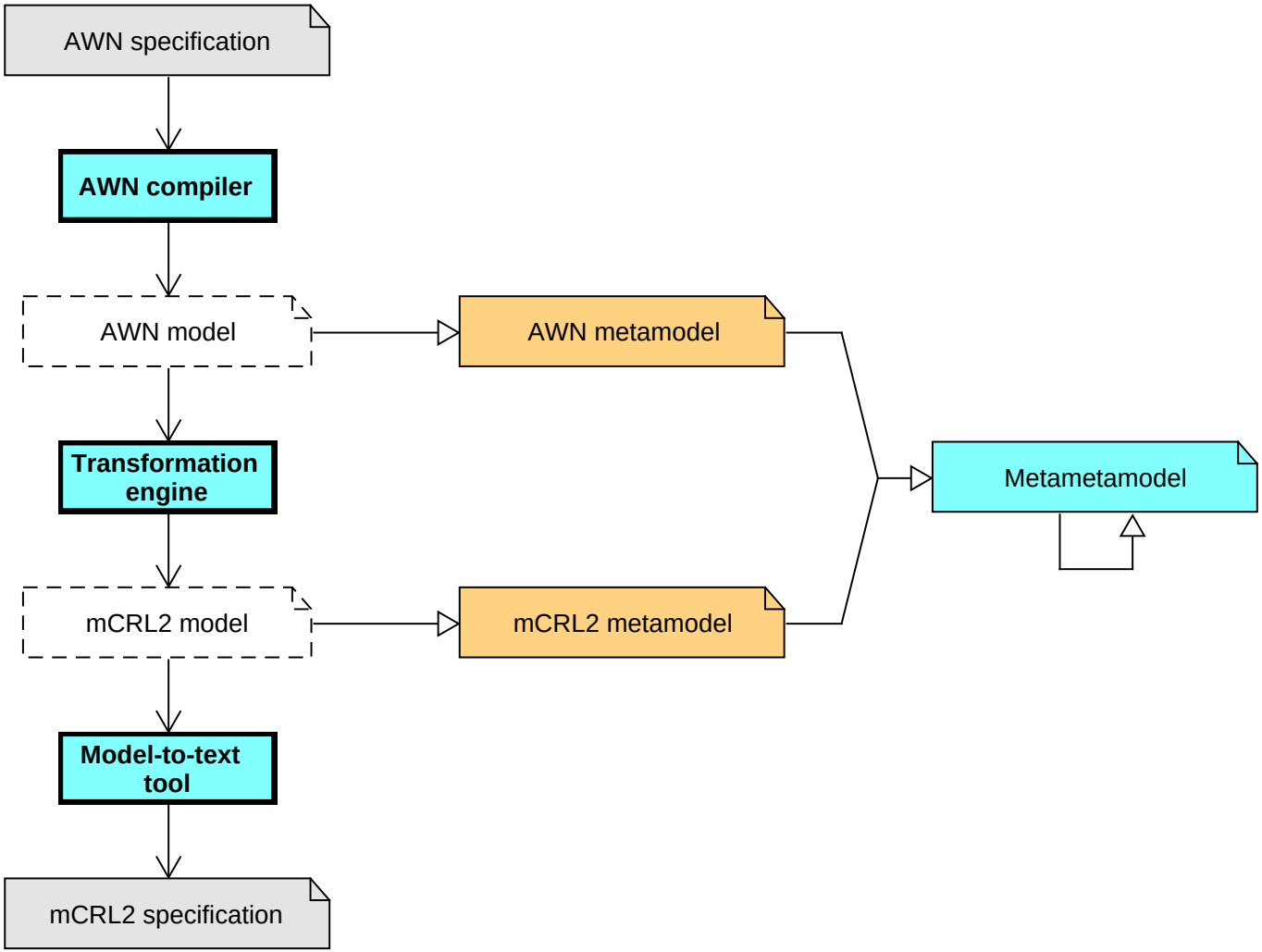


Edit metamodels

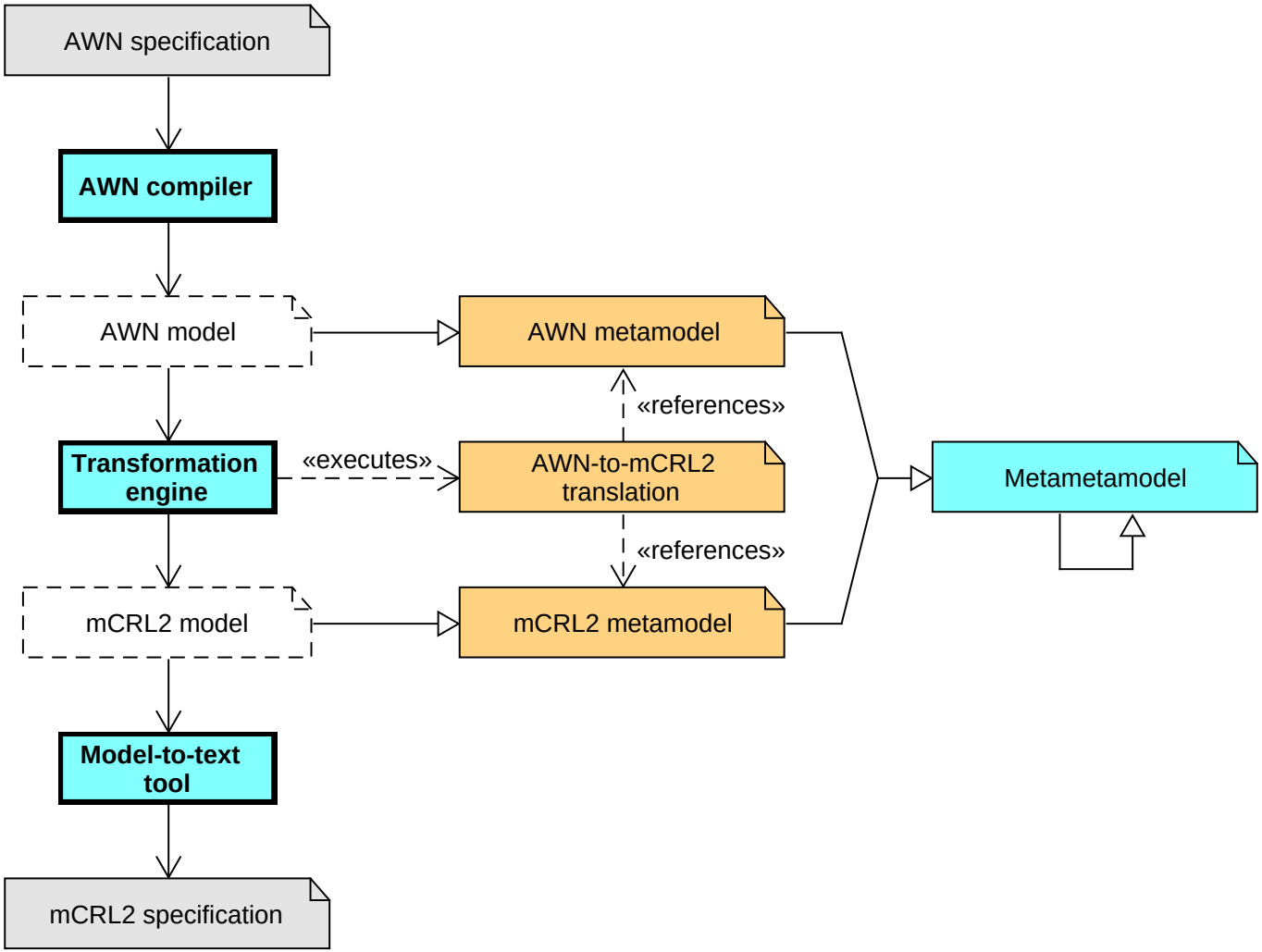
Example | Example.ecore

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore' ;

package Example
{
    abstract class Fruit
    {
        attribute description : String[?];
        attribute vitamines : ecore::EInt[?];
    }
    class Apple extends Fruit;
    class Banana extends Fruit;
    class Cherry extends Fruit;
    class OrderItem
    {
        attribute amount : ecore::EInt[?];
        property fruit : Fruit[?];
    }
    class Order
    {
        property items : OrderItem[*|1] { ordered composes };
        attribute customer : String[?];
    }
    class Store
    {
        property fruitInAssortment : Fruit[*|1] { ordered composes };
        property orders : Order[*|1] { ordered composes };
        attribute name : String[?];
    }
}
```

From AWN to mCRL2 using MDE



From AWN to mCRL2 using MDE



AWN to mCRL2 transformation

$T(p + q) = T(p) + T(q)$

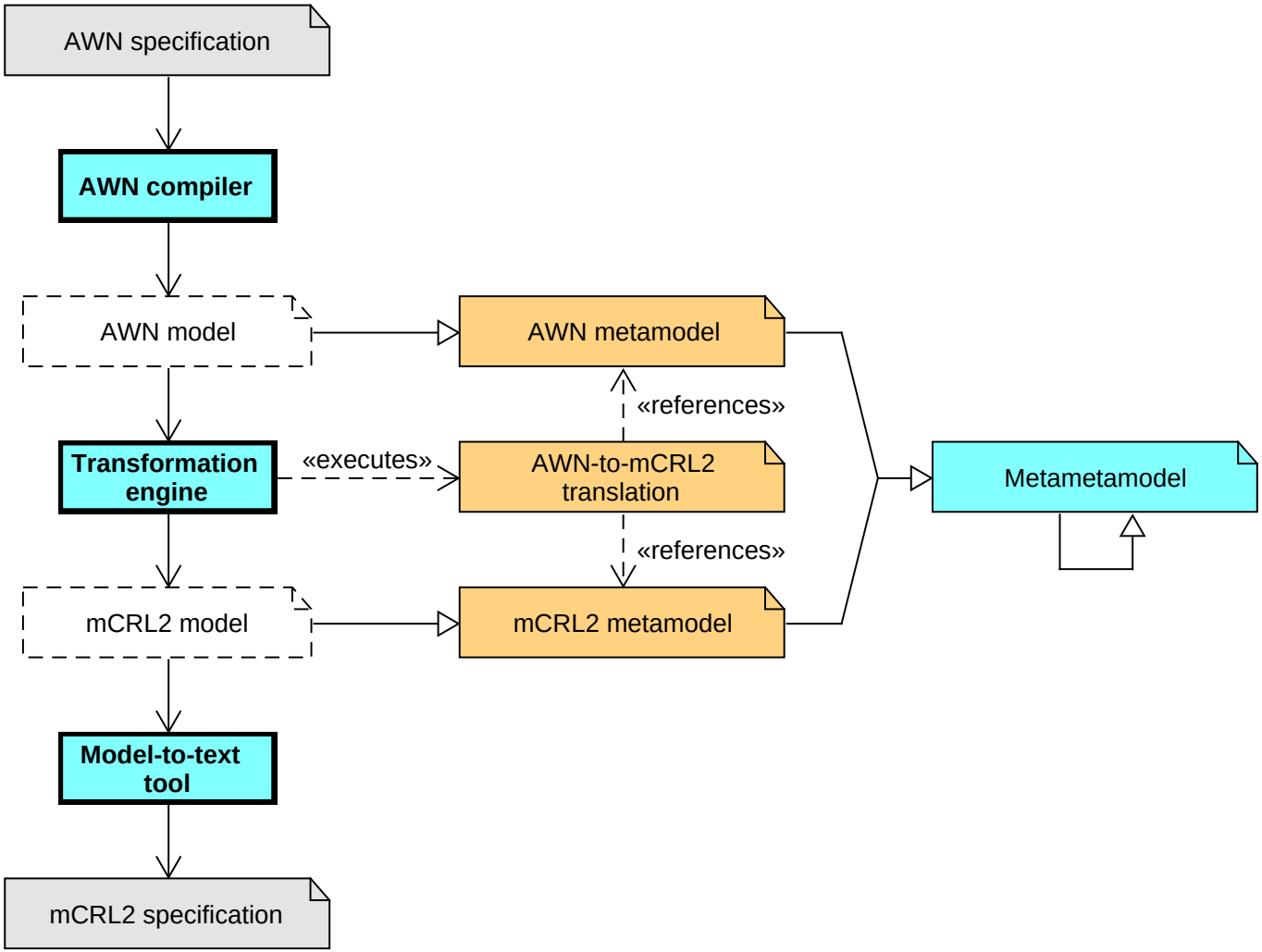
```
mapping AWN::Choice::transform() : MCRL2::Choice {
  lhs := self.lhs.map transform();
  rhs := self.rhs.map transform();
}
```

AWN to mCRL2 transformation

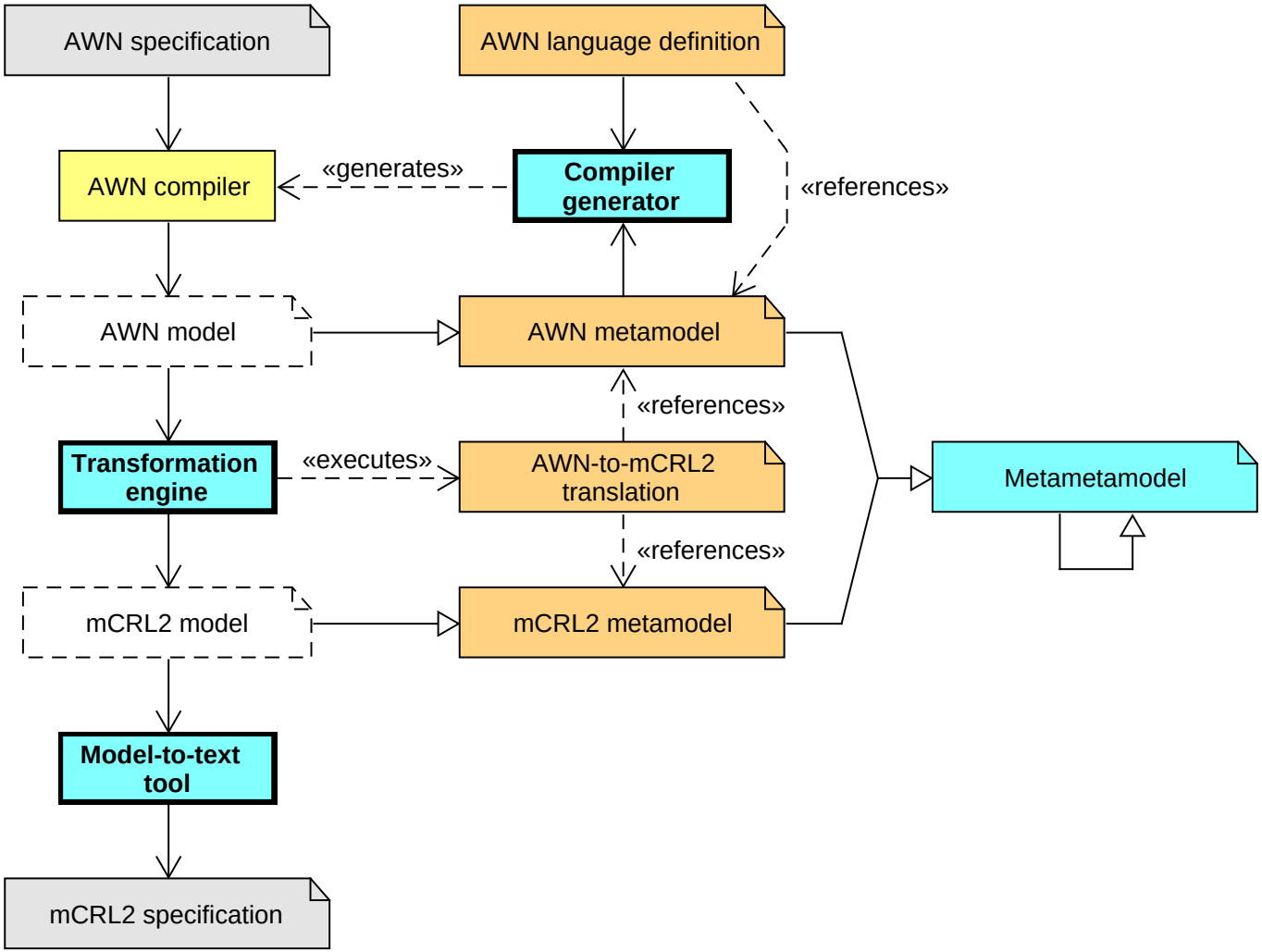
$T(\mathbf{broadcast}(msg).p) = \sum_{R \in \mathcal{P}(IP)} \mathbf{broadcast}(R, R, msg).T(p)$

```
mapping AWN::Broadcast::transform() : MCRL2::SumOperator {
  sumVariable := object MCRL2::Variable {
    name := "R";
    type := "Set(IP)";
  };
  p := object MCRL2::ActionExpr {
    action := object MCRL2::Action {
      name := "broadcast";
      params := Sequence {
        sumVariable.createReference(),
        sumVariable.createReference(),
        self.msg.map transform()
      };
    };
    p := self.p.map transform();
  };
}
```

From AWN to mCRL2 using MDE



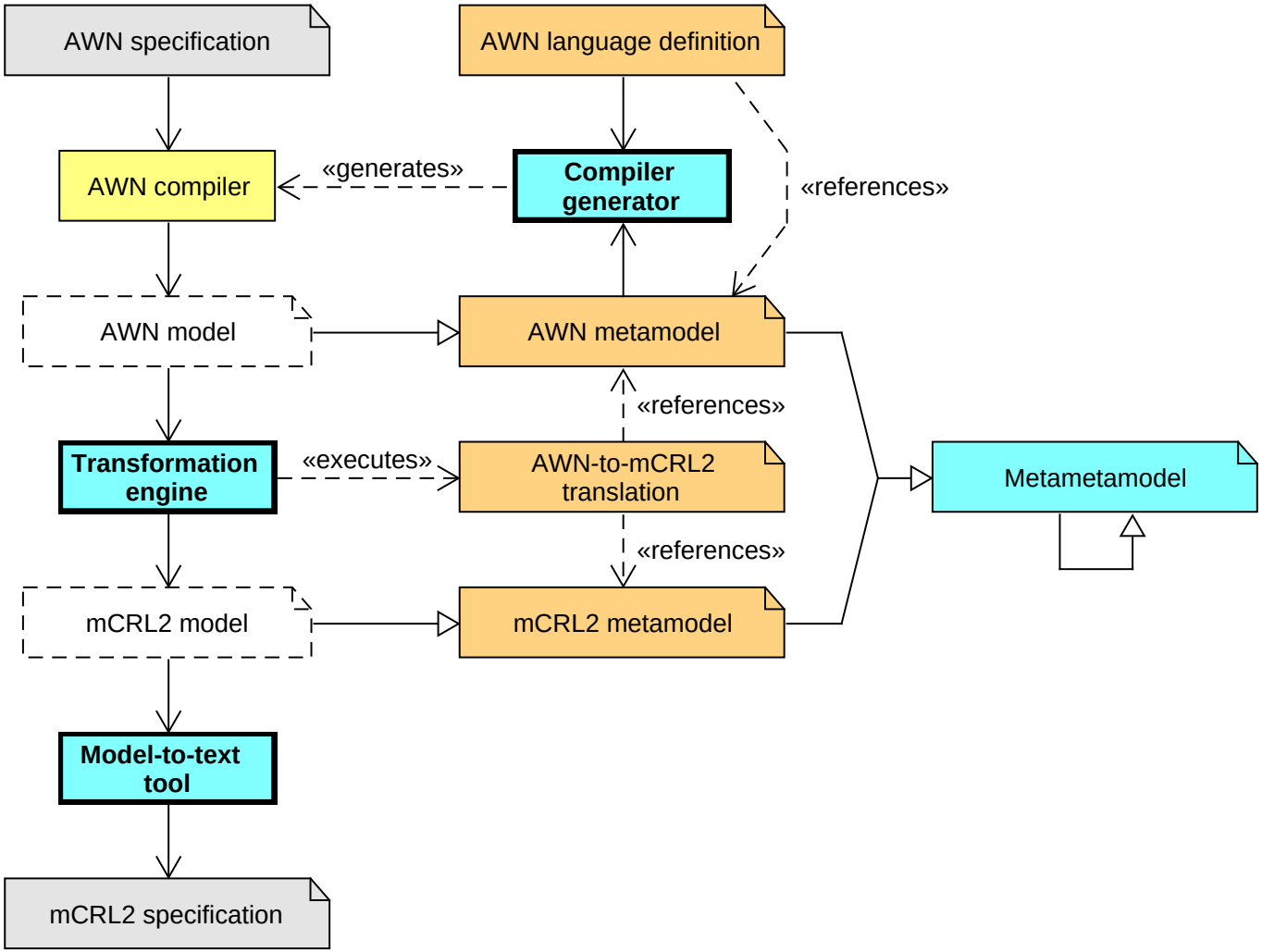
From AWN to mCRL2 using MDE



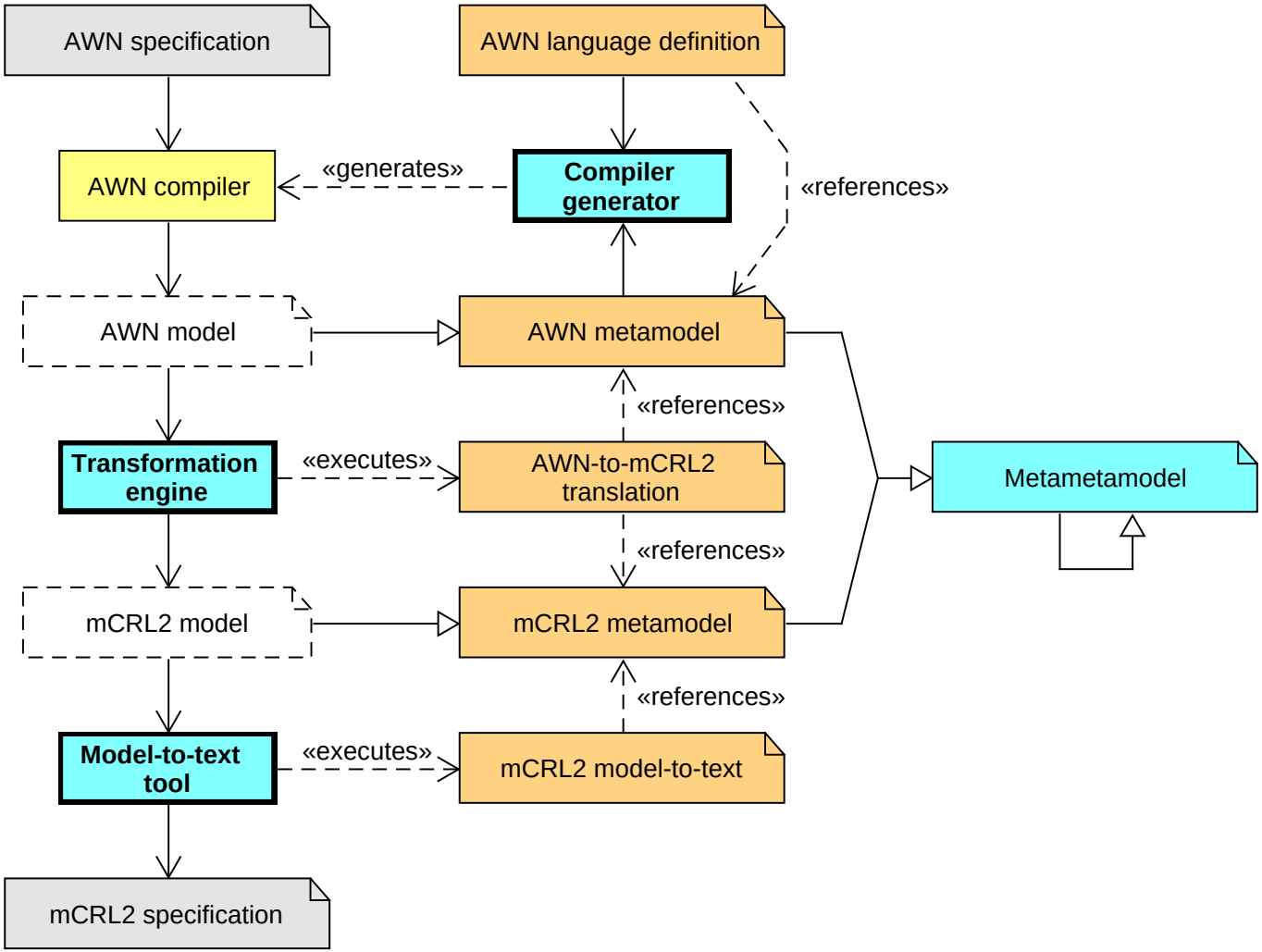
AWN language definition

```
// automatically generated by Xtext
grammar csiro.data61.jawn.Jawn hidden(WS, ML_COMMENT, SL_COMMENT)

import "csiro/data61/jawn/rawASTModelDef"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

InputModelRoot returns Protocol:
    {Protocol}
    (wired?='wired' | 'wireless')? 'protocol' name=ID ';'
    ('message' '(' globalDefs+=MessageVariable (',' globalDefs+=MessageVariable)* ')' ';')?
    (
        globalDefs+=Constant
        | globalDefs+=NamedExprType
        | globalDefs+=Function
        | globalDefs+=Equation
        | globalDefs+=RegularProcess
        | globalDefs+=SimulationProcess
        | globalDefs+=ParallelProcess
        | networks+=Network
        | simulations+=SimulateInit
    )*
;

SimulateInit returns SimulationProcess:
    'simulate' {SimulationProcess} rootAction=Action ';'
;

Constant returns Constant:
    {Constant}
    'const' name=ID ':' type=ExprType ':=' expression=Expression ';'
;

MessageVariable returns MessageVariable:
    {MessageVariable}
    name=ID ':' type=ExprType
;

```

From AWN to mCRL2 using MDE



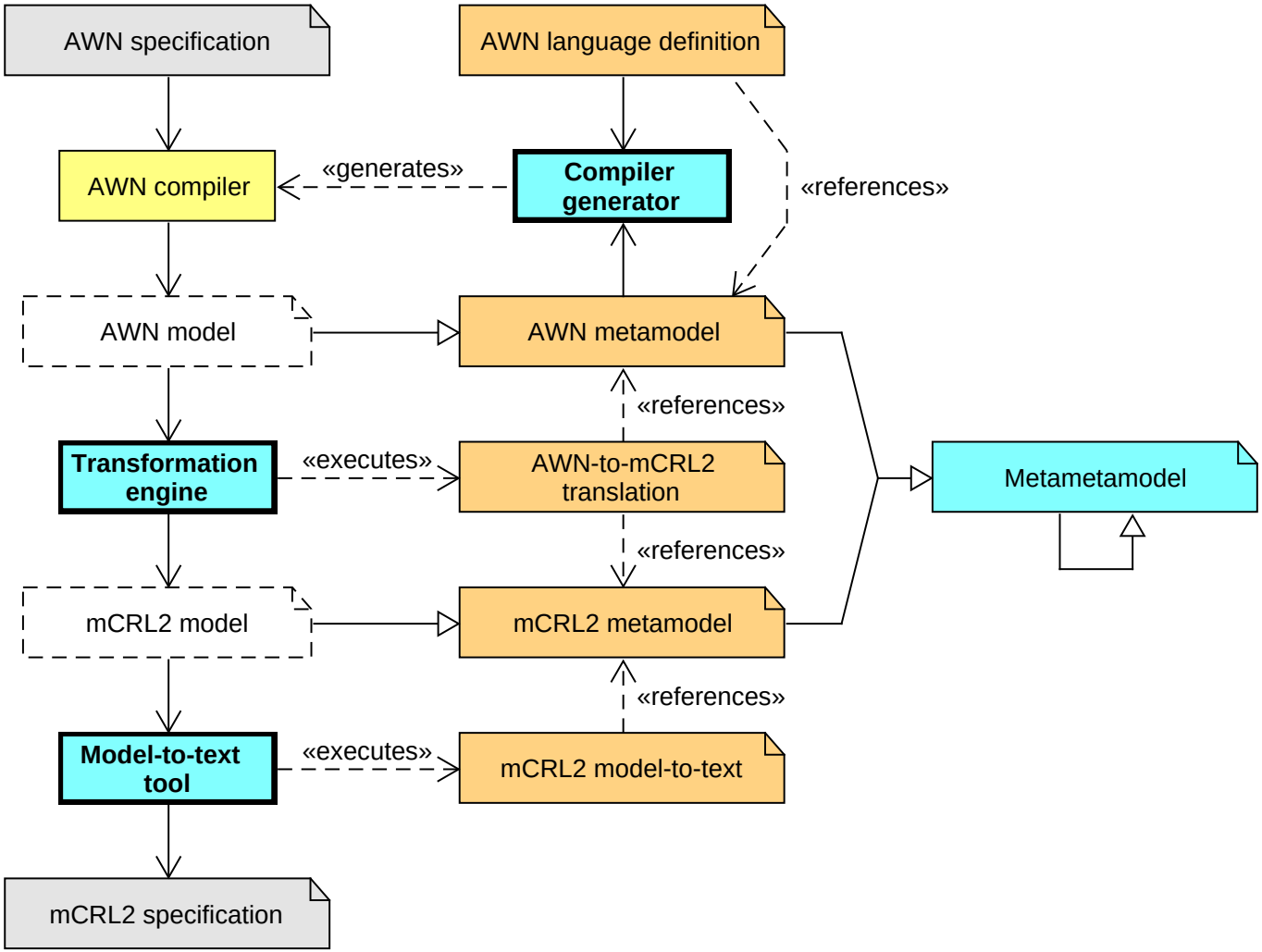
From AWN to mCRL2 using MDE



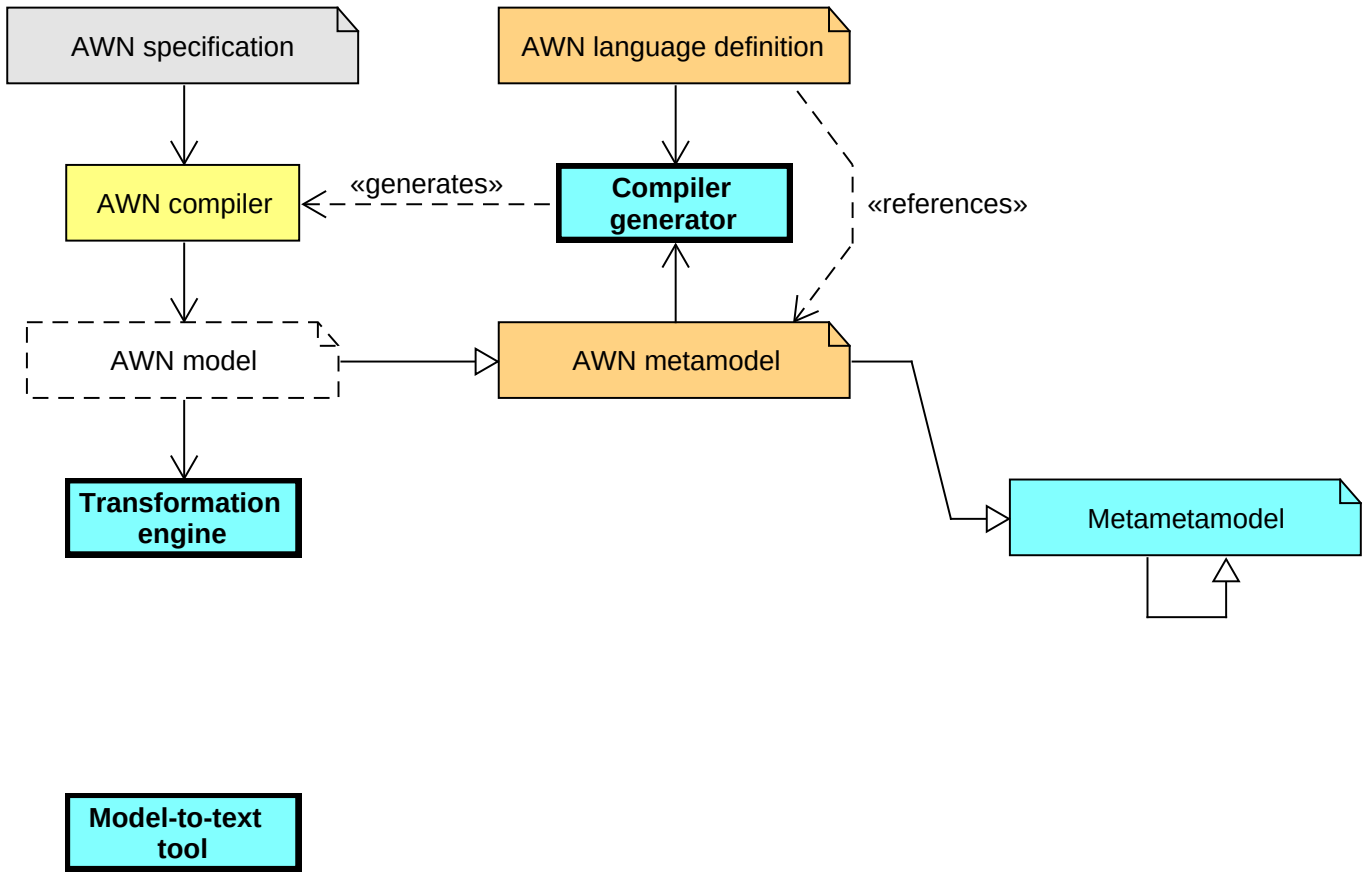
mCRL2 model-to-text

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')/]

[template public generate(aClass : Class)]
[file (aClass.classFileName(), false)]
    package [aClass.containingPackages().name->sep('.')/]

    // [protected ('imports')]
    // [/protected]

    public class [aClass.name.toUpperFirst()/] {
    [for (p: Property | aClass.attribute) separator('\n')]
        private [p.type.name/] [p.name/];
    [/for]

    [for (p: Property | aClass.attribute) separator('\n')]
        public [p.type.name/] get[p.name.toUpperFirst()/]() {
            return this.[p.name/];
        }
    [/for]

    [for (o: Operation | aClass.ownedOperation) separator('\n')]
        public [o.type.name/] [o.name/]() {
            // [protected (o.name)]
            // TODO should be implemented
            // [/protected]
        }
    [/for]
```

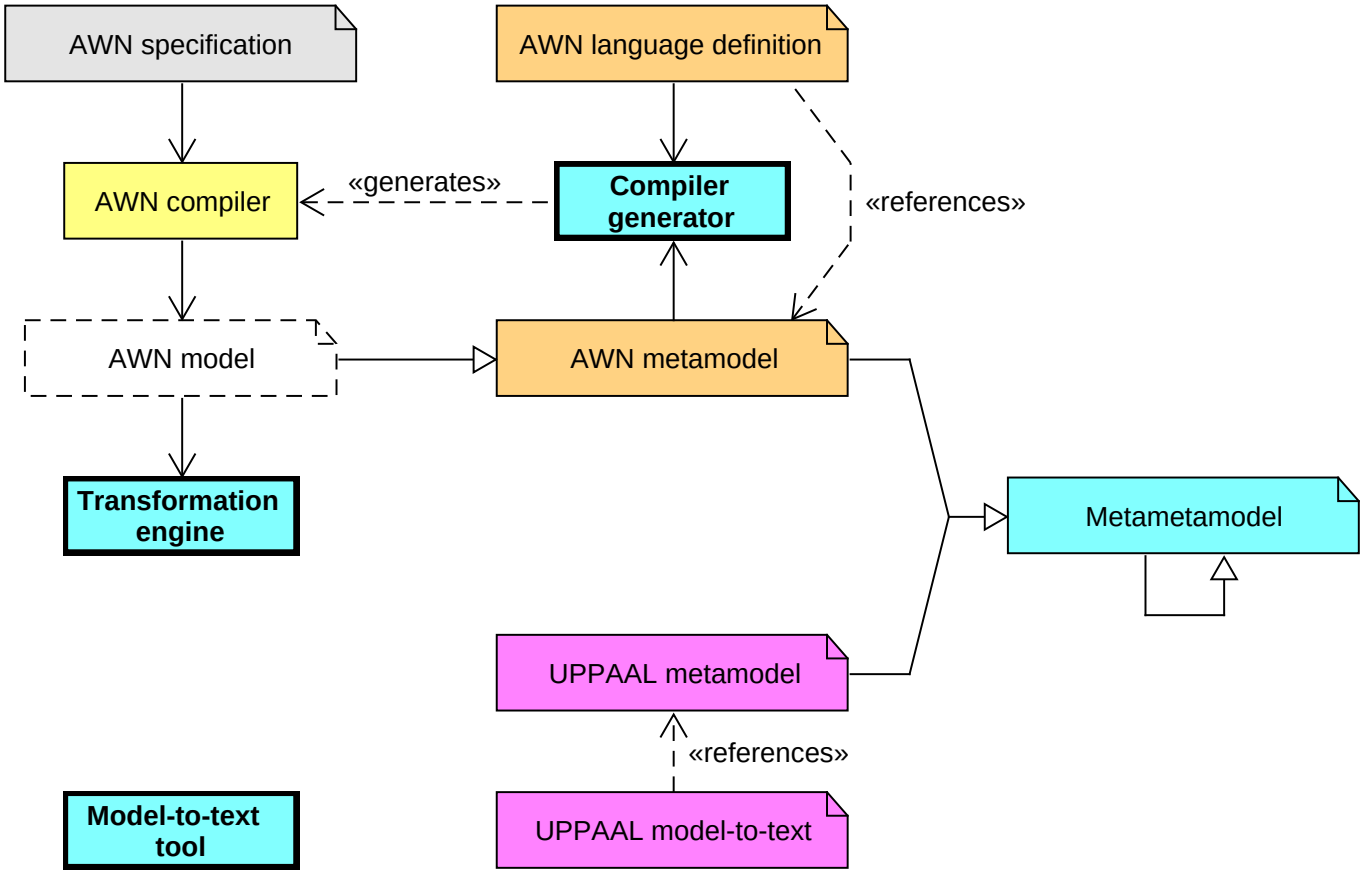
From AWN to mCRL2 using MDE



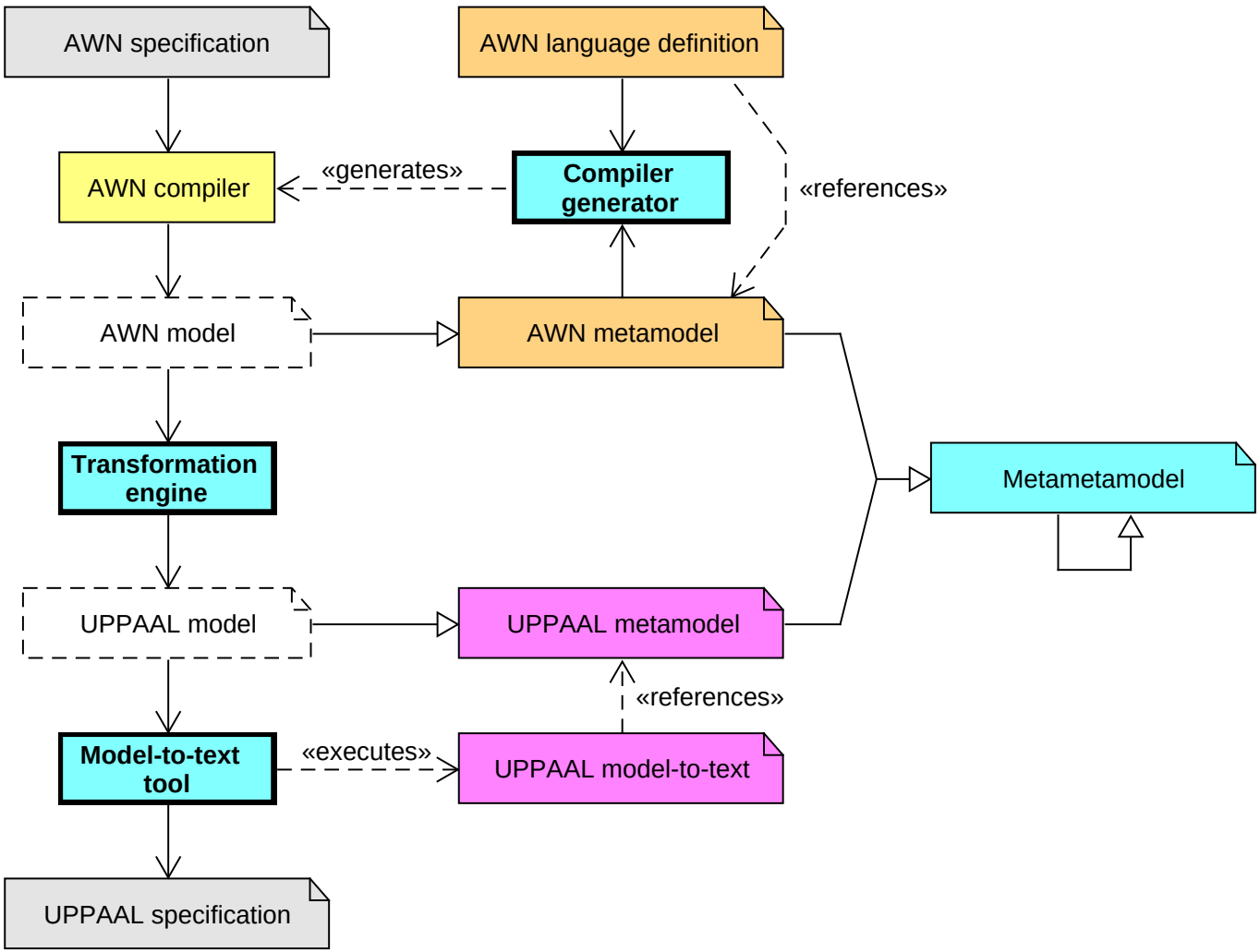
From AWN to mCRL2 using MDE



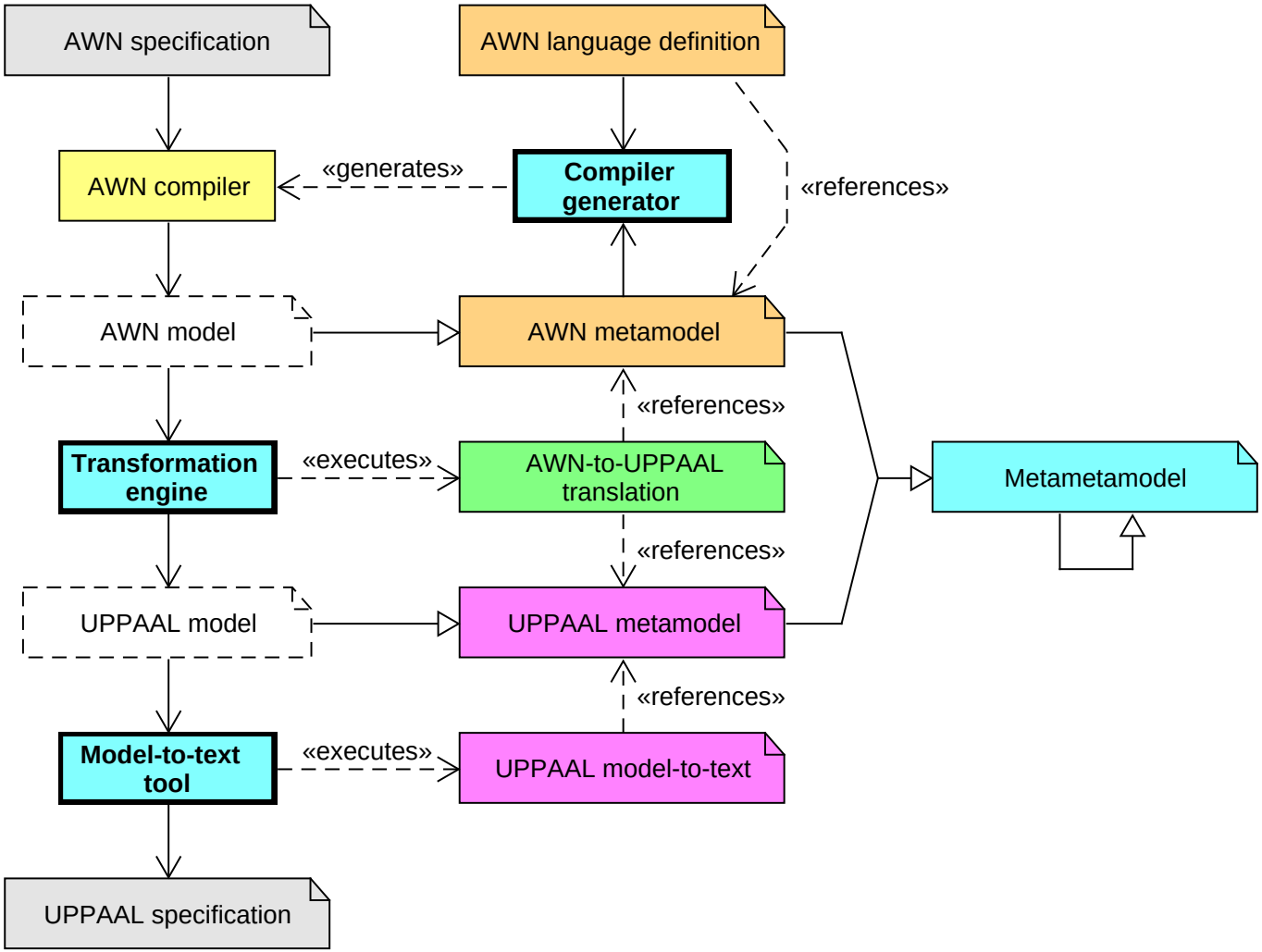
From AWN to mCRL2 using MDE



From AWN to mCRL2 using MDE



From AWN to mCRL2 using MDE



Advantages/Disadvantages

Advantages	**Disadvantages**
Fewer files required	Extra design efforts
Domain rules apply automatically	Knowledge of DSLs required
Peripheral tools available	
Improved interoperability	
Improved reusability	
Improved maintainability	

Future work

Tasks:

- Finish the translation from AWN to mCRL2.
- Prove validity of the translation.
- Automate the translation.
- Determine the suitability of MDE techniques in practice.

The project will take 6 months.