PRACE NAUKOWE UNIWERSYTETU EKONOMICZNEGO WE WROCŁAWIU nr 133
Pozyskiwanie wiedzy i zarządzanie wiedzą 2010

Antoni Ligęza, Grzegorz J. Nalepa
University of Science and Technology, Kraków

HEKATE RULE DESIGN METHODOLOGY OVERVIEW

Summary: This paper presents a methodology and tools for design and development of rule-based systems. Efficient knowledge representation is based on the Extended Tabular Trees – XTT2 formalism. The design process is a top-down one. A conceptual model is developed with the ARD+ diagrams representing functional dependencies among attributes. On the basis of ARD+ a scheme of the XTT2 rule-based model is generated and filled with knowledge. A set of tools supporting design and verification is described and design methodology is outlined.

Keywords: rule-based systems, design and development of rule-based systems.

1. Introduction

Rule-based systems constitute one of the most powerful and most popular knowledge representation formalisms [3; 5]. They offer a relatively easy way of knowledge encoding and interpretation. Formalization of knowledge within a rule-based system can be based on mathematical logic (e.g. propositional, attributive, first-order, or even higher order ones) or performed on the basis of an engineering intuition. They have found numerous applications in various domains of engineering, science, law, and computer science. Rule-based systems are also a prominent example of successful application of Artificial Intelligence [16].

Rules are omnipresent in our everyday life, professional work, leisure and sport activities; they are also popular in science and technology. They are result of physical (and mathematical) laws, formed by men, tradition, culture, civilization, law. The most precise rules are the ones found in technical and technological systems, however, many rule-based expert systems addressed the issues of medical diagnosis or business decision support as well.

In computer science rule-based systems appeared just after symbolic programming languages had been developed. First such systems were termed production systems or production rules. The golden age for rules came in the 1970s and 1980s with developments and practical applications of expert systems. They were dedi-

cated to solving specific problems in narrow, well-formalized domains. A typical construction of such a system was based on two components: a declarative rule-base encoding domain-specific knowledge and an inference engine of general purpose (the so-called expert system shell [5; 6]). Rule-based systems are also applied in Decision Support Systems. In business the high-level knowledge encoding procedures, experience, domain knowledge and preferences is often encoded by various forms of rule-based systems, decision tables, and decision trees.

Modern rule-based systems, such as Clips, Jess, Drools, Xpert, Aion, Ilog, G2, follow the classical paradigm. Rules are developed using some predefined knowledge representation framework (often close to attribute logic). The current rule-based systems and tools for their development have reached a certain level of maturity. Specialized editors that are able to check the syntax of rules are in use and they provide tools for computer-aided development of final applications. However, they have inherited a number of traditional features of classical rule-based systems, which nowadays can be considered as drawbacks.

In this paper we mainly address the following issues which seem worth improvement: 1) single rules constitute items of low knowledge processing capabilities, while for practical applications a higher level of abstraction is desirable, 2) inference engines, especially ones forward-checking, are highly inefficient with respect to the focus on the goal to be achieved, (in a certain sense, they perform a blind search), 3) no practical methodology acceptable by engineers and assuring quality of rules is available.

This paper presents a new approach to design and development of rule-based systems, based on the ideas presented in [7; 11]. More precisely, we present the state-of-the-art of the HeKatE methodology (an acronym for Hybrid Knowledge Engineering, see http://hekate.ia.agh.edu.pl), for design and development of complex rule-based systems for control and decision support. This methodology is supported with visual tools for development of knowledge bases and novel inference engine.

The main paradigm for rule representation, namely the eXtended Tabular Trees2 (XTT2) [9; 10; 14], ensures high density and transparency of visual knowledge representation. Contrary to traditional, flat rule-based systems, the XTT approach is focused on groups of similar rules rather than single rules. In this way we address the first issue of low processing capabilities of single rules. Such groups form decision tables which are connected into a network for inference.

Efficient inference is assured thanks to firing only rules necessary for achieving the goal. It is achieved by selecting the desired output tables and identifying the tables necessary to be fired first. The links representing the partial order assure that when passing from a table to another one, the latter can be fired since the former one prepares an appropriate context knowledge. Hence only rules working in the current context of inference are explored. The partial order among tables allows to avoid examining rules which should be fired later on.

Another distinctive feature is the design methodology allowing for formal verification of rules. A top-down design methodology based on successive refinement of the project is put forward. It starts with development of an Attribute Relationship Diagrams+ (ARD) which describes relationships among process variables. Basing on the ARD the scheme of particular tables and links among them is generated. The tables are filled with expert-provided definitions of constraints over the values of attributes; these are in fact the rule preconditions. The code for rules representation is generated and interpreted with provided inference engine [12].

The rest of the paper is organized as follows: In the second section the perspective on the rule-based decision support is given, to present some important issues addressed in this paper. This gives a background for the motivation for the research given next. The HeKatE project, introduced in the subsequent sections aims at providing solutions for the problems identified in the motivation section. One of the main goals of the project is to provide a new rule-based inference engine solution assuring flexible and efficient control during the inference process. The practical design of the XTT2 knowledge bases is supported by visual editors, and other tools shortly presented later on. Concluding remarks are given in the final section.

2. Rule-based control and decision support

Rule-based systems are widely applied for operational knowledge representation in business and technical applications. Technical applications include control, monitoring and diagnosis support. Business applications include decision support, preferences modelling, recommendations systems, and many other applications. There exist many tools for implementing the RBS in the form of so-called shell-systems.

An important step was the development of CLIPS (an acronym for C Language Integrated Production System), a descendant of the OPS5 rule-based production system written in Lisp. Clips was developed by NASA and, despite being developed in C, it follows the Lisp-like style of knowledge specification (full of parentheses and hardly readable for engineers). It has become perhaps one of the most popular rule-based engines since it is relatively fast (employing the Rete algorithm), simple, and free (now in the public domain). A more recent reincarnation of CLIPS is JESS (Java Expert System Shell [1]), developed in Java, but still employing a bit ancient Lisp-like style of rules encoding. Another example is Drools [15], also based on Rete. Yet another example is the Sphinx system [17]. Wider application of the technology of rule-based systems in the domain of automatic control started in the early 1980s, as soon as the technology itself has reached some maturity.

Some overview of current developments with respect to theory, knowledge representation, tools and application areas is provided in [5]. A list of current tools is enclosed at the back of [6]. A very recent book [2] gives a good overview of some emerging technologies and tools in the area of rule-based solutions.

3. Motivation

Certain persistent limitations of the existing approaches to the design of rule-based intelligent control systems exist. They are especially visible in the case of designing of complex systems. They often make the high quality design as well as the refinement of such systems very difficult. These limitations are related to the three aspects of rule design: 1) knowledge representation, where the lack of formally described rule base often makes the semantics unclear, 2) transparent structure and efficient inference in the rule base, where the focus is on the design of single rules, with no structure explicitly identified by the user, and 3) well founded systematic and complete design process, that preserves the quality aspects of the rule model, and allows for gradual system design and automated implementation of rules.

The majority of rule-based systems work according to the classical principles of forward chaining. They incorporate a relatively simple, blind inference engine. In order to speed up the interpretation of rules they often employ some indexation of changes that occur in the fact base and how they influence the rule preconditions satisfaction, e.g. the Rete network. With respect to the knowledge representation language being used the following issues may be raised: lack of formal relation of the knowledge representation language to classical logic, and, as a consequence, difficulties with understanding the expressive power of the language, and, as a consequence, lack of knowledge portability.

With respect to the internal structure and inference mechanism the criticism may be even stronger: typically, the set of rules is flat (it has no internal structure, so hundreds of different rules are considered equally important, and equally unrelated), the inference engine (at least potentially) tries to examine all rules in turn for firing within every cycle, there is no definite way to select which rules from the conflict set should be fired, irrelevant rules can be fired even if they do not contribute to the problem solution.

With respect to the development methodology it should be pointed out that most of the rule-based tools are just shells providing a rule interpreter and sometimes an editor. Hence: the knowledge acquisition task constitutes a bottleneck and is arguably the weakest point in the design and development process, typical rule shells are not equipped with consistent methodology and tools for efficient development of the rule base; on the other hand, general methodologies such as KADS or Common-KADS are too complex for practical engineering or business applications, verification of the knowledge is rarely supported, and, if supported verification is performed only after the design of rules.

Three main principles following from the above analysis define the foundations of the approach advocated here are:

– Formal Language Definition, we insist on a precise definition of a formal language, its formal properties and inference rules. This is crucial for determining

the expressive power, definition of inference mechanism and solving verification issues;
- Internal Knowledge Structure, we introduce internal structure of the rule base. Similar rules, aimed at working within a specific context, are grouped together and form the XTT2 tables;
- Systematic Design Procedure, we argue for 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.

We emphasize the need for a constant on-line verification of the system model w.r.t. critical formal properties, such as determinism and completeness.

These principles served as guidelines for the HeKatE approach we introduce in the next section.

4. HeKatE concepts

In order to deal with the shortcoming of current systems the presented methodology is based on the following assumptions.

Support for the visual design methods. The traditional approach to the development of expert systems consisted in providing the so-called shells. Shells offer a rule language and a ready-to-use inference engine. Classic solutions, such as CLIPS, do not offer any special support for the design of the rule base. The new generation of tools focuses on the design phase, and provides some advanced visual tools that support the designer and speed up the process.

Markup language-based representation. In the wake of large number of tools, the problem of data exchange has gained an increased importance. The most common solution to overcome it is to provide an XML-based knowledge encoding. There are even some dedicated XML-based languages, such as RuleML.

Strong integration with existing software tools and environments. Today expert systems often work as a knowledge-based component integrated into a large software. This is why the new expert system development tools provide interfaces to some common programming platforms, such as Java.

Metaprogramming and code generation. Integrating the knowledge engineering approach with classic programming paradigms still poses some difficult problems. One of the techniques used is the approach, where the code in a programming language, such as Java, is automatically generated from the knowledge-based model.

Formal knowledge verification and evaluation. Several more advanced tools not only lead the designer through the design, but also support formal means to analyze and verify the knowledge base. Such a verification should be performed as soon as possible during the design. Assuring some key formal system properties, such completeness can lead to increased system performance and security.

Reuse of old solutions, ideas and tools. What is striking in most of the technologically advanced tools, is the general lack of new concepts and approaches. Most of the tools rely on some well-known, and often exploited techniques, such as the Rete algorithm when it comes to the rule processing, and simple decision trees support in the design. Few tools support the analysis of the rule base.

Following are the results of the survey of the selected tools. Jess can be considered a modern Java-based implementation of the classic CLIPS system. Drools attempts to create rule-based programming platforms for a number of languages. Basing on the well tested Prolog technology, VisiRules provides a visual support for the system design. Experimental tools, KbBuilder and MirellaDesigner, provide some advanced analysis and verification capabilities, while supporting integrated visual design.

5. HeKatE methods

The goals of the *Hybrid Knowledge Engineering* (HeKatE) methodology fulfil the principles described previously. In this approach a system is controlled by an intelligent rule-based controller. The control logic is expressed using forward-chaining decision rules. The controller logic is decomposed into multiple modules represented by attributive decision tables (the so called object-attribute-value tables) The controller is designed using the HeKatE methodology.

The goals of the Hybrid Knowledge Engineering (HeKatE) methodology are to provide:

– an expressive formal logical calculus for rules, allowing for formalized inference and analysis,
– a structured visual rule representation method with formally described syntax and semantics, based on decision tables, and
– a complete hierarchical design process based on the above, with an effective on-line verification methods as well as automated implementation facilities.

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 and business intelligence systems [8], in the field of software engineering [18].

In the case of the first area the meaning of the term “controller” is straightforward, and falls into the area discussed in this paper as intelligent control. In the case of the second area the term denotes a well isolated software component implementing the so-called application logic, or logical model. In the case of business software this component is commonly integrated using a dedicated architectural pattern, such as the Model-View-Controller [18].

In the HeKatE project a formalized language for an extended rule representation is introduced. Instead of simple propositional formulas, the language uses ex-

pressions in the so-called attributive logic [6]. This calculus has a stronger expressivity than the propositional logic, while providing tractable inference procedures for extended decision tables [13]. The current version of the rule language is called XTT2 [14]. The current version of the logic, adopted for the XTT2 language, is called ALSV(FD) (Attributive Logic with Set Values over Finite Domains).

Based on the attributive logic a rule language called XTT is provided [10; 14]. XTT stands for eXtended Tabular Trees, since 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. This solution allows for identifying system contexts during the rule base design. XTT introduces explicit inference control solutions, allowing for a fine grained and more optimized rule inference than in the classic Rete-like solutions. XTT has been introduced with the visual design support in mind. The representation has a compact and transparent visual representation suitable for visual editors.

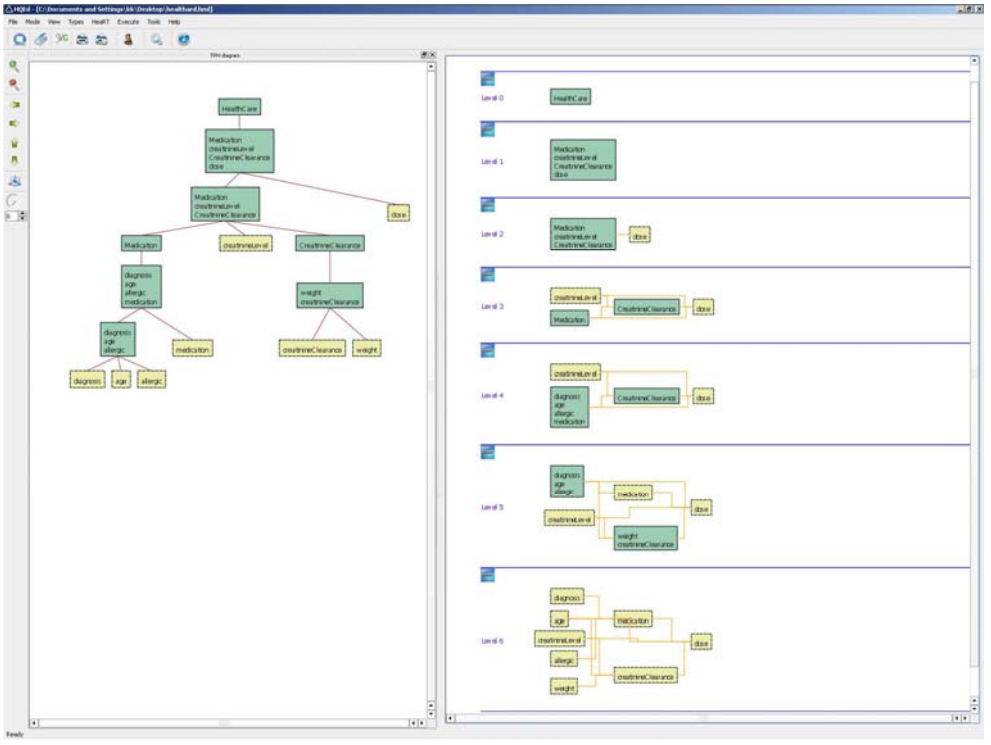

Figure 1. ARD+ conceptual design of a drug dosage system

Source: own research.

The HeKatE project also provides a complete hierarchical design process for the creation of the XTT-based rules. The process is based on the ideas originally introduced in [9]. The main phase of the XTT rule design is called the logical design. This phase is supported by a CASE tool called HQEd [4]. 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 [12], and later refined in [6]. 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+. The ARD+ design is supported by two visual tools, VARDA and Hjed. The practical implementation on the XTT rule base is performed in the physical design phase. In this stage the visual model built with HQed is transformed into an algebraic presentation syntax. A custom inference engine then can run the XTT model. Design tools are described in the next section.

6. HeKatE tools

The HeKatE design process is supported by a number of tools supporting the visual design and the automated implementation of rule-based systems.

The ARD+ design process is supported by the HJEd visual editor. 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. 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. VARDA is a prototype semivisual 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. 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. It is able to import a HML file with the ARD+ model, visualize it 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 XTT2 called HeaRT. The role of the engine is twofold: run the rule logic designed with the use of the editor, as well as provide on-line formal analysis of the rule base. The communication uses a custom TCP-based protocol. An example of an ARD+ design capturing functional dependencies between system attributes is shown in Figure 1, with the logical XTT2 design shown in Figure 2.

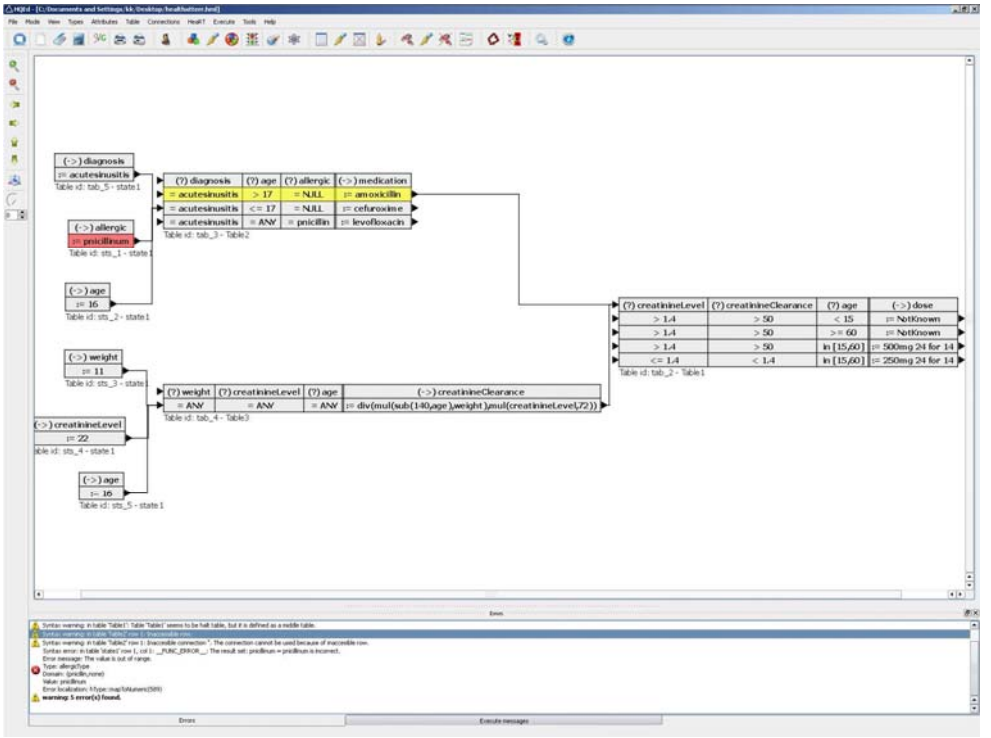

Figure 2. XTT2 design of a drug dosage systems, anomalies detected

Source: own research.

HeKatE Run Time (HeaRT) is a dedicated inference engine for the XTT2 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 XTT2 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. HeaRT also provides a modularized verification framework, also known as HalVA (HeKatE Verification and Analysis). So far several plugins are available, including completeness, determinism and redundancy checks. The plugins can be run from the interpreter or indirectly from the HQEd editor using the communication protocol.

7. Concluding remarks

The paper outlines some important elements of the HEKATE methodology for design and development of the rule-based systems for business and technological applications. The presented tools, namely: visual design support, automatics XTT2

scheme generation and automatic final code generation introduce a new quality in the area of Rule-Based System design and development. In the paper a new approach to the practical development of rule-based expert systems is presented. The approach developed at the Institute of Automatics of AGH UST provides a multi-level hierarchical development process for Prolog-backed rule-based expert systems. It uses new knowledge representation methods, such as XTT (eXtended Tabular Trees) and ARD (Attribute Relationship Diagrams). Using computer tools developed in the HeKatE project it allows for the visual design of the knowledge base with dynamic Prolog-code generation.

References

[1] FRIEDMAN-HILL E. J., *Jess, The Rule Engine for the Java Platform*. Distributed Computing Systems, Sandia National Laboratories, Livermore, CA, Version 6.1p8 (23 March 2005), http://www.jessrules.com/jess/docs/61.

[2] Giurca A., Gasevic D., Taveter K., *Handbook of Research on Emerging Rule-Based Languages and Technologies*: *Open Solutions and Approaches*, Information Science Reference, 2009.

[3] Harmelen F. van, Lifschitz V., Porter B., *Handbook of Knowledge Representation*, Elsevier Science, 2007.

[4] Kaczor K., Nalepa G.J., *Design and Implementation of HQED, the Visual Editor for the XTT+ Rule Design Method* (No. CSLTR02/2008), AGH University of Science and Technology, Kraków 2008.

[5] Liebowitz J., *The Handbook of Applied Expert Systems*, CRC Press, Boca Raton 1998.

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

[7] Ligęza A., Nalepa G.J., (2005). *Visual design and on-line verification of tabular rule-based systems with XTT*, [in:] K.P. Jantke, K.-P. Fuhnrich, W.S. Wittig (eds.), *Marktplatz Internet: Von e-a Learning bis e-Payment* : *13*. *Leipziger Informatik-Tage*, *LIT 2005*, "Lecture Notes in Informatics", Gesellschaft für Informatik, Bonn 2005, pp. 303-312.

[8] Nalepa G.J., *Business rules design and refinement using the XTT approach*, *Paper presented at the FLAIRS-20*: *Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference*, May 7-9, 2007, Key West, Flo., Menlo Park, Cal., 2007.

[9] Nalepa G.J., Ligęza A., *A graphical tabular model for rule-based logic programming and verification*, "Systems Science" 2005, Vol. 31, No. 2, pp. 89-95.

[10] Nalepa G.J., Ligęza A., *HeKatE methodology, hybrid engineering of intelligent systems*, "International Journal of Applied Mathematics and Computer Science" 2009 [accepted for publication].

[11] Nalepa G.J., Ligęza A., *A visual edition tool for design and verification of knowledge in rule-based systems*, "Systems Science" 2005, Vol. 31, No. 3, pp. 103-109.

[12] Nalepa G.J., Ligęza A., *Conceptual modelling and automated implementation of rule-based systems*, [in:] K. Zieliński, T. Szmuc (eds.), *Software Engineering*: *Evolution and Emerging Technologies*, Frontiers in Artificial Intelligence and Applications, Vol. 130, IOS Press, Amsterdam 2005, pp. 330-340.

[13] Nalepa G.J., Ligęza A., *Prolog-based analysis of tabular rule-based systems with the XTT approach*, [in:] G.C.J. Sutcliffe, R.G. Goebel (eds.), *FLAIRS 2006: Proceedings of the Nineteenth*

International Florida Artificial Intelligence Research Society Conference [Melbourne Beach, Florida, May 11-13, 2006], FLAIRS, AAAI Press, Menlo Park, Cal., 2006.

[14] Nalepa G.J., Ligęza A., *XTT+ rule design using the alsv(fd)*, [in:] A. Giurca, A. Analyti, G. Wagner (eds.), *ECAI2008*: *18th European Conference on Artificial Intelligence*: *2nd East European Workshop on Rule-based applications*, RuleApps2008: Patras, 22 July 2008, University of Patras, Patras 2008.

[15] Rupp A.N., *An Introduction to The DROOLS Project.* TheServerSide.COM, Enterprise Java Community, May 2004, http://www.theserverside.com/articles/article.tss?l=Drools.

[16] Russell S., Norvig P., *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 2003.

[17] Simiński R., *Dynamiczna weryfikacja poprawności baz wiedzy w procesie ich weryfikacji*, Instytut Podstaw Informatyki PAN, Warszawa 2002.

[18] Sommerville I., *Software Engineering*, International Computer Science, Pearson Education, 2004.

PRZEGLĄD METODOLOGII PROJEKTOWANIA SYSTEMÓW REGUŁOWYCH HEKATE

Streszczenie: W pracy przedstawiono metodologię i narzędzia do projektowania i budowy systemów regułowych. Prezentowane rozwiązanie cechuje wysoka efektywność reprezentacji wiedzy oparta na użyciu sieci specjalnych atrybutowych tablic decyzyjnych XTT2. Sam proces projektowania ma charakter zstępujący. W pierwszej fazie projektowany jest diagram zależności funkcyjnych pomiędzy atrybutami ARD+. Na podstawie diagramu ARD+ generowany jest automatycznie schemat tablic XTT2 i połączeń między nimi. Tablice te muszą zostać następnie wypełnione szczegółową wiedzą eksperta. W pracy przedstawiono zestaw narzędzi wspomagający proces projektowania i implementacji.