RODIC plenary 2024

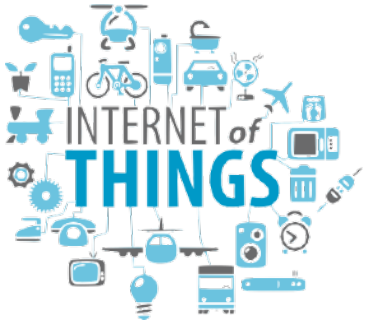
Short introduction to MDE and the Eclipse GEMOC Studio

Erwan Bousse Hiba Ajabri Jean-Marie Mottu

Nantes Université

16/05/2024

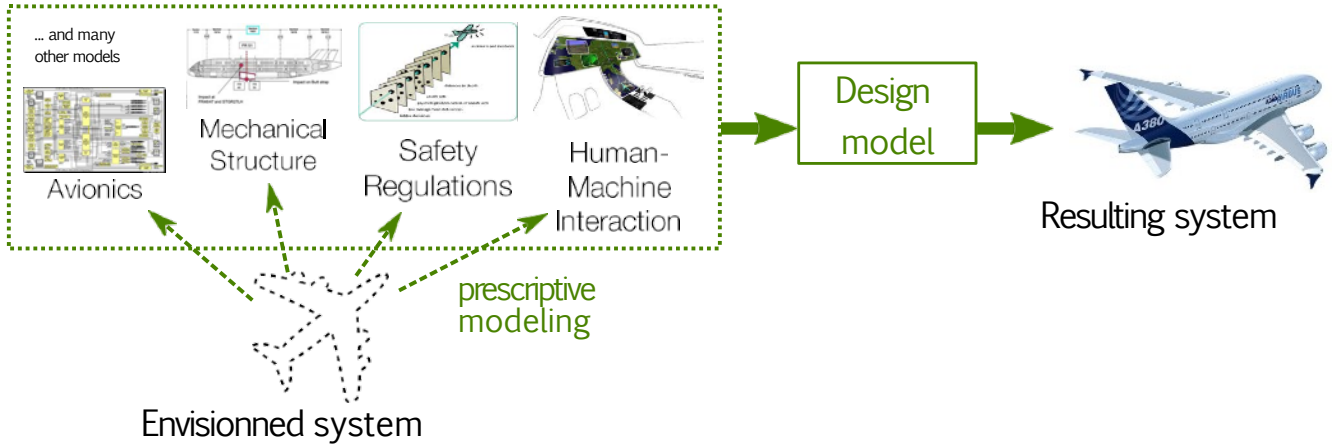
Context: increasing complexity of systems



- Cyber physical systems, internet of things, massively multiplayer online games, artificial intelligence, …
  - complexity everywhere!
  - involving multiple stakeholders and concerns from heterogeneous domains
- Increasing use of software, aka. software–intensive systems

Model-Driven Engineering (MDE)

My view of MDE in a nutshell



1. Separation of concerns through the use of models
   - defined using domain specific languages (DSLs)
   - each representing a particular aspect of a system
2. Composition of all often heterogeneous models
3. Implementation (or generation) of the final resulting system

Domain-Specific Languages (DSLs)

Definition

Well scoped language, often small

- Targets particular tasks in a certain domain
- Relies on dedicated notations (textual or
- graphical)

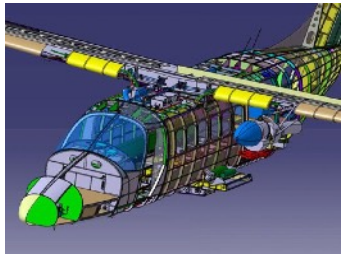
Promises

Less redundancy

- Better separation of concerns
- Accessible for domain experts
-

```
ComputeDt: ∀n∈ℕ,
δt^{n=0}=option_δt_ini; ,
δt^{n+1}=option_δt_cfl*min{j∈cells}(δtj{j});
```

∇Nabla
(Numerical-analysis)



CATIA
(Computed-aided manufacturing)

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

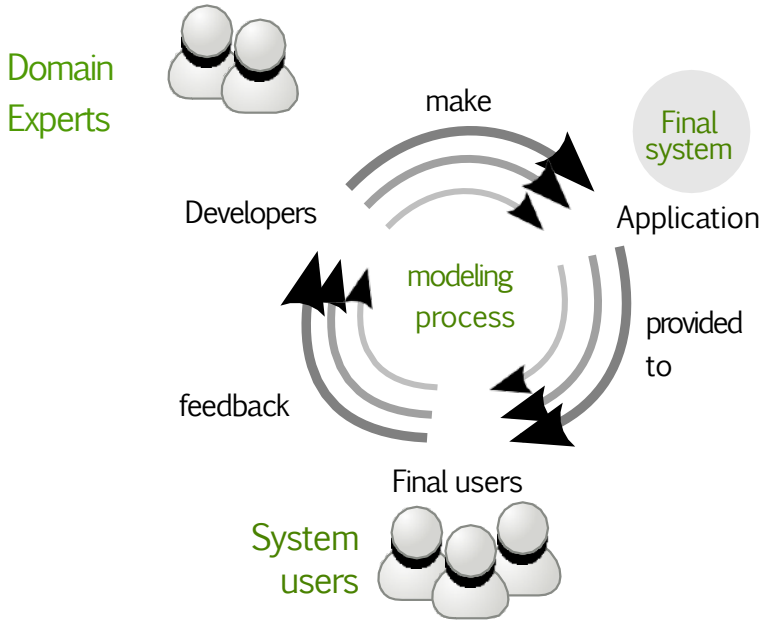
HTML
(Web development)

```
background   { color rgb <0.25, 0.25, 0.25> }
camera       { location  <0.0, 0.5, -4.0>
               direction 1.5*z
               right     x*image_width/image_height
               look_at   <0.0, 0.0, 0.0> }
light_source { <0, 0, 0>
               color rgb <1, 1, 1>
               translate <-5, 5, -5> }
```

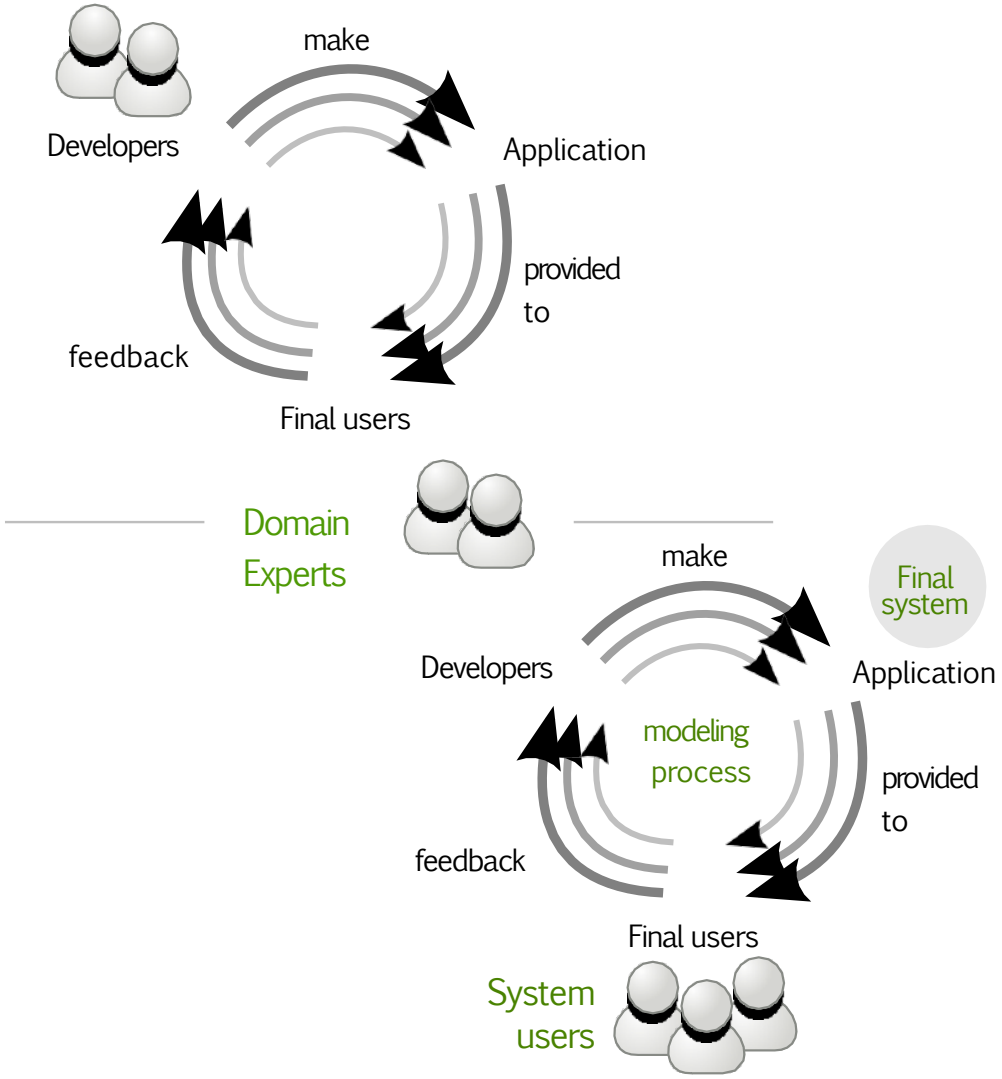
POV-Ray
(Computer graphics)

Engineering Domain Specific Languages (DSLs)

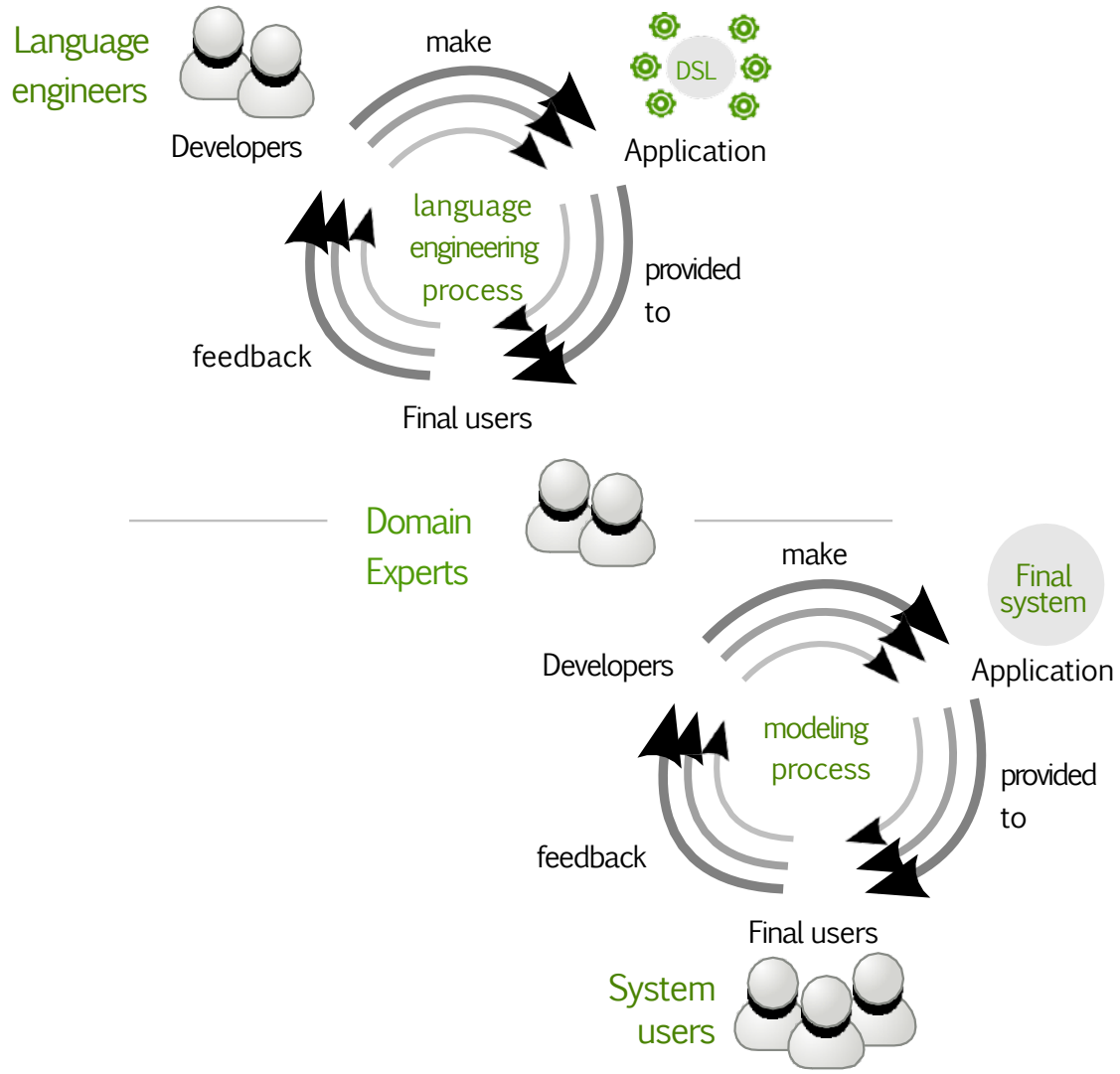
Engineering Domain Specific Languages (DSLs)



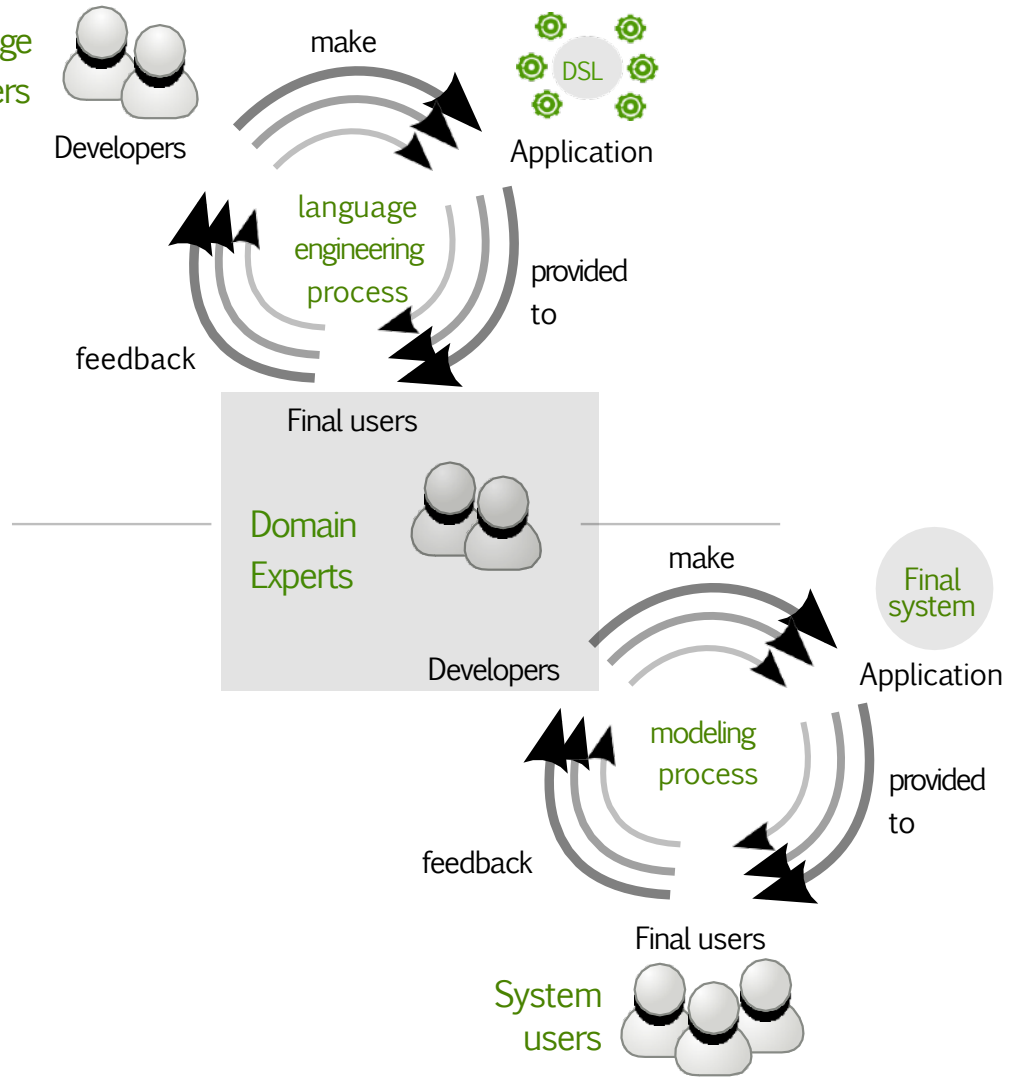
Engineering Domain Specific Languages (DSLs)



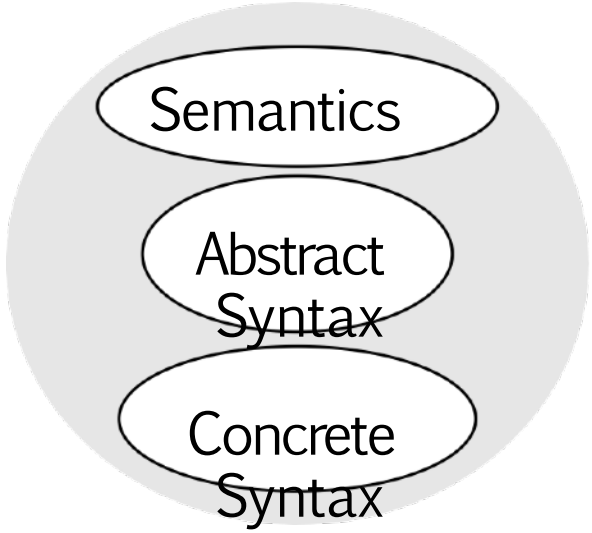
Engineering Domain Specific Languages (DSLs)



Engineering Domain Specific Languages (DSLs)

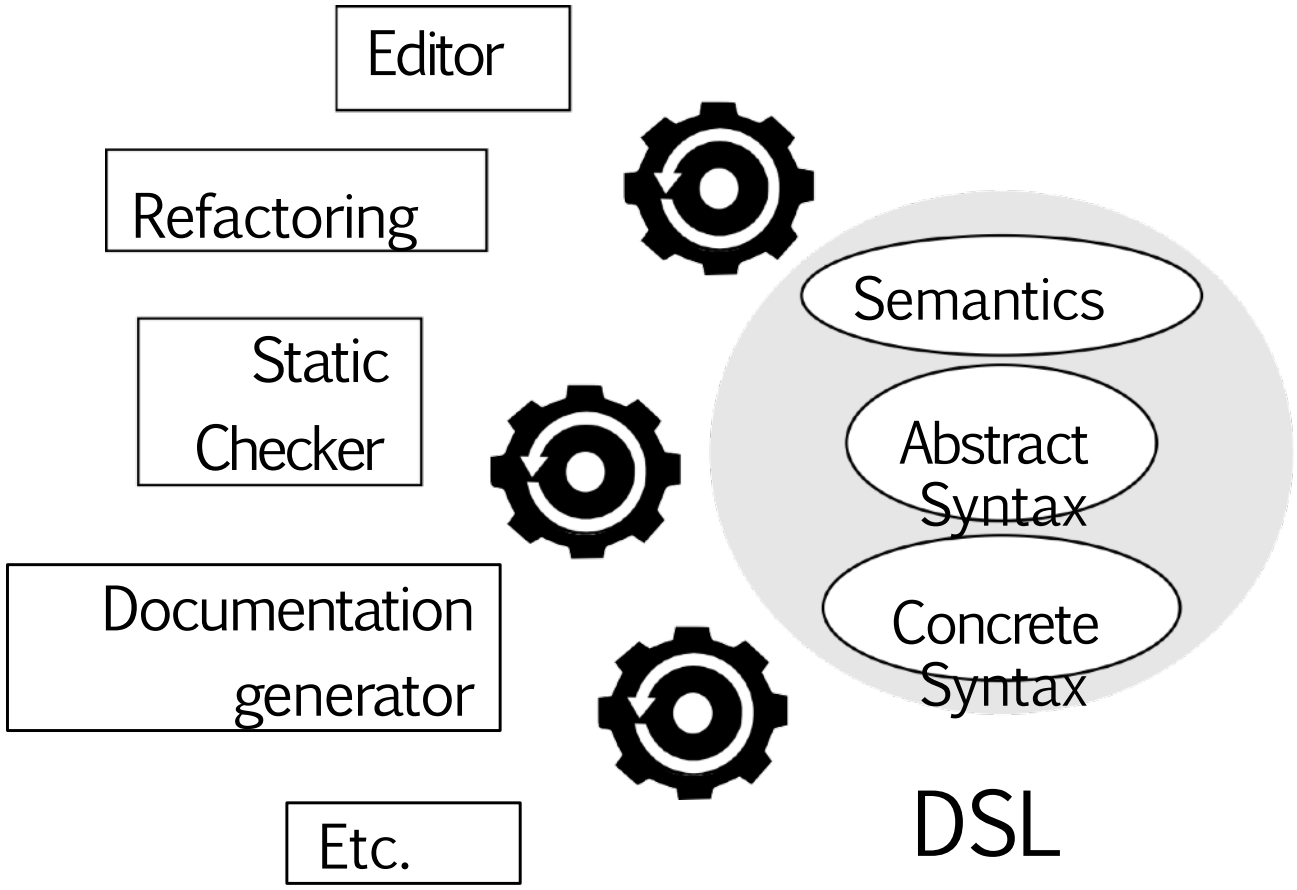


Anatomy and tooling of a DSL

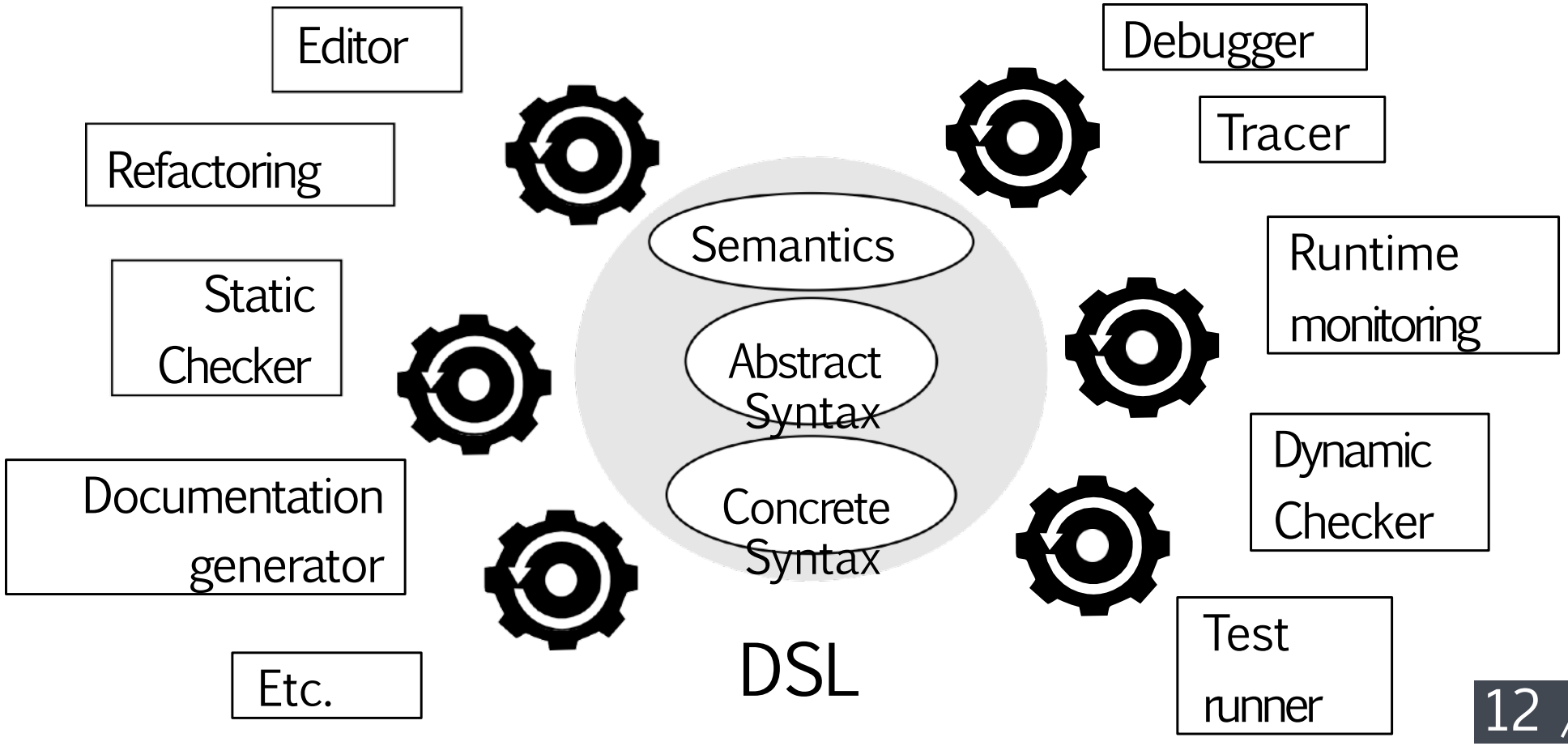


DSL

Anatomy and tooling of a DSL

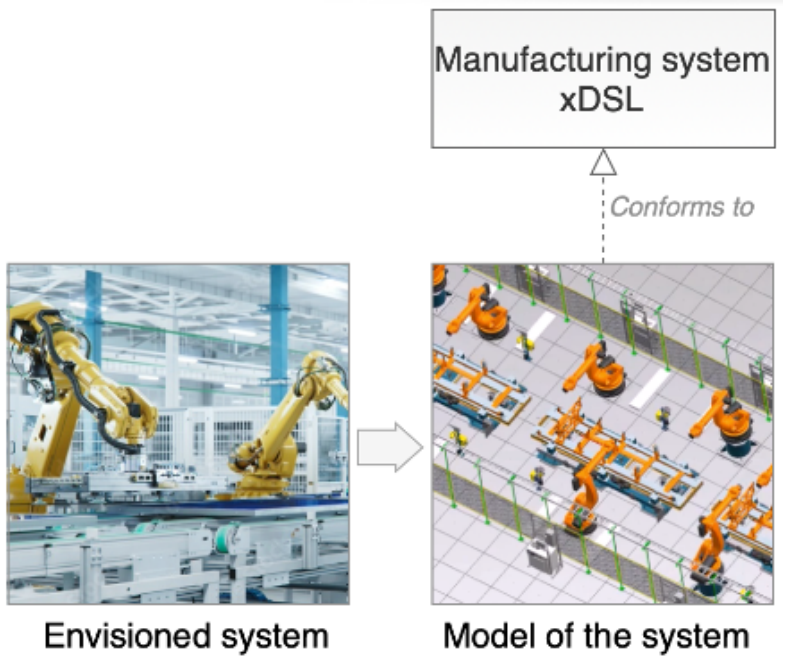


Anatomy and tooling of a DSL



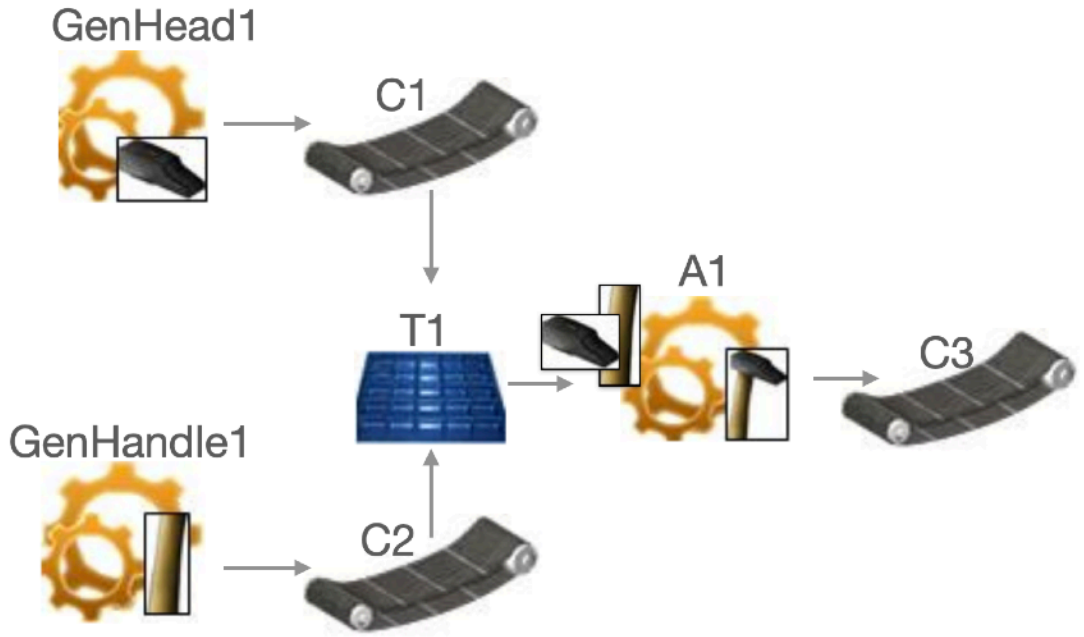
Example of Executable Domain Specific Languages xDSL

- In industry, we tend to reconfigure the production lines according to the situation.
  - Implementing each time a new configuration in a real world to verify its correctness is time-consuming and error-prone.
  - An early solution is to establish configuration models before real implementation.
- Engineering a Domain-Specific Language (DSL) can provide the necessary environment to model the configurations.
  - A DSL can be executable (xDSL) to simulate its behavior

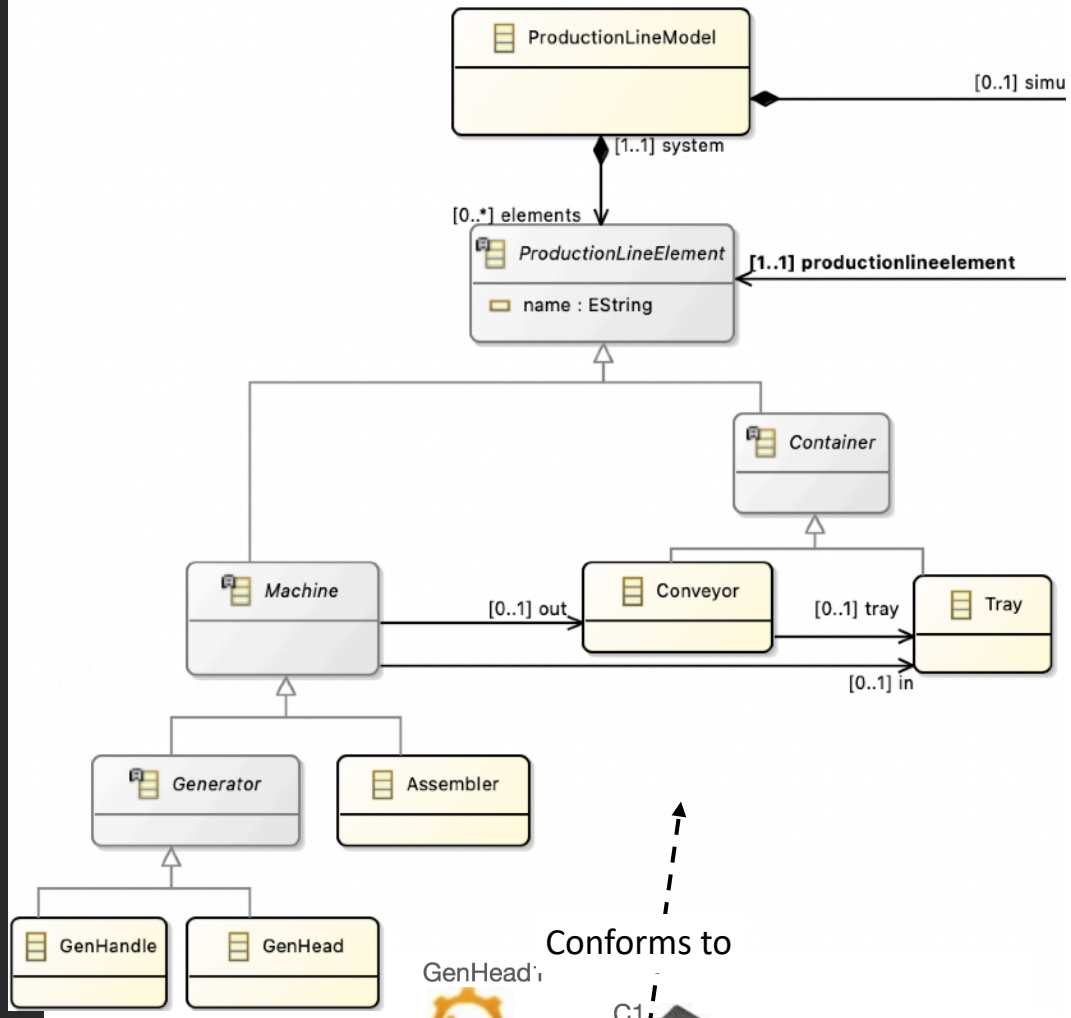


Example of Executable Domain Specific Languages xDSL

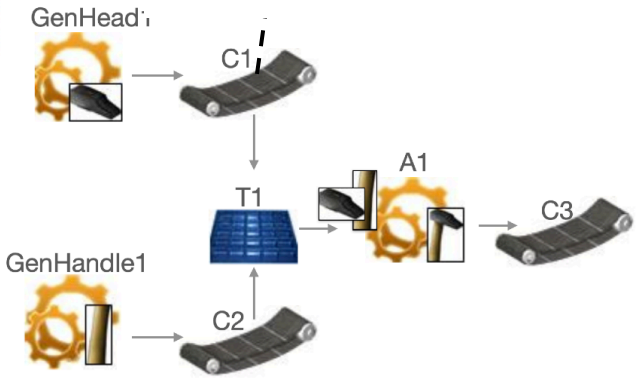
In this workshop, we want to model a simple production line Hammer Factory.

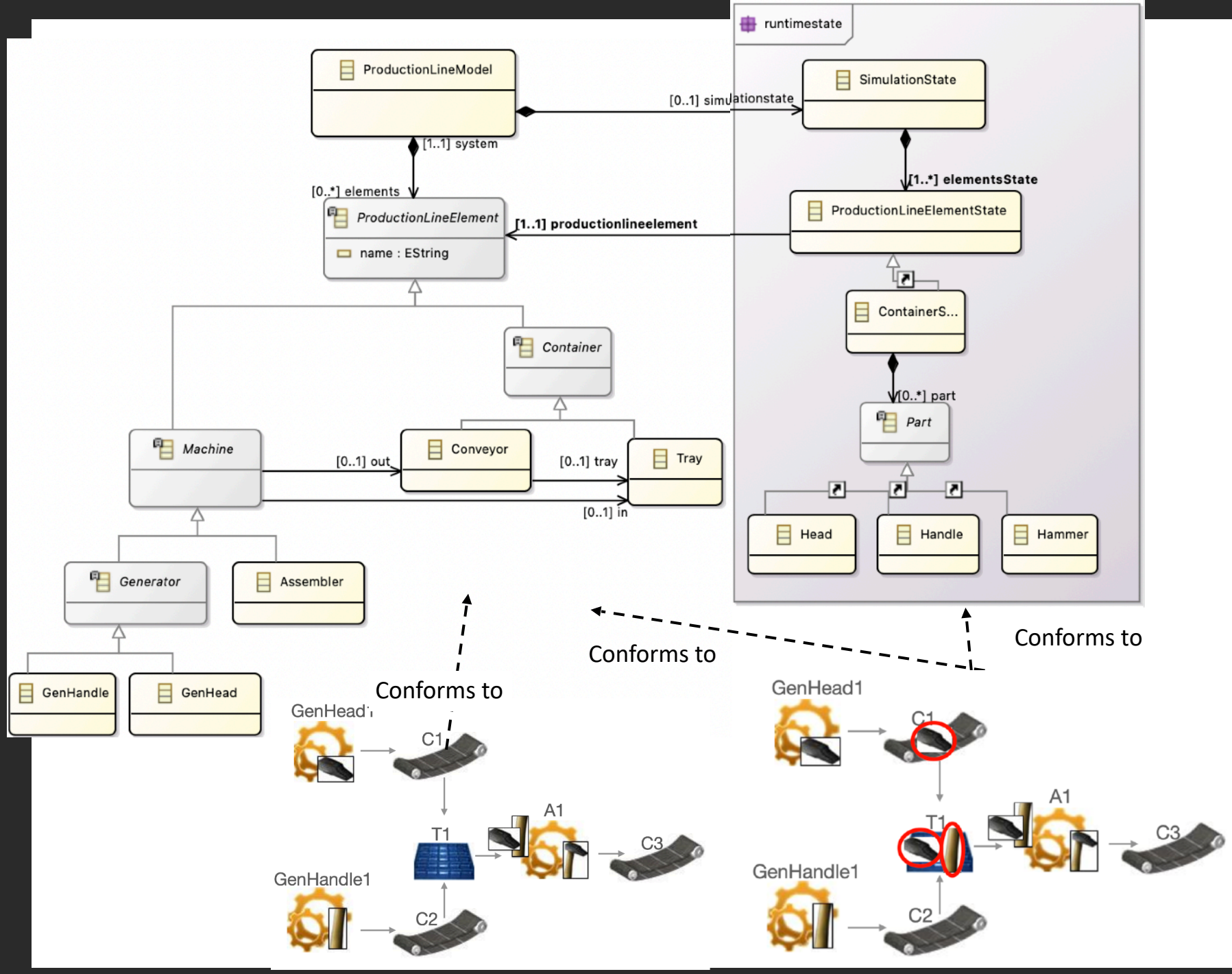


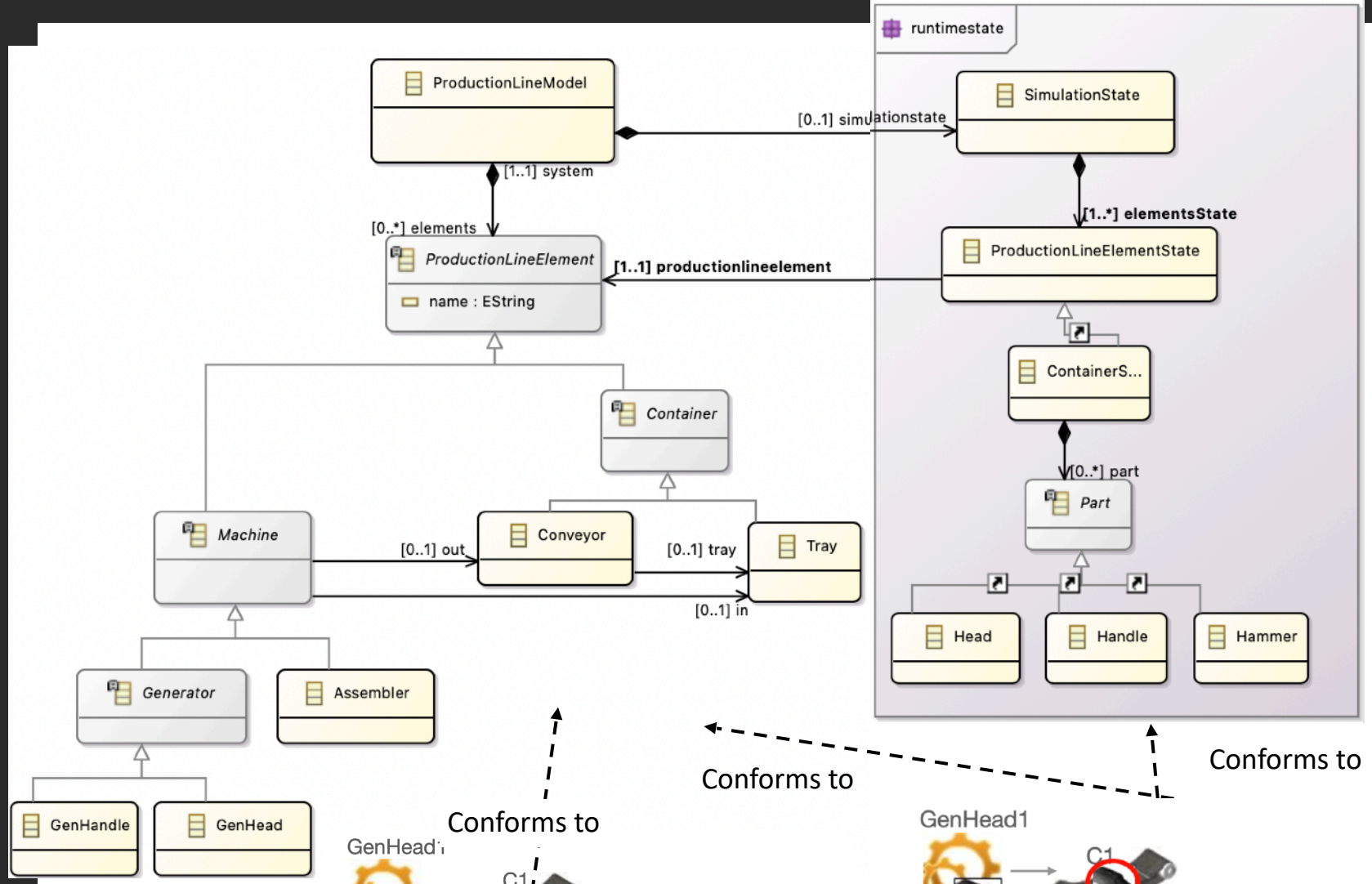
To do that, we create a Mini Simple Manufacturing Systems xDSL (minisms) enables modeling production line systems.

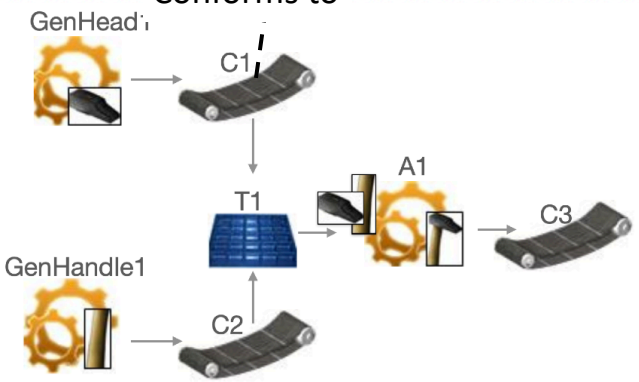

ProductionLineModel
[0..1] simu
[1..1] system
[0..*] elements
ProductionLineElement
name : EString
[1..1] productionlineelement
Container
Machine
[0..1] out
Conveyor
[0..1] tray
Tray
[0..1] in
Generator
Assembler
GenHandle
GenHead

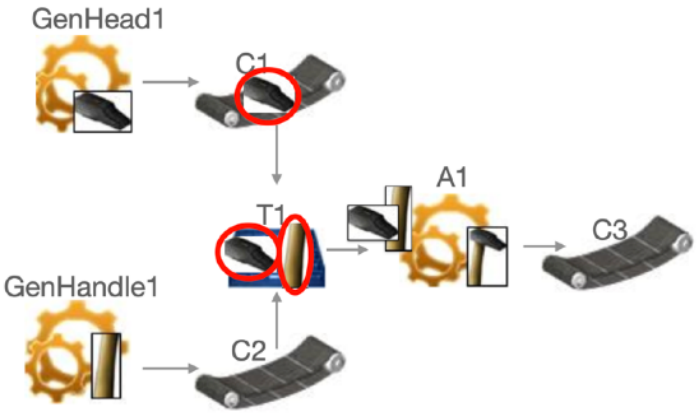
Conforms to


GenHead1
C1
A1
T1
C3
GenHandle1
C2


runtimestate
ProductionLineModel
[0..1] simulationstate
SimulationState
[1..1] system
[0..*] elements
[1..*] elementsState
ProductionLineElement
name : EString
[1..1] productionlineelement
ProductionLineElementState
ContainerS...
Container
[0..*] part
Part
Machine
[0..1] out
Conveyor
[0..1] tray
Tray
[0..1] in
Head
Handle
Hammer
Generator
Assembler
GenHandle
GenHead
Conforms to
Conforms to
Conforms to
GenHead1
C1
A1
T1
C3
GenHandle1
C2

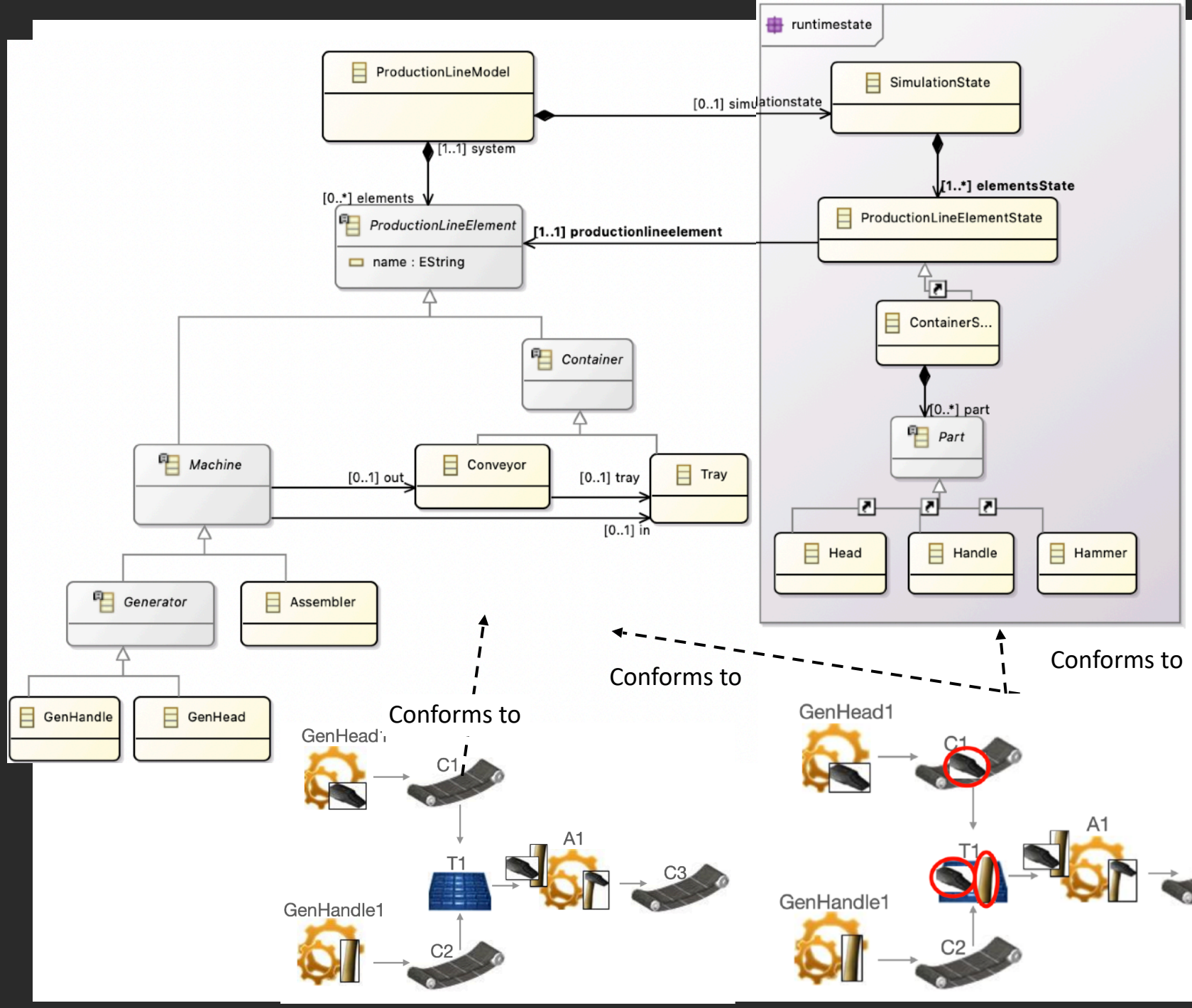






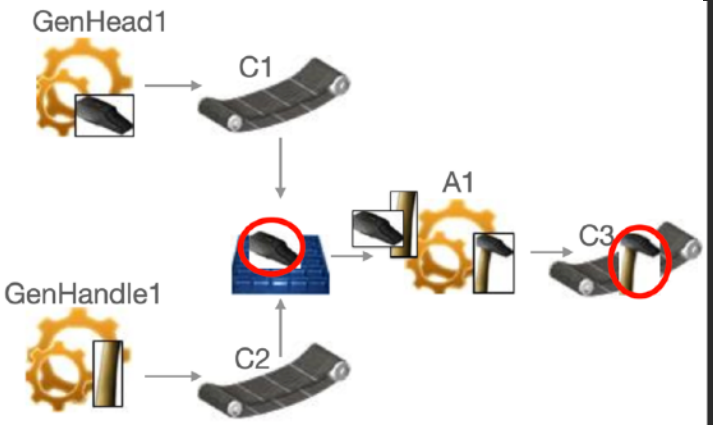
Execution rules: Assembler

```
@Aspect(className=Assembler)
class AssemblerAspect extends MachineAspect {
    @Step
    @OverrideAspectMethod
    def void start(){
        /**
         * Look if there are products in the tray
         */
        if(_self.in.currentParts.exists[p|p instanceof Head ] &&
            _self.in.currentParts.exists[p|p instanceof Handle ]
        ){
            println("I am in the Assembler machine name '"+_self.name+"'")
            /** Retrieve a Handle product */
            var anHandle=_self.in.currentParts.filter[e|e instanceof Handle].get(0)
            /** Retrieve a Head product */
            var aHead=_self.in.currentParts.filter[e|e instanceof Head].get(0)

            /** Take products from the tray */
            _self.in.currentParts.remove(anHandle)
            _self.in.currentParts.remove(aHead)

            /** Assemble them and generate a Hammer and put it on the conveyer in output */
            var aHammer = runtimestateFactory.eINSTANCE.createHammer
            _self.out.currentParts.add(aHammer)
            return
        }
    }
}
```



Execution rules: Assembler

```
@Aspect(className=Assembler)
class AssemblerAspect extends MachineAspect {
    @Step
    @OverrideAspectMethod
    def void start(){
        /**
         * Look if there are products in the tray
         */
        if(_self.in.currentParts.exists[p|p instanceof Head ] &&
            _self.in.currentParts.exists[p|p instanceof Handle ]
        ){
            println("I am in the Assembler machine name '"+_self.name+"'")
            /** Retrieve a Handle product */
            var anHandle=_self.in.currentParts.filter[e|e instanceof Handle].get(0)
            /** Retrieve a Head product */
            var aHead=_self.in.currentParts.filter[e|e instanceof Head].get(0)

            /** Take products from the tray */
            _self.in.currentParts.remove(anHandle)
            _self.in.currentParts.remove(aHead)

            /** Assemble them and generate a Hammer and put it on the conveyer in output */
            var aHammer = runtimestateFactory.eINSTANCE.createHammer
            _self.out.currentParts.add(aHammer)
            return
        }
    }
}
```

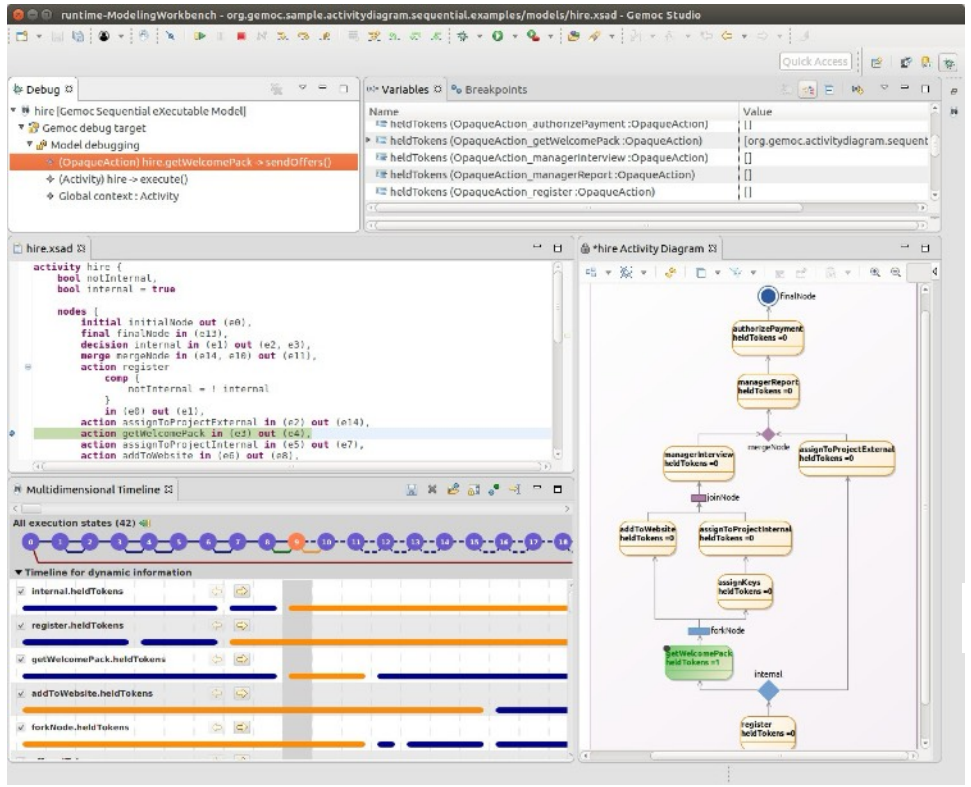
```
@Aspect(className=Assembler)
class AssemblerAspect extends MachineAspect {
    @Step
    @OverrideAspectMethod
    def void start(){
        /**
         * Look if there are products in the tray
         */
        if(_self.in.currentParts.exists[p|p instanceof Head ] &&
            _self.in.currentParts.exists[p|p instanceof Handle ]
        ){
            println("I am in the Assembler machine name '"+_self.name+"'")
            /** Retrieve a Handle product */
            var anHandle=_self.in.currentParts.filter[e|e instanceof Handle].get(0)
            /** Retrieve a Head product */
            var aHead=_self.in.currentParts.filter[e|e instanceof Head].get(0)

            /** Take products from the tray */
            _self.in.currentParts.remove(anHandle)
            _self.in.currentParts.remove(aHead)

            /** Assemble them and generate a Hammer and put it on the conveyer in output */
            var aHammer = runtimestateFactory.eINSTANCE.createHammer
            _self.out.currentParts.add(aHammer)
            return
        }
    }
}
```

The Eclipse GEMOC Studio

An SLE research playground

Description

- Open-source Eclipse-based workbench atop the Eclipse Modeling Framework (EMF), in two parts:
  - language workbench: used by language designers to *build and compose new executable DSLs*,
  - modeling workbench: used by domain designers to create, execute and coordinate models conforming to executable DSLs.



Status and purpose

Status of the Eclipse GEMOC Studio

- Handled by the GEMOC initiative, an informal group with partners from both the academia and the industry
- Now also an official research consortium of the Eclipse foundation

Purpose of the Eclipse GEMOC Studio

- Research platform to experiment with SLE research proof-of-concepts / prototypes, and to integrate them together
- The Eclipse GEMOC Studio is not an industry-ready tool, and will never be one!

Installation

❖17 JDK Installation

- Link : https://www.oracle.com/java/technologies/downloads/#jdk17
- You may verify the java version using :

```
> java -version
```

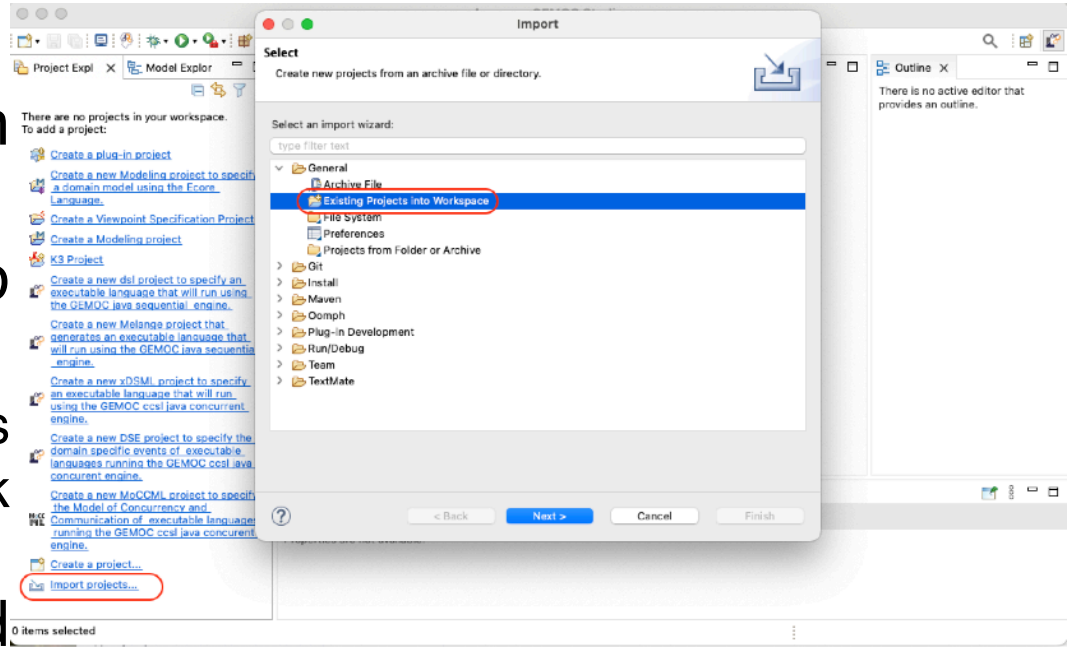
❖Eclipse GEMOC Studio

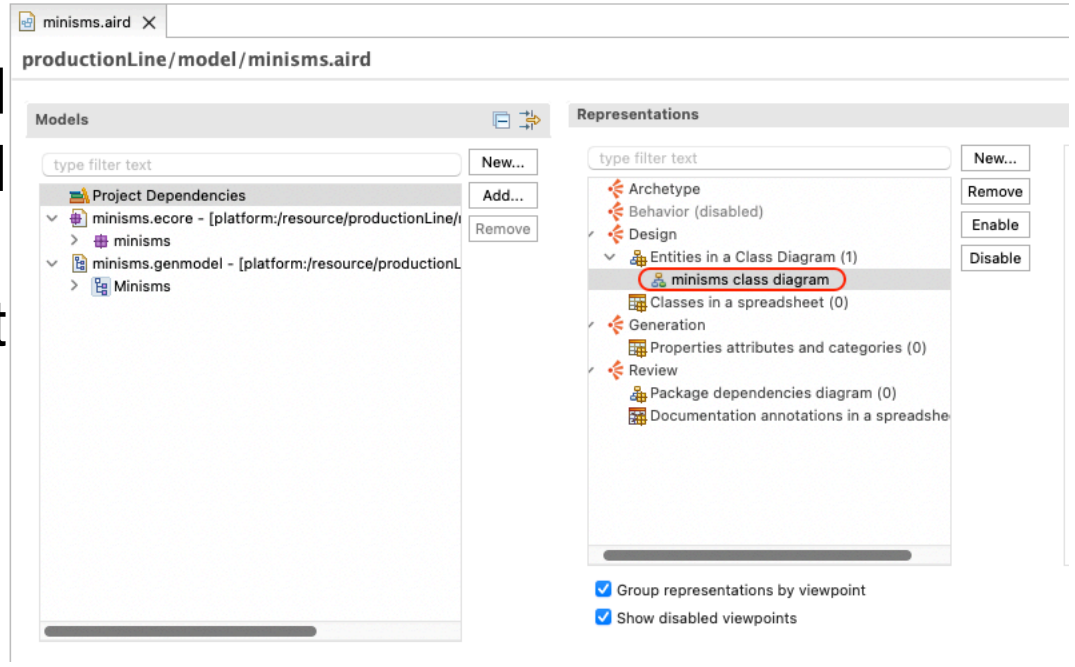
- Link : https://download.eclipse.org/gemoc/packages/nightly/
- On mac, execute the following command in your terminal :

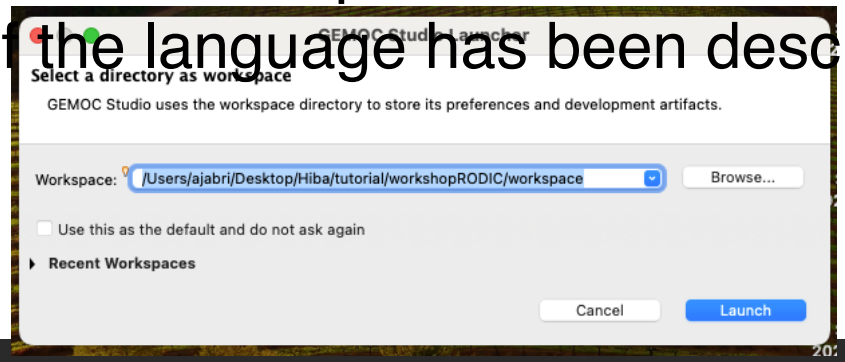
```
xattr -d com.apple.quarantine /path/to/Eclipse.app
```

Abstract Syntax

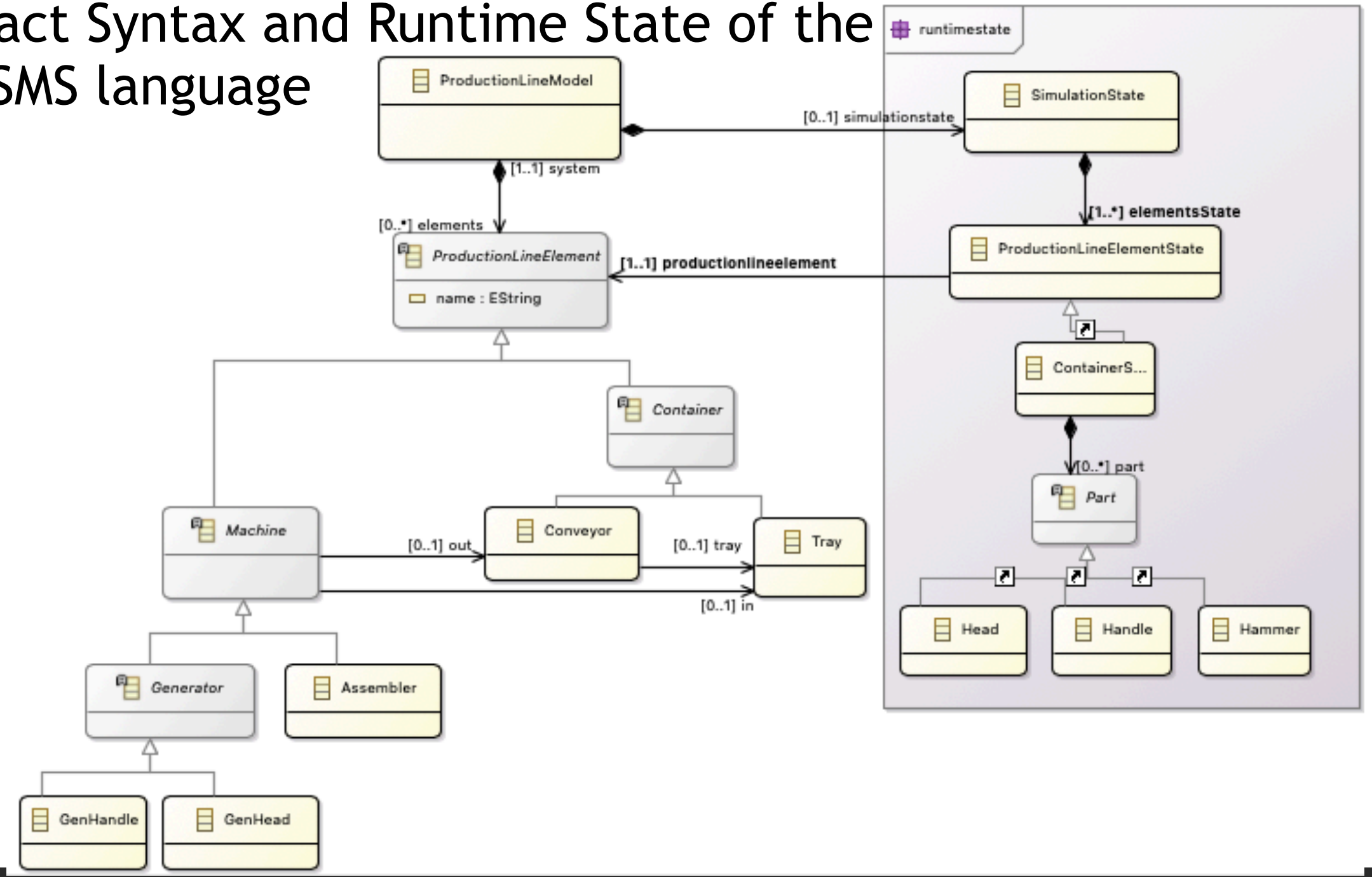
- Open the Eclipse GEMOC Studio in an empty workspace.
- Import…> General > Existing Projects into Workspace:
  - To import the project from the provided files (Files/EMF projects/EMF project 1), and click Finish.
- Extend the ***productionLine*** project and double-click on ***minisms.aird***.
- In the ***minisms.aird*** tab, you may extend the ***Entities in a Class Diagram*** and double-click on ***minisms class diagram***.
- A new tab will be open where the abstract syntax of the language has been described.







Abstract Syntax and Runtime State of the mini SMS language

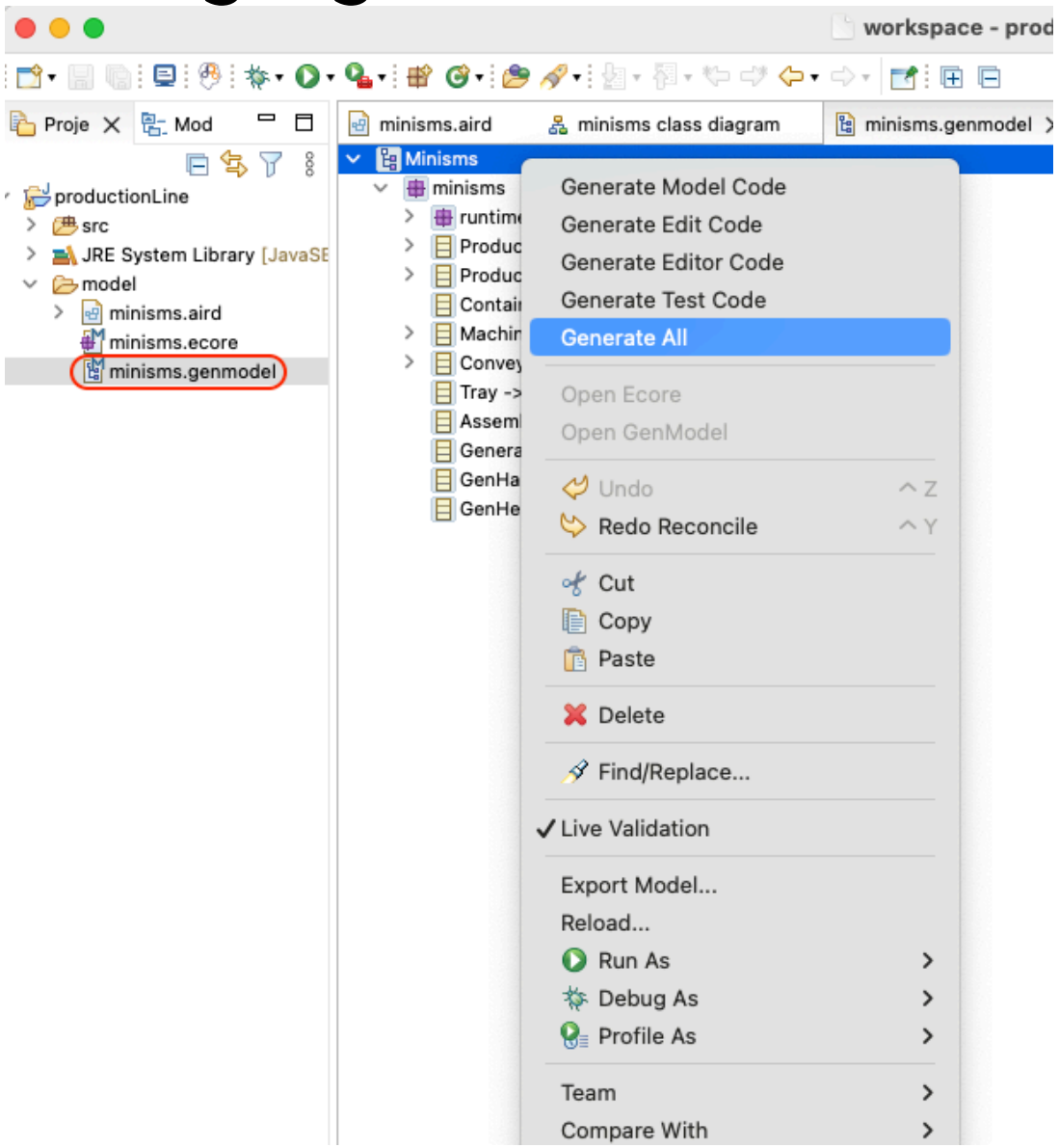


Abstract Syntax of the mini SMS language

To generate the code:

- Double-click on the
  minisms.genmodel file.
- Extend the tree in the opened tab and rigth-click, then select ***Generate All***.

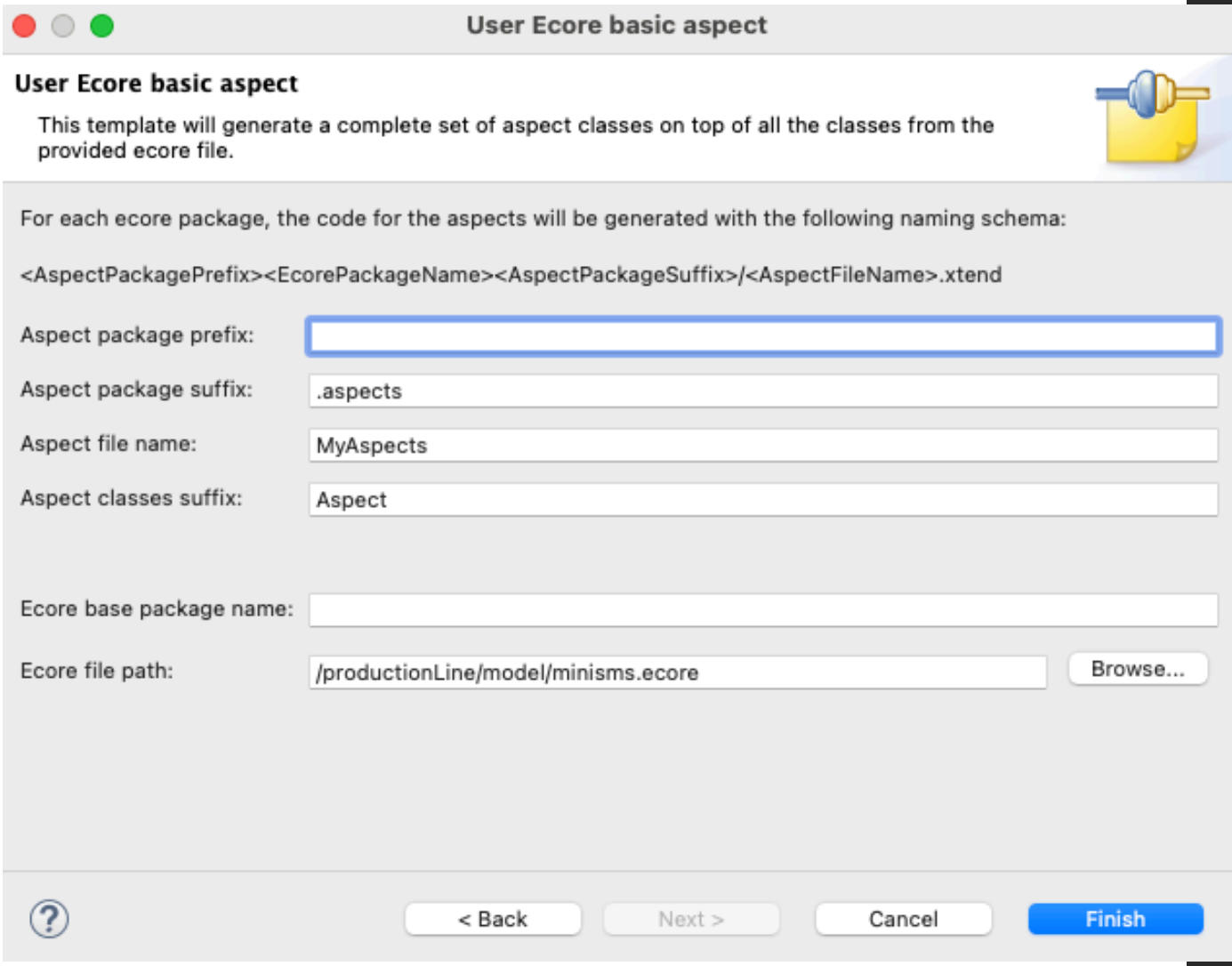
This will generate the code in the src folder and other project that assist in using the language.



Operational Semantics : Simulator

To describe the semantics of the concepts of the language,

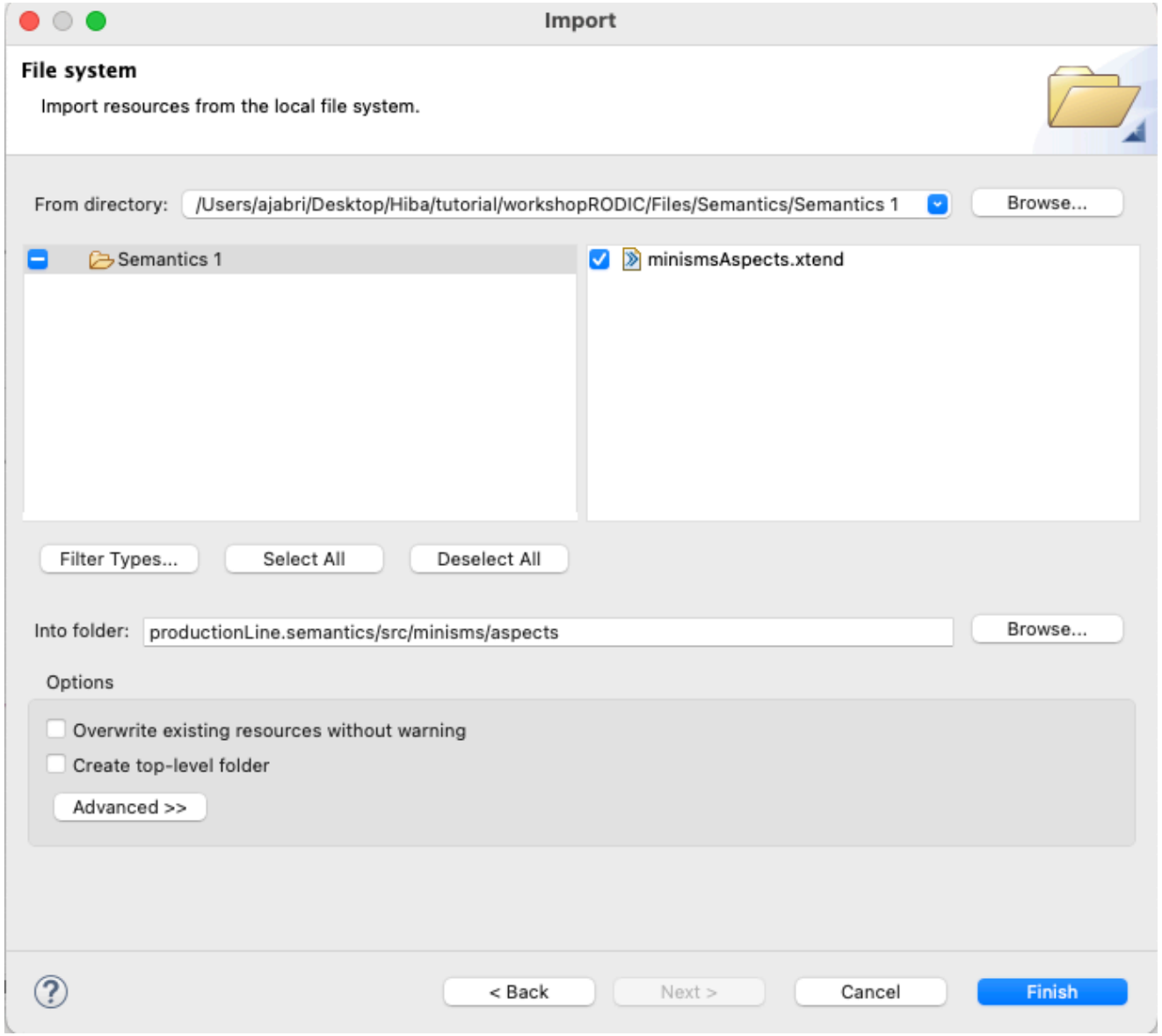
- Create a Kermeta 3 project :
- File > New > Other … > Kermeta 3 > K3 Project > Next **(and not *Finish*).**
- You may name the project ***productionLine.semantics***, and press Next **(and not *Finish*).**
- Tick the box ***Create a plug-in using a template*** and select ***User Ecore Basic Aspects***, and **press *next* (and not *Finish*).**
- Use only the ***Browse*** button to select the ***Ecore file path***, and choose your abstract syntax Ecore metamodel, and press ***Finish***.

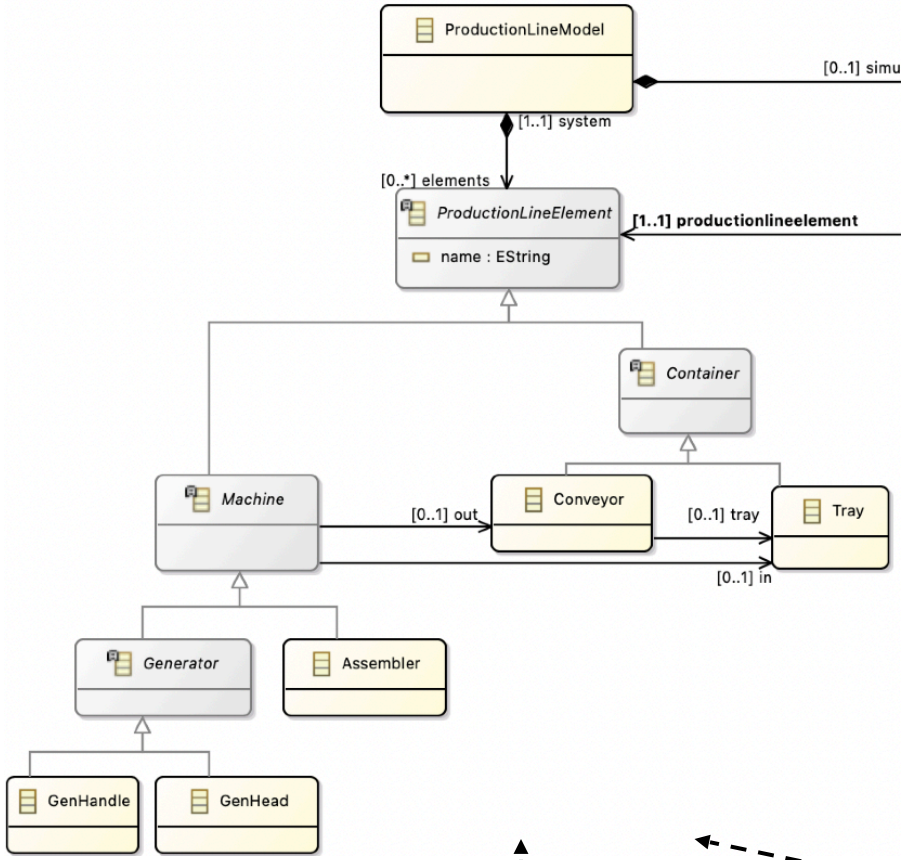


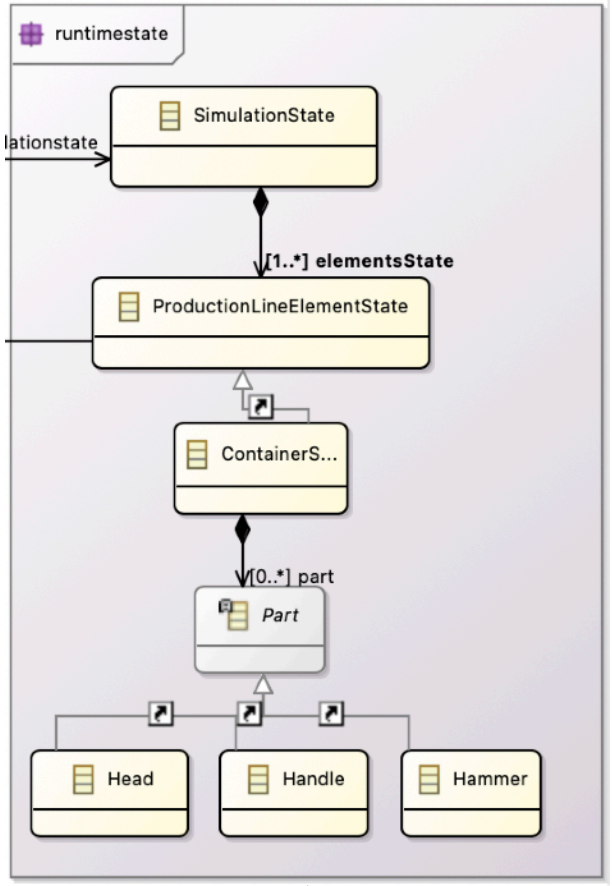
Operational Semantics : Simulator

You have create a kermeta 3 project.

- Extend the project, then open the ***src*** folder then the ***minisms.aspects*** packge.
  We will replace the ***minismsAspects.xtend*** file, to do that:
- Click-right on the ***minisms.aspects*** packge> Import… > File System > Next.
- Click on the ***Browse*** button and select the folder ***Files/Semantics/Semantics 1*** from provided files, and click ***Open***.
- Tick the box ***minismsAspects.xtend***, and press ***Finish.***







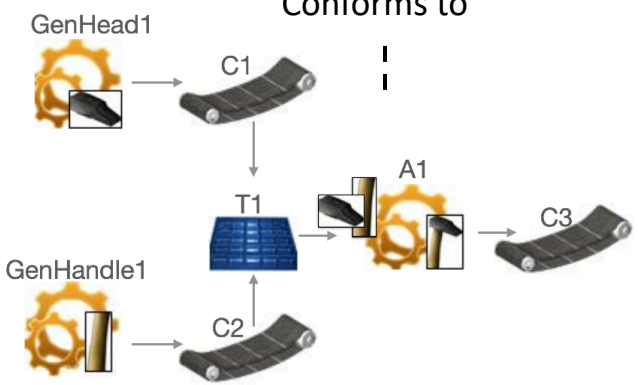
Execution rules: Assembler

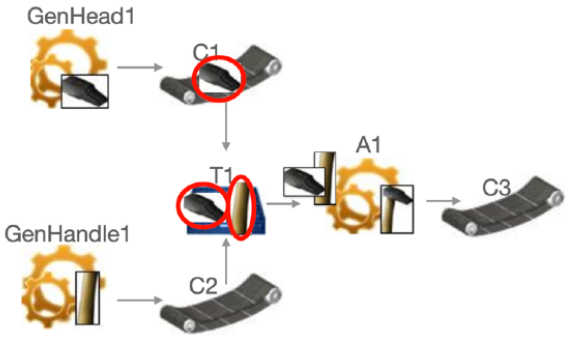
```
@Aspect(className=Assembler)
class AssemblerAspect extends MachineAspect {
    @Step
    @OverrideAspectMethod
    def void start(){
        /**
         * Look if there are products in the tray
         */
        if(_self.in.currentParts.exists[p|p instanceof Head ] &&
            _self.in.currentParts.exists[p|p instanceof Handle ]
        ){
            println("I am in the Assembler machine name '"+_self.name+"'")
            /** Retrieve a Handle product */
            var anHandle=_self.in.currentParts.filter[e|e instanceof Handle].get(0)
            /** Retrieve a Head product */
            var aHead=_self.in.currentParts.filter[e|e instanceof Head].get(0)

            /** Take products from the tray */
            _self.in.currentParts.remove(anHandle)
            _self.in.currentParts.remove(aHead)

            /** Assemble them and generate a Hammer and put it on the conveyer in output */
            var aHammer = runtimestateFactory.eINSTANCE.createHammer
            _self.out.currentParts.add(aHammer)
            return
        }
    }
}
```

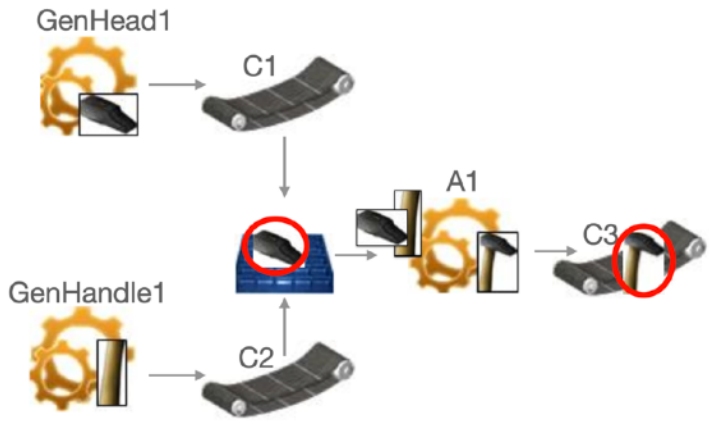
Conforms to

Conforms to

Conforms to







```
@Aspect(className=Assembler)
class AssemblerAspect extends MachineAspect {
    @Step
    @OverrideAspectMethod
    def void start(){
        /**
         * Look if there are products in the tray
         */
        if(_self.in.currentParts.exists[p|p instanceof Head ] &&
            _self.in.currentParts.exists[p|p instanceof Handle ]
        ){
            println("I am in the Assembler machine name '"+_self.name+"'")
            /** Retrieve a Handle product */
            var anHandle=_self.in.currentParts.filter[e|e instanceof Handle].get(0)
            /** Retrieve a Head product */
            var aHead=_self.in.currentParts.filter[e|e instanceof Head].get(0)

            /** Take products from the tray */
            _self.in.currentParts.remove(anHandle)
            _self.in.currentParts.remove(aHead)

            /** Assemble them and generate a Hammer and put it on the conveyer in output */
            var aHammer = runtimestateFactory.eINSTANCE.createHammer
            _self.out.currentParts.add(aHammer)
            return
        }
    }
}
```

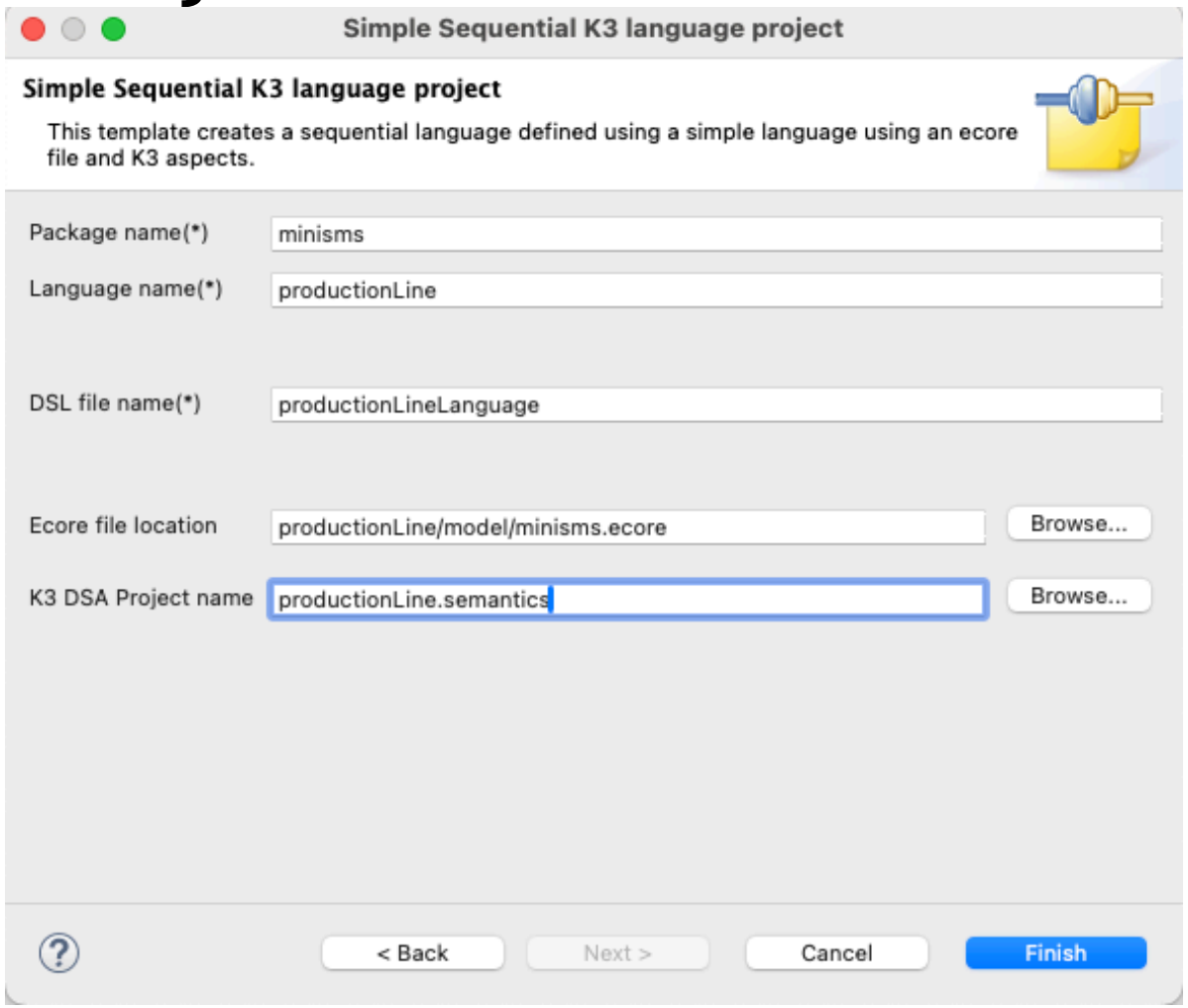
Create the Language Definition Project

Creating a langugage requires to assemble :

- Abstract syntax and,
- Semantics.

This is possible through the GEMOC xDSML project:

- File > New > Other > GEMOC Language > GEMOC Java xDSML Project.
- Give a name to the project as *productionLine.xdsml* > Next > Next.
- For the package name you can enter : *minisms*
- For the language name and the DSL file Name you can keep : the default suggestion.
- Select the Ecore file.
- Select the kermeta 3 project: ***productionLine.semantics***
- Click ***Finish***.



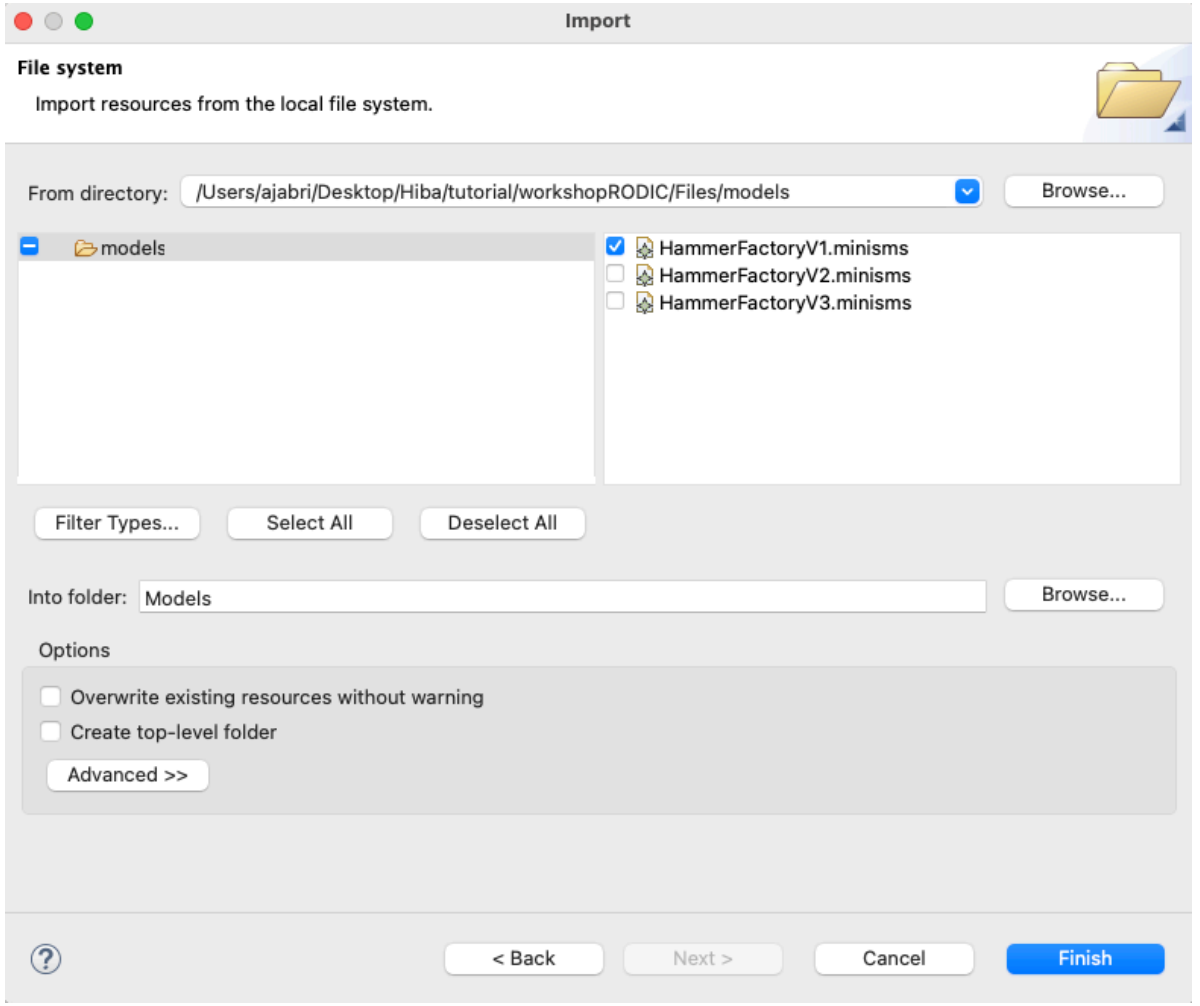
Execute a model conforming to the Language

Now, the language is ready to be used:

- Run > Run Configurations…
- Double click on Eclipse Application > New Configuration.
- Leave default options, and press ***Run***.

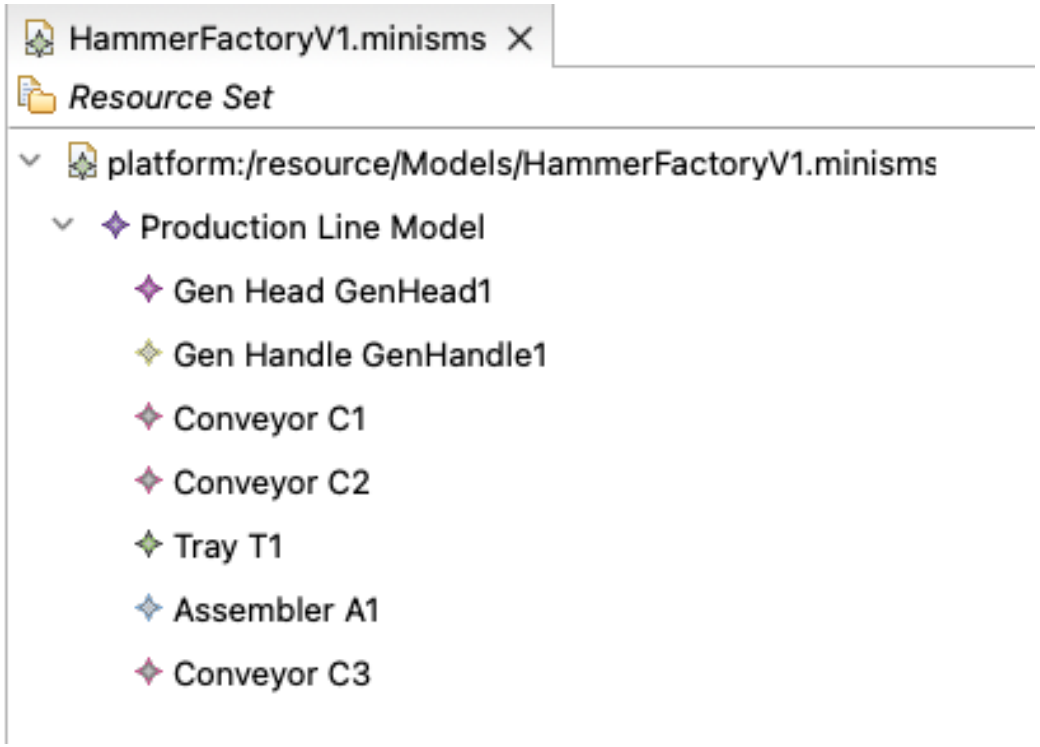
This starts a new Eclipse in an empty workspace – except this new Eclipse contains your compiled DSL.

- Select in the new Eclipse File > New > Other… > General >Project and give a name to the project as ***Models*** and click ***Finish***.
- Click right on the ***Models*** project and select Import.. > File System
- Search for ***models*** folder in the provided files and select ***Open***.
- Select ***HammerFactoryV1.minisms*** and click ***Finish***.
- You may extend the file and look at its components.

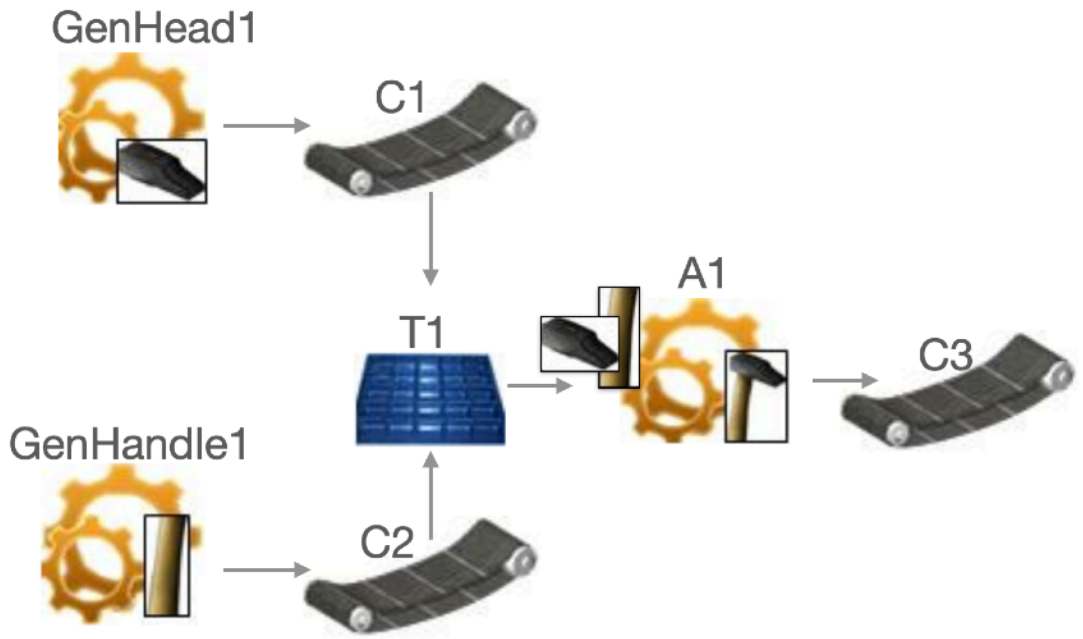


Hammer Model Production Line

Hammer Model with minisms editor



Hammer Model sirius editor

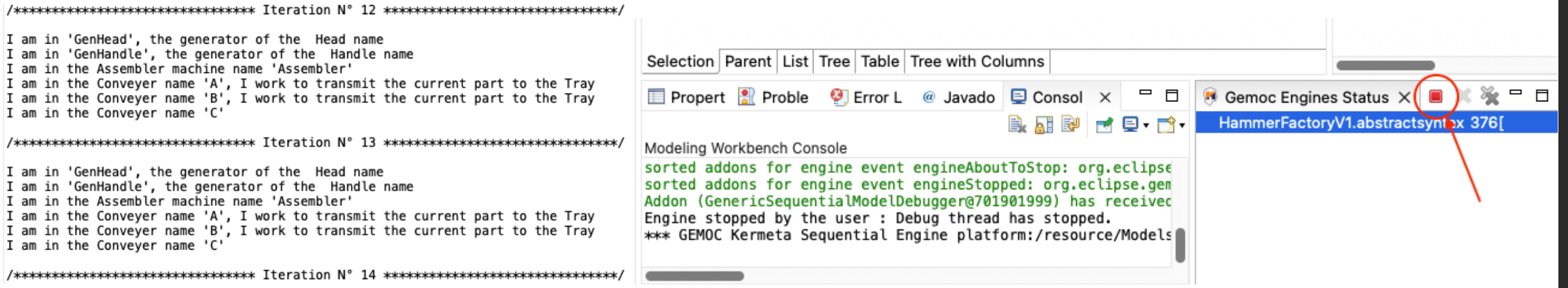


Execute a model conforming to the Language

To execute the model:

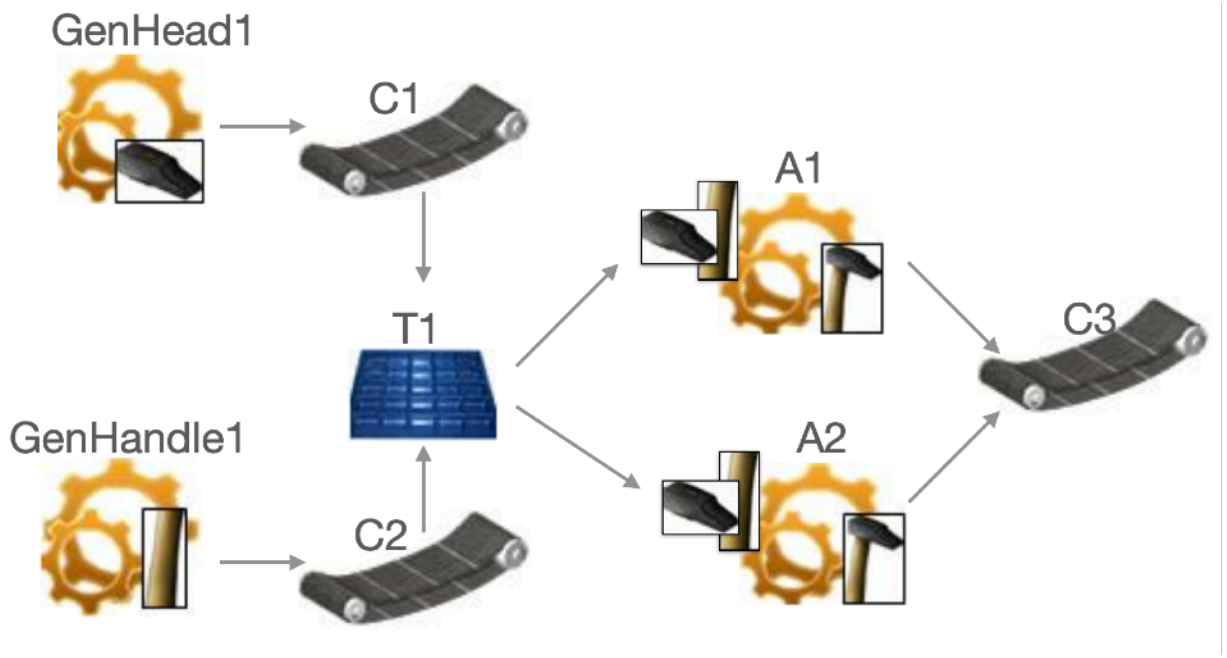
- Run > Run Configurations…
- Double click on Executable model with GEMOC Java engine > New Configuration.
- Select this ***New configuration*** and change its name to ***HammerFactoryV1*** and fill it:
  - Use ***Browse*** to select your ***Model to execute***.
  - Select your language in the drop-down list.
  - Use ***Browse*** in ***Main method*** to select the main method of your semantics.
  - Use ***Browse*** in ***Main model element path*** to select main element of your model.
  - Press ***Run***.

In the console of the first Eclipse, you can see the trace of the execution of the model, and you can see that the execution is infinite, so you can stop it by clicking on the red button of the second Eclipse, as shown in the figure:



Exercises

- Now, you may change the ***HammerFactoryV1*** model by adding a new assembler, that can take products from the tray ***T1*** and put on the conveyor ***C3***.
- The new model is available on the ***models*** folder : the ***HammerFactoryV2.minisms*** model.
- Run the new model.
- What do you observe in the console traces? Does the new assembler work?



Why ?

- New assembler ***A2***, will never be able to work, because :
- At each iteration, only one Head and one Handle are available on the tray ***T1***.
- Thus, only one of the assemblers can work.
- As long as the ***A1*** is prioritized over the ***A2***, the latter will not be able to work during the whole execution.

```
/************************************Execution************************************/

/******************************** Iteration N° 0 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray

/******************************** Iteration N° 1 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Assembler machine name 'A1'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 2 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Assembler machine name 'A1'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 3 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Assembler machine name 'A1'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 4 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Assembler machine name 'A1'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'
```

Semantics Modification

- How can we modify the semantics, so that all elements has equal chance to work?
- You may find the solution in semantics>semantics2 folder and you can replace the ***minismsAspects.xtend*** file:
- You can notice that we have added :

  Collections.shuffle(elements);

- Execute the ***HammerFactoryV2.minisms***.
- What do you notice? Are both assemblers working this time?

```
/******************************** Iteration N° 0 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray

/******************************** Iteration N° 1 ********************************/

I am in the Assembler machine name 'A1'
I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 2 ********************************/

I am in the Assembler machine name 'A2'
I am in 'GenHandle1', the generator of the  Handle name
I am in 'GenHead1', the generator of the  Head name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 3 ********************************/

I am in the Assembler machine name 'A1'
I am in 'GenHead1', the generator of the  Head name
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/******************************** Iteration N° 4 ********************************/

I am in 'GenHead1', the generator of the  Head name
I am in the Assembler machine name 'A1'
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray

/******************************** Iteration N° 5 ********************************/
```

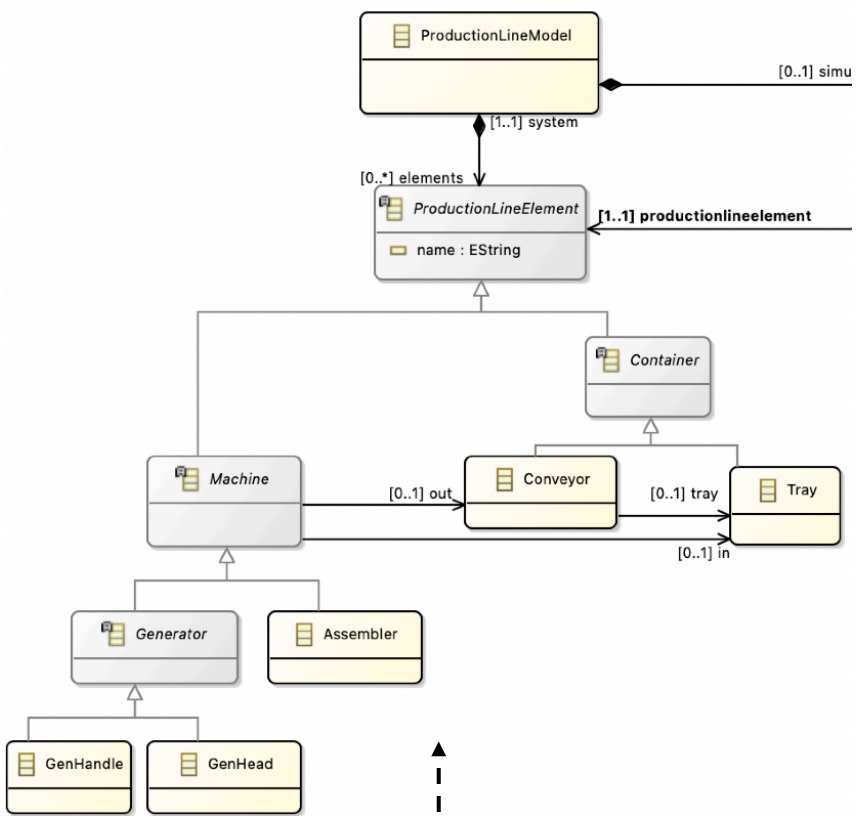
Abstract syntax and Semantics Modification

The execution is infinite.

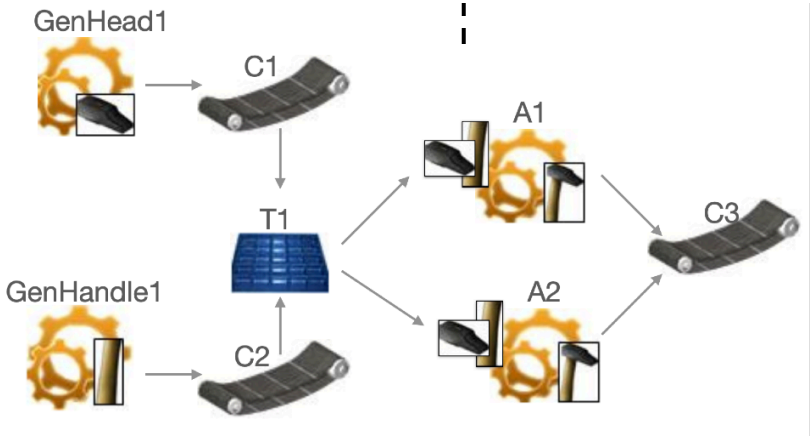
- what would be the solution if we wanted to allow the domain expert to add a stopped condition ?

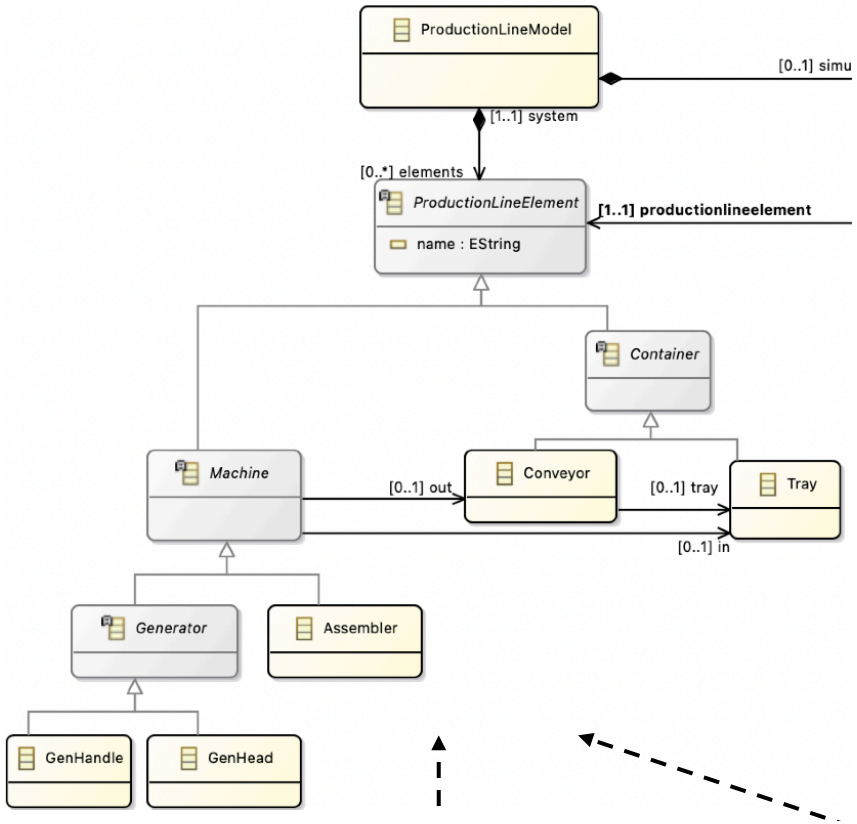
A stopped condition may be :

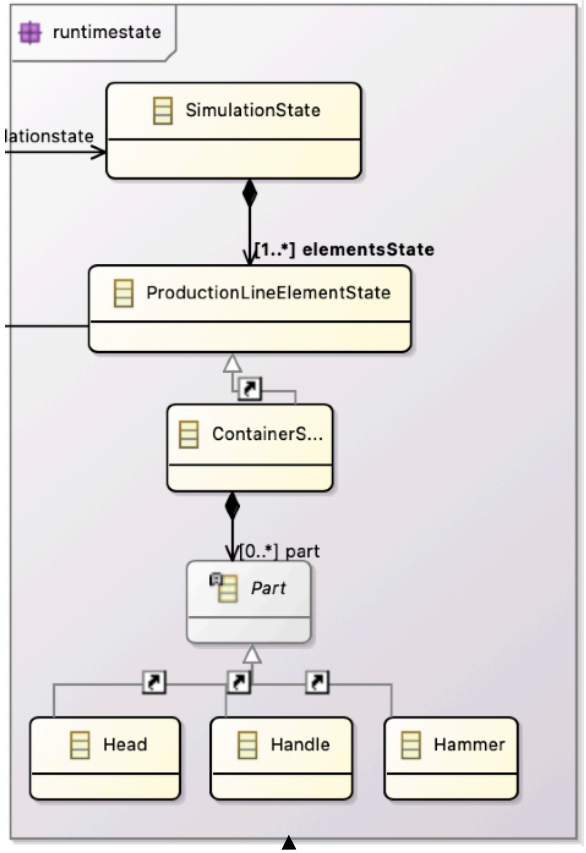
- Stop the simulation in a certain number of iterations given by the domain expert.


ProductionLineModel
[0..1] simu
[1..1] system
[0..*] elements
ProductionLineElement
name : EString
[1..1] productionlineelement
Container
Machine
[0..1] out
Conveyor
[0..1] tray
Tray
[0..1] in
Generator
Assembler
GenHandle
GenHead

Conforms to


GenHead1
C1
A1
T1
C3
GenHandle1
A2
C2

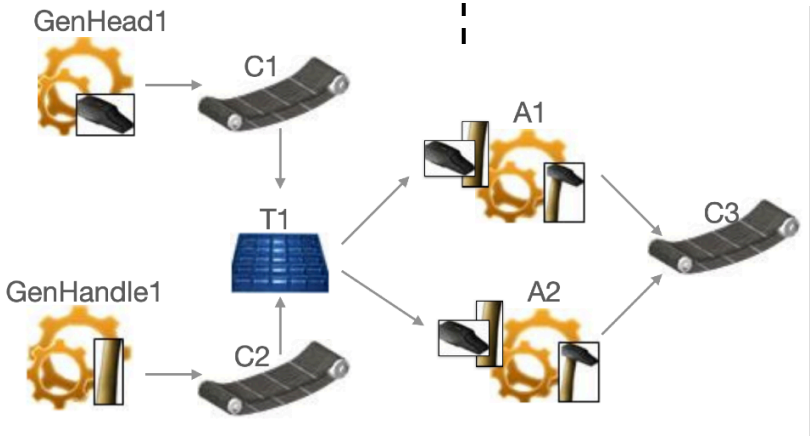


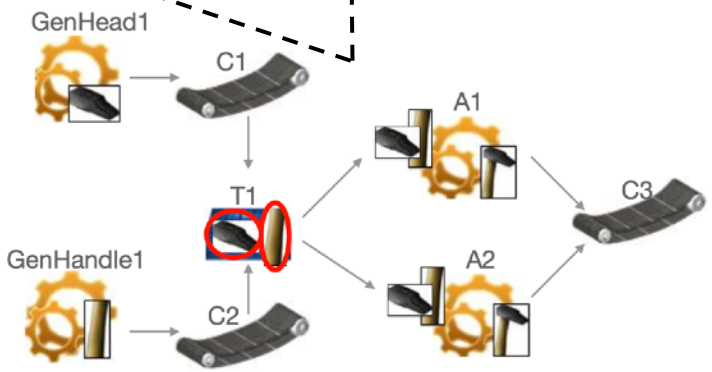


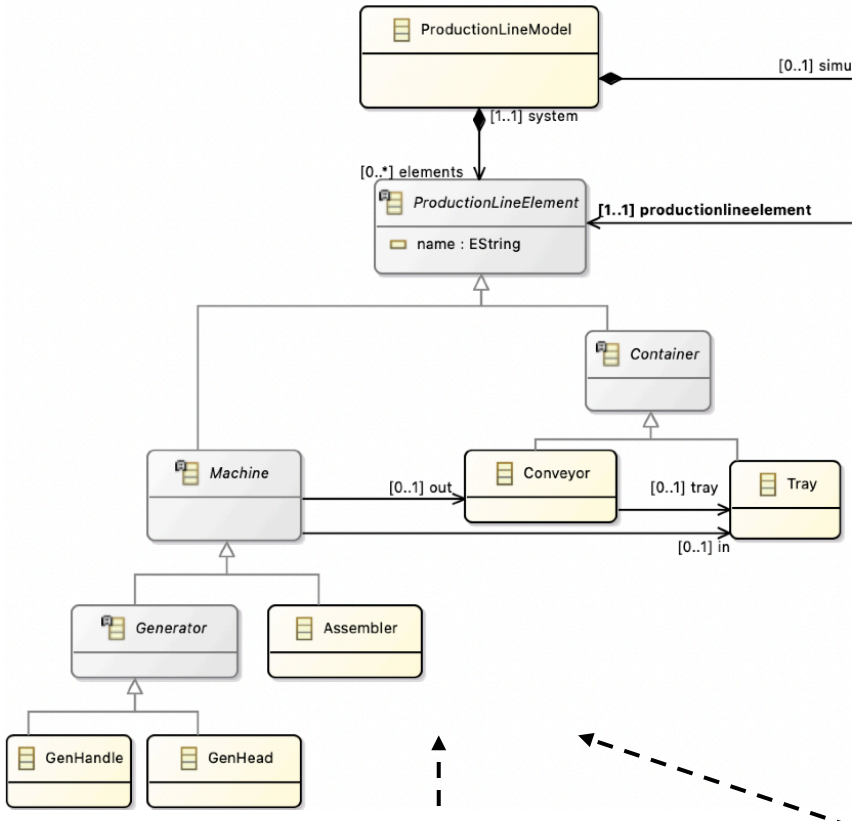
Conforms to

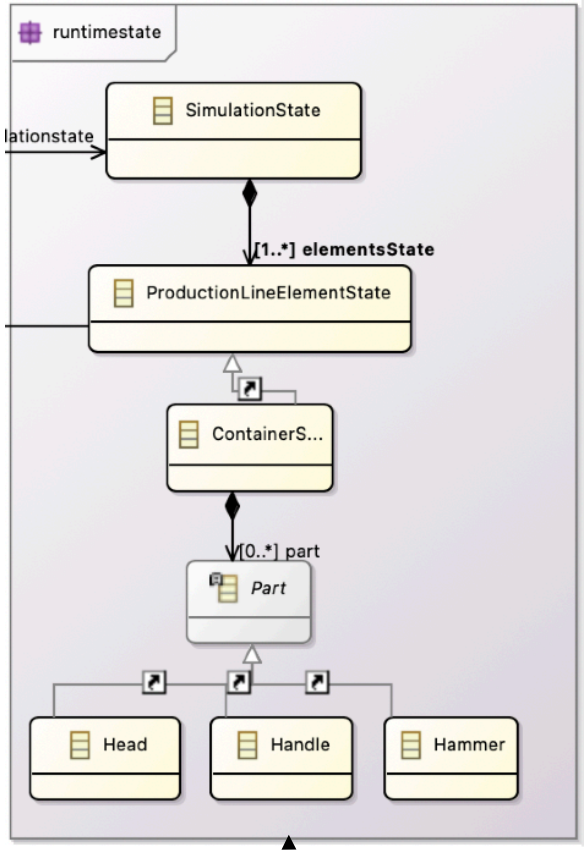
Conforms to

Conforms to







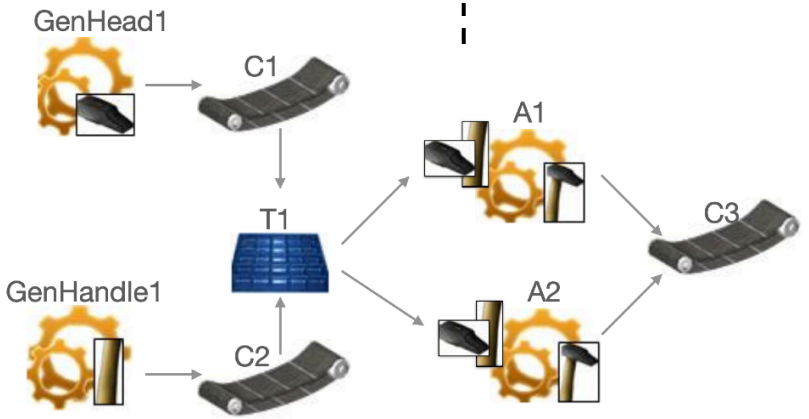


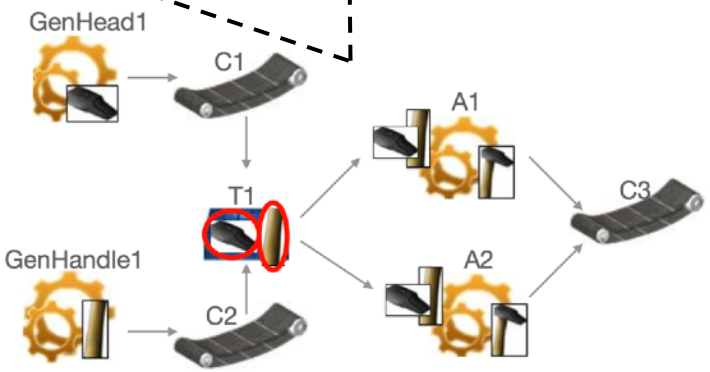
Execution rules Modifications

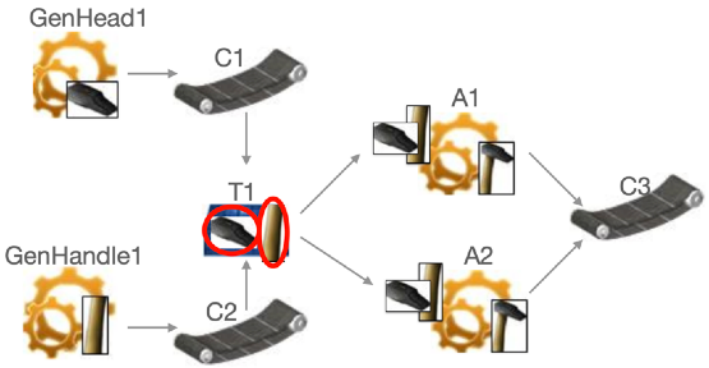
Conforms to

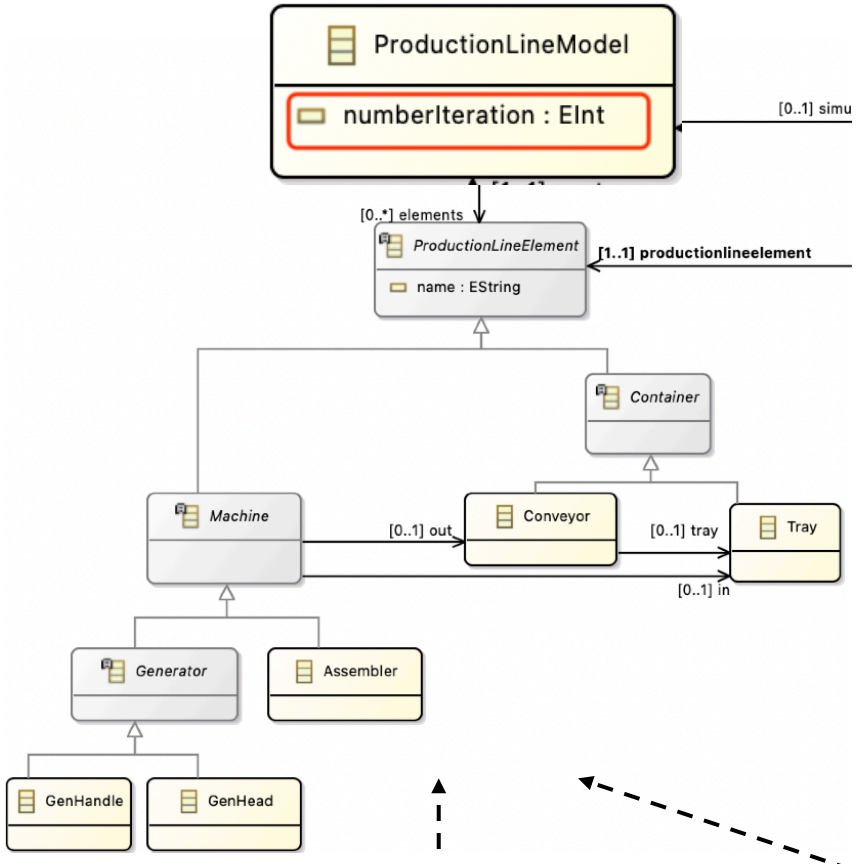
Conforms to

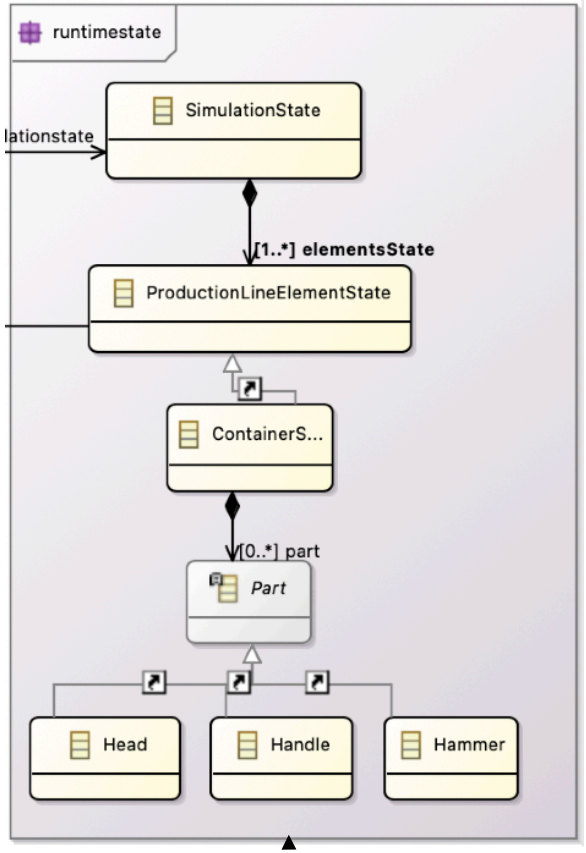
Conforms to









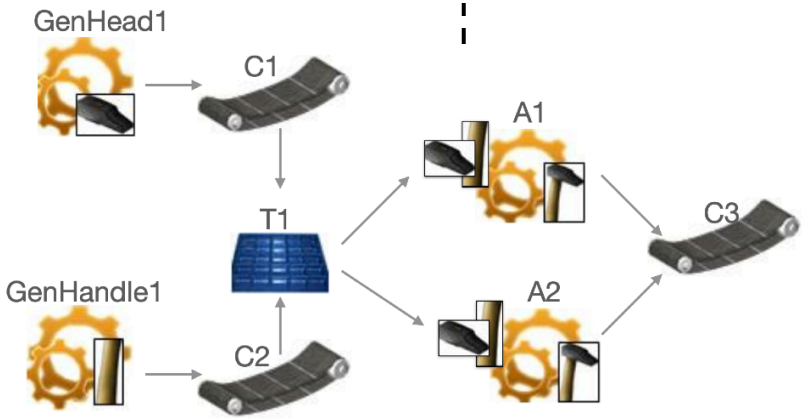


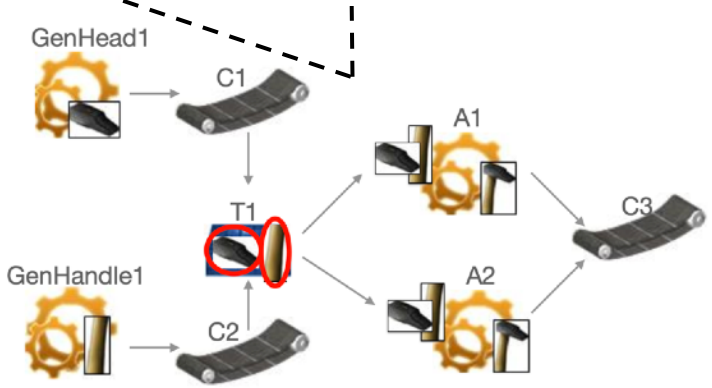
Execution rules Modifications

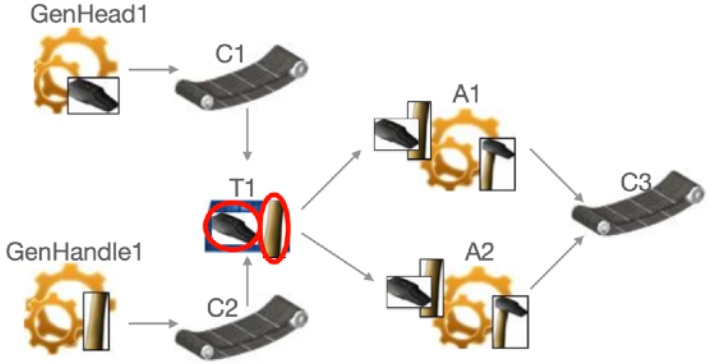
Conforms to

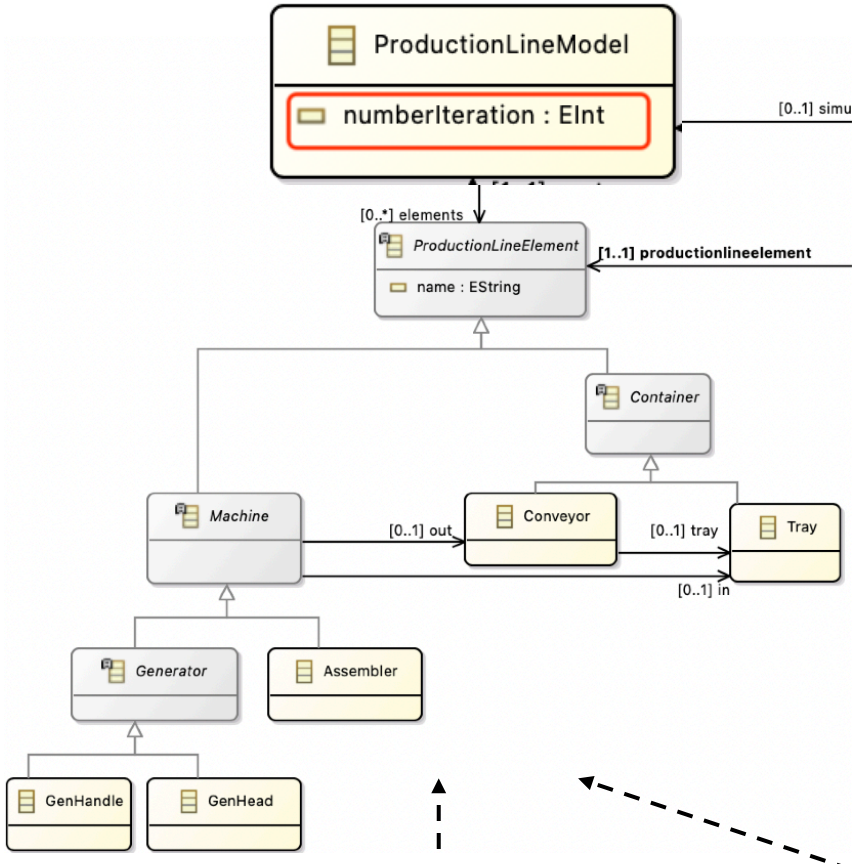
Conforms to

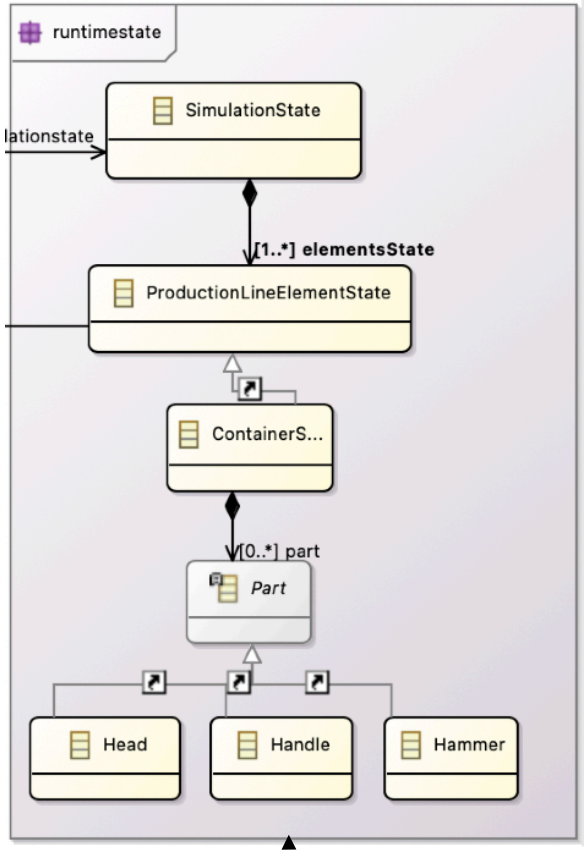
Conforms to









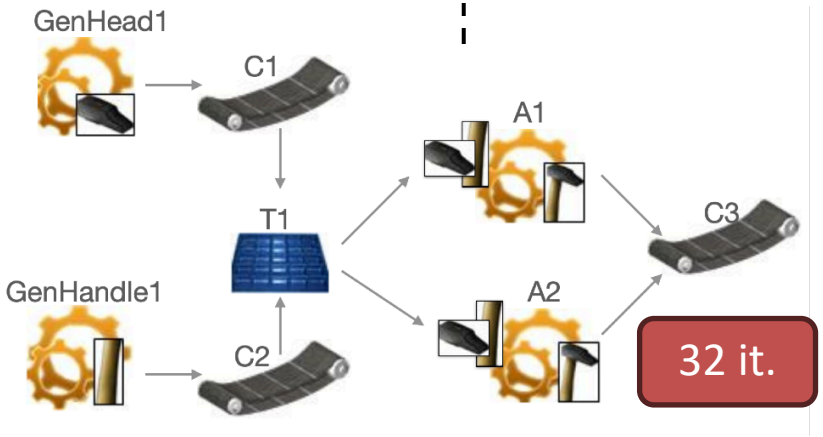


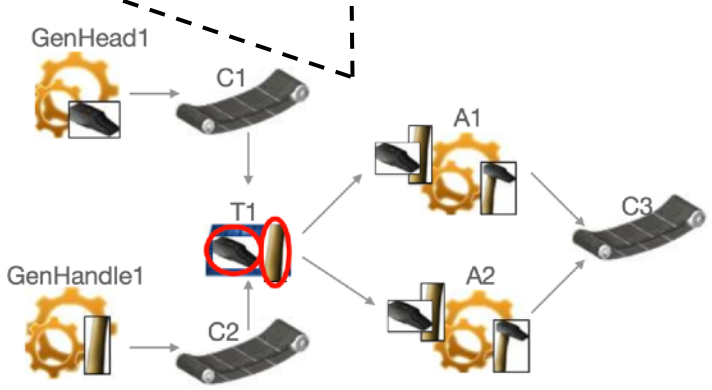
Execution rules Modifications

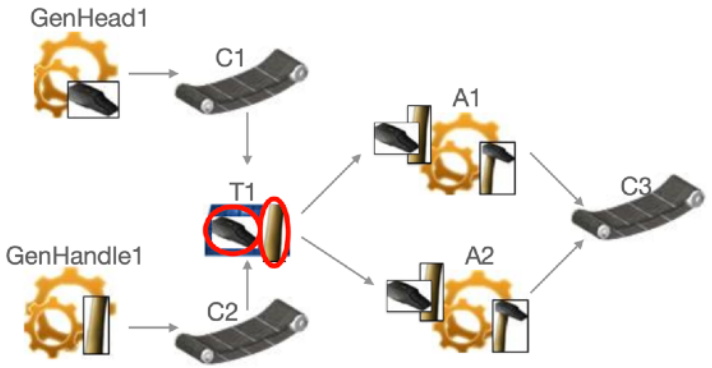
Conforms to

Conforms to

Conforms to







Abstract syntax and Semantics Modification

- The solution at the level of the abstract syntax of the language is available in the ***productionLine*** project on the provided files: EMF projects> EMF project 2.
- The semantics solution is given in the Semantics > Semantics 3.
- And you may execute the ***HammerFactoryV3.minisms*** model provided in models folder. This third version of the HammerFactory model defines a 32 as number of iteration in which the simulation end.

```
I am in the Assembler machine name 'A1'
I am in 'GenHandle1', the generator of the  Handle name
I am in 'GenHead1', the generator of the  Head name
I am in the Conveyer name 'C3'
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray

/********************************* Iteration N° 29 *********************************/

I am in the Assembler machine name 'A1'
I am in 'GenHandle1', the generator of the  Handle name
I am in 'GenHead1', the generator of the  Head name
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/********************************* Iteration N° 30 *********************************/

I am in the Assembler machine name 'A1'
I am in 'GenHandle1', the generator of the  Handle name
I am in 'GenHead1', the generator of the  Head name
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'

/********************************* Iteration N° 31 *********************************/

I am in 'GenHead1', the generator of the  Head name
I am in the Assembler machine name 'A1'
I am in 'GenHandle1', the generator of the  Handle name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray

/********************************* Iteration N° 32 *********************************/

I am in the Assembler machine name 'A2'
I am in 'GenHandle1', the generator of the  Handle name
I am in 'GenHead1', the generator of the  Head name
I am in the Conveyer name 'C2', I work to transmit the current part to the Tray
I am in the Conveyer name 'C3'
I am in the Conveyer name 'C1', I work to transmit the current part to the Tray

/*********************************End*********************************/
```