Knowledge Discovery for Training Intelligent Agents: Methodology, Tools and Applications

Pericles Mitkas

Department of Electrical and Computer Engineering,
Aristotle University of Thessaloniki,
and Informatics & Telematics Institute, CERTH,
Thessaloniki, Greece
Tel.: +30-2310-99-6390, Fax: +30-2310-99-6447
mitkas@eng.auth.gr

Abstract. In this paper we address a relatively young but important area of research: the intersection of agent technology and data mining. This intersection can take two forms: a) the more mundane use of intelligent agents for improved knowledge discovery and b) the use of data mining techniques for producing smarter, more efficient agents. The paper focuses on the second approach. Knowledge, hidden in voluminous data repositories routinely created and maintained by today's applications, can be extracted by data mining. The next step is to transform this knowledge into the inference mechanisms or simply the behavior of agents in multi-agent systems. We call this procedure "agent training." We define different levels of agent training and we present a software engineering methodology that combines the application of deductive logic for generating intelligence from data with a process for transferring this knowledge into agents. We introduce Agent Academy, an integrated open-source framework, which supports data mining techniques and agent development tools. We also provide several examples of multi-agent systems developed with this approach.

1 Introduction

The astonishing rates at which data are generated and collected by current applications are difficult to handle even for the most powerful of data processing systems. This windfall of information often requires another level of distillation to produce *knowledge.* Knowledge is the essence of information and comes in many flavors. Expert systems, knowledge bases, decision support systems, machine learning, autonomous systems, and intelligent agents are some of the many packages researchers have invented in order to describe applications that mimic part of the human mental capabilities. A highly successful and widely popular process to extract knowledge from data repositories is data mining.

The application domain of Data Mining (DM) and its related techniques and technologies have been greatly expanded in the last few years. The continuous improvement of hardware along with the existence of supporting algorithms have enabled the

V. Gorodetsky, J. Liu, and V.A. Skormin (Eds.): AIS-ADM 2005, LNAI 3505, pp. 2 – 18, 2005.

development and flourishing of sophisticated DM methodologies. Numerous approaches have been adopted for the realization of autonomous and versatile DM tools to support all the appropriate pre- and post-processing steps of the knowledge discovery process in databases.

Since DM systems encompass a number of discrete, nevertheless dependent tasks, they can be viewed as networks of autonomous, yet collaborating units, which regulate, control and organize all the, potentially distributed, activities involved in the knowledge discovery process. Software agents, considered by many the evolution of objects, are autonomous entities that can perform these activities.

Agent technology (AT) has introduced a wide range of novel services that promise to dramatically affect the way humans interact with computers. The use of agents may transform computers into personal collaborators that can provide active assistance and even take the initiative in decision-making processes on behalf of their masters. Agents participate routinely in electronic auctions and roam the web searching for knowledge nuggets. They can facilitate 'smart' solutions in small and medium enterprises in the areas of management, resource allocation, and remote administration.

Research on software agents has demonstrated that complex problems, which require the synergy of a number of distributed elements for their solution, can be efficiently implemented as a multi-agent system (MAS). As a result, multi-agent technology has been repeatedly adopted as a powerful paradigm for developing DM systems.

In a MAS realizing a DM system, all requirements collected by the user and all the appropriate tasks are perceived as distinguished roles of separate agents, acting in close collaboration. All agents participating in a MAS communicate with each other by exchanging messages, encoded in a specific agent communication language. Each agent in the MAS is designated to manipulate the content of the incoming messages and take specific actions/decisions that conform to the particular reasoning mechanism specified by DM primitives [1-2].

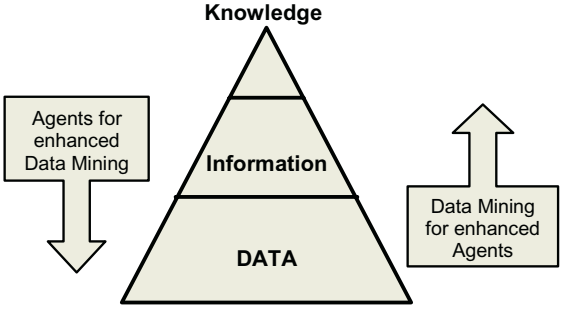

Fig. 1. Mining for intelligence

Considerable effort is expended to formulate improved knowledge models for data mining agents, which are expected to operate in a more efficient and intelligent way (see Fig. 1). Moving towards the opposite direction, we can envision the application

of data mining techniques for the extraction of knowledge models that will be embedded into agents operating in diverse environments [1].

1.1 The Synergy of Data Mining and Agent Technology

A review of the literature reveals several attempts to couple DM and AT. Galitsky and Pampapathi in their work combine inductive and deductive reasoning, in order to model and process the claims of unsatisfied customers [3]. Deduction is used for describing the behaviors of agents (humans or companies), for which we have complete information, while induction is used to predict the behavior of agents, whose actions are uncertain to us. A more theoretical approach on the way DM-extracted knowledge can contribute to AT performance has been presented by Fernandes [4]. In this work, the notions of data, information, and knowledge are modeled in purely logical terms, in an effort to integrate inductive and deductive reasoning into one inference engine. Kero et al., finally, propose a DM model that utilizes both inductive and deductive components [5]. Within the context of their work, they model the discovery of knowledge as an iteration between high level, user-specified patterns and their elaboration to (deductive) database queries, whereas they define the notion of a meta-query that performs the (inductive) analysis of these queries and their transformation to modified, ready-to-use knowledge.

In rudimentary applications, agent intelligence is based on relatively simple rules, which can be easily deduced or induced, compensating for the higher development and maintenance costs. In more elaborate environments, however, where both requirements and agent behaviors need constant modification in real time, these

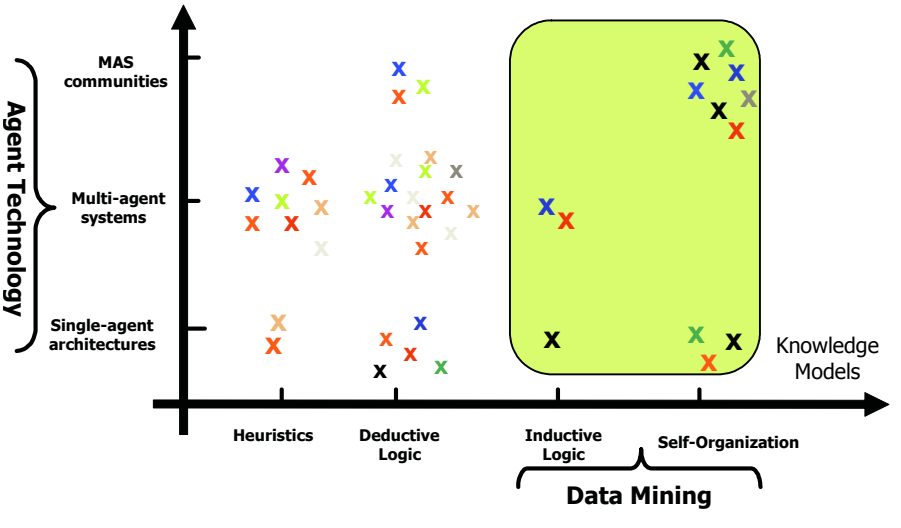

Fig. 2. Agent-based applications and inference mechanisms

approaches prove insufficient, since they cannot accommodate the dynamic transfer of DM results into the agents. To enable the incorporation of dynamic, complex, and reusable rules in multi-agent applications, a systematic approach must be adopted.

Existing agent-based solutions can be classified according to the granularity of the agent system and inference mechanism of the agents, as shown in Figure 2.

According to this qualitative representation of the MAS space, agent reasoning may fall under four major categories ranging from simple rule heuristics to self-organizing systems. Inductive logic and self-organization form two manifestations of data mining. Therefore, the shaded region delineates the scope of our approach.

The main thesis of this paper is that knowledge, hidden in voluminous data repositories, can be extracted by data mining and provide the logic for agents and multi-agent systems. In other words, these knowledge nuggets constitute the building blocks of agent intelligence. Here, intelligence is defined loosely so as to encompass a wide range of implementations from fully deterministic decision trees to self-organizing communities of autonomous agents. In many ways, intelligence manifests itself as efficiency. We argue that the two, otherwise diverse, technologies of data mining and intelligent agents can complement and benefit from each other, yielding more efficient solutions [1].

The dual process of knowledge discovery and intelligence infusion is equivalent to learning, better yet, teaching by experience. Indeed, existing application data (i.e., past transactions, decisions, data logs, agent actions, etc.) are filtered in an effort to distill the best, most successful, empirical rules and heuristics. The process can be applied initially to train 'dummy' agents and, as more data are gathered, it can be repeated periodically or on demand to further improve agent reasoning.

In this paper we describe a unified methodology for transferring DM-extracted knowledge into newly-created agents. As illustrated in Figure 3, data mining is used

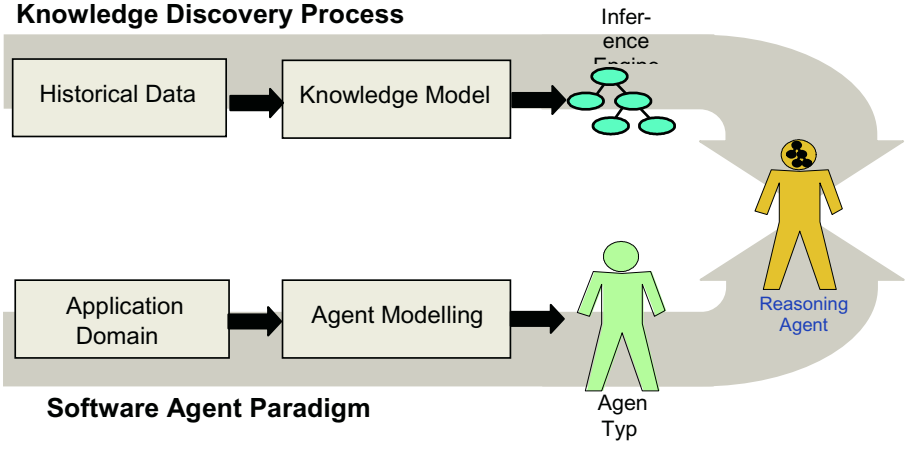

Fig. 3. Embedding DM-extracted knowledge into agents

to generate knowledge models which can be dynamically embedded into the agents.The process is suitable for either upgrading an existing, non agent-based application by adding agents to it, or for improving the already operating agents of an agent-based application. The methodology relies heavily on the inductive nature of data mining, while taking into account its limitations. Our perspective leans more towards agent-oriented software engineering (AOSE) than artificial intelligence (AI).

We have also developed an integrated platform, called *Agent Academy,* which consolidates the steps of our methodology and provides the user with access to a variety of software development tools in a single environment. *Agent Academy* (AA) is an open-source product, which combines data mining functionality with agent design and development capabilities [6,7]. We have used AA to implement a number of agent-based applications. Some of them are briefly described in this paper.

The remainder of the paper is organized as follows: Section 2 presents the methodology for agent training, while Section 3 describes the architecture and basic functionality of AA. Section 4 outlines a few multi-agent systems that were developed with AA and Section 5 contains a summary of this work and our conclusions.

2 Methodology

In this section, we present a unified methodology for MAS development, which relies on the ability of DM to generate knowledge models for agents. As shown in Figure 4,

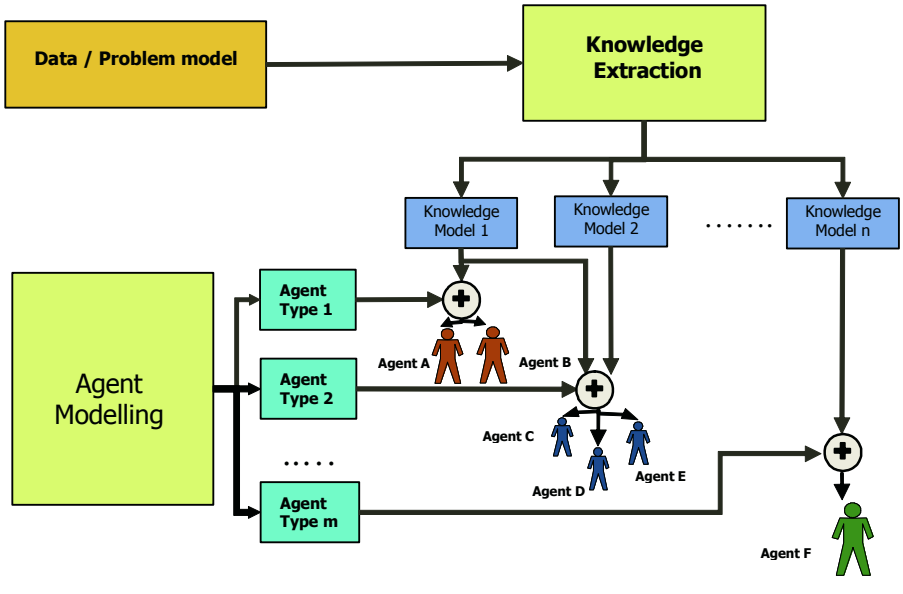

Fig. 4. Different agent types may receive one or more DM-extracted knowledge models

agent types, defined during the agent modelling phase, receive one or more knowledge models extracted through a separate process. Each agent type will give rise to one or more agent instances.

In our approach, we consider three distinct cases, which correspond to three types of knowledge extracted and to different data sources and mining techniques. These three types demarcate also three different modes of knowledge diffusion.

Case 1. Knowledge extracted by performing DM on historical datasets recording the business logic (at a macroscopic level) of a certain application
Case 2. Knowledge extracted by performing DM on log files recording the behavior of the agents (at a microscopic level) in an agent-based application, and
Case 3. Knowledge extracted by the use of evolutionary data DM techniques in agent communities.

The basic methodology encompasses a number of stages and is suitable for all three cases of knowledge diffusion. Most of the steps are common to all cases, while some require minor modifications to accommodate each case (see Figure 5).

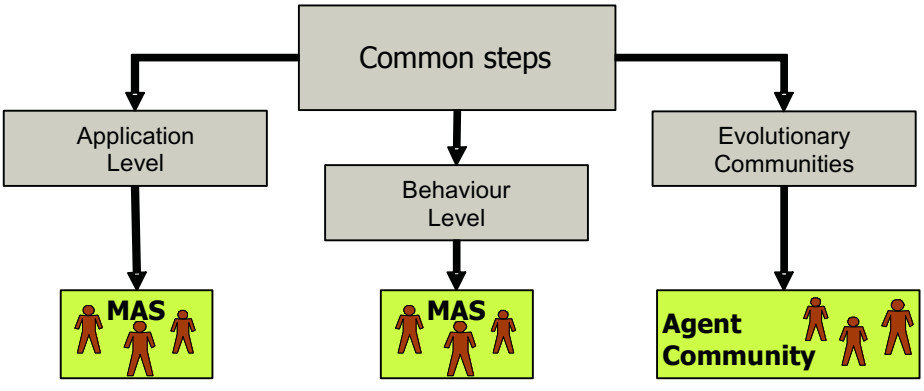

Fig. 5. Three cases of the methodology

The methodology pays special attention to two issues: a) the ability to dynamically embed the extracted knowledge models (KMs) into the agents and b) the ability to repeat the above process as many times as deemed necessary. Standard AOSE processes are followed, in order to specify the application ontology, the agent behaviors and agent types, the communication protocol between the agents, and their interactions. In parallel, DM techniques are applied for the extraction of appropriate KMs.

The ten steps of the methodology are listed below. They are also illustrated in the form of a flow diagram in Figure 6. It must be noted that steps 9 and 10 are optional and needed only when agent retraining is required. The interested reader may find a more detailed description of the methodology in Reference [1].

```
1.  Develop the application ontology
2.  Design and develop agent behaviours
3.  Develop agent types realizing the created behaviours
4.  Apply data mining techniques on the provided dataset
5.  Extract knowledge models for each agent type
6.  Create the agent instances for the application
7.  Dynamically incorporate the knowledge models to the
    corresponding agents
8.  Instantiate multi-agent application
9.  Monitor agent actions
10. Periodically retrain the agents of the system
```

Following this methodology, we can automate (or semi-automate) several of the processes involved in the development and instantiation of MAS. Taking an AOSE perspective, we can increase the adaptability, reusability, and flexibility of multi-agent applications, simply by re-applying DM on different datasets and incorporating the extracted KMs into the corresponding agents. Thus, MAS can be considered as efficient add-on components for the improvement of existing software architectures.

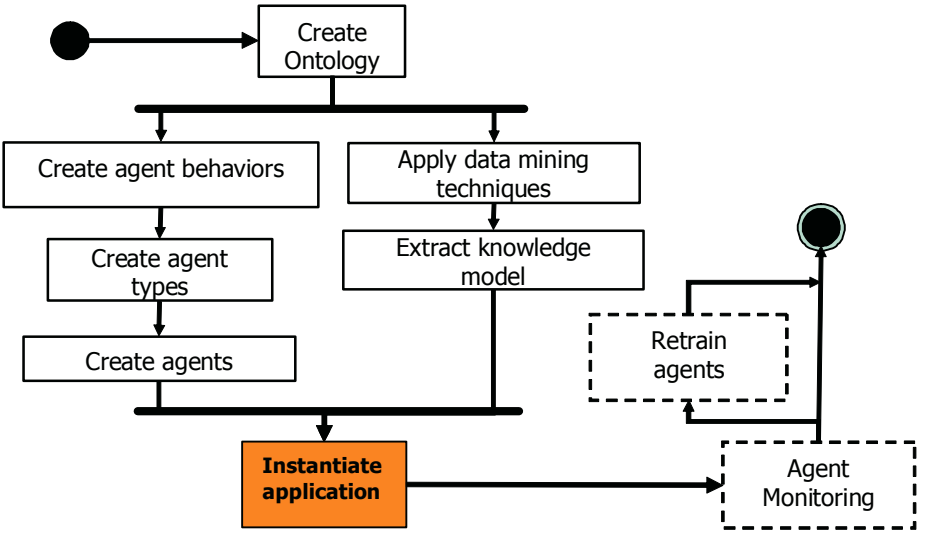

Fig. 6. The MAS development mechanism

Slight variations and/or additions may be needed to the presented methodology in order to adapt it to each one of the three cases. These differentiations may be either related to the agent behaviour creation phase (Cases 2 and 3 - e.g., model agent actions), the training phase (Cases 1, 2, and 3 - e.g., training set selection, knowledge model representation, retraining intervals), or the retraining phase (Cases 1,2, and 3 - e.g., retraining initiation).

Let us consider a company that would like to embrace agent technology and transform its processes from legacy software to a MAS. During the modelling phase, agent

roles and interactions are specified. Then, agent behaviours and logic models have to be defined, a process extremely complicated, since the specification of an appropriate (sound) set of rules is not always straightforward.

In most cases, domain understanding is infused to the system in the form of static business rules, which aim to satisfy refractory, nevertheless suboptimal, MAS performance. In the case the rules are set by a domain expert, there is a high probability of ignoring interesting correlations between events that may occur during system operation. Moreover, the rigidity of the rules introduced, cannot capture the dynamism of the problem at hand (the problem the MAS is assigned to solve). The extraction of useful knowledge models from historical application data is, therefore, considered of great importance.

3 Agent Academy: A Platform for Developing and Training Agents

We have developed an integrated platform that automates most of the steps outlined in the previous section. Agent Academy is an open-source software platform available at `http://www.source-forge.net/projects/AgentAcademy`. AA has been implemented upon the JADE [8,10] infrastructure, ensuring a relatively high degree of FIPA compatibility, as defined in [9,10]. AA is itself a multi-agent system, whose architecture is based on the GAIA methodology [11]. It provides an integrated GUI-based environment that enables the design of single agents or multi-agent communities, using common drag-and-drop operations. Using AA, an agent developer can easily go into the details of the designated behaviours of agents and precisely regulate communication properties of agents. These include the type and number of the agent communication language (ACL) messages exchanged between agents, the performatives and structure of messages, with respect to FIPA specifications [12-14], as well as the semantics, which can be defined by constructing ontologies with Protégé-2000 [15].

AA also supports the extraction of decision models from data and the insertion of these models into newly created agents. Developing an agent application using AA involves the following activities from the developer's side:

a. the creation of new agents with limited initial reasoning capabilities;
b. the addition of these agents into a new MAS;
c. the determination of existing, or the creation of new behaviour types for each agent;
d. the importation of ontology-files from Protégé-2000;
e. the determination of message recipients for each agent.

In case the developer intends to create a reasoning engine for one or more agents of the designed MAS, four more operations are required for each of those agents:

f. the determination of an available data source of *agent decision attributes*;
g. the specification the parameters of the data mining mechanism;
h. the extraction of the decision/knowledge model;
i. the insertion of this model to the agent.

Figure 7 illustrates the Agent Academy functional diagram, which represents the main components and the interactions between them. In the remainder of this section, we discuss the AA architecture and outline its main functionality.

3.1 Architecture

The main architecture of AA is also shown in Figure 7. An application developer launches the AA platform in order to design a multi-agent application. The main GUI of the development environment is provided by the Agent Factory (AF), a specifically designed agent, whose role is to collect all required information from the agent application developer regarding the definition of the types of agents involved in the MAS, the types of behaviours of these agents, as well as the ontology they share with each other. For this purpose, Agent Academy provides a Protégé-2000 front-end. The initially created agents possess no referencing capabilities ("dummy" agents). The developer may request from the system to create rule-based reasoning for one or more agents of the new MAS. These agents interoparate with the *Agent-Training Module* (ATM), which is responsible for inserting a specific decision model into them. The latter is produced by performing DM on data entered into Agent Academy as XML documents or as datasets stored in a database. This task is performed by the *Data Mining Module* (DMM), another agent of AA, whose task is to read available data and extract decision models, expressed in PMML format [16].

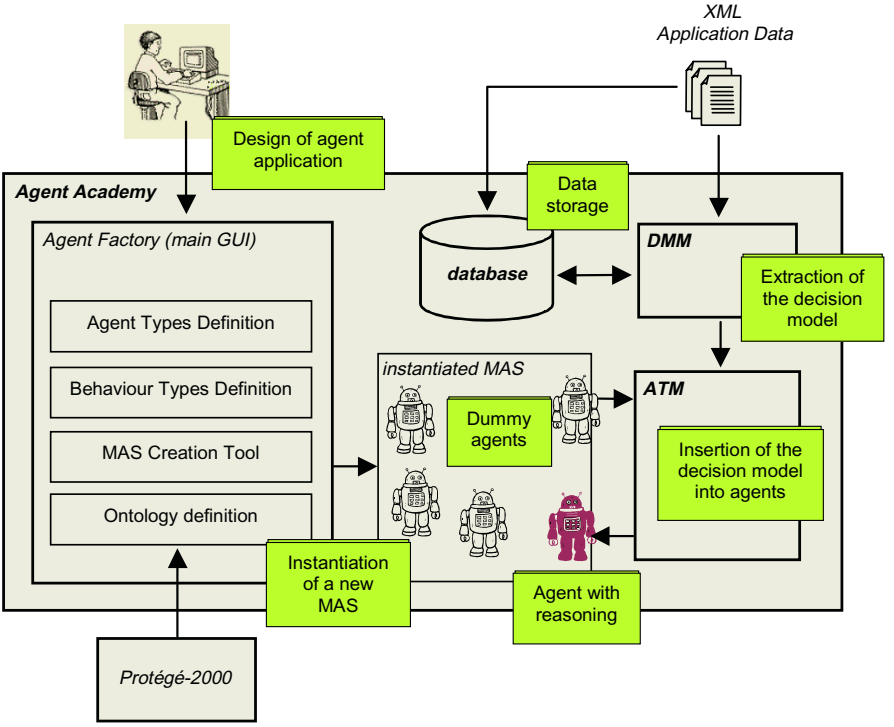

Fig. 7. Diagram of the Agent Academy development framework

AA hosts a database system for storing all information about the configuration of AA-created agents, their decision models, as well as data entered into the system for DM purposes. The whole AA platform was created as a MAS, executed on JADE.

3.2 Developing Multi-agent Applications

Agent Factory consists of a set of graphical tools, which enable the developer to carry out all required tasks for the design and creation of a MAS, without any effort for writing even a single line of source code. In particular, the Agent Factory comprises the Ontology Design Tool, the Behaviour Type Design Tool, the Agent Type Definition Tool, and the MAS Creation Tool.

a) Creating Agent Ontologies
A required process in the creation of a MAS, is the design of one or more ontologies, in order for the agents to interoperate adequately. The Agent Factory provides an *Ontology Design Tool*, which helps developers adopt ontologies defined with the Protégé-2000, a tool for designing ontologies. The RDF files that are created with Protégé are saved in the AA database for further use. Since AA employs JADE for agent development, ontologies need to be converted into special JADE ontology classes. For this purpose, our framework automatically compiles the RDF files into JADE ontology classes.

b) Creating Behaviour Types
The *Behaviour Type Design Tool* assists the developer in defining generic behaviour templates. Agent behaviours are modeled as workflows of basic building blocks, such as receiving/sending a message, executing an in-house application, and, if necessary, deriving decisions using inference engines. The data and control dependencies between these blocks are also handled. The behaviours can be modeled as *cyclic* or *one-shot* behaviours of the JADE platform. These behaviour types are generic templates that can be configured to behave in different ways; the structure of the flow is the only process defined, while the configurable parameters of the application inside the behaviour, as well as the contents of the messages can be specified using the MAS Creation Tool. It should be denoted that the behaviours are specialized according to the application domain.

The building blocks of the workflows, which are represented by nodes, can be of four types:

1. `Receive nodes`, which enable the agent to filter incoming FIPA-SL0 messages.
2. `Send nodes`, which enable the agent to compose and send FIPA-SL0 messages.
3. `Activity nodes`, which enable the developer to add predefined functions to the workflow of the behaviour, in order to permit the construction of multi-agent systems for existing distributed systems.
4. `Jess nodes`, which enable the agent to execute a particular reasoning engine, in order to deliberate about the way it will behave.

c) Creating Agent Types

After having defined certain behaviour types, the *Agent Type Definition Tool* is provided to create new agent types, in order for them to be used later in the MAS Creation Tool. An agent type is in fact an agent plus a set of behaviours assigned to it. New agent types can be constructed from scratch or by modifying existing ones. Agent types can be seen as templates for creating agent instances during the design of a MAS.

During the MAS instantiation phase, which is realized by the use of the MAS Creation Tool, several instances of already designed agent types will be instantiated, with different values for their parameters. Each agent instance of the same agent type can deliver data from different data sources, communicate with different types of agents, and even execute different reasoning engines.

d) Deploying a Multi Agent System

The design of the behaviour and agent types is followed by the deployment of the MAS. The *MAS Creation Tool* enables the instantiation of all defined agents running in the system from the designed agent templates. The receivers and senders of the ACL messages are set in the behaviours of each agent. After all the parameters are defined, the agent instances can be initialized. Agent Factory creates *default AA Agents*, which have the ability to communicate with AF and ATM. Then, the AF sends to each agent the necessary ontologies, behaviours, and decision structures.

3.3 Data Mining and Agent Training

The inference engine of an agent that needs to be trained is generated as the outcome of the application of DM techniques into available data. This operation takes place in the DMM. The mechanism for embedding rule-based reasoning capabilities into agents is realized through the ATM. Both modules are implemented as JADE agents, which act in close collaboration.

The flow of the agent training process is shown in Figure 8. At first, let us consider an available source of data formatted in XML. The DMM receives data from the XML document and executes certain DM algorithms (suitable for generating a decision model), determined by the agent-application developer. The output of the DM procedure is formatted as a PMML document.

PMML is an XML-based language, which provides a rapid and efficient way for companies to define predictive models and share models between compliant vendors' applications. It allows users to develop models within one vendor's application, and use other vendors' applications to visualize, analyze, evaluate or otherwise use the models. The fact that PMML is a data mining standard defined by DMG (Data Mining Group) [16] provides the AA platform with versatility and compatibility to other major data mining software vendors, such as Oracle, SAS, SPSS and MineIt.

The PMML document represents a KM for the agent we intend to train. This model is translated, through the ATM, to a set of facts executed by a rule engine. The implementation of the rule engine is provided by JESS [17], a robust mechanism for executing rule-based reasoning. Finally, the execution of the rule engine becomes part of the agent's behaviour.

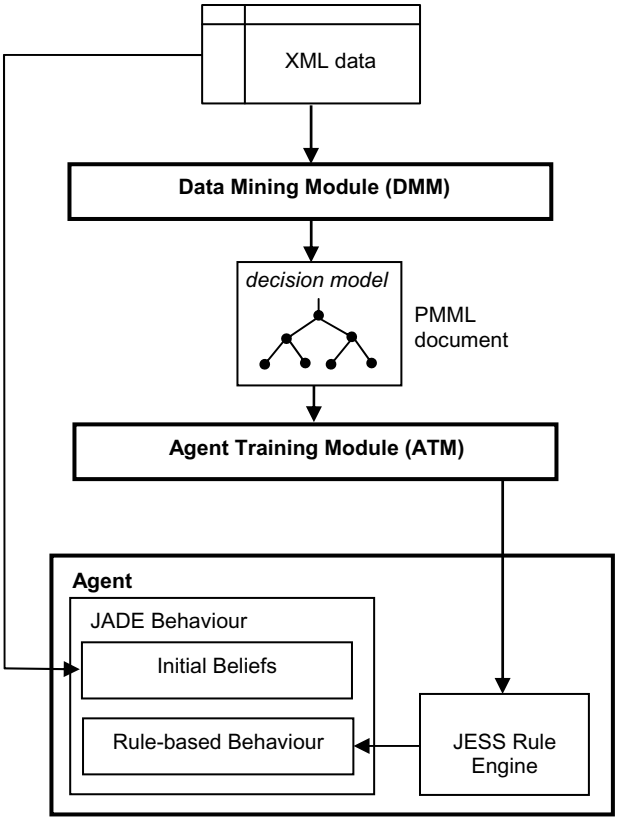

Fig. 8. Diagram of the agent training procedure

As shown in Figure 8, an agent that can be trained through the provided infrastructure encapsulates two types of behaviours. The first is the basic initial behaviour predefined by the AF module. This may include a set of class instances that inherit the `Behaviour` class defined in JADE. The initial behaviour is created at the agent generation phase, using the Behaviour Design Tool, as described in the previous section. This type of behaviour characterizes all agents designed by Agent Academy, even if the developer intends to equip them with rule-based reasoning capabilities. This essential type of behaviour includes the set of initial agent beliefs.

The second supported type of behaviour is the rule-based behaviour, which is optionally created, upon activation of the agent-training feature. This type of behaviour is dynamic and implements the decision model.

In the remainder of this section, we take a closer look at the DMM. In the initial phase of the DM procedure, the developer launches the GUI-based wizard depicted in Figure 9 and specifies the data source to be loaded and the *agent decision attributes* that will be represented as internal nodes of the extracted decision model. In subsequent steps, the developer selects the type of the DM technique from a set of available

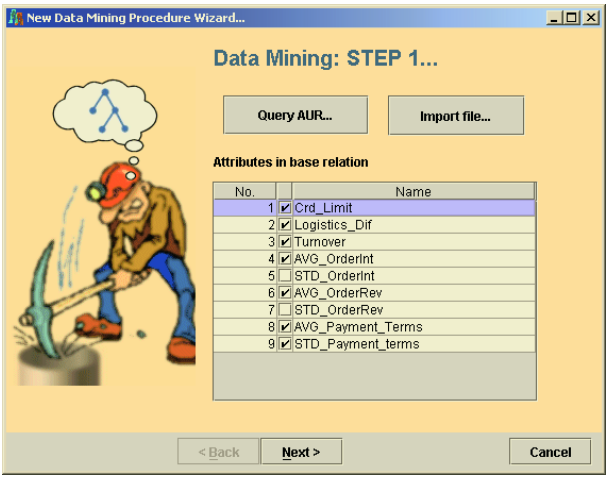

Fig. 9. The first step of the DMM wizard

options, which include classification, clustering, and association rule extraction. For each technique, a number of DM algorithms are at the user's disposal.

DMM was developed by incorporating a set of DM methods based on the WEKA [18] library and tools and by adding some new algorithms. We have also extended the WEKA API in order for it to support PMML. For further information on the developed DM algorithms and on the DMM functionality the interested reader may see [1, 21] or visit `http://www.source-forge.net/projects/DataMiner`.

The completion of the training process requires the translation of the DM resulted decision model into an agent-understandable format. This is performed by the ATM, which receives the PMML output as an ACL message sent by the DMM, as soon as the DM procedure is completed, and activates the rule engine. Actually, the ATM converts the PMML document into JESS rules and communicates, via appropriate messages, with the "trainee" agent, in order to insert the new decision model into it. After the completion of this process, our framework automatically generates Java source code and instantiates the new "trained" agent into the predefined MAS. The total configuration of the new agent is stored in the development framework, enabling future modifications of the training parameters, or even the retraining of the already "trained" agents.

4 Applications

The interesting, non-trivial, implicit and potentially useful knowledge extracted by the use of DM would be expected to find fast application on the development and realization of agent intelligence. The incorporation of knowledge based on previous observations may considerably improve agent infrastructures while also increasing reusability and minimizing customization costs.

In order to demonstrate the feasibility of our approach, we have developed several agent-based applications that cover all three cases of knowledge diffusion [1]. Using

AA we have built an agent-based recommendation engine situated on top of an operating ERP system and capable of tapping the wealth of data stored in the ERP's databases [19]. Such an approach can combine the decision support capabilities of more traditional approaches for supply chain management (SCM), customer relationship management (CRM), and supplier relationship management (SRM) [20]. This was clearly a system that implements Case 1 of knowledge diffusion, since data mining is performed on historical application data.

We have also applied the methodology for the development of an environmental monitoring system which falls into the class of applications called Environmental Monitoring Information Systems (EMIS). The primary function of EMIS is the continuous monitoring of several environmental indicators in an effort to produce sound and validated information. The architectures and functionalities of EMIS vary from naive configurations, focused on data collection and projection, to elaborate decision-support frameworks dealing with phenomena surveillance, data storage and manipulation, knowledge discovery and diffusion to the end users (government agencies, non-governmental organizations, and citizens). The presented synergy of AT and DM can provide to EMIS efficient solutions for the monitoring, management and distribution of environmental changes, while eliminating the time-overhead that often exists between data producers and data consumers. The MAS that we developed is situated upon a sensor network that monitors a number of air quality indicators and pollutants [22]. The MAS interweaves multiple data streams, validates and stores the information, decides whether an alarm must be issued, and, in the latter case, notifies the interested parties.

To demonstrate Case 2 of knowledge diffusion, we have implemented a MAS, which addresses the problem of predicting the future behaviour of agents based on their past actions/decisions. With this system we showed how data mining, performed on agent behaviour datasets, can yield usable behaviour profiles. We have introduced ! -profile, a DM process to produce recommendations based on aggregate action profiles [23]. The system in this case is a web navigation engine, which tracks user actions in corporate sites and suggests possibly interesting sites. This framework can be extended to cover a large variety of web services and/or intranet applications.

Another interesting area that is classified as Case 2, involves the improvement of the efficiency of agents in e-commerce and, more specifically, agents participating in e-auctions [24]. The goal here is to create both rational and efficient agent behaviours, to enable reliable agent-mediated transactions. In fact, through the presented methodology the improvement of agent behaviours in auctions is feasible (see Figure 10). Data mining can be performed on available historical data describing the bidding flow and the results can be used to improve the bidding mechanism of agents for e-auctions. Appropriate analysis of the data produced as an auction progresses (historical data) can lead to more accurate short-term forecasting. By the use of trend analysis techniques, an agent can comprehend the bidding policy of its rivals, and, thus, readjust its own in order to yield higher profits for buyers. In addition, the number of negotiations between interested parties is reduced (m instead of n, where $m<n$), since accurate forecasting implies more efficient bidding.

Finally, we have developed Biotope [25] as a typical example of knowledge diffusion in agent communities and evolutionary systems (Case 3). Biotope is an agent

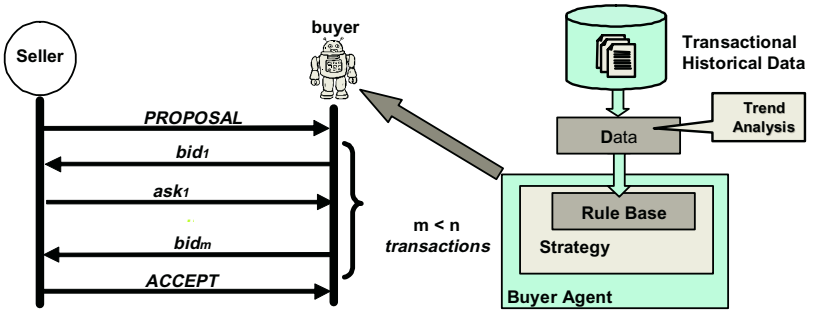

Fig. 10. Improving the behavior of agents participating in e-auctions

community used to simulate an ecosystem, where agents, representing living organisms, live, explore, feed, multiply, and eventually die in an environment with varying degrees of uncertainty. Genetic algorithms and agent communication primitives are exploited to implement knowledge transfer, which is essential for the survival of the community.

Multi-agent systems can also be exploited for the improvement of the software development process [1]. Software houses often develop frameworks for building end-user applications, following a standard methodology. Such frameworks interconnect different software modules and combine different software processes into workflows, in order to produce the desired outcome. Apart from the scheduling and planning, where agents collaborate and negotiate to reach the optimal solution, data mining techniques can be applied on workflow graphs, in order to discover correlations between certain workflow traverses. In addition, DM techniques can be applied to solve any kind of application-level problems, i.e., decision making capabilities.

5 Summary and Conclusions

In this paper we have presented the concept of combining the two, otherwise diverse areas, of data mining and agent technology. Limitations related to the nature of different types of logic adopted by DM and AT (inductive and deductive, respectively), hinder the unflustered application of knowledge to agent reasoning. We argue that our methodology can overcome these limitations and make the coupling of DM and AT feasible. Nevertheless, for a fruitful coupling, either historical data (application or agent-behaviour) must be available, or the knowledge mechanisms of agents must allow self-organization.

Agent Academy is a multi-agent development framework for constructing MAS, or single agents. AA combines a GUI-based, high-level MAS authoring tool with a facility for extracting rule-based reasoning from available data and inserting it into agents. The produced knowledge is expressed as PMML formatted documents. We have presented the functional architecture of our framework and outlined the agent training process.

Through our experience with Agent Academy, we are convinced that this development environment significantly reduces the programming effort for building agent

applications, both in terms of time and code efficiency, especially for those MAS developers who use JADE. For instance, one MAS, that requires the writing of almost 6,000 lines of Java code, using JADE, requires less than one hour to be developed with Agent Academy. This test indicates that AA meets the requirement for making agent programs in a quicker and easier manner. On the other hand, our experiments with the DMM have shown that the completion of the decision model generated for agent reasoning is highly dependant on the amount of available data. In particular, a dataset of more than 10,000 records is adequate enough for producing high-confidence DM results, while datasets with fewer than 3,000 records have yielded non-consistent arbitrary output.

Acknowledgements

Work presented in this paper was partially funded by the European Commission, under the IST initiative as a research and development project (contract number IST-2000-31050, "Agent Academy: A Data Mining Framework for Training Intelligent Agents"). The author would like to thank all members of the Agent Academy consortium for their remarkable efforts in the development of such a large project.

References

1. Symeonidis, A., Mitkas, P.A.: Agent Intelligence Through Data Mining, Springer, (2005)
2. Lind J.: Issues in Agent-Oriented Software Engineering. In: First International Workshop on Agent-Oriented Software Engineering (AOSE-2000), Limerick, Ireland (2000)
3. Galitsky, B. and Pampapathi, R.: Deductive and inductive reasoning for processing the claims of unsatisfied customers. Proc. of the 16th IEA/AIE Conference. Springer-Verlag. (2003) 21–30
4. Fernandes, A. A. A.: Combining inductive and deductive inference in knowledge management tasks. Proc. of the 11th International Workshop on Database and Expert Systems Applications. IEEE Computer Society (2000) 1109–1114
5. Kero, B., Russell, L., Tsur, S., Shen, W. M.: An overview of data mining technologies. Proc. of the KDD Workshop in the 4th International Conference on Deductive and Object-Oriented Databases (1995)
6. Agent Academy: http://www.source-forge.net/projects/AgentAcademy and http://agentacademy.iti.gr/
7. Mitkas, P. A., Kehagias, D., Symeonidis, A. L., Athanasiadis, I.: A framework for constructing multi-agent applications and training intelligent agents. Proc. of the 4th International Workshop on Agent-Oriented Software Engineering. Springer-Verlag. (2003) 1–16
8. Bellifemine F., Poggi A., Rimassa G., Turci P.: An Object-Oriented Framework to realize Agent Systems. In: Proceedings of WOA 2000 Workshop, Parma, Italy (2000) 52–57
9. Foundation for Intelligent Physical Agents, the: FIPA Developer's Guide (2001) available at: http://www.fipa.org/specs/fipa00021/
10. Bellifemine F., Caire G., Trucco T., Rimassa G.: JADE Programmer's Guide. (2001) available at: http://sharon.cselt.it/
11. Wooldridge, M, Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. In: Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3, No. 3. (2000) 285–312

12. Foundation for Intelligent Physical Agents, the: FIPA Communicative Act Library Specification. (2001) available at: http://www.fipa.org/specs/fipa00037/
13. Foundation for Intelligent Physical Agents, the: FIPA SL Content Language Specification (2002) available at: http://www.fipa.org/specs/fipa00008/
14. Foundation for Intelligent Physical Agents, the: FIPA ACL Message Structure Specification (2002) available at http://www.fipa.org/specs/fipa00037/
15. Noy, N.F., Sintek, M., Decker S., Crubezy, M., Fergerson, R.W., & Musen, M.A.: Creating Semantic Web Contents with Protégé-2000. In: IEEE Intelligent Systems 16 (2): (2001) 60–71
16. Data Mining Group, the: Predictive Model Markup Language Specifications (PMML), ver. 2.0 available at: http://www.dmg.org
17. Java Expert System Shell (JESS): http://herzberg.ca.sandia.gov/jess/
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann publishers, San Francisco, CA (2000)
19. Symeonidis, A. L., Kehagias, D., Mitkas, P. A.: Intelligent policy recommendations on enterprise resource planning by the use of agent technology and data mining techniques. Expert Systems with Applications, 25 (2003) 589–602
20. Symeonidis A.L, Kehagias D., Koumpis A., Vontas A.: Open Source Supply Chain. In: 10th International Conference on Concurrent Engineering (CE-2003), Workshop on intelligent agents and data mining: research and applications, Madeira, Portugal (2003)
21. Athanasiadis, I.N., Kaburlasos, V.G., Mitkas, P.A., and Petridis, V.: Applying Machine Learning Techniques on Air Quality Data for Real-Time Decision Support. In: First International NAISO Symposium on Information Technologies in Environmental Engineering (ITEE'2003), Gdansk, Poland (2003)
22. Athanasiadis, I.N., Mitkas, P.A.: An agent-based intelligent environmental monitoring system. Management of Environmental Quality. 15. (2004) 229–237
23. Symeonidis, A.L., Mitkas, P.A.: A methodology for predicting agent behaviour by the use of data mining techniques. Proc. of AIS-ADM '05 workshop. St. Petersberg, Russia. (2005)
24. Kehagias, D., Symeonidis, A.L., Mitkas, P.A.: Designing pricing mechanisms for autonomous agents based on bid-forecasting. Journal of Electronic Markets, 15. (2005)
25. Symeonidis, A. L., Seroglou, S., Valtos, E., Mitkas, P. A.: Biotope: An Integrated Simulation Tool for Multiagent Communities Residing in Hostile Environments. To appear in IEEE Transactions on Systems, Man, and Cybernetics (2005)