Technical Report 458, Institute of Computer Science, University of Wuerzburg, 2009

Towards Rapid Knowledge Capture using Textual Subgroup Mining for Rule Prototyping

Martin Atzmueller
University of Würzburg,
Department of Computer Science VI,
Würzburg, Germany
atzmueller@informatik.uni-wuerzburg.de

Grzegorz J. Nalepa
Institute of Automatics,
AGH University of Science and Technology,
Kraków, Poland,
gjn@agh.edu.pl

Abstract

Manual knowledge acquisition is usually a costly and time-consuming process. Automatic knowledge acquisition methods can then significantly support the knowledge engineer, by providing an initial sketch of a knowledge base that can be refined and fine-tuned in subsequent steps. In this paper, we propose an approach for rapid knowledge capture for rule prototyping in the context of ARD+ models. ARD+ is a conceptual prototyping method for decision rules. Its primary objective is to capture the functional dependencies between attributes used in rules. ARD+ introduces a gradual design and refinement design process for rules. The knowledge capture methodology is based on textual subgroup mining for discovering the important concepts and their dependencies. The important concepts and their relations can then be mapped to ARD+ models in order to enable the rule prototyping step. We describe the approach in detail and provide motivating application examples.

1 Introduction

In recent years, there is a trend towards rule-based techniques, e.g., for business rules applied in various intelligent systems [Giarratano and Riley, 2005; Morgan, 2002]. The growing popularity of rule technologies creates a need for developing efficient design support tools suitable not just for knowledge engineers, but also software engineers and the business-oriented community. With this emergence in the technological mainstream, applied AI methods [Russell and Norvig, 2003] play a growing role in supporting software engineering (SE). However, as for other AI approaches, rule formalization requires knowledge acquisition. This is usually costly and time-consuming relying on a domain specialist or knowledge engineer (*knowledge acquisition bottleneck* [Hayes-Roth *et al.*, 1983]).

In this context, knowledge discovery [Klösgen and Zytkow, 2002], including data mining [Han and Kamber, 2000] methods can play a crucial role for supporting the knowledge engineer and for making the knowledge acquisition process more feasible. Then, knowledge discovery (KD) methods can be applied for the acquisition of an initial sketch of the knowledge base that can be refined later.

In this paper, we present an approach applying textual subgroup mining techniques for the rapid knowledge capture of dependencies between decision rule attributes, e.g., [Atzmueller *et al.*, 2007]. ARD+ (*Attribute Relationship Diagrams*) is a rule prototyping method in the HeKatE project (hekate.ia.agh.edu.pl). It supports the logical rule design with the XTT^2 method (*eXtended Tabular Trees*). The textual subgroup mining techniques for discovering attribute dependencies are implemented using the VIKAMINE [Atzmueller and Puppe, 2005; 2007] system (www.vikamine.org).

In a semi-automatic process the discovered attribute relations are inspected, validated, and finally mapped to an ARD+ model. After that, the prototypical instantiation can be utilized for setting up the corresponding decision rules. The focus of this paper is thus on developing a practical KD method for supporting rule design based on knowledge discovery techniques. From the SE point of view the method would support engineers working on application requirements. It directly corresponds to the knowledge acquisition phase of the knowledge engineering (KE) process for obtaining initial rule prototypes that are being refined later.

The rest of the paper is organized as follows: The next section describes the context of ARD+ rule prototyping before we motivate the application of knowledge discovery techniques for this purpose: We shortly introduce subgroup mining as the basic knowledge discovery method. After that, we describe the knowledge discovery methodology for rapid ARD+ rule capture: We introduce the process model for the proposed approach, and discuss the respective steps in detail. Finally, we conclude the paper with a summary and point out interesting directions for future work.

2 Preliminaries

In the following section we first introduce ARD+ models, sketch the proposed prototyping process, and finally introduce subgroup mining – the basic knowledge discovery technique applied for the presented approach.

2.1 ARD+ Conceptual Design

The *Attribute Relationship Diagrams* (ARD+) method [Nalepa and Wojnicki, 2008c], an extension of the original ARD approach [Nalepa and Ligęza, 2005a; Ligęza, 2006] 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 properties. 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.

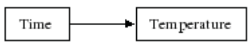

Figure 1: Simple ARD dependency diagram

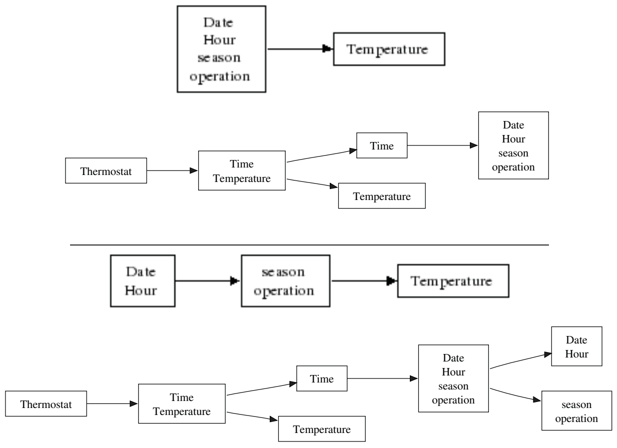

Figure 2: ARD diagram transformation

ARD is a general method, that tries to capture to 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 [Negnevitsky, 2002] 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 model what the specific dependency is, only a simple fact that *some* dependency exists.

In the next 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 decomposed into two subsets that depend on each other.

The nodes of the ARD diagram correspond to so-called *properties* that are described by one or more *attributes*. Attributes can be *conceptual* (general), and *physical* (specific). The specification transformation (between `Time` and `Date, Hour, season, and operation`) is called *finalization*, whereas the other one is called the *split*. These are captured in the *Transformation Process History* diagram (TPH). Together with the ARD dependency diagram they form the *ARD Model*. The two subsequent design phases together with the corresponding TPH are presented in Fig. 2 (TPH diagrams are shown below ARD diagrams). The complete ARD model at this design stage is shown in Fig. 3. In the model the black edges correspond to finalization and split transformations, and the blue edges (in the upper right part of the diagram) show the functional dependencies; these are in fact the exact two dependencies that can be observed in the lower half of the Fig. 2 One can observe, that two properties observed in Fig. 1 where created by finalizing `Time` to `Time, Temperature`, and then splitting the latter.

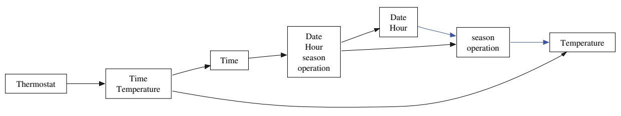

Figure 3: Simple ARD model

2.2 ARD+ Model Prototyping

ARD+ is aimed at supporting rule design, in general it could be used for both forward and backward chaining rules; so far it has been mainly used for forward chaining. The basic ideas is, that having the most detailed, specific ARD dependency diagram, rule prototypes can be automatically built (for the prototyping algorithm see [Nalepa and Wojnicki, 2008a]. A rule prototype consists of a set of attributes present in the rule premise, and a set of decision attributes. The prototype is described by an *attributive* rule language [Ligęza, 2006], such as XTT [Nalepa and Ligęza, 2005b]. The ARD+ design process requires the knowledge engineer to identify attributes, properties and relations in both the ARD and TPH diagrams. As such it has *certain limitations*. First of all, the identification 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. Second of all, it only captures general functional dependencies, without classifying them in any way.

In this paper we aim at addressing the *second problem*. From the example ARD+ designs conducted so far, it seems that the attribute and dependency identification process could be partially automated. This is the motivation for introducing the use of knowledge discovery methods.

The principal idea is to use knowledge discovery methods to identify possible ARD+ attributes and properties, and then to propose possible relationships. The applied knowledge discovery approach is based on subgroup mining techniques introduced in the next section. The knowledge discovery process uses a natural language descriptions of system requirements, given by users or domain experts. It is assumed, that this description is based on a somehow restricted (semi-formalized in a sense) language; an example of such a language is the SBVR [OMG, 2006] controlled language. Using documents containing such descriptions we first apply text mining preprocessing techniques for obtaining a set of structured data records in a word-vector representation. These data records are assumed to contain the important features and concepts of the domain. After that, we apply knowledge discovery techniques for discovering associations between given target concepts represented by subgroup patterns. Using these, we can then generate prototypical ARD+ models.

2.3 Subgroup Mining

In the following, we introduce subgroup mining, e.g., [Wrobel, 1997; Klösgen, 1996; Atzmueller *et al.*, 2005a], for discovering interesting patterns.

Subgroup patterns, often provided by conjunctive rules, describe ’interesting’ subgroups of cases/instances, e.g., "the subgroup of documents containing the term "thermostat" and "regulate"" shows a significantly increased cooccurrence count with the term "temperature" compared to all the documents (with respect to the term "temperature"). In general, the main application areas of subgroup mining are exploration and descriptive induction, to obtain an overview of the relations between a target variable and a set of explaining variables.

The exemplary subgroup above is then described by the relation between the independent (explaining) variable (temperature = true, regulation = true) and the dependent (binary) target variable (thermostat = true). The independent variables are modeled by selection expressions on sets of attribute values. In our case, these are all represented by binary attributes corresponding to certain words or terms that occur or do not occur in a certain document, according to the applied *bag-of-words* document representation.

Let Ω_A denote the set of all attributes. For each attribute $a \in \Omega_A$ a range $dom(a)$ of values is defined. An attribute–value assignment $a = v$, where $a \in \Omega_A, v \in dom(a)$, is called a *feature*. We define the feature space $\mathcal{V}_A$ to be the (universal) set of all features. A single-relational propositional *subgroup description* is defined as a conjunction

$$sd = e_1 \wedge e_2 \wedge \cdots \wedge e_n$$

of (extended) features $e_i \subseteq \mathcal{V}_A$, which are then called selection expressions, where each e_i selects a subset of the range $dom(a)$ of an attribute $a \in \Omega_A$. We define Ω_{sd} as the set of all possible subgroup descriptions. The subgroup size $n(s)$ for a subgroup s is determined by the number of instances/cases (or documents represented by a word-vector instance) covered by the subgroup description sd. For a binary target variable, we define the true positives $tp(sd)$ as those instances containing the target variables and the false positives $fp(sd)$ as those instances not containing the target variable, respectively.

A quality function measures the interestingness of the subgroups and is used to rank these. Typical quality criteria include the difference in the distribution of the target variable concerning the subgroup and the general population, and the subgroup size.

Definition 1 (Quality Function) *Given a particular target variable $t \in \mathcal{V}_A$, a quality function $q : \Omega_{sd} \times \mathcal{V}_A \to R$ is used in order to evaluate a subgroup description $sd \in \Omega_{sd}$, and to rank the discovered subgroups.*

Several quality functions were proposed (cf. [Wrobel, 1997; Klösgen, 2002; Lavrac *et al.*, 2004; Atzmueller and Puppe, 2006]), for example, the functions q_{BT} and q_{RG}:

$$q_{BT} = \frac{(p - p_0) \cdot \sqrt{n}}{\sqrt{p_0 \cdot (1 - p_0)}} \cdot \sqrt{\frac{N}{N - n}} \, ,$$

$$q_{RG} = \frac{p - p_0}{p_0 \cdot (1 - p_0)}, n \geq \mathcal{T}_{Supp} \, .$$

p denotes is the relative frequency of the target variable in the subgroup, p_0 is the relative frequency of the target variable in the total population, N is the size of the total population, and n denotes the size of the subgroup. In contrast to the quality function q_{BT} (the classic binomial test), the quality function q_{RG} only compares the target shares of the subgroup and the total population measuring the *relative gain*. Therefore, a support threshold $\mathcal{T}_{Supp}$ is necessary to discover significant subgroups.

The quality function is usually selected according to the application requirements. We have found that the *relative gain* quality function is easily interpretable and understandable by users. Therefore, in the presented approach we apply this quality function with a minimal support threshold $\mathcal{T}_{Supp} = 5$ in order to guarantee statistically relevant results. Additionally, the discovered patterns can also be sorted and browsed according to the various parameters, for example, the quality, the subgroup size and by the implied support, such that the application requirements can always be kept in focus while performing the mining step.

3 Methodology

The following section first considers the process model for semi-automatic discovery of ARD+ models. After that, we describe the key steps of the process in detail. First, we describe the preprocessing of the textual documents using standard text mining techniques. After that, we discuss the subgroup mining methods for discovering subgroup patterns for the important associations between concepts and the semi-automatic discovery step of the ARD+ models.

3.1 Process Model

In the following we discuss the process model for building ARD+ models using textual subgroup mining methods. It is shown in Figure 4 and described in more detail below. As mentioned above, the input of the process is a *set of text documents* containing natural language descriptions of system requirements, given by users or domain experts. These are based on a somehow restricted (semi-formalized in a sense) language, for example, the SBVR [OMG, 2006] controlled language.

Since the applied text-mining approach is based on statistical techniques we need to ensure that there is a sufficient number of documents for each targeted ARD+ model. Usually about 5-10 documents per model is a good starting point for the knowledge capture process.

1. **Extract Dataset:** For the textual subgroup mining approach we first need to preprocess the unstructured text data in order to obtain a structured representation, that is, a dataset of data records. We create a common word-vector representation: In this representation, the data records are represented by vectors containing binary features that correspond to the occurrence of a word in the respective document.
2. **Define Target Concepts:** Before applying the subgroup mining techniques we need to identify a list of target concepts. This set is either obtained by selecting a subset of the important concepts from the dataset extraction step. Additionally, the important concepts are simply defined using background knowledge. For example, in the thermostat case we know that time and temperature are important. The background knowledge is provided by the user, for example, the hint that "the thermostat is about setting the temperature". Then, "temperature" becomes the target concept.
3. **Apply Subgroup Mining:** In this step, we apply subgroup mining for each target concept and consider all other concepts contained in the dataset as independent variables. Doing this we obtain concepts and/or combinations of concepts that are closely related to the target concept. These concept combinations are then represented by subgroup patterns that indicate associations with the respective target concept.
4. **Inspect Subgroup Network:** After the subgroup patterns have been discovered, we visualize the relations between them in multiple subgroup networks: Each subgroup pattern is linked to its target pattern (represented by a node for the respective target concept). The network also contains links between the individual subgroup patterns, if one pattern contains a

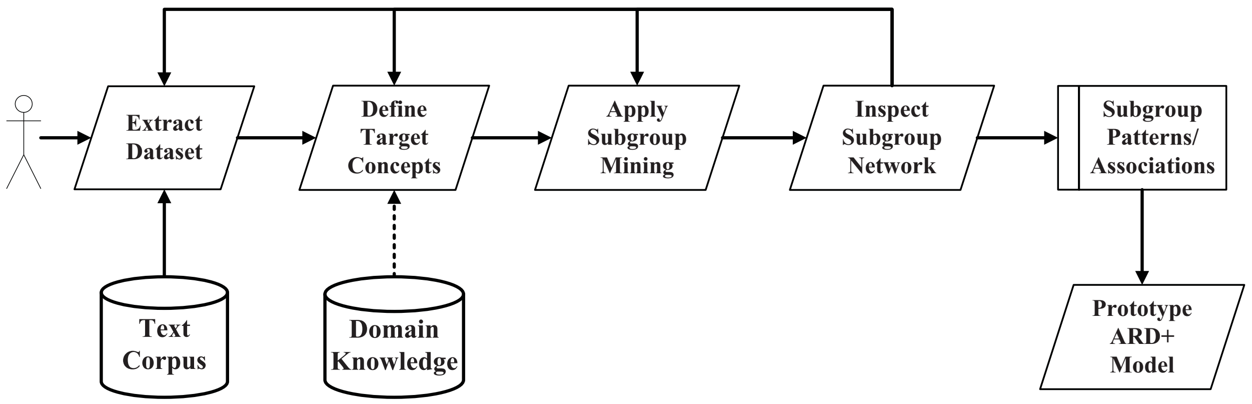

Figure 4: Process model for the semi-automatic acquisition/prototyping of ARD+ models.

(target) concept of the second pattern. Additionally we can also visualize associations between concepts, cf., [Atzmueller and Puppe, 2007], for a more comprehensive overview.

5. **Prototype ARD+ Model:** Since the ARD+ method aims at capturing relations between attributes, we can simply map fragments of the subgroup pattern networks to dependencies between the contained attributes/concepts. Then, these dependencies directly correspond to the ARD+ functional dependencies between system properties given by the attributes/concepts. After that, the ARD+ model is usually refined by the user in order to provide a good starting point for prototyping decision rules.

The process is incremental and can be iterated by the user as needed. For the dataset extraction step, the user can incrementally refine the set of extracted concepts. As discussed below we apply a TF-IDF (term-frequency/inverted document frequency) [Baeza-Yates and Ribeiro-Neto, 1999] threshold-based method, so the applied threshold can also be tuned as needed. The target concepts can also be extended/reduced as needed especially considering the output of the subgroup mining step: After a network of subgroup patterns has been constructed we can identify and add further target concepts. These are given by attributes contained in the *interesting patterns* linked to the given target concepts that are refined for a further layer of the ARD+ model.

The process is implemented using the VIKAMINE [Atzmueller and Puppe, 2005; 2007] system for knowledge-intensive subgroup mining. VIKAMINE provides all the required visualizations. The relation/rule instantiation phase is then implemented using a special plugin of VIKAMINE.

3.2 Preprocessing

Before the subgroup mining approach can be applied we need to construct a structured data representation (set of data records) from the given textual documents. In order to obtain this representation that aims to cover the set of *important* concepts (words) represented in the vector, we need to perform a feature selection step: For this step, we apply standard methods implemented, e.g., in the ASV Toolbox [Biemann *et al.*, 2008] and in the KNIME textprocessing plug-in [KNIME, 2009] for the term and terminology extraction, e.g., [Wermter and Hahn, 2005].

Additionally, we apply a classic TF-IDF (term-frequency/inverted document frequency) [Baeza-Yates and Ribeiro-Neto, 1999] threshold-based metric for finally converting the documents with the extracted terms to a word-vector representation containing the important terms. The TF-IDF metric is based on the idea that the descriptive terms of a document should appear often in the respective document but should not appear in many other documents.

The TF-IDF computes a *weight* denoting the importance of a term/concept with respect to a collection of documents. The *term frequency*

$$tf_{i,j} = freq_{i,j} / \max_l freq_{l,j}$$

is computed for each term i depending on document j. The denominator specifies the maximum frequency of all terms contained in the document. The *inverted document frequency* $idf_i = \log \frac{N}{n_i}$ trades of N, the number of documents of the document collection, and n_i the number of documents containing the term i. The weight of a term i is then computed as

$$w_{i,j} = tf_{i,j} \cdot idf_i$$

Using a suitable threshold we can then identify the important terms of the document. For a more in-depth discussion we refer to, e.g., [Baeza-Yates and Ribeiro-Neto, 1999].

The approach is semi-automatic such that the user can already inspect the word list to be used at this step, and refine the results if needed, e.g., by tweaking the applied threshold.

3.3 Semi-Automatic Subgroup Mining

Consider the previously discussed thermostat system. The general idea is to describe a system that controls the temperature in an office depending on time specification. The user description could be as follows: "In our office temperature changes over time. In fact, it is different on different days of the week. In general the temperature is different during business hours. Of course it also depends on the season, so it is different in different months."

At the ARD stage, we assume a preliminary, general description with no actual specification of attribute values. In this phase we are interested at identifying possible important concepts, that would form the rule vocabulary (attributes). As outlined above, we apply the subgroup mining method for each target concept and collect the obtained dependencies in multiple subgroup pattern networks. These networks then need to be inspected, assessed and validated by the user [Atzmueller and Puppe, 2008].

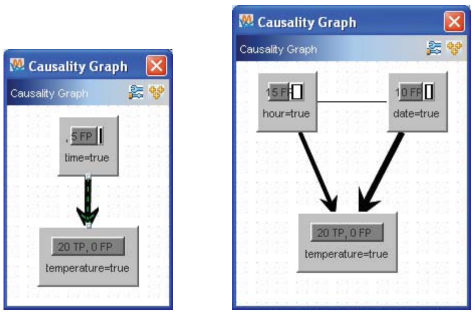

Figure 5: Exemplary dependency network (subgroup patterns) between the attributes time, hour, date, temperature

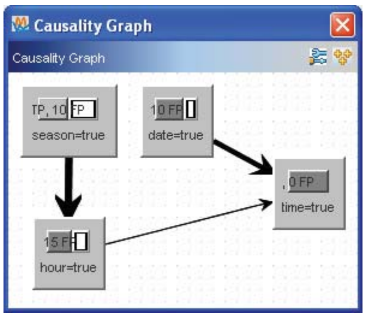

Figure 6: Exemplary dependency network (subgroup patterns) between the attributes time, hour, date, season

Figure 5 and Figure 6 show such fragments for subgroup networks referring to the thermostat example. These are simple networks for illustrating the presented approach using an example data set for the thermostat example. In practice such networks will be quite large motivating the techniques discussed below. Figure 5 shows the dependency between time, hour and date, and the attribute temperature, while Figure 6 shows the dependencies between the attributes season, hour, date and the attribute time. For the latter, hour and date are causally connected to time, while season is mediated through the attribute hour in this diagram. By composing these diagrams we can then build ARD+ models as discussed above.

For the interactive inspection of the constructed subgroup network suitable browsing, filtering and zooming methods are essential in order to support the user. The user can then select a subset of the relations that are being used for rule template instantiation, i.e., for setting up templates for relations that can then be refined to rules. The VIKAMINE environment provides a wide range of visualization methods that enable an easy-to-apply validation and assessment step of the discovered patterns.

Using both automatic and interactive techniques, the discovered relations can be post-processed in an effective manner, as shown in several projects in medical and technical domains [Atzmueller *et al.*, 2005a; 2005b; 2007]. Additionally, VIKAMINE allows for an integrated user experience with stream-lined discovery and analysis capabilities by the ability to include background knowledge specific for the problem at hand. We can optionally apply causal methods for refining the discovered subgroup net [Atzmueller and Puppe, 2007; Atzmueller *et al.*, 2009], such that only the causal relations are retained. After the discovered subgroup patterns representing associations between the concepts have been inspected and validated they are then used for the construction of prototypical ARD+ models.

3.4 Creating ARD+ Models and Rule Prototypes

Using the dependencies discovered using the subgroup mining techniques, we construct subgroup networks as shown above. After the relations have been inspected and validated by the user [Atzmueller and Puppe, 2008], we can map fragments of these networks to ARD+ models.

For example, using the pairs, temperature/time, temperature/hour, date, and time/hour, date, season we can create a possible ARD+ structure similar to the one discussed above for the ARD+ models shown in Figure 1 to Figure 3. However, in the general case different ARD dependency diagrams, corresponding to different design phases could be proposed. Then, the engineer can refine them manually, select correct ones, and utilize the tool to automatically built the TPH diagram.

A practical algorithm providing a transition from the ARD+ design to rule design has been introduced by [Nalepa and Wojnicki, 2008a]. The goal of the algorithm is to automatically build prototypes for rules from the ARD+ design. The input of the algorithm is the most detailed ARD+ diagram, that has all of the physical attributes identified (in fact, the algorithm can also be applied to higher level diagrams, generating rules for some parts of the system being designed). The output is a set of *rule prototypes* in a very general format (`atts` stands for attributes):

```
rule: condition atts | decision atts
```

The algorithm is *reversible*, that is having a set of rules in the above format, it is possible to recreate the most detailed level of the ARD+ diagram. The rule prototyping algorithm has been successfully implemented in Prolog as a part of the VARDA tool [Nalepa and Wojnicki, 2008d].

3.5 Design Tool Support

The ARD+ model obtained by the knowledge discovery process may be partial and not optimal in a general case. This is why a manual refinement may be needed. Currently the ARD+ design process is supported by two tools VARDA and recently HJED.

VARDA (*Visual ARD+ Rapid Development Alloy*) [Nalepa and Wojnicki, 2008b; 2008d] is a prototyping environment for ARD+; the ImageMagick tool provides an instant displaying of the prototype at any design stage.

VARDA is designed in a multi-layer architecture:

- knowledge base to represent the design: attributes, properties, dependencies,
- low-level primitives: adding and removing attributes, properties and dependencies,
- transformations: finalization and split including defining dependencies and automatic TPH creation,
- low-level visualization primitives: generating data for the visualization tool-chain,
- high-level visualization primitives: displaying actual dependency graph among properties and the TPH.

At the design stage, proper visualization of the current design state, is a key element. It allows to browse the design easily and identify gaps, misconceptions or mistakes more easily. ARD+ and TPH are directed graphs. Proper graph visualization, node distribution, edge distribution and labeling is a separate domain [Ross and Wright, 2002]. Therefore, instead of reinventing these concepts, or implementing them from scratch, a tool-chain of well proved tools to provide actual visualization is assembled: For visualization, GraphViz readable code is generated using the knowledge elements describing the ARD+ and TPH. GraphViz (see `graphviz.org`) is a graph visualization software enabling representing structural information as diagrams of abstract graphs and networks. The structural information needs to be expressed in a simple text-based source file. GraphViz renders the source file and generates a visual representation of the structural information taking into account appropriate vertex distribution, edge placement and labeling. The visual representation is provided in many different formats, including but not limited to, bitmap formats such as: JPG, PNG, TIF, scalable formats: SVG, PS, as well as editable ones: FIG, DIA, VRML. As for the tool-chain, GraphViz generates a PNG bitmap which subsequently is displayed by ImageMagick. ImageMagick does not merely display the diagram, but it also allows for panning, making annotations and saving it as a file in many bitmap formats including PNG and JPG.

HJED[1] is a newer ARD+ visual design tool implemented in Java. It supports the full ARD design, including ARD and TPH visualization. It also provides ability to manually refactor the model. The model can be exported to HML and the imported by the rule design tools provided in HeKatE.

3.6 ARD Design Markup

ARD+ is a conceptual design method in the HeKatE project. Knowledge in the HeKatE design process is described in HML, [2] a machine readable XML-based format. HML consists of three logical parts: attribute specification (ATTML), attribute and property relationship specification (ARDML) and rule specification (XTTML).

The attribute specification regards describing attributes present in the system. It includes attribute names and data types used to store attribute values. The attribute and property relationship specification describes what properties the system consists of and which attribute identifies these properties. Furthermore, it also stores all stages of the design process. The rule specification stores actual structured rules. As for the ARD-only description including ARD+ and TPH diagrams ARDML with ATTML is used.

Below the excerpt of the XML representation of the Thermostat ARD+ design is given.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE hml SYSTEM "hml.dtd">
<hml>
  <attributes>
    <attr name="Thermostat" id="att_0"/>
    <attr name="Time" id="att_1"/>
    <attr name="Temperature" id="att_2"/>
    <attr name="Date" id="att_3"/>
    . . .
  </attributes>
  <properties>
    <property id="prp_0">
      <attref ref="att_0"/>
    </property>
    <property id="prp_1">
      <attref ref="att_1"/>
      <attref ref="att_2"/>
    </property>
    <property id="prp_2">
      <attref ref="att_1"/>
    </property>
    <property id="prp_3">
      <attref ref="att_2"/>
    </property>
    . . .
  </properties>
  <tph>
    <trans src="prp_0" dst="prp_1"/>
    <trans src="prp_1" dst="prp_2"/>
    <trans src="prp_1" dst="prp_3"/>
    <trans src="prp_2" dst="prp_4"/>
    <trans src="prp_4" dst="prp_5"/>
    <trans src="prp_4" dst="prp_6"/>
    . . .
  </tph>
  <ard>
    <dep independent="prp_12"
         dependent="prp_7"/>
    <dep independent="prp_14"
         dependent="prp_15"/>
    <dep independent="prp_15"
         dependent="prp_8"/>
    <dep independent="prp_16"
         dependent="prp_8"/>
    <dep independent="prp_7"
         dependent="prp_17"/>
    <dep independent="prp_8"
         dependent="prp_17"/>
  </ard>
</hml>
```

This representation is used by VARDA to store the model.

3.7 Prolog Representation

The same model is internally represented by VARDA in Prolog as in the excerpt presented below.

```
:- dynamic ard_att/1.
ard_att('Thermostat').
ard_att('Time').
ard_att('Temperature').
ard_att('Date').
ard_att('Hour').
ard_att(season).
ard_att(operation).
ard_att(day).
ard_att(month).
ard_att(today).
ard_att(hour).
ard_att(thermostat_settings).

:- dynamic ard_property/1.
ard_property(['Thermostat']).
ard_property(['Time', 'Temperature']).
ard_property(['Time']).
ard_property(['Temperature']).
ard_property(['Date','Hour',season,operation]).
ard_property(['Date', 'Hour']).
 . . .

:- dynamic ard_depend/2.
ard_depend([month], [season]).
ard_depend([day], [today]).
ard_depend([today], [operation]).
ard_depend([hour], [operation]).
ard_depend([season],
           [thermostat_settings]).
ard_depend([operation],
           [thermostat_settings]).
```

[1]https://ai.ia.agh.edu.pl/wiki/hekate:hjed

[2]https://ai.ia.agh.edu.pl/wiki/hekate:hekate_markup_language

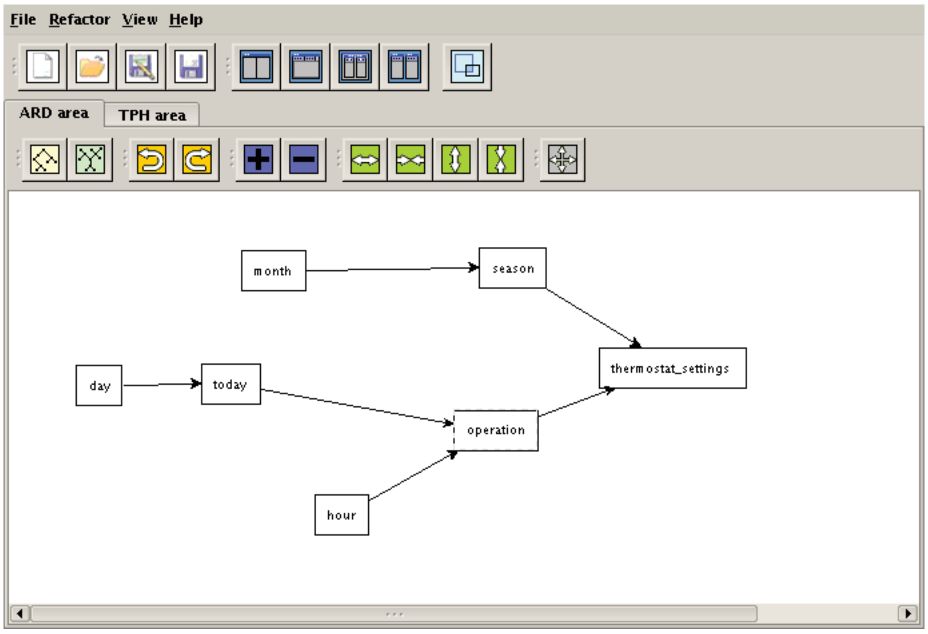

Figure 7: Thermostat design in HJEd

```
:- dynamic ard_hist/2.
ard_hist(['Thermostat'],
         ['Time', 'Temperature']).
ard_hist(['Time', 'Temperature'],
         ['Time']).
ard_hist(['Time', 'Temperature'],
         ['Temperature']).
ard_hist(['Time'],
         ['Date','Hour',season,operation]).
 . . .
```

In VARDA this form is a suitable representation for simple manipulation and transformation, including and enabling knowledge refinement.

4 Conclusion

In this paper, we have presented a methodological approach for rapid model capture and rule prototyping in the context of ARD+ models. The methodology is based on a textual subgroup mining method that is applied for identifying dependencies between concepts. The discovered and validated dependencies are then mapped to the functional dependencies of ARD+ models. After that, decision rules can be (semi-)automatically created by refining the proposed rule instantiations.

So far, we have implemented a prototypical system of the approach: First experiments and evaluations show promising results as shown by the presented examples. For future work we plan to perform a comprehensive evaluation: We aim to study many different ARD+ models, as well as varying textual descriptions in order to improve the quality of the presented method. A further goal is the analysis of the relationship between the quality of the textual descriptions (size, term frequencies, etc.) compared to the accuracy of the generated ARD+ model. Furthermore, we want to improve the preprocessing method using sampling techniques. Another promising direction for future work is given by including more background knowledge for further supporting the ontology and rule capture, prototyping, and modeling process.

Acknowledgements

This paper is supported by the *HeKatE* Project funded from 2007–2009 resources for science as a research project, and by the German Research Council (DFG) under grant Pu 129/8-2.

References

[Atzmueller and Puppe, 2005] Martin Atzmueller and Frank Puppe. Semi-Automatic Visual Subgroup Mining using VIKAMINE. *Journal of Universal Computer Science*, 11(11):1752–1765, 2005.

[Atzmueller and Puppe, 2006] Martin Atzmueller and Frank Puppe. SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. In *Proc. 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006)*, pages 6–17, Berlin, 2006. Springer.

[Atzmueller and Puppe, 2007] Martin Atzmueller and Frank Puppe. A Knowledge-Intensive Approach for Semi-Automatic Causal Subgroup Discovery. In *Proc. Workshop on Prior Conceptual Knowledge in Machine Learning and Knowledge Discovery (PriCKL'07), at the 18th ECML/PKDD 2007)*, pages 1–6. University of Warsaw, Poland, 2007.

[Atzmueller and Puppe, 2008] Martin Atzmueller and Frank Puppe. Semi-Automatic Refinement and Assessment of Subgroup Patterns. In *Proc. 21th International Florida Artificial Intelligence Research Society Conference (FLAIRS-2008)*, pages 518–523. AAAI Press, 2008.

[Atzmueller *et al.*, 2005a] Martin Atzmueller, Frank Puppe, and Hans-Peter Buscher. Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In *Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 647–652, Edinburgh, Scotland, 2005.

[Atzmueller *et al.*, 2005b] Martin Atzmueller, Frank Puppe, and Hans-Peter Buscher. Profiling Examiners using Intelligent Subgroup Mining. In *Proc. 10th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005)*, pages 46–51, Aberdeen, Scotland, 2005.

[Atzmueller *et al.*, 2007] Martin Atzmueller, Joachim Baumeister, Peter Kluegl, and Frank Puppe. Rapid Knowledge Capture Using Subgroup Discovery with Incremental Refinement. In *Proc. 4th International Conference on Knowledge Capture (K-CAP 2007)*, pages 31–38. ACM Press, 2007.

[Atzmueller *et al.*, 2009] Martin Atzmueller, Frank Puppe, and Hans-Peter Buscher. A Semi-Automatic Approach for Confounding-Aware Subgroup Discovery. *International Journal on Artificial Intelligence Tools (IJAIT)*, 18(1):1 – 18, 2009.

[Baeza-Yates and Ribeiro-Neto, 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Harlow, 1st edition edition, 1999.

[Biemann *et al.*, 2008] C. Biemann, U. Quasthoff, G. Heyer, and F. Holz. ASV Toolbox – A Modular Collection of Language Exploration Tools. In *Proc. 6th Language Resources and Evaluation Conference (LREC) 2008*, 2008.

[Giarratano and Riley, 2005] Joseph C. Giarratano and Gary D. Riley. *Expert Systems*. Thomson, 2005.

[Han and Kamber, 2000] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publisher, 2000.

[Hayes-Roth *et al.*, 1983] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building Expert Systems*. Addison-Wesley, London, 1983.

[Klösgen and Zytkow, 2002] Willi Klösgen and Jan Zytkow, editors. *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, New York, 2002.

[Klösgen, 1996] Willi Klösgen. Explora: A Multipattern and Multistrategy Discovery Assistant. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI Press, 1996.

[Klösgen, 2002] Willi Klösgen. *Handbook of Data Mining and Knowledge Discovery*, chapter 16.3: Subgroup Discovery. Oxford University Press, New York, 2002.

[KNIME, 2009] KNIME. Knime: Konstanz information miner. www.knime.org, 2009.

[Lavrac *et al.*, 2004] Nada Lavrac, Branko Kavsek, Peter Flach, and Ljupco Todorovski. Subgroup Discovery with CN2-SD. *Journal of Machine Learning Research*, 5:153–188, 2004.

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

[Morgan, 2002] Tony Morgan. *Business Rules and Information Systems. Aligning IT with Business Goals*. Addison Wesley, Boston, MA, 2002.

[Nalepa and Ligęza, 2005a] 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.

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

[Nalepa and Wojnicki, 2008a] Grzegorz J. Nalepa and Igor Wojnicki. Hierarchical Rule Design with HaDEs the HeKatE Toolchain. In M. Ganzha, M. Paprzycki, and T. Pelech-Pilichowski, editors, *Proceedings of the International Multiconference on Computer Science and Information Technology*, volume 3, pages 207–214. Polish Information Processing Society, 2008.

[Nalepa and Wojnicki, 2008b] Grzegorz J. Nalepa and Igor Wojnicki. An ARD+ design and visualization toolchain prototype in prolog. In David C. Wilson and H. Chad Lane, editors, *FLAIRS-21: Proc. 21st Intl. Florida Artificial Intelligence Research Society conference: 15–17 may 2008, Coconut Grove, Florida, USA*, pages 373–374. AAAI Press, 2008.

[Nalepa and Wojnicki, 2008c] Grzegorz J. Nalepa and Igor Wojnicki. Towards formalization of ARD+ conceptual design and refinement method. In David C. Wilson and H. Chad Lane, editors, *FLAIRS-21: Proc. 21st Intl. Florida Artificial Intelligence Research Society conference: 15–17 may 2008, Coconut Grove, Florida, USA*, pages 353–358, Menlo Park, California, 2008. AAAI Press.

[Nalepa and Wojnicki, 2008d] Grzegorz J. Nalepa and Igor Wojnicki. Varda rule design and visualization toolchain. In Andreas R. Dengel and et al., editors, *KI 2008: Advances in Artificial Intelligence: 31st Annual German Conference on AI, KI 2008: Kaiserslautern, Germany, September 23–26, 2008*, volume 5243 of *LNAI*, pages 395–396, Berlin; Heidelberg, 2008. Springer Verlag.

[Negnevitsky, 2002] Michael Negnevitsky. *Artificial Intelligence. A Guide to Intelligent Systems*. Addison-Wesley, Harlow, England; London; New York, 2002. ISBN 0-201-71159-1.

[OMG, 2006] OMG. Semantics of business vocabulary and business rules (sbvr). Technical Report dtc/06-03-02, Object Management Group, 2006.

[Ross and Wright, 2002] Kenneth A. Ross and Charles R. Wright. *Discrete Mathematics*. Prentice Hall, 5th edition, 2002.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[Wermter and Hahn, 2005] Joachim Wermter and Udo Hahn. Finding New Terminology in Very Large Corpora. In *K-CAP '05: Proc. 3rd Intl. Conf. on Knowledge Capture*, pages 137–144, New York, USA, 2005. ACM.

[Wrobel, 1997] Stefan Wrobel. An Algorithm for Multi-Relational Discovery of Subgroups. In *Proc. 1st Europ. Symp. Principles of Data Mining and Knowledge Discovery*, pages 78–87, Berlin, 1997. Springer.