

# Framework for Requirements-Driven System Design Automation

Mihai Fonoage, PhD Candidate – [mfonoage@fau.edu](mailto:mfonoage@fau.edu)

Dr. Ionut Cardei – [icardei@cse.fau.edu](mailto:icardei@cse.fau.edu)

Dr. Ravi Shankar – [ravi@cse.fau.edu](mailto:ravi@cse.fau.edu)

Center for Systems Integration  
Department of Computer Science and Engineering  
Florida Atlantic University

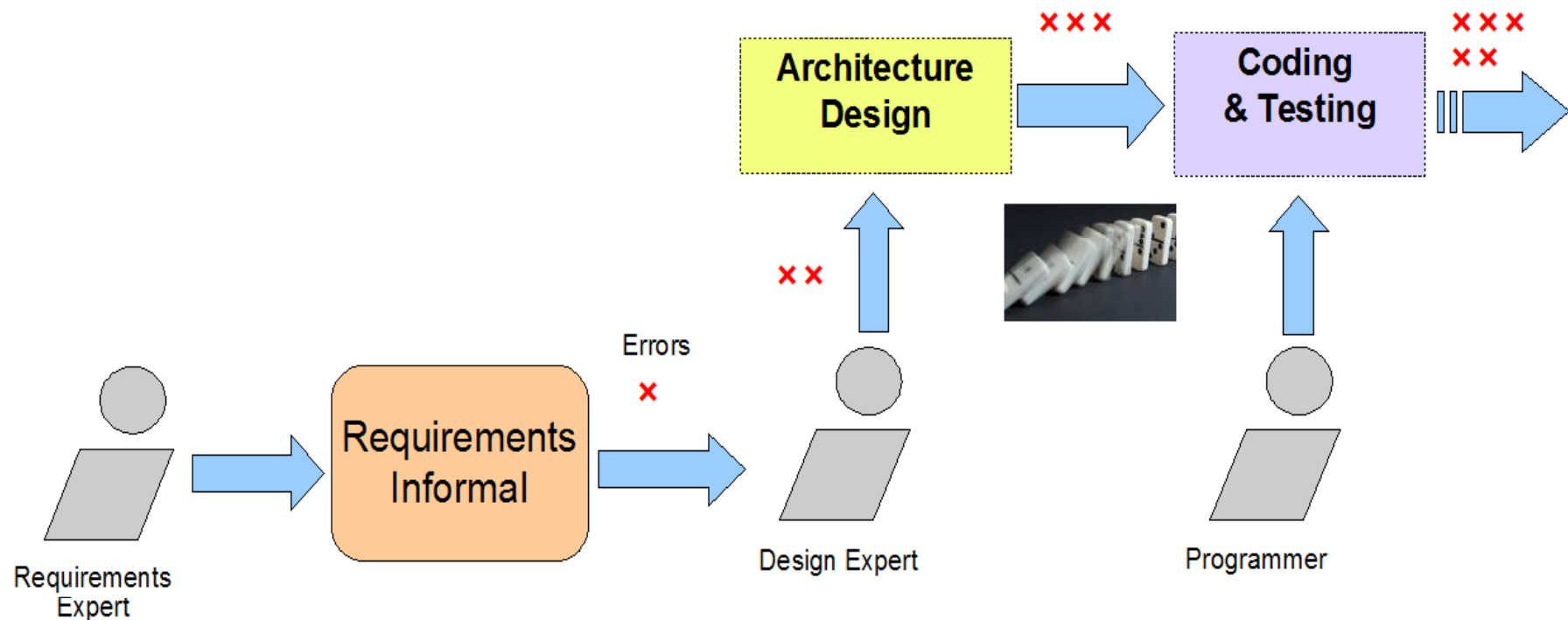
*04/11/07*

# Outline

- Introduction
- Methodology
- RDDA Architecture
- Related Work
- Conclusions
- References

# Introduction

- Current Development Method:



- ISSUES:

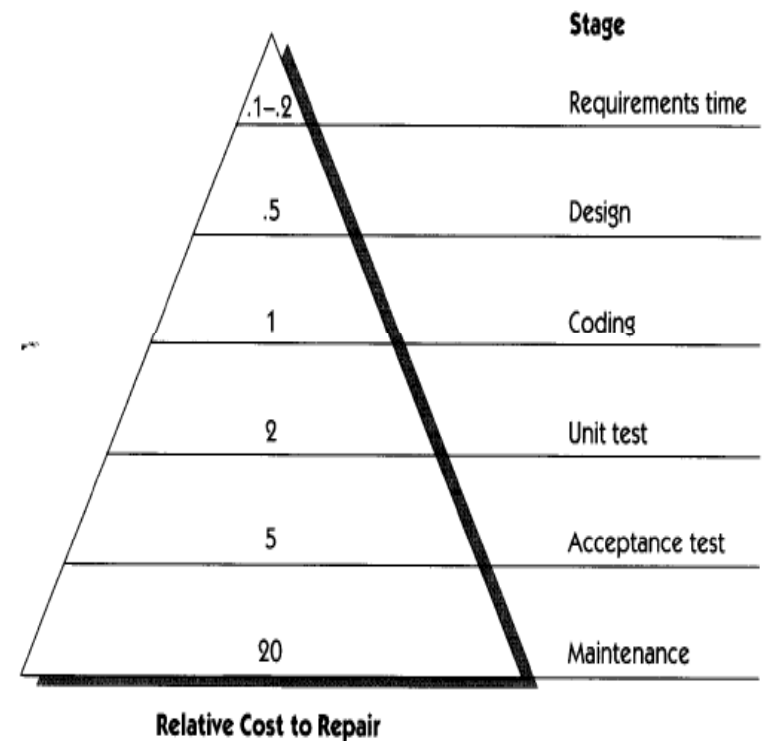
- Requirements errors permeate through all development stages...
- Requirements changes domino effect affect all development stages...

# Introduction (cont.)

## Software Development Cost Breakdown

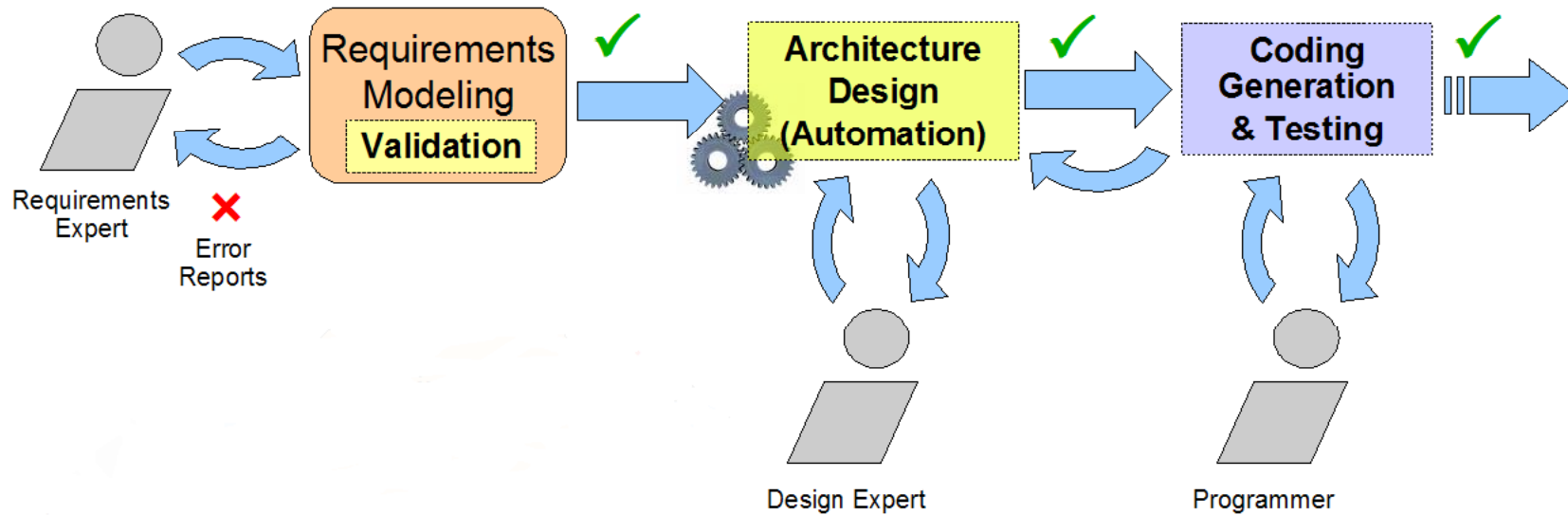
<u>Component</u>	<u>Percentage</u>
Problem comprehension	27%
Requirements analysis	24%
Design	18%
Implementation	18%
Testing	13%

[Grady, Hewlett-Packard]



**Figure 1-2** Relative cost to repair a defect at different lifecycle phases. (Data derived from Davis [1993].)

# Introduction (cont.)



Components of a development iteration:  
(investigated in our [approach](#))

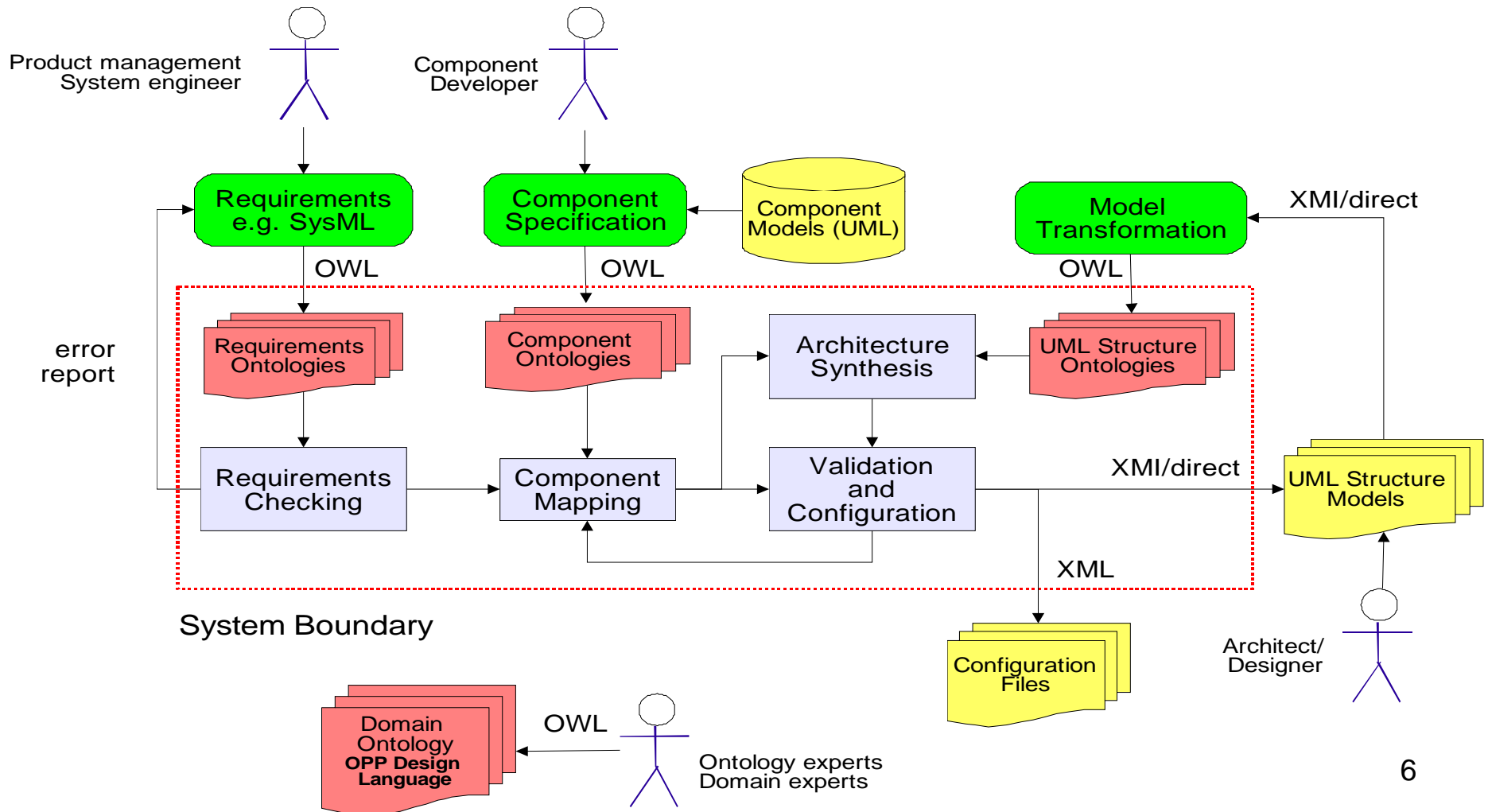
1. [Formal requirements model specification](#)
2. Requirements specification tools
3. [Requirements model validation](#)

Components (cont.):

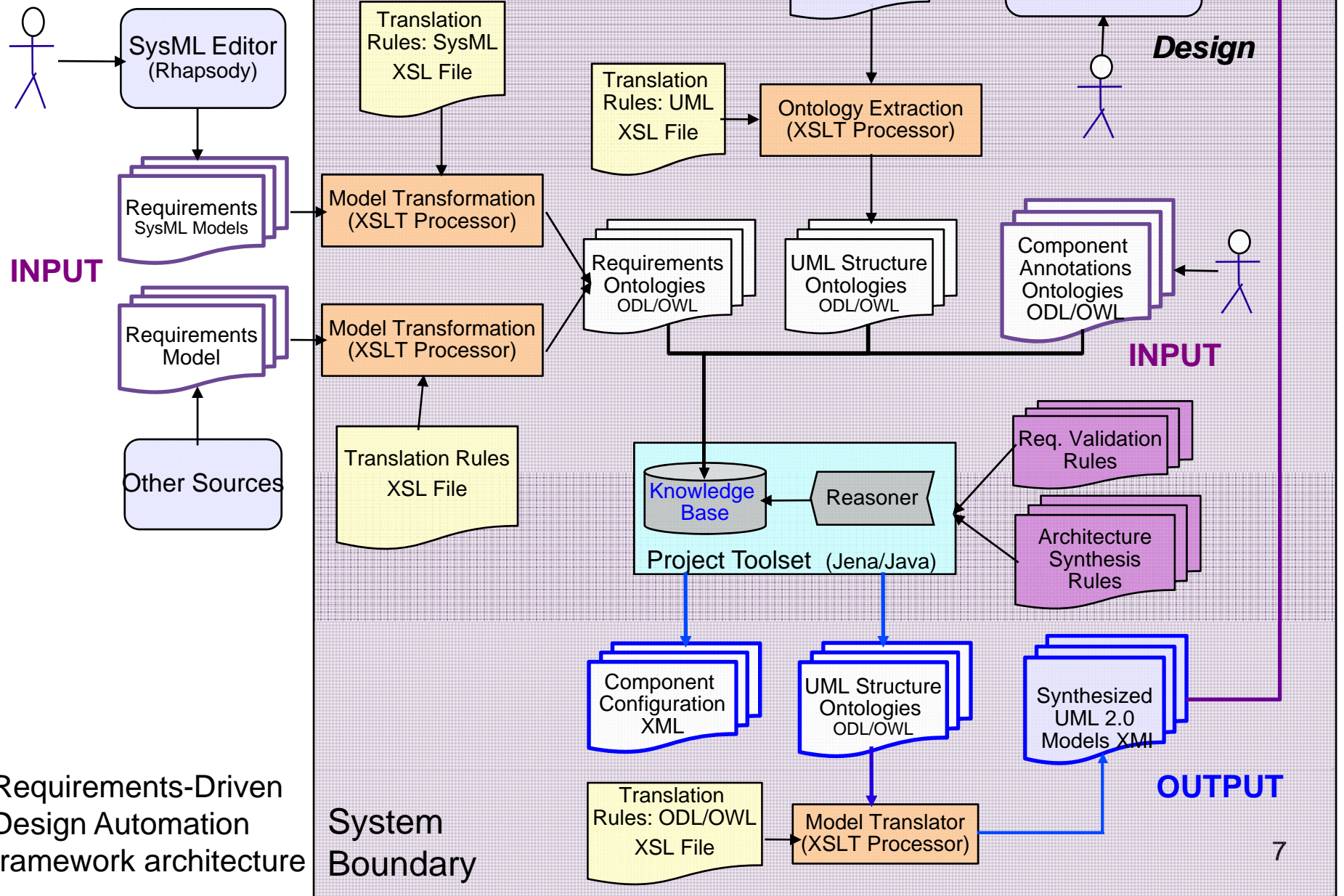
4. Automation design
  - 4.1. [Architecture composition](#)
  - 4.2. [Component selection](#)
  - 4.3. [Model semantic annotation](#)
5. Design model validation
6. Code generation & testing

# Methodology

- RDDA (Requirements-Driven Design Automation) methodology flow:



# Architecture



# RDDA Architecture

- The OPP Design Language (ODL)



# RDDA Architecture

- *The Requirements Domain:*
  - defines concepts, relationships, and properties necessary to describe requirements

Concepts from the requirements ontology.

Concept	Purpose
RqProduct	The top-level system that is the subject of these requirements (system, application or a component).
RqSubsystem	A subsystem of the product design hierarchy.
RqFeature	A feature that is required to be supported by the designed system. Features can be capabilities or behaviors.
RqConstraint	A generic constraint that applies to a subsystem or component. Constraints can be physical constraints (weight, volume), QoS constraints (data rate, delay), and system resource constraints (CPU load, memory, bandwidth).
RqReqStmt	Represents the natural language text for one requirement statement.
RqVersion	Represents a requirements model version number.

# RDDA Architecture

- *The Design Domain:*
  - ODL Ontologies describe UML and SysML structural models
- *The Component Annotation Domain*
  - Specifies semantic descriptions for component annotations
    - Component capabilities, constraints, QoS, design elements (i.e. provided and required interfaces)
  - Covers platform-specific information (i.e. hardware platform, OS, programming language)

# RDDA Architecture

- *Requirements Specification Example:*

Textual Requirements:

“1.1 The GPS provides a location accuracy of minimum 10m, a query response time of maximum 10s.”

ODL-based Requirements:

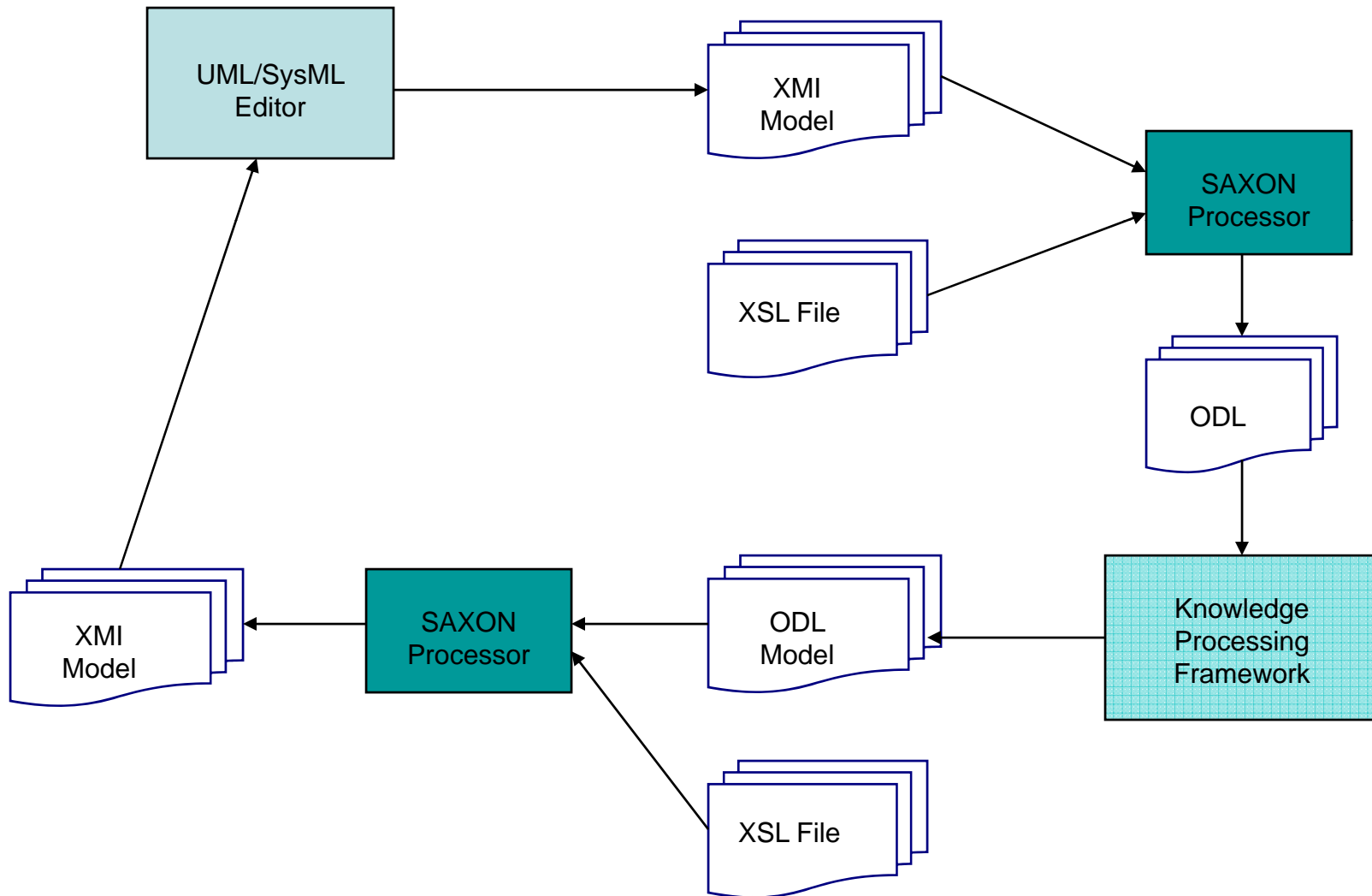
```
<RqReqStmt rdf:ID = "ReqStmt_1.1">
  <hasVersion rdf:resource = "#ReqVersion_v0.1" />

  <rqTracks>
    <RqSubsystem rdf:resource = "#Subsys_GPS">
      <hasCapability rdf:resource = "#GPS_Positioning" />
      <hasConstraint rdf:resource = "#QoS_minLocationAccuracy_10m" />
      <hasConstraint rdf:resource = "#QoS_maxQueryResponseTime_10s" />
    </RqSubsystem>
  </rqTracks>

  <reqStmtText>
    1.1 The GPS provides a location accuracy of minimum 10m, a query
    response time of maximum 10s.
  </reqStmtText>
</RqReqStmt>
```

# RDDA Architecture

- Model Translation:



# RDDA Architecture

- XSL transformation rules for UML Interfaces from XMI code:

```
<xsl:template match = "UML:Package/UML:Namespace.ownedElement/  
    UML:Interface">  
    <odl:MSwInterface rdf:ID="{../../@name}.{@name}">  
        <odl:hasName rdf:datatype="~xsd:string">  
            <xsl:value-of select="@name"/>  
        </odl:hasName>  
        <odl:inPackage rdf:resource="#{../../@name}"/>  
        <odl:relEndVisibility rdf:resource="#{@visibility}"/>  
        <odl:hasXmild rdf:resource="#{@xmi.id}"/>  
    </odl:MSwInterface>  
</xsl:template>
```

# RDDA Architecture

- Knowledge Base and the Reasoner
  - ODL Ontologies are loaded into a knowledge base
  - KB is supported by:
    - The Jess framework
    - The OWLJessKB semantic web library
  - The reasoner component, provided by Jess, implements a rule-based interface engine
    - Supports backward and forward chaining
  - Knowledge processing is guided by a set of rules
    - Requirements Validation Rules
    - Architecture Synthesis Rules

# Related Work

- Software Maintenance using Semantic Web [2].
- Requirements processing using Ontologies [3].
- CASSANDRA – assist and guide developers through the software development process [4].
- Dynamic composition of Web Services [5], [6].
- Semantic Streams framework – use declarative statements to query a sensor network [7].

# Conclusions

- Framework for improving design productivity through automation
- Main idea: describe product requirements and component semantics using a common language (ODL – OWL-based language)
- OWL supports automated reasoning
- The reasoner performs:
  - Requirement validation
  - Matching requirements with components
  - UML diagram synthesis

# References

- [1] – **Ionut Cardei, Mihai Fonoage, Ravi Shankar**, “Framework for *Requirements-Driven System Design Automation*”. In *IEEE System Conference*, April 2007
- [2] - **D. Hyland-Wood, D. Carrington, and S. Kaplan**, “*Enhancing software maintenance by using semantic web techniques.*” In *International Semantic Web Conference (ISWC)*, 2006.
- [3] - **Haruhiko Kaiya and Motoshi Saeki**, “*Ontology based requirements analysis: Lightweight semantic processing approach*”. In *Fifth International Conference on Quality Software (QSIC 2005)*, 2005.
- [4] - **Markus Schacher**, “*CASSANDRA: An automated software engineering coach.*” In *KnowGravity.com*, 2000.
- [5] - **Evren Sirin, James Hendler, and Bijan Parsia**, “*Semiautomatic composition of web services using semantic descriptions.*” In *Web services: modeling, architecture and infrastructure workshop in ICEIS*, 2003.
- [6] - **W3C**, “*OWL-S: Semantic markup for web services.*” <http://www.w3.org/Submission/OWL-S/>.
- [7] - **J. Liu and F. Zhao**, “*Towards semantic services for sensor-rich information systems.*” In *The 2<sup>nd</sup> International Conference on Broadband Networks*, pages 44–51, 2005.
- [8] - **Michael Kay**, “*XSL Transformations (XSLT) Version 2.0.*” <http://www.w3.org/TR/xslt20/>.
- [9] - **W3C**, “*The ontology web language.*” <http://www.w3.org/2004/OWL>.
- [10] – “*Jess, the rule engine for Java.*” <http://www.jessrules.com>.