HaDEs – Presentation of the HeKatE Design Environment*

Krzysztof Kaczor and Grzegorz J. Nalepa

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland
kk@agh.edu.pl, gjn@agh.edu.pl

Abstract. TOOL PRESENTATION: The paper introduces the HeKatE design environment called HaDEs. The HeKatE project aims at delivering new knowledge representation methods for rule-based systems. Principal ideas include an integrated hierarchical design process covering stages from conceptual, through logical to physical design. These stages are supported by specific knowledge representation methods: ARD+, XTT^2, and HMR. The whole design process is supported by a number of tools, namely: VARDA and HJEd in the ARD+conceptual design stage and rule prototyping, HQEd for the XTT^2 logical design and finally HeaRT, the rule runtime environment. The goal of this tool presentation is to introduce the design process using a practical example.

1 Introduction

Practical design methodologies for intelligent systems remain a field of active development. Developing such a methodology requires an integration of accurate knowledge representation and processing methods [1], as well as practical tools supporting them. Some of the important features of such approaches are: scalable visual design, automatic code generation, support for existing programming frameworks. At the same time quality issues, as well as a formalized description of the designed systems should be considered.

The *HeKatE* project (see `hekate.ia.agh.edu.pl`) aims at providing an integrated methodology for the design, implementation, and analysis of rule-based systems [2,3]. An important goal of the project is to allow for an easy integration of knowledge and software engineering methods and approaches, thus providing a *Hybrid Knowledge Engineering* methodology. The project delivers several new knowledge representation methods, as well as a set of practical tools supporting the whole design process.

This paper provides a short overview of the project including its main objectives and tools in Sect. 2. Then in Sect. 3 the HeKatE design toolchain called HaDEs is introduced. The paper accompanies a tool presentation given at the KESE 2009 workshop.

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

2 HeKatE Project Overview

2.1 Research Objectives

The main principles of the HeKatE project are based on a critical analysis of the state-of-the art of the rule-based systems design, see [4].

Formal Language for Knowledge Representation. It should have a precise definition of syntax, properties and inference rules. This is crucial for determining its expressive power, and solving formal analysis issues.

Internal Knowledge Base Structure. Rules working within a specific context, are grouped together and form the extended decision tables. These tables are linked together forming a partially ordered graph structure which encodes the flow of inference.

Systematic Hierarchical Design Procedure. A complete, well-founded design process that covers all of the main phases of the system lifecycle, from the initial conceptual design, through the logical formulation, all the way to the physical implementation is proposed. A constant verification of the model w.r.t. critical formal properties, such as determinism and completeness is provided.

In the HeKatE approach the control logic is expressed using forward-chaining decision rules. They form an intelligent rule-based controller or simply a business logic core. The controller logic is decomposed into multiple modules represented by attributive decision tables. The emphasis of the methodology is its possible application to a wide range of intelligent controllers. In this context two main areas have been identified in the project: control systems, in the field of intelligent control, and business rules [5] and business intelligence systems, in the field of software engineering. In the case of the first area the meaning of the term "controller" is straightforward. In the second area the term denotes a well isolated software component implementing the application logic, or logical model.

2.2 Main Methods

HeKatE introduces a formalized language for rule representation [4]. Instead of simple propositional formulas, the language uses expressions in the so-called *attributive logic* [3]. This calculus has a stronger expressiveness than the propositional logic, while providing tractable inference procedures for extended decision tables. The current version of the rule language is called XTT^2 [6]. The current version of the logic, adopted for the XTT^2 language, is called ALSV(FD) (*Attributive Logic with Set Values over Finite Domains*).

Based on the logic, a rule language called XTT is provided [7,6]. XTT stands for *eXtended Tabular Trees*. The language is focused not only on providing an extended syntax for single rules, but also allows for an explicit structurization of the rule base. XTT introduces explicit inference control solutions, allowing for a fine grained and more optimized rule inference than in the classic Rete-like solutions. The representation has a compact and transparent visual representation suitable for visual editors.

HeKatE also provides a complete hierarchical design process for the creation of the XTT-based rules. The main phase of the XTT rule design is called the *logical design*. The logical rule design process may be supported by a preceding *conceptual design* phase. In this phase the rule prototypes are built with the use of the so-called *Attribute Relationship Diagrams*. The ARD method has been introduced in [8], and later refined in [3]. The principal idea is to build a graph, modelling functional dependencies between attributes on which the XTT rules are built. The version used in HeKatE is called ARD+ as discussed in [9]. The practical implementation on the XTT rule base is performed in the physical design phase. In this stage the visual XTT model is transformed into an algebraic presentation syntax called HMR. A custom inference engine, HeaRT runs the XTT model described in HMR.

The complete framework including the discussed methods and tools is depicted in Fig. 1.

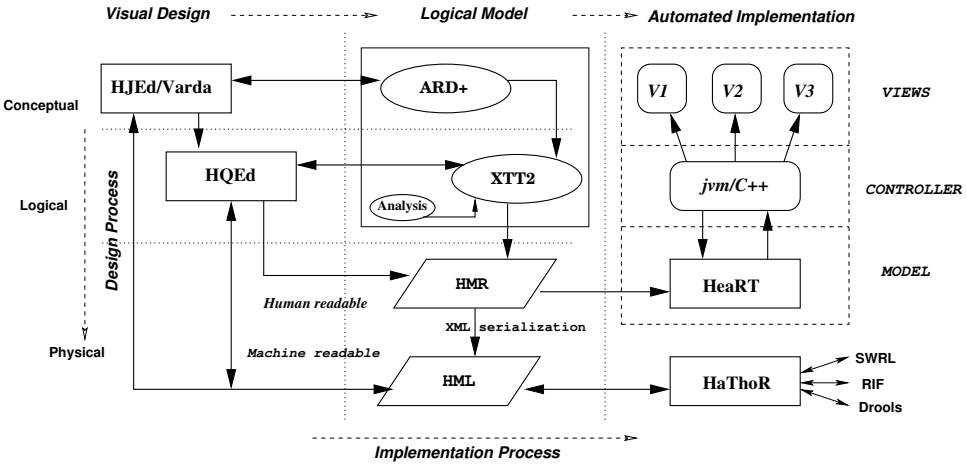

Fig. 1. The complete design and runtime framework

3 HaDEs Design Toolchain

The HeKatE design process is supported by a number of tools supporting the visual design and the automated implementation of rule-based systems (see `https://ai.ia.agh.edu.pl/wiki/hekate:hades`).

HJEd visual editor supports the ARD+ design process. It is a cross-platform tool implemented in Java. Its main features include the ARD+ diagram creation with on-line design history available through the TPH diagram. An example of a design capturing functional dependencies between system attributes is shown in Fig. 2. It is a medical diagnosis system. The diagram on the left shows the dependencies between rule attributes, whereas the right one captures the design

history. Once created, the ARD+ model can be saved in a XML-based HML (HeKatE Markup Language) file. The file can be then imported by the HQEd design tools supporting the logical design.

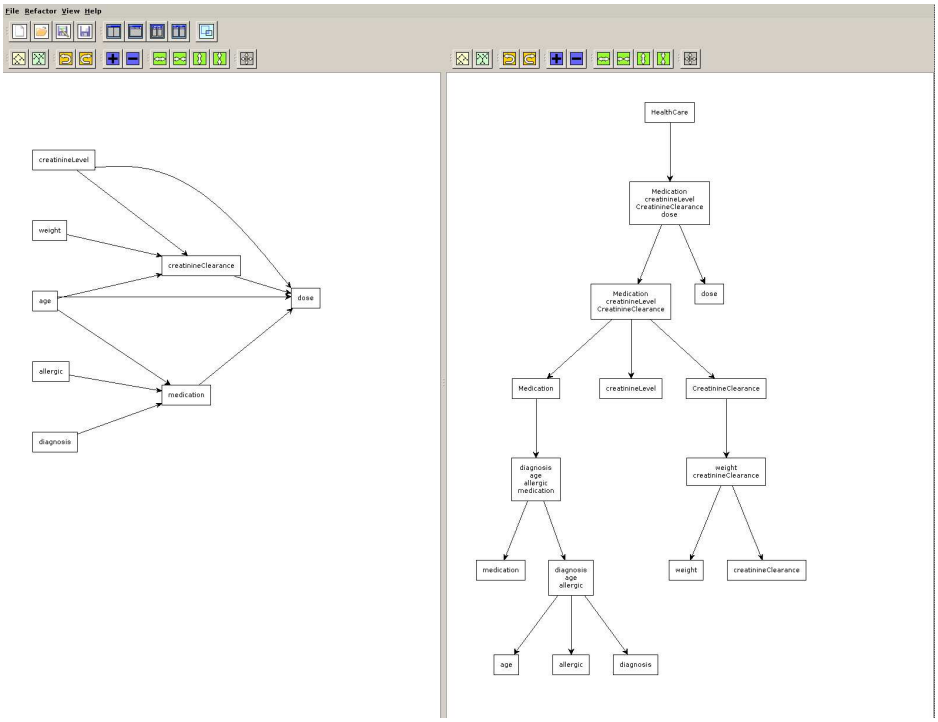

Fig. 2. ARD+ design in HJEd

VARDA is a prototype semi-visual editor for the ARD+ diagrams implemented in Prolog, with an on-line model visualization with Graphviz. The tool also supports prototyping of the XTT model, where table headers including a default inference structure are created, see Fig. 3. In this case three tables are generated. The ARD+ design is described in Prolog, and the resulting model can be stored in HML.

HQEd provides support for the logical design with XTT, see Fig. 4. In the figure some additional decision tables to input attribute values are present. It is able to import a HML file with the ARD+ model and generate the XTT prototype. It is also possible to import the prototype generated by VARDA. HQEd allows to edit the XTT structure with on-line support for syntax checking on the table level. Attribute values entered are checked against their domains and a number of possible anomalies is eliminated.

The editor is integrated with a custom inference engine for XTT^2 called HeaRT. The role of the engine is twofold: run the rule logic designed with the

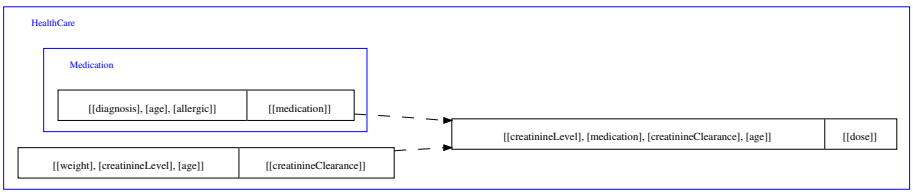

Fig. 3. XTT model generation in VARDA

use of the editor, as well as provide constant formal analysis of the rulebase. The communication uses a custom TCP-based protocol.

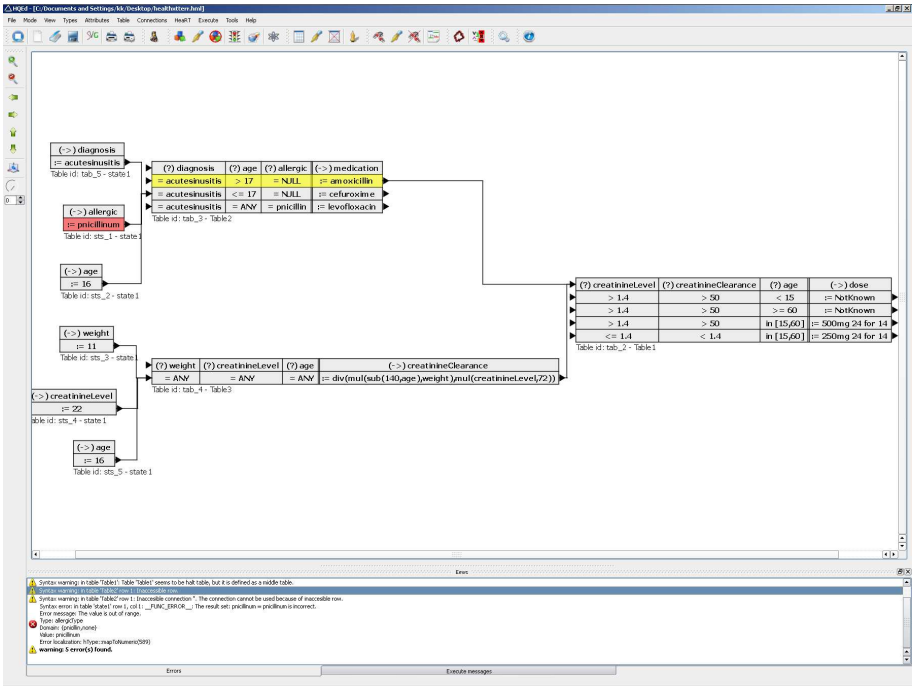

Fig. 4. XTT model edited in HQEd with anomalies detected

HeaRT (HeKatE Run Time) is a dedicated inference engine for the XTT^2 rule bases. It is implemented in Prolog in order to directly interpret the HMR representation which is generated by HQEd. HMR (HeKatE Meta Representation) is a textual representation of the XTT^2 logic designed by HQEd. It is a human readable form, as opposed to the machine readable HML format. The HeaRT engine implements the inference based on the ALSV(FD) logic [6,4].

HalVA (HeKatE Verification and Analysis) is a modularized verification framework provided by HeaRT. So far several plugins are available, including completeness, determinism and redundancy checks. The plugins can be run form the interpreter or from HQEd using the communication protocol.

4 Conclusions

The paper shortly introduces the main concepts of the HeKatE project, its methods and tools. The main motivation behind the project is to speed up and simplify the rule-based systems design process, while assuring the formal quality of the model. The HeKatE design process is supported by the HeKatE design environment called HaDEs. During the presentation given at the KESE workshop the tools were presented using practical examples.

References

1. van Harmelen, F., Lifschitz, V., Porter, B., eds.: Handbook of Knowledge Representation. Elsevier Science (2007)
2. Giarratano, J., Riley, G.: Expert Systems. Principles and Programming. Fourth edition edn. Thomson Course Technology, Boston, MA, United States (2005) ISBN 0-534-38447-1.
3. Ligęza, A.: Logical Foundations for Rule-Based Systems. Springer-Verlag, Berlin, Heidelberg (2006)
4. Nalepa, G.J., Ligęza, A.: Hekate methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science (2009) accepted for publication.
5. Ross, R.G.: Principles of the Business Rule Approach. 1 edn. Addison-Wesley Professional (2003)
6. Nalepa, G.J., Ligęza, A.: Xtt+ rule design using the alsv(fd). In Giurca, A., Analyti, A., Wagner, G., eds.: ECAI 2008: 18th European Conference on Artificial Intelligence: 2nd East European Workshop on Rule-based applications, RuleApps2008: Patras, 22 July 2008, Patras, University of Patras (2008) 11–15
7. Nalepa, G.J., Ligęza, A.: A graphical tabular model for rule-based logic programming and verification. Systems Science **31**(2) (2005) 89–95
8. Nalepa, G.J., Ligęza, A.: Conceptual modelling and automated implementation of rule-based systems. In: Software engineering : evolution and emerging technologies. Volume 130 of Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam (2005) 330–340
9. Nalepa, G.J., Wojnicki, I.: Towards formalization of ARD+ conceptual design and refinement method. In Wilson, D.C., Lane, H.C., eds.: FLAIRS-21: Proceedings of the twenty-first international Florida Artificial Intelligence Research Society conference: 15–17 may 2008, Coconut Grove, Florida, USA, Menlo Park, California, AAAI Press (2008) 353–358