VMTS

a general purpose Meta-Modelling and Transformation Environment

Xiaoxi Dong
March 20, 2008
COMP763 Reading Presentation

Summary

- Introduction to VMTS
  - History
  - Features
- Design of VMTS
  - VMTS is a n-layer storage system
  - VMTS is a transformation system
  - VCFL: Visual Control Flow Language
- Implementation
- Case Study

Introduction to VMTS

- VMTS
- Visual Modeling and Transformation System
  - n-layer meta-modeling environment
  - model transformation functionalities

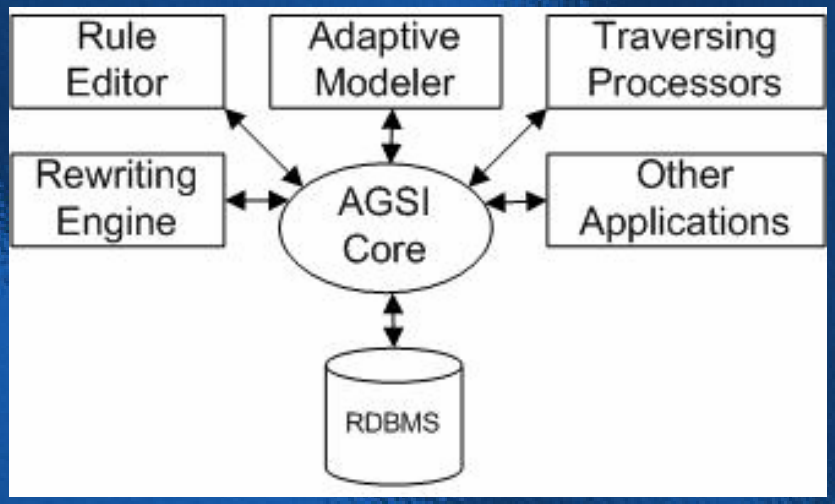
- by Tiham´er Levendovszky, L´aszl´o Lengyel and Gergely Mezei from Budapest University of Technology and Economics in 2004
- Current version status: 0.95

Introduction to VMTS

- Features
  - Automated support for preserving, guanranteeing and validating constraints.
  - Pattern language: specified by UML class diagram, supports multiplicity and inheritance
  - User Interface: Plugins based on the VMTS Presentation Framework
  - Support software evolution
  - Free

Design of VMTS

- VMTS is a n-layer storage system
- VMTS is a transformation system
- VCFL: Visual Control Flow Language
- Realise QVT with VMTS



The VMTS architecture

(i) Attributed Graph Architecture Supporting Inheritance (AGSI) - the core part

(ii) Database,

(iii) Server side - AGSI Core, Rewriting engine,

(iv) Client side - Rule Editor, Modelers and other applications.

Design
n-layered stroage system

- Metamodeling is based on the instrantiation relationship (UML class diagram and UML object diagram)
- VMTS uses a simplifed UML class diagram as a metamodel language
- Instantiate UML class diagram in VMTS involves 5 layers:
  - UML object diagram, UML class diagram, the meramodel of UML class diagram
  - Read-only meta-metamodel
  - Labled directed graph in the internal structure
- Model attributes are stoed in the labels

Design
- Transformation System

- Graph Rewriting: LHS and RHS
  - LHS must be an instantiation insteat of an isomorphic subgraph is natural representation of multiplicities
  - Causality: relationship between LHS and RHS of a tranformation rule
    - Internal causality ->operations of LHS and RHS elements
    - External cayusality ->parameter passing between rules

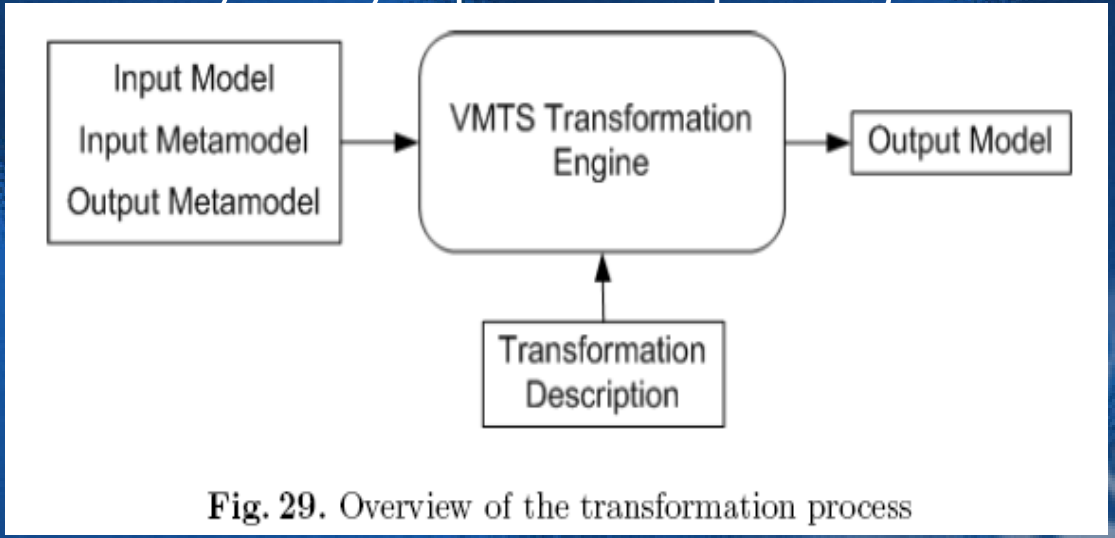


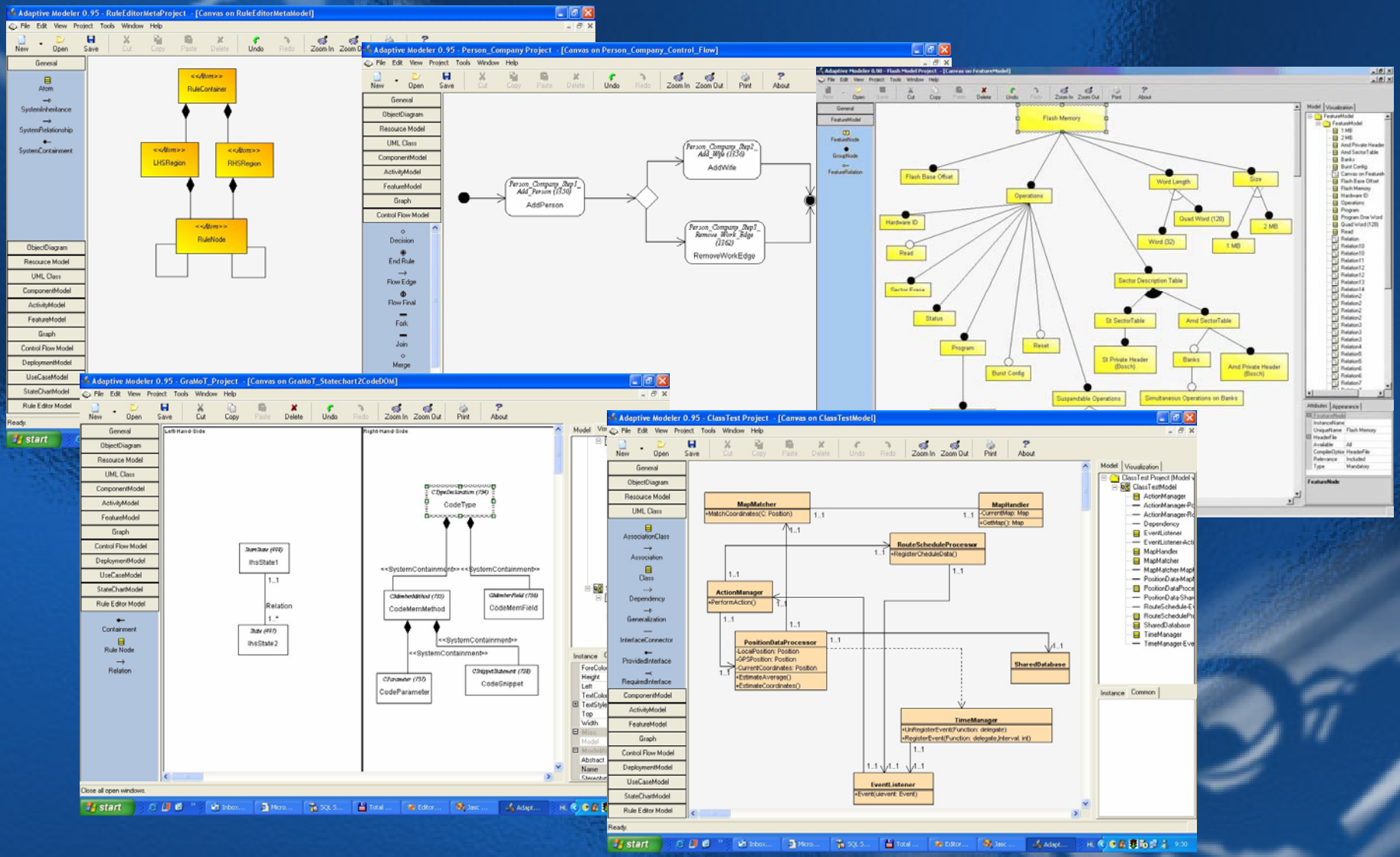
Fig. 29. Overview of the transformation process

Design
VCFL & OCL Constraints

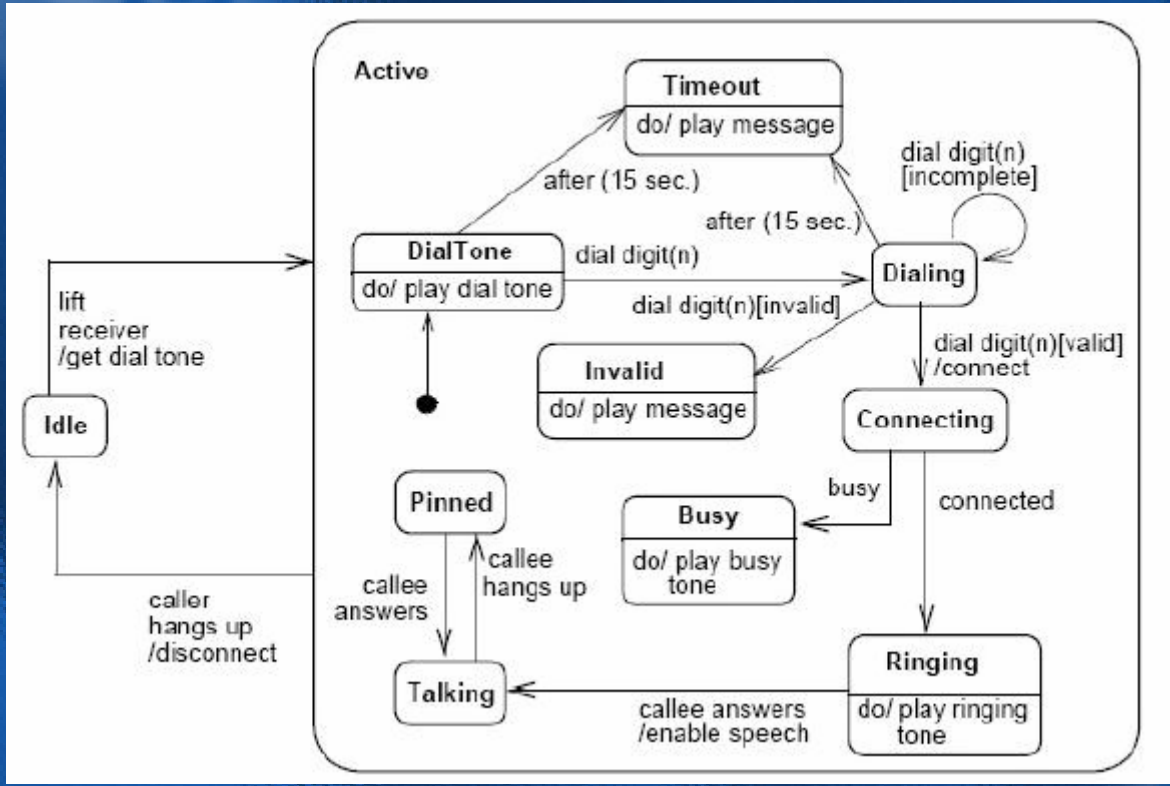
- VMTS Contral Flow Language
  - A stereotyped UML activity diagram
  - The activity is provided with transformation rule
  - OCL constraints are used to decide which branch to pass to contrral
  - Allow external causality
  - OCL Constraints

Implementation

- In AGSI the model attributes are stored in XML attributes
- Rule execution is specified by VCFL
- Abstract oriented techniques are used to eliminate the constrantss that cross cut the transformation
- Constraints given in OCL are compiled into a .NET assembly
- Relational Database management system uses SQL Server
- Generate code from UML class diagram from CodeDOM code representation, from which .NET generate code automatically


Adaptive Modeler 0.95 - RuleEditorMetaProject - [Canvas on RuleEditorMetaModel]
RuleContainer
LHSRegion
RHSRegion
RuleNode
Adaptive Modeler 0.95 - Person_Company Project - [Canvas on Person_Company_Control_Flow]
Person_Company_Step1_Add_Person (1350)
AddPerson
Person_Company_Step2_Add_Wife (1334)
AddWife
Person_Company_Step3_Remove_Work_Edge (1362)
RemoveWorkEdge
Flash Memory
Flash Base Offset
Operations
Word Length
Size
Hardware ID
Quad Word (128)
2 MB
Read
Word (32)
1 MB
Sector Description Table
Status
St SectorTable
Amd SectorTable
Program
Reset
Burst Config
St Private Header (Bosch)
Banks
Amd Private Header (Bosch)
Suspendable Operations
Simultaneous Operations on Banks
Adaptive Modeler 0.95 - GraMoT_Project - [Canvas on GraMoT_Statechart2CodeDOM]
Left Hand Side
Right Hand Side
IhsState1
Relation
IhsState2
CodeType
CodeMemMethod
CodeMemField
CodeParameter
CodeSnippet
Adaptive Modeler 0.95 - ClassTest Project - [Canvas on ClassTestModel]
MapMatcher
+MatchCoordinates(C: Position)
MapHandler
-CurrentMap: Map
+GetMap(): Map
RouteScheduleProcessor
+RegisterCheduleData()
ActionManager
+PerformAction()
PositionDataProcessor
-LocalPosition: Position
-GPSPosition: Position
-CurrentCoordinates: Position
+EstimateAverage()
+EstimateCoordinates()
SharedDatabase
TimeManager
+UnRegisterEvent(Function: delegate)
+RegisterEvent(Function: delegate,Interval: int)
EventListener
+Event(uievent: Event)

Case Study – state chart



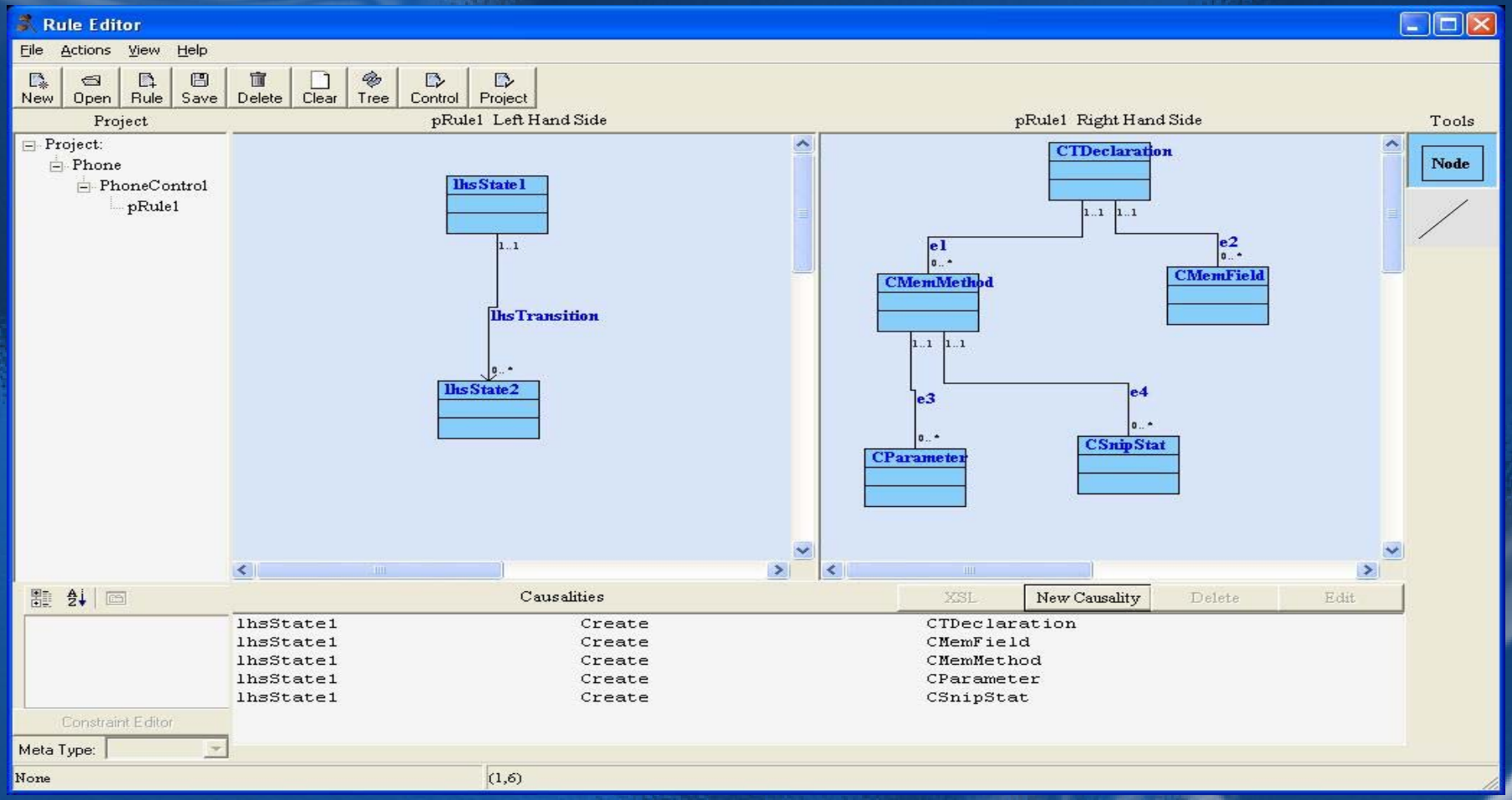
The State Chart Model

Case Study – state chart

```
<State>
        <Name>Connecting</Name>
        <Note></Note>
        <History></History>
        <ParentState>Active</ParentState>
        <InternalTransition>
                <EventName>Q_ENTRY_SIG</EventName>
                <Parameter></Parameter>
                <GuardCondition></GuardCondition>
                <Action></Action>
        </InternalTransition>
        <InternalTransition>
                <EventName>Q_EXIT_SIG</EventName>
                <Parameter></Parameter>
                <GuardCondition></GuardCondition>
                <Action></Action>
        </InternalTransition>
        <InternalTransition>
                <EventName>BUSY</EventName>
                <Parameter></Parameter>
                <GuardCondition></GuardCondition>
                <Action>Q_TRAN(&amp;amp;Phone::Busy)</Action>
        </InternalTransition>
        <InternalTransition>
                <EventName>CONNECTED</EventName>
                <Parameter></Parameter>
                <GuardCondition></GuardCondition>
                <Action>Q_TRAN(&amp;amp;Phone::Ringing)</Action>
        </InternalTransition>
</State>
```

A sample XML for the **Connecting** state:

Case Study – state chart



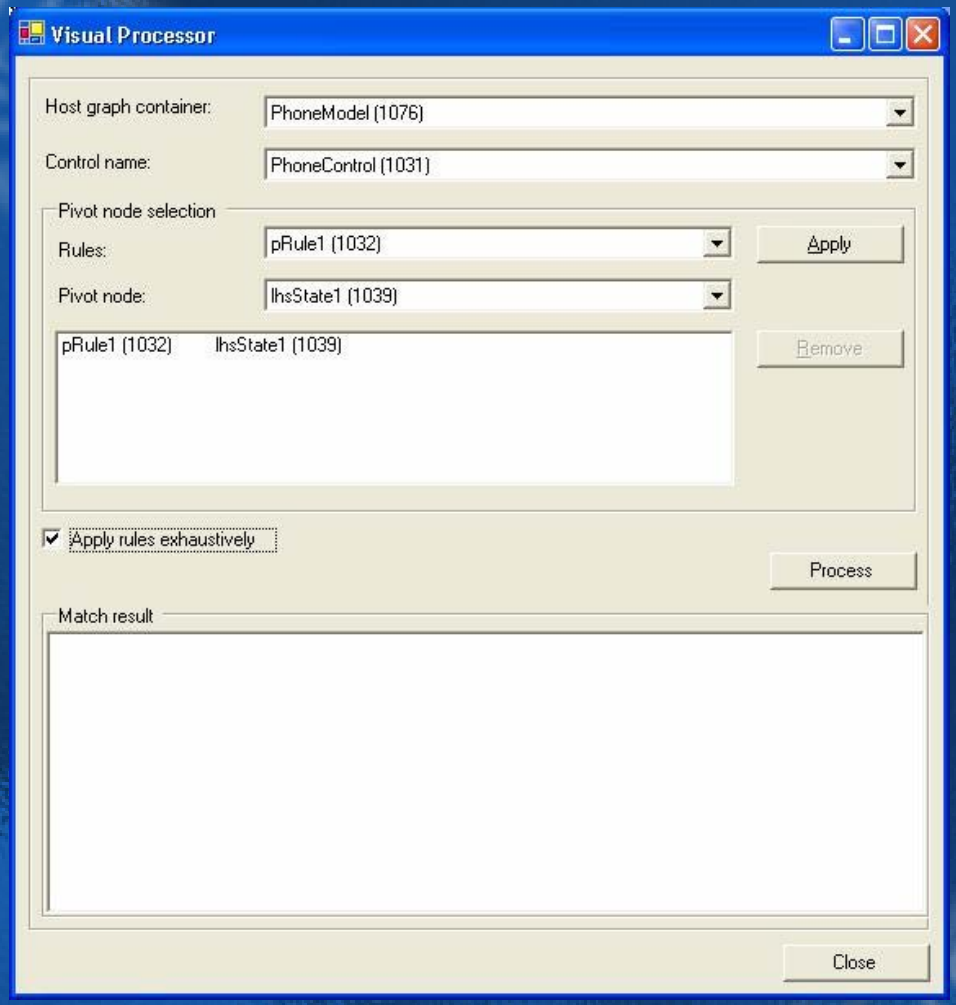
The rewriting rule (and the Rule Editor environment):

Case Study – state chart

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="xml" version="1.0" standalone="yes" indent="yes"/>
 <xsl:template match="/">
  <RewriteResult>
   <CodeMemberMethod>
    <Name><xsl:value-of select="//Name"/></Name>
    <ReturnType>QSTATE</ReturnType>
    <Attributes>private</Attributes>
   </CodeMemberMethod>

   <CodeParameterDeclarationExpression>
    <Name>e</Name>
    <Type>QEvent const *</Type>
   </CodeParameterDeclarationExpression>

   <CodeSnippetStatement>
    <Statement>switch (e->sig) {</Statement>
   </CodeSnippetStatement>

   <xsl:for-each select="//InternalTransition">
    <xsl:call-template name="codeSnippetStatement"/>
   </xsl:for-each>

   <CodeSnippetStatement>
    <Statement>}</Statement>
   </CodeSnippetStatement>

   <CodeSnippetStatement>
    <Statement>return (QSTATE)&amp;amp;QPhone::<xsl:value-of
select="//ParentState"/>;</Statement>
   </CodeSnippetStatement>

  </RewriteResult>
 </xsl:template>

 <xsl:template name="codeSnippetStatement">
  <CodeSnippetStatement>
   <xsl:variable name="action" select="Action"/>
   <Statement>case <xsl:value-of select="EventName"/> : printf("<xsl:value-of
select="../Name"/>-<xsl:value-of select="EventName"/>;");<xsl:value-of
select="Action"/> <xsl:if test="not( contains($action, 'Q_TRAN'))"></xsl:if>
return 0;</Statement>
  </CodeSnippetStatement>
 </xsl:template>

</xsl:stylesheet>
```

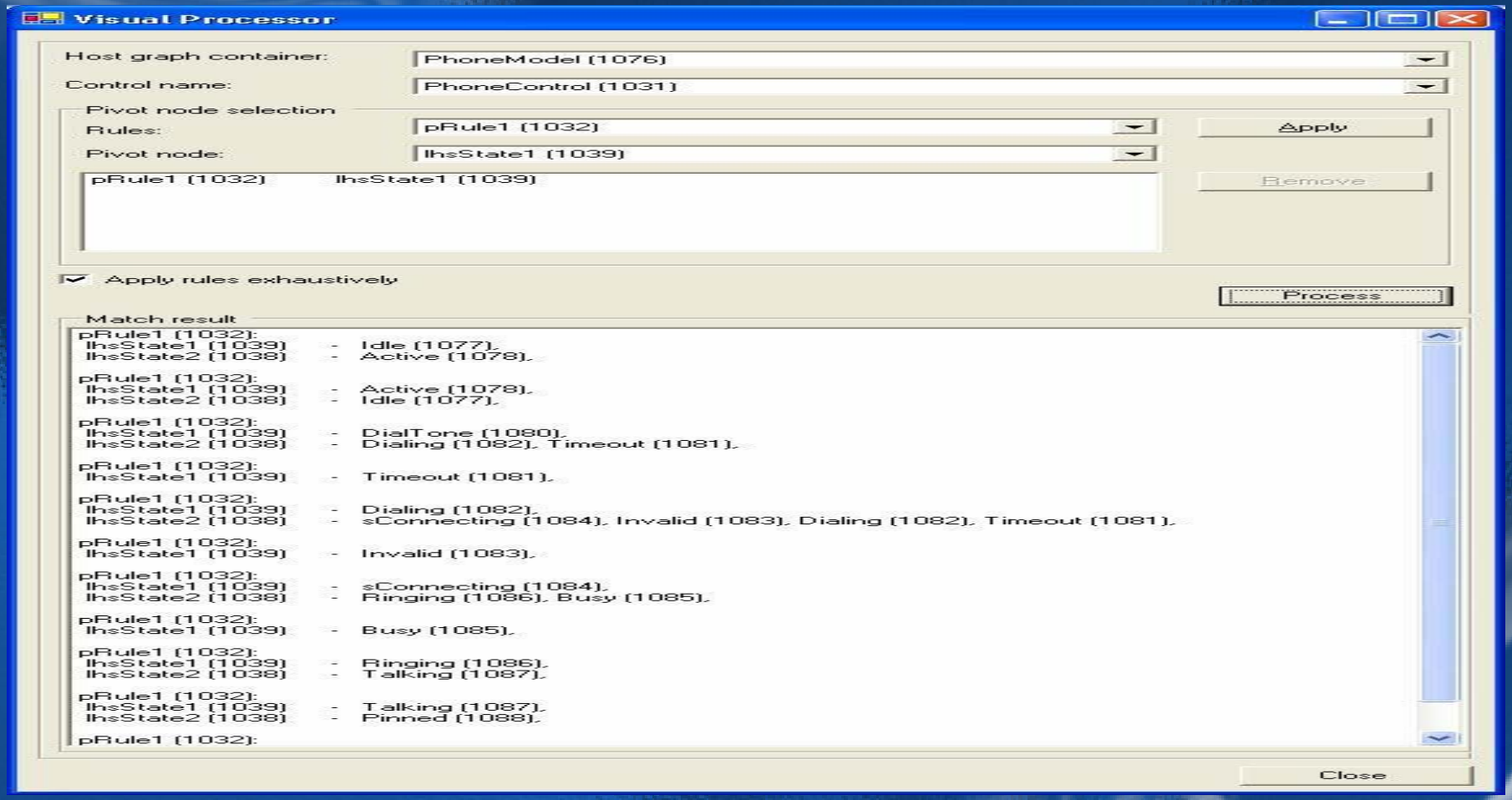
The XSL file which convert the XML files to CodeDOM structure:

Case Study – state chart



Visual Processor

Case Study – state chart



The Visual Processor and matches

Further development

- Included in the next published version
  - Generative programming
  - MDA tranformation

Summary

- Introduction to VMTS
  - History
  - Features
- Design of VMTS
  - VMTS is a n-layer storage system
  - VMTS is a transformation system
  - VCFL: Visual Control Flow Language
- Implementation
- Case Study