# User Manual

For

# Vehicular Architectural Blended Modeling in EAST-ADL (Timing and Variability Packages)

## 1. Introduction

Blended modeling is an emerging system development technique in Model Based System Engineering (MBSE). Unlike traditional modeling through graphical notations, blended modeling allows the development of models by mixing the graphical and textual notations interchangeably. Furthermore, it provides seamless runtime switching between graphical and textual notations i.e. the graphical notations can be switched to equivalent textual notations and vice versa. As a result, different stakeholders like developers and system engineers can contribute simultaneously through programming-like textual and graphical notations respectively. This provides much-need modeling flexibility to boost collaboration and modeling time.

EAST-ADL (Electronics Architecture and Software Technology – Architecture Description Language) was first introduced in the ITEA EAST-EEA project. It is a Domain Specific Modeling Language (DSML) that allows the development of a complete system model through four levels of abstraction i.e., Vehicle, Analysis, Design and Implementation level. Furthermore, it comprises several other packages to cater different aspects like variability, timing in system modeling. Consequently, EAST-ADL provides the comprehensive modeling of vehicular embedded systems with simplicity. Moreover, it facilitates the early analysis of the system model comprising both hardware and software design. Furthermore, it speeds up overall system development by automatically generating required implementations from the system model with the help of model transformations. Therefore, EAST-ADL is becoming a de-facto standard for developing vehicular embedded systems.

In BUMBLE project, we are motivated to broadly transfer blended modeling benefits in EAST-ADL architecture to enhance automotive embedded systems development. The overview of the proposed blended modeling approach is shown in **Figure 1**. On top of EAST-ADL architecture (meta-model), several tools and frameworks like EATOP, RUBUS are developed supporting system modeling in a graphical editor. Here, we advocate the use of graphical editor from existing EAST-ADL toolchains where the system model can be represented in EAXML format, as shown in **Figure 1**. For the development of a textual editor, we propose the Xtext platform, where grammars can be specified for EAST-ADL architectural packages of choice like timing, variability, requirements and so forth. This allows modeling flexibility as grammars can be written to support programming language or natural language alike syntax in the textual editor as per stakeholder's requirements. We propose the EAXML standard as a central synchronization point between graphical and textual editors during blended modeling. Particularly, the system model from the graphical editor in EAXML format can be deserialized to extract the required information. Subsequently, the changes from the graphical editor can be propagated in the textual editor through model transformations, as shown in **Figure 1**. On the other hand, changes in the textual editor can be serialized back to EAXML format and systematically merged with graphical instance model (in EAXML). Subsequently, the changes from textual editor are propagated to the graphical editor. Consequently, the proposed approach allows EAST-ADL architectural blended modeling through both graphical and textual editors simultaneously. Furthermore, it can be applied broadly as EAXML is a standard model exchange format for EAST-ADL and is well-supported in most of the existing tools.

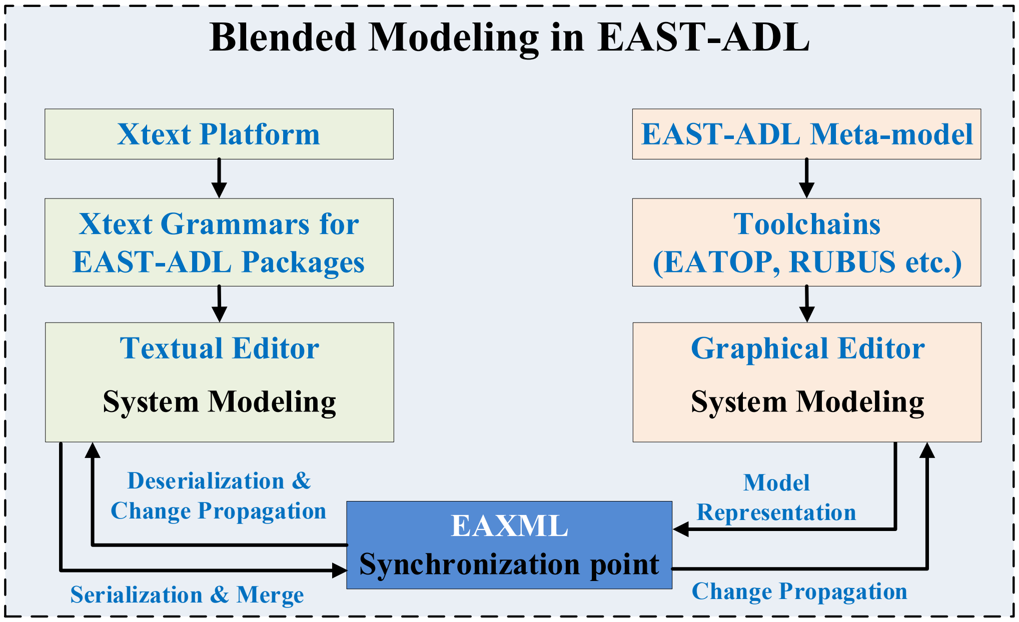


**Figure 1**: Overview of proposed blended modeling approach in EAST-ADL architecture

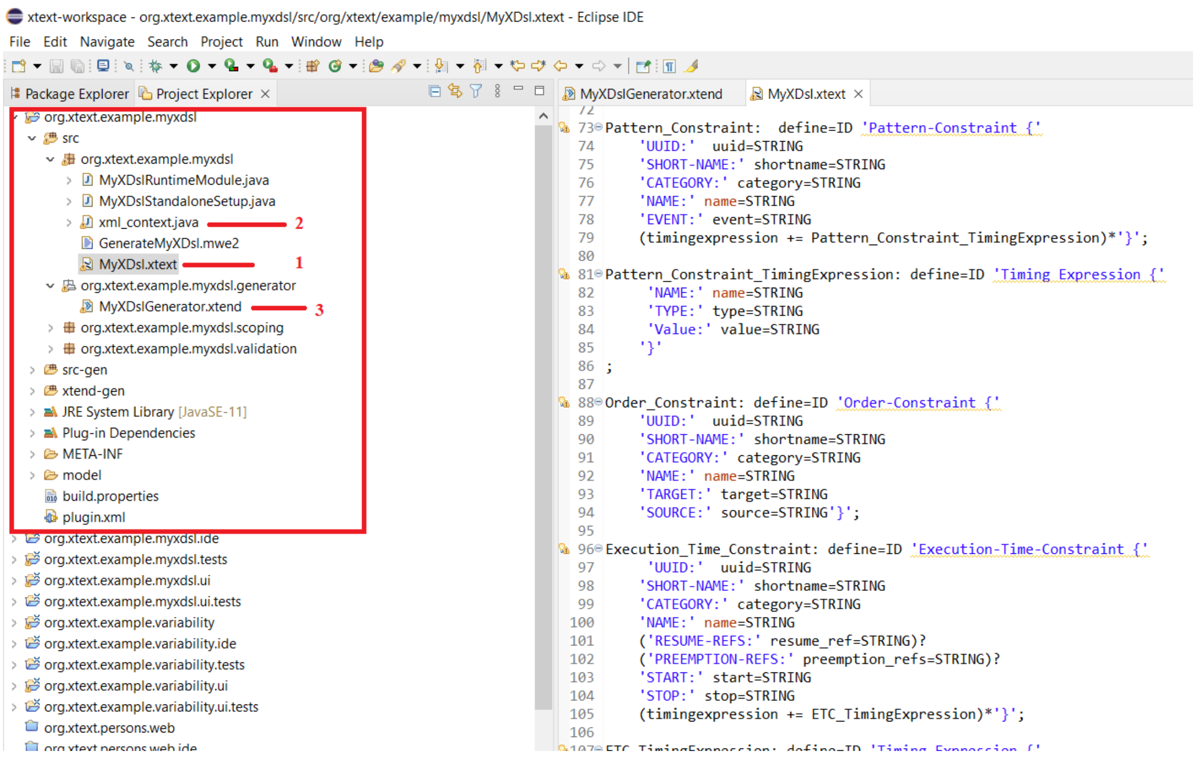
In this manual, we demonstrate the usage / working of proposed blended modeling approach through a car wiper use case from industrial partner – Volvo. Particularly, Xtext grammars comprises EAST-ADL timing and variability concepts are implemented to provide textual editors with programming language-like syntax. Moreover, the EATOP tool having a tree-based editor is utilized for graphical modeling. Finally, blended modeling of car wiper use case is accomplished simultaneously through textual (Xtext) and graphical (EATOP) editors.

It is pertinent to mention that all components of proposed approach along with the detailed installation guide are publicly available at GitHub i.e. https://github.com/MDH-BUMBLE/EASTADL-Blended. Furthermore, video demo is available at: https://drive.google.com/file/d/1HTrXkQ7M6jPnkFq8RgZ9tDoSNSa5wk_f/view?usp=share_link. We describe the essential steps to achieve the blended modeling of EAST-ADL timing constraints and variability in Section 2.

## 2. Blended Modeling of EAST-ADL Timing Constraints and Variability

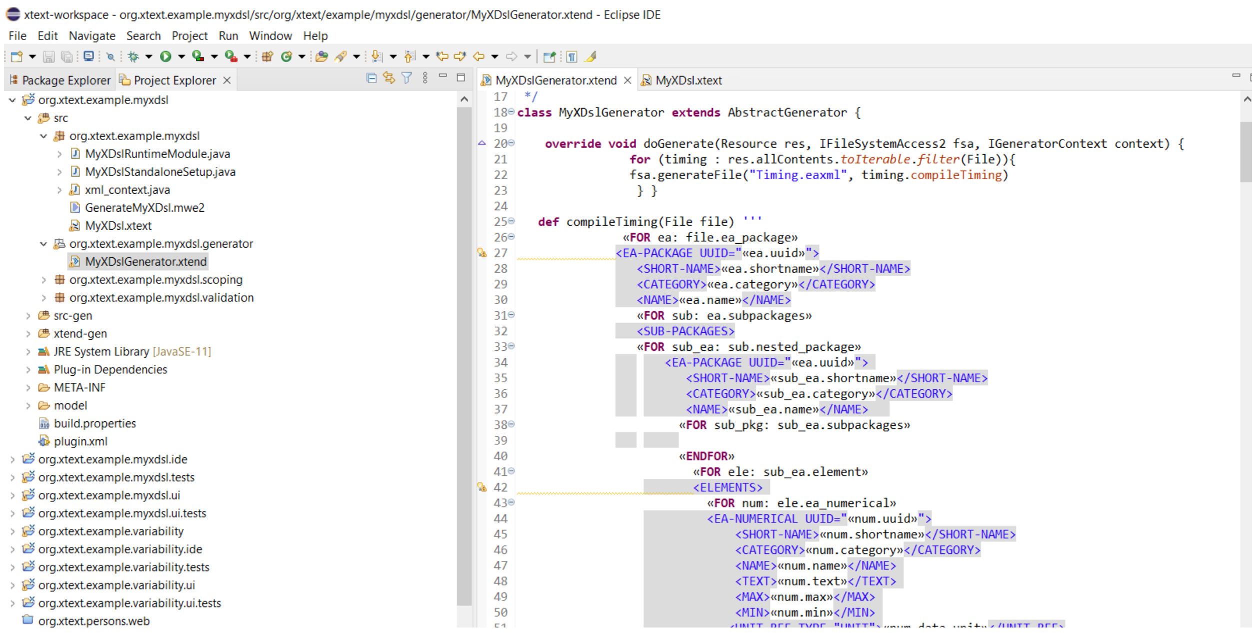
Steps to execute blended modeling of EAST-ADL timing constraints are as follows:

**Step 1 - Components Description**:- We develop Xtext grammar for EAST-ADL timing constraints as shown in **Figure 2**. The grammar rules are inline with the EAST-ADL timing meta-model concepts. The change propagation from graphical to the textual editor is achieved through xml_context.java where different prototypes from graphical editor are accessible in textual editor. The transformation rules to generate EAXML from textual instance are implemented in MyXDslGenerator.xtend as shown in **Figure 3**. Finally, we achieve change propagation from textual to the graphical editor through integration_timing component (discussed later).


xtext-workspace - org.xtext.example.myxdsl/src/org/xtext/example/myxdsl/MyXDsl.xtext - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
Project Explorer
MyXDslGenerator.xtend
MyXDsl.xtext
org.xtext.example.myxdsl
src
org.xtext.example.myxdsl
MyXDslRuntimeModule.java
MyXDslStandaloneSetup.java
xml_context.java
2
GenerateMyXDsl.mwe2
MyXDsl.xtext
1
org.xtext.example.myxdsl.generator
MyXDslGenerator.xtend
3
org.xtext.example.myxdsl.scoping
org.xtext.example.myxdsl.validation
src-gen
xtend-gen
JRE System Library [JavaSE-11]
Plug-in Dependencies
META-INF
model
build.properties
plugin.xml
org.xtext.example.myxdsl.ide
org.xtext.example.myxdsl.tests
org.xtext.example.myxdsl.ui
org.xtext.example.myxdsl.ui.tests
org.xtext.example.variability
org.xtext.example.variability.ide
org.xtext.example.variability.tests
org.xtext.example.variability.ui
org.xtext.example.variability.ui.tests
org.xtext.persons.web

```
Pattern_Constraint:  define=ID 'Pattern-Constraint {'
    'UUID:'  uuid=STRING
    'SHORT-NAME:' shortname=STRING
    'CATEGORY:' category=STRING
    'NAME:' name=STRING
    'EVENT:' event=STRING
    (timingexpression += Pattern_Constraint_TimingExpression)*'}';

Pattern_Constraint_TimingExpression: define=ID 'Timing Expression {'
     'NAME:' name=STRING
     'TYPE:' type=STRING
     'Value:' value=STRING
    '}'
;

Order_Constraint: define=ID 'Order-Constraint {'
    'UUID:'  uuid=STRING
    'SHORT-NAME:' shortname=STRING
    'CATEGORY:' category=STRING
    'NAME:' name=STRING
    'TARGET:' target=STRING
    'SOURCE:' source=STRING'}';

Execution_Time_Constraint: define=ID 'Execution-Time-Constraint {'
     'UUID:'  uuid=STRING
    'SHORT-NAME:' shortname=STRING
    'CATEGORY:' category=STRING
    'NAME:' name=STRING
    ('RESUME-REFS:' resume_ref=STRING)?
    ('PREEMPTION-REFS:' preemption_refs=STRING)?
    'START:' start=STRING
    'STOP:' stop=STRING
    (timingexpression += ETC_TimingExpression)*'}';

```

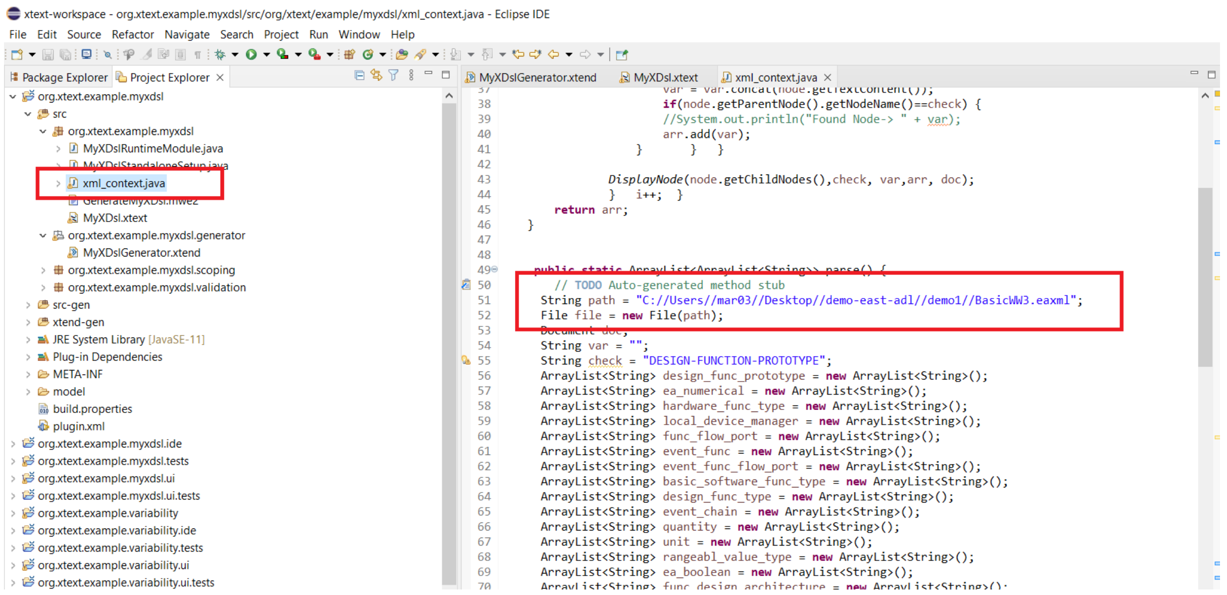
**Figure 2**: EAST-ADL timing grammar in Xtext


xtext-workspace - org.xtext.example.myxdsl/src/org/xtext/example/myxdsl/generator/MyXDslGenerator.xtend - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
Project Explorer
org.xtext.example.myxdsl
src
org.xtext.example.myxdsl
MyXDslRuntimeModule.java
MyXDslStandaloneSetup.java
xml_context.java
GenerateMyXDsl.mwe2
MyXDsl.xtext
org.xtext.example.myxdsl.generator
MyXDslGenerator.xtend
org.xtext.example.myxdsl.scoping
org.xtext.example.myxdsl.validation
src-gen
xtend-gen
JRE System Library [JavaSE-11]
Plug-in Dependencies
META-INF
model
build.properties
plugin.xml
org.xtext.example.myxdsl.ide
org.xtext.example.myxdsl.tests
org.xtext.example.myxdsl.ui
org.xtext.example.myxdsl.ui.tests
org.xtext.example.variability
org.xtext.example.variability.ide
org.xtext.example.variability.tests
org.xtext.example.variability.ui
org.xtext.example.variability.ui.tests
org.xtext.persons.web
MyXDslGenerator.xtend
MyXDsl.xtext

```
 */
class MyXDslGenerator extends AbstractGenerator {

    override void doGenerate(Resource res, IFileSystemAccess2 fsa, IGeneratorContext context) {
                for (timing : res.allContents.toIterable.filter(File)){
                fsa.generateFile("Timing.eaxml", timing.compileTiming)
                 } }

   def compileTiming(File file) '''
               «FOR ea: file.ea_package»
              <EA-PACKAGE UUID="«ea.uuid»">
                 <SHORT-NAME>«ea.shortname»</SHORT-NAME>
                 <CATEGORY>«ea.category»</CATEGORY>
                 <NAME>«ea.name»</NAME>
                 «FOR sub: ea.subpackages»
                 <SUB-PACKAGES>
                 «FOR sub_ea: sub.nested_package»
                     <EA-PACKAGE UUID="«ea.uuid»">
                        <SHORT-NAME>«sub_ea.shortname»</SHORT-NAME>
                        <CATEGORY>«sub_ea.category»</CATEGORY>
                        <NAME>«sub_ea.name»</NAME>
                       «FOR sub_pkg: sub_ea.subpackages»

                       «ENDFOR»
                         «FOR ele: sub_ea.element»
                         <ELEMENTS>
                           «FOR num: ele.ea_numerical»
                           <EA-NUMERICAL UUID="«num.uuid»">
                               <SHORT-NAME>«num.shortname»</SHORT-NAME>
                               <CATEGORY>«num.category»</CATEGORY>
                               <NAME>«num.name»</NAME>
                               <TEXT>«num.text»</TEXT>
                               <MAX>«num.max»</MAX>
                               <MIN>«num.min»</MIN>
                              <UNIT-REF TYPE="UNIT">«num.data.unit»</UNIT-REF>
```

**Figure 3**: Implementation of Transformation Rules in Xtend

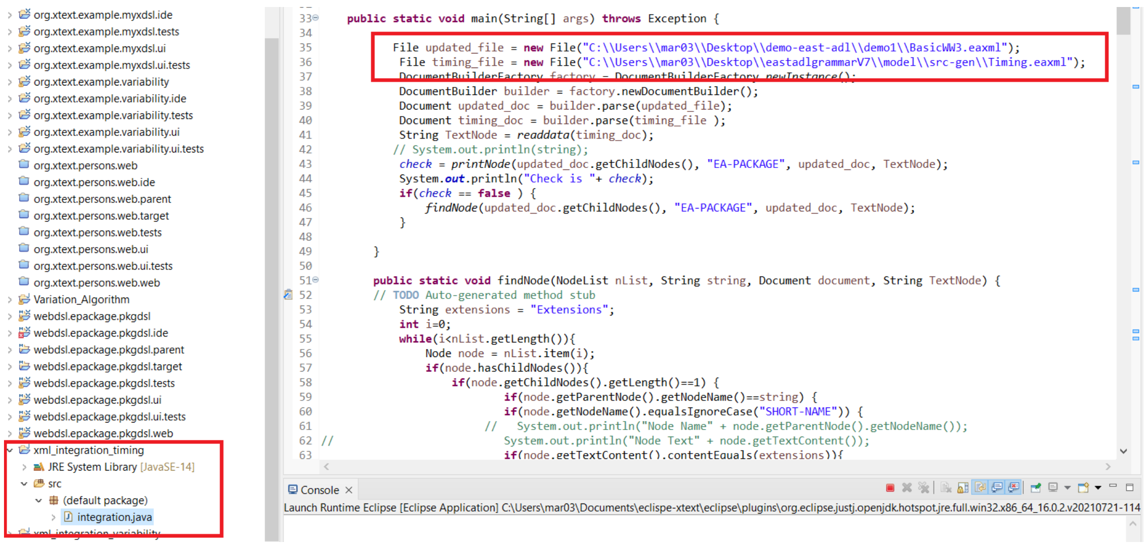
**Step 2 – Update EAST-ADL model (EAXML) Path in Context Assistance**:- The prototypes and other entities from graphical editor are displayed in textual editor through xml_context.java. Therefore, it is essential to update the path of EAST-ADL model (EAXML) – for instance car wiper use case (BasicWW3.eaxml) path. You should update the path as per system configurations. The code portion (xml_context.java) where updates are required is shown in **Figure 4**.


xtext-workspace - org.xtext.example.myxdsl/src/org/xtext/example/myxdsl/xml_context.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Project Explorer
org.xtext.example.myxdsl
src
org.xtext.example.myxdsl
MyXDslRuntimeModule.java
MyXDslStandaloneSetup.java
xml_context.java
GenerateMyXDsl.mwe2
MyXDsl.xtext
org.xtext.example.myxdsl.generator
MyXDslGenerator.xtend
org.xtext.example.myxdsl.scoping
org.xtext.example.myxdsl.validation
src-gen
xtend-gen
JRE System Library [JavaSE-11]
Plug-in Dependencies
META-INF
model
build.properties
plugin.xml
org.xtext.example.myxdsl.ide
org.xtext.example.myxdsl.tests
org.xtext.example.myxdsl.ui
org.xtext.example.myxdsl.ui.tests
org.xtext.example.variability
org.xtext.example.variability.ide
org.xtext.example.variability.tests
org.xtext.example.variability.ui
org.xtext.example.variability.ui.tests
MyXDslGenerator.xtend
MyXDsl.xtext
xml_context.java

```
                    var = var.concat(node.getTextContent());
                    if(node.getParentNode().getNodeName()==check) {
                    //System.out.println("Found Node-> " + var);
                    arr.add(var);
                }       }   }

            DisplayNode(node.getChildNodes(),check, var,arr, doc);
            }   i++;  }
        return arr;
    }


    public static ArrayList<ArrayList<String>> parse() {
        // TODO Auto-generated method stub
     String path = "C://Users//mar03//Desktop//demo-east-adl//demo1//BasicWW3.eaxml";
     File file = new File(path);
     Document doc;
     String var = "";
     String check = "DESIGN-FUNCTION-PROTOTYPE";
     ArrayList<String> design_func_prototype = new ArrayList<String>();
     ArrayList<String> ea_numerical = new ArrayList<String>();
     ArrayList<String> hardware_func_type = new ArrayList<String>();
     ArrayList<String> local_device_manager = new ArrayList<String>();
     ArrayList<String> func_flow_port = new ArrayList<String>();
     ArrayList<String> event_func = new ArrayList<String>();
     ArrayList<String> event_func_flow_port = new ArrayList<String>();
     ArrayList<String> basic_software_func_type = new ArrayList<String>();
     ArrayList<String> design_func_type = new ArrayList<String>();
     ArrayList<String> event_chain = new ArrayList<String>();
     ArrayList<String> quantity = new ArrayList<String>();
     ArrayList<String> unit = new ArrayList<String>();
     ArrayList<String> rangeabl_value_type = new ArrayList<String>();
     ArrayList<String> ea_boolean = new ArrayList<String>();
     ArrayList<String> func_design_architecture = new ArrayList<String>();
```

**Figure 4**: Update wiper case study path in xml_context.java file

**Step 3 – Update models (EAXML) paths in integration_timing** :- The changes from textual to the graphical editor are propagated through integration_timing component. For this, textual instance in EAXML, as generated through xtend transformations, needs to be merged with the main graphical model - EAXML (car wiper case study model). Therefore, for seamless execution of blended approach, you need to update the paths of both graphical and textual models in integration,java file as shown in **Figure 5**. Here, updated_file refers to main graphical model – car wiper EAXML (BasicWW3.eaxml). The timing_file refers to the textual instance in EAXML which is generated through xtend transformations and therefore, it is usually available in Eclipse runtime src-gen folder. You can find the exact path by exploring the properties of generated EAXML in textual editor as shown in **Figure 6**. Note the address of main graphical model needs to be changed at line 164 and 188 of the same file (integration.java).

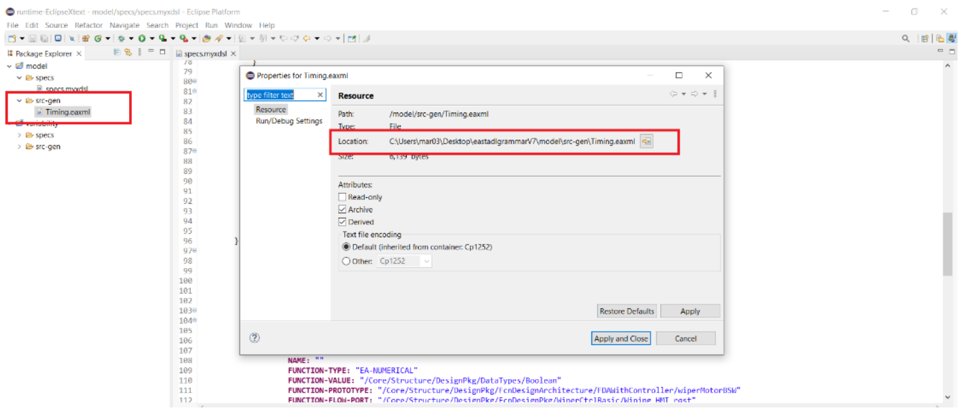


```
    public static void main(String[] args) throws Exception {

        File updated_file = new File("C:\\Users\\mar03\\Desktop\\demo-east-adl\\demo1\\BasicWW3.eaxml");
         File timing_file = new File("C:\\Users\\mar03\\Desktop\\eastadlgrammarV7\\model\\src-gen\\Timing.eaxml");
         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
         DocumentBuilder builder = factory.newDocumentBuilder();
         Document updated_doc = builder.parse(updated_file);
         Document timing_doc = builder.parse(timing_file );
         String TextNode = readdata(timing_doc);
        // System.out.println(string);
         check = printNode(updated_doc.getChildNodes(), "EA-PACKAGE", updated_doc, TextNode);
         System.out.println("Check is "+ check);
         if(check == false ) {
             findNode(updated_doc.getChildNodes(), "EA-PACKAGE", updated_doc, TextNode);
         }

    }

    public static void findNode(NodeList nList, String string, Document document, String TextNode) {
    // TODO Auto-generated method stub
         String extensions = "Extensions";
         int i=0;
         while(i<nList.getLength()){
             Node node = nList.item(i);
             if(node.hasChildNodes()){
                 if(node.getChildNodes().getLength()==1) {
                         if(node.getParentNode().getNodeName()==string) {
                         if(node.getNodeName().equalsIgnoreCase("SHORT-NAME")) {
                     //   System.out.println("Node Name" + node.getParentNode().getNodeName());
//                       System.out.println("Node Text" + node.getTextContent());
                         if(node.getTextContent().contentEquals(extensions)){
```

```
Launch Runtime Eclipse [Eclipse Application] C:\Users\mar03\Documents\eclispe-xtext\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-114
```

org.xtext.example.myxdsl.ide
org.xtext.example.myxdsl.tests
org.xtext.example.myxdsl.ui
org.xtext.example.myxdsl.ui.tests
org.xtext.example.variability
org.xtext.example.variability.ide
org.xtext.example.variability.tests
org.xtext.example.variability.ui
org.xtext.example.variability.ui.tests
org.xtext.persons.web
org.xtext.persons.web.ide
org.xtext.persons.web.parent
org.xtext.persons.web.target
org.xtext.persons.web.tests
org.xtext.persons.web.ui
org.xtext.persons.web.ui.tests
org.xtext.persons.web.web
Variation_Algorithm
webdsl.epackage.pkgdsl
webdsl.epackage.pkgdsl.ide
webdsl.epackage.pkgdsl.parent
webdsl.epackage.pkgdsl.target
webdsl.epackage.pkgdsl.tests
webdsl.epackage.pkgdsl.ui
webdsl.epackage.pkgdsl.ui.tests
webdsl.epackage.pkgdsl.web
xml_integration_timing
JRE System Library [JavaSE-14]
src
(default package)
integration.java
Console

**Figure 5**: Update Paths in integration.java file



Properties for Timing.eaxml

Resource

Path: /model/src-gen/Timing.eaxml

Type: File

Location: C:\Users\mar03\Desktop\eastadlgrammarV7\model\src-gen\Timing.eaxml

Size: 6,139 bytes

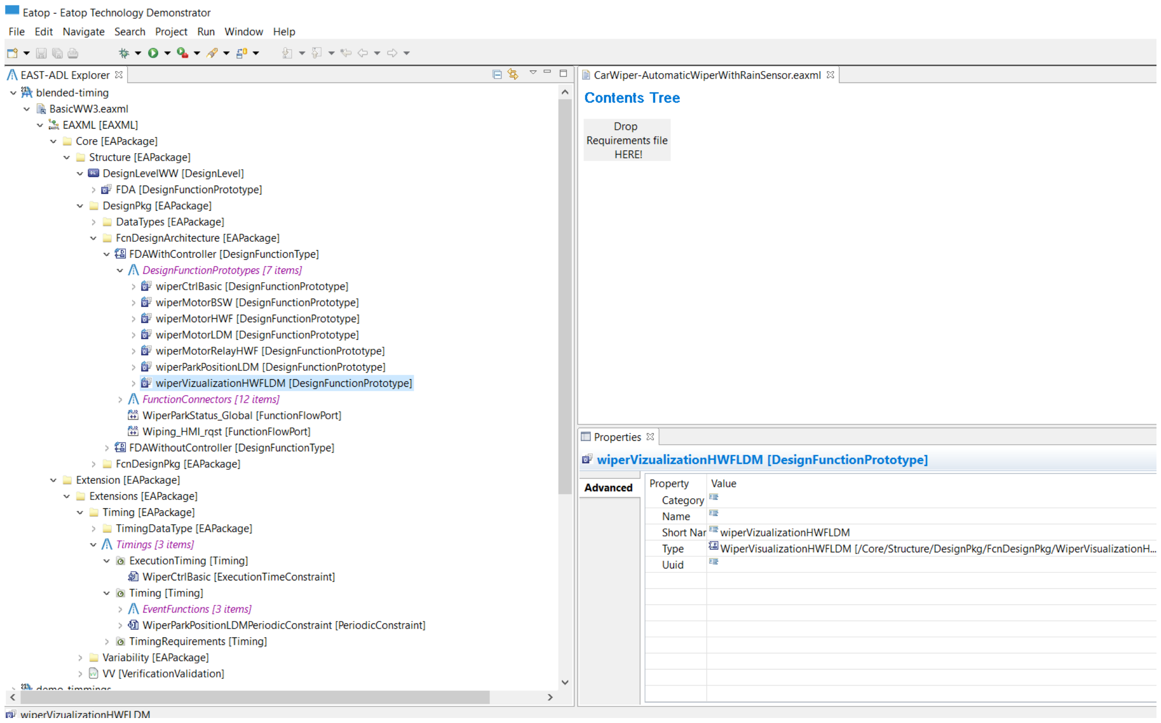
Attributes:
- [ ] Read-only
- [x] Archive
- [x] Derived

Text file encoding
- ● Default (inherited from container: Cp1252)
- ○ Other: Cp1252

```
NAME: ""
FUNCTION-TYPE: "EA-NUMERICAL"
FUNCTION-VALUE: "/Core/Structure/DesignPkg/DataTypes/Boolean"
FUNCTION-PROTOTYPE: "/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperMotorBSW"
FUNCTION-FLOW-PORT: "/Core/Structure/DesignPkg/FcnDesignPkg/WiperCtrlBasic/Wiping_HMI_cost"
```

**Figure 6**: Finding location of textual instance (EAXML) in Eclipse runtime

**Step 4 – Load car wiper case study (EAXML) in EATOP**:- You can load and visualize car wiper case study in EATOP as shown in **Figure 7**.



Eatop - Eatop Technology Demonstrator

File Edit Navigate Search Project Run Window Help

EAST-ADL Explorer

```
blended-timing
  BasicWW3.eaxml
    EAXML [EAXML]
      Core [EAPackage]
        Structure [EAPackage]
          DesignLevelWW [DesignLevel]
            FDA [DesignFunctionPrototype]
          DesignPkg [EAPackage]
            DataTypes [EAPackage]
            FcnDesignArchitecture [EAPackage]
              FDAWithController [DesignFunctionType]
                DesignFunctionPrototypes [7 items]
                  wiperCtrlBasic [DesignFunctionPrototype]
                  wiperMotorBSW [DesignFunctionPrototype]
                  wiperMotorHWF [DesignFunctionPrototype]
                  wiperMotorLDM [DesignFunctionPrototype]
                  wiperMotorRelayHWF [DesignFunctionPrototype]
                  wiperParkPositionLDM [DesignFunctionPrototype]
                  wiperVizualizationHWFLDM [DesignFunctionPrototype]
                FunctionConnectors [12 items]
                WiperParkStatus_Global [FunctionFlowPort]
                Wiping_HMI_rqst [FunctionFlowPort]
              FDAWithoutController [DesignFunctionType]
            FcnDesignPkg [EAPackage]
      Extension [EAPackage]
        Extensions [EAPackage]
          Timing [EAPackage]
            TimingDataType [EAPackage]
            Timings [3 items]
              ExecutionTiming [Timing]
                WiperCtrlBasic [ExecutionTimeConstraint]
              Timing [Timing]
                EventFunctions [3 items]
                WiperParkPositionLDMPeriodicConstraint [PeriodicConstraint]
              TimingRequirements [Timing]
          Variability [EAPackage]
          VV [VerificationValidation]
demo-timmings
wiperVizualizationHWFLDM
```

CarWiper-AutomaticWiperWithRainSensor.eaxml

Contents Tree

Drop Requirements file HERE!

Properties

wiperVizualizationHWFLDM [DesignFunctionPrototype]

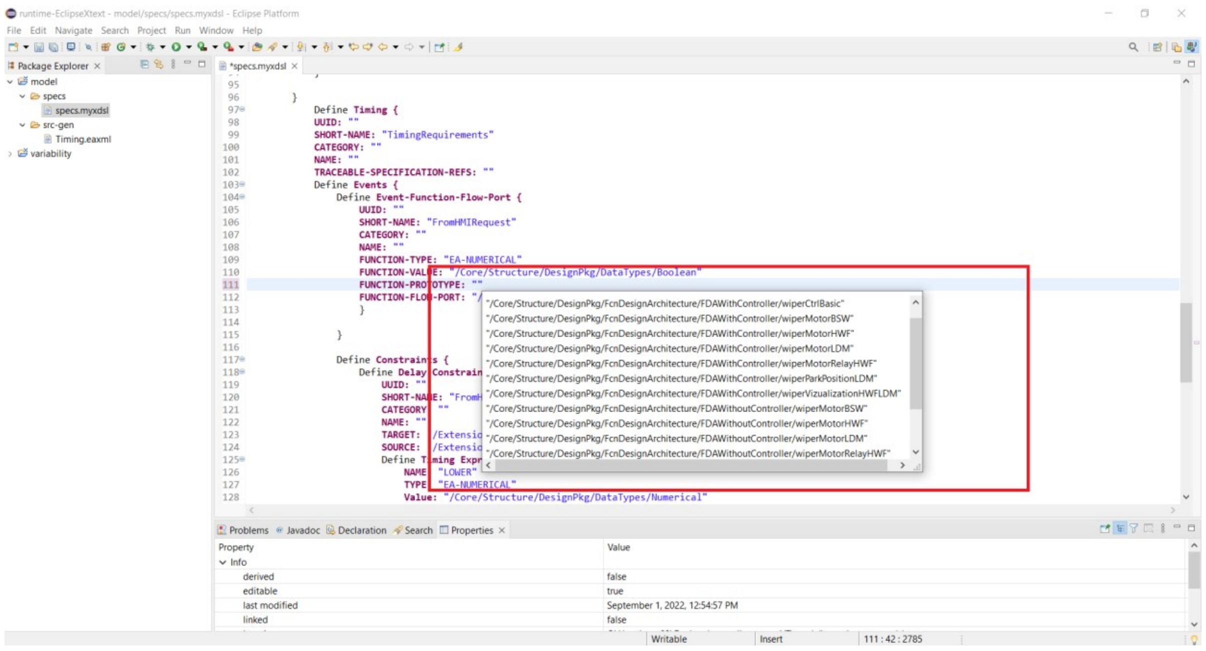
Advanced

| Property | Value |
|---|---|
| Category | |
| Name | |
| Short Nar | wiperVizualizationHWFLDM |
| Type | WiperVisualizationHWFLDM [/Core/Structure/DesignPkg/FcnDesignPkg/WiperVisualizationH... |
| Uuid | |

**Figure 7**: Car wiper case study in EATOP

**Step 4 – Load Textual Editor and Sample Model in Eclipse Runtime**:- We provide sample textual model (TimingRuntime) on GitHub. You can import it in textual editor (Eclipse runtime) and perform experimentation. All prototypes and entities are accessible through context assistant (CTL + Space) as shown in **Figure 8**. Note you need to run integration_timing component whenever you required to propagate changes from textual to graphical editor. Subsequently, you need to refresh EATOP editor to visualize the changes there.

For the blended modeling of EAST-ADL variability, steps are similar. You need to perform these steps on variability components available on GitHub. Furthermore, the same case study can be utilized and sample textual instance is also available for experimentation as shown in **Figure 9**.

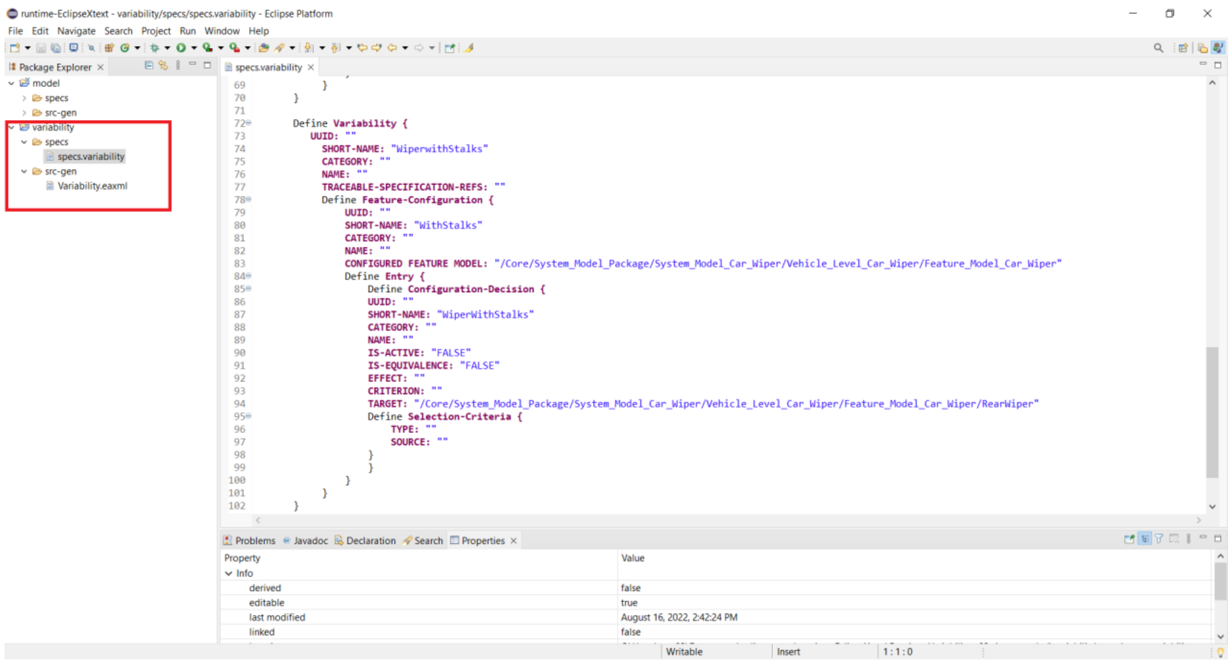


```
        }
            Define Timing {
            UUID: ""
            SHORT-NAME: "TimingRequirements"
            CATEGORY: ""
            NAME: ""
            TRACEABLE-SPECIFICATION-REFS: ""
            Define Events {
                Define Event-Function-Flow-Port {
                    UUID: ""
                    SHORT-NAME: "FromHMIRequest"
                    CATEGORY: ""
                    NAME: ""
                    FUNCTION-TYPE: "EA-NUMERICAL"
                    FUNCTION-VALUE: "/Core/Structure/DesignPkg/DataTypes/Boolean"
                    FUNCTION-PROTOTYPE: ""
                    FUNCTION-FLOW-PORT: "/
                    }

                }

                Define Constraints {
                    Define Delay Constrain
                        UUID: ""
                        SHORT-NAME: "FromH
                        CATEGORY  ""
                        NAME: ""
                        TARGET:  /Extensio
                        SOURCE:  /Extensio
                        Define Timing Expr
                            NAME  "LOWER"
                            TYPE  "EA-NUMERICAL"
                            Value: "/Core/Structure/DesignPkg/DataTypes/Numerical"
```

```
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperCtrlBasic"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperMotorBSW"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperMotorHWF"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperMotorLDM"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperMotorRelayHWF"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperParkPositionLDM"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithController/wiperVizualizationHWFLDM"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithoutController/wiperMotorBSW"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithoutController/wiperMotorHWF"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithoutController/wiperMotorLDM"
"/Core/Structure/DesignPkg/FcnDesignArchitecture/FDAWithoutController/wiperMotorRelayHWF"
```

| Property | Value |
|---|---|
| Info | |
| derived | false |
| editable | true |
| last modified | September 1, 2022, 12:54:57 PM |
| linked | false |

**Figure 8**: Car Wiper Timings Modeling in Textual Editor



```
            }
        }

        Define Variability {
           UUID: ""
             SHORT-NAME: "WiperwithStalks"
             CATEGORY: ""
             NAME: ""
             TRACEABLE-SPECIFICATION-REFS: ""
             Define Feature-Configuration {
                 UUID: ""
                 SHORT-NAME: "WithStalks"
                 CATEGORY: ""
                 NAME: ""
                 CONFIGURED FEATURE MODEL: "/Core/System_Model_Package/System_Model_Car_Wiper/Vehicle_Level_Car_Wiper/Feature_Model_Car_Wiper"
                 Define Entry {
                     Define Configuration-Decision {
                     UUID: ""
                     SHORT-NAME: "WiperWithStalks"
                     CATEGORY: ""
                     NAME: ""
                     IS-ACTIVE: "FALSE"
                     IS-EQUIVALENCE: "FALSE"
                     EFFECT: ""
                     CRITERION: ""
                     TARGET: "/Core/System_Model_Package/System_Model_Car_Wiper/Vehicle_Level_Car_Wiper/Feature_Model_Car_Wiper/RearWiper"
                     Define Selection-Criteria {
                         TYPE: ""
                         SOURCE: ""
                     }
                     }
                 }
             }
        }
```

| Property | Value |
|---|---|
| Info | |
| derived | false |
| editable | true |
| last modified | August 16, 2022, 2:42:24 PM |
| linked | false |

**Figure 9**: Car Wiper Variability Modeling in Textual Editor