NooJ Graphical User Interfaces Modernization

Z. Gotti, S. Mbarki, S. Gotti and N. Laaz

MISC Laboratory,
Faculty of Science, Ibn Tofail University
Kenitra, MOROCCO

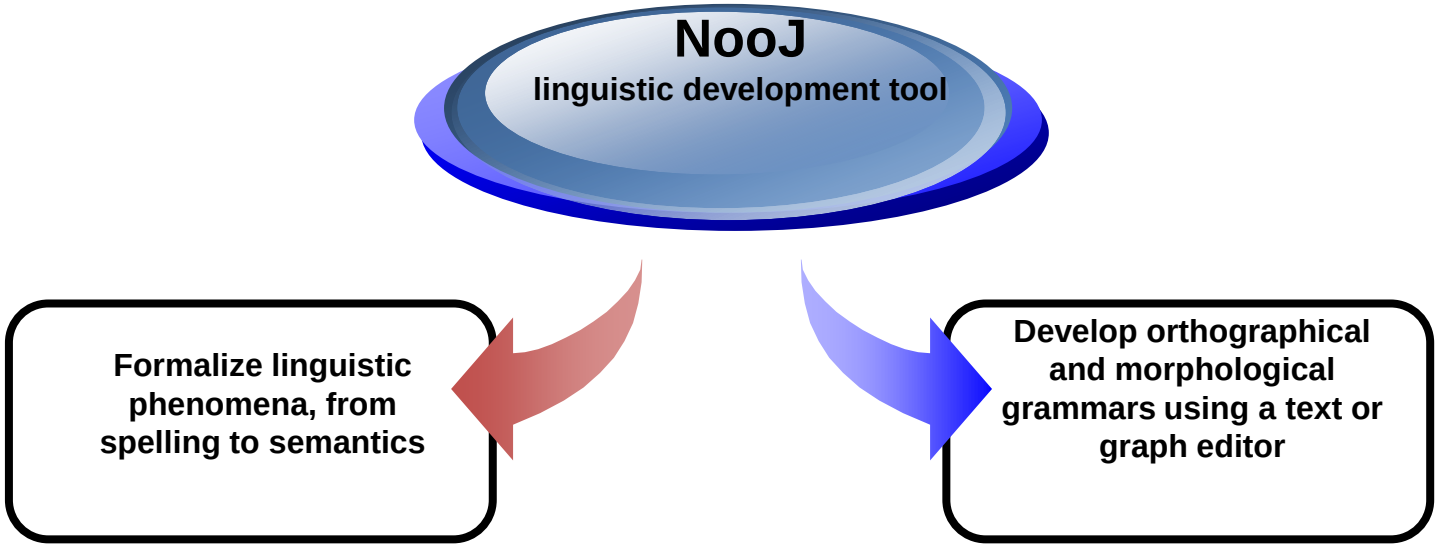
Plan

- Introduction
- Context
- Contribution
- NooJ System
- Implementation
- Conclusion & Futur Works

Introduction

Observation

Solution



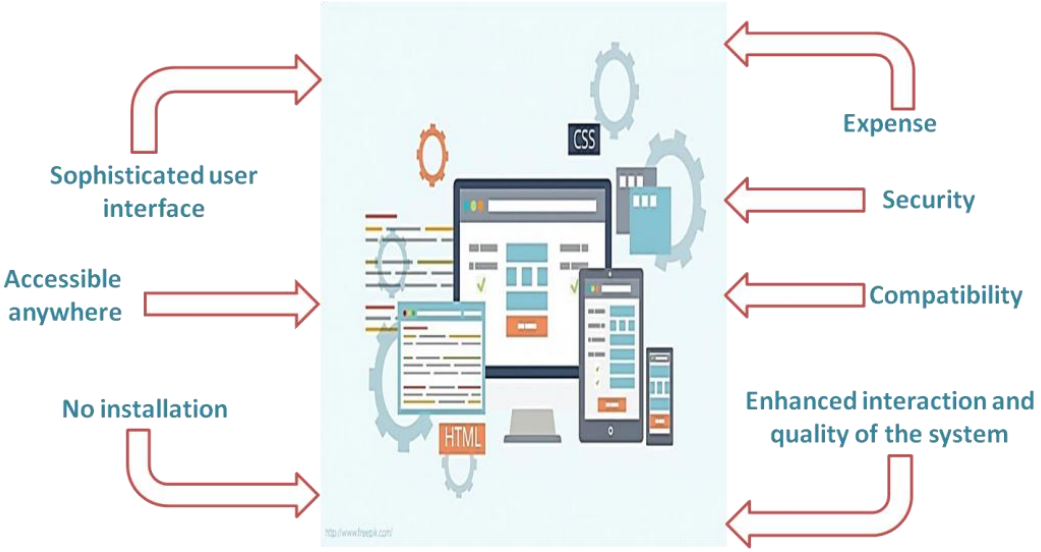
- This kind of systems are critical assets that must be updated continuously to reflect evolving practices.
- Repeated modification has a cumulative effect on system complexity.
- The rapid evolution of technology quickly renders existing technologies obsolete.

Introduction

Observation

Solution

•We opted for a modernization process that transform the old java NooJ system to the web that satisfies new requirements.

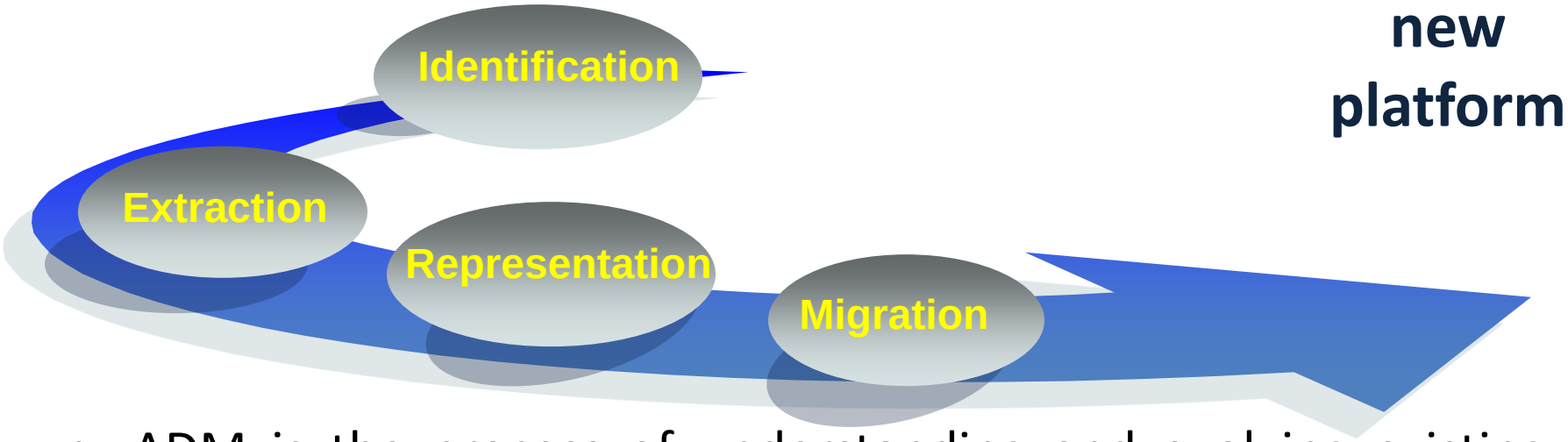


•The approach is based on the Architecture-Driven Modernization process as a best solution for the legacy system's adaptive and perfective maintenance.

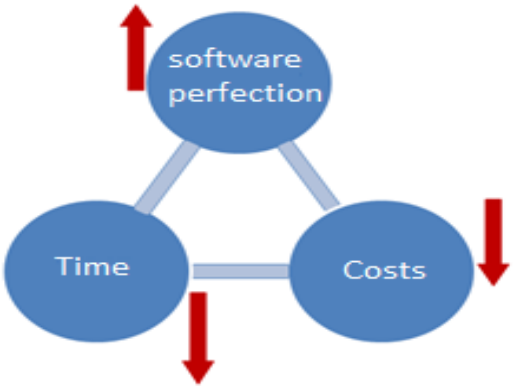
Context

Architecture-Driven Modernization (ADM)

- the ADM approach offers a generic and abstract solution based on models, allowing:



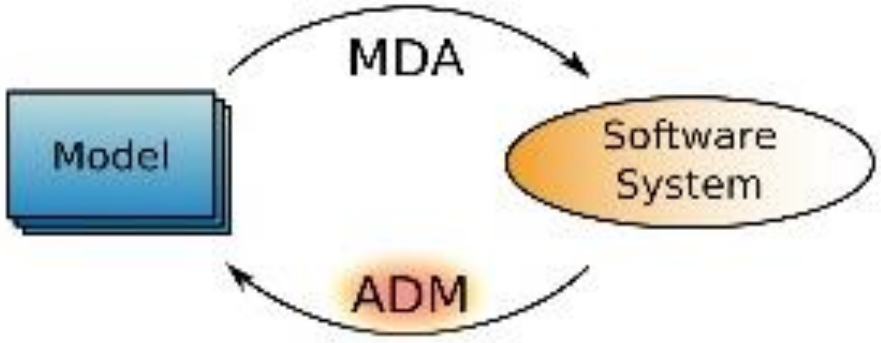
- ADM is the process of understanding and evolving existing software assets.



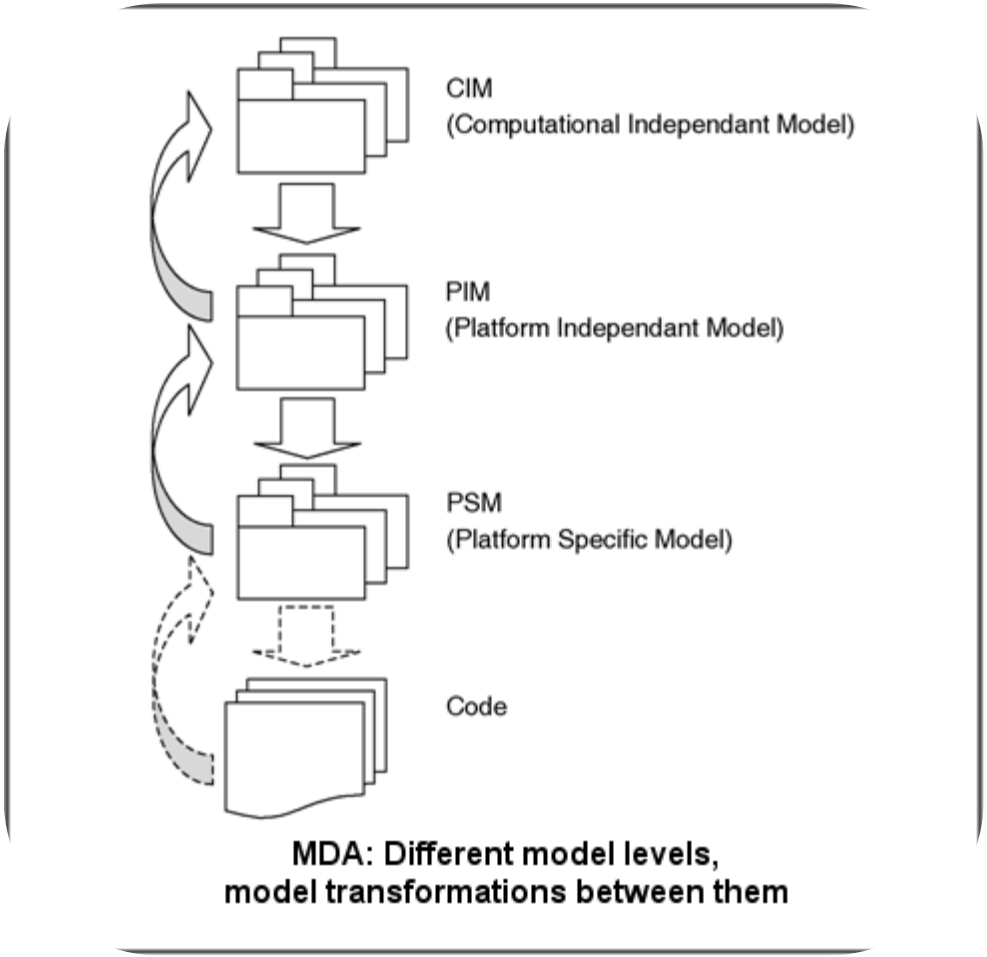
Context

Architecture-Driven Modernization (ADM)

- ADM is an Object Management Group initiative related to building and promoting standards that can be applied to modernize legacy systems.
- It has emerged complementing OMG Model Driven Architecture standard (MDA).
- The outstanding ideas behind MDA are:
  - separating the specification of the system functionality from its implementation on specific platforms.
  - managing the software evolution from abstract models to implementations.



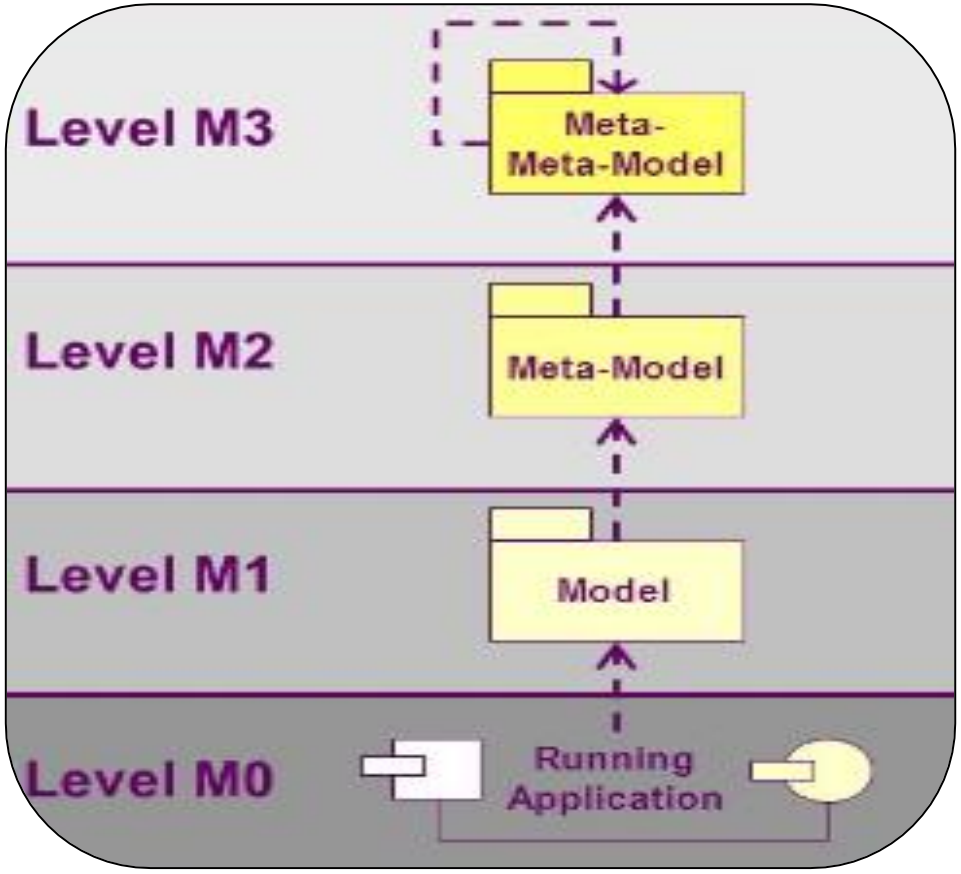
Context
Architecture-Driven Modernization (ADM)



MDA: Different model levels, model transformations between them

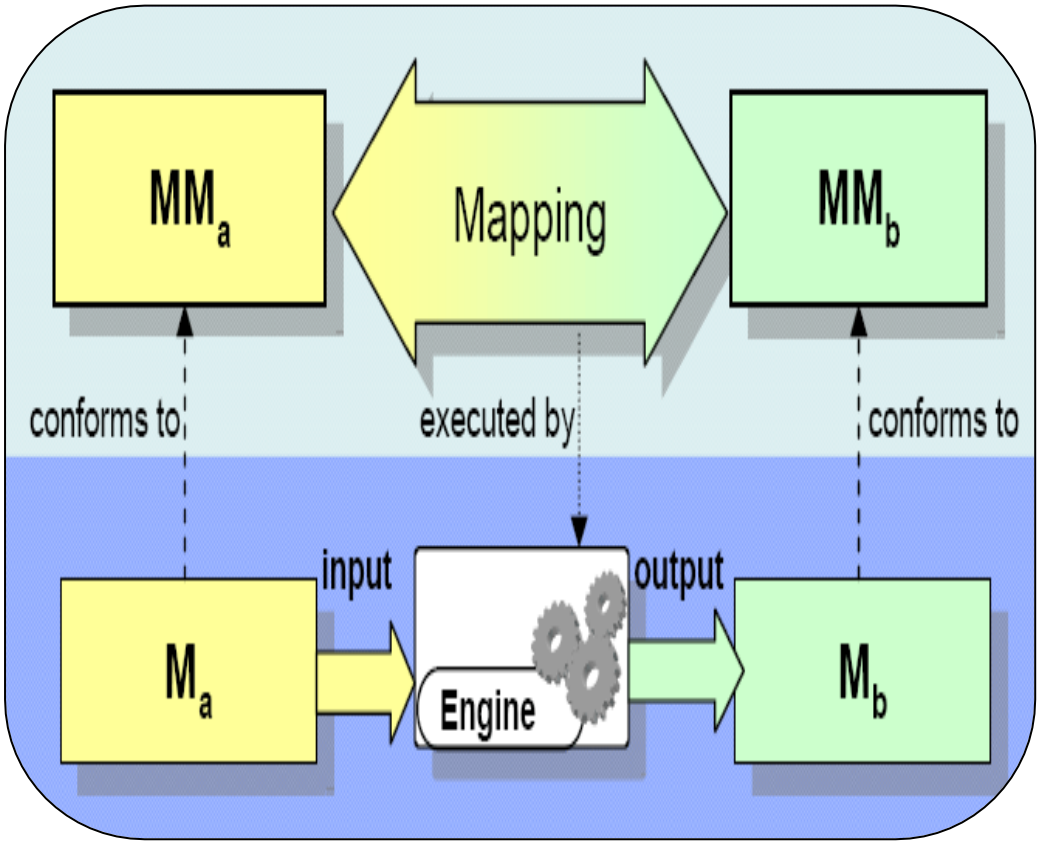
Context

Architecture-Driven Modernization (ADM)



Context

Architecture-Driven Modernization (ADM)



Context

knowledge discovery Meta model(**KDM**) & Abstract syntax tree Meta model(**ASTM**)

The OMG ADM Task Force (ADMTF) is developing a set of standards (metamodels) to facilitate interoperability between modernization tools.

ASTM

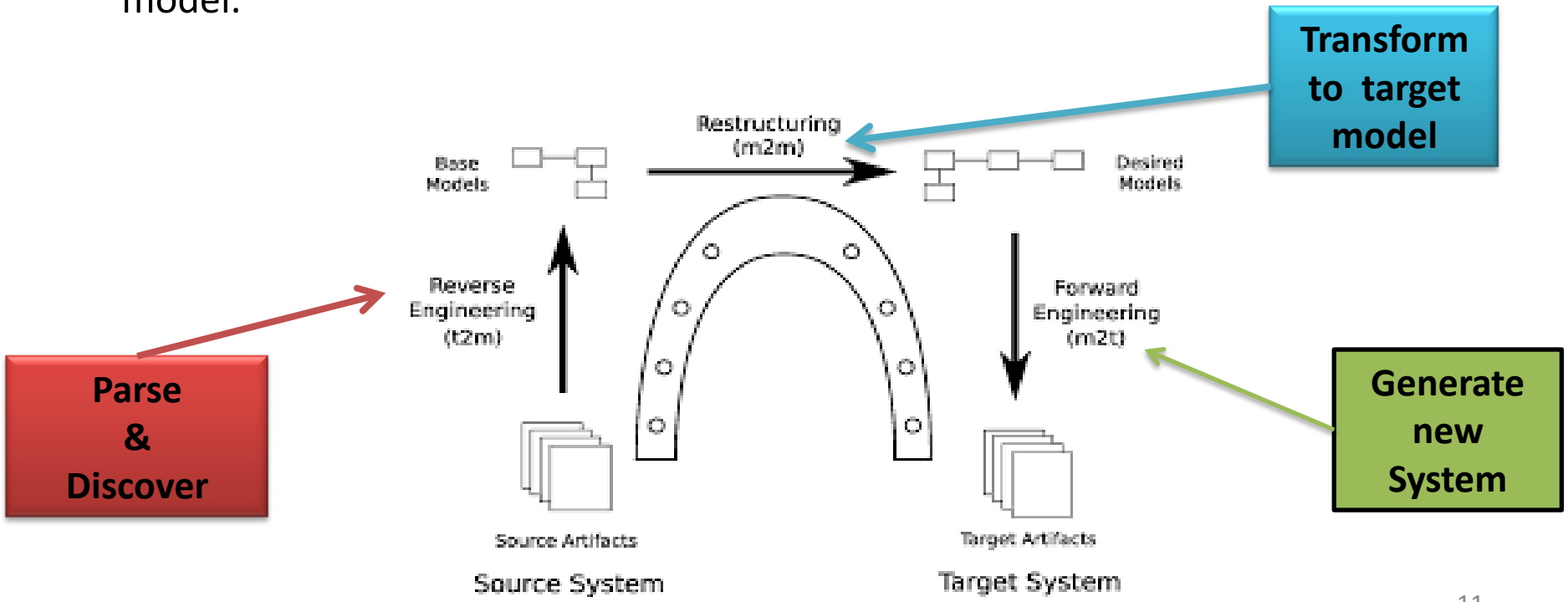
The ASTM standard provides the most granular view of the system architecture.
Predicted for low-level modeling, close to source code.

KDM

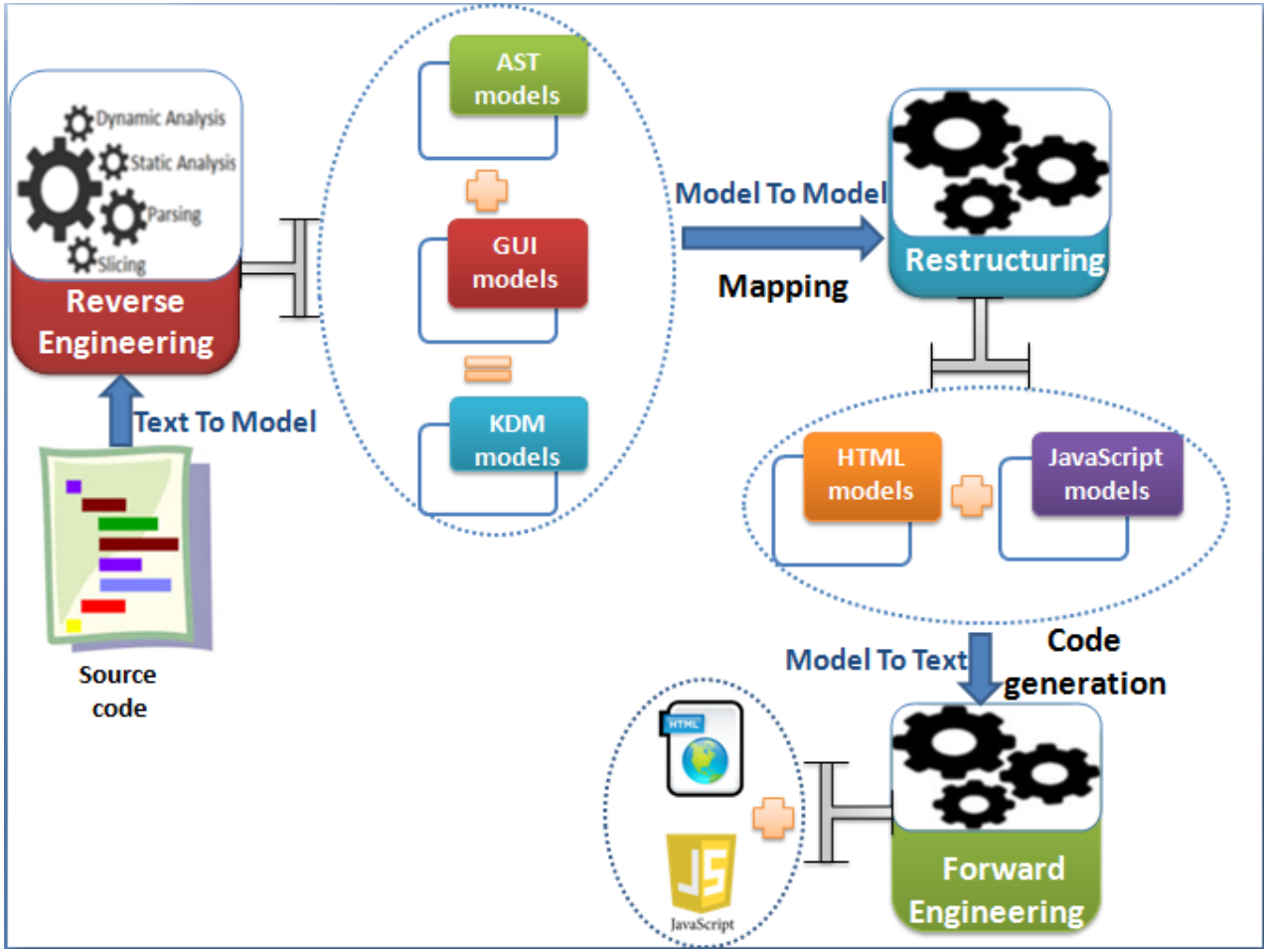
The KDM standard is a common intermediate representation for existing software systems.
It represents all aspects of the existing system architecture.
The KDM can leverage information captured by the ASTM.

Contribution

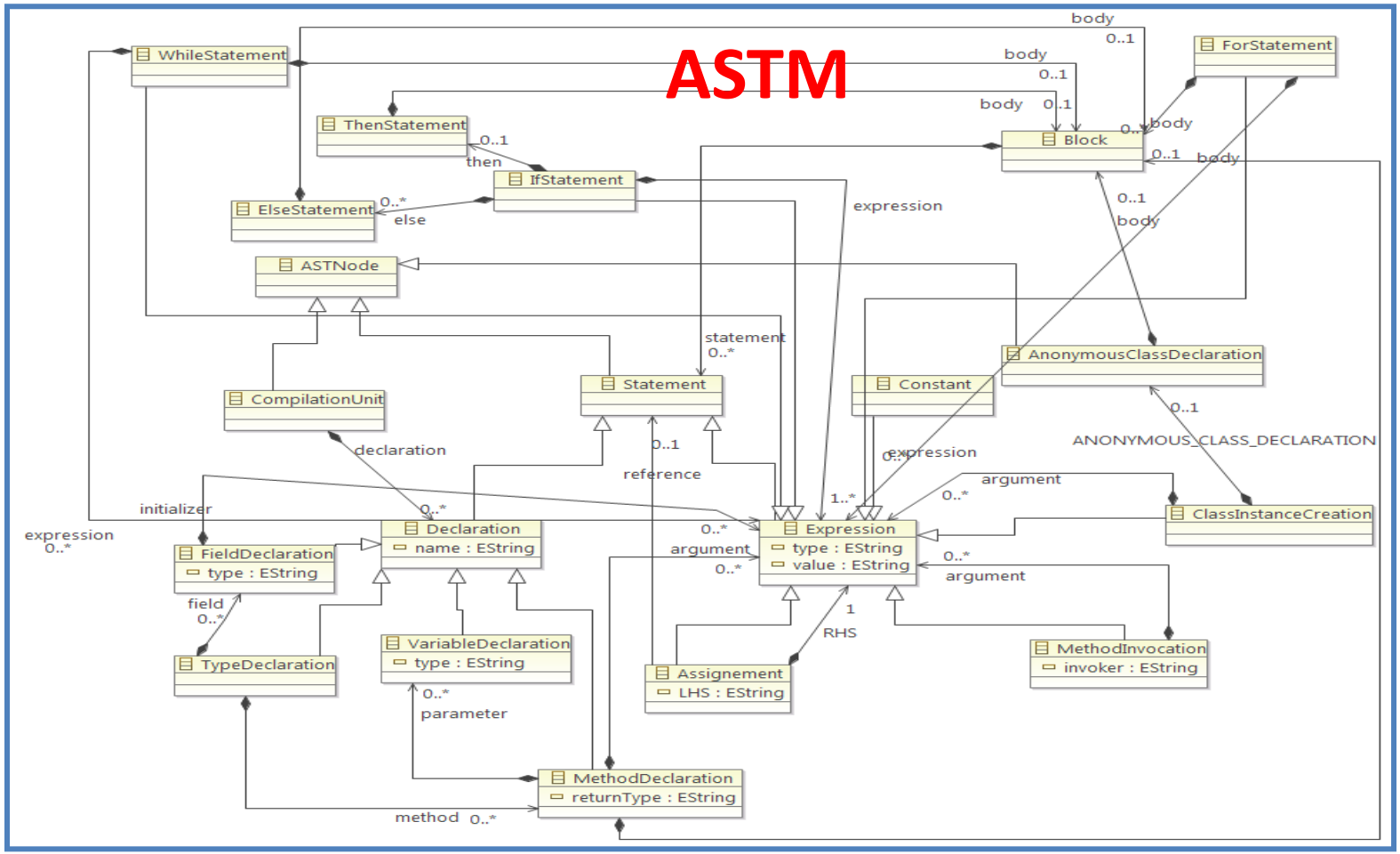
- The migration ADM based process consists of three phases:
  - The reverse engineering of the legacy java nooj system that represents the extraction of information from the source code at a higher level of abstraction;
  - The Restructuring: it is a model to model transformation for constructing the target models;
  - The forward engineering that generates the new web system from the target model.



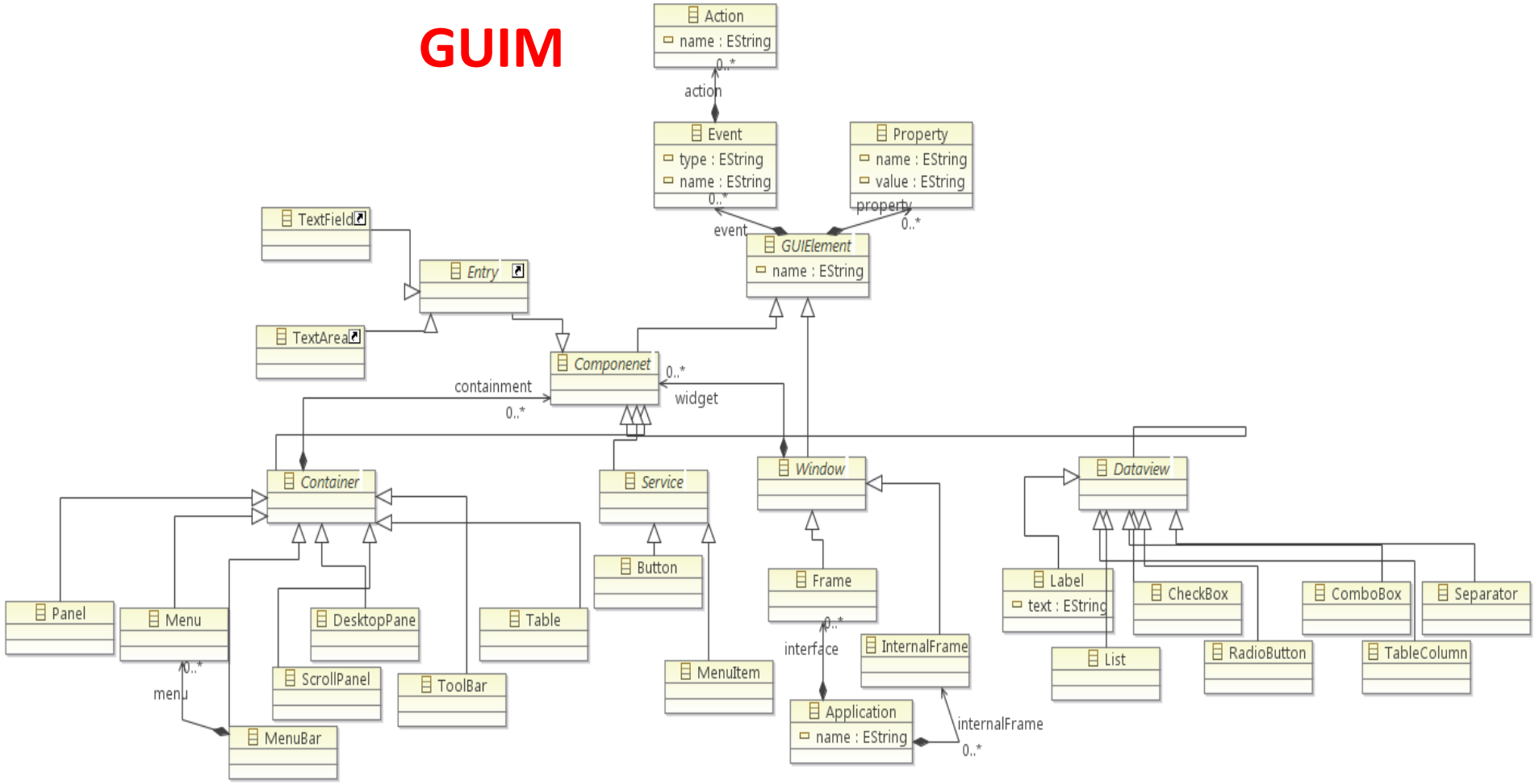
Contribution
Overview of our migration process



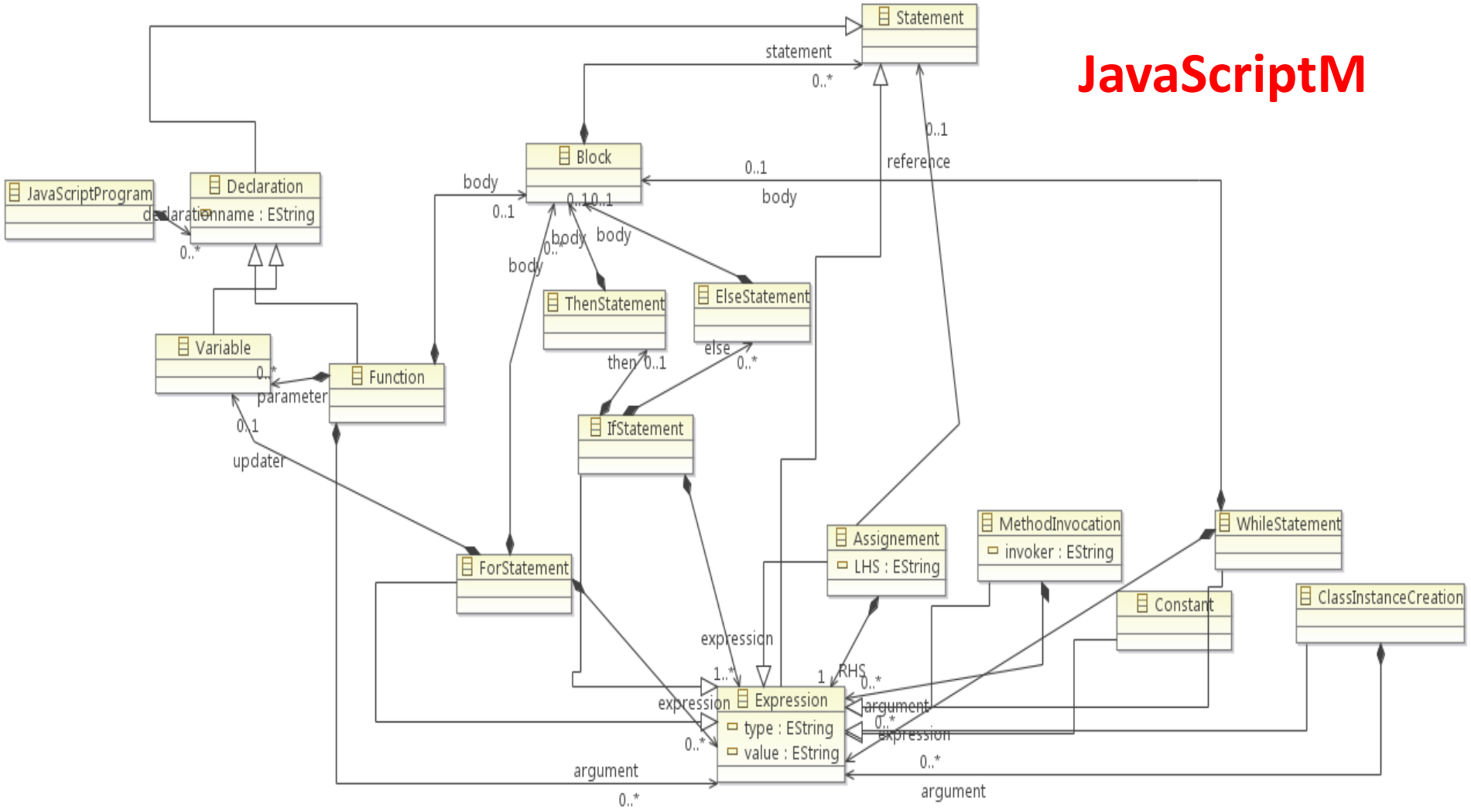
Contribution
Overview of our migration process



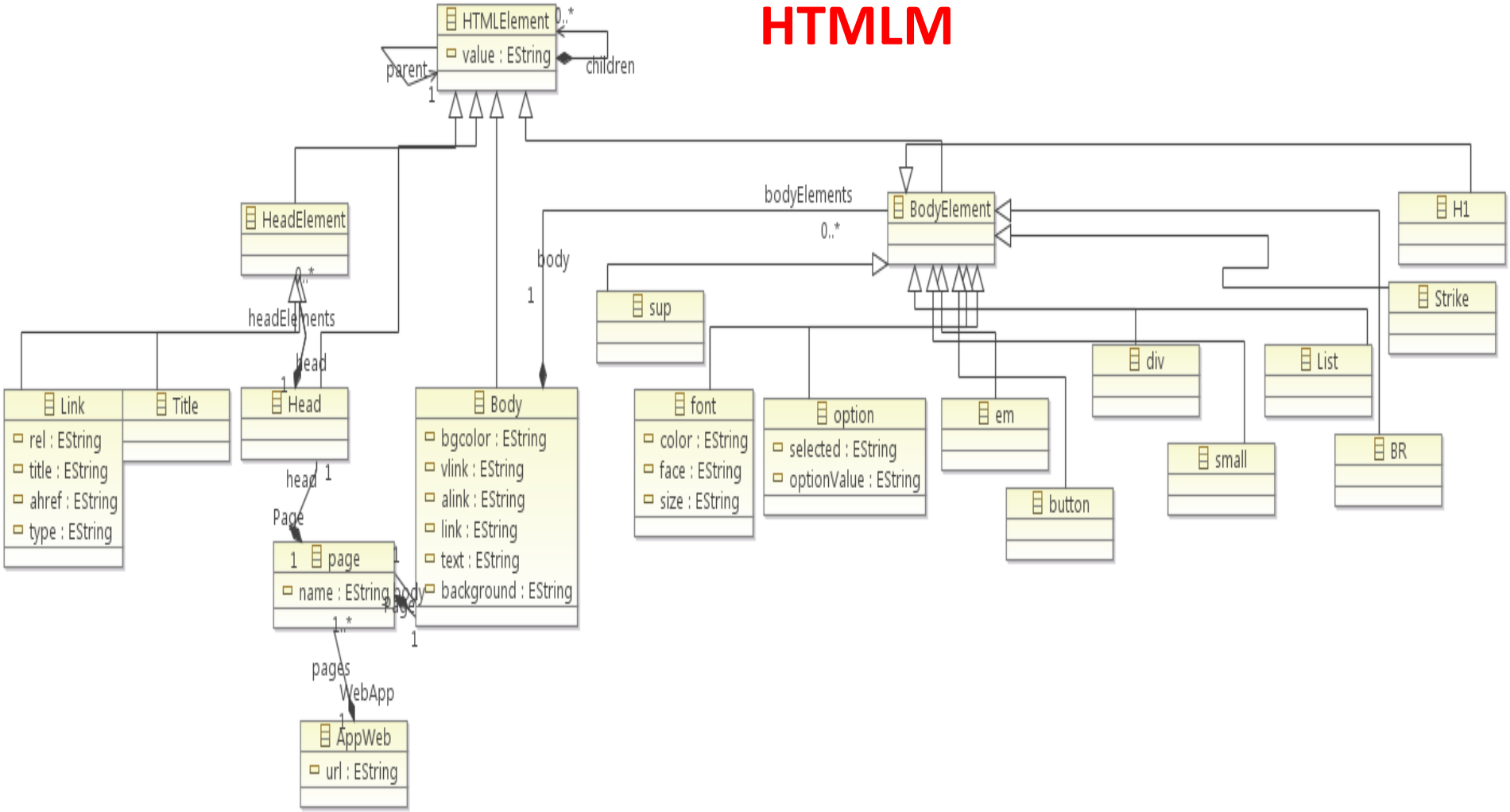
Contribution
Overview of our migration process



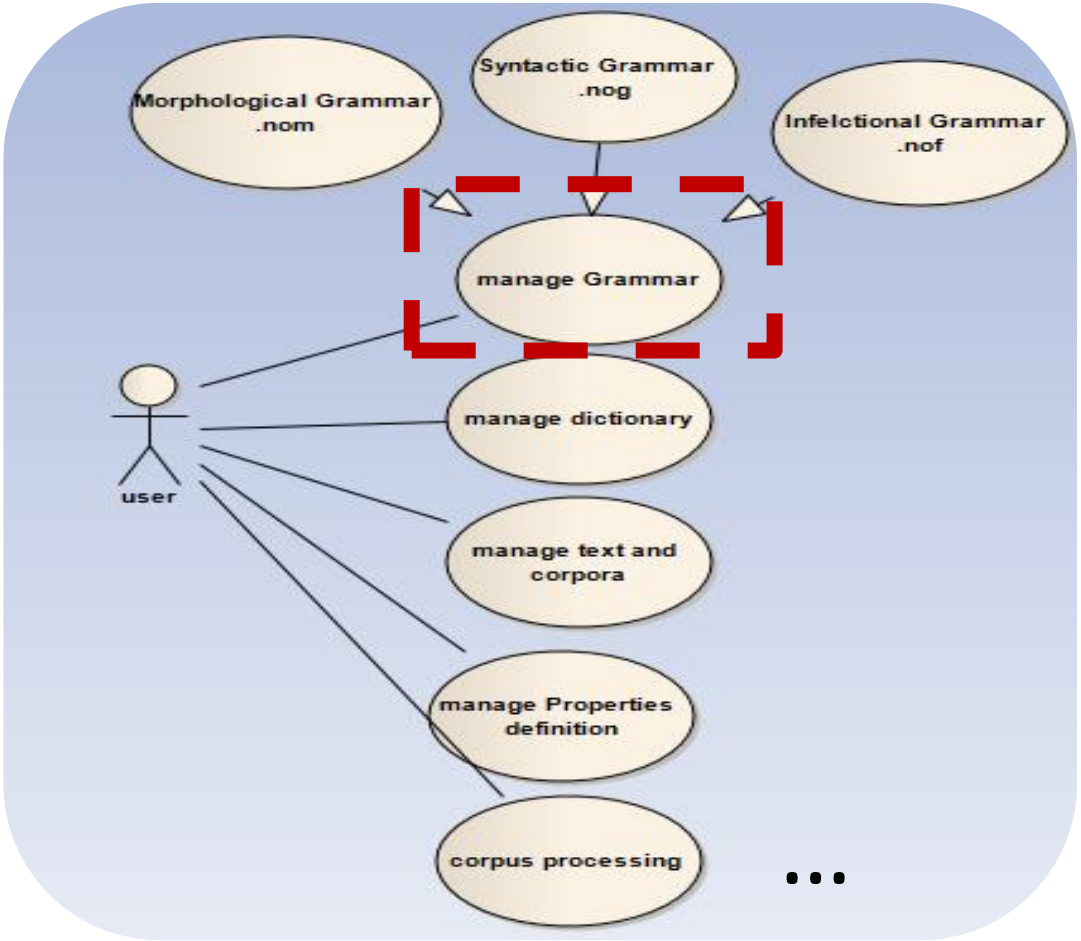
Contribution
Overview of our migration process



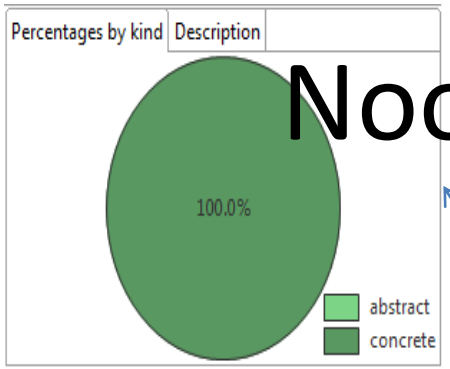
Contribution
Overview of our migration process

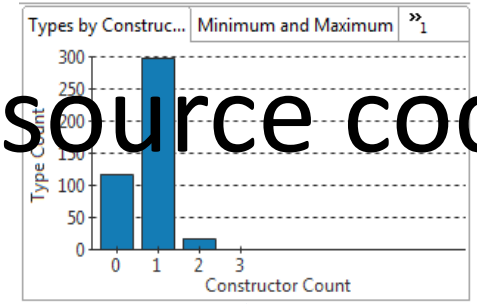


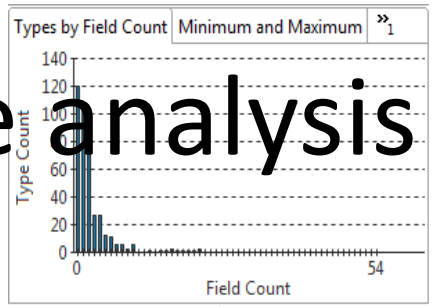
NooJ functionalities



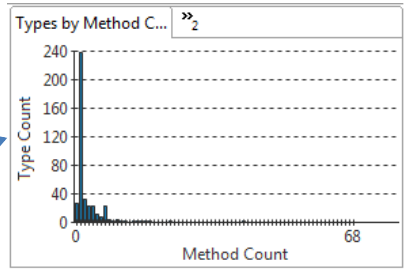
NooJ source code analysis

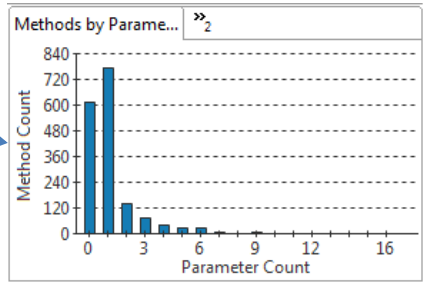




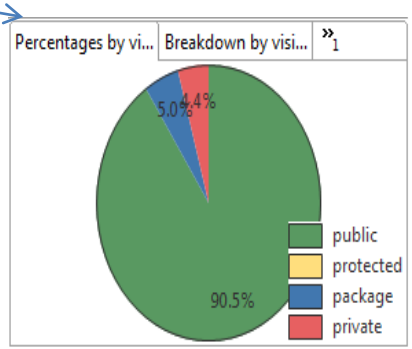


Metric	Value
Abstractness	0%
Average Block Depth	1.19
Average Cyclomatic Complexity	5.22
Average Lines Of Code Per Method	27.96
Average Number of Constructors Per Type	0.77
Average Number of Fields Per Type	3.44
Average Number of Methods Per Type	4.01
Average Number of Parameters	1.25
Comments Ratio	9.3%
Efferent Couplings	341
Lines of Code	64,265
Number of Characters	2,805,333
Number of Comments	6,006
Number of Constructors	340
Number of Fields	1,919
Number of Lines	90,604
Number of Methods	1,756
Number of Packages	49
Number of Semicolons	34,959
Number of Types	437
Weighted Methods	10,952





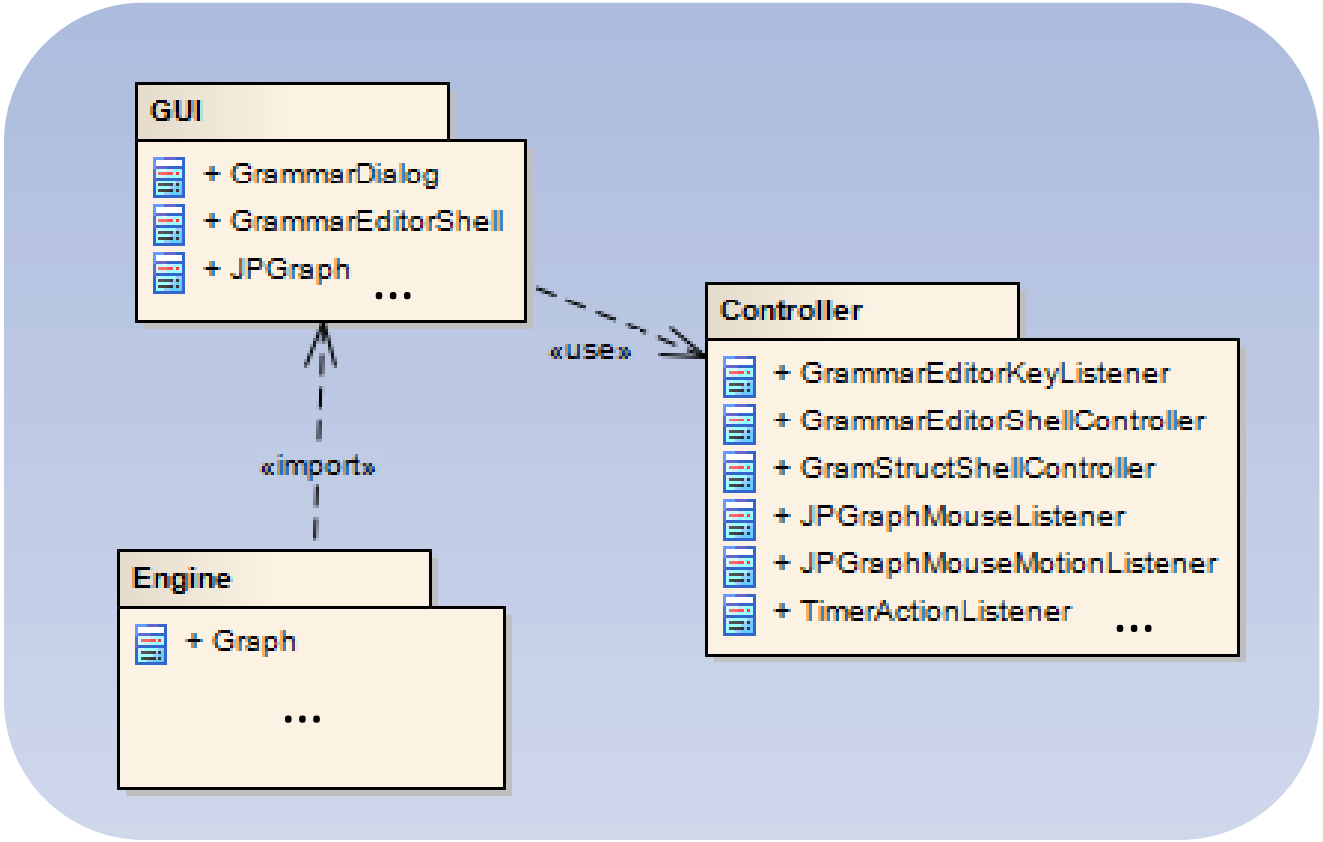
Name	Value
instance	1518
public	1149
protected	12
package	42
private	315
static	238
public	160
protected	0
package	33
private	45



interface	0
public	0
protected	0
package	0
private	0
class	437
public	327
protected	6
package	103
private	1

Name	Value
public	711
instance	328
static	383
protected	0
instance	0
static	0
package	41
instance	41
static	0
private	1167
instance	1015
static	152

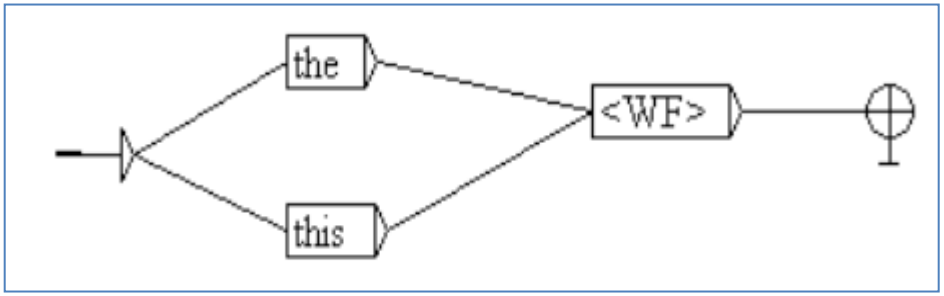
NooJ Platform Architecture

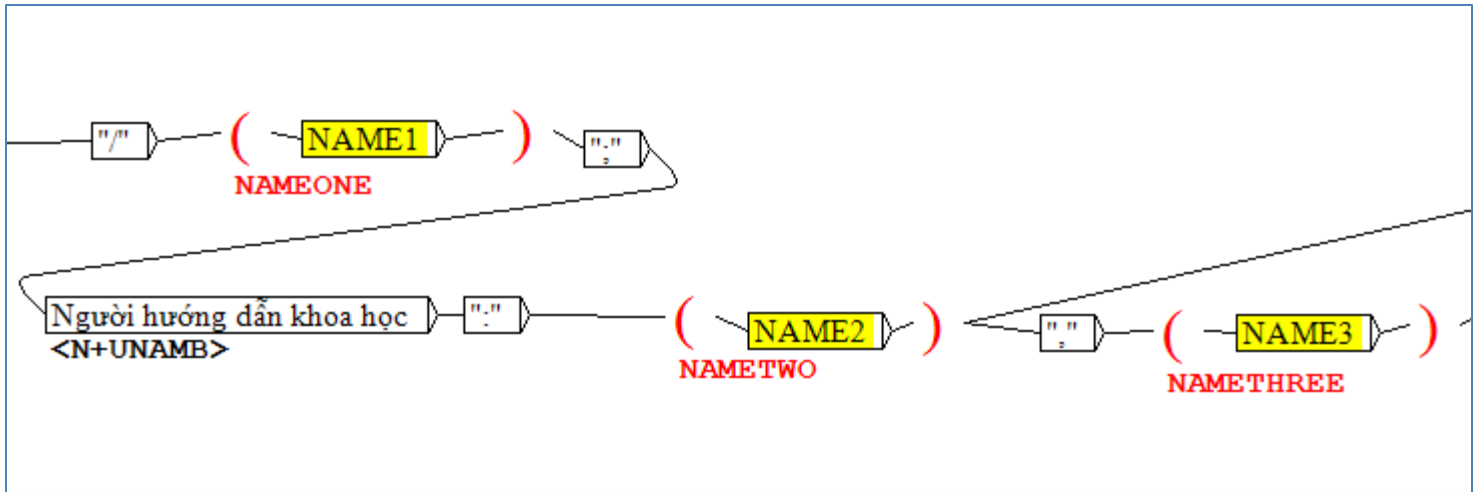


Graphical Editor Sub-System

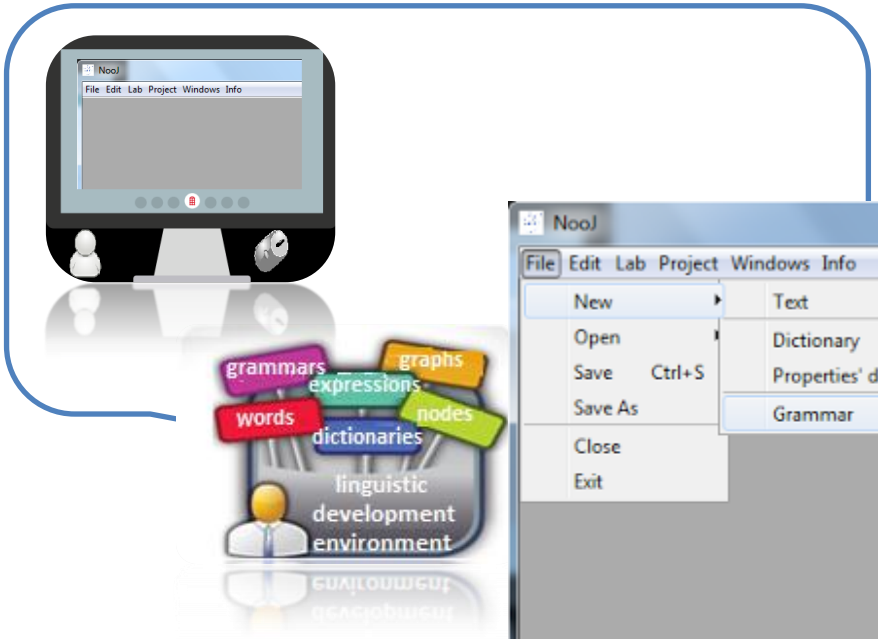
•the NooJ's graphical editor provides tools to edit, test and debug local grammars, in order to apply them to texts.

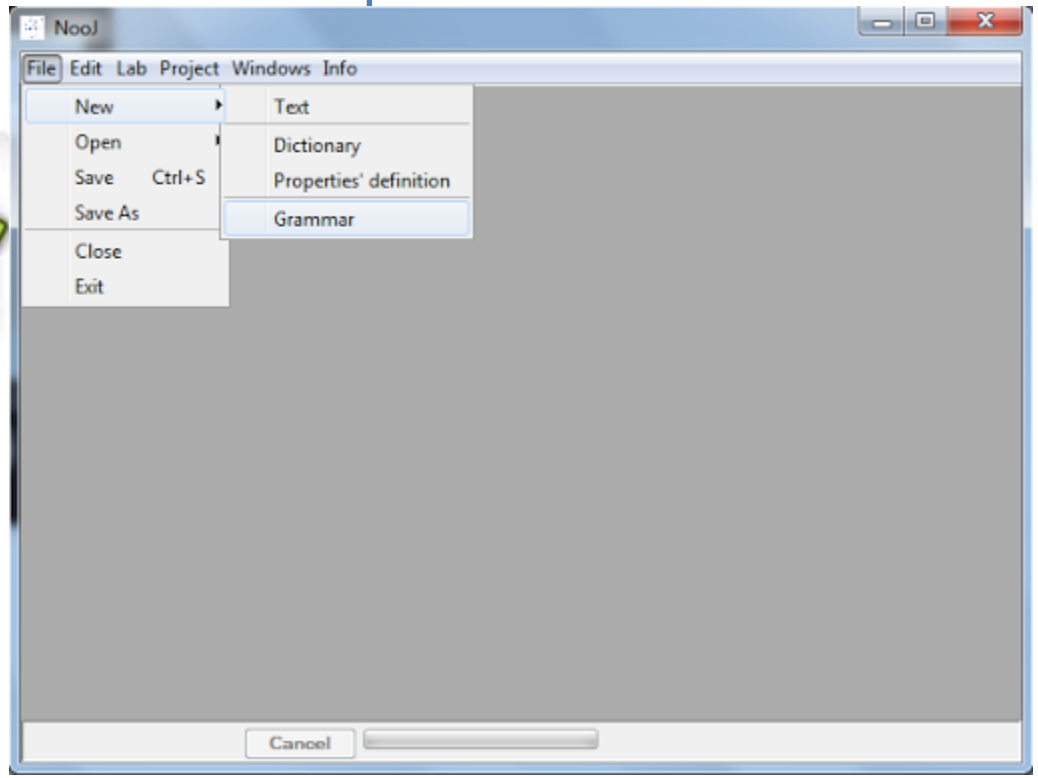
•The grammars are represented by organized sets of graphs.



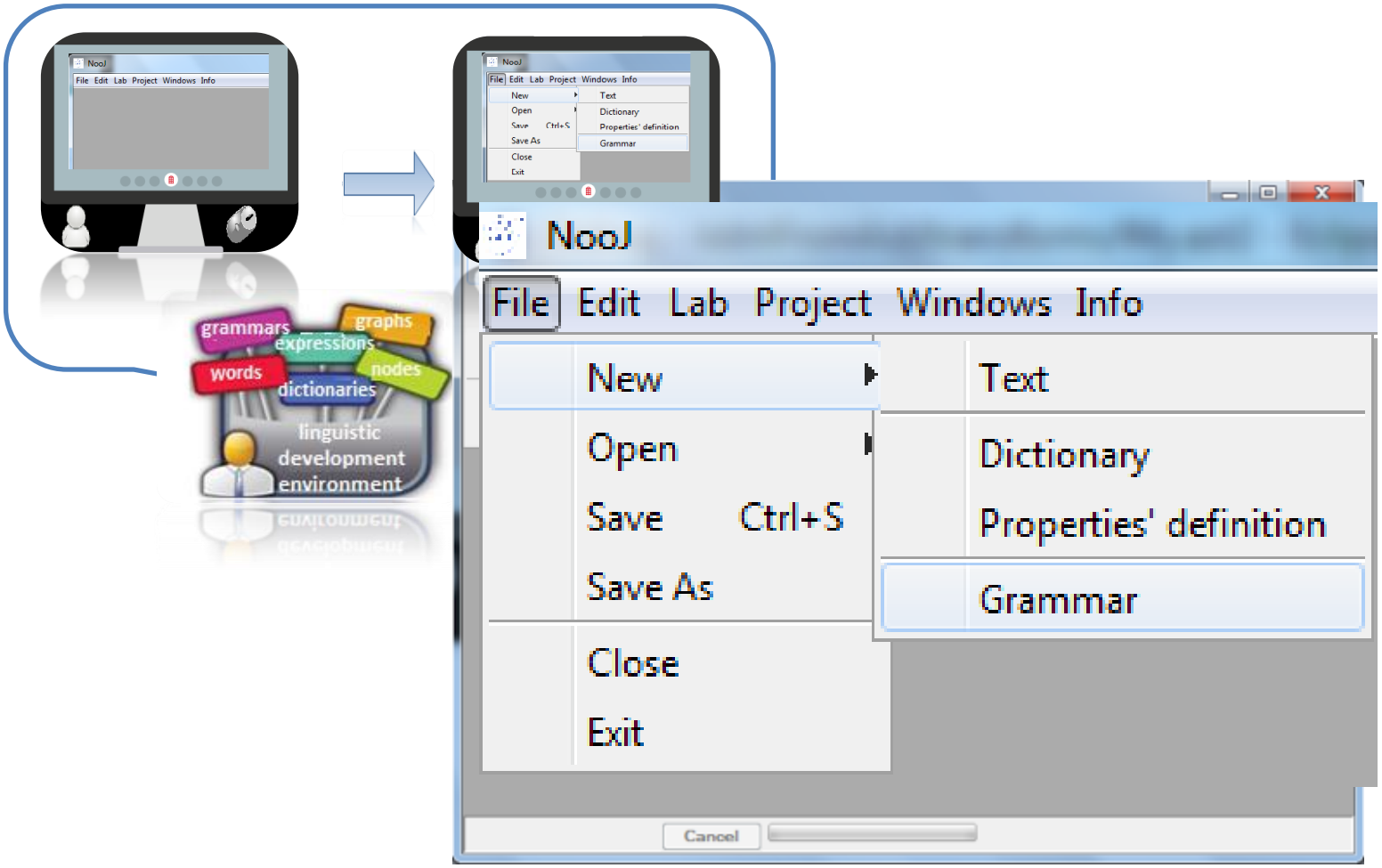


Dynamic Analysis

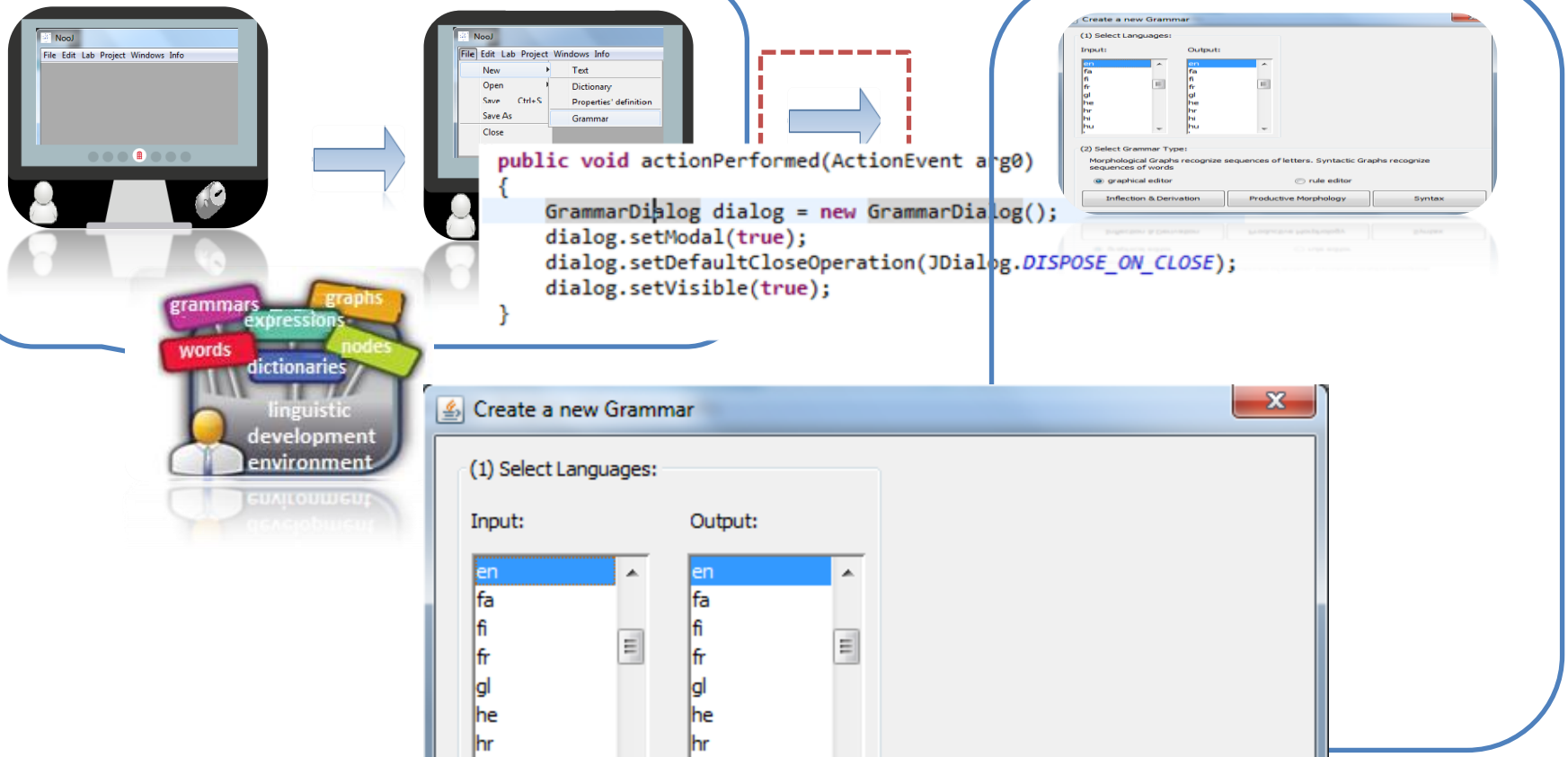




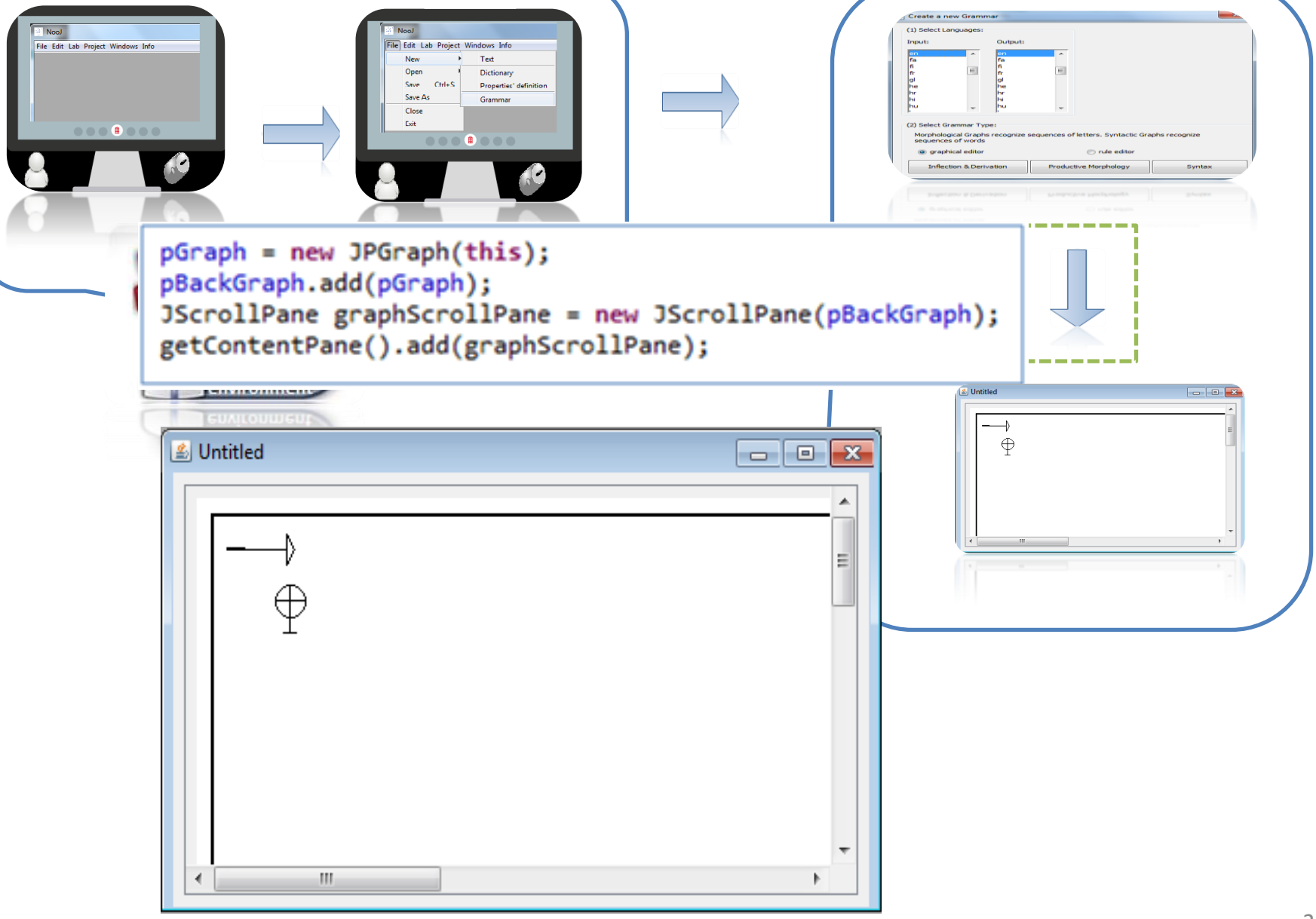
Dynamic Analysis



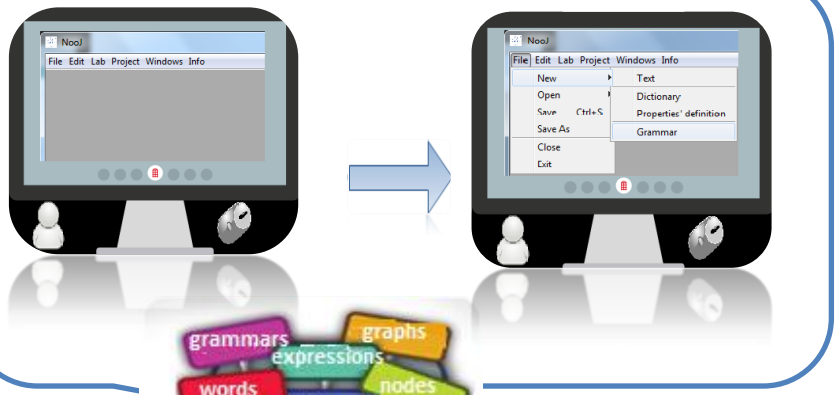
Dynamic Analysis

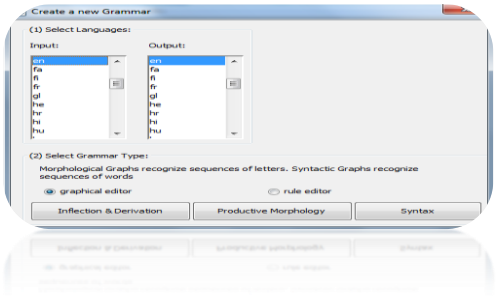


Dynamic Analysis

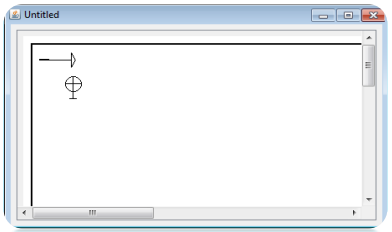


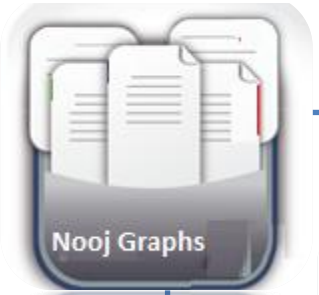
Dynamic Analysis

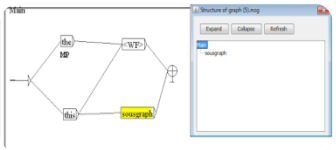




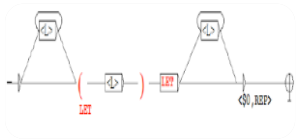

```
pGraph.addMouseMotionListener(new JPGraphMouseMotionListener(this));
pGraph.addMouseListener(new JPGraphMouseListener(this));
pGraph.addMouseWheelListener(new JPGraphMouseWheelListener(this));
this.addKeyListener(new GrammarEditorKeyListener(this));
rtBox.addKeyListener(new TextBoxKeyListener(this));
```



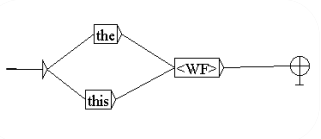




Graph2

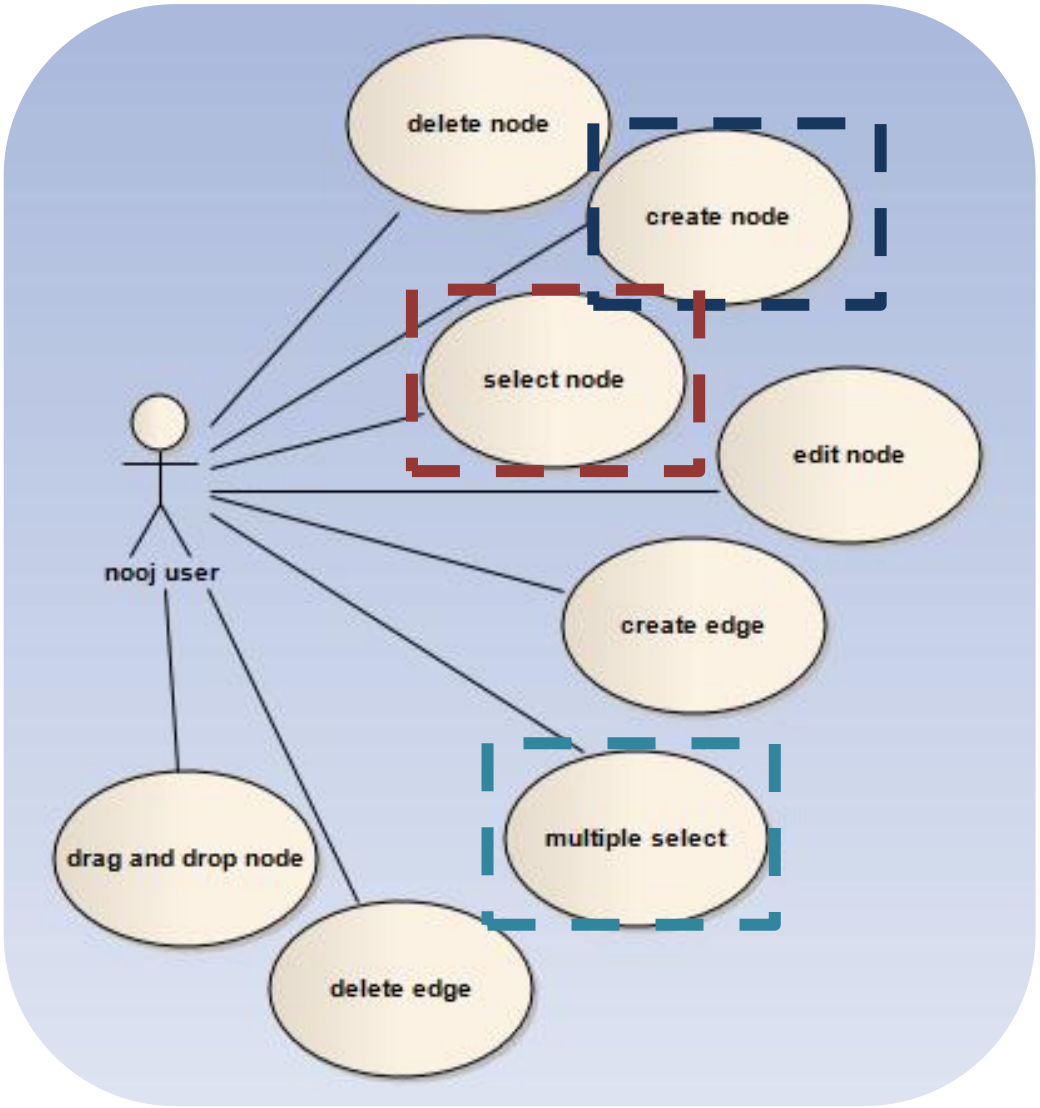


Graph2

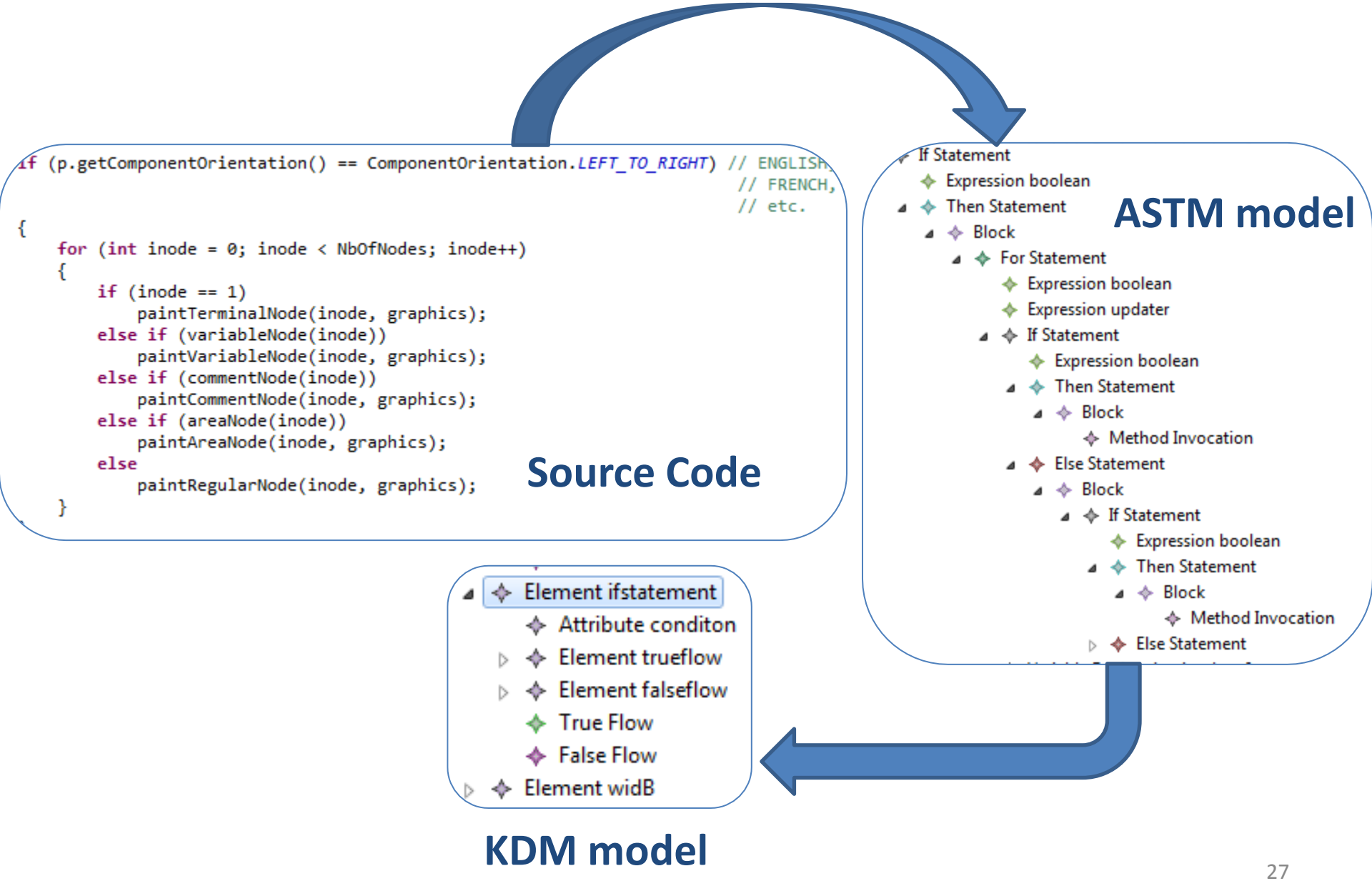


Graph1

Graphical editor functionnalities

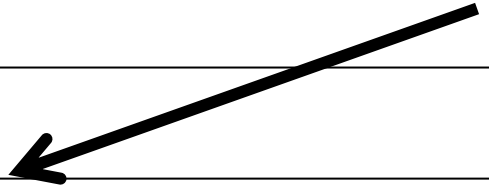


Create Node Case



Create Node Algorithm obtained

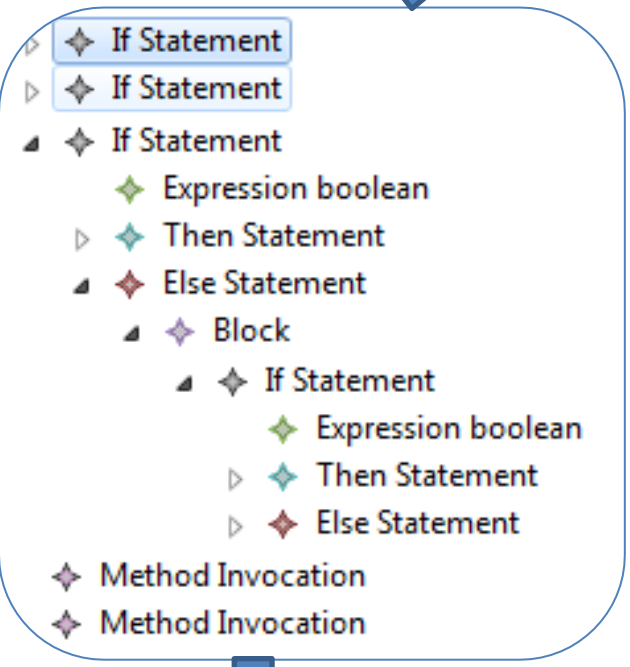
If mouse is not in a node
    If iscontrolDown or isAltDown
        Retrieve the mouse coordinates
        Add the node coordinates, width and height
        Collect all selected nodes to new node by adding the new node to the children group of each selected node
        unselect all nodes
        call the **repaint** method to paint all figures in the panel



if (node indice is 1)
    call paintTerminalNode methode
else if (node is variableNode)
    call paintVariableNode
else if (node is commentNode)
    call paintCommentNode
else if (node is areaNode)
    call paintAreaNode
else if (node is RegularNode**)**
    call paintRegularNode

Nodes Selection Case

ASTM model




```
timerSelCount = 0;
timerSel = new Timer(100, new TimerActionListener(editor));
timerSel.start();
```

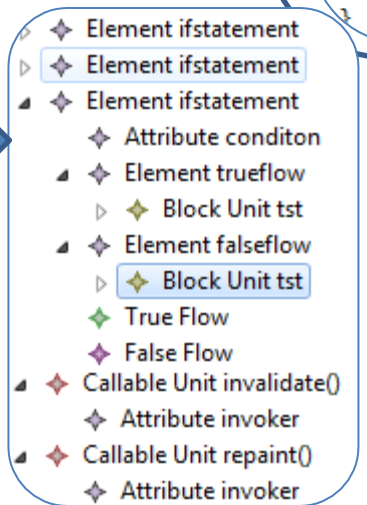
```
public void actionPerformed(ActionEvent arg0)
{
    if (editor.getController().grf == null)
        return;

    if (timerSelCount > 9)
        timerSelCount = 0;
    else
        timerSelCount++;

    if (timerSelCount >= 8)
        editor.getController().grf.tColor = new Color(255, 215, 0); // Color.Gold
    else if (timerSelCount < 2)
        editor.getController().grf.tColor = editor.getController().grammar.bColor;
    else
        editor.getController().grf.tColor = editor.getController().grammar.sColor;

    editor.pGraph.invalidate();
    editor.pGraph.repaint();
}
```

Source Code



KDM model

Nodes Selection Case Algorithm

//allowing the **selected nodes** to blink
If (nodes are selected())
    Increment a **variable timer var** each number of second
    While (**var** has changed its value)
        Change the selected node flashing color

Node Selection Case

```
2221    // add marks if node is selected
2222    if (selected != null && selected.get(inode))
2223    {
2224        Rectangle rect1 = new Rectangle(x - 10, y - hei.get(inode) - 10, 10, 10);
2225        Rectangle rect2 = new Rectangle(x - 10, y + hei.get(inode), 10, 10);
2226        Rectangle rect3 = new Rectangle(x + wid.get(inode), y - hei.get(inode) - 10, 10, 10);
2227        Rectangle rect4 = new Rectangle(x + wid.get(inode), y + hei.get(inode), 10, 10);
2228
2229        g.setColor(pen.get("penS").color);
2230        g.setStroke(new BasicStroke(pen.get("penS").stroke));
2231        g.drawArc(rect1.x - rect1.width / 2, rect1.y + rect1.height / 2, rect1.width, rect1.height, 0, 360);
2232        g.drawArc(rect2.x - rect2.width / 2, rect2.y + rect2.height / 2, rect2.width, rect2.height, 0, 360);
2233        g.drawArc(rect3.x - rect3.width / 2, rect3.y + rect3.height / 2, rect3.width, rect3.height, 0, 360);
2234        g.drawArc(rect4.x - rect4.width / 2, rect4.y + rect4.height / 2, rect4.width, rect4.height, 0, 360);
2235        // reset color
            g.setColor(pen.get("pen").color);
        }
```

Source Code

- If Statement
  - Expression boolean
  - Then Statement
    - Block
      - Variable Declaration rect1 = new Rectangle(x - 10, y - hei.get(inode) - 10, 10, 10)
      - Variable Declaration rect2 = new Rectangle(x - 10, y + hei.get(inode), 10, 10)
      - Variable Declaration rect3 = new Rectangle(x + wid.get(inode), y - hei.get(inode) - 10, 10, 10)
      - Variable Declaration rect4 = new Rectangle(x + wid.get(inode), y + hei.get(inode), 10, 10)
      - Method Invocation
      - Method Invocation
        - Class Instance Creation BasicStroke
      - Method Invocation
      - Method Invocation
      - Method Invocation
      - Method Invocation
      - Method Invocation

ASTM model

- Element ifstatement
  - Attribute conditon
  - Element trueflow
    - Block Unit tst
      - Element rect1 = new Rectangle(x - 10, y - hei.get(inode) - 10, 10, 10)
      - Element rect2 = new Rectangle(x - 10, y + hei.get(inode), 10, 10)
      - Element rect3 = new Rectangle(x + wid.get(inode), y - hei.get(inode) - 10, 10, 10)
      - Element rect4 = new Rectangle(x + wid.get(inode), y + hei.get(inode), 10, 10)
      - Callable Unit setColor(pen.get("penS").color)
      - Callable Unit setStroke(new BasicStroke(pen.get("penS").stroke))
      - Callable Unit drawArc(rect1.x - rect1.width / 2, rect1.y + rect1.height / 2, rect1.width, rect1.height, 0, 360)
      - Callable Unit drawArc(rect2.x - rect2.width / 2, rect2.y + rect2.height / 2, rect2.width, rect2.height, 0, 360)
      - Callable Unit drawArc(rect3.x - rect3.width / 2, rect3.y + rect3.height / 2, rect3.width, rect3.height, 0, 360)
      - Callable Unit drawArc(rect4.x - rect4.width / 2, rect4.y + rect4.height / 2, rect4.width, rect4.height, 0, 360)
      - Callable Unit setColor(pen.get("pen").color)

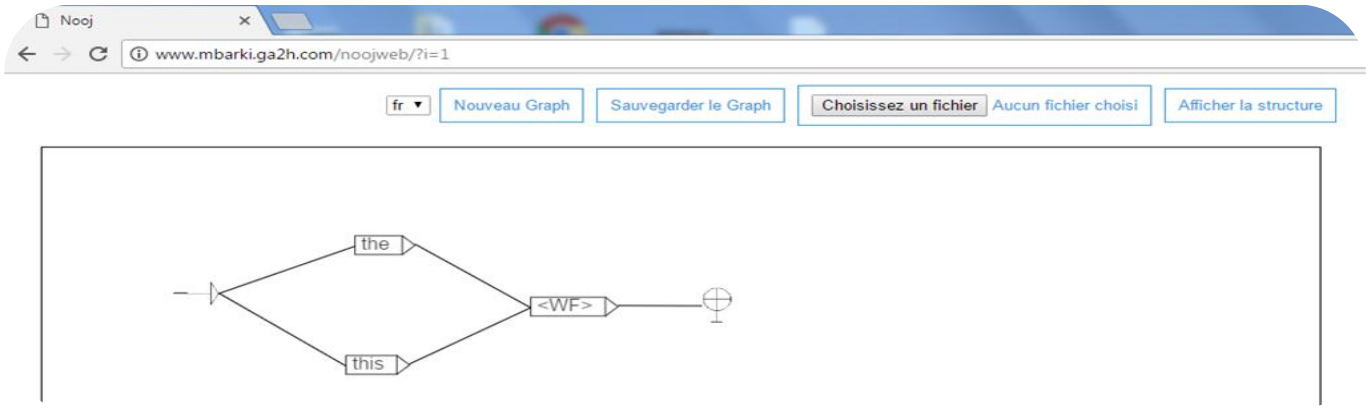
KDM model

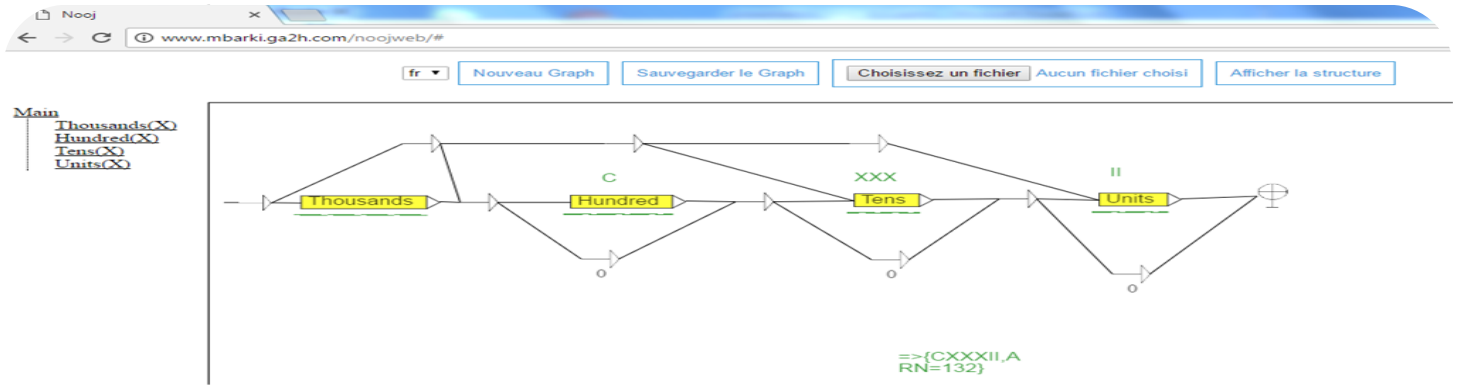
Node Selection Case Algorithm

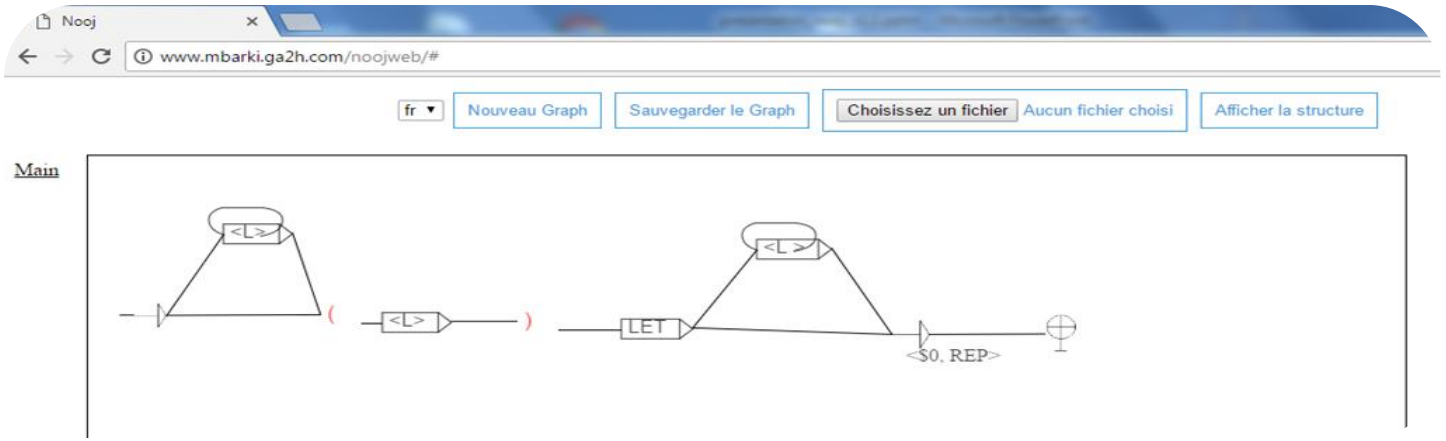
If (node is selected)
Get the node coordinates, width and height
Draw four arcs in the Top, Left, Bottom and Right
Call the above algorithm to change the color of

Repaint() the graph by adding marks to the selected node

NooJ web


Nooj
www.mbarki.ga2h.com/noojweb/?i=1
fr
Nouveau Graph
Sauvegarder le Graph
Choisissez un fichier
Aucun fichier choisi
Afficher la structure
the
<WF>
this


Nooj
www.mbarki.ga2h.com/noojweb/#
fr
Nouveau Graph
Sauvegarder le Graph
Choisissez un fichier
Aucun fichier choisi
Afficher la structure
Main
Thousands(X)
Hundred(X)
Tens(X)
Units(X)
Thousands
C
Hundred
XXX
Tens
II
Units
=>{CXXXII,A
RN=132}


Nooj
www.mbarki.ga2h.com/noojweb/#
fr
Nouveau Graph
Sauvegarder le Graph
Choisissez un fichier
Aucun fichier choisi
Afficher la structure
Main
<L>
(
<L>
)
<L>
LET
<$0, REP>

Conclusion

- In this work we have focused on an approach to automate the process of extracting the GUI's characteristics and functionalities.
- The approach is based on the ADM initiative as a best solution in the legacy system's evolution.
  - We used a static and dynamic analysis to obtain knowledge of the structure and behavior of source code.
  - We represented all necessary information in a higher level of abstraction.
  - We migrated the abstract KDM model obtaned into new specific platforms which are the JavaScript and HTML.
- In this work we focused only on the NooJ graphical editor.
- This work can be extended to treat the other NooJ functionalities to be migrated in different platforms.