HOMEWORK 2

PART 1

The architecture that has been described in this article is the open-source JML-Plugin environment. JML [1] stands for Java Modeling Language, a behavioral specification language. The existing JML platform is a standalone application. My current research work is to incorporate the JML architecture as a plugin extensible component to Eclipse [2] environment.

The following article attempts to formalize this existing architecture as a plugin to the Eclipse environment.

JML4 [3], an Integrated Verification Environment(IVE) is the starting point of my research interest. Though JML4 attempts to come up with an evolutionary tool, it fails to formalize to its lower levels. The architecture of the JML-Plugin is described using Acme Studio [4].

Description:

The features that is chosen so as to exercise some of the basic capabilities that any JML extension to Eclipse would need to support are

- recognizing and processing JML syntax inside specially marked comments, both in *.java files as well as *.jml files;
- storing JML-specific nodes in an extended AST hierarchy,
- statically enforcing a modified type system, and
- providing for runtime assertion checking (RAC).

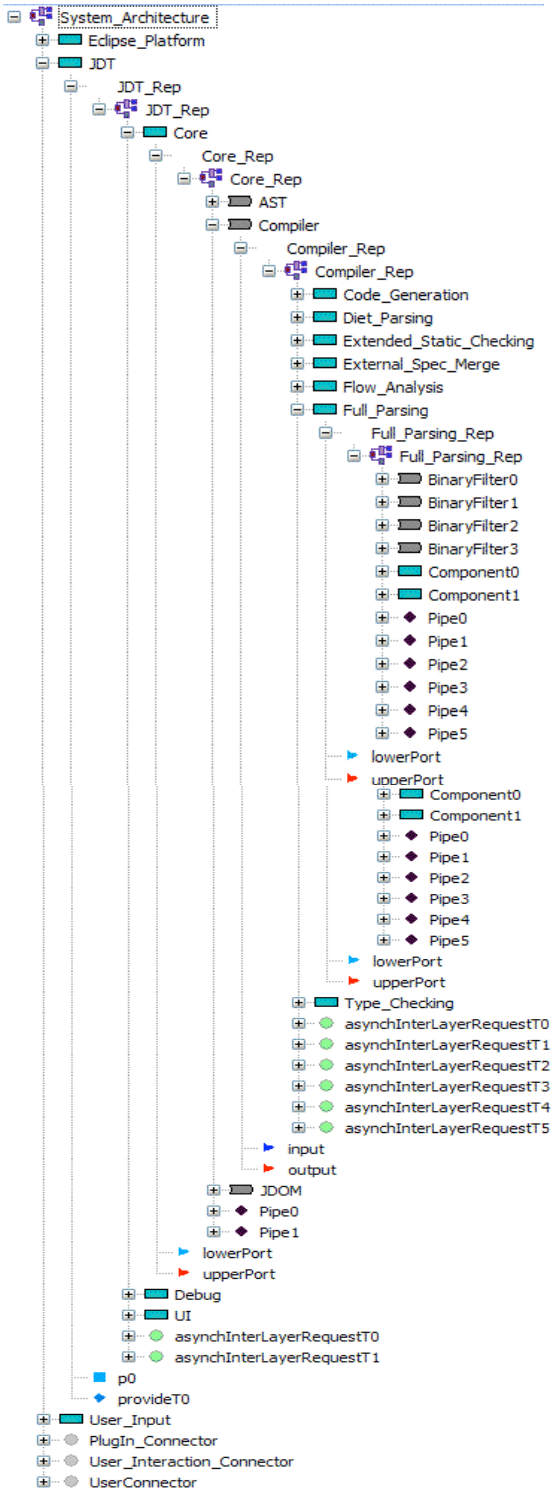
Also, the functionality added by the chosen subset of features is useful in its own right, somewhat independent of other JML features; i.e. the capabilities form a natural extension to the existing embryonic Eclipse support for nullity analysis.

In the remainder of this section, I present the proposed means of extending Eclipse to support JML, appealing at times to the specific way in which the JML4 features described have been realized.

Outline of the Architecture:

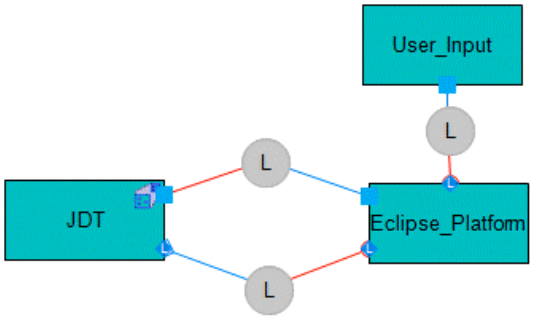
The entire architecture is built on a system of architectures i.e. it is composed of 5 sub-systems each of different style. This article focuses only on the JML aspect and not on the Eclipse platform.

The following diagram enlists the outline of the architecture:


System_Architecture
Eclipse_Platform
JDT
JDT_Rep
JDT_Rep
Core
Core_Rep
Core_Rep
AST
Compiler
Compiler_Rep
Compiler_Rep
Code_Generation
Diet_Parsing
Extended_Static_Checking
External_Spec_Merge
Flow_Analysis
Full_Parsing
Full_Parsing_Rep
Full_Parsing_Rep
BinaryFilter0
BinaryFilter1
BinaryFilter2
BinaryFilter3
Component0
Component1
Pipe0
Pipe1
Pipe2
Pipe3
Pipe4
Pipe5
lowerPort
upperPort
Component0
Component1
Pipe0
Pipe1
Pipe2
Pipe3
Pipe4
Pipe5
lowerPort
upperPort
Type_Checking
asynchInterLayerRequestT0
asynchInterLayerRequestT1
asynchInterLayerRequestT2
asynchInterLayerRequestT3
asynchInterLayerRequestT4
asynchInterLayerRequestT5
input
output
JDOM
Pipe0
Pipe1
lowerPort
upperPort
Debug
UI
asynchInterLayerRequestT0
asynchInterLayerRequestT1
p0
provideT0
User_Input
PlugIn_Connector
User_Interaction_Connector
UserConnector

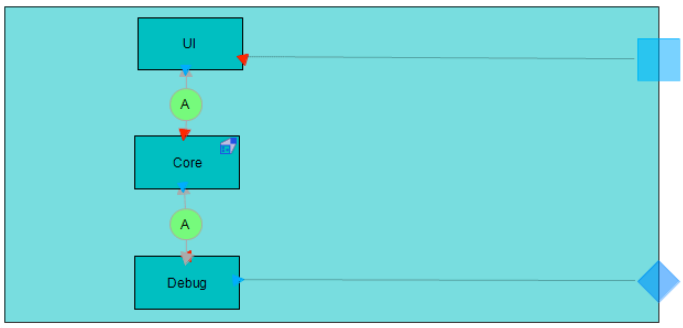
Architectural Diagrams:

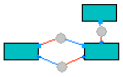
The architecture of the system is given below:



Subhadeep Chakraborty 2/10/08 8:23 PM

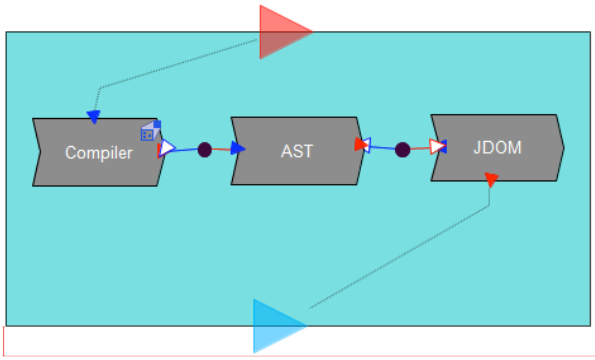
Comment: The top-level design of the system. It consists of 3 components viz. JDT, Eclipse_Platform, User_interface.
The Java Development Tooling is further disintegrated. It depicts a Tiered architecture.





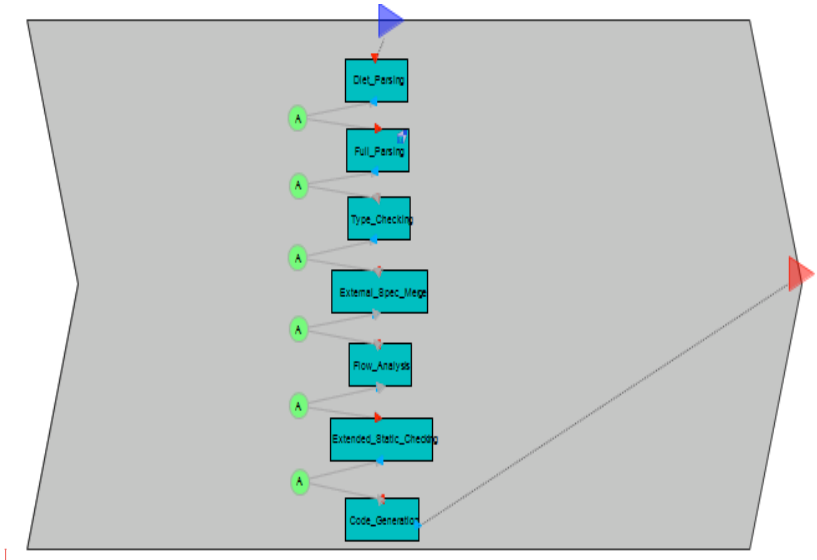
Subhadeep Chakraborty 2/10/08 8:22 PM

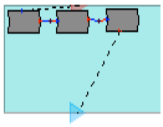
Comment: The JDT itself is composed of 3 layers. The most important is the core. It depicts a LAYERED architecture



Subhadeep Chakraborty 2/10/08 8:34 PM

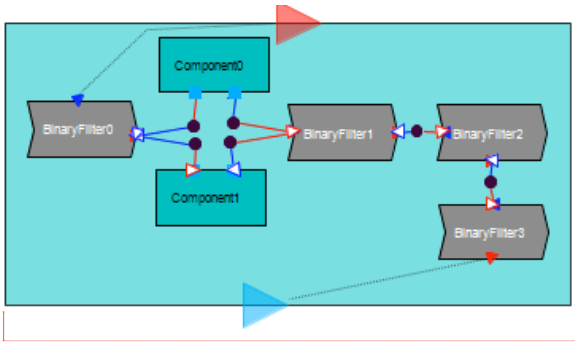
Comment: The core sub-layer is further decomposed into a PIPES-FILTER architecture which contains 3 filters.

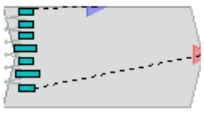




Subhadeep Chakraborty 2/10/08 8:39 PM

Comment: The Compiler is further subdivided into 7 LAYER architecture. The layer Full_Parsing is responsible for the parsing, scanning and lexing the source file.





Subhadeep Chakraborty 2/10/08 8:40 PM

Comment: The innermost layer is composed of 4 Binary filters and 2 components. It is in fact a HETEROGENOUS architecture.

Acme Code:

The following pages describe the JML-Plugin Architecture in Acme using Acme notations following the syntax and semantics rule of Acme.

```
import $AS_GLOBAL_PATH/families/TieredFam.acme;
import $AS_GLOBAL_PATH/families/LayeredFam.acme;
import $AS_GLOBAL_PATH/families/PipesAndFiltersFam.acme;
System System_Architecture : TieredFam = new TieredFam extended with {
```

```
Component JDT : TierNodeT = new TierNodeT extended with {
 Port p0  = {}
 Port provideT0 : provideT = new provideT extended with {}
 Property allowShareHost = true;
 Property host = "JDT Plugin";
 Representation JDT_Rep = {
  System JDT_Rep : LayeredFam = new LayeredFam extended with{
   Component UI : layerT = new layerT extended with {
    Port upperPort  = {
     Property protocol = MSG;
     Property synchronous = false;
    }
    Port lowerPort  = {
     Property protocol = MSG;
     Property synchronous = false;
    }
    Property handlesAsynchRequests = true;
    Property layerLevel = 0;
    Property multiThreaded = true;
    }
   Component Core : layerT = new layerT extended with {
    Port upperPort  = {
     Property protocol = MSG;
     Property synchronous = false;
    }
    Port lowerPort  = {
     Property protocol = MSG;
     Property synchronous = false;
    }
    Property handlesAsynchRequests = true;
    Property layerLevel = 1;
    Property multiThreaded = true;
    Representation Core_Rep = {
     System Core_Rep : PipesAndFiltersFam = new PipesAndFiltersFam
                                            extended with {
      Component Compiler : BinaryFilter = new BinaryFilter extended
                                          with {
       Property function = "Compiles .java file to .class file";
       Representation Compiler_Rep = {
       System Compiler_Rep : LayeredFam = new LayeredFam extended with
       {
       Component Diet_Parsing : layerT = new layerT extended with {
       Port upperPort  = {
        Property protocol = MSG;
        Property synchronous = false;
       }
       Port lowerPort  = {
         Property protocol =MSG;
         Property synchronous = false;
       }
       Property handlesAsynchRequests = true;
       Property layerLevel = 0;
       Property multiThreaded = true;
       }
      Component Full_Parsing : layerT = new layerT extended with {
       Port lowerPort  = {
        Property protocol = MSG;
```

```
     Property synchronous = false;
    }
    Port upperPort  = {
     Property protocol = MSG;
     Property synchronous = false;
    }
    Property handlesAsynchRequests = true;
    Property layerLevel = 1;
    Property multiThreaded = true;
    Representation Full_Parsing_Rep = {
     System Full_Parsing_Rep : PipesAndFiltersFam = new
            PipesAndFiltersFam extended with {
      Component BinaryFilter0 : BinaryFilter = new BinaryFilter
                extended with {
       Property function = "Auto-generated from grammar file";
      }
      Component Component0  = {
       Port Port0  = {}
       Port Port1  = {}
      }
      Component Component1  = {
       Port Port0  = {}
       Port Port1  = {}
      }
      Component BinaryFilter1 : BinaryFilter = new BinaryFilter
                extended with {}
      Component BinaryFilter2 : BinaryFilter = new BinaryFilter
                extended with {}
      Component BinaryFilter3 : BinaryFilter = new BinaryFilter
                extended with {}
      Connector Pipe0 : Pipe = new Pipe extended with {}
      Connector Pipe1 : Pipe = new Pipe extended with {}
      Connector Pipe2 : Pipe = new Pipe extended with {}
      Connector Pipe3 : Pipe = new Pipe extended with {}
      Connector Pipe4 : Pipe = new Pipe extended with {}
      Connector Pipe5 : Pipe = new Pipe extended with {}
     Attachment Component0.Port0 to Pipe0.sink;
     Attachment BinaryFilter0.output to Pipe1.source;
     Attachment Component1.Port0 to Pipe1.sink;
     Attachment BinaryFilter1.input to Pipe2.sink;
     Attachment Component0.Port1 to Pipe2.source;
     Attachment BinaryFilter1.input to Pipe3.sink;
     Attachment Component1.Port1 to Pipe3.source;
     Attachment BinaryFilter1.output to Pipe4.source;
     Attachment BinaryFilter3.input to Pipe5.sink;
     Attachment BinaryFilter2.output to Pipe5.source;
     Attachment BinaryFilter0.output to Pipe0.source;
     Attachment BinaryFilter2.input to Pipe4.sink;
    }
   Bindings {
   Full_Parsing.upperPort to BinaryFilter0.input;
   Full_Parsing.lowerPort to BinaryFilter3.output;
   }
  }
 }
Component Type_Checking : layerT = new layerT extended with {
 Port lowerPort  = {
```

```
  Property protocol = MSG;
  Property synchronous = false;
 }
 Port upperPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Property handlesAsynchRequests = true;
 Property layerLevel = 2;
 Property multiThreaded = true;
}
Component External_Spec_Merge : layerT = new layerT extended with {
 Port lowerPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Port upperPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Property handlesAsynchRequests = true;
 Property layerLevel = 2;
 Property multiThreaded = true;
}
Component Flow_Analysis : layerT = new layerT extended with {
 Port lowerPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Port upperPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Property handlesAsynchRequests = true;
 Property layerLevel = 4;
 Property multiThreaded = true;
}
Component Extended_Static_Checking : layerT = new layerT extended
          with {
 Port lowerPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Port upperPort  = {
  Property protocol = MSG;
  Property synchronous = false;
 }
 Property handlesAsynchRequests = true;
 Property layerLevel = 5;
 Property multiThreaded = true;
}
Component Code_Generation : layerT = new layerT extended with {
 Port lowerPort  = {
 Property protocol = MSG;
 Property synchronous = false;
}
Port upperPort  = {
```

```
   Property protocol = MSG;
   Property synchronous = false;
  }
  Property handlesAsynchRequests = true;
  Property layerLevel = 6;
  Property multiThreaded = true;
 }
Connector asynchInterLayerRequestT0 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {
 Role requestSender  = {
  Property protocol = MSG;
 }
}
Connector asynchInterLayerRequestT1 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {
 Role requestReceiver  = {
  Property protocol = MSG;
 }
}
Connector asynchInterLayerRequestT2 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {
 Role requestSender  = {
  Property protocol = MSG;
 }
 Role requestReceiver  = {
  Property protocol = MSG;
 }
}
Connector asynchInterLayerRequestT3 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {
 Role requestSender  = {
  Property protocol = MSG;
 }
}
Connector asynchInterLayerRequestT4 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {
 Role requestSender  = {
  Property protocol = MSG;
 }
}
Connector asynchInterLayerRequestT5 : asynchInterLayerRequestT = new
          asynchInterLayerRequestT extended with {}
Attachment Extended_Static_Checking.lowerPort to
           asynchInterLayerRequestT0.requestReceiver;
Attachment Code_Generation.upperPort to
           asynchInterLayerRequestT0.requestSender;
Attachment Extended_Static_Checking.upperPort to
           asynchInterLayerRequestT1.requestSender;
Attachment Flow_Analysis.upperPort to
           asynchInterLayerRequestT2.requestSender;
Attachment External_Spec_Merge.lowerPort to
           asynchInterLayerRequestT2.requestReceiver;
Attachment External_Spec_Merge.upperPort to
           asynchInterLayerRequestT3.requestSender;
Attachment Type_Checking.lowerPort to
           asynchInterLayerRequestT3.requestReceiver;
```

```
  Attachment Full_Parsing.lowerPort to
             asynchInterLayerRequestT4.requestReceiver;
  Attachment Type_Checking.upperPort to
             asynchInterLayerRequestT4.requestSender;
  Attachment Diet_Parsing.lowerPort to
             asynchInterLayerRequestT5.requestReceiver;
  Attachment Full_Parsing.upperPort to
             asynchInterLayerRequestT5.requestSender;
  Attachment Flow_Analysis.lowerPort to
             asynchInterLayerRequestT1.requestReceiver;
  }
 Bindings {
  Compiler.input to Diet_Parsing.upperPort;
  Compiler.output to Code_Generation.lowerPort;
  }
 }
}
Component AST : BinaryFilter = new BinaryFilter extended with {}
Component JDOM : BinaryFilter = new BinaryFilter extended with {}
Connector Pipe0 : Pipe = new Pipe extended with {
 Property bufferSize = 5;
}
Connector Pipe1 : Pipe = new Pipe extended with {
 Property bufferSize = 5;
}
Attachment Compiler.output to Pipe0.source;
Attachment AST.input to Pipe0.sink;
Attachment JDOM.input to Pipe1.sink;
Attachment AST.output to Pipe1.source;
}
Bindings {
 Core.upperPort to Compiler.input;
 Core.lowerPort to JDOM.output;
 }
}
}
Component Debug : layerT = new layerT extended with {
  Port upperPort  = {
    Property protocol = MSG;
    Property synchronous = false;
    }
   Port lowerPort  = {
    Property protocol = MSG;
    Property synchronous = false;
   }
   Property handlesAsynchRequests = true;
   Property layerLevel = 2;
   Property multiThreaded = true;
   }
  Connector asynchInterLayerRequestT0 : asynchInterLayerRequestT = new
            asynchInterLayerRequestT extended with {}
  Connector asynchInterLayerRequestT1 : asynchInterLayerRequestT = new
            asynchInterLayerRequestT extended with {
    Role requestSender  = {
     Property protocol = MSG;
    }
   }
```

```
   Attachment UI.lowerPort to asynchInterLayerRequestT0.requestSender;
   Attachment Core.upperPort to
              asynchInterLayerRequestT0.requestReceiver;
   Attachment Core.lowerPort to
              asynchInterLayerRequestT1.requestReceiver;
   Attachment Debug.upperPort to
              asynchInterLayerRequestT1.requestSender;
  }
  Bindings {
   JDT.p0 to UI.upperPort;
   JDT.provideT0 to Debug.lowerPort;
  }
 }
}
Component Eclipse_Platform : TierNodeT = new TierNodeT extended with {
 Port provideT0 : provideT = new provideT extended with {}
 Port provideT1 : provideT = new provideT extended with {}
 Property allowShareHost = true;
 Property host = "RunTime Platform";
 }
Component User_Input  = {}
Connector UserConnector : LocalConnT = new LocalConnT extended with {
 Property blocking = true;
}
Connector PlugIn_Connector : LocalConnT = new LocalConnT extended with{
 Property blocking = false;
}
Connector User_Interaction_Connector : LocalConnT = new LocalConnT
          extended with {
 Property blocking = false;
}
Attachment User_Input.p to UserConnector.callee;
Attachment Eclipse_Platform.provideT0 to UserConnector.caller;
Attachment Eclipse_Platform.p to PlugIn_Connector.callee;
Attachment JDT.p0 to PlugIn_Connector.caller;
Attachment Eclipse_Platform.provideT1 to
           User_Interaction_Connector.caller;
Attachment JDT.provideT0 to User_Interaction_Connector.callee;
}
```

PART 2

Description of the JML Architecture:

The JML Architecture is a heterogeneous architecture combining Layered, Tiered and PipesAndFilters family of architectures. It is impossible to evaluate this system from the architectural point of view. Thus for clarity and simplification let me discuss a part of the system.

The part that I would discuss is the innermost layer of this architecture, namely Full_parsing. This component is itself composed of a PipesAndFilter architecture. Thus this innermost layer or the Core of the architecture belongs to the Pipes and Filter style.

Advantages of this style

- It allows the designer to understand the input/ output behavior of the system.
- The system designed is reusable.
- It can be easily maintained and enhanced.
- It allows analysis of the system and can identify deadlocks.
- It allows concurrent usage.

Disadvantages of this style

- It leads to batch organization of processing.
- It is difficult to maintain correspondence between two separated but related streams.
- It gives low performance and high complexity.

PART 3

Architectural Style

An important general capability for the description of architectures is the ability to define styles – or families – of systems. The following code illustrates the definition of the 'JML' family, together with a sample system declaration using the family.

JML Family

The family defines four component types, three component types and the implicit properties that are extended from PipesAndFilters, Tiered and Layered Families.

```
import $AS_GLOBAL_PATH/families/PipesAndFiltersFam.acme;
import $AS_GLOBAL_PATH/families/TieredFam.acme;
import $AS_GLOBAL_PATH/families/LayeredFam.acme;

Family JML extends PipesAndFiltersFam, TieredFam, LayeredFam with {
 Component Type TierNodeT0 extends TierNodeT with {
  Port provideT0 : provideT = new provideT extended with {}
  Port remoteProvideT0 : remoteProvideT = new remoteProvideT extended
                                          with {}
 }
 Component Type BinaryFilter0 extends BinaryFilter with {}
  Connector Type LocalConnT0 extends LocalConnT with {}
  Connector Type Pipe0 extends Pipe with {}
 Component Type Component0 = {
  Port Port0  = {}
  Port Port1  = {}
 }
```

```
 Component Type layerT0 extends layerT with {}
 Connector Type asynchInterLayerRequestT0 extends
               asynchInterLayerRequestT with {}
}
```

The System: StyleArchitecture

The JML architectural system is defined using the JML family. This system is defined as an instance of the family.

```
import families/JML.acme;
System StyleArchitecture : JML = {
 Component Eclipse : TierNodeT0 = {}
 Component JDT : TierNodeT0 = {
  Representation JDT_Rep = {
   System JDT_Rep : JML = new JML extended with {
    Component UI : LayerT0 = {}
     Component Core : LayerT0 = {
      Representation Core_Rep = {
       System Core_Rep : JML = new JML extended with {
        Component Compiler : BinaryFilterT0 = {
         Representation Compiler_Rep = {
          System Compiler_Rep : JML = new JML extended with {
           Component L1 : LayerT0 = {}
           Component Full_Parsing : LayerT0 = {
            Representation Full_Parsing_Rep = {
             System Full_Parsing_Rep : JML = new JML extended with {
              Component JikesPG : BinaryFilterT0 = {}
              Component Parser_Code : ComponentT0 = {}
              Component Parser_Table : ComponentT0 = {}
              Component Parser : BinaryFilterT0 = {}
              Component Scanner : BinaryFilterT0 = {}
              Connector p1 : PipeT0 = {}
              Connector p2 : PipeT0 = {}
              Connector p3 : PipeT0 = {}
              Connector p4 : PipeT0 = {}
              Connector p5 : PipeT0 = {}
              Attachment JikesPG.output to p1.source;
              Attachment Parser_Code.input to p1.sink;
              Attachment JikesPG.output to p2.source;
              Attachment Parser_Table.input to p2.sink;
              Attachment Parser_Code.output to p3.source;
              Attachment Parser.input to p3.sink;
              Attachment Parser_Table.output to p4.source;
              Attachment Parser.input to p4.sink;
              Attachment Parser.output to p5.source;
              Attachment Scanner.input to p5.sink;
             }
             Bindings {
              Full_Parsing.upperPort to JikesPG.input;
              Scanner.output to Full_Parsing.lowerPort;
             }
            }
           }
```

```
      Component L3 : LayerT0 = {
      Component L4 : LayerT0 = {}
      Component L5 : LayerT0 = {}
      Component L6 : LayerT0 = {}
      Component L7 : LayerT0 = {}
      Connector c1 : asynchInterLayerRequestT0 = {}
      Connector c2 : asynchInterLayerRequestT0 = {}
      Connector c3 : asynchInterLayerRequestT0 = {}
      Connector c4 : asynchInterLayerRequestT0 = {}
      Connector c5 : asynchInterLayerRequestT0 = {}
      Connector c6 : asynchInterLayerRequestT0 = {}
      Attachment L1.lowerPort to c1.RequestReceiver;
      Attachment Full_Parsing.upperPort to c1.RequestSender;
      Attachment Full_Parsing.lowerPort to c2.RequestReceiver;
      Attachment L3.upperPort to c2.RequestSender;
      Attachment L3.lowerPort to c3.RequestReceiver;
      Attachment L4.upperPort to c3.RequestSender;
      Attachment L4.lowerPort to c4.RequestReceiver;
      Attachment L5.upperPort to c4.RequestSender;
      Attachment L5.lowerPort to c5.RequestReceiver;
      Attachment L6.upperPort to c5.RequestSender;
      Attachment L6.lowerPort to c6.RequestReceiver;
      Attachment L7.upperPort to c6.RequestSender;
      }
     Bindings {
      Compiler.input to L1.upperPort;
      L7.lowerPort to Compiler.output;
     }
    }
   }
   Component AST : BinaryFilterT0 = {}
   Component JDom : BinaryFilterT0 = {}
   Connector firstPipe : Pipe0 = {}
   Connector secondPipe : Pipe1 = {}
   Attachment Compiler.output to firstPipe.source;
   Attachment AST.input to firstPipe.sink;
   Attachment AST.output to secondPipe.source;
   Attachment JDom.input to secondPipe.sink;
   }
  Bindings {
   Core.upperPort to Compiler.input;
   JDom.output to Core.lowerPort;
  }
 }
}
Component Debug : LayerT0 = {}
Connector asyn1 : asynchInterLayerRequestT0 = {}
Connector asyn2 : asynchInterLayerRequestT0 = {}
Attachment UI.lowerPort to asyn1.RequestReceiver;
Attachment Core.upperPort to asyn1.RequestSender;
Attachment Core.lowerPort to asyn2.RequestReceiver;
Attachment Debug.upperPort to asyn2.RequestSender;
}


Bindings {
 JDT.remoteProvideT0 to UI.upperPort;
```

```
    Debug.lowerPort to JDT.ProvideT0;
   }
  }
 }
 Component UserInterface : TierNodeT0 = {}
 Connector firstConn : LocalConnT0 = {}
 Connector secConn : LocalConnT0 = {}
 Connector thirdConn : LocalConnT0 = {}
 Attachment JDT.ProvideT0 to firstConn.callee;
 Attachment Eclipse.remoteProvideT0 to firstConn.caller;
 Attachment Eclipse.ProvideT0 to secConn.callee;
 Attachment JDT.remoteProvideT0 to secConn.caller;
 Attachment UserInterface.ProvideT0 to thirdConn.caller;
 Attachment Eclipse.remoteProvideT0 to thirdConn.callee;
}
```

Reference:

1. Gary T. Leavens, Albert L. Baker, and Clyde Ruby. JML: A Notation for Detailed Design [postscript] [PDF]. In Haim Kilov, Bernhard Rumpe, and Ian Simmonds (editors), *Behavioral Specifications of Businesses and Systems*, chapter 12, pages 175-188. Copyright Kluwer, 1999.
2. http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html
3. Patrice Chalin. Perry R. James, and George Karabotsos. The Architecture of JML4, a Proposed Integrated Verification Environment for JML. Dependable Software Research Group, Concordia University, ENCS-CSE-TR 2007-006. May, 2007.
4. Acme: An Architecture Description Interchange Language, David Garlan, Robert T. Monroe, David Wile, Proceedings of CASCON '97, November 1997.
5. http://www.cs.cmu.edu/~acme/docs/language_overview.html
6. http://www.cs.cmu.edu/~acme/AcmeStudio/tutorials.html