Experience Management with Task-Configurations and Task-Patterns for Descriptive Data Mining

Martin Atzmueller

University of Würzburg
Department of Computer Science VI,
Am Hubland, 97074 Würzburg, Germany
atzmueller@informatik.uni-wuerzburg.de

Abstract. Determining and instantiating an appropriate data mining task and method is often rather difficult, especially for inexperienced users. In this paper, we present a methodological approach for capturing, reusing, and generalizing experiences (task-configurations) for an easier selection and instantiation of appropriate methods and tasks in descriptive data mining. We show how the cases describing the task-configurations are structured, how they are retrieved, applied, and maintained. Furthermore, we present a process model that integrates the proposed techniques.

1 Introduction

In general, task selection (e.g., classification, clustering, description) and method selection in data mining and knowledge discovery are the critical steps for successfully solving the data mining problem at hand. However, these selection problems are often quite difficult [1] – especially for inexperienced users.

Usually, an abstract task description given by the user needs to be mapped to a concrete data mining task, i.e., to a specific supervised or unsupervised approach. After the task selection, an appropriate data mining method needs to be determined: Method selection significantly depends on the application context and on domain knowledge of the specific application provided by the domain specialist. This matching process needs to incorporate the requirements of the user in order to successfully match the given (abstract) task description to a complex real-world data mining task. After the selection, the specific task and method also needs to be instantiated, e.g., suitable parameters must be determined for configuring the selected method. This provides for another dimension before the data mining task can be performed.

Knowledge-based task and method selection is still difficult using explicitly formalized knowledge [1]. Since in practice the transformation process is often learned from experience, a promising approach is given by case-based methods: For task and method selection the application restrictions are mapped to a complex task specification containing the applied method, its characteristics, the parameter configuration of the applied method, and additional information. The contained configuration and additional information (artifacts) then instantly provide the opportunity for supporting the instantiation and application of the selected task and method.

In this paper we present a methodological approach that reuses experiences for the selection and instantiation of tasks and methods in descriptive data mining: We utilize stored cases (experiences) for selecting an appropriate task and method, reusing those stored task-configurations that are similar to a (partially) defined characterization. The contained configurations are furthermore applied for supporting the instantiation of the selected task and method. This way, we provide a guided approach for knowledge extraction and knowledge discovery. Similar to design patterns in software-engineering we also propose generalized experiences, i.e., task-patterns. These describe established best-practices for specific situations and can then be applied as templates.

In the context of experience management [2–4] and case-based reasoning [5], cases contain specific knowledge of previously experienced, concrete problem situations [6]. A case consists of a problem characterization, and additional attached artifacts, for example, meta-information such as the description of the context of the case [7]. In this paper, the applied cases (experiences) are represented by extended task-configurations that capture the knowledge and concrete experiences of specific data mining task and additional descriptive meta-information.

The user can browse (a selection of) the cases (task-configurations and task-patterns) contained in the experience base, retrieve most similar cases for a specific task description (problem), or can be guided in a mixed-initiative consultation mode for obtaining a set of similar cases. After an (adapted) task-configuration has been determined, it can be applied as a template for a new task, or it can just be reproduced, e.g., when using a new data set, or for multi-centric studies in the medical domain. The approach was motivated by experiences in several knowledge discovery projects, e.g., [8–10], using the VIKAMINE [11] system for knowledge-intensive subgroup mining.

The rest of the paper is organized as follows: Section 2 contains a detailed description of the proposed approach. We first describe the process model for capturing and reusing extended task-configurations. After that, we show how to structure, store, retrieve and reapply stored cases, and also discuss maintenance issues. Next, Section 3 presents a discussion of the proposed approach including related work. Finally, we conclude the paper in Section 4 and show promising directions for future work.

2 Capturing and Reusing Task-Experiences

In general, task selection considers the determination of a concrete data mining task, e.g., considering supervised or unsupervised approaches. Each task is a kind of problem, so different tasks require different kinds of algorithms and methods. Exemplary data mining tasks include *data analysis*, *discovery of associations*, *classification*, *clustering*, and *information retrieval*. After task selection has been performed, method selection can be applied, e.g., for selecting the *APRIORI* algorithm [12] for the task of discovering association rules. However, method selection itself can be considered as a subproblem of task instantiation, that is, determining a specific method for a given task. Furthermore, task instantiation also requires the *configuration* of the selected method, e.g., by supplying suitable parameters of the method. In general, task selection and instantiation significantly depend on the application context and on background knowledge. Both processes need to incorporate the requirements of the user that are often

learned from previous experiences [1]. The presented approach aims at task selection and instantiation for descriptive data mining. We aim to capture and to reuse descriptions of previous knowledge discovery tasks – as task-configurations, that is, cases describing configured data mining tasks tat are stored in an experience base. After the cases have been collected, they can be refined/generalized to task-patterns, i.e., abstracted cases by experienced user. These generalized experiences can then easily be used as templates, implementing a rapid-prototyping approach.

In the context of this paper, we aim to match an abstract task to a concrete task-configuration implementing a specific data mining method and its configuration. Therefore, we do not apply a specific method selection step since we perform a combined task and method selection, when considering a given task-configuration.

Some examples of abstract tasks that are available in the VIKAMINE [11] system for knowledge-intensive subgroup mining (e.g., [8]), or in the well-known WEKA [13] data mining workbench are given by

- *Statistical description*, e.g., implemented using methods for simple frequency analysis, OLAP-techniques, or simple correlation analysis.
- *Discovering associations*, e.g., using basic subgroup discovery, or applying methods for discovering association rules (e.g., utilizing the *APRIORI* algorithm [12]).
- *Exploration*, e.g., using extended subgroup mining for larger search spaces and discovery dimensions – as a supervised technique, or applying clustering methods in the unsupervised case.

In the following sections, we first introduce the process model for selecting, instantiating, reusing and capturing task-experiences. Before discussing the core steps of the process in detail, we first describe the applied case and experience structure. After that, we discuss the maintenance issues for generalizing task-experiences to task-patterns.

2.1 Overview: Process-Model

For the process of capturing and reusing complex task-experiences for task and method selection, we distinguish four main steps: 1. *Experience Retrieval*, 2. *Task Instantiation*, 3. *Task Evaluation and Deployment*, and 4. *Experience Maintenance*.

In the following we consider both task selection and task instantiation for a specific task-configuration. The basic steps discussed above are embedded into the process model shown in Figure 1. The steps of that process are then arranged in a CBR-like cycle (e.g., [6]), and are outlined below.

1. Experience Retrieval: For retrieving appropriate experiences, the problem first needs to be defined by an abstract task description including information about the application domain. This query is applied for retrieving a set of most similar existing cases (experiences) for the defined problem. The retrieval step can be performed (semi-)automatically but also in interactive fashion as discussed below. Depending on the applied retrieval technique, that is browsing, interactive retrieval, or automatic retrieval, it is easy to see that the query needs to be specified in more detail for the automatic methods, whereas the other techniques allow for an incremental query refinement.

2. Task Instantiation: After the experience retrieval step the identified candidate task-configurations can be applied for selecting and configuring a specific data mining task and method. Essentially, the retrieved cases (task-configurations) need to be selected and adapted, for example, by merging (parts of) their characterizations in order to propose a candidate task-configuration fitting the abstract task characterization. The generated task-configuration can then be applied by the user. Depending on the query, the characterizations of the retrieved experiences either need to be transferred, modified or extended.
3. Task Evaluation and Deployment: When the data mining task has been completely instantiated it is ready for being applied by the user. After the data mining process has been performed, the task and its results need to be evaluated. Depending on these, for example, on the success of the data mining task, and on the quality of its results, the user can store the applied and instantiated case for future use in the experience base, replace an existing case, or merge/extend similar cases using the obtained results and the applied task-configuration.
4. Experience Maintenance: This step is concerned with the generalization, extension and adaptation of the experience base that collects the extended task-configurations and task-patterns. For maintenance, basic task-configurations can be refined to task-patterns by generalization and extension in an incremental process. Additionally, the set of experiences contained in the experience base can also be modified and refined. The maintenance step is especially applicable, for example, if there are too many similar cases for a certain task, or if an experts just extracts a set of prominent cases for an important task.

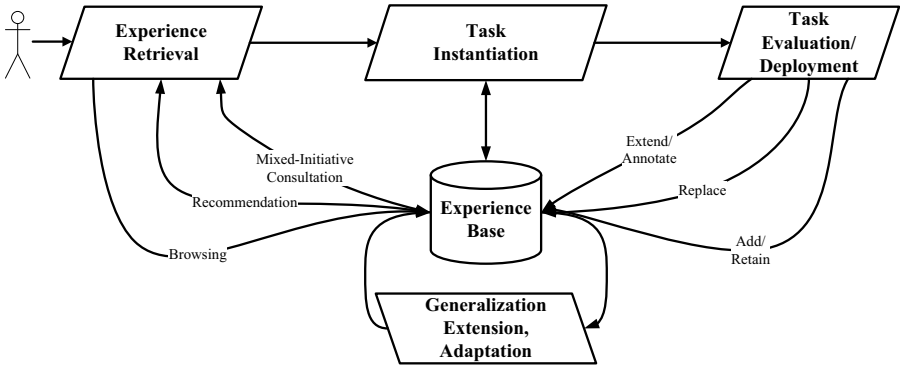

Fig. 1. Process Model: Experience Management with Task-Configurations and Task-Patterns

2.2 Case and Experience Structure

In a classical case-based reasoning (CBR) scenario, a case consists of a problem description and a solution. However, as discussed in [3], this view is too restrictive: Hence, in a more general view a case consists of a characterization containing a set of (structured) information entities, and optional artifacts.

In the context of this work, the characterization of a case contains mostly structured information (attribute – value pairs), while the optional meta-information contains unstructured textual information. The problem description and the solution part that is considered in classical CBR is then 'determined' by the type of the query, and by the set of the contained information entities. In this way, the characterization provided by the query is used to indicate the problem and solution elements of the retrieved cases. Since the combination (reuse) of the retrieved cases is a semi-automatic process that ultimately needs to be evaluated by the user, the appropriate entities and potential artifacts can be flexibly applied.

Extended Task-Configurations In the following, we sketch the characterization components of an extended task-configuration. We distinguish the abstract task (problem) description – relating to the applied data set and the interesting concepts (domain objects), and the specific task description – for example, considering the applied task, the specific method and its configuration. Furthermore, additional meta-information (artifacts) can be included. For the application context of descriptive data mining we can then define the components of an extended task-configuration as follows:

- Abstract task description:
  - Abstract task: Selection from a list of predefined categories (e.g., association, clustering, ...).
  - Applied data set: The name of the used data set
  - Optional: Data characteristics of the applied data set: These include, e.g., simple characteristics such as the number of instances contained in the data set, the number of attributes, and advanced characteristics such the distribution of the attributes, or the class distribution (cf. [1]).
  - Analysis objects: Selection of objects contained in the domain ontology. These can either contain attributes and/or attribute-value expressions.
- Specific task description:
  - Applied method: Selection from a list of predefined methods (e.g., Apriori, SD-Map, ...).
  - Configuration: Parameter configuration of the selected method, applied background knowledge, ...
- The meta-information includes the following artifacts:
  - Results: Results of the discovery run.
  - Rating: Explanatory text, concrete experiences.
  - Additional contextual information:
    - Date (When was the task performed?)
    - Responsible Person (By whom was the task performed?)
  - ...

As an example, consider the task of detecting associations using e.g., the Apriori-algorithm [12] for discovering association rules.

- Abstract task description:
  - Abstract Task: Association
  - Applied Data Set: Credit-g (UCI Repository)
  - Analysis Domain Objects: checkingstatus, savings_status, credit_history, employment, housing, personal_status, purpose, class
- Specific task description:
  - Applied Method: Apriori
  - Configuration: Minimal Support = 5%, Minimal Confidence = 80%
  - Results: ...
  - Contextual information:
    - Date: 2007-06-06
    - Responsible Persion: John Smith

Additionally, an extended task-configuration can optionally contain other information entities such as keywords or tags categorizing the characterization. Keywords can be selected from a pre-defined set or from a taxonomy, for example, or they can be provided as a simple free text artifact. User-defined tags can be utilized, for example, in order to specify type information for the task characterization, such as a rating (best practice, how to) of the task-configuration. This can also provide important guidelines for generalizing task-configurations to task-patterns.

Task-Patterns Task-patterns are a generalization of task-configurations. They can be defined by an experienced user, or by a domain specialist, either based on existing cases, or based on domain knowledge. Task-patterns typically capture the information for important tasks that are applied frequently by the users. Similar to design patterns [14] in software-engineering, task-patterns capture recipes and typical solutions of common problems. In summary, task-patterns contain the following additional elements:

- Name (Name and Classification): The name summarizes the tackled problem and its solution.
- Motivation (Example): This section describes a short example that shows the application of the pattern. The elements of the encapsulating extended task-configuration itself then serve as a concrete example.
- Applicability (Context): This section describes when the pattern can be applied.
- Consequences: This annotation describes the results and possible restriction when the pattern is being applied.

The name and the motivation section is used for a quick overview when browsing the patterns. Especially helpful for the user are the applicability and the consequences of the pattern, i.e., including specific methods, run-time restrictions, and potential statements regarding the validity and quality of the results. Thus, task-configurations can be refined to task-patterns by including additional annotations that are similar to these used for design patterns in software-engineering.

2.3 Experience Retrieval

Experience retrieval is concerned with the identification of appropriate (similar) experiences for a query task-configuration. For retrieving such experiences we distinguish three strategies:

1. Interactive Browsing: The user can browse the experience base, and can inspect an interesting set of cases using suitable selection criteria.
2. Semi-Automatic retrieval: The user applies a mixed-initiative CBR-approach for selecting task-configurations based on a partial problem characterization.
3. Automatic retrieval: Using a complete task-description, i.e., a completely specified query case, we can automatically retrieve a set of most similar cases using standard case-based reasoning techniques.

These options have different advantages and disadvantages: For the first strategy, the user can simply browse all cases using suitable filtering conditions. Starting with an overview of all cases and zooming in on an interesting subset of cases, this simple strategy enables a comprehensive view on the set of experiences. Browsing is the most simple and easy to apply option while it already requires a certain understanding of the domain and thus some domain knowledge by the user. Thus, the first option is both appropriate for inexperienced and experienced users that want to obtain an overview of the case base, or that approximately know which cases they want to select.

The second option enables a guided approach for selecting task-configurations. The proposed mixed-initiative approach (e.g., [15, 16]) interacts with the user and only requires a partial formalization of the query case. The user can inspect partial matches on the fly. The mixed-initiative approach starts with a partial task-description (case) given by the user. Then, a set of similar and related cases can already be retrieved. The approach further refines and optimizes the available set of cases by asking further relevant questions, aiming to ask the questions first that help to distinguish the set of cases most. This option is most appropriate for semi-experienced to experienced users that already have some knowledge about the typical applications.

Finally, the third approach is completely automatic: The user can determine a set of most similar cases automatically by providing a (nearly) complete problem/task description for the problem at hand. Using standard techniques from case-based reasoning (e.g., [6, 7]) we can retrieve a set of most similar cases. We apply a weighted similarity measure that considers the structure of a task-configuration, i.e., its characterization. We can utilize a partial similarity measure for each component of the characterization of the case, for example, for the abstract task, the data set name, and the analysis domain. For each of these, we apply a matching feature similarity measure, e.g., [17] and combine the partial measure into the total similarity measure. A suitable similarity measures makes it possible to identify not only completely matching cases, but also near-matches, that then need to be adapted. The last retrieval option is probably most suitable for inexperienced users that need some guidance during the selection process.

In addition to inspecting the most similar cases, the user can also opt to view a set of most similar but diverse cases, i.e., cases that are most similar to the query but not very similar to each other by utilizing diversity-conscious retrieval techniques, e.g., [17]. In this way, the user can obtain a broad view on the complete problem setting of the proposed tasks and methods.

2.4 Task Instantiation

Task instantiation aims to apply the retrieved task-configurations in order to reuse the contained experiences (e.g., [22]) for selecting a specific task and method, and for instantiating the provided query task-template. Depending on the query, different elements of the characterization of the task-configuration need to be considered. For structural elements voting mechanisms can be applied for recommending an adapted characterization considering the specific task and method. Furthermore, considering the parameters of the applied method, we can propose and/or apply each of these (for the same selected method). Concerning the analysis objects (for the same data set), potential options include selecting the most frequent objects (using a suitable threshold), or taking the union or the intersection of these. Essentially the recommendation which of the retrieved experiences are most appropriate for the query is determined by the similarity measure. Then, the experiences with a higher similarity to the query can also obtain a higher weight in the adaptation process.

However, the user ultimately needs to decide on these options in order to apply the proposed adaptation changes to the query task-template. The user can thus browse the contextual artifacts of a case, for example, the results, textual comments, and the rating of the case. This meta-information provides important guidelines for selecting the appropriate subset of the recommended experiences.

2.5 Task Evaluation and Deployment

After the data mining task has been performed, and the final results have been obtained, an important step is the evaluation of the results. Furthermore, the entire data mining project needs to be evaluated with respect to its performance, i.e., the performed task and method is reviewed, and the user needs to assess what potentially went wrong, and why. Then, also the results can be considered with respect to these questions in order to obtain a final documentation of the applied task-configuration.

The results of the evaluation are appended to the task-configuration, as textual annotations. Furthermore, the discovery results, i.e., the output of the discovery method are added, and the parameter configuration of the selected method is appended as a structured artifact. Further annotations of a task-configuration include other experiences provided by the user with respect to the project. These includes, for example, comments, and best practices concerning the performed task. The extended task-configuration itself can also be categorized using keywords or by assigning type information, such as a *best practice* or *how to* annotation.

Further important user information concerns the applicability and validity of a given case in a specific context: If the case can be successfully applied in many situations, then it should obtain a higher quality rating. Therefore, user feedback concerning the success and applicability of an experience should be appended to a case in order to estimate its quality (cf. [23]). Depending on the specific application domain, the user can also add other quality entities, such as an evaluation of the results (e.g., if they denote known or novel knowledge), and/or the case can be annotated with context-specific information, for example, a medical evaluation of the results.

2.6 Experience Maintenance: From Instances to Patterns

Maintenance is essential, when the size of the experience base grows, and if a large number of cases for the same task/method are collected. Then, the user needs to merge and/or refine cases in order to guarantee a experience base that includes relevant cases. Furthermore, if data mining methods are superseded by new algorithms or implementations, or if they are removed completely, then, the corresponding collected experiences need to be removed or refined. Additionally, experiences can be annotated regarding their validity and applicability for a specific task. This can be performed in a user-driven case base maintenance strategy [23]. Potential candidates for maintenance can then be indicated by the user in an experience feedback loop.

As shown in the process model in Section 2.1, the domain specialist can also provide higher-level templates, i.e., task-patterns by generalizing a set of cases: These patterns typically capture the information for important tasks that are applied frequently by the users.

As described in Section 2.2 task templates capture recipes and typical solutions for common problems, similar to design patterns [14] in software-engineering. The additional meta-information of task-patterns are thus similar to design patterns/coding idioms in software-engineering. They include the *name* and *classification* of a typical task/method, a motivational *example* describing the application of the template, and its *context*, i.e., describing the situation when the pattern can be applied. Furthermore, a task-pattern also includes a description of the *consequences* of its application, i.e., the (typical) results and possible restriction when the pattern is being applied.

3 Discussion

In this paper, we have presented a methodological approach for capturing and reusing experiences given by task-configurations. The presented techniques aim to enable an easier selection and instantiation of appropriate methods and tasks in descriptive data mining.

In comparison to other approaches, e.g., the *AST*-approach by Lindner and Studer [1] we focus on a descriptive setting, that is, knowledge discovery for explorative data analysis and data mining. Therefore, we do not necessarily focus on selecting the appropriate or the "best" algorithm with respect to the prediction performance. Also, the data characteristics do not play such a prominent role in the presented approach, while these play a very important role in classifiation/prediction settings for selecting an appropriate method. For our setting (description) these characteristics are not that important, because we do not need to construct a (predictive) model. Instead, we focus on descriptive data mining tasks such as description, exploration and (statistical) summarization of the data. However, the presented approach could also be extended for predictive and classification tasks by adapting the extended task-configurations.

In comparison to the approach described by Bartlmae [20, 21] that provides a experience management process for the whole CRISP-DM [19] model, we focus especially on the task selection and method instantiation problem. Therefore, the presented

approach is much more specialized concerning the selection, and especially concerning the instantiation step. In the presented approach, the adaptation is performed semi-automatically, that is, recommendations are proposed to the user who then needs to obtain the final instantiated task-configuration.

Compared to the Mining Mart approach discussed by Euler [18] that focuses on preprocessing chains and stores these as cases, our approach is focused on the general application of a specific data mining task and method. Therefore, as discussed above, it focuses on descriptive methods, but could also be generalized for a specified set of preprocessing methods. Furthermore, we provide powerful techniques for proposing suitable task-configurations in order to determine appropriate tasks and methods.

The proposed process model can be integrated into certain stages/parts of the knowledge discovery process. One prominent process model is given by the CRISP-DM [19] model – an iterative and incremental process consisting of several phases (according to the CRISP-DM standard) summarized below:

- Business understanding: Determine the business objective, the data mining goals, and produce a project plan.
- Data understanding: Collect and describe the data, evaluate its quality and perform initial exploration.
- Data preparation (preprocessing): Perform the common steps of data preprocessing, i.e., data selection, cleaning, integration.
- Data mining (modelling, i.e. build a model of the data): Select the appropriate data mining method, build a data mining model, obtain the parameter settings that can iteratively be revised and refined.
- Model evaluation (post-processing, refining the model): Evaluate the model and review the process.
- Deployment of the model: Plan and perform the deployment of the generated results of the data mining task; produce a final report and review the project collecting the obtained experiences as a documentation of the project.

The relevant steps with respect to the presented approach are business understanding, modeling, evaluation, and deployment. During the business understand and modeling phases, we can retrieve relevant experiences from the case base for determining, selecting and tuning the appropriate method for the respective data mining task. During the modeling and evaluation phase the final parameter configuration and the results of the applied data mining method are obtained. Finally, during evaluation and deployment the user can annotate the task-configuration with additional meta-information and experiences that were obtained during the process.

Appropriate tools are an important prerequisite for the application of the presented process. The proposed approach can then either be provided as a stand-alone tool, or it can be integrated into the applied data mining workbench. While the first option provides the application of a set of experiences for potentially many data mining tools, the second option provides for a more intuitive application by the user: Since the recommendation tool is directly included in the mining application, the experience-based process can be seamlessly embedded in the knowledge discovery workflow. For this reason, we have opted to include a experience-management plugin into the VIKAMINE [11] system.

4 Conclusion

In this paper, we have presented a methodological approach for experience management with task-configurations and task-patterns for the selection and instantiation of data mining tasks and methods. We have shown how to structure task experiences, how they can be retrieved, reused, captured, maintained, and how they can be applied in a given data mining context. These steps are integrated into a process model for experience management with task-configurations and task-patterns.

The approach was motivated by experiences in several knowledge discovery projects, e.g., [8–10] using descriptive data mining methods. We plan to apply the presented techniques in future projects in order to collect a rich experience and pattern library for descriptive data mining.

In the future we also plan to consider extended strategies for the retrieval of cases. Furthermore, alternative methods for recommending appropriate tasks and methods in data mining is a further interesting option for future work: A hybrid approach, for example, using explicitly formalized knowledge in conjunction with an experience-driven wizard seems an interesting alternative. Such an approach can then potentially be used for a step-by-step tutoring of inexperienced users.

Additionally, we aim to utilize the stored experiences in a broader context: The result artifacts of successful experiences can potentially be reused as known patterns, or background knowledge for knowledge-intensive data mining applications (e.g., [8]). Furthermore, such result patterns can also be utilized for continuous evaluation of interesting relations, e.g., for studies in the medical domain.

Acknowledgements

This work has been partially supported by the German Research Council (DFG) under grant Pu 129/8-1.

References

1. Lindner, G., Studer, R.: AST: Support for Algorithm Selection with a CBR Approach. In: Proc. 3rd Europ. Conf. on Principles of Data Mining and Knowledge Discovery (PKDD '99), Springer (1999) 418–423
2. Althoff, K.D., Hanft, A., Schaaf, M.: Case Factory - Maintaining Experience to Learn. In: Advances in Case-Based Reasoning, Proc. 8th European Conference on Case-Based Reasoning (ECCBR 2006), Berlin, Springer (2006) 429–442
3. Althoff, K.D., Decker, B., Hartkopf, S., Jedlitschka, A., Nick, M., Rech, J.: Experience Management: The Fraunhofer IESE Experience Factory. In: Proc. Industrial Conference on Data Mining, Leipzip, Germany, IBAI (2001) 12–29
4. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Springer, Berlin (2002)
5. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann Publisher (1993)
6. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications **7(1)** (1994) 39 – 59

7. Bartsch-Spörl, B., Lenz, M., Hübner, A.: Case-Based Reasoning: Survey and Future Directions. In: XPS-99: Knowledge-Based Systems - Survey and Future Directions, Proc. 5th Biannual German Conference on Knowledge-Based Systems. (1999) 67–89
8. Atzmueller, M., Puppe, F., Buscher, H.P.: Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery. In: Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland (2005) 647–652
9. Atzmueller, M., Puppe, F., Buscher, H.P.: Profiling Examiners using Intelligent Subgroup Mining. In: Proc. 10th Intl. Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005), Aberdeen, Scotland (2005) 46–51
10. Atzmueller, M., Baumeister, J., Hemsing, A., Richter, E.J., Puppe, F.: Subgroup Mining for Interactive Knowledge Refinement. In: Proc. 10th Conference on Artificial Intelligence in Medicine (AIME 05). LNAI 3581, Berlin, Springer (2005) 453–462
11. Atzmueller, M., Puppe, F.: Semi-Automatic Visual Subgroup Mining using VIKAMINE. Journal of Universal Computer Science **11**(11) (2005) 1752–1765
12. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, (VLDB), Morgan Kaufmann (1994) 487–499
13. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edition edn. Morgan Kaufmann (2005)
14. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.M.: Design Patterns: Abstraction and Reuse of Object-Oriented Design. In: Proc. 7th European Conference on Object-Oriented Programming. (1993) 406–431
15. Aha, D.W., Breslow, L., Munoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence **14**(1) (2001) 9–32
16. Aktas, M.S., Pierce, M., Fox, G.C., Leake, D.: A Web based Conversational Case-Based Recommender System for Ontology aided Metadata Discovery. In: Proc. 5th IEEE/ACM International Workshop on Grid Computing (GRID '04), Washington, DC, USA, IEEE Computer Society (2004) 69–75
17. McSherry, D.: Diversity-Conscious Retrieval. In: Proc. 6th European Conference on Advances in Case-Based Reasoning, Berlin, Springer (2002) 219–233
18. Euler, T.: Publishing Operational Models of Data Mining Case Studies. In: Workshop on Data Mining Case Studies, 5th IEEE Intl. Conf. on Data Mining (ICDM), Houston, Texas, USA (2005) 99–106
19. Reinartz, T.: Stages of the Discovery Process. In: Handbook of Data Mining and Knowledge Discovery. Oxford University Press, Inc., New York, NY, USA (2002) 185–192
20. Bartlmae, K.: Optimizing Data-Mining Processes: A CBR Based Experience Factory for Data Mining. In: Internet Applications. Proc. 5th International Computer Science Conference. Volume 1749 of LNCS., Berlin, Springer 21–31
21. Bartlmae, K., Riemenschneider, M.: Case Based Reasoning for Knowledge Management in KDD Projects. In: Proc. 3rd Intl. Conf. on Practical Aspects of Knowledge Management, Basel, Switzerland (2000)
22. Althoff, K.D., Birk, A., Hartkopf, S., Müller, W., Nick, M., Surmann, D., Tautz, C.: Systematic Population, Utilization, and Maintenance of a Repository for Comprehensive Reuse. In: Learning Software Organizations, Methodology and Applications. Proc. 11th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE '99), Berlin, Springer (2000) 25–50
23. Nick, M.: Reducing the Case Acquisition and Maintenance Bottleneck with User-Feedback-Driven Case Base Maintenance. In: Proc. 19th Intl. Florida Artificial Intelligence Research Society Conference, AAAI Press (2006) 376–382