Design Process Ontology – Approach Proposal*

Grzegorz J. Nalepa[1] and Weronika T. Furmańska[1]

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

Abstract. The ARD+ method supports the conceptual design of the XTT^2-based rules. However, number of limitations of the method have been identified. In this paper a new approach to the conceptual design of rules is proposed. The main idea comes down to the use of the Semantic Web methods and tools to represent ARD+ and overcome the limitations of the original method. In the approach proposed in this paper an OWL ontology capturing the semantics of the ARD+ model is proposed. Such an ontology models the functional dependencies between rule attributes, as well as the history of the design process. At the same time it is more flexible that the original method, opening up possibility of integration with other modeling methods and tools, e.g. UML.

1 Introduction

Practical design support is important for intelligent systems [1]. The phase of knowledge acquisition and initial conceptual modeling with help of human experts largely influences the quality complex systems. In case of knowledge-based system a hierarchical and iterative feature of this process improves the design.

ARD+ [2,3,4] method has been invented to support the conceptual design of the XTT^2-based rules. ARD+ (*Attribute Relationship Diagrams*) is a rule prototyping method in the HeKatE project (`http://hekate.ia.agh.edu.pl`). It supports the logical rule design with the XTT^2 method (*eXtended Tabular Trees*) [5,6]. However, number of limitations of the method have been identified.

In this paper a new approach to the conceptual design of rules is proposed. The main idea comes down to the use of the Semantic Web methods and tools to represent ARD+ and overcome the limitations of the original method. In the approach proposed in this paper an OWL ontology capturing the semantics of the ARD+ model is proposed. Such an ontology models the functional dependencies between rule attributes, as well as the history of the design process. At the same time it is more flexible than the original method, opening up possibility of integration with other modeling methods and tools, e.g. UML.

The rest of the paper is organized as follows: The next section describes the context of ARD+ rule prototyping. Then the limitations of the original method

* The paper is carried out within the AGH UST Project No. 10.10.120.105.

are outlined, and the motivation for its extension given. The proposed *Design Process Ontology* (DPO) is a new approach to the conceptual design of rules. The DPO is introduced in the subsequent section. Possible directions for the future work are given in the final section.

2 ARD+ Rule Prototyping Method

The *Attribute Relationship Diagrams* (ARD+) method [2,3,4] supports the conceptual design of rule systems. The primary assumption is, that the state of the intelligent system is described by the attribute values, which correspond to certain system characteristics. The dynamics of the system is described with rules. In order to build the model of the dynamics, the attributes (in this approach state variables) need to be identified first. The identification process is a knowledge engineering procedure, where the designer (knowledge engineer) uses ARD to represent the identified attributes, together with their functional dependencies captured. Using them, rules can be built in the next logical design phase.

ARD is a general method, that tries to capture two features of the design: the attributes, with functional relations between them, and the hierarchical aspect of the process. The second feature is related to the fact that in practice the knowledge engineering process is a gradual refinement of concepts and relations.

In Fig. 1 a simple ARD dependency diagram can be observed. It is in fact one of the phases of the benchmark thermostat case study [7] studied in detail in the HeKatE project. The diagram models a simple dependency read as "thermostat `Temperature` depends on `Time` specification". This is a general statement – currently ARD does not model what the specific dependency is, only a simple fact that some dependency exists.

In the following design stage this model can be refined, by specifying `Time` as a compound attribute, and later on discovering that the set of newly introduced attributes (`Date, Hour, season, and operation`) can be in fact decomposed into two subsets that depend on each other. The nodes of the ARD diagram correspond to so-called *characteristics* (properties) that are described by one or more *attributes*. Attributes can be *conceptual* (general), and *physical* (specific).

Two transformations of the model are possible: finalization and split. The specification transformation (between `Time` and `Date`, `Hour`, `season`, and `operation`) is called *finalization*, whereas the other one is called *split*. These are captured in the *Transformation Process History* diagram (TPH). Together with the ARD dependency diagram they form the *ARD Model*. In the model on the right (Fig. 3) the black edges correspond to finalization and split transformations, and the blue edges show the functional dependencies.

In general, ARD could be used support the design of both forward and backward chaining rules. However, so far it's been mainly used for forward chaining. The basic idea is that having the most detailed, specific ARD dependency diagram, rule prototypes can be automatically built. A rule prototype is a pair of sets of attributes present in the rule premise, and a set of decision attributes. The prototype is aimed at an *attributive* rule language [3], such as XTT^2 [5,6].

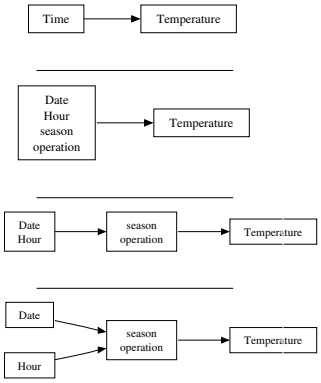

Fig. 1. ARD diagram

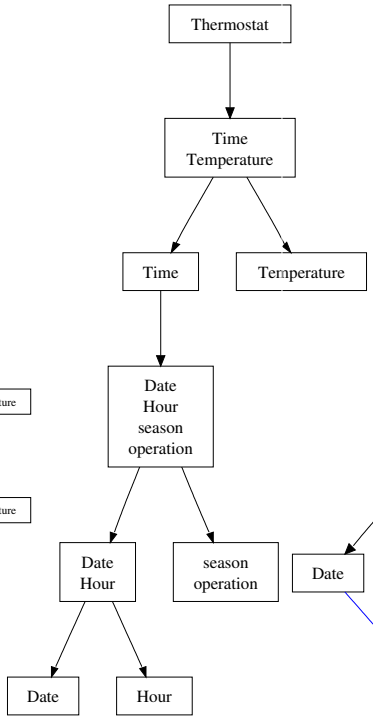

Fig. 2. TPH diagram

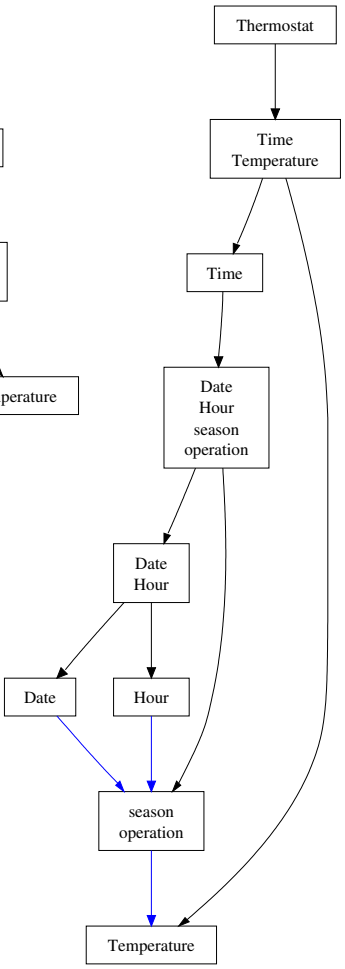

Fig. 3. ARD+ model

3 Motivation

The ARD+ method [4] is an extension of the original ARD [2,3]. Compared to its predecessor it has a formalized description and a well-defined set of model transformations. ARD+ also introduced the concept of capturing the *evolution of the model design* by the means of the TPH diagram. The method also provided a practical algorithm for building XTT prototypes. The ARD+ design process requires the knowledge engineer to identify attributes, characteristics and relations in both the ARD and TPH diagrams. However, it is apparent that the method has certain important *limitations* or drawbacks.

1. The identification of attributes and dependencies is a straightforward task only in the case of simple, small systems with 10-30 rules. However, it could

turn out a very tedious and time consuming tasks in case of complex systems having tens of attributes and hundreds of rules.
2. ARD+ allows only to capture general functional dependencies, without classifying them in any way. In fact the "ARD dependency" has a very unspecific semantics. In ARD+ it is possible to state "A depends on B and C" but it is not possible to specify *how* it depends, what is the nature of the dependency.
3. The semantics of TPH is also a very broad one. An edge in the TPH diagram simply means that the new ARD+ characteristics is somehow related to another characteristics on the previous stage of the design. Again, it is not explicitly specified *how* it is related. In fact, in ARD+ two transformations are possible: finalization and split. The goal of the TPH is to capture these transformations. However it does not explicitly differentiates them (there is only one class of TPH edges).
4. The last problem concerns a coherent description of the ARD+ model. In [4] two diagrams are described: the ARD+ diagram capturing the functional dependencies between properties grouping attributes at a given design stage, and the TPH diagram, capturing the history of the design process. Later on, in the design tool VARDA [8] a combined diagram – here referred as an *ARD+ model* – has been introduced. It combines the dependency and history diagrams as observed in Fig. 3. However, it has not been formally described and analyzed.

The first problem concerns support for knowledge acquisition in general, and has been addressed in [9]. A practical approach to a partial automation of the attribute and dependency identification process by the use of knowledge discovery methods has been introduced there.

The focus of this paper is to propose a single coherent solution for the three remaining problems. A richer knowledge representation model is proposed. In particular it should:

1. allow to specify different classes of functional dependencies,
2. provide more expressive means for the history description,
3. allow to build a single coherent model combining both functional dependencies and history information in a single model.

In the next section a proposal of using standard Semantic Web methods meeting the above mentioned requirements is put forward.

4 Design Process Ontology Proposal

The basic idea presented here comes down to proposing an *ontology* – called the *Design Process Ontology* (DPO) –capturing the functional dependencies present in the main ARD diagram and history information captured in the TPH.

In general, an ontology is a knowledge representation [10] that serves as a formal definition of objects in the universe of discourse and relationships among them. The domain is described by means of concepts (classes), roles (properties)

and instances (individuals). Relations can be specified both among classes and individuals. Ontologies allow for a formal definition of the vocabulary in the universe of discourse, together with its intended meaning and constraints present in the domain. In this paper the Web Ontology Language (OWL) [11], specifically the OWL DL dialect, based on Description Logics (DL) [12] is used.

Similarities of the ontology-based modelling approach and the ARD method has been investigated in [13,14]. Alternative approaches to a mapping between ontology concepts and system attributes have been considered. One of them consists in representing system attributes and characteristics as concepts in an ontology. Another proposal is to treat attributes as instances of a generic class *Attribute*. Both approaches allow for describing the relations between system attributes using ontology properties. In this paper an ontology based on the former approach is presented.

Design Process Ontology is a proposal of a *task ontology* [15]. Its aim is to capture the system characteristics together with dependencies among them, as well as represent the gradual refinement of the design process. Basically, DPO consists of a general class *Attribute* and four properties: `dependsOn`, `transformedInto`, `splitInto`, and `finalizedInto`.

The property `dependsOn` is very general and may be further specialized. It is used to represent *functional dependencies* among the system characteristics. At this stage we do not formally specify the semantics, which intuitively may be put as "one attribute depends on the other". *Functional* in this context have a different meaning than *functional properties* used in OWL (`owl:FunctionalProperty`), where they denote that the property has an unique value for each instance.

The other three properties (`transformedInto`, `splitInto` and `finalizedInto`) denote the TPH relations – the design process transformations. A hierarchy of the TPH properties may be introduced as follows (DL convention):
$split\_into \sqsubseteq transformed\_into$, $finalized\_into \sqsubseteq transformed\_into$.
The domain and the range of all the properties is the general class `Attribute`.

DPO may be specialized by concrete ontologies for specific design tasks. In this case system characteristics (conceptual and physical attributes) subclass the `Attribute` class. All the characteristics and attributes identified in a system are represented as independent classes. The properties may be specialized accordingly, so that they range over concrete system classes rather than the general `Attribute` class. An example of such an ontology for the Thermostat system is depicted in Fig. 4. The ontology has been built in OWL using Protegé.

ARD is a method used in a gradual refinement process. As the process progresses, the functional dependencies change and new TPH relations are added. It is worth emphasizing that the historical TPH relations remain unchanged, whereas the functional ones are different at each process stage (observe Fig. 2). Thus, the ontology is different at various design stages. At a given moment an ontology represents all of the characteristics and attributes identified in the system from the beginning of the design process. All the TPH relations, such as split and finalization are shown. As for the functional dependencies, only the most specific relations identified at certain moment are shown (observe Fig. 4).

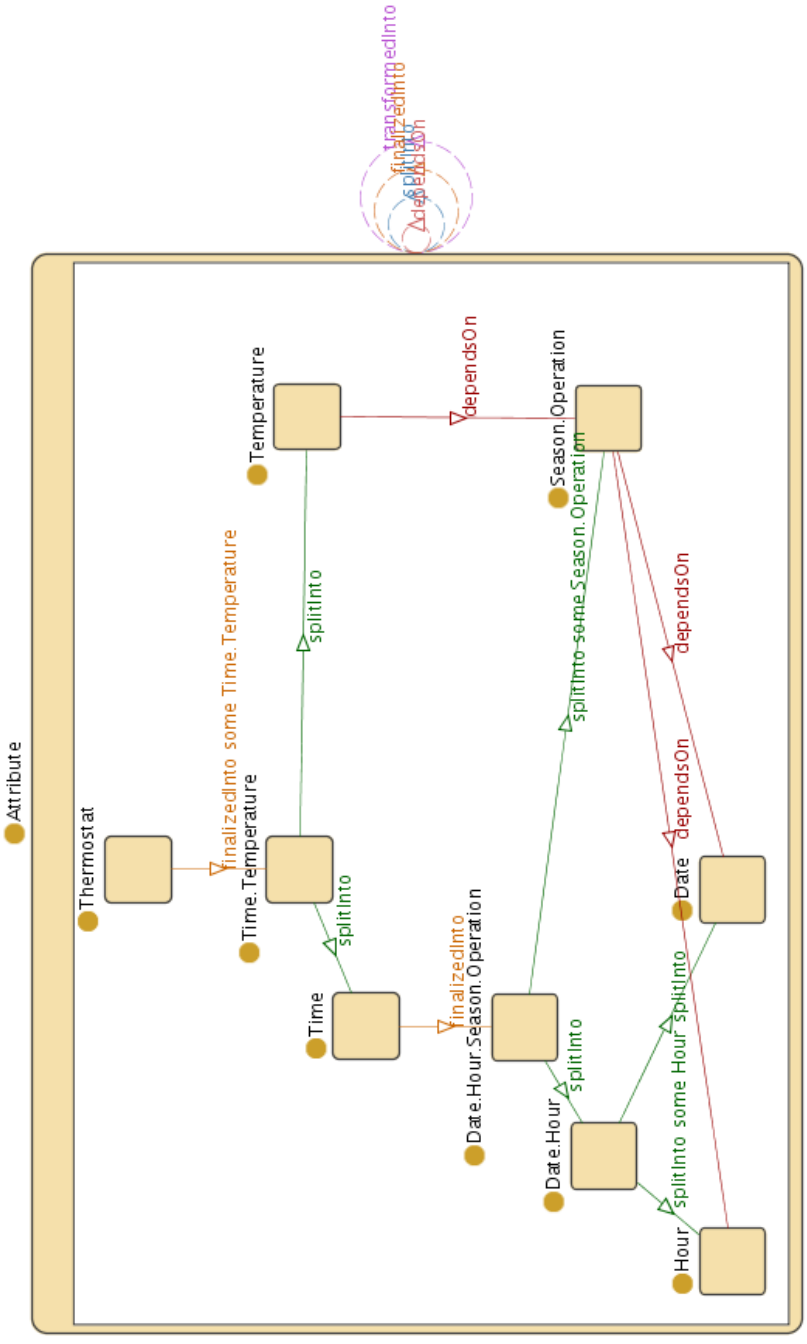

Fig. 4. Simple DPO in OWL designed in Protegé

5 Conclusions and Future Work

The paper concerns the conceptual prototyping of decision rules. The ARD+ method discussed in a paper provides simple means for capturing functional dependencies between attribute present in rules. Moreover, it allows to capture and represent the evolution of the model, the history of the design. However, it has some limitations addressed in the paper.

In order to solve these problems, it is proposed to use the Semantic Web tools in the system conceptual design phase. The proposal is to capture the system elements and various dependencies among them, using an ontology. In the *Design Process Ontology*, certain specific relations are defined. The ontology presented in this paper includes only the basic relations and serves as the illustration of the approach. A concrete ontology specializing the DPO is equivalent to ARD model of a system. Moreover, it provides a more coherent model, while allowing to introduce more relations, which would not be possible in the original ARD.

Future work concerns further investigation of the possibilities of using ontologies in the design process. This includes formalizing various ARD+ model features in Description Logics. As for now, the dependencies between the system characteristics are modelled with various OWL properties (roles). It will be considered, if some of those relations can be incorporated into class descriptions. Certain formal descriptions would help to verify relations such as `split_into`.

The set of various dependencies represented in an ontology will be enlarged. The general functional relation `depends_on` should be specialized, including the differentiation between AND/OR dependencies (as used in AND/OR graphs in diagnostic systems). The set of TPH relations may also be enriched.

As the design process progresses, the Design Process Ontology changes. The transformations between subsequent ontologies will be analyzed and their formalization will be proposed. Use of rules on top of the DPO is considered. These rules could be possibly introduced as DLP (DL Programs) or expressed in SWRL.

A future requirement – not directly addressed here – concerns support for certain annotations present in the UML class diagram. In this case, the reworked method would be closer in semantics to the UML-based design. For more details see [16,17]. The possibilities of integrating OWL with UML using Protege4 is also to be discussed. This approach could provide means to use and integrate both Semantic Web technologies as well as classic software engineering methods to design intelligent systems.

References

1. 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.
2. 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

3. Ligęza, A.: Logical Foundations for Rule-Based Systems. Springer-Verlag, Berlin, Heidelberg (2006)
4. 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
5. Nalepa, G.J., Ligęza, A.: A graphical tabular model for rule-based logic programming and verification. Systems Science **31**(2) (2005) 89–95
6. Nalepa, G.J., Ligęza, A.: Hekate methodology, hybrid engineering of intelligent systems. International Journal of Applied Mathematics and Computer Science (2009) accepted for publication.
7. Negnevitsky, M.: Artificial Intelligence. A Guide to Intelligent Systems. Addison-Wesley, Harlow, England; London; New York (2002) ISBN 0-201-71159-1.
8. Nalepa, G.J., Wojnicki, I.: Varda rule design and visualization tool-chain. In Dengel, A.R., et al., eds.: KI 2008: Advances in Artificial Intelligence: 31st Annual German Conference on AI, KI 2008: Kaiserslautern, Germany, September 23–26, 2008. Volume 5243 of LNAI., Berlin; Heidelberg, Springer Verlag (2008) 395–396
9. Atzmueller, M., Nalepa, G.J.: A textual subgroup mining approach for rapid ard+ model capture. In Lane, H.C., Guesgen, H.W., eds.: FLAIRS-22: Proceedings of the twenty-second international Florida Artificial Intelligence Research Society conference: 19–21 May 2009, Sanibel Island, Florida, USA. (2009)
10. van Harmelen, F.: Applying rule-based anomalies to kads inference structures. ECAI'96 Workshop on Validation, Verification and Refinement of Knowledge-Based Systems (1996) 41–46
11. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview, w3c recommendation 10 february 2004. Technical report, W3C (2004)
12. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
13. Szostek-Janik, J.: Translations of knowledge representations for rule-based systems. AGH University of Science and Technology (2008) MSc Thesis.
14. Nalepa, G.J., Furmańska, W.T.: Proposal of a new rule-based inference scheme for the semantic web applications. In: 1st International Conference on Computational Collective Intelligence - Semantic Web, Social Networks & Multiagent Systems. (2009) To be published.
15. Guarino, N.: Formal ontology and information systems. In: Proceedings of the First International Conference on Formal Ontologies in Information Systems. (1998) 3–15
16. Nalepa, G.J., Kluza, K.: Uml representation proposal for xtt rule design method. In Nalepa, G.J., Baumeister, J., eds.: 4th Workshop on Knowledge Engineering and Software Engineering (KESE2008) at the 32nd German conference on Artificial Intelligence: September 23, 2008, Kaiserslautern, Germany, Kaiserslautern, Germany (2008) 31–42
17. Nalepa, G.J.: Xtt rules design and implementation with object-oriented methods. In Lane, H.C., Guesgen, H.W., eds.: FLAIRS-22: Proceedings of the twenty-second international Florida Artificial Intelligence Research Society conference: 19–21 May 2009, Sanibel Island, Florida, USA. (2009)