Model Engineering Exam 2 Material

The following document contains some background information on the metamodels, models and the Henshin Textual Syntax that will be used in Exam 2. You can print this document for the second exam or prepare it on a second screen. The metamodels will also be provided as single graphics in the Tuwel course.

Metamodels - APLML

For the practical Atl, Henshin and Xtend examples you will work with the Automated Production Line Modeling Language (APLML) and the Simple Xml Language. You probably already know parts of these metamodels from the tutorials. Make yourself familiar with the APLML Metamodel and the "Hammer Factory" model and the Simple Xml metamodel.

APLML describes automated production lines. Each model describes a factory. In a *Factory* multiple different *ItemTypes* are processed. You should be familiar with the *ItemType* submodel from the tutorial videos. Note that we deleted the class *Marker*.

A *Factory* can have multiple *AssemblyLines*. Each *AssemblyLine* can have multiple sections, which have an input and an output *Depot*, multiple *IndustrialRobots* and one *Program*. Each *IndustrialRobot* has an *EndOfArmTooling*. If the *EndOfArmTooling* is a *Driller* it also contains at least one *DrillTool*. A *Program* consists of multiple *Commands*. A *SplitItem* command defines a *splitLength* as parameter. A *DrillItem* command refers to the used *DrillTool* and the *MoveItem* command referes to the source and the target *AssemblyLineElemet*.

The "Hammer Factory" is a model conforming to APLML.

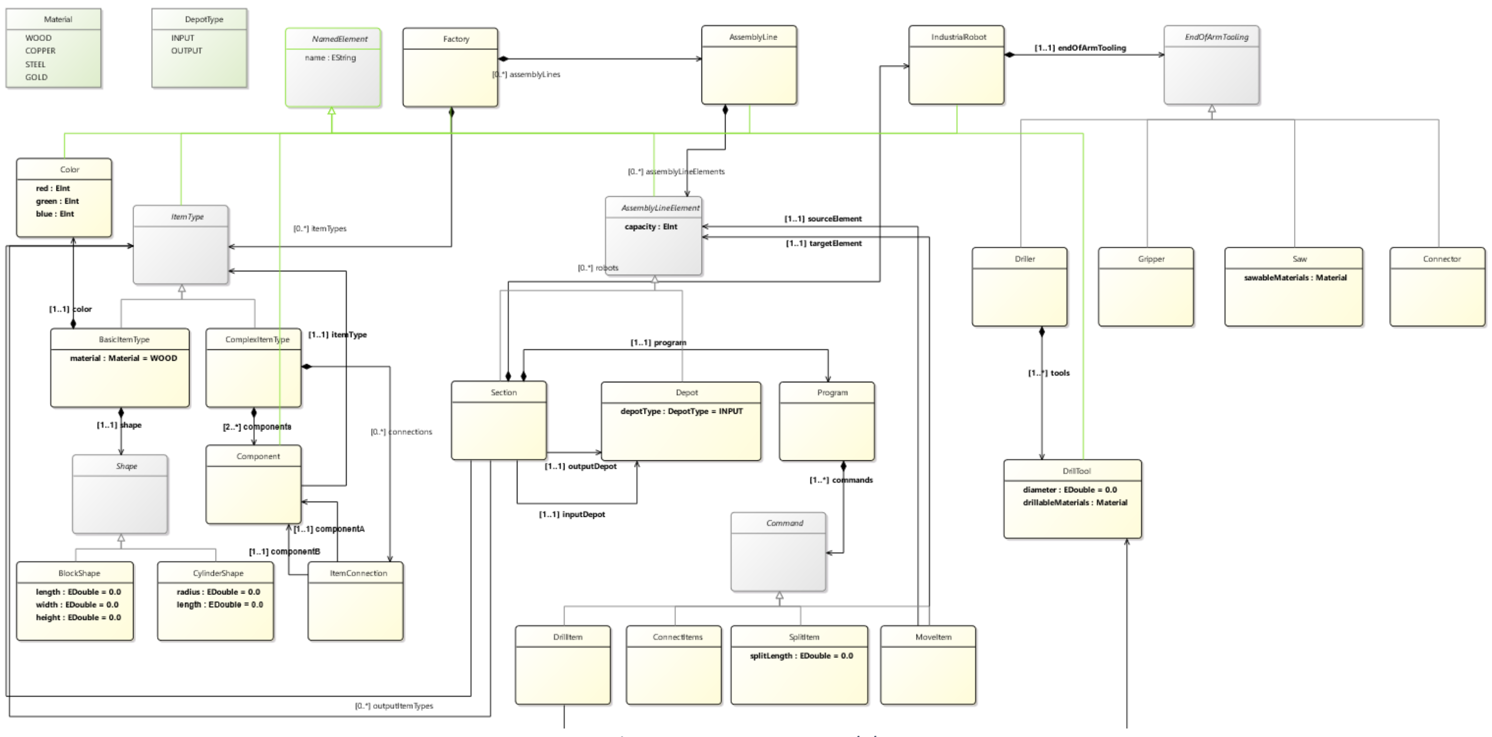
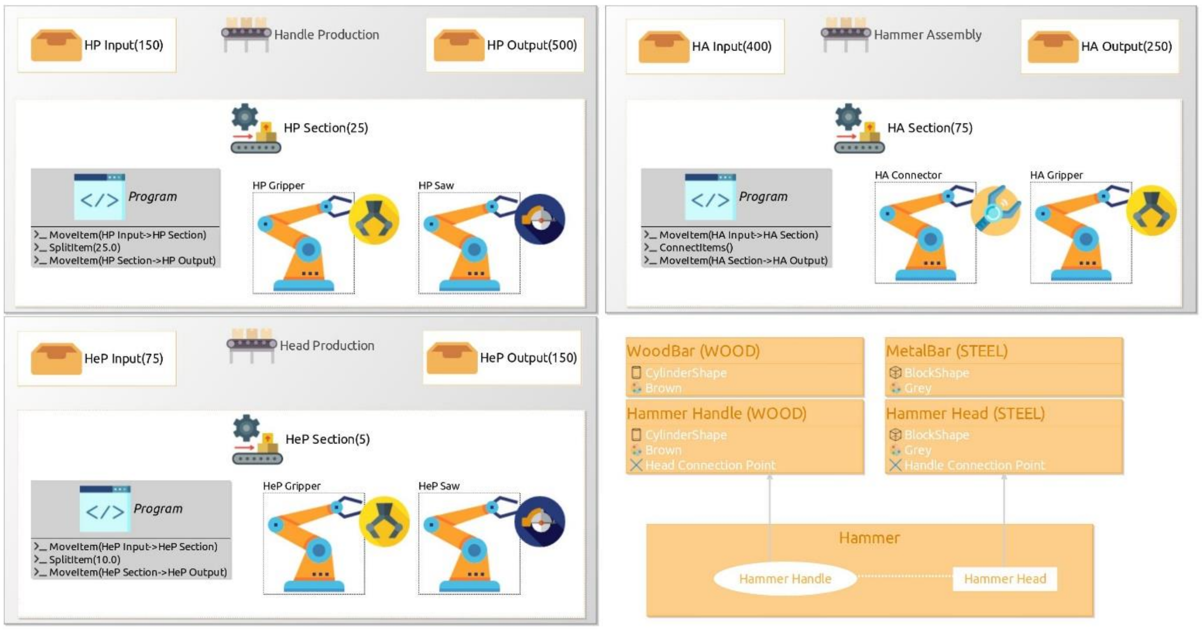


Figure 1 APLML Metamodel



```
>_ MoveItem(HP Input->HP Section)
>_ SplitItem(25.0)
>_ MoveItem(HP Section->HP Output)
```

```
>_ MoveItem(HA Input->HA Section)
>_ ConnectItems()
>_ MoveItem(HA Section->HA Output)
```

```
>_ MoveItem(HeP Input->HeP Section)
>_ SplitItem(10.0)
>_ MoveItem(HeP Section->HeP Output)
```

Figure 2 Hammer Factory Model conforming to APLML

Metamodels – Simple Xml

With Simple Xml basic Xml documents can be modeled. In Simple Xml elements can either have a text value or contain other elements. All elements can have attributes assigned to them. In the following example you can see how a Simple Xml model represented as object diagram maps to a Xml representation.

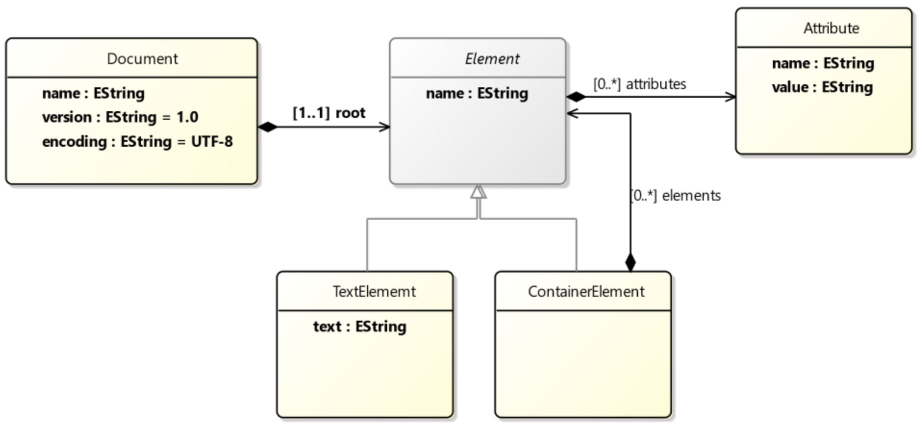


Figure 3 Simple Xml Metamodel

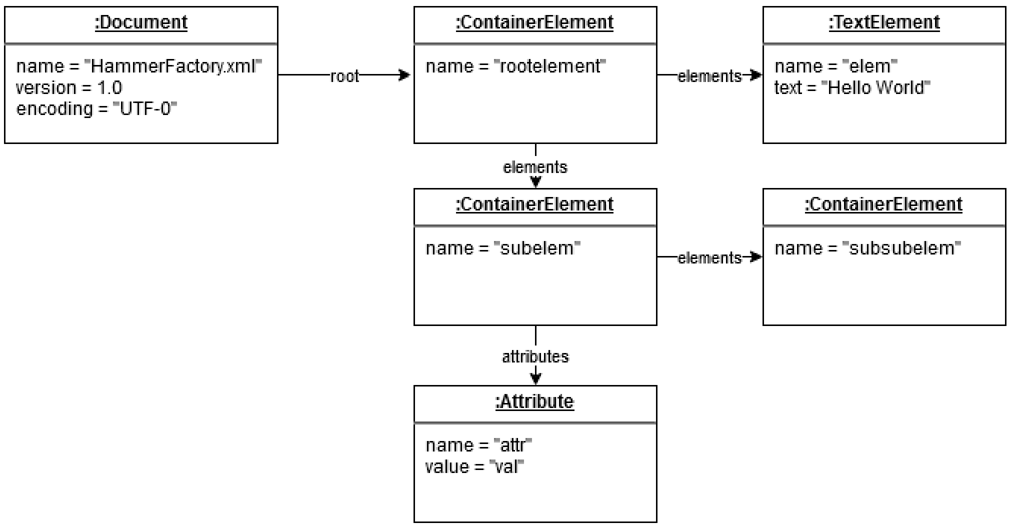


Figure 4 Simple Xml Example Model

Simple Xml Example – Xml representation:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rootelement>
    <elem>Hello World</elem>
    <subelem attr="val">
        <subsubelem/>
    </subelem>
</rootelement>
```

Henshin – Textual Syntax

In the exam you will have to define a Henshin Rule in the textual syntax. Basically, you define a Henshin graph with nodes and edges in a textual form. The example bellow shows a textual solution of the Scenarios 2 and 3b of the lab3 assignment. These two rules show you all language concepts you will need in the exam. For further information we provided you a full textual Henshin solution of the lab3 in the lab 3 sample solution Github repository. You can also find some information in the textual Henshin documentation.

```
ePackageImport sbsml


/* Scenario 2 */
rule RemoveUnusedSensorNodes() {
	graph {
		multiRule RemoveUnusedSensorNodes {
			graph {
				preserve node configuration: Configuration
				preserve node controller: Controller
				delete node sensorNode: Node
				preserve node sensor: Sensor
				forbid node threshold: Threshold

				edges [
					(delete configuration->sensorNode: nodes),
					(preserve configuration->controller: controllers),
					(delete sensorNode->sensor: thing),
					(forbid controller->threshold: threshold),
					(forbid threshold->sensorNode: source)
				]
			}
		}
	}
}


/* Scenario 3b */
rule MoveController(IN nodeName:EString, IN controllerName:EString) {
	graph {
		preserve node controller: Controller {
			name=controllerName
		}
		preserve node fogNode: Node {
			name=nodeName
		}
		preserve node fogDevice: FogDevice

		edges [
			(create controller->fogNode: computationNode),
			(preserve fogNode->fogDevice: thing)
		]
	}
}
```