University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1996-3

Interactive Exploration of Discovered Knowledge: A Methodology for Interaction, and Usability Studies

Mika Klemettinen, Heikki Mannila, and Hannu Toivonen

Helsinki, February 1996

Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

The papers in the series are intended for internal use and are distributed by the author. Copies may be ordered from the library of Department of Computer Science.

Interactive Exploration of Discovered Knowledge: A Methodology for Interaction, and Usability Studies

Mika Klemettinen, Heikki Mannila, and Hannu Toivonen

University of Helsinki, Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
E-mail: {mklemett, mannila, htoivone}@cs.Helsinki.FI
Report C-1996-3, February 1996, 27 pages

Abstract

We introduce a methodology for knowledge discovery in databases (KDD) where one first discovers large collections of patterns at once, and then performs interactive retrievals from the collection of patterns. The proposed methodology suits very well such KDD formalisms as association and episode rules, where large collections of potentially interesting rules can be found efficiently. We present methods that support interactive exploration of large collections of association and episode rules. With these methods the user can flexibly specify the focus of interest, and also iteratively refine it. We claim that the proposed methodology is very usable for a class of knowledge discovery problems. We describe an implementation of the explorative part. We discuss some usability aspects and usability testing of KDD systems, and present an actual usability test of the described implementation.

Computing Reviews Categories and Subject Descriptors:
H.3.3 [Information Systems]: Information Storage and Retrieval – Information Search and Retrieval
H.5.2 [Information Systems]: Information Interfaces and Presentation – User Interfaces

General Terms:
KDD methodology, KDD process, System usability, Usability tests.

Additional Key Words and Phrases:
Data mining, Knowledge discovery, Interactive data mining, User interfaces, Association rules, Episodes.

1 Introduction

The aim of *knowledge discovery* (KDD) as a research field is to examine methods for understanding large amounts of data by discovering interesting patterns in the data. Most of the research has concentrated on the discovery of patterns; however, the inference of patterns is only a small part of the whole problem of obtaining useful knowledge from data. A KDD process consists typically of the following steps (Figure 1):

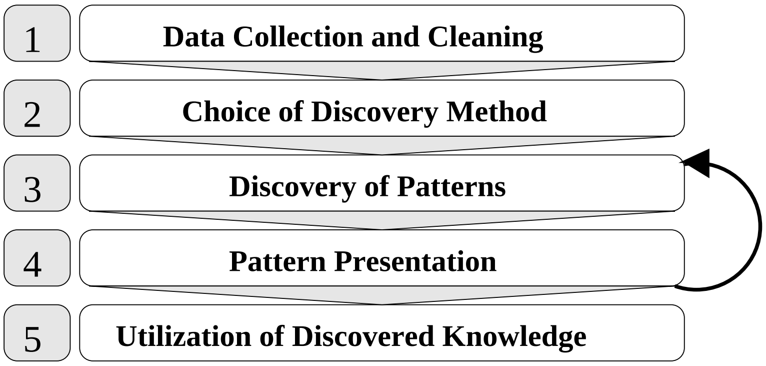

Figure 1: Knowledge discovery process steps

1. **Data collection and cleaning**: selecting attributes, dealing with errors and missing data, identification of the necessary background knowledge, etc.

2. **Choice of pattern discovery method**: deciding on the types of knowledge to be discovered, parameter selection, etc.

3. **Discovery of patterns** (data mining): running algorithms for discovering different types of patterns.

4. **Pattern presentation**: selection of truly interesting patterns, visualization of results, etc.

5. **Putting the knowledge into use**.

KDD is an iterative process and requires interaction with the user. The background knowledge and interests of users are different. And even for a domain expert, it is often impossible to specify what is interesting before actually seeing what the results are like — after receiving some knowledge the user can better focus the search to more interesting areas. However, many KDD system descriptions do not mention iteration at all. In most systems iteration is achieved by repeatedly discovering and presenting patterns.

In Section 2 of this article we introduce a KDD methodology which gives special roles to the pattern discovery and pattern presentation phases of the KDD process. First, large collections of patterns are discovered at once. The emphasis in our methodology is in the pattern presentation part, where interesting information is iteratively searched for, without repeating the time consuming pattern discovery phase.

As examples of the feasibility of the methodology, we use two case studies. The first one concerns association rules [1] in course enrollment data, and the other one handles rules derived from episodes [16] in telecommunication network alarm sequences. In Section 3 we discuss the pattern presentation phase in more detail and present methods for finding interesting association and episode rules from large collections of rules.

Our claim is that the proposed methodology is very usable for a class of knowledge discovery problems. To discuss the important issue of usability in KDD on a concrete level, we describe in Section 4 an implementation of the proposed pattern presentation phase. We give a quick overview of some aspects of usability of KDD systems, and present an actual usability test of the described implementation. Section 5 is a concluding discussion.

Related work KDD process and methodology have been discussed in [4, 5]. The process described in [4] also underlines the importance of interaction and iteration. The most essential iteration in the process is in repeating the pattern discovery and presentation phases. The efficiency is improved by reusing information about earlier iterations. With the exception of these rather general studies, the KDD research has pretty much concentrated on the pattern discovery phase of the process.

Something can be said, however, of the processes supported by various KDD systems, none of which supports the methodology proposed here. Explora [11, 13] finds interesting instances of statistical patterns. In Explora, the pattern discovery phase is focused by the user. The system selects and presents the best patterns to the user, and, based on the results, the user can change the focus and repeat the pattern discovery. The patterns discovered by 49er [27] are contingency tables, equations, and logical equivalencies.

The user can interactively change the focus, e.g., independent and dependent variables, and require for a new pattern discovery. The Key Finding Reporter (Kefir) [17, 21] discovers and explains deviations, and gives recommendations for corrective actions. Applications of Kefir are tailored with a lot of domain knowledge to be aware of the interestingness criteria, corrective actions, etc., of the domain. Given a database from the domain, a Kefir-based application produces a report of the deviations without iteration. Note that the patterns discovered by all these systems, i.e., Explora, 49er, and Kefir, are more expressive than association and episode rules, and focusing the pattern discovery is thus more important.

Research on methods for finding (potentially very large) collections of association rules has been very intense (e.g., [2, 7, 10, 23]); however, studies of presentation issues have remained few [12, 26].

The discussion about usability in KDD has so far mainly concerned the interestingness of the discovered knowledge and efficiency issues, but not the usability of KDD tools.

2 A Knowledge Discovery Methodology

We propose a KDD process with the following characteristics (Figure 2).

1. In the pattern discovery phase, find *all* potentially interesting patterns according to some rather loose criteria.

2. Provide flexible methods in the presentation phase for iteratively and interactively creating different views to the discovered patterns.

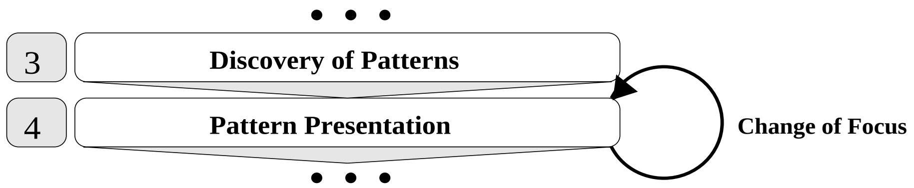

Figure 2: Main process phases of our proposed methodology

At the first glance this may seem wasteful: why find a very large number of patterns instead of focusing the search to the interesting ones?

The methodology is based on two simple facts:

- **Iteration is essential in KDD**. In the field of data mining or exploratory data analysis, it can be hard, or even impossible to specify beforehand, what is interesting. The goal is to discover previously unknown information — but if it is unknown, how do you know it is interesting? Our answer is: by iterative exploration.

- **Pattern generation from very large data sets is time consuming.** Efficient algorithms exist, but the pattern extraction phase still is a barrier for smooth interaction.

By producing all patterns at once different views of the data can be created very efficiently in the pattern presentation stage.

There are several KDD formalisms that fit in this setting. Association rule [1] and episode [16] algorithms can efficiently discover thousands of rules from relatively simple databases (see, e.g., [2, 23] for efficient algorithms and [12] for the large number of rules). A formal treatment of the setting of discovering all interesting sentences and an analysis of a general algorithm can be found in [14].

Example 1 The association rule discovered from a course enrollment database

Introduction to Unix $\Rightarrow$ *Programming in C (0.84, 0.34)*

states that 84 % of the students that have taken "Introduction to Unix" also have taken "Programming in C", and that 34 % of all the students actually have taken both courses.

Such rules can be found in a student enrollment database of courses in computer science at the University of Helsinki. The database consists of registration information of 1066 students who had registered for at least two courses. In the data there is a row per student, containing the courses the student has registered for. On average, a row contains 5 courses. The total number of courses is 112.

The goal in analyzing student enrollment data is to obtain accurate and useful information about the interrelationships between enrollments to various courses. Such relationships can be valuable, e.g., in the planning of the curriculum and in allocating resources.[1] □

[1] University enrollment data has been analyzed also in [22], from a slightly different perspective.

The idea is that any pattern that occurs frequently enough can potentially be interesting. The algorithms find all rules that fulfill the given threshold criteria, and the decision of what is interesting is for the most part left to the end-user to explore.

Example 2 There are 2010 association rules in the course enrollment database that match at least 11 students (i.e., the frequency (or support) threshold is 0.01). □

This approach of discovering all patterns can be contrasted with numerous methods, e.g., in machine learning, which are more focused and produce one or at most a few patterns that match the given problem specification. These methods usually require that the searched or learned subject is quite carefully described in advance, and they leave any other potentially interesting phenomena unfound. The advantage of these systems is that the patterns they find are more expressive than the relatively simple association and episode rules.

Our examples use association and episode rules, but the basic idea — iteration in the pattern presentation phase – can be applied to any formalisms that have some similar properties as association and episode rules:

- There is an algorithm that produces lots of potentially interesting patterns.
- The time requirement for discovering all potentially interesting patterns is not considerably longer than if the discovery was focused to a small subset of the potentially interesting patterns.
- The desired focus is not known definitely in advance.

To complete the picture of our KDD process model (Figures 1 and 2), we now give a short overview of it as a whole.

In the first phase of the process, the raw data is collected and prepared into a suitable form for the following phases. A general overview of the data can be produced at this phase. Attributes identified as irrelevant are removed, and potentially useful new attributes can be derived. All relevant background knowledge should be taken advantage of, here, as well as in the other phases. For instance, in the case of telecommunication alarm data attributes can be derived to contain, e.g., information about the network topology and time fields such as if the current day is a workday or not. This initial phase of the process can be surprisingly tedious. The goal is that it need not be repeated later.

Based on the knowledge about the data, partially obtained during the data collection and cleaning phase, the discovery methods are selected. We skip the discussion, and move directly to the discovery of patterns as done in our proposed model. In the data mining phase all potentially interesting patterns are generated from the data set. The (rather loose) criteria for interestingness can be, e.g., frequency, prevention of taking rarely occurring phenomena into account, or strength of the patterns. Descriptions of the association rule and episode algorithms we use can be found, e.g., in [2, 15] and in [16], respectively.

Example 3 A telecommunication alarm database contains event data from a number of interconnected components, such as switches, in telecommunication networks. Alarms are produced by the components to report abnormal situations. There are typically thousands of alarms a day, and they are of thousands of different types. Our example data set contains 43 478 alarms from a period of 10 days.

Examples of the rule types discovered in the alarm data are (the first two are episode rules and the last one is association rule):

- If an alarm of type A occurs, then an alarm of type B occurs within 30 seconds with probability 0.8.
- If alarms of types A and B occur within 5 seconds, then an alarm of type C occurs within 60 seconds with probability 0.7.
- If an alarm type is A and the alarming network element is X, then with probability 0.9 the alarm severity class is 1. □

Example 4 There are 2395 parallel and 4531 serial episode rules that hold in the alarm sequence (with a frequency threshold of 0.005 and a window width of 5 minutes). When considering associations between attribute values of individual alarms, there are 1497 association rules that match at least 435 alarms (i.e., frequency threshold is 0.1).

In the pattern discovery phase, the parameters have to be adjusted properly. For example, in the case of telecommunication network alarm data there usually is a roughly known time window size within which interesting relationships exist. If the window size is not set properly, relationships can remain unfound if they do not fit into the window, or relationships can be buried under noise, if the window is too large. □

The goal is, again, that returning to the pattern discovery phase is largely unnecessary. It is realistic to expect, however, that the algorithms need to be run several times. Parameter settings may need adjustment, or errors in the data cleaning phase may be revealed so that returning to the first phase of the process is necessary.

The presentation of discovered knowledge is a main part of our methodology. In this phase the interesting patterns should be located from large collections of potentially interesting patterns. But what is interesting? How to define the interestingness? The issue of interestingness of discovered knowledge has been discussed, e.g., in [13, 20]. For the most part, what is interesting depends on the case, and is highly based on the user's personal aims and perspective. As argued already, iteration is needed. With iteration, it is possible to refine the focus after seeing what the results are like.

Example 5 Discovered rules can fail to be interesting for several reasons.

- **A rule can correspond to prior knowledge or expectations.** The course enrollment database, for instance, satisfies a number of rules that correspond to normal process in the studies. Basic courses are taken before graduate courses, so the following kind of rules can be expected:

  Design and Analysis of Algorithms $\Rightarrow$ *Introduction to Computers (0.97, 0.03).*

  (Note that the rule implies no temporal order. It only states that the students who have taken "Design and Analysis of Algorithms", have at some time also taken "Introduction to Computers".)

  Or, the alarm data contains a number of strong associations between certain alarms. For example, it can be known from the implementation that if an element sends an alarm A, it will also send an explanatory alarm B.

- **A rule can refer to uninteresting attributes or attribute combinations.** E.g., the rule

  Introduction to Computers $\Rightarrow$ *Programming in Pascal (0.95, 0.60)*

  is useless, if the user is only interested in graduate courses.

In the case of the network alarm data, rules containing e.g. low severity alarms may be non-interesting, and the user would like to filter out all such rules.

- **Rules can be redundant.** For example, the latter of the following rules has no additional predictive power over the first one, and could be pruned as redundant:

  Data Communications, Programming in C $\Rightarrow$ *Unix Platform (0.14, 0.03)*

  and

  Data Communications, Programming in C, Introduction to Unix $\Rightarrow$ *Unix Platform (0.14, 0.03).*

  In the alarm data the rules can contain alarms of different abstraction levels but actually referring to the same fault. □

Iteration serves the explorative task of becoming familiar with the data. Iteration also has an important role in recalling passive background knowledge — every data miner knows how hard it can be to describe the general characteristics even of a familiar data set. Looking at and filtering the discovered patterns iteratively simplifies the task considerably.

Example 6 Assume the user is exploring a collection of association rules that hold in the course enrollment data. When browsing the rule set and searching for interesting information, the user may first want to filter out rules referring to uninteresting courses. As an example, he/she may not be interested in basic courses such as "Programming in Pascal", and filters out all rules containing basic courses. With this operation, the original rule set of size 2010 rules is reduced almost to half.

Now the user could focus to, e.g., all rules containing the course "Programming in C", in order to find out what other (not basic) courses are connected with it.

Then the user notices that some of the rules contain information that is trivial, e.g., strong connection with the course "Unix Platform". By filtering out all rules containing the course "Unix Platform", the user can focus into the more interesting rules. □

Example 7 With our other example, alarm data, the user may have the specific intention of finding reasons for problems between two adjacent network elements. With that purpose in mind, he/she can first select all rules containing alarms from those network elements and then filter out all rules with lower severity alarms, because those having high severity usually contain the answer. After that he/she can, e.g., iteratively search possible reasons either by further pruning the rule set or by using definite guesses based on previously found possible reasons as guidance for search criteria. □

Of course, the iteration itself is not enough. The found knowledge has to be understood and applied. The role of the KDD system is to provide appropriate tools for displaying and browsing the discovered knowledge, methods for expressing the desired focus, support for iterative specification of the focus, etc.

3 Presentation of Rules

As a result of the pattern discovery phase of our KDD process model, all patterns considered potentially interesting are produced. All the patterns are, however, not supposed to be interesting to a particular user or in a particular situation. On the contrary: the exploration of the discovered patterns must be assisted. The aim of the presentation phase of our methodology is to support efficient, interactive, and focused views to the rules. In this section, we present methods for exploring large sets of association and episode rules. The ideas of this section can be best applied on large, unstructured sets of rules and other similar, simple pattern formalisms.

Three types of operations are useful in processing a large collection of rules (Figure 3):

1. pruning: reduction of the number of rules;
2. ordering: sorting of rules according, e.g., to statistical significance; and
3. structuring: organization of the rules, e.g., to clusters or hierarchies.

Most KDD systems prune and order the set of patterns automatically, typically aiming at a small, non-redundant collection of the most interesting (e.g., strong, useful, surprising, or valuable) patterns. A pattern set can be structured automatically, e.g., arranged to hierarchies of exceptionality.

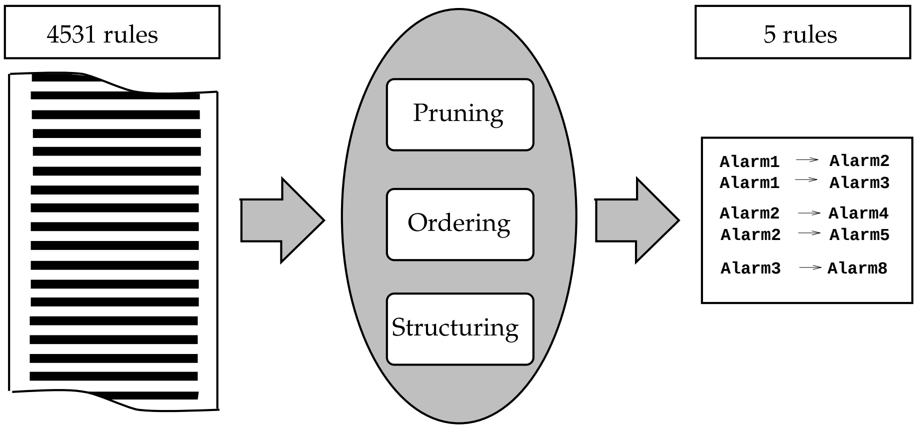

Figure 3: Rule set processing

We adopt a different approach. The user has total, explicit control over the operations on a rule set. This way the user can always be sure that the view on the data is in a sense complete: the system has not put aside any rules automatically. In addition, automatic pruning techniques are often sensitive to the data, and a slight change in the focus could change the rules that are shown.

Almost trivial ordering criteria for association and episode rules are rule confidence, frequency, and statistical significance. Rules can be pruned by setting thresholds on these properties.

A very useful formalism for pruning a rule set are *templates* [12], pattern expressions that describe the form of rules that are to be selected or rejected. In the context of association and episode rules, templates define a set of rules by specifying what attributes occur in the antecedent and what in the consequent. A hierarchy of attributes is very useful with templates.

Example 8 In our example domain of computer science course enrollment, the courses are divided into three classes: basic, undergraduate, and graduate courses. In addition, all courses belong to the parent class 'any course'. Thus, we have such generalizations as

{Introduction to Computers, Programming in Pascal, Information Systems} $\subset$ Basic Course $\subset$ Any Course,

{Artificial Intelligence, Programming in C, Data Communications} $\subset$ Undergraduate Course $\subset$ Any Course,

{Design and Analysis of Algorithms, User Interfaces, Neural Networks} $\subset$ Graduate Course ⊂ Any Course. □

A template is an expression

$$A_1, \ldots, A_k \Rightarrow A_{k+1}, \ldots, A_l,$$

where each A_i is either an attribute name, a class name, or an expression $C+$ or $C*$, where C is a class name[2]. Here $C+$ and $C*$ correspond to one or more and zero or more instances of the class C, respectively. A rule

$$B_1, \ldots, B_m \Rightarrow B_{m+1}, \ldots, B_n$$

matches a pattern if the rule can be considered to be an instance of the pattern. The order of the components in the templates has no significance if the attribute order in the rules is not significant (association rules and parallel episode rules).

Example 9 Assume the user is interested in the course "Design and Analysis of Algorithms". In particular, it would be interesting to know associations where "Design and Analysis of Algorithms" is on the right-hand side and there is another graduate course on the left-hand side, such as

Neural Networks ⇒ Design and Analysis of Algorithms (0.48, 0.02)

or

Artificial Intelligence, Stringology ⇒ Design and Analysis of Algorithms (0.65, 0.01).

Such rules match the pattern

Graduate Course, Any Course∗ ⇒ Design and Analysis of Algorithms.

□

Templates are one way of focusing the view to a large space of rules. With templates, the user can explicitly specify both what is interesting and what is not. To be interesting, a rule has to match a *selective template*. If a rule, however, matches an *unselective template*,[3] it is considered uninteresting. To be presented to the user, a rule must be considered interesting — i.e. match one of the selective templates — and it must not be uninteresting — i.e. not match with any of the unselective templates.

[2] A_i can also be a regular expression.

[3] "Selective template" and "unselective template" were in [12] referred to as "inclusive template" and "restrictive template", respectively.

Example 10 Using the selective template

> *Graduate Course, Any Course∗ ⇒ Design and Analysis of Algorithms*

the user obtains a number of rules with (at least) a graduate course in the antecedent and the course "Design and Analysis of Algorithms" as the consequent. Examples of such rules are

> *Compilers, Introduction to Computers ⇒ Design and Analysis of Algorithms (0.16, 0.01),*
>
> *Neural Networks, Data Communications ⇒ Design and Analysis of Algorithms (0.33, 0.01), and*
>
> *User Interfaces, Information Systems ⇒ Design and Analysis of Algorithms (0.02, 0.10).*

In the example database there are 15 rules that match the above specifications and have confidence of 0.2 or more and frequency of 0.01 or more. This sort of rules give an idea what the students do in addition to taking "Design and Analysis of Algorithms", and the discovered rules can be utilized when planning the course. (Remember, the arrow implies no temporal order.) □

It is obvious how selective templates can be used to set the focus. Unselective templates are very useful for excluding something explicitly from the focus.

Example 11 The user may consider some of the rules of the previous example uninteresting, such as those including basic courses:

> *Compilers, Introduction to Computers ⇒ Design and Analysis of Algorithms (0.16, 0.01).*

By adding an unselective template

> *Basic Course, Any Course∗ ⇒ Any Course*

the user can filter out all rules with a basic course on the left-hand side. □

Example 12 On the other hand, what rules do we have with "Design and Analysis of Algorithms" on the left-hand side? Using the selective template

*Design and Analysis of Algorithms, Any Course** $\Rightarrow$ *Any Course*

and the unselective templates

*Basic Course, Any Course** $\Rightarrow$ *Any Course*

and

*Any Course** $\Rightarrow$ *Basic Course*

one obtains rules with “Design and Analysis of Algorithms” on the left side and without any basic courses.

Again, with a confidence threshold of 0.2 and a frequency threshold of 0.01, there are 47 such rules. One of them states that 60 % of students taking “Design and Analysis of Algorithms” also take “Artificial Intelligence”.

Another rule is

Design and Analysis of Algorithms $\Rightarrow$ *Neural Networks (0.22, 0.02).*

Thus, only 22 % of students taking “Design and Analysis of Algorithms” also take “Neural Networks”; on the other hand, above it was found that 48 % of students taking “Neural Networks” also take “Design and Analysis of Algorithms”. □

This technique is surprisingly powerful. For example, background knowledge about the normal curricula of computer science studies can be taken into account by using templates that unselect all rules that correspond to the normal course of studies.

Note that in our proposed methodology the use of background knowledge can be embedded in a KDD session. While looking at the patterns, an expert typically recognizes rules that he/she actually know. The focus can be instantly changed accordingly.

Templates can be used as kind of user profiles. For instance, each user could have his/her own set of templates, coding some of his/her background knowledge and interests; or, template collections could correspond to specific tasks with different focuses and assumptions.

It should be noted that sometimes considerable amounts of rules remain, even when the user has found the desired focus with the described methods. Automatic pruning, ordering, and structuring methods should at this point be available for invocation by the user, especially for removal of redundancy.

A method for pruning a set of strong association or episode rules by removing redundancy has been presented in [26]. The method is based on computing a *rule cover*. A rule cover is a subset of rules that has predictive power equal to the original set of rules, in the sense that whenever a rule of the original set mathces a row, then there is a rule in the cover that matches the row.

An approximate order of interestingness could be assigned to the discovered rules by giving weights to templates: positive weights to selective templates and negative weights to unselective templates. The ranking of rules would then correspond to the sum of weights of matched templates.

Clustering has been proposed as a method for structuring a set of association or episode rules [26]. The idea is to take rules with the same right-hand side, and to group the rules so that rules in the same cluster tend to match the same cases in the database. Clustering of rules has a remote connection to clustering of database rows.

4 Usability Aspects and Tests

The goal of knowledge discovery is to find useful patterns. An obvious requirement for success in this task is that the KDD tools and the knowledge representation formalisms are such that the user is able to find and understand the useful patterns. In short, the usability of KDD tools has an essential, often perhaps under-estimated role. In this section we study usability. Our approach is very concrete: we examine the central phase in the proposed model, pattern presentation, and present a usability test of an implementation.

We will give an overview of the implementation of the rule presentation phase in `TASA` (Telecommunication Network Alarm Sequence Analyzer) [9]. `TASA` is a system for discovering knowledge from telecommunication alarm databases, and it is being built in cooperation with several telecommunication companies. `TASA` incorporates components for the pattern extraction (or data mining) and presentation phases of the KDD process as described in this article. The pattern types discovered by `TASA` are episodes [16] and association rules [1]. After the overview, we describe the test procedure used to test `TASA`, while also quite shortly presenting some general issues about usability testing.

Generally used criteria for the usability of a system are the following [19]:

- The system should be easy to learn.

- The system should be efficient to use.
- The system should be memorable, i.e. easy to remember.
- The error rate should be low and no critical errors should occur.
- The system should be pleasant to use.

While KDD is largely guided by real world needs and applications, there still is the risk that a system is very efficient from the scientific point of view, but it fails to meet the users needs because of, e.g., an unintuitive pattern formalism or an unattractive user interface.

A very efficient way to improve the usability of a system is to test it with its intended users. To give the developers useful feedback, the tests have to be planned and executed carefully. Also, the analysis of the results requires special care, and an open mind as well.

4.1 Pattern Presentation Implementation in `TASA`

The presentation of discovered knowledge in `TASA`, the part visible to the user, is based on a hypertext user interface that is implemented in HTML (HyperText Markup Language). A hypertext solution was selected, because it is very suitable for the presentation of large amounts of data with inherent links but with no particular structure. Lists of rules, figures, text documents, and other diverse material can be easily mixed.

HTML is a simple, general text based description language for creating hypertext documents and extensively used in the World Wide Web (WWW). Documents can be viewed with standard WWW browsers such as Netscape or Mosaic. The exploration tools, in particular templates, are embedded into the HTML pages and are based on the fill-out form feature of the HTML language. HTML can easily be generated automatically. Using HTML also means that porting the rule presentation system to different platforms is very simple, and that the system can be used both locally and over a network.

The basic idea of the system is as follows. Each rule set generated by the algorithms is converted to an HTML document (Figure 4, right). Templates, thresholds, and sorting criteria are specified by means of an HTML form (Figure 4, left). The operations are performed on the rule set, and a new document is created from the old one "on-the-fly" with the given criteria. A document log is maintained, so that the user can go back and reformulate the selections.

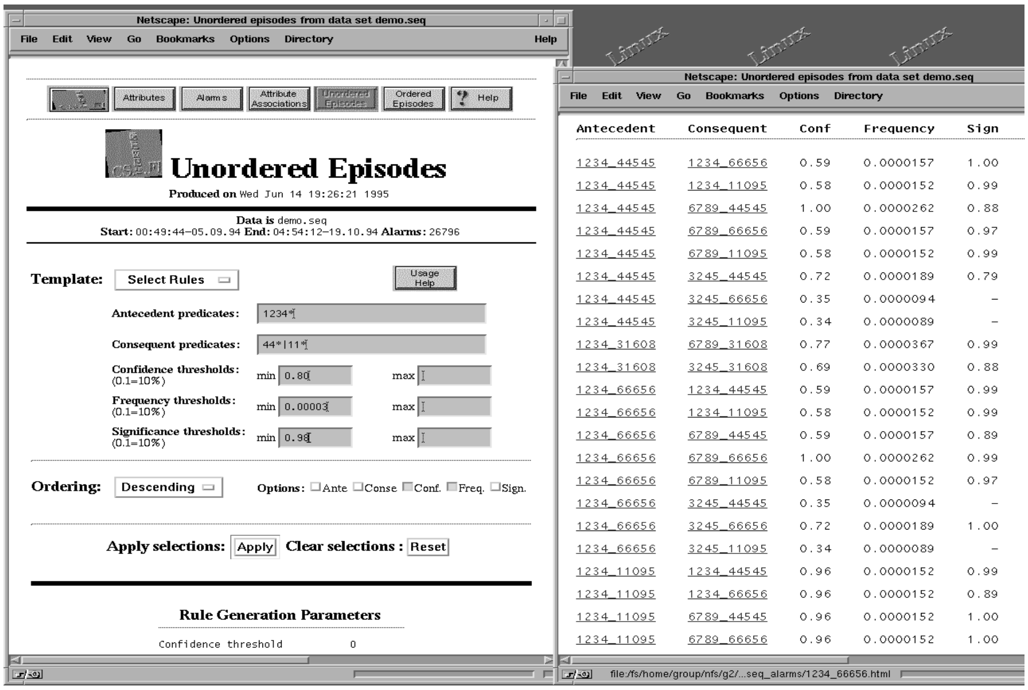

Figure 4: Exploration tool and some discovered rules in TASA

Example 13 Templates and other rule selection criteria are given on an HTML fill-out form (Figure 5). The user defines a template by selecting the template type (select/unselect), and by giving patterns for the antecedent and the consequent. In addition, thresholds for rule frequency, confidence, and significance can be given.

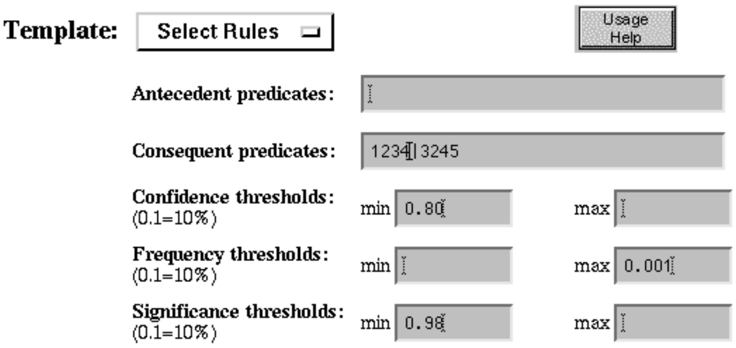

Figure 5: Example template in TASA

The selective template in Figure 5 matches rules that have

- an alarm of the class 1234 or 3245 as consequent and
- confidence at least 0.8, frequency at most 0.001, and significance at least 0.98, i.e., infrequent but rather strong and statistically significant rules.

The following two rules from Figure 4 match the criteria:

1234_11095 $\Rightarrow$ *1234_44545 (0.96, 0.0000152, 0.99).*

1234_11095 $\Rightarrow$ *1234_66656 (0.96, 0.0000152, 0.89).*

$\square$

Example 14 By clicking on an alarm name in a rule the user arrives in the alarm description; from there the user can follow a hyperlink to have a look on a histogram describing the arrival times of the alarm. Online user documentation is also available. $\square$

The system also contains pages to give a brief overview of the most descriptive parameters of the data set and statistics about the basic components of the analyzed data. The original data is also linked to the system since the users may want to see how the discovered knowledge is actually supported by the data.

An overview of both the analyzed alarm sequence as a whole and the individual alarms are produced automatically as a form of HTML documents (Figure 6). The overview can be used at the very beginning of the KDD process when excluding useless attributes.

New tools, such as Java programs, i.e. applets, and the oncoming extensions to HTML language open new possibilities also for fast creation of user interfaces for complex applications.

4.2 Usability Tests

4.2.1 Test Objectives and Methods

Even the most valuable information remains unusable, if it is not found or understood. Considering that, it is important that the users comprehend the

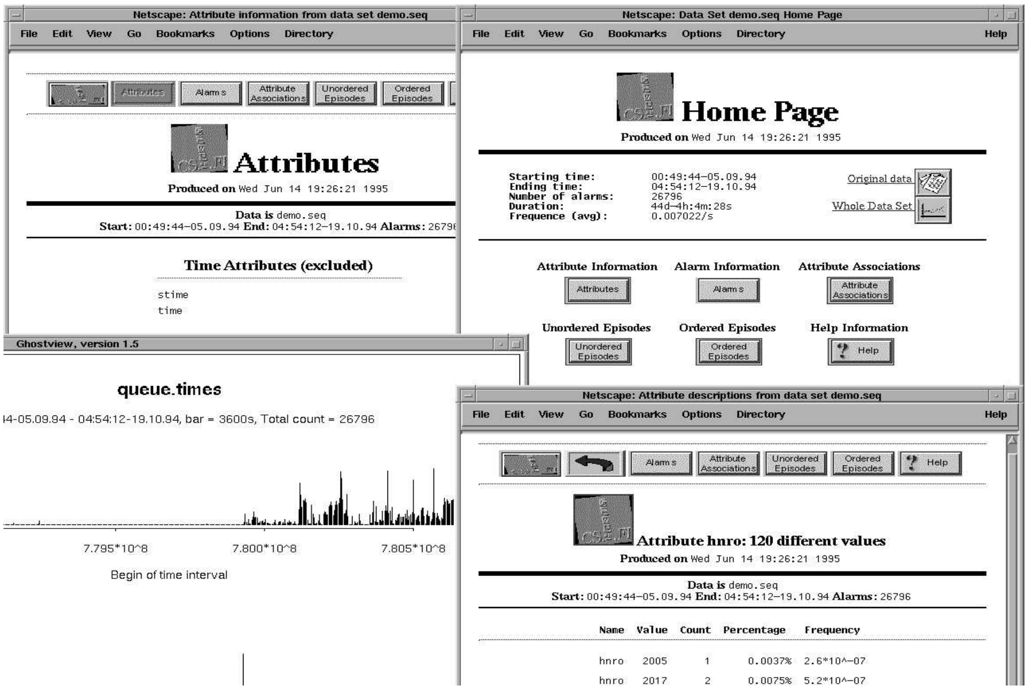

Figure 6: Overview of alarm sequence and alarm attributes in TASA

used presentation techniques and the connections between tasks and available methods.

Although many WWW-based applications have been developed during the last years, we are not aware of documented surveys concerning the use of WWW environment as the basis of a user interface of a more complex system.

The usability of the TASA system was tested in the usability laboratory of the Helsinki University of Technology. We had two goals:

- to improve the user interface of the system, and
- to evaluate the overall usability of the system, i.e. whether the system is usable for the intended task of telecommunication alarm data analysis.

Because the tested system is in prototype phase, the emphasis was put on *qualitative evaluation*. Qualitative evaluation concentrates on rather abstract aspects like “is the method and terminology usable”, or “if errors happen, why do they happen”. Qualitative evaluation is thus suitable for preliminary tests, whereas *quantitative evaluation* is more suitable for testing a more mature system or for comparing several systems. Quantitative evaluation deals, e.g., with concrete measures: times in which the users perform given tasks, error rates, etc. Our test consisted of the following elements:

1. heuristic evaluation,
2. visual walkthroughs, and
3. test cases modeling real-life situations and problems.

4.2.2 Tests

Heuristic Evaluation Before the actual user tests, the system was evaluated heuristically by the experimenters, i.e. by those in charge of running the tests. "Heuristic evaluation is done by looking at an interface and trying to come up with an opinion about what is good and bad about the interface" [19]. In practice some usability heuristics are used; we applied usability principles presented in [18], including, e.g.,

- *Simple and natural dialogue*: dialogues should contain only relevant information in a natural and logical order; and
- *Speak the user's language*: the used terminology should be non-system-oriented, but rather familiar to the user.

Similar rules, probably more widely known, are the eight "Golden Rules" of user interface design introduced by Shneiderman [24].

The evaluation method we used was the *pluralistic usability walkthrough* technique [3], which utilizes different kinds of expertise. In this case, the group of evaluators contained one developer, one usability specialist, and one novice user with no former experience of the system to be tested. The purpose of the heuristic evaluation was to overcome obvious problems before the actual user tests. Based on the results, the user interface was improved for the user tests.

User Tests The user tests consisted of three parts. First the background knowledge of the test persons was measured with a questionnaire. Then combinations of visual walkthroughs and test cases were performed. Finally, the users were asked to fill in another questionnaire. In combination with the last phase, the test persons were also interviewed. The role of the questionnaires and interviews was twofold: to get feedback about the tested system, and to survey the background of the test persons for analyzing the results with respect to the background of each particular user.

The group of test users consisted of four alarm handling experts from telecommunication companies. Their backgrounds were more or less technical, and they had no former experience with knowledge representation methods. All the test persons have to analyze, or at least understand, alarm data and network fault management in their everyday work. One additional person was used in a pilot test, to ensure that the test tasks were comprehensible, and to measure the time required to accomplish the tests.

The test persons were presented 11 different tasks from the following three classes:

1. Simple navigation tasks, such as: "You want to get an overview of the behavior of all the alarms in the alarm database. Where would you try to find it?"

2. Navigation tasks for more specified targets.

3. Tasks of finding interesting rules from the discovered ones, like: "You want to find very strong relationships that hold in the alarm sequence. How do you act?"

The tasks in the first and third set were alternated with visual walkthroughs. In a visual walkthrough the user was shown a screen from the user interface. Then he/she was asked to tell what he/she sees there, whether the terminology is clear, what kinds of functions can be invoked, and what the results of those functions would be (without actually trying). In the test tasks the *thinking aloud* method was used, i.e. the user was asked to express his/her thoughts verbally like during the walkthroughs.

The tests were recorded with two video cameras for later analysis.

While it took about one and a half hours for each test person to pass the tests, the total work effort required for the testing was over five person weeks, divided by three testers. Three weeks worth of work was spent before the user tests in planning, heuristic analysis, user interface (layout) changes in the system, pilot test, and in the preparation of the user tests. The tests themselves took three days worth of work, the analysis of video tapes and questionnaires took another week.

4.2.3 Results

The TASA system was acknowledged both visually and environmentally appealing; the hypertext based solution was considered to suit the application area containing lots of interconnected information. Also, the graphical user interface elements and shortcut links were regarded as useful.

On the basis of the results, navigation in the system was easy. However, many of the test cases were hard for the test users, and only half of them accomplished the test tasks of the third set (finding interesting rules). The main reason was that the terminology used in the system was quite unfamiliar. Especially this was the case for the episode and association rule concepts. The test persons were uncertain in connecting terms and the given tasks — they were given a document that described the concepts and gave examples of their use in the alarm handling domain, but obviously it did not help much.

After the test tasks, every test person was shown the intended way of doing the given tasks. In most cases that demonstration cleared the usage and architecture of the TASA to them. Demonstrating the usage seemed thus to be an effective way to train users.

The visual walkthroughs were very fruitful. The mental models of the users[4] were at least partially uncovered, clarifying the reasons for difficulties in the test cases. Visual walkthroughs also trained the test users to continue thinking aloud in the test cases.

The size of the test group was too small to get any statistically reliable results. However, recalling the qualitative and preliminary nature of the test, four persons can be considered an adequate size.

Despite the problems, the system as a whole got encouraging and positive comments, thus leading the main focus of improvement to the terminology. The rather abstract and formal concepts should be replaced by terminology more common in the application domain. A problem is that there is no universal terminology in the field of alarm handling: the test users themselves used different terms. More tests will be done from the basis of these results.

[4]I.e. how the users understand the structure, the concepts, and the functionality of the system.

5 Discussion

We have introduced a knowledge discovery methodology with the following two characteristics: (1) a large, unfocused collection of potentially interesting patterns is discovered at once, and (2) different views can be focused on the discovered patterns iteratively and interactively. The motivation is that after the discovery of all potentially interesting rules, iteration is very efficient.

The necessary conditions for the feasibility of this methodology are the following.

- There is an algorithm that produces, possibly without much focus, lots of potentially interesting patterns.
- The time requirement for discovering all potentially interesting patterns is not considerably worse than if the discovery was focused.
- The desired focus is not known definitely in advance.

There are well-known KDD formalisms that match this criteria. Our experience with both association and episode rules supports the claim that the methodology is useful in practice.

For formalisms stronger than association and episode rules, strong focus in the pattern discovery phase may be essential for keeping the computation tractable. For well-defined problems this is probably fine, but for exploring regularities in large data sets the interaction may be cumbersome.

The problem of locating a small set of truly interesting information is a generic problem in data mining (see, e.g., [20]): while formal statistical criteria for strength and significance of the discovered items abound, it is much harder to know which parts of the discovered knowledge really interest the user. For several KDD systems the focus of the pattern discovery is expressed as some sort of a template, often including both syntactic criteria, e.g., pattern form, or a fixed rule right-hand side, and statistical criteria for the interestingness, where the patterns and the criteria can be rather complex (e.g., Explora [13], IMACS [4]). However, we are not aware of any system where the focus can be set *after* the pattern discovery. Domain knowledge is used by many systems to avoid redundancy or to infer different interestingness measures. For instance, the Key Finding Reporter (Kefir) [21] detects and explains deviations and recommends corrective actions. Kefir is an example of a system where a large body of domain knowledge is used to guide the search for patterns without interaction with the user. Once the domain knowledge has been acquired and coded, a Kefir-based application generates automatically reports from databases of the domain.

Some possibilities exist for eliminating unnecessary details from a set of rules. Han et al. [6, 8] consider an attribute-oriented approach for finding generalizations of relations. This approach could perhaps be applied in finding generalizations of rules. On the other hand, association rule algorithms can be extended to find such general rules, too, if an attribute hierarchy is available [7, 10, 25]; the methods presented for association rules can be applied on episodes as well.

We described a set of simple but powerful tools for exploring large collections of association and episode rules. The approach is based on using templates, pattern expressions that match a class of rules based their syntax and an attribute taxonomy. Templates can be used both to make a selection from a rule collection, as well as to eliminate a selection from consideration. A system where templates can be easily added and modified and where the results are shown immediately, supports interactive exploration of large rule sets. In practice, templates have proved to be a powerful yet easy-to-use mechanism for exploring a large set of rules.

An overview of the implementation of the rule presentation phase of `TASA`, Telecommunication Alarm Sequence Analyzer, was given. We chose to use hypertext as the presentation method, since information with inherent links but with no particular structure can be made easily accessible with it. Also different material, such as rule lists and figures, can be easily presented.

For knowledge discovery to succeed, the tools and formalisms should be usable, otherwise the user is not able to find or understand the useful patterns. To discuss the matter on a very concrete level, we described a usability test of the implementation. The test consisted of heuristic evaluation, visual walkthroughs, and test cases modeling real-life situations and problems. The test cases contained navigation in the application and tasks of finding interesting rules, where interestingness was defined in various ways. The tests were performed by four domain experts with no prior experience with the system or KDD. The results are very useful. Overall, the system was considered visually and environmentally appealing. The comments of the test persons and also the problems they had help us in improving the user interface of the system. An important discovery was a problem with the terminology: some of the concepts of the system were not familiar to the test persons, leading to uncertainty in connecting terms and the given tasks. We are convinced of the profits of usability testing of KDD systems: several improvements we would perhaps not have thought of otherwise became obvious to us.

While the usability of the discovered rules was not within the scope of the test, we would like to report that useful rules have been discovered with `TASA` and taken into use in alarm handling systems.

Acknowledgements

We would like to thank the Helsinki University of Technology for providing us the usability laboratory for usability tests. We would also like to thank all involved in testing; both the experimenters and the test users.

References

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, May 1993.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] R. Bias. Walkthroughs: Efficient collaborative testing. *IEEE Software*, 8(5):94 – 95, 1991.

[4] Ronald J. Brachman and Tej Anand. The process of knowledge discovery in databases: A first sketch. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, July 1994.

[5] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1 –34. AAAI Press, Menlo Park, CA, 1996.

[6] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *Proceedings of the Eighteenth International Conference on Very Large Data Bases (VLDB'92)*, pages 547 – 559, August 1992.

[7] Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 420 – 431, 1995.

[8] Jiawei Han and Yongjian Fu. Exploration of the power of attribute-oriented induction in data mining. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399 – 421. AAAI Press, Menlo Park, CA, 1996.

[9] Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Knowledge discovery from telecommunication network alarm databases. In *12th International Conference on Data Engineering (ICDE'96)*, New Orleans, Louisiana, February 1996. To appear.

[10] Marcel Holsheimer, Martin Kersten, Heikki Mannila, and Hannu Toivonen. A perspective on databases and data mining. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 150 – 155, Montreal, Canada, August 1995.

[11] Peter Hoschka and Willi Klösgen. A support system for interpreting statistical data. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 325 – 345. AAAI Press, Menlo Park, CA, 1991.

[12] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, MD, November 1994. ACM.

[13] Willi Klösgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[14] Heikki Mannila and Hannu Toivonen. On an algorithm for finding all interesting sentences. In *Cybernetics and Systems Research '96*, Vienna, Austria, April 1996. To appear.

[15] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and

Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192, Seattle, Washington, July 1994.

[16] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, August 1995.

[17] Christopher J. Matheus, Gregory Piatetsky-Shapiro, and Dwight McNeill. Selecting and reporting what is interesting. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 495 – 515. AAAI Press, Menlo Park, CA, 1996.

[18] R. Molich and J.Nielsen. Improving a human-computer dialogue. *Communications of the ACM*, 33(3):338 – 348, 1990.

[19] Jakob Nielsen. *Usability engineering*. Academic Press, Cambridge, MA, 1993.

[20] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229 – 248. AAAI Press, Menlo Park, CA, 1991.

[21] Gregory Piatetsky-Shapiro and Christopher J. Matheus. The interestingness of deviations. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 25 – 36, Seattle, Washington, July 1994.

[22] Arun P. Sanjeev and Jan M. Zytkow. Discovering enrollment knowledge in university databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 246 – 251, 1995.

[23] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 432 – 444, 1995.

[24] Ben Shneiderman. *Designing the user interface: Strategies for effective human-computer interaction, 2nd edition.* Addison-Wesley, Reading, MA, 1992 (first edition in 1987).

[25] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 407 – 419, 1995.

[26] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping of discovered association rules. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 47 – 52, Heraklion, Crete, Greece, April 1995.

[27] Robert Zembowicz and Jan M. Zytkow. From contingency tables to various forms of knowledge in databases. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 329 – 349. AAAI Press, Menlo Park, CA, 1996.