Visual Design Tools for XTT-based Rulebases*

Grzegorz J. Nalepa[1 2]

[1] Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
<*gjn@agh.edu.pl*>
[2] Institute of Physics,
Kielce Pedagogical University
Ul. Swiętokrzyska 15, 25-406 Kielce, Poland

Abstract *XTT is a hybrid design method for rule-based expert systems, combining decision trees and tables. The XTT method provides means for logical formulation of a hierarchical rulebase with explicit inference control. It also has a well-defined visual representation, which is useful in the design process. The paper presents new tools for the conceptual and logical design. The first tool is based is based on the popular Dia vector drawing application. The second tool is a custom visual editor, that supports the visual design during the logical phase. It allows for drawing XTT tables and defining the attribute values. The tool is based on the common object-oriented Qt library, available for multiple platforms. The tools have been developed by students in the Institute of Automatics for the HeKatE project. They can be considered a work in progress. The Hybrid Knowledge Engineering project aims at providing AI design tools for general software engineering.*

1. Introduction Rule-Based systems, are the main class of the Knowledge Based Systems [4, 1]. In recent years there has been a renaissance of this technology, with enterprise systems build on the rule-based knowledge representation. However, building Rule-Based Systems (RBS) is a complex task. It requires efficient design methods for the knowledge base, as well as implementation tools for the inference engine. What is even more difficult, is the analysis of the knowledge base, which is critical in order to achieve system safety and superior performance. In case of any real-life systems these task have to automated, and supported by dedicated computer tools.

This paper is dedicated to the presentation of several design tools for the XTT knowledge representation method for RBS [8]. XTT provide an integrated design and analysis process for such systems. The paper presents the evolution and the development of these tools from the first prototype, to the currently available solutions.

In Sect. 2 the XTT method is briefly discussed. Then the approach to the development of the design tools is given. In the subsequent sections, several tools are presented, with the prototype Mirella Designer in Sect. 3, and new tools in Sects. 4–6. The paper ends with plans for the future work.

2. Visual Rule Design with XTT XTT (*eXtended Tabular Trees*) [8] is an integrated method for the design, analysis, and implementation of rule-based expert systems (RBS). The method is based on the idea of a logical design of the rule base, using a structured, hierarchical representation. The method combines extended attributive decision tables, with explicit inference control, that allows for building decision-tree-like meta structure.

The logical design can be supported by the conceptual design that allows for specifying systems attributes, and their functional dependencies. The ARD (*Attribute Relationship Diagrams*) method [7] serves for this purpose. It allows for visualizing the gradual specification of the func-

*The paper is supported by the *HEKATE* Project funded from 2007–2009 resources for science as a research project.

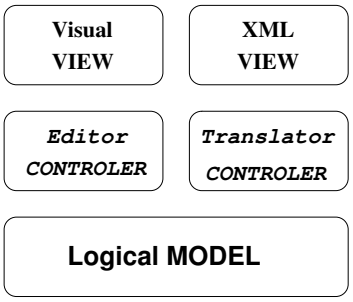

Figure 1. The MVC architecture for XTT tools.

tional relations of attributes. The process specifies attributes on different levels of abstraction.

The logical design of a RBS can be automatically transformed into a corresponding representation in Prolog [10]. It allows for both formal system analysis, as well as providing an executable system prototype. This approach allows for closing the so-called *semantic gaps* in the RBS engineering [6]. The XTT method is being currently extended into a *generalised rule programming* suitable for applications in general software engineering [11].

One of the principal ideas around the XTT method, is the *constant visual support* of the design, on both the conceptual and logical level. In order to provide practical design and implementation support for XTT, specialized computer tools need to be developed, including graphical editors, as well as knowledge base translators. Using the well established *Model View Controller* design pattern, these tool should use the XTT knowledge base as the *model* of the design, whereas the visual representation, or the corresponding Prolog form, as well as the XML encoding can be treated as the *views* of the knowledge base. This concept is presented in Fig. 1.

Using this approach several tools have been developed so far, and the development process continues. In the development several requirements have been taken into consideration:

- a user-friendly interface, following some common guidelines,
- a possible use of existing structural editors,
- use of common system-independent run-time platforms.

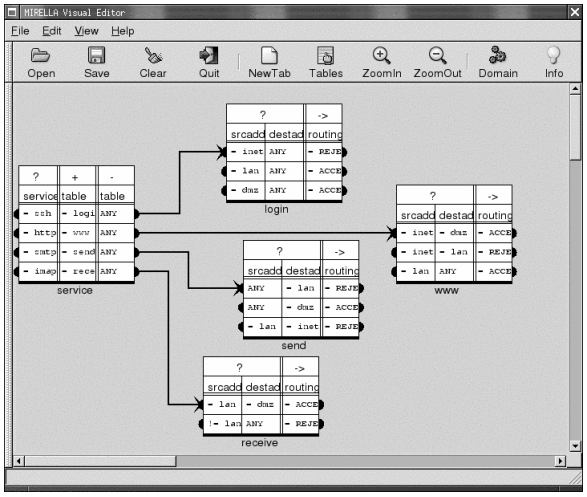

Figure 2. The Mirella Designer.

The first prototype tool was the Mirella designer presented in the next section. Recently, several new tools have been developed in the AGH Institute of Automatics. They are introduced in the following sections.

3. Mirella Designer Prototype Mirella Designer was the first tool developed for the XTT [9]. Basic goals of this editor were to provide the visual design for XTT, as well as the Prolog translation. The knowledge base was encoded in the XTTML format, using XML. The editor itself was implemented in ANSI C language, using the multiplatform GTK+ library for the GUI, with the GNOME canvas library for the structured graphics edition. A screenshot of the editor is presented in Fig. 2.

While successful, Mirella remained a prototype. It had some important limitations, such as portability and scalability. It didn't support the conceptual design phase with ARD, and new XTT extensions. Since Mirella was hard to maintain, a decision has been made to develop new tools, that would possibly support toolkits superior to GTK+. A possibility of using diagram editors has also been evaluated.

4. Dia-based ARD Design *Dia* (`www.gnome.org/projects/dia`) is a universal diagram drawing software for the GNOME desktop environment. Dia uses the GNOME stack, including

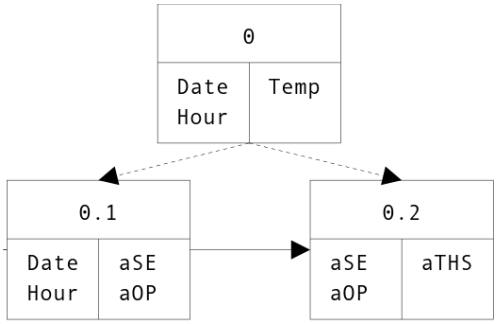

Figure 3. ARD design in Dia.

the GTK+ library, and works on any Unix-like environment, with version for MS Windows also available. It is a free-software project, available under the GNU GPL license.

The idea behind Dia is to support an advanced canvas for structured graphics, on top of which predefined object sets can be used. Thanks to this approach the editor could be possibly extended with any new diagrams. Currently Dia supports over 30 different diagrams types, from simple ones, including electrical and logic circuits, to more complex, including BPMN and UML. Every diagram set is described using XML files, with every shape available in the diagram, having a separate XML description. The description uses a predefined Dia format, that also includes SVG integration.

A student project in the Knowledge Engineering Methods class 2007 in Institute of Automatics has been completed [5]. The goal of the project was to develop a Dia library to support ARD diagrams. In this way, ARD diagrams could be drawn using a standard editor, without a need of developing a dedicated one. The library has been successfully implemented. A screenshot of a diagram drawn with Dia using the ARD library can be observed in Fig. 3.

While implementing the ARD module for Dia, some major limitations of the editor have been exposed. It turns out, that in case of a more advanced diagrams, the XML shape description is insufficient. In order to allow a more complex diagram modelling and labeling, the XML shape description needs to be accompanied by a C-based executable plugin that manages these futures. Implementing such plugins for Dia while possible thanks to number of examples, is not trivial due to poor design and developer documentation of the editor. It is also a non-transparent method, that mixes a clear XML representation with an executable code.

The limitations of Dia made the implementation of ARD editing difficult. They also showed, that building an XTT editor in a similar way would not be possible at all, due to a larger complexity of XTT diagrams. So, a decision has been made to abandon Dia, and implement a custom ARD and possibly XTT editor.

5. Qt ARD Editor To implement an ARD editor a popular, and well documented developer toolkit has been selected. A student project in the Knowledge Engineering Methods class in Institute of Automatics has been completed [3] in 2006. The goal of the project was to develop a standalone ARD diagram editor (called *Qrde*) using the portable Qt library. Qt (`www.trolltech.com`) is a multiplatform GUI library for Unix and Windows systems, implemented in C++. Currently it also supports number of additional classes for data manipulation, XML parsing, etc. It has excellent commercial support and documentation. It is available on both commercial, and open-source licenses.

The project [3] includes requirements specification for the editor, as well as a simple design in UML in the MVC paradigm. The diagram for the ARD model can be observed in Fig. 4. Using Qt library, a prototype editor has been successfully implemented. The view and the controller part have been implemented for the Qt canvas widget, and are described in detail in [3].

A screenshot of the editor can be observed in Fig. 5. It allows for hierarchical ARD design, and simple attribute specification. The canvas support diagram scaling, and XML export. A simple ATTML format has been provided. The XML representation of the level 0 diagram presented in Fig. 5 is as follows:

```
<?xml version="1.0">
<diagram>
 <table x="140" y="30">
  <header>0</header>
```

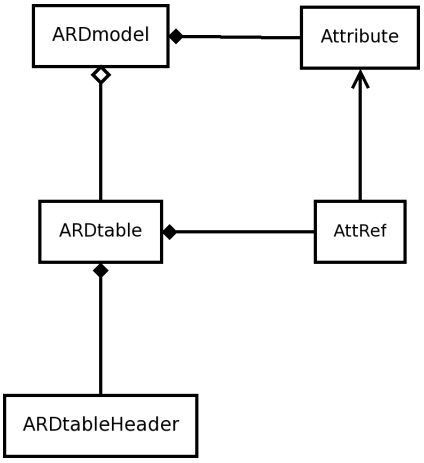

Figure 4. ARD model in UML.

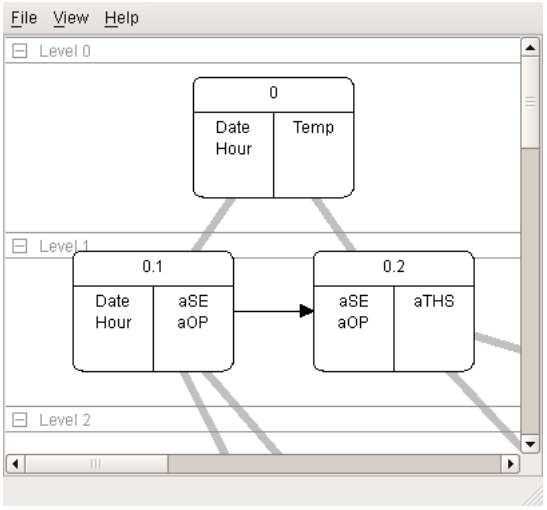

Figure 5. ARD model design.

```
 <inputAttributes>
  <attribute>Date</attribute>
  <attribute>Hour</attribute>
 </inputAttributes>
 <outputAttributes>
  <attribute>Temp</attribute>
 </outputAttributes>
</table>
```

Implementing a custom diagram editor with Qt proved to simple and straight forward. The new, 4th version of the library, released in late 2006, brought major improvements to the toolkit including a better canvas, and XML support, as well as direct export to the SVG, an XML-based vector graphics format. So a decision has been made to continue the development of the XTT editor with the Qt library, using the advanced Qt features.

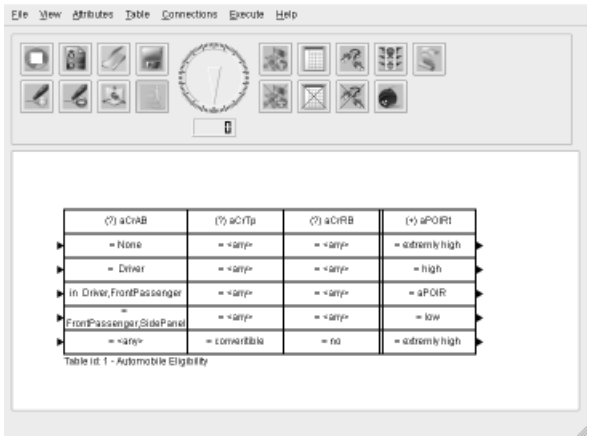

Figure 6. Hekate Qt Editor.

6. HQED, a Hekate Editor In 2006, the HeKatE project [12] has been started, see `hekate.ia.agh.edu.pl`. The main goal of the project is to extend the ARD/XTT design and representation, to support general software engineering. An advanced design and runtime environment based on the MVC approached is to be provided. In this case the model would be described with the use XTT, in a purely declarative way, whereas the view and the controller parts could be developed in sequential or object-oriented languages, e.g. Java. The advanced runtime would allow for an efficient integration of these solutions.

Basing on the experiences with the *Qrde* project, another student project in the Knowledge Engineering Methods class in Institute of Automatics has been completed [2] in 2007. The goal of the first phase of the project was to develop a prototype standalone XTT diagram editor using the Qt library. Some of the requirements where not just to support the visual XTT design, but also provide an improved XML representation of the diagrams, as well as the SVG export. The first phase has been successfully passed. A screenshot of the *Hqed* (Hekate Qt EDitor) can be observed in Fig. 6.

The editor provides an improved XTT description in XML. A part of the diagram from the Fig. 6 can be observed below:

```
<XTTML version="2.0">
 <xtt_list_table>
  <xtt_table table_type="middle"
```

```
    name="Automobile Eligibility">
<description>Automobile
  Eligibility Rules</description>
 <row label="">
  <cell content="= None"/>
  <cell content="= <any>"/>
  <cell content="= <any>"/>
  <cell content="= extremly high"/>
 </row>
 <context_size conditional="3"
  assert="1" retract="0" action="0"/>
 <column_attributes>
  <ca>aCrAB</ca>
  <ca>aCrTp</ca>
  <ca>aCrRB</ca>
  <ca>aPOIRt</ca>
 </column_attributes>
```

In the future, this format could serve as a bridge between XTT and other rule design and exchange methods A translation to the *R2ML*, a new and powerful rule markup format from the Working Group I1 of the *REWERSE* project (see `www.rewerse.net`) is planned.

The second phase of the development is conducted by the same author in 2007/8 as a masters thesis. The goal of this phase is to extend the XTT editor, and integrate it with the ARD editor. So far a preliminary design has been carried out, and data exchange XML format has been formulated. In the future, the XTT export to Prolog will also be possible. Ultimately Hqed should fully replace the Mirella prototype, while providing superior quality and features.

7. Future Work This paper presents the evolution of XTT design tools from the early prototype, to the current implementations. The research is a work in progress, concerning both the ARD and XTT methods as well as the supporting tools. First of all, there is an active research towards use cases and refinement of the design process and both representation methods. The ARD design method needs to be reworked to provide a clear integration with XTT. The XTT itself is being extended towards supporting general software engineering via an extended semantics. This work is being conducted within the HeKatE project.

On the tool level two other aspects are currently investigated. The first one consists in using a well know *Eclipse* IDE (`www.eclipse.org`) as a foundation of the visual editor. Some preliminary research has already carried out within this thread. The so-called *Eclipse Modelling Framework* is a modeling framework and code generation facility for building tools and other applications based on a structured data model specification described in XMI. It is based on the MVC paradigm, and it provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. However, it seems, that using the predefined UML-related Eclipse modeling facilities may be difficult to apply to the ARD/XTT design method.

Another area of the future research involves a „UML serialization" of the XTT model. In this case, the model would be mapped to a UML-based model, that could be then processed, and edited as other UML models. This solution opens up possibilities of using standard tools. On the other hand it largely restricts editing features, since UML syntax and semantics is vastly different from this of the XTT.

One of the areas of applications of XTT models, is using them to provide rule bases for some other tools and environments. One of such environments is *Drools* (now *JBoss Rules*, see `http://labs.jboss.com/drools/`). It is a rule engine based on the classic Rete algorithm, integrated with the well-known *JBoss* application server for Java. It is becoming an important open-source rule engine, with important commercial applications. However, it lacks good visual design tools. It is possible, that XTT could provide visual rule design for this platform.

Since the design methods evolve, so are the supporting tools. Currently the main development effort is focused on the Hqed editor itself. It is a foundation of the future *HaDEs*, that is the HeKatE Design environment, providing a unified design environments for the integrated ARD and XTT methods, with XML and Prolog support.

References

[1] Peter Jackson. *Introduction to Expert Systems*. Addison–Wesley, 3rd edition, 1999. ISBN 0-201-87686-8.

[2] Krzysztof Kaczor. Metody inżynierii wiedzy – projekt: Mirella_xtted. AGH UST, 2007.

[3] Tadeusz Kania and Tomasz Szypenbejl. Metody inżynierii wiedzy – projekt: Edytor diagram ow ard. AGH UST, 2006. Supervised by G. J. Nalepa, Ph. D.

[4] Antoni Ligęza. *Logical Foundations for Rule-Based Systems*. Springer-Verlag, Berlin, Heidelberg, 2006.

[5] Dariusz Marynowski. Metody inżynierii wiedzy – projekt: Mirella_diaard. AGH UST. Supervised by G. J. Nalepa, Ph. D.

[6] Dennsi Merrit. Best practices for rule-based application development. *Microsoft Architects JOURNAL*, 1, 2004.

[7] Grzegorz J. Nalepa and Antoni Ligęza. Conceptual modelling and automated implementation of rule-based systems. In Tomasz Szmuc Krzysztof Zieliński, editor, *Software engineering : evolution and emerging technologies*, volume 130 of *Frontiers in Artificial Intelligence and Applications*, pages 330–340, Amsterdam, 2005. IOS Press.

[8] Grzegorz J. Nalepa and Antoni Ligęza. A graphical tabular model for rule-based logic programming and verification. *Systems Science*, 31(2):89–95, 2005.

[9] Grzegorz J. Nalepa and Antoni Ligęza. A visual edition tool for design and verification of knowledge in rule-based systems. *Systems Science*, 31(3):103–109, 2005.

[10] Grzegorz J. Nalepa and Antoni Ligęza. Prolog-based analysis of tabular rule-based systems with the xtt approach. In Geoffrey C. J. Sutcliffe and Randy G. Goebel, editors, *FLAIRS 2006: proceedings of the 19th international Florida Artificial Intelligence Research Society conference*, pages 426–431. AAAI Press, 2006.

[11] Grzegorz J. Nalepa and Igor Wojnicki. Proposal of generalized rule programming model. In Joachim Baumeister and Dietmar Seipel, editors, *KESE'07 3rd Workshop on Knowledge Engineering and Software Engineering*. Department of Computer Science, University of Würzburg, Germany, 2007.

[12] Grzegorz J. Nalepa and Igor Wojnicki. A proposal of hybrid knowledge engineering and refinement approach. In Geoffrey C. J. Sutcliffe and Randy G. Goebel, editors, *FLAIRS 2007 : proceedings of the 20th international Florida Artificial Intelligence Research Society conference*. AAAI Press, 2007. to be published.