Synthesis of Objects: A Rough Sets Approach to Automatic Construction and Maintenance of Software Components

Anders Torvill Bjorvand[1]

Abstract. This paper adresses the issue of automatic construction and maintenance of software components by utilizing data mining techniques. Some methods are shown, and more specifically, an implementation based on rough set theory and IBM PC systems is described. This approach is particularly useful for the synthesis of embedded decision support components and intelligent agents.

1 INTRODUCTION

In the following, a working knowledge of the object-oriented paradigm, data mining and rough sets (see eg [8] , [9] and [10]) is recommended.

The object-oriented paradigm, invented at the Norwegian Computing Center in the 1960s ([5]), has gained a tremendous momentum in both academia and industry, and is now widely regarded as the most important current methodology of constructing large information systems.

Compared to previous approaches, these information systems are easier to both build and maintain since they are divided into autonomous parts; making delegation of responsibilities easier. A lower degree of coupling between different parts of the system is a natural consequence of this approach.

This paper introduces the notion of Object Mining which is the process/idea of using data mining for both creation of and maintenance of such components. It will become clear that this approach merges creation and maintenance in a way that gives full support to the full life cycle of the system. Most practical investigations show that maintenance of a software system represents a greater cost than the initial creation.

1.1 Structure of this document

I will explain how the Object Mining technology merges these techniques to become a powerful tool for both development and maintenance of computer software. Much of this technology is explained through an example from real life showing the entire process from data analysis to the finalized component.

First we will introduce Object Mining, then we will illustrate it with some examples, then we will describe an implementation of it.

1.2 Related research

The idea of synthesizing software into components is new, but its ancestors are many. The most important are the fields of embedded and distributed expert systems and intelligent agents. A fair overview of this can be seen in [7] .

The Object Mining technology is, as will become evident, related to the field of expert systems. Traditional expert systems were constructed as monolithic applications that corresponded with the world through a human operator, but the adaptation to an embedded expert system seems a natural one, where the expert system automatically interacts with the rest of the system. This, however, would require changes to the intended object model of the system with respect to communications, interpreters, etc. The Object Mining technology produces components that fit directly into the existing object model. This is crucial to maintaining an elegant, understandable and therefore maintainable design.

This paper is exemplified by an implementation that produces components in the Java language conforming to the Javabean standard. The idea of exporting rulebased systems into runnable code of a programming language has been shown earlier in eg the Rosetta system that supports the export of Prolog rules (see [11]).

According to [1] , the research field of case-based reasoning can be divided into the following five areas: knowledge representation, retrieval methods, reuse methods, revise methods and retain methods. These areas form a process, and it is worth mentioning that the Object Mining technology fits well into this paradigm. An Object Mining component can be integrated with techniques for revision to perform self-assessment. In this way, through interaction with the environment, the component can learn/improve.

2 OBJECT MINING

An object can be viewed as a set of attributes/variables and a set of methods to set/get and/or analyze those attributes. For the great class of objects where this definition of an object holds true in a strict fashion, our method is applicable (some known exceptions are typical user interface objects and objects with attribute-values that are hard to discretize (typically pure mathematically speaking computational objects)). Data mining techniques take sets of attributes and their combinations of values as input. The product is a set of rules. By putting these attributes and rules together, we have an automatically produced object (also called software component in more modern terminology).

The Object Mining technology is the idea of utilizing data mining techniques to produce software objects. The choice of data mining techniques will vary with the domain that is being modelled, and is not part of the Object Mining technology even though this particular paper will further investigate the application of rough sets techniques to accomplish this.

To formalize, the Object Mining technology is best applied through the following three steps:

- Given an object model of your system, check which objects you either have or can easily sample data/measurements about.
- Given the databases for these objects (information systems in rough set terminology), choose the parameters for which you

[1] Department of Informatics, University of Oslo.
P.O. Box 1080 Blindern, 0316 Oslo
Email: torvill@trolldata.no

ECAI'98. *Workshop on Synthesis of Intelligent Agent Systems from Experimental Data*
Edited by J. Komorowski, I. Düntsch and A. Skowron

want the component to supply you with results. These parameters are called the decision parameters. The set of decision parameters can very well be the entire set of parameters. This will often be the case in components intended to perform simulations.
- Apply data mining techniques to produce a set of rules on the form "if *param_1=6* and *param_2=34* and .... and *param_n = 56* then decision_param_1=4 and ... and decision_param_m=75". The initial number of parameters should also decrease because the dependency analysis performed by the data mining system should prove some parameters redundant/superfluous. This is provided for very nicely in rough set theory by reducts.
- Create a component with the following features:
  - Internal variables representing all the parameters and decision parameters.
  - A data structure for the ruleset.
  - Methods for setting and reading the values of each variable, eg set_param_1(), get_param_1(), etc.
  - Methods retrieving the values of the decision parameters. These methods rely on data mining strategies for applying the ruleset.
- Incorporate these objects in your system according to the object model - interfaces might be an issue here.

The component described would be relatively simple. To use it, you set its conditional values and every time you want a decision value you ask for it explicitly - upon which it is calculated and returned. These tasks of interacting with the object might of course be delegated to several objects. Some might provide with data acquisition, some want to read the decisions etc.

A natural extension is an "active component" where the objects that rely on information from the component can be automatically informed of changes. This is accomplished by letting the dependent objects register with the component. Every time one of the parameters of the component changes, the set of decision parameters are immediately recalculated (without an explicit method invocation). If the values of one or more of the decision values have changed, an event informing of the change is automatically posted to the objects that have registered themselves.

Active objects fit well into the framework of process control and other systems where the object plays an active part. An obvious application domain where this sort of functionality would be required, is the field of intelligent agents. A more advanced way of registering for notification might be subscription lists with detailed information when each object should be notified. An object in charge of crisis situations might for instance only be noticed when the parameters reach a critical level.

We give some simple examples in the following section.

3 EXAMPLES

We have an example concerning Dogs and a more realistic example concerning credit assessment.

3.1 Dog example

We have a database of 10000 dogs. Each dog is stored in a record with several parameters like hair length, color, length of tail, breed etc. We have an image processing system that automatically extracts/recognizes all of these features except breed. We want a component that in each case can classify a dog based on certain features. Following the process outlined above, we get the following:

- We have a dog database, we choose the breed parameter as our only decision parameter.
- We apply data mining techniques and obtain the ruleset for determining the breed of dogs based on the information/experience in the database. Let's oversimplify and say that in our case, all other parameters than size and color are proved redundant in order to decide the breed. We get a ruleset with rules of the form: "if color=black and size=big then breed=Doberman Pincher"
- We can then automatically create a component with the following features:
  - Variables: color, size and breed.
  - A data structure for the rules.
  - Methods setting and reading the values of each variable:
    - set_param_size(), get_param_size()
    - set_param_color(), get_param_color()
    - The method: get_decision_breed() relying on data mining strategies for applying the ruleset.

To summarize, the Object Mining technology applies the techniques of data mining to the task/problem of producing objects/components.

The potential/usability of this technology will now be illustrated with a more realistic example.

3.2 Credit Assessment Example

Norwegian providers of access to cellullar telephone networks have experienced a tremendous increase in new customers over the past few years. Not all of those customers have proven to be credit worthy - ie they have not paid their phonebills. In order to deal with this problem, the Norwegian Telecom have introduced a credit assessment system. This system consists of an automatic and a manual part:

- Automatically check for flaws in the customer's credit history (previously unpaid invoices etc).
- If the customer proves credit worthy, the cellullar phone service may be provided. If not, the customer will be further evaluated by a manual check procedure.

See Figure 1 for a graphical representation of the process:

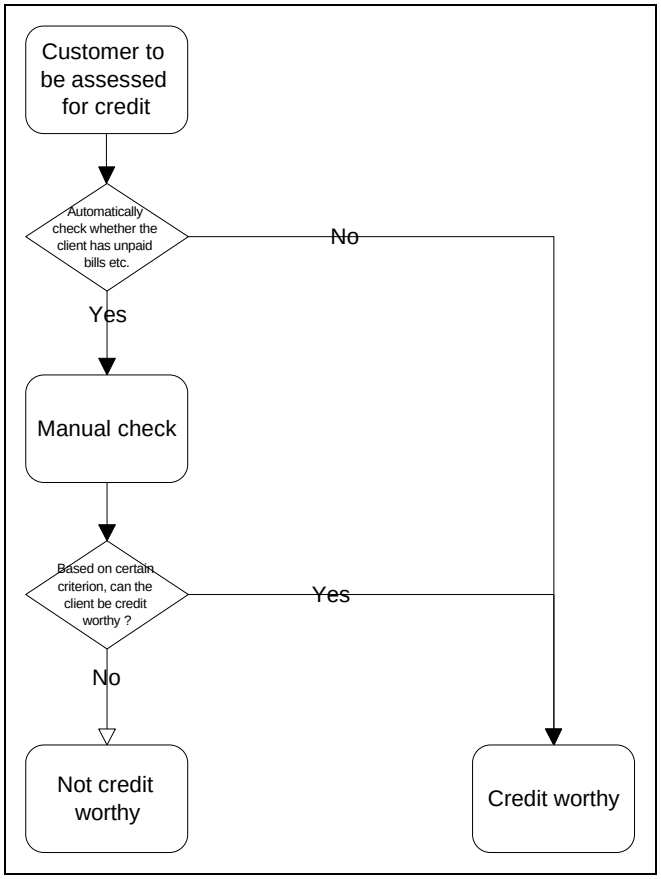

Figure 1. Credit assessment work flow

The component/system providing the automatic check in this particular case is hand tailored using the specifications of the trained credit assessors. When the manual assessors observe a customer profile that never gets through the automatic assessment but always gets through the manual assessment, they post a request for change concerning the automatic credit assessment system. When the computer department have the resources, they reprogram/alter the routines and/or ruleset of the automatic assessor.

The problems with this traditional approach are many:

- The process for discovery of new rules and recognition/automation of ad hoc practices is neither defined nor performed well.
- It is often a major task to rebuild the system to accomodate the new set of rules. This gives errors and expences. These expences are due to the cost of the programmers and the cost of system failure at the occurence of for instance system shutdown or plain wrong credit assessments due to erroneous rules.
- A sub optimal automation of the assessment process is achieved. This means that the number of cases to be handled manually will be unnecessarily large.

By augmenting their development efforts with the Object Mining technology, the initial (one time only) production of the system would go as follows:

- Collect information/results from the manual process of credit assessment
- Apply data mining techniques to reduce the number of attributes and produce a ruleset.
- From this, a credit assessment component can be automatically generated
- Build the total system with input, output and database connectivity including this crucial assessment component

The life cycle of the system involves maintenance and adaptation to new situations:

Every time you want the system to be updated - this can be programmed with wake-up events or by periodicity (eg every friday night).

The system can analyze the latest data and automatically adapt to the new situation. A new component is created, and the old one is simply replaced. Of course, components can also be produced that update themselves regularly based on corrections from the outside world.

The Object Mining technology thereby gives good solutions to all of the mentioned problems with the standard approach.

4 IMPLEMENTATION

The crucial parts of the technology have already been implemented as an add-on to the data mining system "Rough Enough" which is also copyrighted by the author of this paper. "Rough Enough" as a pure data mining tool (without the Object Mining capabilities) was developed in 1995 at the Nowegian University of Science and Technology in Trondheim, Norway. The Object Mining capabilities have been added during 1997 and parts of 1998 and is part of version 4.0 of the Rough Enough system. The system can be downloaded from the World Wide Web address: http://www.trolldata.com/.

4.1 Rough Enough

Rough Enough is a general purpose data mining tool based on rough set techniques. It is further described in [2] and [4] .

The most important capabilities of the system are:

- Runs on Windows'95 and Windows NT platforms on IBM PCs or compatibles.
- Editing and preprocessing of data
- Calculation of set approximations - upper, lower, boundary
- Calculation of cores and reducts
- Creation and application of rules
- Creation of software components based on the ruleset

All these features follow standard rough set approaches and are further described in [2] , [3] and [4] . We will use the remains of this section to describe the component creation functionality.

4.2 Component creation

The system will produce components conforming to the JavaBean standard from Sun Microsystems which is the most popular component standard right now built upon the Java language ([6]). OMG[2]s CORBA[3] IDL[4] files will also be automatically produced to allow the components more easily to be a part of an open distributed system.

[2] OMG - Object Management Group - a consortium of leading information technology companies creating and maintaining industrial standards for object oriented development

[3] CORBA - Common Object Request Broker Architecture - an OMG standard for distributed access to objects

[4] IDL - Interface Definition Language - a middle language between the CORBA server and the different objects which may be implemented in Java, C++, Smalltalk, Cobol etc.

The Java event model also supports the listener registration interface so that objects can register with the Object Mining developed components so as to receive updates through the event system. Active objects, as explained in section 2, is therefore straightforward in Java.

4.3 The dog example revisited

We will now return to the dog example and show a practical case of applying Rough Enough.

4.3.1 Case

We have a table of information about dog breeds (Table 1). We want a component that can classify between these breeds, which makes the breed-attribute our decision attribute.

Table 1 . Dog Information System

Object number	Breed	Tail	Hair	Size
1	Doberman Pincher	Short	Short	Big
2	Dalmatian	Long	Short	Big
3	Cocker Spaniel	Short	Long	Small
4	German Shepherd	Long	Medium	Big

4.3.2 Processing with rough set techniques

First, we import the data into Rough Enough: see Figure 2.

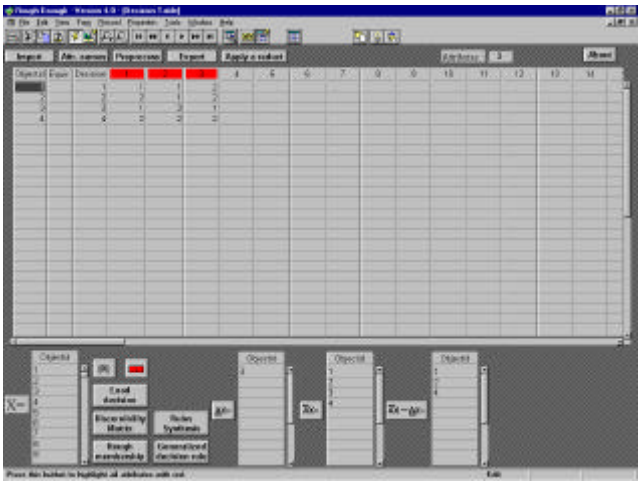

Figure 2. Main screen of Rough Enough

By applying the reduct analyzer, we get the result that tail and hair represent the relative reduct with respect to deciding the breed. This makes the size parameter superfluous in our case.

By utilizing the rules synthesis engine, shown in Figure 3, the ruleset is obtained.

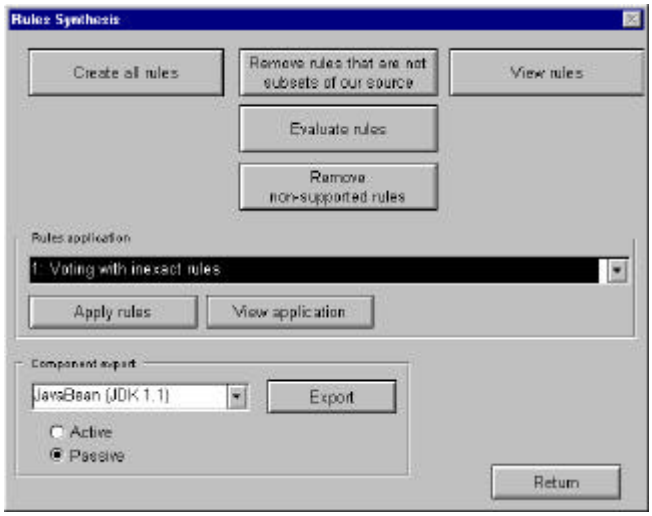

Figure 3. Rules synthesis screen of Rough Enough

4.3.3 Creation of a component

A component is created from the rules. Currently, the system does only support JavaBeans conforming to the JDK 1.1 standard (the most recent Java standard as of this writing).

The system supports bot passive and active objects and several rules-application strategies.

The particular component created in this example was a passive one called Dogs. It contains the following structure:

- Internal variables: AttributeTail, AttributeHair, AttributeDecision and Ruleset (double array)
- Methods:
  - Dogs(): constructor
  - getAttributeTail and setAttributeTail: sets and gets the tail attribute
  - getAttributeHair and setAttributeHair: sets and gets the hair attribute
  - calculateDecision: sets the value of the AttributeDecision variable utilizing the Ruleset variables
  - getAttributeDecision: invokes the method calculateDecision and then returns the value of the variable AttributeDecision

4.3.4 Listing of the source for the dog component

The following code shows that each conditional attribute has an internal variable with external methods for setting and getting its value. The rules are represented with a double array, and the decision is calculated on demand. Due to space limitations, the full text of the method calculateDecision() could not be included. It is not part of the general franework either, so displaying it would not be very interesting - the method for applying rules should be chosen from the domain, possibly even on a situation to situation basis.

```
public class Dogs implements
java.io.Serializable
{
  // Property declarations
  protected int AttributeDecision = 0;
  protected int AttributeTail = 0;
  protected int AttributeHair = 0;

  // Ruleset
  protected int[][] ruleSet = {
       {-1,   2,     4},
```

```
        {-1,   3,     3},
        {1,    1,     1},
        {1,    3,     3},
        {2,    1,     2},
        {2,    2,     4},
  };

  // Constructor
  public Dogs()
  {
  }

  protected void calculateDecision()
  {
        // here, different schemes for
        // rule-interpretation can be
        // implemented
  }

  public void setAttributeTail(int
attribute_value)
  {
        AttributeTail = attribute_value;
  }

  public int getAttributeTail()
  {
        return AttributeTail;
  }

  public void setAttributeHair(int
attribute_value)
  {
        AttributeHair = attribute_value;
  }

  public int getAttributeHair()
  {
        return AttributeHair;
  }

  public int getAttributeDecision()
  {
        calculateDecision();
        return AttributeDecision;
  }
}
```

4.3.5 Using the Component

Figure 4 shows a Java Applet utilizing the Dogs component. The Applet was created in the Symantec Visual Cafe Java environment. The Applet consists of textfields, a button, the Dogs component and a component that translates between the domain (tail-lengths, breeds) and our discretized component working internally in integers. The Dogs component was used as is, and did not need manual refinement. When the Tail and Hair parameters are given, pressing the Calculate-button returns with the name of the Breed.

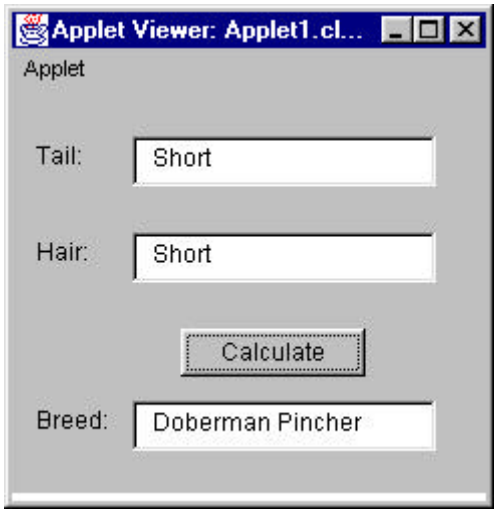

Figure 4 Java Applet using the Dogs JavaBean

5 A FRAMEWORK FOR AN INTERNET AGENT

A modern usage of agent technology is as information providers and filters on the Internet. Through customization, you can have an agent that listens to and/or polls a number of service providers to get the kind of information that you might be interested in. These agents often have algorithms that let them adapt to the user's need through interactive feedback.

An agent of this kind kan easily be modelled through Object Mining technology. The component that contains information on your likes and dislikes can both be created and interactively maintained/adapted through the framework described in this paper.

With the new version of the Internet Protocol, IP v6, a clearer distinction between different categories of services will be implemented. That will make it even easier to provide for the knowledge representation and data acqusition of such an approach.

6 CONCLUSION

We have identified the possibilities and importance of utilizing data mining techniques for synthesis of objects from experimental data. Several examples have been given, and an implementation is described.

Rough sets with its focus on discretized sets of attributes proves to be a well suited way of performing this data mining.

The application of data mining to software engineering, called Object Mining in this paper, shows great promise, and will hopefully be further investigated in the future.

The Object Mining technology has the potential of becoming an important supporting framework for agents and other embedded technologies that must adapt to their environment automatically or semi-automatically.

7 FUTURE RESEARCH

The techniques outlined here suggests that a central system and a database might generate components based on the latest information with regular intervals. The next logical step would be to let these components enhance themselves through rule improvement by feedback. This opens up several important applications for embedded decision support systems and autonomous intelligent agents. A problem, especially with embedded system, is that a lightweight object like the ones outlined in this paper often will be the only feasible way to go. A single

superobject might sound glorious, but it might be too large or too inefficient.

The efficiency of utilizing data mining to improve the software development and maintenance process should be verified through real projects.

By automating software construction relying on experimental data, the issue of correctness immediately becomes a major one. Completeness, inconsistency and handling of indeterminant decisions should be further investigated. There is a great potential for automating several of these tasks easier than in traditional correctness preservation of programs approaches.

Creating general “superobjects” with their own rule storage and techniques for exploring their own domain, a total system can be created with only one object class. In many ways, this is only a different perspective on agent oriented programming. I believe that an integration of that approach and the principles of genetic programming would be a fruitful direction of research.

ACKNOWLEDGEMENTS

I would like to thank my wife Annette for her support and my little daughter Susanne for her cheerful smile.

I would also like to thank the people at Oslo Research Park for fruitful discussions concerning the Object Mining technology.

REFERENCES

[1] Aamodt, Agnar and Plaza, Enric (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

[2] Bjorvand, Anders Torvill (1997). *’Rough Enough’ - A System Supporting the Rough Sets Approach.* Proceedings of the Sixth Scandinavian Conference on Artificial Intelligence 1997. Helsinki, Finland. IOS Press, Amsterdam.

[3] Bjorvand, Anders Torvill and Komorowski, Jan (1997). Practical Applications of Genetic Algorithms for Efficient Reduct Computation. Proceedings of the 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics. Wissenschaft & Technik Verlag, Berlin.

[4] Bjorvand, Anders Torvill (1997). *Rough Enough - Software Demonstration*. Proceedings of the 15th IMACS World Congress 1997 on Scientific Computation, Modelling and Applied Mathematics. Wissenschaft & Technik Verlag, Berlin.

[5] Dahl, O.J., Myrhaug, B. and Nygaard, K. (1968). *SIMULA 67 Common Base Language*. Norwegian Computing Center, Oslo.

[6] Gosling, James and Joy, Bill and Steele, Guy (1996). *The Java Language Specification*. Addison-Wesley.

[7] Kaelbling, Leslie Pack (1993). Learning in Embedded Systems. MIT Press.

[8] Pawlak, Zdzislaw (1982). “Rough Sets”. Printed in *International Journal of Computer and Information Sciences*, **11**, pp. 341-356.

[9] Pawlak, Zdzislaw (1991). *Rough Sets - Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Dordrecht.

[10] Skowron, Andrzej and Rauszer, Cecylia (1992). "The Discernibility Matrices and Functions in Information Systems". Printed in *Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory*. Edited by Roman Slowinski. Pp. 331-362. Kluwer Academic Publishers, Dordrecht.

[11] Øhrn, Aleksander. and Komorowski, Jan. (1997), *ROSETTA - A Rough Set Toolkit for Analysis of Data.* Proceedings of the Third International Joint Conference on Information Sciences, Fifth International Workshop on Rough Sets and Soft Computing, Durham, NC, USA, March 1-5, Vol. 3, pp. 403-407.