KARaCAs: Knowledge Acquisition with Repertory Grids and Formal Concept Analysis for Dialog System Construction

Hilke Garbe[1], Claudia Janssen[2], Claus Möbus[1], Heiko Seebold[2], and Holger de Vries[2]

[1] University of Oldenburg, Germany
{garbe, moebus}@uni-oldenburg.de
[2] OFFIS Oldenburg, Germany

Abstract. We describe a new knowledge acquisition tool that enabled us to develop a dialog system recommending software design patterns by asking critical questions. This assistance system is based on interviews with experts. For the interviews we adopted the repertory grid method and integrated formal concept analysis. The repertory grid method stimulates the generation of common and differentiating attributes for a given set of objects. Using formal concept analysis we can control the repertory grid procedure, minimize the required expert judgements and build an abstraction based hierarchy of design patterns, even from the judgements of different experts. Based on the acquired knowledge we semi-automatically generate a Bayesian Belief Network (BBN), that is used to conduct dialogs with users to suggest a suitable design pattern for their individual problem situation. Integrating these different methods into our knowledge acquisition tool KARaCAs enables us to support the entire knowledge acquisition and engineering process. We used KARaCAs with three design pattern experts and derived approximately 130 attributes for 23 design patterns. Using formal concept analysis we merged the three lattices and condensed them to approximately 80 common attributes.

1 Introduction

Design patterns are an accepted method for improving the quality of software. The standard book about design patterns in object oriented software design is Gamma et al. [1], where a citation of Christopher Alexander is used to explain what patterns are: " Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice." [2]. Although Alexander was an architect his description is also suitable for patterns in the domain of object oriented software design. One purpose of design patterns is to capture design experience and make it available to other developers, so they can improve their designs. The standard elements of a pattern description are:

S. Staab and V. Svatek (Eds.): EKAW 2006, LNAI 4248, pp. 3–18, 2006.

Name. Every design pattern has a name.
Problem. This section describes the problem situation in which a pattern is applicable.
Solution. The solution describes how objects and classes can work together to solve the problem. The solution is a template that has to be instantiated for each specific situation.
Consequences. The consequences describe e.g. trade-offs that have to be considered if a design pattern should be applied.

Gamma et al. [1] describe 23 design patterns organized as a catalog. In addition there is an increasing amount of new patterns including for example architectural and J2EE patterns. Nevertheless, knowledge about design patterns is not very wide spread among software developers and the literature about patterns is often organized as catalogs; pattern by pattern. A software engineer has to read all (or at least many) pattern descriptions before he can decide which pattern is suitable for his problem situation. Tool support for selecting appropriate design patterns is rare. Meffert [3] supposes semantic source code annotations which can be analysed to capture the intent of a piece of software. These annotations can be compared to given pattern templates and if a match can be found, the source code elements can gradually be transfered to a pattern. Gomes [4] describes a case based reasoning approach for reusing software which uses Baysian Networks and WordNet as a common sense ontology. Both approaches work on source code or UML model artifacts. Our goal was to develop a dialog system that is able to approve possible design patterns [5] without analysing given source code or models. This dialog system can assist software developers in choosing a design pattern. Who, even if he has not read all the design pattern descriptions is able to benefit from the experiences of other software developers. Our dialog system questions him about his design problem and makes a suggestion which pattern might be applicable.

This article is organized as follows: Section 2 provides an overview of our proposed knowledge acquisition and engineering process. Section 3 describes our knowledge acquisition procedure containing repertory grids (3.1), formal concept analysis (3.2), and their integration in KARaCAs (3.3). The merging process for the results of different expert interviews is outlined in section 4. Our dialog system is based on BBNs (section 5) that are generated semi-automatically from the data we obtained in the expert interviews with KARaCAs. In section 6 we discuss conclusions and further work.

2 Requirements and Proposed Knowledge Acquisition Process

Our goal was to design an efficient dialog system that assists software developers to choose a design pattern for their specific problem situation. The intended dialog system should question the user about his specific problem situation. The user answers these questions with “yes”, “no” or “I don’t know”. During this dialog the system successively reduces the set of applicable patterns until it proposes one.

This dialog system has to fulfill the following requirements:

1.1 **User input.** Requirements on how user input is processed:
  1.1.1 **Undo.** The user should have the possibility to reconsider a given answer and change it. The dialog has to adjust itself to the altered information.
  1.1.2 **Fault Tolerance.** The dialog has to be "fault tolerant". Even if a user gives a wrong answer about his problem situation he should have the possibility to continue the dialog and get a pattern suggestion.
  1.1.3 **No answer.** Additionally, he should be able to skip questions, since he might not be able to classify his problem situation completely and therefore might not be able to answer every question.
1.2 **Probabilities for suggestions.** The dialog system may not be able to comprehend the entire problem situation and context. This may be because the user has a diverging conception of his situation or the posed question and therefore enters misleading information in the dialog system. Another reason might be that there are institutional programming standards that are unknown for the dialog system. Because of this, a certain pattern can only be suggested with an approximated probability. The user should be informed about this probability. This information should be presented to him during the entire dialog.

As we were not able to extract the required knowledge from design pattern literature, we used a knowledge acquisition with design pattern experts. Deduced from the requirements for the dialog, we had the following requirements for the knowledge acquisition procedure:

2.1 **Free generation and naming of attributes.** The experts should not be restricted in generating and naming attributes for the problem situations a specific pattern can be applied to. We were interested in the terms experts use to describe problem situations for software engineers.
  2.1.1 **Shared attributes.** To set up an efficient dialog, attributes were required that are shared between two or more patterns. These attributes can be used to successively reduce the set of patterns that are relevant for the ongoing dialog. This is a very difficult task for design pattern experts. We made the experience that they first think of specific problem situations for every pattern, and finding shared attributes was a difficult task for them.
  2.1.2 **Differentiating attributes.** To identify a situation in which one specific design pattern can be suggested by the dialog system, every pattern should be identifiable by its attributes. Therefore the attribute sets of the patterns should be pairwise disjoint.
2.2 **Probabilities.** A design pattern is a template that can be adjusted to a given situation and the situations to which it can be applied can vary. Therefore the attributes of these situations can vary and so a specific attribute must not always be present. To capture this, we needed a method that can deal with probabilities.
2.3 **Visualization of results.** The results of the knowledge acquisition should be visualized in such a way, that the domain expert can understand them.

2.4 **Merging.** As we wanted to integrate the knowledge of different experts who can freely name attributes, the knowledge acquisition procedure should allow the possibility to merge different sets of attributes for the design patterns. This includes for example the assistance in identifying attributes that are used synonymously, as generalisation or specialisation of others. Because of the amount of attributes we have to deal with, this procedure has to be supported by the knowledge acquisition method.

To fulfill these requirements we propose a three stage knowledge acquisition process (Fig. 1). We first acquire knowledge from different experts. For these interviews we adapted the repertory grid technique and integrated Formal Concept analysis. The results of these interviews are merged with support of Formal Concept Analysis. From this aggregated information we semi-automatically generate a BBN for the dialog system. Using these different knowledge acquisition methods and engineering and integrating them into our tool KARaCAs we can support the whole process.

The three stages of the process will be described in sections 3 - 5.

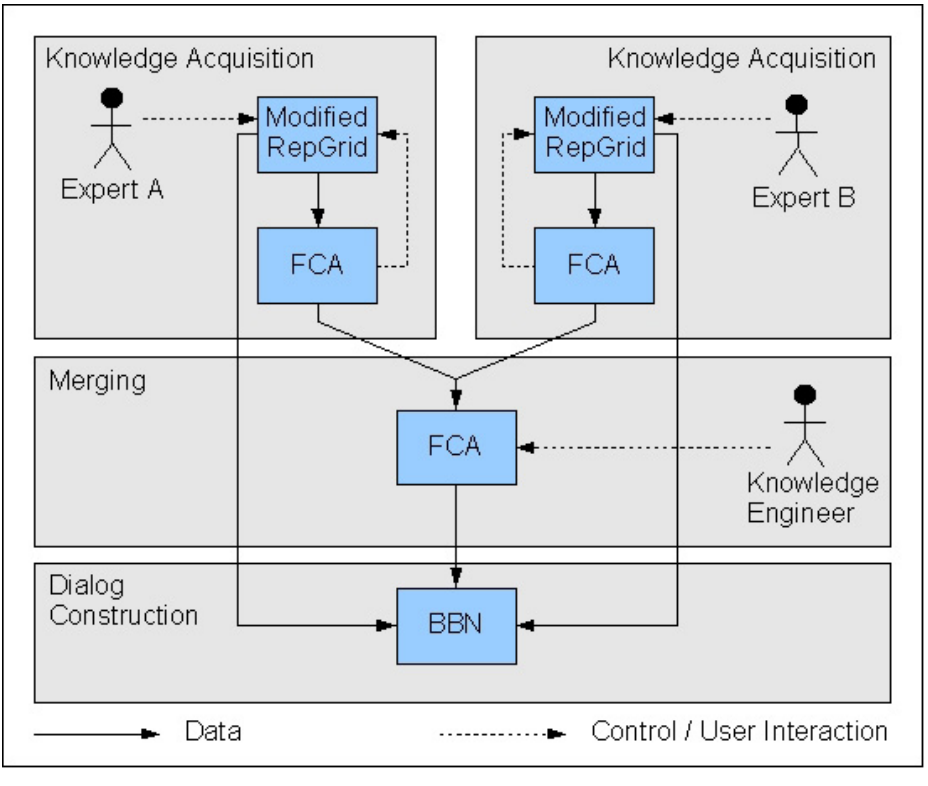

Fig. 1. Proposed Knowledge Acquisition Process

3 Knowledge Acquisition

For our knowledge acquisition process we used modified repertory grids and formal concept analysis. Both methods have distinct advantages but also some strong disadvantages in regard to our requirements. By combining the two the disadvantages for our application could be overcome.

3.1 Knowledge Acquisition with Repertory Grids

The Repertory Grid techniques was proposed by Kelly [6]. It was embedded in his Personal Construct Psychology. Delugach states: "In accordance with the theory, the repertory grid technique distinguishes the objects of a problem domain (called elements) through their attributes (called constructs)." [7]

The Repertory Grid technique has three steps.

1. The elements for the procedure are chosen. In Personal Construct Psychology the considered elements are usually persons or situations. Often elements are preselected instead of being acquired through an interview.
2. At the second step constructs are acquired by classification from three randomly chosen elements (triade) into two classes. The interviewed person has to explain in which way two elements of the triade are similar and how they differ from the remaining one. Elements and constructs are listed in a two-dimensional matrix called grid.
3. The last step is to rate all elements concerning the given constructs. This is normally done by rating each element on a given scale between the two poles.

Gaines and Shaw state: "The repertory grid was an instrument designed by Kelly to bypass cognitive defenses and give access to a persons underlying construction system by asking the person to compare and contrast relevant examples (significant people in the person's life in the original application)." [8]

Since 1955 several variations of the repertory grid technique have been developed, Castro-Schez et al. [9] give a short overview. Apart from the use in psychology (e.g. Spangenberg and Wolff [10]) Repertory Grids and its variations have often been used for knowledge acquisition in different domains (e.g. Gaines and Shaw [8], Richards [11] and Castro-Schez et al. [9]).

The repertory grid technique has a lot of advantages that makes it well suited for our knowledge acquisition process:

- It can be performed in natural spoken language without given items (requirement 2.1). Nevertheless we obtain formal and structured results which can be used as input in other procedures.
- The method supports the experts in generating shared (requirement 2.1.1) and differentiating (requirement 2.1.2) attributes for design patterns. By asking them to group two of three items they can concentrate on these. At each step only a small subset of the design patterns has to be considered. Therefore the expert is supported in generating these differentiating attributes.

– The two poles of every construct can be interpreted as attributes. By rating the attributes for the design patterns probabilities can be assigned to the relations (requirement 2.2).

There are some disadvantages:

– The triads are randomly chosen. In the worst case this leads to a very large amount of needed triads to get enough differentiating attributes. In addition, the repertory grid method can not assure that the sets of attributes for all design patterns are pairwise disjoint. (Requirement 2.1.1) If for example two very similar elements are presented in a triad, it can be expected that they are always grouped together (cp. Choisel and Wickelmaier [12]).
– The generated attributes depend on the presented triads, so possibly not every attribute the expert thinks to be important might be generated. If for example all elements have a very important attribute in common this would never be mentioned during the procedure. Considering the attributes that are generated during the repertory grid for a specific design pattern the expert might miss one or more attributes that are relevant for this pattern. The expert should have the possibility to complete the set of attributes to properly describe the design pattern.
– Bruder, Lengnink and Prediger [13] used repertory grids to ask mathematics students about their subjective theories about mathematical tasks. As a result of their studies they describe an additional problem: It was often very difficult for the students to find exact antipodes for attributes.

The result of the repertory grid method is a matrix with ratings for the elements concerning the constructs. This data has to be analysed and visualized properly to be understood and interpreted by the domain expert and the knowledge engineer. Several methods have been used for this and were implemented in tools (cp. Gaines and Shaw [8]). In our work we focus on Formal Concept Analysis for this purpose.

3.2 Ontology Engineering and Formal Concept Analysis

This section gives a short introduction to formal concept analysis. The combination of repertory grids and formal concept analysis is described in section 3.3. Formal Concept Analysis (FCA) is a method for qualitative analysis of data. Subject to FCA is a formal context. A formal context (G, M, I) is a subset of the Cartesian product ($I \subseteq G \times M$) of a quantity of objects G (Gegenstände in German) and a quantity of attributes M (Merkmale in German). The term $g\,I\,m$ means: object g owns attribute m [14]. FCA structures data into units, which are formal abstractions of concepts of mind. A formal concept comprises two parts, its extension and its intension. The extension enfolds all objects belonging to this concept. The intension covers all attributes shared by those objects. The extension of a concept determines the intension and the intension determines the extension [14]. This approach allows gathering all concepts of a context and introduces a subsumption hierarchy between them. The amount of all formalized

concepts is called concept lattice of the formal context. This lattice can be visualized as a conceptual hierarchy (Hasse Diagram), which enables a different view on the structure of the data and supports its analysis. An area of application for FCA is Ontology Engineering by utilizing the ability to structure data by means of concept lattices. The generated hierarchies may be used as a starting point for the manual or semi-automatic creation of ontologies [15]. The concept lattice enables domain experts to identify incorrectness or missing coherence in the dataset (requirement 2.3). Gaps in the conceptual hierarchy indicate probable missing objects or attributes. With these hints the data ascertainment can be completed [16]. Because the concept lattices represent the data in a way domain experts intuitively understand [16,17].

3.3 Integrating the Methods - KARaCAs

A combination of repertory grids and FCA has already been used in different domains.

Spangenberg and Wolf used FCA [10] to reduce the amount of data acquired with repertory grids and discuss alternative approaches. They transform the repertory grid data into a multivalued context, which has to be reduced to an univalent context. Their repertory grid has a rating scale from 1 to 6. They interpret votes with values from 3 to 4 as indecisiveness and ignore those votes.

Bruder, Lengnink and Prediger [18] used line diagrams produced by FCA to visualize the structure of the repertory grid results. They asked mathematics students about their subjective theories about mathematical tasks and investigated the change over time. Based on their studies they describe two major problems [13]:

1. The set of objects (tasks) has a direct influence on the acquired attributes, therefore it must be possible for the students to add additional relevant attributes after the repertory grid procedure is performed.
2. Often it was very difficult for the students to find exact antipodes for attributes.

Delugach and Lampkin presented a method for knowledge acquisition using repertory grids, Formal Concept Analysis and Concept Graphs. They used: "repertory grids for acquisition, formal concept analysis for analysis, and conceptual graphs for representation."[7]

These groups mainly used FCA to analyse the results of repertory grids. They first questioned experts with repertory grids and analysed the data using formal concept analysis after the interviews. We integrated the two methods for a better use of the advantages and to overcome some of the disadvantages. We developed two versions of KARaCAs with an ascending level of integration.

We adapt the repertory grid method to allow attributes that are not exact opposite of each other (cp. Bruder et al. [13]). Steps 1 and 2 are performed the same way as describe in section 3.1. Step 3 is modified as follows:

3.1 First the expert is asked to assign one of the two attributes to each object (pattern in our case), if possible. This is done due to the fact that these two

attributes are not necessarily the opposite of each other and neither may apply in some cases.

3.2 In the next step the expert has to quantify how certain a given attribute applies to a pattern. In contrast to Kelly's repertory grid method this evaluation is not done with the two attributes as boundary or poles.

Similar to the card sorting method Kelly proposed for the Grid we designed our graphical user interface using the card sorting metaphor (Fig. 2).

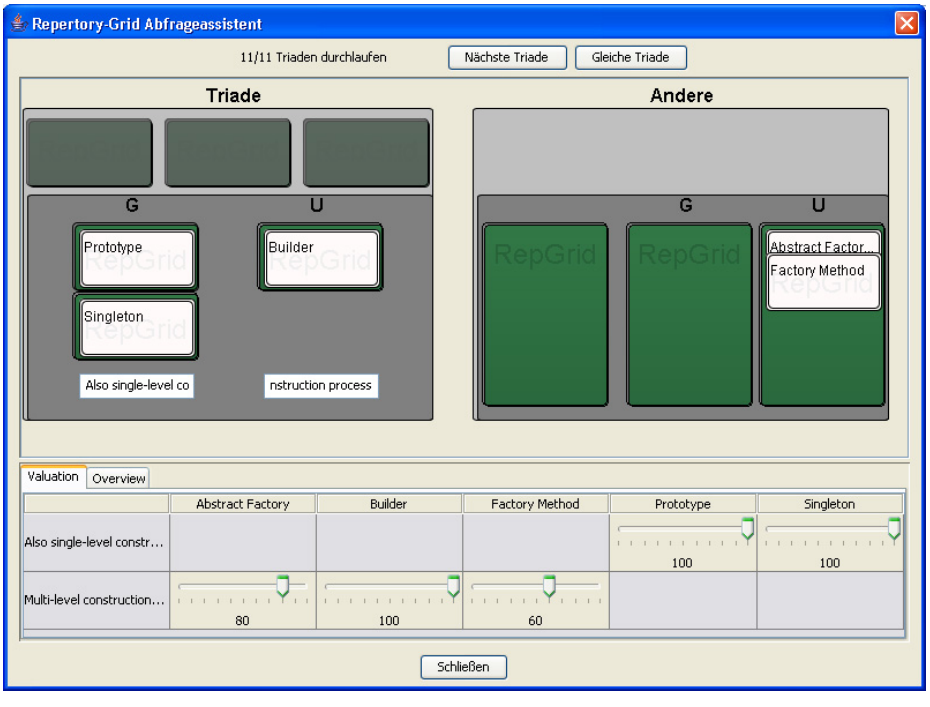

Fig. 2. KARaCAs: GUI for the adapted Repertory Grid

After each triad KARaCAs analyses the data with formal concept analysis. The elements and the associated attributes are transferred to a formal context. In the formal context each attribute is set in relation to an element it is assigned to. From this context a concept lattice is generated. The lattice in Fig. 3 shows a part of the generated attributes from an expert interview.

This new integration of FCA and Repertory Grid in one tool has two distinct advantages:

On the one hand performing the formal concept analysis after each triad gives the domain expert the possibility to inspect the results in form of a lattice (Fig. 3, Requirement 1.7) even during the interview with the repertory grid. By integrating both methods into one tool it is possible to switch between two different views of the acquired data, resulting in synergistic effects. The knowledge

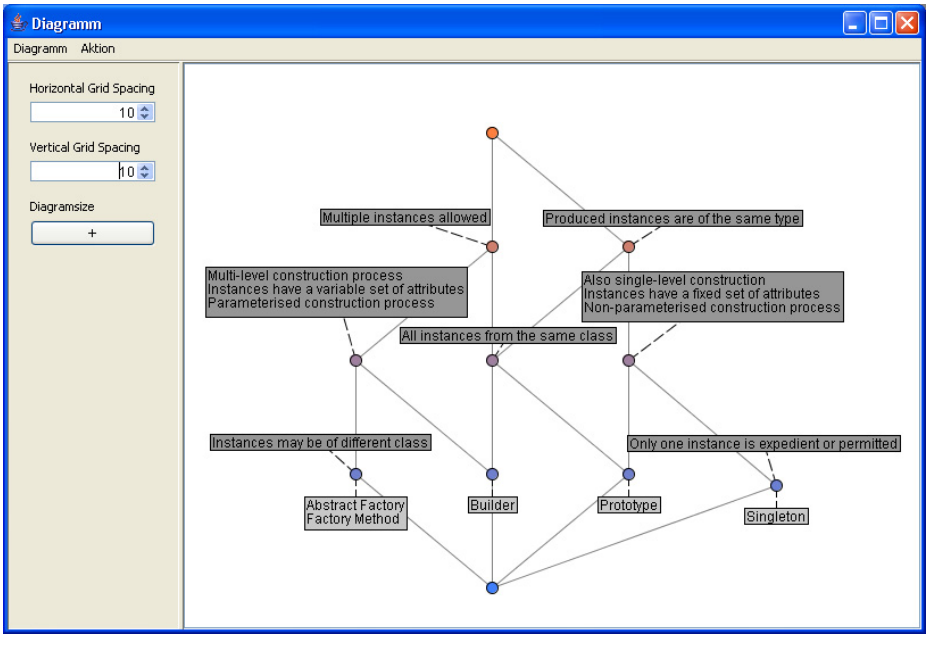

Fig. 3. KARaCAs: Corresponding Line Diagram for the context

engineer is able to present the lattice representing the so far obtained attributes to the expert during the acquisition and discuss it. The expert can see if a sufficient set of attributes was already acquired. The lattice in Fig. 3 e.g. shows that the design patterns Abstract Factory and Factory Method can not be distinguished by the attributes given so far. If the expert identifies such missing attributes while looking at the graph (compare Bruder, Lengnink and Prediger [13]) they can be directly added to the formal context and are also available after switching back to the repertory grid. The first version of KARaCAs contained this integration level of the two methods. It was used to acquire approximately 130 attributes from three different experts. Each of the interviews lasted about an hour which is a relatively short time span for the given task and shows the data acquisition efficiency of our method.

Furthermore, we can control the ongoing repertory grid. The second version of KARaCAs analyses the data with respect to the following questions:

1 Are there any two or more elements with attribute sets that are not pairwise disjoint?
In this case, the next triad will be chosen so that the domain expert is forced to generate a differentiating attribute.
1.1 Are there three or more elements with equal sets of attributes?
The algorithm produces a triad containing three of these elements. Doing so, the expert is forced to distinguish one of these elements from the others and generate an appropriate attribute.

1.2 Are there two elements with equal sets of attributes?
A third element for the triad has to be chosen that has a maximum amount of attributes in common with the other two elements. Otherwise it is likely that the two remain grouped together and the third one is separated. We choose an element contained in a superconcept node or a subconcept node of a shared superconcept node as third element because they have the most attributes in common. It is assumed that elements contained in concepts near to each other in the concept lattice are more similar than object in concepts far from each other.

1.2.1 Are there two elements with equal sets of attributes that have been presented together in five triads without being separated? Or is no new triad possible containing these two elements?
The elements are presented as pairs and separating attributes have to be generated. At this point we use this limit because we assume that the two are very similar in relation to the remaining elements and would always be grouped together when presented with a third element.

2 If the attribute sets for all elements are pairwise disjoint no new triad is presented.

Using this algorithm we can ensure that the acquired attribute sets for all elements are pairwise disjoint at the end of the procedure. By choosing elements as similar as possible for triades we increase the efficiency of the repertory grid technique in regard to our requirements and minimize the required expert judgements.

4 Merging

Ontology merging is widely discussed and a lot of tools are developed to support this process (Stumme and Maedche [19] give a short overview). Ganter and Stumme describe the general task of ontology merging as follows: "Merging two ontologies means creating a new ontology in a semi-automatic manner by merging concepts of the source ontologies." [20] Stumme proposed the use of FCA for this purpose [19,21]. Our acquired source ontologies share the same objects: the given design patterns. Therefore, we only have to merge the acquired attributes for these objects. KARaCAs enables us to merge the results of different expert interviews (requirement 2.4) by aggregating the attribute sets in one large formal context. From this joint context a line diagram is generated, which helps the knowledge engineer to analyse the attributes.

We primarily investigated the lattices concerning the following two questions:

– Are different attributes used synonymously?
If this is the case, one (or more) of them is redundant and is deleted from the merged context. In the corresponding lattice these attributes would be annotated to the same nodes (concepts).

– Do different experts assign very similar attributes A and B to different design pattern sets?

  This might occur if an attribute is only weakly connected to a specific design pattern. One expert might assign this attribute with a low probability to a specific design pattern and the other does not connect the two. Another possibility is that one expert forgot that an attribute is relevant for a design pattern. In this case, starting at the node that A is assigned to B would be annotated to a node on an upwards or downwards path.

Supported by this feature we condensed the approximately 130 acquired attributes to about 80 which form the basis for our dialog system.

5 Knowledge Representation with Bayesian Belief Networks

Bayesian Belief Networks ([22,23]) are the representation of choice for modeling uncertain knowledge (e.g.[24,25,26,27]). A BBN models this knowledge as a directed acyclic graph that represents a probability distribution. The nodes of the graph represent propositional variables and directed arcs represent probabilistic relationships between them. Probabilistic independence between variables is indicated by the types of path in the network and the lack of them. Furthermore, the relations are conditional probabilities (each variable conditioned on its parents in the network) that define a joint probability distribution of the variables.

We used BBN to express knowledge about the applicability of design patterns for a given problem situation. For this purpose the qualitative structure of the BBNs is grouped in two levels (see Fig. 4): the first level contains the actual design patterns while the second contains the attributes of a problem situation. Due to this template it is possible to create the qualitative and quantitative structure of the BBN automatically from the acquired data as presented in the following section.

5.1 Generating Bayesian Belief Networks

The generation of the BBN was performed in two steps: In the first step the qualitative structure of the network was created. The acquired attributes assigned to the corresponding design patterns and the necessary conditional probabilities were identified. For each attribute assigned to a design pattern in KARaCAs a relation between these two was created in the qualitative network. This step could be done automatically by KARaCAs. In the second step a quantitative structure of the networks was build. While acquiring the different attributes to the design patterns the experts were asked to judge the probability of the relation between design patterns and attribute. This probability statements were taken as simple conditional probabilities like $P(attribute\ A = yes | design\ pattern\ singleton = yes) = 0.9$, i.e. the expert judged that in 90% of the cases where a singleton is used the attribute A applies. However, to build the quantitative structure of the network more complex conditional statements which expressed the validity of an

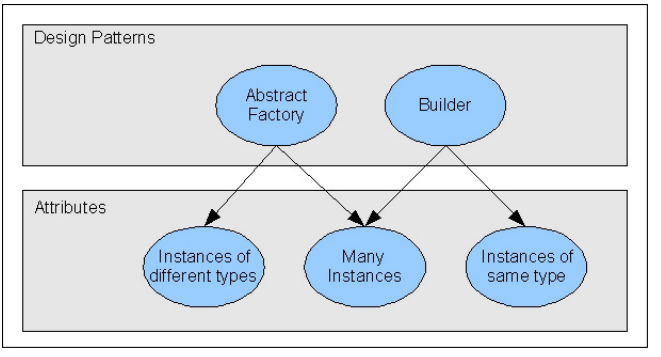

Fig. 4. Small Part of the generated Bayesian Belief Network

attribute under the influence of several design patterns were necessary. These probabilities could be calculated by using the "noisy-or" ([23]) by making some assumptions:

1. The chance that an attribute is valid without any of the regarding design patterns present is 0.05. This states how probable it is for an attribute to apply, if no design pattern is given. To get more accurate values for these probabilities it would be necessary to perform a more general statistical study about the frequency of occurrence of the attribute without a design pattern associated.
2. We assume that the factors inhibiting the influence of design patterns on an attribute are independent from each other.
3. In the early stage of the project the experts had to identify attributes for design patterns without stating the probability of the relation. Because in this stage the experts were asked to state only attributes which certainly apply, we assumed the probability of the relations to be 0.9.

Under these assumptions we were able to calculate the necessary conditional probabilities from the expert statements by using the "noisy-or" approach ([23]). Finally we had to obtain the a-priori-probability of the design patterns. Since we had no information if a design pattern is appropriate for a given problem, we assumed it has a 50% chance of being applicable at the beginning. To get a more accurate value for this probability a general statistical study about the frequency of application of the design patterns has to be performed. The calculation of the necessary probabilities is done automatically by KARaCAs.

5.2 A Dialog System Using Bayesian Belief Networks

The generated BBN is the basis for our dialog system. This system questions the user successively about his problem situation using the acquired design pattern attributes. The dialog is performed using the following algorithm:

1. Determine the design pattern node with the highest probability for the state "yes". If this probability is higher than 0.9 recommend the design pattern as applicable to the user. If the probability is lower proceed with the algorithm.

2. Determine the relevant attribute nodes for this design pattern node. This is done using Shachter's Bayes Ball algorithm [28].
3. Determine the node with the highest probability for "yes" from this set and question the user about the associated attribute of this node.
4. If the user answers "yes" or "no" enter this as evidence in the BBN and perform an inference on the net. If the user answer with "I don't know" mark the node as being asked already.
5. Restart with 1.

By always determining the design pattern that is applicable with the highest probability and asking about it's attributes we try to conduct the dialog according to the users expectations. We assume this dialog strategy minimizes questions that astonish the user because they are not related to his problem situation. Using BBN for the dialog we can fulfill the requirements listed in section 2. Each answer a user gives is entered as evidence in the BBN and the new probabilities for all design patterns and attributes are calculated. An undo (requirement 1.1.1) function for user inputs is easily implemented by deleting one piece of evidence and doing an inference on the BBN.

If probabilities of 0 or 1 are avoided in the specification of the attribute nodes the dialog can be made fault tolerant for unexpected user input (requirement 1.1.2). If a user answers for example "no" where the situation would imply a "yes" the applicable design pattern would get a very low possibility. But after the user answered the following questions (that try to establish an alternative design pattern) also with "no", the suitable pattern is assumed to get the pattern with the highest probability again. In this case the dialog will last longer than in the best case, but the user has the possibility to get a suggestion. If the user is not quite sure about how to answer a certain question he also has the possibility to say "I don't know" (requirement 1.1.3). The attribute node in the BBN is then marked and will not be answered again, but no evidence is entered into the BBN. Because new evidence is directly propagated through the network, it is possible to present a ranking list of the design patterns after each answer. The user always has an overview of the probabilities with which each pattern is suitable concerning his so far entered information.

Using Bayesian Belief Networks as knowledge representation for the dialog enabled us to fulfill the requirements from section 2. Alternative representations could have been for example the concept lattice produced by FCA or a decision tree. But both can't fulfill our requirements. Especially the representation of probabilities and the required opportunity to change any already given answer during the dialog is difficult to implement with these techniques.

6 Conclusion and Further Work

6.1 Conclusion

A knowledge acquisition method was presented that combines repertory grids and formal concept analysis on two ascending levels of integration. The first

version of KARaCAs was used to acquire approximately 130 attributes from three design pattern experts. Each of the three interviews only lasted about one hour. During the interviews the visualization was used to analyse the data so far obtained. The concept lattice was easy to understand for the experts and helped them to reflect on their answers. Based on the experiences from these interviews we developed the second version of KARaCAs. The results of the three interviews were merged as described and condensed to about 80 attributes. These attributes are the basis for our dialog system. KARaCAs supports the entire knowledge acquisition process and automatically generates a BBN from the data that is used in the dialog system.

6.2 Further Work

The dialog system will be evaluated with computer science students at the University of Oldenburg. A special question for this evaluation is whether the attributes given by design pattern experts can be understood by beginners. Another one is the suitability of the dialog strategy which determines the questions that are presented to the user. To further support the experts in generating attributes we plan to do interviews with a dyad of experts. A small test with two experts showed that they discuss the grouping of design patterns and the naming of the attributes a lot. It seemed promising to do interviews with a peer group of experts to increase the quality of the acquired data. Furthermore we want to expand the dialog system. Our focus is on the domain dependent applicability of design patterns.

Acknowledgements

The development of KARaCAs and the dialog system was embedded in the multi-partner project InPULSE which was granted by the BMBF (German Federal Ministry of Education and Research). We thank Jan-Patrick Osterloh and Lars Weber for implementing our ideas and algorithms in KARaCAs and Steffen Kruse and Malte Zilinski for critical comments.

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addision-Wesley (1995)
2. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language. Oxford University Press, New York (1977)
3. Meffert, K.: Supporting design patterns with annotations. In: 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06). (2006) 437–445
4. Gomes, P.: Software design retrieval using bayesian networks and wordnet. In: Proceedings of the 7th European Conference on Case-Based Reasoning, ECCBR 2004. (2004) 184–197

5. Möbus, C., Seebold, H., Garbe, H.: A greedy knowledge acquisition method for the rapid prototyping of knowledge structures. In Clark, P., Schreiber, G., eds.: Proceedings of the 3rd International Conference on Knowledge Capture, 2005, New York, NY: ACM Press (2005) 211 – 212
6. Kelly, G.A.: Psychology of Personal Constructs. New York: W. W. Norton (1955)
7. Delugach, H., Lampkin, B.: Troika: Using grids, lattices and graphs in knowledge acquisition. In Stumme, G., ed.: Working with Conceptual Structures: Contributions to ICCS 2000, Aachen, Germany: Shaker Verlag (2000) 201–214
8. Gaines, B., Shaw, M.: Knowledge acquisition tools based on personal construct psychology. The Knowledge Engineering Review **8**(1) (1993) 49–85
9. Castro-Schez, J.J., Jennings, N.R., Luo, X., Shadbolt, N.: Acquiring domain knowledge for negotiating agents: a case study. International Journal of Human-Computer Studies **61**(1) (2004) 3–31,
10. Spangenberg, N., Wolff, K.: Datenreduktion durch die Formale Begriffsanalyse von Repertory Grids. In: Einführung in die Repertory Grid-Technik, Band 2, Klinische Forschung und Praxis. Bern, Göttingen, Toronto, Seattle: Verlag Hans Huber (1993) 38–54
11. Richards, D.: Ripple-down rules with formal concept analysis: A comparison to personal construct psychology. In Gaines, B., Musen, M., eds.: Proceedings of 11th Workshop on Knowledge Acquisition, Modeling and Management, Banff Canada, SRDG Publications, Calgary, Canada (1998)
12. Choisel, S., Wickelmaier, F.: Extraction of auditory features and elicitation of attributes for the assessment of multichannel reproduced sound. In: 118th Convention of the Audio Engineering Society, Barcelona, Spain (2005)
13. Bruder, R., Lengnink, K., Prediger, S.: Ein Instrumentarium zur Erfassung subjektiver Theorien über Mathematikaufgaben. Preprint Nr. 2265 des Fachbereichs Mathematik, TU Darmstadt (2003)
14. Ganter, B., Wille, R.: Formal Concept Analysis. Mathematical Foundations. Springer, Berlin, Heidelberg, NewYork (1999)
15. Cimiano, P., Hotho, A., Stumme, G., Tane, J.: Conceptual knowledge processing with formal concept analysis and ontologies. In: Proceedings of the Second International Conference on Formal Concept Analysis - ICFCA04. (2004) 189 – 207
16. Kollewe, W.: Begriffliche Wissensverarbeitung: Wie Begriffsstrukturen die Pflege und Recherche in Wissensdatenbanken unterstützen. In: Bitkom KnowTech. (2002)
17. Düwel, S.: BASE - ein begriffsbasiertes Analyseverfahren für die Software-Entwicklung. PhD thesis, Philipps-Universität Marburg (2000)
18. Lengnink, K., Prediger, S.: Development of the personal constructs about mathematical tasks - a qualitative study using repretory grid methodology. In: Proceedings of the 27th Annual Meeting of the International Group for the Psychology of Mathematics Education (PME), Hawaii (2003)
19. Stumme, G., Maedche, A.: FCA - MERGE: Bottom-up merging of ontologies. In: IJCAI. (2001) 225–234
20. Ganter, B., Stumme, G.: Creation and merging of ontology top-levels. In de Moor, A., Lex, W., Ganter, B., eds.: Conceptual Structures for Knowledge Creation and Communication, 11th International Conferebce on Conceptual Structures, ICCS 2003, Proceedings, Springer (2003) 131 – 145
21. Stumme, G.: Ontology merging with formal concept analysis. In Kalfoglou, Y., Schorlemmer, M., Sheth, A., Staab, S., Uschold, M., eds.: Semantic Interoperability and Integration. Number 04391 in Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany (2005)

22. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Revised second printing edn. Morgan Kaufman Publishers, San Mateo, CA. (1998)
23. Jensen, F.: Bayesian Networks and Decision Graphs, Statistics for Engineering and Information Science. Berlin: Springer (2001)
24. Folckers, J., Möbus, C., Schröder, O., Thole, H.J.: An intelligent problem solving environment for designing explanation models and for diagnostic reasoning in probabilistic domains. In Frasson, C., Gauthier, G., Lesgold, A., eds.: Intelligent Tutoring Systems. LNCS (1086), ITS 96, Montreal, Canada, Berlin: Springer (1996) 353–362
25. Mislevy, R., Almond, R.G., Yan, D., Steinberg, L.: Bayes nets in educational assessment: Where do the numbers come from? CSE Technical Report 518, Center for the Study of Evaluation, University of California, Los Angeles (2000)
26. Bunt, A., Conati, C.: Assessing effective exploration in open learning environments using bayesian networks. In Cerri, S.A., Gouardres, G., Paraguacu, F., eds.: Intelligent Tutoring Systems, Berlin: Springer (2002) 698 – 707
27. Zapata-Rivera, J., Greer, J.: Student model accuracy using inspectable bayesian student models. In Hoppe, U., Verdejo, F., Kay, J., eds.: Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies, Amsterdam: IOS Press (2003) 65 – 72
28. Schachter, R.D.: Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In Cooper, G.F., Moral, S., eds.: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98), Morgan Kaufmann (1998) 480487