Business Rules Design and Analysis Approaches

Grzegorz J. Nalepa

Institute of Automatics
AGH University of Science and Technology
POLAND
Outline of the presentation

• Introduction – the AI perspective to BRA
• Business Rules Concepts
• BR and Classic RBS
• Computer Tools for Business Rules
• Observations, Quality Issues
• A Business Rule Base Design Example
• The XTT Approach
• Conclusions
Introduction

• Focus: building Knowledge-Based Systems in the field of business software
  – Building real-life KBS is a complex task
  – We need Knowledge Engineering
  – Rules: a classic knowledge representation
• Business Rules (BR) - a rediscovery of a classic AI technology of RBS
• Software engineering becomes more and more knowledge-based, see the BRA
• How can AI experts help business engineers?
RBS in the Classic AI

• Rule language
• Rule base design
• Rule inference: forward vs. backward
• Rule interpretation algorithm
• An inference engine
• Complete shells
• Quality issues!
RBS Analysis

- KBS vs. computational intelligence (ANN, GA)
- Strong logical foundations (PL, FOPL, SAL)
- Formalized methods for design
- Formal analysis:
  - Verification - whitebox
  - Validation
  - Evaluation - blackbox
  - Refinement
Basics of the Business Rules Approach

• Guidelines – an informal approach
• Rule types: e.g. reactive, computational
• Business concepts vocabularies (attributes)
• Emphasis on the:
  – KE process, including knowledge acquisition, business vocabularies
  – Visual BR modelling
  – BR Management
• No explicit analysis, just testing?
Three Main Problem Areas

• Logical foundations
  → BR classification seems to be incomplete, mixing rule syntax, semantics, and inference
  → Semantics close to NL, no logical calculus
  → Inference: deduction, abduction, induction

• Visual representation
  → Scalability issues
  → Automatic transformations

• Quality issues
  → Testing is not enough!
  → Late evaluation problem
BR Tools Classes

- Business Rules Engines – RBS shells - Jess
- Design tools – dedicated, general spreadsheet
- Markup languages - RuleML
- Dedicated representation methods - URML
- Integrated solutions - Business Rules Management Systems...
- What about analysis? - VALENS
Tools Main Features

• Visual knowledge representation, mainly \textit{classic} methods, e.g. decision trees
• Machine readable rule encoding: XML
• Automatic code generation: Java
• Use of well established tools: spreadsheet?

What is the innovation of the design \textit{process}?
General Problems: semantic gaps

The *semantic gaps* between:
- Requirements specification
- Design Representation
- Logical Model
- Physical Implementation

Not so many improvements...
(NASA: R2D2C)
Problems: Quality Issues

Quality assurance:
• Automated testing
• Evaluation (blackbox)
• Validation
• Refinement
• Verification (whitebox)
A step back, compared to e.g. VALENS
What do we need V&V for?

• basic formal properties, e.g.:
  – Determinism
  – Completeness
  – Non redundancy

• translate into important system features e.g.:
  – Performance
  – Maintainability
  – Safety!
  – Happy customers?
A Simple BR Example

OpenRules – an open integrated solution
• Semi-visual design in spreadsheet
• Automated Java code generation
• Optimization: rule solver

Loan pre-qualification
• 16 rules
• 15 attributes
• forward chaining
OpenRules Design Process

1. describe the rules in the natural language,
2. define a glossary (attributes description),
3. use spreadsheet as the “design” tool for business rules,
4. fill the cells with some parts of Java code,
5. generate data for the runtime environment.
A Room for Improvement

• there is a clear structure in the rule base, but
• the “design” tool has no facilities to properly model this structure
• the conceptual, rule-related parts, are mixed with pseudo-code, or parts of Java code
• no analysis!
XTT Knowledge Representation

• simplicity, transparency, due to an intuitive visual tree-table knowledge specification,
• hierarchical knowledge representation
• power of the decision table representation
• knowledge manipulation flexibility
• direct knowledge representation mapping into Prolog and rule-based systems
• direct mapping to XML-based languages
<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>An</th>
<th>-X</th>
<th>+Y</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1l</td>
<td>a1n</td>
<td>x1</td>
<td>y1</td>
<td>h1</td>
<td></td>
</tr>
<tr>
<td>an1</td>
<td>ann</td>
<td>xn</td>
<td>yn</td>
<td>hn</td>
<td></td>
</tr>
</tbody>
</table>
XTT Design Process

1. Conceptual modelling, system attributes and their functional relationships are identified, ARD helper method is used here

2. Logical design with on-line verification, system structure is represented as XTT hierarchy, which can be instantly analyzed

3. Automated implementation, when an executable Prolog code is generated, as well as XML representation.
### XTT Visual Design

#### Income Valid

<table>
<thead>
<tr>
<th>gCM</th>
<th>gCMID</th>
<th>gLA</th>
<th>gVR</th>
<th>gVRID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
</tbody>
</table>

#### Debt Valid

<table>
<thead>
<tr>
<th>gMH</th>
<th>gCG</th>
<th>gLM</th>
<th>gGCB</th>
<th>gELB</th>
<th>gMR</th>
<th>gAD</th>
<th>gDRR</th>
<th>gDRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>high</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>low</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>high</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>high</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>low</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>high</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>low</td>
<td>any</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>high</td>
<td>any</td>
</tr>
</tbody>
</table>

#### Define Summary

<table>
<thead>
<tr>
<th>gVR</th>
<th>gDRR</th>
<th>gLA</th>
<th>gDA</th>
<th>gDFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>any</td>
<td>high</td>
<td>any</td>
<td>1</td>
<td>not_defined</td>
</tr>
<tr>
<td>0</td>
<td>any</td>
<td>0</td>
<td>any</td>
<td>valid</td>
</tr>
</tbody>
</table>
XTT Prolog Representation

- Transformation from XTT to a Prolog-based representation - a *logically equivalent* code
- It can be executed, analyzed, verified, optimized, translated to another language.
- Rules are represented as Prolog *facts*. This allows for encoding virtually any structured information, a need for a meta-interpreter.

```
rule(2, 3, [f(aTD, atomic, wd), f(aTM, interval, interval(9, 17))], [f(aOP, atomic, _)], [f(aOP, atomic, true)], [], 3, 7).
```
Rule Analysis in XTT

• The external analysis, verification and optimization modules are implemented in Prolog. Each module carries out the analysis of the given property, e.g. subsumption:

```prolog
vsu(T):-
    rule(T,N1,P1,R1,A1,D1,_,_), rule(T,N2,P2,R2,A2,D2,_,_),
    N1 \= N2, subsumes(P1,P2), covers(D1,D2),
    write('*** Rule: '),
    write(T),write('.'),write(N1),write(' subsumes: '),
    write(T),write('.'),write(N2), nl, fail.
vsu(T):-
    write('No more subsumption in table '), write(T), nl.
```
XTT Usability – Future Work

Promising results, but:
• Research and concepts, not technology.
• Tools – early prototypes (Mirella Designer).
• As of now analysis plugins need extensions.
• Business rules support not complete.
• Only preliminary integration with Java.
The Hekate Project

Goals of the project are:
• Incorporate KE into SE
• Extend rule-based technology to build the logic core of the application
• Run the core using embedded Prolog
• Integrate it with a Java business application
• Provide tools for formal analysis of the core
• Fill the semantic gaps in the design process...
Concluding Remarks

• BRA, an application of the “good old” RBS.
• BR community could benefit from talking to AI experts.
• We need more conceptual innovation, not just technology integration.
• Quality issues are important!
• System analysis after the design is late!
• XTT aims at providing visual design and formal analysis during the design.
  • See references in the slide notes.
Thank you!