A Guided Tour through the Data Mining Jungle

Robert Engels[1], **Guido Lindner**[2] **and Rudi Studer**[1]
[1] University of Karlsruhe, Institute AIFB, D-76128, Karlsruhe,
Email: {engels,studer}@aifb.uni-karlsruhe.de
[2] Daimler Benz AG, Research and Technology, F3S/E
p.o. Mercedes Benz AG, T-402 D-70322 Stuttgart
Email: lindner@str.daimler-benz.com

May 6, 1997

Abstract

An important success factor for the field of KDD lies in the development and integration of methods for supporting the construction and execution of KDD processes. Crucial aspects in this context are the (incremental) development of a precise problem description, a decomposition of this top level problem description into manageable and compatible subtasks which can be reused, and a selection and combination of adequate algorithms for solving these subtasks. In this paper we describe an approach for supporting the systematic decomposition of a KDD process into subtasks and for selecting appropriate problem-solving methods and algorithms for solving these subtasks. We propose to use pre-/postconditions (i) to characterize (sub)-tasks and methods, (ii) to guide the decomposition process, and (iii) to handle the dependencies between the subtasks and the methods. On the other hand we exploit and integrate techniques to describe the characteristics of the available (database) data to further guide the selection of applicable methods and algorithms. We use statistic measures, measures from the field of machine learning, e.g. missing values or noise, as well as information available in the database schemata, e.g. attribute types or the size of relations. Our approach has been partially integrated into the CLEMENTINE framework and has been used to develop a real world application in the area of prediction.

Keywords : *KDD-processes, User-support, Task Decompositions, Knowledge Acquisition*

1 Introduction

Knowledge Discovery in Databases (KDD) is currently a fast growing field both from an application and from a research point of view (Fayyad *et al.* 1996). This is due to the fact that companies see a high chance for deriving valuable information from the huge amount of available data which can then be used for improving their business. We think that an important success factor for the KDD field lies in the development and integration of methods for supporting the construction and execution of KDD processes (compare e.g. (Brachman & Anand 1996)). Crucial aspects in this context are the (incremental) development of a precise problem description, a decomposition of this top level problem description into manageable and compatible subtasks which can be reused, and a selection and combination of adequate algorithms for solving these subtasks (Engels 1996). Therefore, methods and corresponding tool support are required which assist the developer of a KDD application in building up a high quality KDD process with as less effort as possible.

In this paper we describe an approach for supporting the systematic decomposition of a KDD process into subtasks and for selecting appropriate problem-solving methods and algorithms for solving these subtasks. Our approach is on the one hand based on the notion of task analysis (Chandrasekaran, Johnson, & Smith 1992) and reusable problem-solving methods ((Eriksson *et al.* 1995), (Breuker & van de Velde 1994)). We propose to use pre-/postconditions (i) to characterize (sub)-tasks and methods, (ii) to guide the decomposition process, and (iii) to handle the dependencies between the subtasks and the methods. On the other hand we exploit and integrate techniques to describe the characteristics of the available (database) data to further guide the selection of applicable methods and algorithms. We use statistic measures (see e.g. the result of the STATlog project (Michie, Spiegelhalter, & Taylor 1994)), measures from the field of machine learning, e.g. missing values or noise, as well as information available in the data dictionary, e.g. attribute types or the size of relations. Our approach has been partially integrated into the CLEMENTINE framework and has been used to develop a real world application in the area of prediction.

The rest of the paper is organized as follows: we first describe the application problem and the solution we have developed using the CLEMENTINE tool. In section 3 we introduce our approach for supporting the construction of KDD processes. How we have used this approach for solving the given application problem is described in section 4. Finally, we discuss related work and provide a conclusion.

2 Application Environment

2.1 The KDD-Tool

Our approach is partially integrated in the CITRUS project of Daimler Benz (Wirth *et al.* 1997). Tool support in CITRUS is built on an existing commercial KDD-Tool, Clementine[1]. This tool supports all steps of the KDD-process. In Clementine the KDD-process is represented as data flows, called streams, which can be created by the user through a visual programming interface. The user can select icons which represent data sources, data manipulation algorithms, data mining algorithms, graphical and statistical data analysis techniques. These icons are combined into streams. Part of the CITRUS project deals with user-guidance to give support during the different stages of the KDD-process and to interactively construct a KDD-solution for a given problem in the form of such streams (Engels 1996).

2.2 Application Scenario

This section gives an introduction to a real world application which was presented at the KDD96 (Wirth & Reinartz 1996). We will use this example to illustrate our ideas on providing user guidance for KDD-processes. This example of a multistrategy process was developed for an early warning system in a quality information system by Mercedes Benz AG. The approach is called the early indicator approach. The main idea is to find and characterise sub populations of faulty cars that already show future behaviour of the whole car population at a later point in time. Such a sub population can then be used for prediction.

The early indicator approach is mainly defined by three steps:

1a Construct a fault profile for characterising faulty cars at a certain time T_X. Such a fault profile is based on so-called dynamic attributes, i.e. attributes that have values that change over time. Examples of these are mileages, number of faults, costs, etc. Dynamic attributes are important for the selection process that should be able to make predictions, since they are the attributes that change over time and thereby provide the possibility to select a dataset for modelling. A model to classify the cars using those dynamic attributes will result from this step.

1b In the second step one selects faulty cars at an earlier time point (T_Y, where $T_Y < T_X$) and classifies them using the model from step one. The cars which have the same characteristic fault profile at T_Y as the cars at T_X are called early indicators cars (EIC). In the next step one selects static attributes for the EIC (and NonEIC) concerning the car configuration and simple observational attributes concerning topics like the sales area of the cars. Such attributes are completely independent from the fault profile attributes.

2 Now one can learn characterisations for the EIC using these static attributes. The characterisation can be used for the fault prediction of another (later) production period.

In steps 1 and 2 there are different possibilities to choose a Machine Learning or Data Mining algorithm. In the original approach a modified implementation of ECOWEB was used for step one and a CN2 implementation (Clark & Niblett 1989) as well as the C4.5 algorithm (Quinlan 1993) used in step 2. In figure 1 a realisation of this approach with Clementine is given. Of the last two algorithms used in the original approach only C4.5 has been integrated in Clementine.

In figure 1 two streams are found. The first stream (DLO12-TypeNode-...)is the realisation of step 1a. In the second stream steps 1b and 2 are realized. The substream (DLO06-TypeNode-Fault_Class) classifies the cars at T_Y and the merge node joins the two data sources without the fault profile attributes. The rest of the stream is the realization of step 2.

Since we are defining a framework for providing user guidance in Clementine we implemented the approach mentioned above according to Clementine's possibilities. This implementation has shown to be acceptable for the domain experts.

Through this example it becomes clear that the KDD-process is an iterative and recursive process. In the following sections we will show our approach to user support.

2.3 The EIC-task Decomposed

For showing how user support works it is necessary to decompose the scenario of the former section. Figure 2 and figure 3 show that decomposition. The three

[1]Trademark by INTEGRAL SOLUTIONS LTD. (ISL) (Shearer 1996).

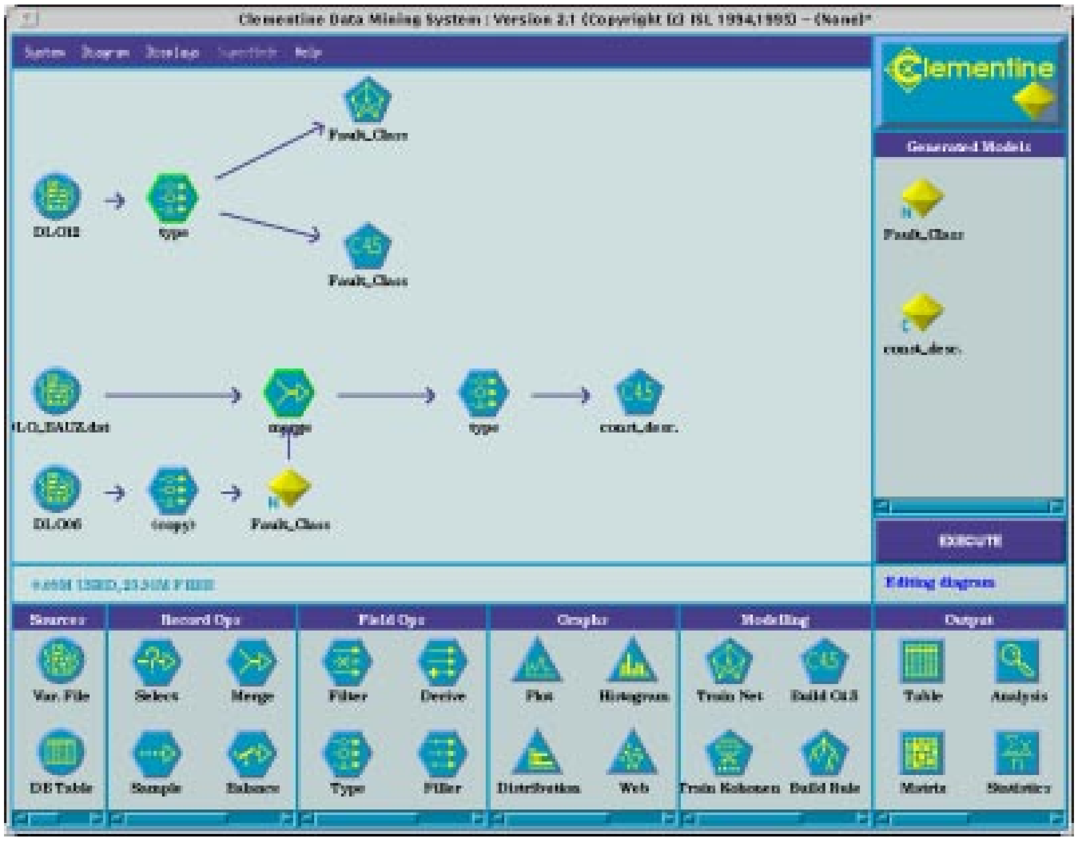

Figure 1: The example as implemented KDD-process

steps of the EIC approach can be mapped to the KDD-process model (Reinartz & Wirth 1995) in the following way.

Step 1a and step 1b comprise the preprocessing step of figure 2, where step 1a and 1b can be further refined as a KDD-process on its own that selects data on basis of a learned model. In step 1a we learn such a model for a certain point in time (Data Mining in figure 3), whereas in step 1b we deploy that learned model for data selection and change the representation from the fault profile of the cars to their configuration data (Deployment in figure 3). Step 2 forms the top level Data Mining step (see figure 2), that finally delivers the interpretable model that is looked for based on the defined set of attributes.

3 The Approach

This section deals with the approach that we take w.r.t. user guidance for KDD-processes. In (Engels 1996) we already presented a first outline to our approach, making use of concepts known from the knowledge acquisition community. In this paper we will extend upon the terminology and ideas introduced in (Engels 1996). The following sections will deal with the observation that KDD-processes often have a recursive nature, something which is compatible with our approach to task decomposition. Furthermore, the two-staged process of mapping tasks to algorithm classes as well as the final selection of an algorithm that matches the task and data characteristics is dealt with.

3.1 Recursiveness in KDD Processes

The decomposition of a task is a refinement problem that we solve using a multi-strategy approach. A framework describing the toplevel decomposition of a KDD-task is available from (Reinartz & Wirth 1995) and is used for the toplevel decomposition in figure 2. The iterative nature of KDD-processes is certainly not denied in our point of view, but is less important for the proposal of *initial* solutions for a KDD-problem. This means that in our framework one finds references to the iterative nature of KDD-process, e.g. in the method for the postprocessing subtask marked with "UI" (User Interaction) in figure 2.

On the one hand we refine the toplevel task according to a library of so-called PSMs[2] and doing so, reuse predefined methods that solve certain specified tasks. For other parts of the task decomposition reusable components might not be available. In such cases we can either look for a single technique solving the subtask or, when that does not solve the problem, use a planner in order to find a series of techniques that can solve the subtask. Many KDD-processes show recursiveness in the sense that a subtask of the task decomposition really is a (smaller) KDD-process on its own. This is not to mix up with iterativeness, where certain subtasks are repeated until a certain criterion is reached. Recursiveness also shows up in our example (compare figure 2 and figure 3) and let us reuse the same PSM that describes the KDD-process at the top level as well as at the data preprocessing stage. A singular PSM can in this case be retrieved from a library and reused twice in different instantiations. The several stages that are defined in such a PSM introduce certain constraints on its subtasks. Such a framework then is used to guide an initial decomposition of the task at hand.

[2]Problem Solving Methods. See also: (Breuker & van de Velde 1994), (Angele, Fensel, & Studer 1996), (Fensel *et al.* 1996) for more on PSM's.

3.2 Assigning Algorithm Classes to Tasks

Decomposing an initial task using PSMs delivers a tree like structure describing the task at several levels of abstraction. Each PSM is described by its preconditions and postconditions and together they define the functionality of a PSM. Tasks and subtask are described using the same concept of pre- and postconditions so that tasks can be mapped on methods.

Given the set of modelling algorithms that is used in KDD-processes it might occur that only a subset of them is applicable. A selection of an algorithm class is then made. The left part of figure 4 shows the relation between problem characteristics described by a task decomposition and the selection of algorithm classes. Such an algorithm class should later be broken down until a single algorithm can be proposed.

3.3 Algorithm Selection using Data Characteristics

An important part for planning the process is the selection of the Data Mining techniques (algorithm). The choice of the algorithm that can be used here depends on two things.

First it is important which kind of task is identified during the task analysis phase, f.e. the task can be a classification problem or finding association rules, etc.. So the task itself determines characteristics that possibly point to groups of Data Mining techniques. Furthermore, even for solving a specific type of task several different techniques may be applied in general.

Secondly the data itself includes a lot of information on which the selection of a Data Mining technique should be based (See figure 4). Since most large databases comes with a data dictionary, it is natural

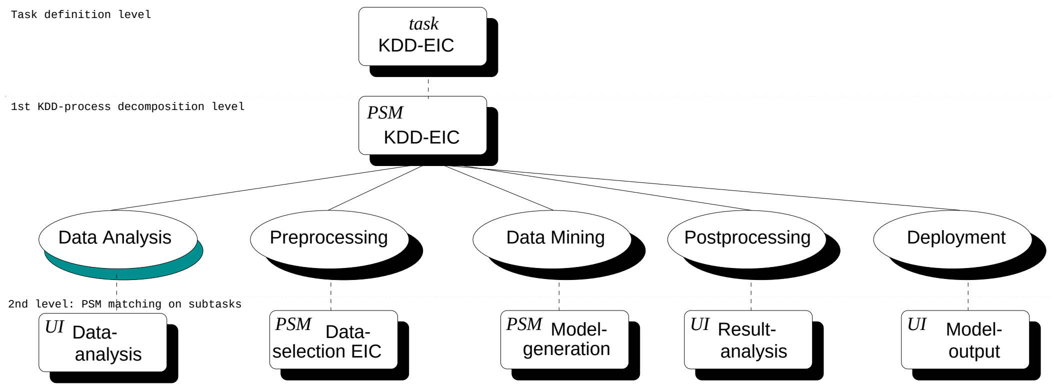

Figure 2: Composition of task, KDD-process model and PSM's matching on subtasks

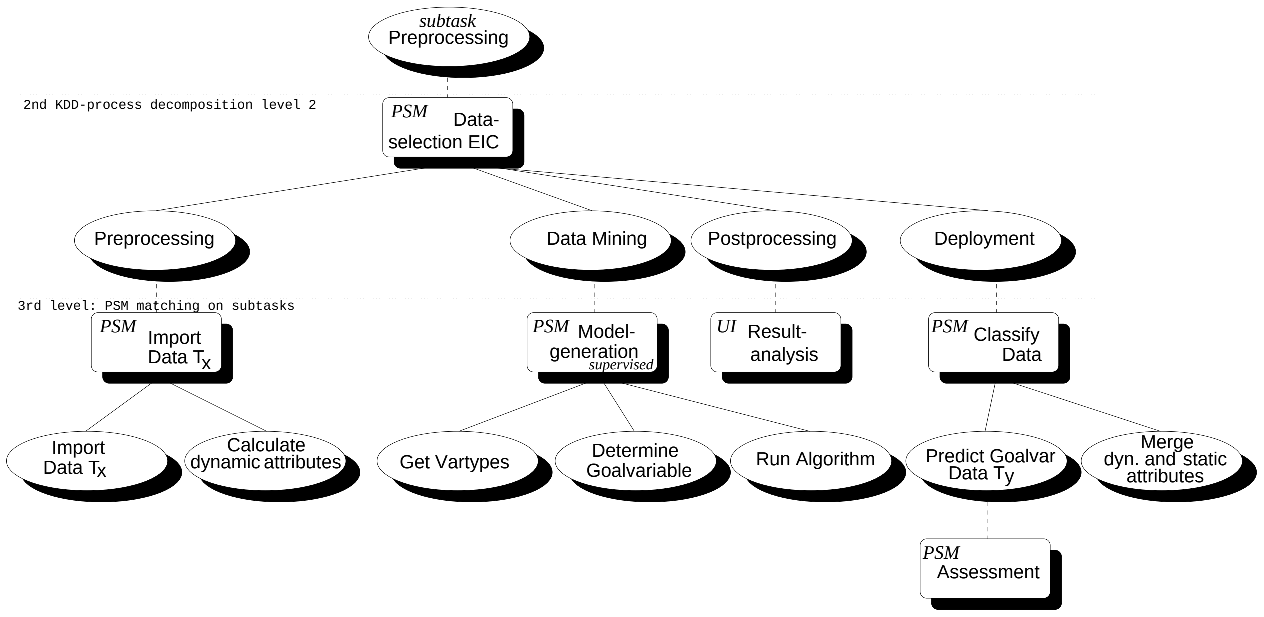

Figure 3: Decomposition of PSM KDD-Datapreselection of figure 2.

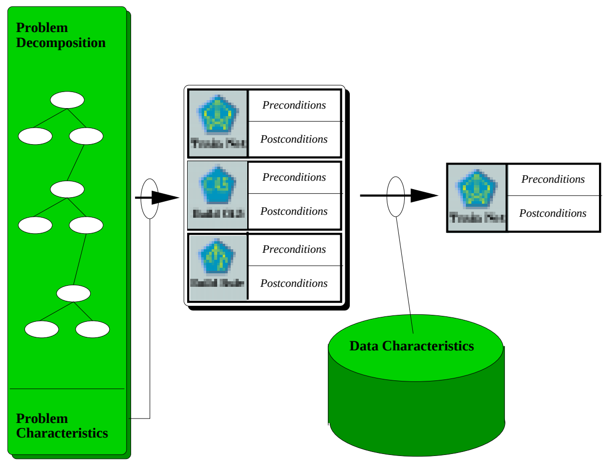

Figure 4: The process of selecting an algorithm component for a task.

to use this data dictionary to extract characteristic information on the data. Furthermore simple statistic measures like standard deviations, means, attribute types and possible sets, possibilities etc. that characterise and describe the data can be calculated. In the Statlog project (Michie, Spiegelhalter, & Taylor 1994) such measures were used for determining the applicability of statistical and data mining algorithms. We aim at using similar measurements for selecting algorithms which fit to the identified subtask.

Third, significant measurement types for characterization of datasets are measurements that are proven to be useful for describing machine learning algorithms (Michie, Spiegelhalter, & Taylor 1994). Here we want to exploit information on the percentage of positive and negative examples that the database contains, missing values, consistency of the dataset, etc.

4 How to Juggle the Jungle; the Example in our Approach

This section aims at clarification of some of the parts concernced with our approach by using the example of section 2. The decomposition of the application task in figure 2 and figure 3 have their implemented counterpart in figure 1. After showing the idea of recursiveness and iterativeness as represented in our approach it provides an example on the how and why of assigning algorithm classes to subtask types. There are a few steps that are basic in our approach:

- A description of his or her application problem.
- Mapping of this initial problem onto an appropriate task.
- Refinement of this task until a certain level of refinement is reached.

For the next sections we will presume that there already is a problem description available as well as a mapping to an initial appropriate task type. We will use these task characteristics of the example to show how an initial algorithm class is selected. Then the same is valid for the last section, where an example is provided for a case where the data characteristics play a role in the final algorithm selection.

4.1 Recursiveness in the Example

Our example clearly shows the recursive nature of KDD-tasks. On the top level the KDD-process is decomposed according to the steps of a standard KDD-process (see figure 2). Upon decomposition the need for a preprocessing step arises in order to be able to build a model with user defined characteristics in the modeling subtask (Data Mining in figure 2). Such characteristics can be dynamic attributes from the dataset[3]. In our case it is required to redescribe the dataset (subtask "calculate dynamic attributes" of figure 3) since the preconditions of the modeling subtask are not satisfiable using the data in the dataset. Relevant attributes (i.e. the average mileage between repairs), can be calculated from the database. The PSM that is selected for performing this preprocessing in our example resembles a KDD-process of its own (figure 3). Requirements here are that one uses dynamic attributes to model the dataset. This subset is taken from a certain time point. The generated model in this step (PSM$_{modelgeneration}$ in figure 3) is then used to extract those examples out of a second data subset (from an earlier point in time) that show the same behaviour on these dynamic attributes. Lateron the static attributes (like the model, type, engine, accesories, etc.) are merged with the data subset resulting of this extraction ("merge" step in figure 3). During this preprocessing subtask all the subtasks of a KDD-process are found, except for the data analysis subtask.

Figure 5 shows how the elements of figure 1 fit onto the toplevel task decomposition. From the figure one can also see the complexity of the preprocessing stage, which is decomposed as a KDD-process of its own.

It is clear that in order to be able to derive such a refinement structure one needs to define the functionality of tasks and use them for selecting the appropriate PSMs and finally the right algorithms. Part of this retrieval is supported by annotations of KDD-processes and algorithms. A prototypical implementation in Clementine for annotation and retrieval is

[3]Dynamic attributes are required since they represent the changing characteristics of the dataset concept.

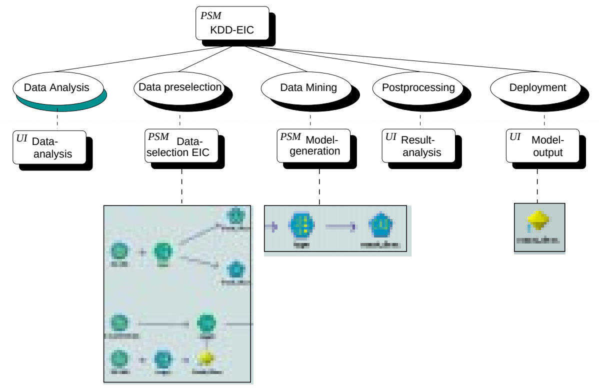

Figure 5: Toplevel decomposition mapped on Clementine stream.

made. The next two sections will deal with an example of this process as shown in figure 4.

4.2 Assigning Algorithm Classes to our Task

From a problem description one can retract pre- and postconditions. We will use a small set of such (informally represented) conditions in this section. According to our initial task description we are dealing with a problem where we are looking for:

1. $\exists$ Dataset (DS)
2. DynamicAttribs (DA) in DS
3. StaticAttribs (SA) in DS
4. $\exists$ Attrib (A) describing time
5. Possible classes that are marked as interesting by the user

Such characteristics are collected through the user guidance interface. An example of such a condition is the last user provided input, where we learn that there are classes known by the user that could be interesting for a first approach to the problem. First let us look for a way to reach the user's goals (also retrieved through the same interface):

1. model(M) should be usable for prediction.
2. model(M) should be interpretable by an expert (in this case in the form of production rules).
3. model(M) should model temporal dependency $T_Y < T_X$

For the first round we can decide to take a class of supervised learning algorithms due to the fact that a domain specialist provided input that there are classes known that might be used for learning and that there is a need for an interpretable model. This leaves us with a set of algorithms that are possibly interesting, namely the set of supervised learning algorithms that generate interpretable models. Now we need to close in on one of them in order to guide the task decomposition process.

4.3 Closing in on a specific Algorithm using Data Characteristics

In the dataset that is used for the EIC approach there are certain characteristics of data available. So now we can just follow a kind of 'strike-through' approach and eliminate algorithms that would not work in our case. An example of this would be to eliminate neural nets from the class of supervised learning methods, since our problem characteristics require an interpretable model. We can now retrieve the $\text{PSM}_{DataselectionEIC}$ for the preprocessing subtask based on the set of requirements and constraints that is formed by the triple $< DC, UR, BPC >$, where DC are the data characteristics, UR are the user-requirements, and BPC are the backpropagated constraints that follow from earlier decisions as shown below. We retrieve data characteristics from the Data Dictionary or using the parameters that are defined in the Statlog project. The software that was developed during the course of the Statlog project is also integrated in our tool. Doing so we get information on the set of classes that comprises our class descriptions (which are provided by the user, see point 5 section 4.2). In our case this set was not so large (4 items), otherwise we should have started user-interaction in order to decrease the number of classes, or maybe to ignore the concept and start data analysis without it. In our example one could think of replacing the data mining step of the preprocessing subtask with a subtree that enables unsupervised learning. A match on algorithm classes would in such a case f.e. refer to a set of clustering algorithms instead of the supervised learning techniques used in the example.

In the last section we showed a few conditions that play a role in our example. Some of those lead to the advice to take a certain learning algorithm (C4.5 in our case) for the Data Mining step of figure 2 (step 2 in section 2.2), since:

1. There are interesting classes that are provided by the user.
2. Those classes are not represented as numeric attributes.

3. The results can be represented either in the form of a decision tree, or as a set of production rules (this also starts further user interaction in order to make such decisions).

Such a decision for a specific algorithm then specifies the constraints (postconditions) for the prior step in our decomposition, together with the constraints that follow from data characteristics and the user input. For example, in our case, we got the requirement to predict some concept's attributes over time, and get some time depending data from our dataset. Together with the requirements from the Data Mining subtask, the need to model over static attributes and the availability of dynamic attributes we can define a preprocessing subtask that delivers a dataset that exactly provides the data with the required characteristics.

An additional feature of task decompositions is that we make use of the context that such a decomposition represents. It provides us possibilities to set parameters of algorithms (once they are selected) along the lines of the context they have to function in (f.e. one might set pruning, quality and iteration parameters according to data characteristics known or altered in other stream parts). In the end, such a process stream is what we want to provide to our user.

5 Related Work

Approaches that include the user as the key-factor for succesfully performing data analysis problems are found in the field of machine learning ((Piatetsky-Shapiro *et al.* 1996), (Fayyad *et al.* 1996), (Craw *et al.* 1992)), and the field of statistics ((Hand 1994b), (Hand 1994a)). Although this literature outline possibilities or needs for a stronger inclusion of the user, there are no applications mentioned where the ideas are tested, as is the case in our approach.

Breaking down a task's complexity by decomposition is an approach that is known from the field of Knowledge Acquisition, and is found in approaches such as KADS (Wielinga, Schreiber, & Breuker 1992), MIKE (Angele, Fensel, & Studer 1996) and Generic Tasks (Chandrasekaran, Johnson, & Smith 1992)). Those ideas combine very well with our aim to provide a user with a possibility to reuse parts of previous problem solutions and also provide a means for storage of newly defined KDD-processes.

From the KDD point of view there is also a need for defining a method for performing KDD. A few approaches in this direction can be found ((Brodley 1995), (Reinartz & Wirth 1995)). The steps that are defined in such methodologies can form the basics for our task decompositions and thus enable a first step decomposition, as shown in this paper. Furthermore, there seems to be more and more interest in ML applications. One particular survey dealt with the question if and how companies apply inductive logic techniques (Verdenius 1995). This research shares the conclusion with many other papers like (Piatetsky-Shapiro *et al.* 1996) and (Brodley 1995) that the process of machine learning application should primarily be user-driven, instead of data- or technology driven. Unfortunately, as (Verdenius 1995) also endorses, there are a fair amount of approaches that do show a data- or technology driven process, something which might be caused by a lack of methodology (see also (Piatetsky-Shapiro *et al.* 1996), (Brachman & Anand 1996)).

Two approaches exist that deal with the support of selecting data mining algorithms for a certain task. One approach (Brazdil, Gama, & Henery 1994) is based on learning of a decision tree for the applicability of algorithms given data characteristics. The other approach is the more user centered Consultant part of the MLT-project (Consortium 1993).

6 Conclusions and Future work

The approach taken in this paper shows some strengths and some weaknesses. Most of all, we are dealing with a huge unknown factor, and that is the user himself. The problem is that the user might be unaware of some important problem characteristics himself. Although providing feedback to the user in such cases is aimed at, it can happen that the system choses the wrong track due to his ignorance in those areas where it relies on user provided information.

Given this uncertainty we propose to extract some of the most important characteristics of such a user's problem, get hold of the characteristics in the data that he or she provides, and make use of a library containing reusable task decomposition parts that have been defined priorly.

Decomposing tasks, as done in our approach, means to the user that he gets a better overview of what he/she should do, and also enables us to reuse parts of the decomposition.

Preconditions and postconditions help us to make choices on which PSM's we can possibly join together in order to define an initial task decomposition. Such pre- and postconditions are also used for providing feedback to the user. In our example this happens when, based on the users problem description, a data mining step is defined that poses certain assumptions on its preceeding steps. These assumptions can require more user interaction such as definition of certain concepts, or redescription of attributes and so on. Our approach is not only applicable for processes as mentioned but is of course also valid for other algorithm classes (such as (conceptual) clustering, regression analysis, etc.).

For the future we will extend upon this framework and possibly incorporate a planning mechanism that can fill gaps in cases that there are no PSM's available from the (user defined-) library. Furthermore we will have to deal with the problem of finding a uniform representation for data mining algorithms and corresponding task characteristics. Finally, a knowledge base is needed that can help instantiate algorithms once their context is defined, in order to reduce the amount of effort that is needed to find a succesful approach as much as possible.

Although there is plenty to do, we think that concentrating more on the human in KDD-processes is worth the effort.

Acknowledgements

We want to thank Daimler Benz for granting and providing the possibility to get industrial input in the form of real world problems. Furthermore we thank our collegues for many fruitful discussions and Udo Grimmer for integrating the Analyst software in Clementine.

References

Angele, J.; Fensel, D.; and Studer, R. 1996. Domain and Task Modelling in MIKE. In *Proceedings of the IFIP WG8.1/13.2 Joint Working Conference on Domain Knowledge for Interactive System Design.*

Brachman, R. J., and Anand, T. 1996. The Process of Discovery in Databases: Human-Centered Approach. In Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurasamy, R., eds., *Advances in Knowledge Discovery and Data Mining.* MIT Press.

Brazdil, P.; Gama, J.; and Henery, B. 1994. Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning. *Proceedings of the European Conference on Machine Learning* 784:83 – 102.

Breuker, J., and van de Velde, W. 1994. *CommonKADS Library for Expertise Modelling.* IOS Press.

Brodley, C. 1995. Applying Classification Algorithms in Practice. In Aha, D., and Riddle, P., eds., *Working Notes for Applying Machine Learning in Practice: A Workshop at the Twelvth International Machine Learning Conference.* Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.: (Technical Report AIC-95-023).

Chandrasekaran, B.; Johnson, T. R.; and Smith, J. W. 1992. Task-Structure Analysis for Knowledge Modeling. *Communications of the ACM* 35(9):124–137.

Clark, P., and Niblett, T. 1989. The CN2 Induction Algorithm. *Machine Learning* 3:261–283.

Consortium, M. 1993. Final public report. Technical report. Esprit II Project 2154.

Craw, S.; Sleeman, D.; Granger, N.; Rissakis, M.; and Sharma, S. 1992. CONSULTANT: Providing Advice for the Machine Learning Toolbox. In Bramer, M., and Milne, R., eds., *Research and Development in Expert Systems*, 5–23.

Engels, R. 1996. Planning tasks for Knowledge Discovery in Databases; Performing Task-oriented User-Guidance. In E. Simounis, J. H., and Fayyad, U., eds., *Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases.*

Eriksson, H.; Shahar, Y.; Tu, S.; Puerta, A.; and Musen, M. 1995. Task Modeling with Reusable Problem-Solving Methods. *Artificial Intelligence* 79(2):293–326.

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurasamy, R. 1996. *Advances in Knowledge Discovery and Data Mining.* Cambridge, London: MIT press.

Fensel, D.; Schoenegge, A.; Groenboom, R.; and Wielinga, B. 1996. Specification and Verification of Knowledge-Based Systems. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW-96).* Calgary, Banff.

Hand, D. 1994a. Decomposing Statistical Questions. *Journal of the Royal Statistical Society* 317–356.

Hand, D. 1994b. Statistical Strategy: step 1. In Cheeseman, P., and Oldford, R. W., eds., *Selecting Models from Data: Artificial Intelligence and Statistics IV*, volume 89, 3–9. Lecture Notes in Statistics.

Michie, D.; Spiegelhalter, D.; and Taylor, C. 1994. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood.

Piatetsky-Shapiro, G.; Brachman, R.; Khabaza, T.; Kloesgen, W.; and Simoudis, E. 1996. An Overview of Issues in Developing Industrial Data Mining and Knowledge Discovery Applications. In Han, J.; Simoudis, E.; and Fayyad, U., eds., *Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases*, 89–95. Menlo Park, California: AAAI press.

Quinlan, J. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufman.

Reinartz, T., and Wirth, R. 1995. Towards a task model for KDD-processes. In Kodratoff, Y.; Nakhaiezadeh, G.; and Taylor, C., eds., *Workshop notes* Statistics, Machine Learning, and Knowledge Discovery in Databases. *MLNet Familiarisation Workshop*, 19–24.

Shearer, C. 1996. Using Driven Data Mining. In *Unicom Data Mining Conference.*

Verdenius, F. 1995. Applications of Inductive Learning Techniques: State of the Art. In *Proceedings of the 7th Dutch AI Conference (NAIC-95).*

Wielinga, B.; Schreiber, A.; and Breuker, J. 1992. KADS: A modelling approach to knowledge engineering. Special Issue "The KADS approach to knowledge engineering". *Knowledge Acquisition* 4(1):5–53.

Wirth, R., and Reinartz, T. 1996. Detecting Early Indicator Cars in an Automotive Database: A Multi-Strategy Approach. In Han, J.; Simounis, E.; and Fayyad, U., eds., *Proceedings of the 2nd Int. Conference on Knowledge Discovery in Databases.*

Wirth, R.; Shearer, C.; Grimmer, U.; Reinartz, T.; Schloesser, J.; Breitner, C.; Engels, R.; and Lindner, G. 1997. Towards Process-Oriented Tool Support for KDD. *submitted to PKDD97*.