Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 36 (2009) 2433–2450

Expert Systems with Applications

www.elsevier.com/locate/eswa

VODKA: Variant objects discovering knowledge acquisition

Shian-Shyong Tseng [a,b,*], Shun-Chieh Lin [a]

[a] Department of Computer and Information Science, National Chiao Tung University, No. 1001, Ta Hsueh Road, Hsinchu 300, Taiwan, ROC
[b] Department of Information Science and Applications, Asia University, No. 500, Lioufong Road, Wufong Shiang, Taichung 413, Taiwan, ROC

Abstract

Many knowledge acquisition methodologies have been proposed to elicit rules systematically with embedded meaning from domain experts. But, none of these methods discusses the issue of discovering new modified objects in a traditional classification knowledge based system. For experts to sense the occurrence of new variants and revise the original rule base, to collect sufficient relevant information becomes increasingly important in the knowledge acquisition field. In this paper, the method variant objects discovering knowledge acquisition (VODKA) we proposed includes three stages (log collecting stage, knowledge learning stage, and knowledge polishing stage) to facilitate the acquisition of new inference rules for a classification knowledge based system. The originality of VODKA is to identify these new modified objects, the variants, from the way that the existing knowledge based system fails in applying some rules with low certainty degree. In this method, we try to classify the current new evolving object identified according to its attributes and their corresponding values. According to the analysis of the collected inference logs, one of the three recommendations (including adding a new attribute-value of an attribute, modifying the data type of an attribute, or adding a new attribute) will be suggested to help experts observe and characterize the new confirmed variants. VODKA requires E-EMCUD (extended embedded meaning capturing and uncertainty deciding). EMCUD is a knowledge acquisition system which relies on the repertory grids technique to manage object/attribute-values tables and to produce inferences rules from these tables. The E-EMCUD we used here is a new version of EMCUD to update existing tables by adding new objects or new attributes and to adapt the original embedded rules. Here, a computer worm detection prototype is implemented to evaluate the effectiveness of VODKA. The experimental results show that new worm variants could be discovered from inference logs to customize the corresponding detection rules for computer worms. Moreover, VODKA can be applied to the e-learning area to learn the variant learning behaviors of students and to reconstruct the teaching materials in improving the performance of e-learners.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Knowledge acquisition; Variant discovering; EMCUD; VODKA; Computer worm

1. Introduction

As time goes on, new objects in many domains are incrementally evolving due to the environment changes and the knowledge explosion. Besides unchangeable objects in the course of time, some previously created objects may become obsolete and disappear after a long period of time, and some new objects might be added or modified for adapting to the dynamic environmental changes with the time, which is the phenomenon of object evolution. In this paper, knowledge can be classified as static knowledge and dynamic knowledge according to the stability of knowledge in a dynamic environment. Static knowledge remains the same while dynamic knowledge will be updated when environment changes. For example, due to the nature of evolution, some well-known virus could evolve into a new virus which, suggested by the experts, should be singled out as a new variant to classify all kinds of viruses definitely.

[*] Corresponding author. Address: Department of Computer and Information Science, National Chiao Tung University, No. 1001, Ta Hsueh Road, Hsinchu 300, Taiwan, ROC. Tel.: +886 3 5712121x56658; fax: +886 3 5721490.
E-mail addresses: sstseng@cis.nctu.edu.tw (S.-S. Tseng), jielin@cis.nctu.edu.tw (S.-C. Lin).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.12.055

As we know, a knowledge based system (KBS) is an intelligent computer program using knowledge and inference procedures to solve problems which are difficult enough to require significant human expertise for the solutions, such as disease diagnosis, investment prediction, or computer science problems. Since the phase of knowledge acquisition is a bottleneck when constructing a KBS, many methodologies and related tools, e.g., NeoETS (Boose & Bradshaw, 1986), AQUINAS (Boose & Bradshaw, 1987), KITTEN (Shaw & Gaines, 1987), EMCUD (Hwang & Tseng, 1990), KADS (Wielinga, Schreiber, & Breuker, 1992), KANAL (Kim & Gil, 2001), OMCS-2 (Singh et al., 2002), INDUS (Caragea et al., 2005) have been proposed to rapidly build prototypes in the past twenty years.

Most of the existing knowledge acquisition systems employ the repertory grid test, originally developed by personal construct theory (PCT) for eliciting static knowledge, to identify different objects and distinguishing these objects in a selected domain. PCT can help domain experts formulate the quality of static knowledge with/without knowledge engineers. But only the static knowledge used for classifying and distinguishing well-known objects by a finite set of training cases in most the systems, such as ID3 (Quinlan, 1986), KAISER (Tsujino, Takegaki, & Nishida, 1990; Tsujino, Dabija, & Nishida, 1992), C-KAT (Zacklad & Fontaine, 1995), version space (Hong & Tseng, 1997), AVT-DTL (Zhang & Honavar, 2003). And few of them attack the topic of acquiring dynamic knowledge, in fact, the incremental learning of dynamic knowledge by continuously being aware of the variant objects as time goes on is important

Consequently, the lack of sufficient information about variants may result in a problem of observing the occurrence of evolving objects for human experts. Now, how to collect sufficient relevant information to help experts sense the occurrence of variants and reuse the original rule base becomes one of the important issues.

Embedded meaning capturing and uncertainty deciding (EMCUD), one of a repertory grid based knowledge acquisition tools, was proposed to elicit the embedded meanings of knowledge (embedded rules bearing on objects and object attributes) to classify objects. Based upon an attribute ordering table (AOT) to record the relative importance of each attribute to each object, EMCUD guides the experts to determine the certainty degree of each embedded rule, in order to extend the coverage of original rules generated by traditional repertory grid.

With different certainty degree, some variant objects modified from original objects may not be classified by the original rules but may be classified by the other embedded rules; even these embedded rules may have margin values of certainty factor (CF) due to the weak suggestions of domain experts. Since the higher the CF value, the more reliable the results are, this kind of objects could be singled out as a variant object class because the new characteristics of these objects are emerging. Therefore, if the variants could be detected and recommended as the new objects by experts, the related ambiguous attributes (minor attributes) could be refined or new attributes could be added to improve the classification ability in a KBS. These attributes might result in the marginally acceptable CF values of weak embedded rules.

In this paper, a new iterative methodology, variant objects discovering knowledge acquisition (VODKA) is proposed to learn new variant objects based upon monitoring the inference behaviors of weak embedded rules. We focus our attention on incrementally collecting new cases of modified objects and learning the dynamic knowledge to modify the original classification KBS. In order to clearly single a new variant object out, the previous methodologies need to be applied recurrently, and therefore the originally elicited knowledge might not be easily reused. The goal of VODKA is to facilitate the acquisition of new inference rules of the new modified objects for a classification KBS, from which new modified objects will be identified from the attribute values. The new rules, generated by VODKA, will be able to cope with the new variant objects. They are similar to previously known embedded rules in the KBS.

Since the relative importance of each attribute can be represented by using AOT, some minor attributes will be relaxed or ignored in order to capture the embedded meanings of embedded rules with acceptable CF values in EMCUD. In other words, these kinds of attributes are not used to classify the object. New variants classified by weak embedded rules should be singled out of original objects. Obtained from the collection of minor attribute-value pairs, the fired frequency of weak embedded rules is a big help for the experts to discover new information of variant objects. These attribute-value pairs are then to be provided to help experts determine whether to single variant objects out or not.

According to the analysis of the collected inference logs, one of the three different recommendations: adding a new attribute-value of a minor attribute, modifying the data type of a minor attribute, or adding a new attribute, will be suggested to help experts characterize the corresponding rules to refine the KBS if a new variant object is confirmed to be singled out.

To evaluate the performance of VODKA, a simple computer worm detection prototype system with VODKA and worm classification embedded rule base based upon DRAMA (Lin, Tseng, & Tsai, 2003) has been implemented to discover the new variant worms generated by an attacking traffic generator. Experts can be helped by VODKA to quickly and easily single out the new variant objects and customize the detection rules for computer worms. Also, VODKA has been applied on an e-learning domain to learn the variant learning behaviors of e-learners, which shows that VODKA is a big help for teachers to prepare or to reconstruct the teaching materials, the variant course materials, for these e-learners to easily understand the learning materials. In a word, VODKA can enhance the classification ability for new objects of a KBS since the new evolving objects can be incrementally learned and discovered by collecting the sufficient information.

2. Related work

Several knowledge acquisition methodologies and related systems are introduced in this section first. Then repertory grid, one of the popular indirect knowledge acquisition techniques, is also discussed. Finally, the elicitation of embedded meaning and some problems of traditional knowledge acquisition methodologies are discussed.

2.1. Knowledge acquisition systems

As we know, knowledge in many domains and the experience of domain experts is continuously growing. Many knowledge acquisition methodologies have been proposed to help knowledge engineers acquire the useful knowledge and then to transfer this knowledge into a knowledge base or other computerized representation forms. In general, there are three approaches for knowledge acquisition (Crowther & Hartnett, 1996; Hwang & Tseng, 1990; Mcgraw & Harbison-Briggs, 1989):

(1) Interviewing experts by experienced knowledge engineers: formalizing the elicited knowledge after interviewing experts. However, it is usually time-consuming if the communication between domain experts and knowledge engineers is insufficient.
(2) Machine learning: learning the knowledge by collecting many useful cases and instances with/without the involvement of domain experts. However, the quality of the results usually relies on the selected training cases.
(3) Knowledge acquisition systems: assisting domain experts in generating useful rules using knowledge acquisition systems with/without the help of knowledge engineers. These tools could reduce the effort of communication between knowledge engineers and domain experts and could reduce the risk and difficulty of selecting the suitable training cases.

In the past decades, many knowledge acquisition systems, e.g., NeoETS (Boose & Bradshaw, 1986), AQUINAS (Boose & Bradshaw, 1987), KITTEN (Shaw & Gaines, 1987), EMCUD (Hwang & Tseng, 1990), KADS (Wielinga et al., 1992), MCRDR (Kang, 1996), KAMET (Cairo, 1998), MedFrame/CADIAG-IV (Boegl, 1997; Kolousek, 1997; Leitich et al., 2001), KANAL (Kim & Gil, 2001), OMCS-2 (Singh et al., 2002), INDUS (Caragea et al., 2005) have been developed to build prototypes and to iteratively elicit the knowledge from domain experts. However, most of them can not be used to construct the dynamic knowledge to classify the variant objects in a dynamic environment using the previous obtained attributes. For acquiring the dynamic knowledge, the traditional approach needs to rerun their knowledge acquisition process.

Ontology, which defines concepts and the relationships between concepts, was indicated to be useful in a knowledge acquisition process in recent years. In computer science area, the ontology is a conceptualized data structure to be used in knowledge systems. Based on the same ontology, different systems can communicate with each other, or the knowledge inside computer systems may be structured and presented more accurately. "An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms" (Uschold, King, Moralee, & Zorgios, 1998).

To build up the domain ontology, identifying the domain concepts is the first step. Moreover, a systematical approach to acquire the relations of concepts from expertise is also a key factor for ontology construction. In recent years, due to the increasing requirement for inducing domain knowledge into computer systems (Kitamura & Mizoguchi, 2003; Noy & Sintek, 2001), many researches (Alani & Kim, 2003; Eriksson, 2003; Fensel & Angele, 1988; Frank & Farquhar, 1999; Leibbink, Witteman, & Mayer, 2002; Maedche & Staab, 2001) were proposed to discover, represent, and use of ontologies. Especially in KBSs, ontology becomes a key factor to build a successful knowledge base with more meaningful knowledge content for users. Moreover, the software agents technique is used in knowledge acquisition systems to acquire knowledge from autonomous, distributed, and semantically heterogeneous data sources with a domain specific ontology (Caragea, Silvescu, & Honavar, 2001; Caragea et al., 2005; Rosaci, 2005; Rosaci, Terracina, & Ursino, 2004).

However, ontology used in these researches was usually assumed to be existent or be able to be obtained directly from experts. However, in real cases, when a knowledge engineer starts at designing a knowledge acquisition process, the ontology of the domain concepts is usually not available yet. It is usually a time-consuming process to build an ontology only by interviewing a domain expert.

2.2. Repertory grid methodology and relevant systems

Repertory grid, based on Kelly's personal construct theory (Kelly, 1955) which reports how people make sense of the world, could be used as an efficient knowledge acquisition technique in identifying different objects and distinguishing these objects in a domain. It is the basis of several computer assisted knowledge acquisition tools, such as ETS (Boose, 1984, 1985), AQUINAS (Boose & Bradshaw, 1987) and KSSO (Gaines, 1987).

A single repertory grid represented as a matrix whose columns have element objects (labels) and whose rows have construct's attributes (labels) can classify a class of objects, or individuals. The value assigned to an element-construct pair need not be Boolean. Grid values have numeric ratings, probabilities, and other characteristics, where each value reflects a degree. Then, the expert is asked to fill the grid with 5-scale ratings, where "1" represents the most relevant attribute to the object; "2" represents that the attribute may be relevant to the object; "3" represents "unknown" or "no relevance"; "4" represents that the object may have the opposite characteristic; "5" represents the most relevant opposite characteristic to the object. The whole concept of repertory grid technique can be described with the following steps:

(1) Elicit all of the element objects, e.g., E_1, E_2, E_3, E_4, E_5 from the expert.
(2) Elicit the construct attributes (and their opposites), e.g., $C_1, C_2, C_3, C_4(C_1', C_2', C_3', C_4')$, from the expert. Each time three elements are chosen to ask for a construct to distinguish one element from the other two.
(3) Rate all of the [element, construct] entries of the grid with value range from 1 to 5. An illustrative example is given in Table 1.

As repertory grid technique has been widely used by researchers, some extensions have been made to enrich its representative ability for covering more knowledge, the value assigned to an element-construct pair may be Boolean, numeric ratings, probabilities, etc. For example, Dixit and Pindyck (1994), and Hwang (1995) extended the repertory grid technique to the fuzzy table, in which constructs were fuzzy attributes that could be rated by means of fuzzy linguistic terms from a finite set. Castro-Schez, Jennings, Luo, and Shadbolt (2004) developed a technique using a fuzzy repertory grid for acquiring the finite set of attributes or variables that the expert uses in a classification problem to characterize and discriminate a set of elements.

Moreover, several models have been proposed for handling uncertainties in expert systems through generating more meaningful rules from the repertory grid oriented approaches. EMYCIN certainty factor (CF) model was first used to determine the degree of the belief of a rule for uncertain reasoning (Shortliffe & Buchanan, 1975). Embedded meaning capturing and uncertainty deciding (EMCUD) knowledge acquisition system was proposed to extract rules with embedded meaning from hierarchical repertory girds by defining the impacts of the constructs of each element (Hwang & Tseng, 1990) and was successfully applied in a medical diagnostic system of acute exanthema in Taiwan (Hwang & Tseng, 1991). WebGrid (Shaw & Gaines, 1996), Calgary's web-based knowledge modeling and inference tool, is based on repertory grid elicitation and analysis. Blythe, Kim, Ramachandran, and Gil (2001) proposed an acquisition interface that integrates previously developed techniques to guide users to set constraints in different aspects of knowledge acquisition.

Table 1
The illustrative example of a repertory grid with ratings

Element construct	E_1	E_2	E_3	E_4	E_5	
C_1	5	1	5	1	1	C_1'
C_2	4	4	4	1	4	C_2'
C_3	1	4	5	1	4	C_3'
C_4	1	4	4	5	5	C_4'

Although these methodologies are proposed to extend the ability of uncertain reasoning to classify the well-known objects, none of them discusses the issue of discovering and classifying new variant objects. It is also difficult for experts to sense the occurrence of variant knowledge, which is modified in the dynamic environment as time goes on. Therefore, a new knowledge acquisition system based upon EMCUD is proposed in this paper to guide domain experts to create additional attributes for classifying new variant objects through the observations of the interested inference results.

2.3. Elicitation of embedded meanings

The embedded meanings referred to here represent the information that domain experts take for granted but which is implicit to the people who are not familiar with the application domain. The lack of embedded meaning will probably make an expert system fail to infer some cases being trivial to experts. SEEK (Politakis & Weiss, 1984) and SEEK2 (Ginsberg, Weiss, & Politakis, 1988) have been proposed to obtain embedded meanings by some efficient refinement processes. However, the major problem of SEEK and SEEK2 is that the case database is assumed to be available although it is difficult to collect sufficient cases in some applications. Moreover, it would be also time-consuming and boring for experts to offer a conclusion for each case in the database before starting the refinement procedure. Thus, EMCUD (Hwang & Tseng, 1990) is proposed to elicit the embedded meanings of knowledge from the existing hierarchical repertory grids given by experts. Additionally, it will also guide experts to determine the certainty degree of each rule with embedded meaning for extending the coverage of generated original rules.

To capture the embedded meanings of the resulting grids, the Attribute Ordering Table (AOT), which is used to record the relative importance of each attribute to each object, is employed. The values in each AOT entry, a pair of attribute and object, may be labeled "X", "D" or an integer number. "X" means no relationship existing between the attribute and the object. "D" means that the attribute dominates the object, i.e., if the attribute is not equal to the entry value, it is impossible for the object to be implied. Integer numbers are used to represent the relative importance degree of the attribute of the object instead of dominating the corresponding object. If the attribute does not equal the attribute-value, it is still for the object to be implied. A larger integer number implies that attributes must be more important to the object.

Using AOT, the original rules generate some rules with embedded meaning, and the CF value of each rule, which is between -1 and 1, could be determined to indicate the degree of supporting the inference result. The higher the CF value, the more reliable the results are. The EMCUD algorithm is listed as follows.

Algorithm 1. EMCUD algorithm

Input: The hierarchical grids.
Output: The guiding rules with embedded meaning.
Step 1: Build the corresponding AOT with each grid of the hierarchical multiple grids.
Step 2: Generate the possible rules with embedded meaning.
Step 3: Select the accepted rules with embedded meaning through the interaction with experts.
Step 4: Generate automatically the CF of each rule with embedded meaning.

All rules generated by EMCUD can be categorized into two classes: original and embedded rules with acceptable CF value, and discarded rules with unacceptable CF value, according to the confidence degree of domain experts. To determine the CF value of each embedded rule, we have to firstly determine the upper and the lower bounds of CF values of accepted embedded rules. CF value of each rule can be automatically determined by a fuzzy mapping function. Thus, the useful embedded rules with acceptable CF values could be used to cover more uncertainty cases.

Since embedded rules with weak acceptable CF values (the CF values below a user defined threshold) usually mean that domain experts might lack strong confidence, objects matching weak embedded rules derived from original objects may be the candidates for new variants. For example, the object satisfying the conditions (attribute-value pairs) of the embedded rules with $CF = 0.5$ means the expert might suggest that it would be marginally classified into the object class and the minor attributes of the embedded rule might be not clearly defined. Therefore, the fired frequencies of this kind of weak embedded rules should be used to discover the occurrence of new variant objects.

2.4. Problems with conventional knowledge acquisition methods

With the changing environment, the adaptation of the acquired rules should be required to cope with new variants. However, experts may not be aware of the occurrence of new variant candidates and may have insufficient evidence to construct the knowledge of the variants using conventional repertory grid approaches. Although EMCUD could be used to generate more useful embedded rules for covering more similar objects in extended object classes, it still lacks the ability of grid evolution for singling these new variants out; e.g., EMCUD should manually regenerate the original and embedded rules to classify these variant objects by interacting with domain experts after collecting sufficient information about these variants. Therefore, enhancing the adaptation ability of embedded rules becomes increasingly important to achieve the ability of grid evolution in a classification KBS.

In this paper, the embedded rules from EMCUD are categorized into three classes: the original rules with strong CF values, the embedded rules with marginally acceptable CF values, and the discarded rules with low CF values. Hence, a new knowledge acquisition methodology is proposed to discover the occurrence of new variant objects using the fired frequency of embedded rules with marginally acceptable CF values. A simple computer worm detection prototype in Example 1 is used to illustrate the inability for discovering variants using EMCUD.

Example 1. The example of classifying four computer worms.

In recent years, the number of computer worms is dramatically increasing to threaten the reliability of Internet. Table 2 shows the acquisition table of four computer worms (Kienzle & Elder, 2003; Mirkovic, Martin, & Reiher, 2002; Moore, Shannon, Voelker, & Savage, 2003; Weaver, Paxson, Staniford, Cunningham, & Maulik, 2003) including Nimda, CodeRed, Blaster, and Welchia using five attributes including 100-thread, System reboot, DoS type, Email attached file, and TCP port. The 100-thread means 100-threads with Boolean are simultaneously executed by one program. The system reboot Boolean attribute will be set to True if the system has been automatically rebooted. The attacking methodologies of worms could be classified into one kind of DoS type with String attribute (Mirkovic et al., 2002). The email attached file attribute with Set data type is also a useful attribute to classify these worms. Most of the worms could communicate with each other using different TCP port with Set data type.

An example of constructing an AOT table from the acquisition table shown in Table 2 is given as follows:

EMCUD: If DoS type is not equal to Email flood, is it possible for Nimda to be implied?
EXPERT: No.

Table 2
The acquisition table of four computer worms

Attribute	Object			
	Nimda	CodeRed	Blaster	Welchia
100-thread (A_1)	X	True	X	X
System reboot (A_2)	X	True	True	True
DoS type (A_3)	Email flood	TCP flood	Windows Update flood	ICMP flood
Email attached file (A_4)	{sample.exe; puta!!scr}	X	X	X
TCP port (A_5)	X	{80}	{135;4444}	{80;135}

The answer means the DoS type dominate Nimda, and hence AOT [Nimda, DoS type] = “D”.

EMCUD: If Email attached file is not equal to any element of {sample.exe, puta!!scr}, is it possible for Nimda to be implied?
EXPERT: YES.

The answer means that Email attached file does not dominate Nimda. The questions for 100-thread and Nimda will not be asked, since the entry [Nimda, 100-thread] is labeled “X”. Therefore, the entry AOT [Nimda, 100-thread] is labeled “X”, too. This is the same for AOT [Nimda, System reboot] and AOT [Nimda, TCP port] entries. The entry AOT [Nimda, Email attached file] is set to be 1, since the Email attached file is the only attribute that does not dominate Nimda. If there are more than one attributes do not dominate the object, e.g., the System reboot, the DoS type, and the TCP port do not dominate Blaster, the following questions will be asked by EMCUD.

(1) Is System reboot more important than DoS type?
(2) Is System reboot less important than DoS type?
(3) Is System reboot as important as DoS type?

The expert indicates that System reboot is as important as DoS type to Blaster. Moreover, the expert also indicates that System reboot is more important than TCP port to Blaster; and hence the entries AOT [Blaster, System reboot] = AOT [Blaster, DoS type] = 2 and AOT [Blaster, TCP port] = 1. After each entry value of AOT is determined as shown in Table 3, the embedded meaning implied by the AOT can be extracted.

Now we use the first column of Table 2 to show the information implied by an AOT. The column expresses the following meanings:

(1) A_3 dominates Nimda: If A_3 is not equal to Email flood, it is impossible for Nimda to be implied.
(2) A_4 does not dominate Nimda: If A_4 is neither equal to sample.exe nor puta!!scr, it is still possible for Nimda to be implied.

In practice, the hierarchy rules can be generated while hierarchical grids are given. To simplify the discussion, Table 4 shows partial detection rules (simple rules) of a classification KBS based upon the Tables 2 and 3 to classify these worms using five attributes in single grids. $R_{i,j}$ represents the jth highest rank of CF in object i, and the highest rank is 0. The $R_{1,0}$ is the original rule of Nimda to classify the original Nimda objects and $R_{1,1}$ is the embedded rule of Nimda to classify the extended Nimda objects.

The Mask Table of minor attributes shown in Table 5 indicates the minor attributes for all embedded rules. Each row in Mask Table is a bit vector of attributes, where the ith bit is set to 1 representing the ith minor attribute that is negated or ignored. For example, the $M_{2,3}$ $(0,1,1,0,0)$ means the 2nd and 3rd minor attributes in $R_{2,3}$ are ignored.

On the Internet, each worm can be represented as a set of attribute-value pairs. We can automatically collect such attribute-value pairs and feed them into the classification KBS to classify them in the suitable category. Since new worms might have been derived from old worms, the difference between their attribute values seems to be slight. As mentioned above, EMCUD could generate lots of

Table 3
The AOT table of four computer worms

Attributes	Object			
	Nimda	CodeRed	Blaster	Welchia
A_1	X	2	X	X
A_2	X	1	2	2
A_3	D	1	2	1
A_4	1	X	X	X
A_5	X	X	1	2

Table 4
Partial detection rules generated by EMCUD

Rule #	Conditions					Conclusion	CF
	A_1	A_2	A_3	A_4	A_5	Object	
$R_{1,0}$	–	–	Email flood	(sample.exe; puta!!scr)	–	Nimda	0.8
$R_{1,1}$	–	–	Email flood	¬ (sample.exe; puta!!scr)	–	Nimda	0.4
$R_{2,0}$	True	True	TCP flood	–	–	CodeRed	0.8
$R_{2,1}$	True	False	TCP flood	–	–	CodeRed	0.6
$R_{2,2}$	False	True	TCP flood	–	–	CodeRed	0.4
$R_{2,3}$	True	False	¬ (TCP flood)	–	–	CodeRed	0.4
$R_{3,0}$	–	True	Windows update flood	–	{135; 4444}	Blaster	0.7
$R_{3,1}$	–	True	Windows update flood	–	¬{135; 4444}	Blaster	0.57
$R_{3,2}$	–	False	Windows update flood	–	{135; 4444}	Blaster	0.43
$R_{3,3}$	–	True	¬ (Windows update flood)	–	{135; 4444}	Blaster	0.43
$R_{3,4}$	–	False	Windows update flood	–	¬{135; 4444}	Blaster	0.3
$R_{4,0}$	–	True	ICMP flood	–	{80; 135}	Welchia	0.8
$R_{4,1}$	–	True	¬ (ICMP flood)	–	{80; 135}	Welchia	0.67
$R_{4,2}$	–	True	ICMP flood	–	¬{80; 135}	Welchia	0.53
$R_{4,3}$	–	True	¬ (ICMP flood)	–	¬{80; 135}	Welchia	0.4

Table 5
The Mask Table of ignored attributes

Mask #	A_1	A_2	A_3	A_4	A_5
$M_{1,0}$	0	0	0	0	0
$M_{1,1}$	0	0	0	1	0
$M_{2,0}$	0	0	0	0	0
$M_{2,1}$	0	1	0	0	0
$M_{2,2}$	1	0	0	0	0
$M_{2,3}$	0	1	1	0	0
$M_{3,0}$	0	0	0	0	0
$M_{3,1}$	0	0	0	0	1
$M_{3,2}$	0	1	0	0	0
$M_{3,3}$	0	0	1	0	0
$M_{3,4}$	0	1	0	0	1
$M_{4,0}$	0	0	0	0	0
$M_{4,1}$	0	1	0	0	0
$M_{4,2}$	0	0	0	0	1
$M_{4,3}$	0	1	0	0	1

embedded rules with different CF values for accommodating the knowledge of the changed worms due to the property of minor attributes; e.g., $R_{1,1}$ "IF (DoS type = E-mail flood) AND ¬ (Email attached file = (sample.exe; puta!!scr)) THEN Nimda", a marginally acceptable embedded rule with CF = 0.4, may be fired by a new Nimda variant which is treated as a member of original Nimda class. If this rule has been fired frequently due to a specific value of the attribute "email attached file": "readme.exe" (more evidence of the occurrence of the candidates of Nimda variants have been gathered), a new original rule and an embedded rule could be generated: "IF (DoS type = Email flood) AND (Email attached file = readme.exe) THEN Nimda.B", a subset of extended Nimda object class namely Nimda.B, with CF = 0.8 and "IF (DoS type = Email flood) AND ¬ (Email attached file = readme.exe) THEN Nimda.B" with CF = 0.5. These rules help to single the Nimda.B class out of the extended Nimda object class.

3. Knowledge acquisition by discovering variant objects

Although EMCUD and other similar approaches could be manually rerun to acquire variant knowledge from domain experts to classify new variant objects, it might be costly and hard to obtain the knowledge due to the insufficient information about variants. To simplify our discussion, let's assume some objects in O_1 class belong to the original object class (OO_1) of O_1, which can be classified by original rules of O_1. The other objects in O_1 class classified by embedded rules of O_1 belong to the extended object class (EO_1) of O_1, where $OO_1 \subset EO_1$. In the EO_1, some modified objects can be classified by the embedded rules of O_1 with weak CF values, which are singled out to be a variant object class (VO_1) of O_1 with the significant attributes emerged from minor attributes. That is, $VO_1 \subset EO_1$ and $VO_1 \cap O_i = \phi$, where $1 \leqslant i \leqslant m$, and m is the number of distinct object classes. Because the embedded rules with diverse CF values represent different supports to classify objects, the ones with marginally acceptable CF values might be triggered by some candidates of new variant classes. Therefore, our idea is to analyze the behaviors of weak embedded rules (the weak suggestions by experts) to construct the new variants acquisition table for extracting new variant knowledge.

In the following, we will propose a new iterative knowledge acquisition methodology, variant objects discovering

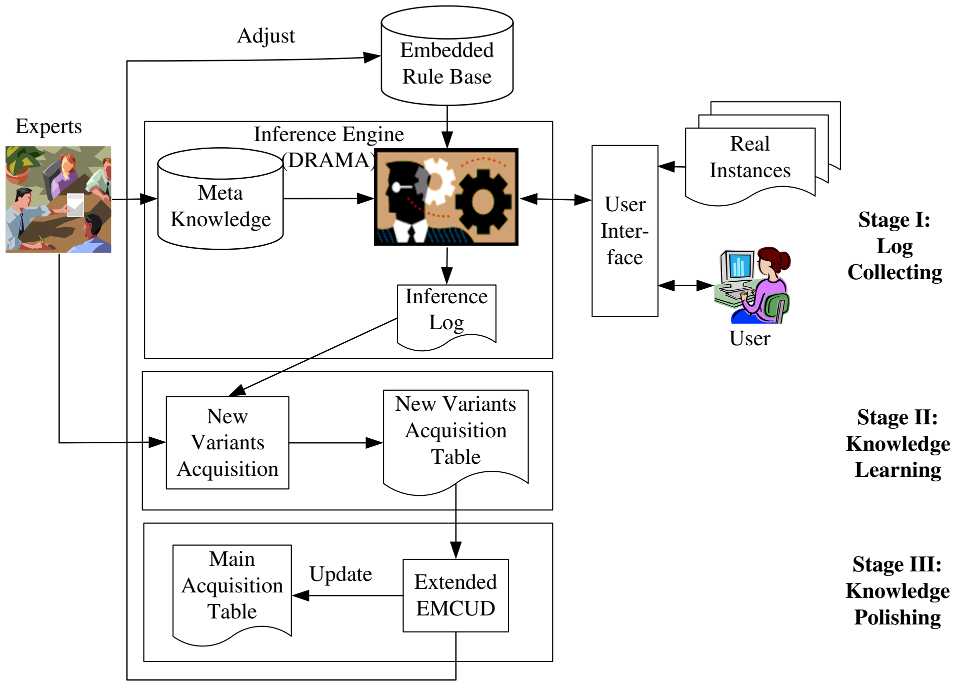

Fig. 1. The framework of VODKA.

knowledge acquisition (VODKA) which is an iterative process as shown in Fig. 1, to provide the ability of grid evolution, where each iteration consists of three stages, log collecting stage, knowledge learning stage, and knowledge polishing stage, to analyze the inference behaviors of the embedded rules in a KBS.

Firstly, the embedded rule base will be created according to the original main acquisition table using EMCUD or VODKA. Then the inference behaviors (facts/attribute-value pairs) will be collected iteratively to discover the candidates of the variants during Stage I according to the meta knowledge. The ignored attribute-value pair of the minor attribute will be treated as an item and a set of ignored attribute-value pairs will be treated as a transaction to discover the association between interesting attribute-value pairs in Stage II. Consequently, the new variants acquisition table based on the discovered knowledge could be generated by interacting with domain experts through the new variants acquisition procedure. Finally, the rules applicable to new variants will be incrementally generated and the main acquisition table will be iteratively adjusted using E-EMCUD in Stage III. The algorithm of VODKA is shown as follows.

Algorithm 2. The algorithm of VODKA

```
Input: The original main acquisition table $T$ and embedded rule base RB.
Output: The rules with embedded meaning about variants.
Stage I: Collect all facts of the weak embedded rules as real inference log of the RB.
Stage II: Generate the new variants acquisition table $T'$.
    Step 1: Discover large itemsets $L$ using the inference log.
    Step 2: Generate $T'$ using $L$ and additional attributes provided by experts.
Stage III: Use E-EMCUD to generate rules of new variants.
    Step 1: Generate rules according to $T'$.
    Step 2: Merge $T'$ into original main acquisition table $T$.
```

3.1. Meta knowledge in log collecting stage

Without loss of generality, let's assume that there are k attributes to classify m objects in the main acquisition table. Thus, the total number of embedded rules is limited. In order to assist domain experts in noticing and analyzing the occurrence of the candidates of variant objects, the following four meta rules are used to collect the inference log (fact/raw data) of weak embedded rules to help experts sense the occurrence of new variants.

MR_1: IF $R_{i,j}$ is fired THEN Increase $C_{i,j}$ by one.
MR_2: IF $CF(R_{i,j}) \leqslant TH_{CF}$, THEN Log $R_{i,j}$.
MR_3: IF $C_{i,j} \geqslant$ THcnt **AND** $CF(R_{i,j}) \leqslant TH_{CF}$ THEN Run **New Variants Acquisition Algorithm** to acquire the new variants acquisition table and Reset TimeOut.
MR_4: IF TimeOut = TH_{Period} THEN Run **New Variants Acquisition Algorithm** and Reset TimeOut.

The meta rule MR_1 is used to count the fired frequency of each embedded rule ($C_{i,j}$). The meta rule MR_2 means that all facts (attribute-value pairs) of the embedded rules with marginally acceptable CF lower than strong CF bound threshold (TH_{CF}) are logged as a record, ($R_{i,j}$, $A_1, A_2, \ldots, A_k$, $CF(R_{i,j})$). The meta rule MR_3 means that if there exists one weak embedded rule with fired frequency exceeding the warning line threshold (TH_{CNT}), new variants may be discovered iteratively using VODKA. The meta rule MR_4 means that VODKA will be executed periodically to refresh the new variants acquisition table. The TimeOut will be reset when MR_3 or MR_4 is triggered.

3.2. New variants discovering methodology at the knowledge learning stage

At the knowledge learning stage, the new variants acquisition table will be generated through interacting with domain experts based upon the observation of inference log. An ignored attribute-value pair, i.e., (DoS type = TCP flood), is treated as an item and the transaction is represented as a set of ignored attribute-value pairs, i.e., {(DoS type = TCP flood), (TCP port = {135;4444})}. The inference log could be automatically transformed into the transaction database (D) and the item set (I) using the Mask Table of ignored attributes. In order to obtain the candidates of new variants, we apply Apriori algorithm (Agrawal, Imielinksi, & Swami, 1993) to discover large itemsets (L) that will provide more useful information.

After generating the large itemsets, new variants acquisition table might be elicited following the new variants acquisition algorithm. The new objects using unclear attributes would be singled out accordingly, if the experts reconfirm the addition of the new variant object. Thus, one of three recommendations including adding a new attribute value to a minor attribute, modifying the data type of a minor attribute, adding a new attribute, will be further given to adjust the main acquisition table. If a new changing object is singled out, the new value of the minor attribute could be added to characterize of new objects. If the initial data type of a certain attribute is too rough to describe the object, a superior data type is recommended and the values of the attribute in both original object and variant should be modified. For example, the BOOLEAN data type may be refined to SINGLE VALUE data type (Hwang & Tseng, 1990). If changing the data type still can not discriminate the new variants from original objects, acquiring a new attribute from domain experts will be suggested in VODKA. Thus, the new variants acquisition table will be created iteratively using the discovered large itemsets in Algorithm 3. However, adding a new attribute, which is very time-consuming because it results in creating a new row in new variant acquisition table, is the last choice for classifying variant objects.

Algorithm 3. New variants acquisition algorithm

```
Input: Inference log and the main acquisition table $T$, the minimal support $\delta$.
Output: The new variant object class VO, new attribute set AN, and new variants acquisition table $T'$.
Step 1: Transform inference log into the transaction data set $D$.
Step 2: Discover large itemsets $L$ by $\delta$ using $D$.
Step 3: For each large itemset, ask experts to determine whether it belongs to a new variant or not.
Step 4: If a new variant is confirmed, ask experts to acquire the related information about this new variant.
    Store $\mathrm{VO}_{\mathrm{new}}$ in VO.
    Add a new column to represent the new variant $\mathrm{VO}_{\mathrm{new}}$ in $T'$.
    Ask experts to confirm whether changing the data type of attribute $A_i$ is needed or not, where
    $1 \leqslant i \leqslant k$, and $k$ is the number of attributes.

    Step 4.1: If no data type needs to be changed, Suggest Recommendation I.
        Add a new value of $A_i$ of the $\mathrm{VO}_{\mathrm{new}}$.
        Else, ask experts to confirm whether adding a new attribute is needed or not.
    Step 4.2: If no new attribute needs to be added, Suggest Recommendation II to modify the data type of $A_i$.
        Ask experts to acquire the mapping function of values between original and new data types.
        Add a column in $T'$ to represent the original object with new mapping values.
    Else, Suggest Recommendation III.
        Ask experts to acquire the values of the new attribute $A_{\mathrm{new}}$.
        Add a new row in $T'$ to represent the new variant $A_{\mathrm{new}}$.
        Store $A_{\mathrm{new}}$ in AN.
        Add a column in $T'$ to represent the original object with new attribute value if needed.
Step 5: Return VO, AN, $T'$.
```

Besides interaction with domain experts, the computational cost of the algorithm is dependent on Step 2. The size of collecting inference log and minimal support threshold setting will affect the computational cost. For different types of inference logs, different learning algorithms can be selected to learn and discover the candidate behaviors of variants.

Example 2. The variant learning example of a Blaster worm

In this example, let's assume that the fired sequence of some embedded rules of Blaster worms with marginally acceptable CF values is given in Table 6.

Let's assume minimal support is set to 30%; the large itemsets L including $L_1 = \{(A_2 = \text{False}); (A_5 = X)\}$ and $L_2 = \{(A_2 = \text{False}, A_5 = X)\}$ will be provided to experts for further recommendation; i.e., L will be used to generate the new variants acquisition table T' according to the recommendations suggested by VODKA.

VODKA: Does the attribute-value pair $(A_2 = \text{False})$ belong to any new variant object?
EXPERT: Yes. /*It means that a new variant contains the selected attribute-value pair $(A_2 = \text{False})$. Otherwise, the large itemset is discarded and another large itemset is chosen to be examined. */
VODKA: What is the name of the new variant object?
EXPERT: $\mathrm{VO}_{\mathrm{new}}$.

A new column will be added in T' to represent the variant, $\mathrm{VO}_{\mathrm{new}}$, separated from the original object.

VODKA: Is the data type of A_2 required to be changed?
EXPERT: No. /* It means the data type does not need to change after adding new variant (**Recommendation I**). Otherwise, VODKA will ask the following questions. */

The row representing A_2 with new attribute value and one column representing the variant object will be created in T'.

VODKA: Is any new attribute required to be added?

Table 6
The partial inference logs of Blaster

Rule #	A_1	A_2	A_3	A_4	A_5	Object	CF
$R_{3,4}$	17	False	Windows Update flood	–	X	Blaster	0.3
$R_{3,2}$	100	False	Windows Update flood	–	{135}	Blaster	0.43
$R_{3,4}$	8	False	Windows Update flood	–	X	Blaster	0.3
$R_{3,2}$	119	False	Windows Update flood	–	{4444}	Blaster	0.43
$R_{3,4}$	17	False	Windows Update flood	–	X	Blaster	0.3
$R_{3,2}$	100	False	Windows Update flood	–	{135}	Blaster	0.43
$R_{3,4}$	11	False	Windows Update flood	–	X	Blaster	0.3
$R_{3,2}$	76	False	Windows Update flood	–	{4444}	Blaster	0.43
$R_{3,4}$	66	False	Windows Update flood	–	X	Blaster	0.3
$R_{3,2}$	100	False	Windows Update flood	–	{135}	Blaster	0.43

EXPERT: No. /* It means no need to add new attribute and **Recommendation II** is then suggested. Otherwise, VODKA will suggest **Recommendation III**. */

Recommendation II.

VODKA: What is the new name and new value set of the attribute A_2?
EXPERT: N_{DTnew}, V_{DTnew}. /* For each old value in A_2, VODKA will ask experts to define the mapping between old and new value sets. */

The row representing A_2 with new mapping values and two columns representing the original and variant objects will be created in T'.

Recommendation III

VODKA: What is the name and value set of the new attribute-value pair?
EXPERT: A_{new}, AV_{new}. /* VODKA will ask experts to provide a set of values (AV_{new}) of the new attribute A_{new}. */

A new row representing the useful attribute namely A_{new} with a set of value (AV_{new}) and two columns representing original object and new variant will be added in T'. If all large itemsets are confirmed, the new variant acquisition table T' can be used to generate embedded rules of discovered variants in the knowledge polishing stage.

3.3. Knowledge polishing using extended EMCUD

Based upon the new variants acquisition table, we propose extended EMCUD (E-EMCUD) algorithm as shown in Algorithm 4 to generate new embedded rules and adjust the original embedded rule base. Therefore, we can gracefully update the embedded rule base using the small new variants acquisition table instead of using the whole large main acquisition table.

Algorithm 4. The E-EMCUD algorithm

```
Input: The original main acquisition table $T$, rule base RB, the new variants acquisition table $T'$, variant objects VO, and new attributes AN, and $\text{TH}_{\text{CF}}$.
Output: The adjusted RB including rules with embedded meaning of new variants.
Step 1: Build the corresponding AOT of $T'$.
Step 2: Generate the embedded rules with embedded meaning of $T'$.
Step 3: Generate the embedded rules with CF values including the original rules with strong CF and embedded rules with marginally acceptable CF.
Step 4: Merge the new variants acquisition table $T'$ into main acquisition table $T$.
    Step 4.1: Append VO and AN as new columns and rows in $T$, respectively.
    Step 4.2: Ask experts to fill the values of the modified attributes of other objects in $T$ if necessary.
    Step 4.3: Ask experts to examine the values of the new attributes of other objects in $T$ if necessary.
Step 5: Reset the new variants acquisition table $T'$.
```

Based upon the AOT table generated in **Step 1**, the embedded rules generated in **Step 2** will be classified into original and embedded rules in **Step 3**. Then, the new variants acquisition table will be merged with the main acquisition table in **Step 4**. Finally, the new variants acquisition table will be reset in **Step 5** to learn other variants.

3.4. The analysis of VODKA

The cost of running VODKA can be divided into two categories: computational cost and interaction cost. Assume there are k attributes to classify m objects in the original main acquisition table, where the grid size is k^*m. For simplifying our discussion, we use $\text{ER}_{k,m}$ to represent the total number of embedded rules in the classification KBS, where $\text{ER}_{k,m} < m^*2^k$.

During the log collection stage, assume n instances are matched by the classification KBS. For each instance, it has P_e probability to be classified by weak embedded rules; hence the size of interesting inference log database in VODKA is n^*P_e.

During the knowledge learning stage, the computational cost is dominated by the learning algorithm we selected. In this paper, Apriori algorithm is used to learn the candidates of variant worms. Hence, the computational cost is O(Aproori). For example, if the size of database has n transactions (each transaction has k attributes) and the maximal length of large itemsets is len, then the time complexity of traditional Apriori algorithm is $O(n^*k^*len)$.

Assume L large itemsets are discovered and used to notify experts to determine the existence of the variants. For each embedded rule, assume it has P_v probability to evolve a variant; hence $P_v^*ER_{k,m}$, denoted V, variants might be discovered. Therefore, the order of interaction with experts is V, where $V < L$.

During the knowledge polishing stage, the E-EMCUD integrates new acquisition table into original acquisition table. The computational cost of E-EMCUD for generating rules is dependent on the size of new acquisition table (GRID), denoted O(GRID). For example, using our E-EMCUD to generate embedded rules, it costs 0.05 ms ~0.15 ms to generate one rule. Fig. 2 shows that the time for generating rules using different grid sizes. The computational time is approximately linear growing when setting a fixed attribute number with different numbers of objects in Fig. 2a, and the growth rate is exponential when setting a fixed object number with different numbers of attributes in Fig. 2b.

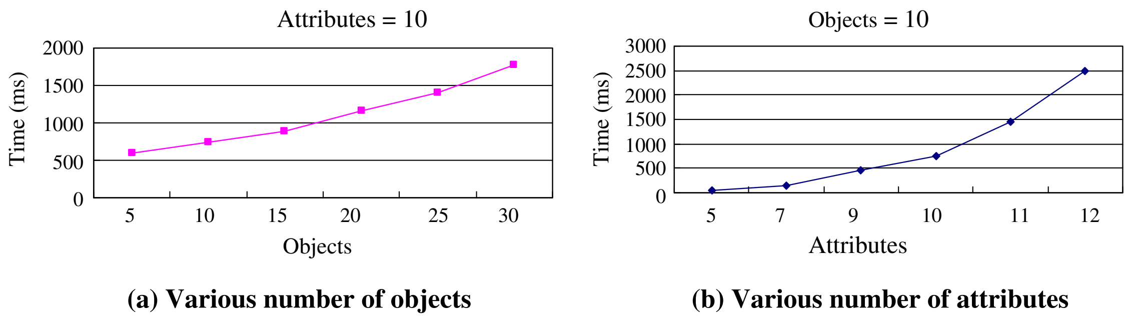

Fig. 2. The time of generating rules using different grid size.

In short conclusion, the cost of VODKA consists two parts: computational cost and interaction cost. The size of interesting inference log database: $n^{*}P_{e}$.

- Computational cost: O(Apriori) + O(GRID).
- Interaction cost: V.

4. VODKA implementation and case studies

Two case studies including computer worms and e-learning domain are used to evaluate the performance of VODKA.

4.1. VODKA implementation

VODKA is implemented by DRAMA (Lin et al., 2003), a new object-oriented rule based system platform implemented using pure Java language, to refine the embedded rule base by observing the inference behaviors of weak embedded rules. It includes DRAMA server, console, knowledge extractor, and rule editor. Also, it provides an application programming interface (API) to access DRAMA server in DRAMA integrated systems. There are four basic relations between knowledge concepts defined in DRAMA: Reference, Extension-of, Trigger and Acquire. The Reference relation represents the association of two different knowledge classes (KCs) if the KCs have common piece of knowledge, which is useful for using original knowledge to construct new knowledge. Extension-of relation is used to extend or modify the KC constructed by other people, which is useful for knowledge sharing and exchanging. The Trigger and Acquire relations are used to represent the interaction of different KCs. The log collecting stage is encoded by four meta-rules described in Section 3.1 in DRAMA; the knowledge learning stage and E-EMCUD are implemented using the JSP to make a communication channel using the API provided by DRAMA.

4.2. The case study of computer worms

With the rapid development of network technology, the network security becomes one of the most important issues today. To prevent network environment from intrusions, lots of researches and different systems are proposed to detect, filter, or prevent intrusions properly. Recently, the computer worms become more complicated owing to combining several signatures of previous known worms. The number of computer worms has grown dramatically to influence the wide computer networks due to the property of easily modifying the source codes of original computer worms to create new variants for escaping the detection of related systems, e.g., Symantec Norton (Symantec, 2005), Network Viruswall (Trend Micro, 2005), etc. They are very difficult for experts to get and analyze the signatures because they have incredibly sophisticated characteristics (Kienzle & Elder, 2003; Moore et al., 2003; Weaver et al., 2003). Some researchers have proposed machine learning, data mining, and clustering approaches to discover and learn outliers or abnormalities (the new types of objects) (Lin, Tseng, & Lin, 2002).

Generally speaking, a computer worm usually self-propagates via a network to acutely generate a crisis in our systems through the following four stages: target selection, exploitation, infection, and propagation (Weaver et al., 2003). At the Target Selection Stage, a worm performs reconnaissance and simply probes potential victims to see if it's running a service on a particular port. If the service is running, the worm goes to exploitation stage, in which it compromises the target by exploiting a particular vulnerability and published exploits. If it succeeds, the worm goes to infection stage, in which it sets up on the newly infected machine. Finally, in propagation stage, the worm starts to spread by choosing new targets. And another victim will enter the next four stages cycle.

In our worm detection prototype system, the knowledge of computer worms can be divided into several KCs, including the service provided by the host may be infected by certain worms and then produced some symptoms in host or network. The related attributes of various computer worms can be collected by some probe tools and used to evaluate the ability of VODKA, which deployed in the prototype system.

In order to evaluate VODKA system, an experimental environment shown in Fig. 3 for detecting various computer worms is built. In this environment, the victim could receive both the normal traffic and the attacking traffic

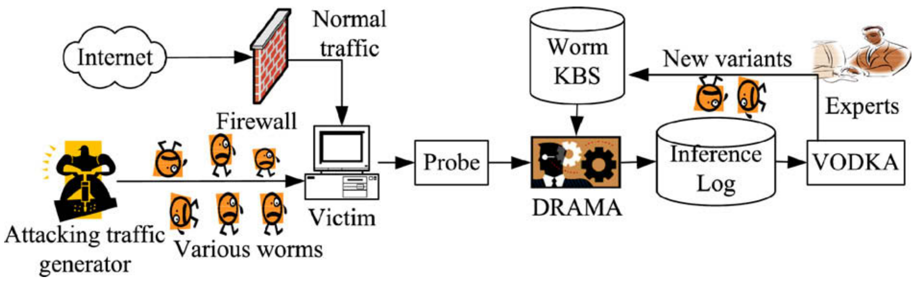

Fig. 3. The experimental environment for detecting computer worms.

(various worm behaviors). All received traffic can be treated as normal or attacking behavior, which can be transformed as attribute-value pairs. The network traffic collected from Internet is assumed as normal traffic since most attacking behaviors with significant signatures will be filtered by a firewall. The attacking traffic generator is designed to randomly generate various worms attacking traffic to infect the victim. Besides the attacking traffic, some signatures, e.g., the system status, host vulnerability information, and large e-mailing behavior, of the victim infected by worms can be also collected. The probe, such as Nessus (Tenable, 2005), is also used to automatically collect these worm related attributes (symptoms). The classification rules of 15 kinds of worm families, including original worms and some variant worms, are extracted using EMCUD and then these worm classification rules are stored into the worm KB. Then, these attributes are used to trigger the corresponding classification rules in the worm KB. If variant worms occurred frequently in a period of time, some candidate worm variants may be discovered by VODKA. Finally, the corresponding embedded rules of variant worms confirmed by experts will be generated to update the worm KB.

To evaluate the effectiveness of VODKA, we generated 20 kinds of test samples including the behaviors of 15 kinds of original worm families and five kinds of new worm families to randomly attack the victims. The experimental result shows that VODKA can detect 100% of original worms since the classification rules are stored in the worm KB. For detecting the occurrence of variant worms, VODKA can learn the 85% of variant worms. However, VODKA can detect 80% of new worm families in our experiments because the significant difference of new family can not be discovered easily. Thus, deploying the most complementary configurations in a collaborative framework could be efficient in discovering the new created worms.

The following example shows that the variant objects in this domain can be discovered by VODKA, where $\mathrm{TH}_{\mathrm{CNT}}$ is set to 4, $\mathrm{TH}_{\mathrm{CF}}$ is set to 0.7, and the minimal support is set to 30%.

Recommendation I. Elicitation of new variants by adding a new attribute value of a minor attribute.

Nimda worm, a famous Email flooding worm, can be propagated to victims through the attached files in email. By monitoring the attached filename in email, we can discover the large itemset $L = (A_4 = \text{readme.exe})$ shown in Table 7 according to the embedded rule $R_{1,1}$ in Table 4.

Based upon the large itemsets, VODKA will ask the following questions.

VODKA: Does the attribute-value pair ($A_4 =$ readme.exe) belong to any new variant object?
EXPERT: Yes.
VODKA: What is the name of the new variant object?
EXPERT: Nimda.B.
VODKA: Is the data type of A_4 required to be changed?
EXPERT: No.

Consequently, the new variant acquisition table of Nimda.B shown in Table 8 will be generated after interviewing the experts in this iteration.

Hence, an original rule "IF (DoS type = Email flood) AND (Email attached file = (readme.exe)) THEN Nimda.B, CF = 0.8" and an embedded rule "IF (DoS type = Email flood) AND ¬ (Email attached file = (readme.exe)) THEN Nimda.B, CF = 0.5" of the Nimda.B will be generated using E-EMCUD based upon the Nimda.B acquisition table.

Recommendation II. Elicitation of new variants by changing the data types of attributes.

In the priori generation of CodeRed worm, generating numerous threads to attack the victim through launching TCP flooding is one of the famous characteristics. Hence, it is useful to detect the CodeRed by analyzing the generated anomaly threads in the protected system. For the partial detection rules of CodeRed shown in Table 9, the following shows how the values in Boolean data type will be logged as the integer value of the attribute A_1 instead of true/false value.

Using the above inference log, the large itemset $L = (A_1 = 600)$ can be discovered based upon the embedded rule $R_{2,2}$ since the minimal support 30% is satisfied. Then, VODKA will ask the following questions.

VODKA: Does the attribute-value pair ($A_1 = 600$) belong to any new variant object?
EXPERT: Yes.
VODKA: What is the name of the new variant object?

Table 7
The partial inference logs of Nimda

Rule #	A_1	A_2	A_3	A_4	A_5	Object	CF
$R_{1,1}$	6	True	Email flood	readme.exe	{137}	Nimda	0.4
$R_{1,1}$	100	False	Email flood	sample1.exe	{25}	Nimda	0.4
$R_{1,1}$	17	True	Email flood	readme.exe	{137}	Nimda	0.4
$R_{1,1}$	14	False	Email flood	readme.exe	{25}	Nimda	0.4
$R_{1,1}$	4	False	Email flood	readme.exe	{445}	Nimda	0.4
$R_{1,1}$	19	False	Email flood	readme.exe	{80}	Nimda	0.4
$R_{1,1}$	44	False	Email flood	hash.exe	{25}	Nimda	0.4
$R_{1,1}$	38	True	Email flood	readme.exe	{138}	Nimda	0.4
$R_{1,1}$	100	False	Email flood	inter.exe	{25}	Nimda	0.4
$R_{1,1}$	28	False	Email flood	readme.exe	{25}	Nimda	0.4

EXPERT: CodeRed.II.
VODIKA: Is the data type of A_1 required to be changed?
EXPERT: Yes.
VODKA: Is any new attribute required to be added?
EXPERT: No.
VODKA: Can the Single Value data type be used to change the original Boolean data type of A_1?
EXPERT: Yes.
VODKA: What is the new name and new value set of the attribute A_1?
Expert: Threads, (100, infinite).
VODKA: What is the new value of the original True value of the attribute A_1 in CodeRed?
EXPERT: 100.

Therefore, the new variant acquisition table of CodeRe-d.II shown in Table 10 will be generated.

Consequently, an original rule "IF (Threads = 600) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed.II, CF=0.9" and an embedded rule "IF ¬ (Threads = 600) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed.II, CF = 0.3" will be generated to classify the CodeRed.II.

Table 8
The new variant acquisition table of Nimda.B

Attributes	Object
	Nimda.B
Threads	X
System reboot	X
DoS type	Email flood
Email attached file	{readme.exe}
TCP port	X

Table 10
The new variant acquisition table of CodeRed.II

Attributes	Objects	
	CodeRed	CodeRed.II
Threads	100	600
System reboot	True	True
DoS type	TCP flood	TCP flood
Email attached file	X	X
TCP port	X	X

Recommendation III. Elicitation of new object by adding new attributes.

As mentioned in Example 2, we can obtain the large itemsets $L = \{(A_2 = \text{False}); \quad (A_5 = X); \quad (A_2 = False, A_5 = X)\}$, which will be used to elicit the embedded rules of the new variant. The symbol "X" means no attribute value of A_5 is logged, similar to "Do not care" attribute, and (A_2 = False, $A_5 = X$) will also be pruned too. Therefore, VODKA system will ask the following questions.

VODKA: Does the attribute-value pair (A_2 = False) belong to the new variant object?
EXPERT: Yes.
VODKA: What is the name of the new variant object?

Table 9
The partial inference logs of CodeRed

Rule #	A_1	A_2	A_3	A_4	A_5	Object	CF
$R_{2,2}$	150	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,2}$	600	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,1}$	100	False	TCP flood	–	{80}	CodeRed	0.6
$R_{2,2}$	600	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,2}$	150	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,1}$	100	False	TCP flood	–	{80}	CodeRed	0.6
$R_{2,2}$	600	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,2}$	600	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,2}$	600	True	TCP flood	–	{80}	CodeRed	0.4
$R_{2,2}$	300	True	TCP flood	–	{80}	CodeRed	0.4

EXPERT: Blaster.B.
VODKA: Is the data type of A_2 required to be changed?
EXPERT: Yes.
VODKA: Is any new attribute required to be added?
EXPERT: Yes.
VODKA: What is the name and value set of the new attribute?
EXPERT: UDP port, (0, 65535).
VODKA: What is the value of the UDP port attribute in Blaster.B and Blaster?
EXPERT: 69, X (means Do not care).

Hence, the Blaster.B acquisition table shown in Table 11 is generated.

Consequently, a new original rule "IF (System reboot = Flase) AND (DoS type = Windows update flood) AND (UDP port = {69}) THEN Blaster.B, CF = 0.8" and an embedded rule "IF (System reboot = Flase) AND (DoS type = Windows update flood) AND ¬ (UDP port = {69}) THEN Blaster.B, CF = 0.5" of new variant Blaster.B will be generated to classify Blaster.B.

As shown in Table 12, four variants (Nimda.B, CodeRed.II, Blaster.B, and Welchia.II) have been successfully singled out using VODKA after several iterations. Table 13 shows the AOT table after interacting with domain experts using E-EMCUD, and Table 14 shows the new embedded rule base of the discovered variants and original worms. If more real instances can be used, the embedded rule base will evolve and become more precise for classifying the computer worms.

4.3. The case study of e-learning

With the vigorous development of the Internet, e-learning systems including online learning, employee training courses, and e-book, have become more and more popular over the world in the past ten years (Beishuizen & Stoutjesdijk, 1999; Hwang, 1999; Yoshikawa, Shintani, & Ohba, 2000). As we know, if the same teaching materials (the knowledge of teachers) are provided to all e-learners based on the predefined strategies or the predefined learning maps, the learning efficiency will be diminished. Therefore, teachers want to apply appropriate teaching strategy to provide personalized learning content and learning sequence for e-learners to improve their learning efficiency. Thus, adaptive learning environments (Shang, Shi, & Chen, 2001; Sheremetov & Arenas, 2002; Triantafllou, Poportsis, & Demetriadis, 2003; Tsai & Tseng, 2002) have been proposed to offer different teaching materials for dif-

Table 11
The new variant acquisition table of Blaster.B

Attributes	Objects	
	Blaster	Blaster.B
100-thread (A_1)	X	X
System reboot (A_2)	True	False
DoS type (A_3)	Windows Update flood	Windows Update flood
Email attached file (A_4)	X	X
TCP port (A_5)	{135;4444}	{135;4444}
UDP port (A_6)	X	69

Table 12
The adjusted main acquisition table of simple computer worms

Attributes	Objects							
	Nimda	Nimda.B	CodeRed	CodeRed.II	Blaster	Blaster.B	Welchia	Welchia.II
Threads	X	X	100	600	X	X	X	X
System reboot	X	X	True	True	True	False	True	True
DoS type	Email flood	Email flood	TCP flood	TCP flood	Windows Update flood	Windows Update flood	ICMP flood	ICMP flood
Email attached file	{samle.exe; puta!!scr}	{readme, exe}	X	X	X	X	X	X
TCP port	X	X	X	X	{135;4444}	{135;4444}	{80;135}	{80;135;445; 3127}
UDP port	X	X	X	X	X	69	X	X

Table 13
AOT table of simple computer worms

Attributes	Objects							
	Nimda	Nimda.B	CodeRed	CodeRed.II	Blaster	Blaster.B	Welchia	Welchia.II
Threads	X	X	2	2	X	X	X	X
System reboot	X	X	1	1	2	2	2	2
DoS type	D	D	1	1	2	2	1	1
Email attached file	1	1	X	X	X	X	X	X
TCP port	X	X	X	X	1	1	2	2
UDP port	X	X	X	X	X	1	X	X

Table 14
The rules generated from Tables 12 and 13

$R_{1,0}$: IF (DoS type = Email flood) AND (Email attached file = (sample.exe; puta!!scr)) THEN Nimda, CF = 0.8
$R_{1,1}$: IF (DoS type = Email flood) AND (Email attached file = ¬ (sample.exe; puta!!scr)) THEN Nimda, CF = 0.4
$R_{2,0}$: IF (Threads = 100) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed, CF = 0.8
$R_{2,1}$: IF (Threads = 100) AND ¬ (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed, CF = 0.6
$R_{2,2}$: IF ¬ (Threads = 100) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed, CF = 0.4
$R_{2,3}$: IF (Threads = 100) AND ¬ (System reboot = True) AND ¬ (DoS type = TCP flood) THEN CodeRed, CF = 0.4
$R_{3,0}$: IF (System reboot = True) AND (DoS type = Windows update flood) AND (TCP port = {135; 4444}) THEN Blaster, CF = 0.7
$R_{3,1}$: IF (System reboot = True) AND (DoS type = Windows update flood) AND ¬ (TCP port = {135; 4444}) THEN Blaster, CF = 0.57
$R_{3,2}$: IF (System reboot = False) AND (DoS type = Windows update flood) AND (TCP port = {135; 4444}) THEN Blaster, CF = 0.43
$R_{3,3}$: IF (System reboot = True) AND ¬ (DoS type = Windows update flood) AND (TCP port = {135; 4444}) THEN Blaster, CF = 0.43
$R_{3,2}$: IF (System reboot = False) AND (DoS type = Windows update flood) AND ¬ (TCP port = {135; 4444}) THEN Blaster, CF = 0.3
$R_{4,0}$: IF (System reboot = True) AND (DoS type = ICMP flood) AND (TCP port = {80; 135}) THEN Welchia, CF = 0.8
$R_{4,1}$: IF (System reboot = True) AND ¬ (DoS type = ICMP flood) AND (TCP port = {80; 135}) THEN Welchia, CF = 0.67
$R_{4,2}$: IF (System reboot = True) AND (DoS type = ICMP flood) AND ¬ (TCP port = {80; 135}) THEN Welchia, CF = 0.53
$R_{4,3}$: IF (System reboot = True) AND ¬ (DoS type = ICMP flood) AND ¬ (TCP port = {80; 135}) THEN Welchia, CF = 0.4
$R_{5,0}$: IF (DoS type = Email flood) AND (Email attached file = readme.exe) THEN Nimda.B, CF = 0.8
$R_{5,1}$ IF (DoS type = Email flood) AND ¬ (Email attached file= readme.exe) THEN Nimda.B, CF = 0.5
$R_{6,0}$: IF ¬ (Threads = 600) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed II, CF = 0.9
$R_{6,1}$: IF (Threads = 600) AND (System reboot = True) AND (DoS type = TCP flood) THEN CodeRed II, CF = 0.3
$R_{7,0}$: IF (System reboot = False) AND (DoS type = Windows update flood) AND (UDP port = {69}) THEN Blaster.B, CF = 0.8
$R_{7,1}$: IF (System reboot = False) AND (DoS type = Windows update flood) AND ¬ (UDP port = {69}) THEN Blaster.B, CF = 0.5
$R_{8,0}$: IF (System reboot = True) AND (DoS type = ICMP flood) AND (TCP port = {80; 135; 445; 3127}) THEN Welchia II, CF = 0.8
$R_{8,1}$: IF (System reboot = True) AND (DoS type = ICMP flood) AND ¬ (TCP port = {80; 135; 445; 3127}) THEN Welchia II, CF = 0.5

ferent learners in accordance with their aptitudes and evaluation results. After learners learned the teaching materials through the adaptive learning environment, the teachers can further analyze the historical learning records and then refine or reorganize the teaching materials and tests if needed. Therefore, more and more attention has been paid to the research of personalized instruction in computer education environment. However, it is difficult to monitor the change of learner's behaviors for teachers quickly.

As we know, the quiz for learners is useful to evaluate their learning achievement. For example, teachers should provide easier learning content or learning sequence for the learners with lower learning achievement. Hence, VODKA provides a good idea to assist teachers in observing the occurrence of variant learning behaviors of e-learners through a sequence of grades of online quiz.

In this case, the objects to be classified are defined as learning behaviors of learners, where each behavior consists of profiles, learning sequence, and a quiz grade of the learner, where learning sequence can be used to generate approximate teaching materials for matching the learner's needs. The learners can be firstly clustered into several groups according to the similarity of the learning behaviors, and teachers can provide appropriate learning content for each group in advance. However, e-learners might change their learning sequences due to the different learning situation, learning equipments (desktop, PDA, etc.), course content (text, video, etc.), and learning time (day or night). This causes the evolution of learning behaviors of e-learners and results in various learning achievements.

Assume that the on-line testing system is implemented in an intelligent e-learning system. VODKA can monitor the variant learning behaviors to evaluate the learning performance of each e-learner, where each learner has different learning sequences. The teachers will be then notified to generate a suitable learning sequence and apply these materials on those e-learners with variant learning behaviors. Here, each learning sequence deviated from one of predefined learning sequences will be treated as a variant learning behavior. In e-learning, it is important for e-learners to gain a good grade after learning some materials with a specific learning sequence. Hence, the grade of quiz is treated as a CF value for collecting these good variant learning sequences. If many learners gained grades higher than a threshold with similar or same learning sequence (high frequency), then some good variant learning sequences will be discovered to notify teachers to determine these new learning sequences. Therefore, the log is collected as the pair of $\langle LS_i, CF_i \rangle$, where LS_i is the learning sequence of the e-learner i; and the CF_i is the grade of this learner. Example 3 illustrates the concept of e-learning using VODKA.

Table 15
The learning sequence of students

Student ID	Learning sequence
1	⟨B, C, A, D, E, F, G, H, I, J⟩
2	⟨A, B, H, D, E, F, C, G, I, J⟩
3	⟨A, D, F, G, H, B, C, I, J⟩
4	⟨A, B, D, E, C, F, G, H⟩
5	⟨A, C, J, F, B, H, D, E, I, G⟩
6	⟨B, H, F, D, E, A, G, C, I⟩
7	⟨A, J, E, H, B, C, I, D, G⟩
8	⟨B, C, G, E, A, H, D, I, J, F⟩
9	⟨C, E, G, F, J, B, H, A, D⟩
10	⟨B, C, A, J, D, E, G, H, F, I⟩

Table 16
The maximal frequent learning patterns of good students

Large itemset	Maximal frequent learning patterns									
L_2	A → F	A → H	A → J	B → H	C → D	C → F	C → H	E → F	F → G	G → H
L_3	A → D → G	B → C → G								
L_4	B → D → E → G									

Example 3. The concept of e-learning using VODKA.

To simplify our discussion, we assume VODKA collects several good learning behaviors of e-learners and their grades of quiz are larger than a threshold. For the learning sequence log shown in Table 15, $LS_1 = \langle B, C, A, D, E, F, G, H, I, J\rangle$ denotes that Learner 1 studies the learning content B first and then studies the learning contents C, A, D, E, F, G, H, I, J sequentially.

In this example, the sequential pattern mining algorithm instead of the original Apriori algorithm is applied (Aggrawal & Srikant, 1995; Srikant & Aggrawal, 1996). Therefore, we use the Modified GSP algorithm to discover the maximal frequent learning patterns as shown in Table 16. The details can be found in (Su et al., 2006). For example, in L_4, we have discovered that one candidate of new learning sequence of good e-learners is to learn B course content first and then to study the learning contents D, E, G sequentially. Hence, these candidates of various learning sequences will be suggested by VODKA for teachers to generate new variant learning sequences.

In this case study, we illustrate that VODKA can collect all interesting learning behaviors (learning sequences) of e-learners whose testing grade from online quiz system is good; hence, then the maximal frequent learning sequence, a part of the whole learning sequence, will be used to recommend teachers to adapt their course for variant learning behaviors if necessary.

5. Conclusion

In this paper, we proposed VODKA methodology to iteratively monitor the frequent inference behaviors of weak embedded rules with margin values of CF to assist human experts in discovering the new variant objects and singling them out of original objects. The frequent ignored attribute-value pairs of minor attributes can be learned by applying the Apriori algorithm and be treated as the new characteristics of variant objects. The new variants acquisition algorithm is proposed to interact with experts to acquire the relationships between new variant objects and object attributes. Three recommendations, including adding a new attribute value of an attribute, changing the data type of an attribute, or adding a new attribute, are proposed to help experts confirm the new variants according to the frequently ignored attribute-value pairs. Additionally, we proposed E-EMCUD to integrate the new variant acquisition table into the main acquisition table for extracting the embedded rules of new variants. A computer worm detection prototype based upon DRAMA has been implemented and deployed in an experimental environment. This environment includes a firewall to filter computer worm traffic from Internet (normal traffic) and an attacking traffic generator to randomly generate various worms to infect a victim. By this means, we evaluate the performance of VODKA.

The results show that new worm variants can be singled out of the corresponding extended worm object classes after observing the occurrence of worm instances in the inference logs. Then, the detection rules for computer worms can be customized. Also, an e-learning case is shown that VODKA can help teachers quickly and easily single out the variant learning behaviors of e-learners. Then teachers can be notified to prepare new learning content and learning sequence for e-learners with similar variant behaviors. Therefore, these kinds of learners could easily understand the new learning materials. VODKA method can enhance the classification ability for new objects of a classification KBS since the new evolving objects can be incrementally learned and discovered by collecting the sufficient information. We are going to extend VODKA into a collaborative framework to help experts discover more evolving objects.

Acknowledgement

This work was partially supported by National Science Council of the Republic of China under Grant Nos. NSC93-2752-E-009-006-PAE, NSC95-2752-E-009-015-PAE and NSC96-2752-E-009-006-PAE.

References

Aggrawal, R., Srikant, R. (1995). Mining sequential patterns. In *Proceedings of 11th international conference on data engineering* (pp. 3–14).

Agrawal, R., Imielinksi, T., Swami, A. (1993). Mining association rules between sets of items in large database. In *Proceedings of the ACM SIGMOD conference* (pp. 207–216).

Alani, H., & Kim, S. (2003). Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems, 18*(1), 14–21.

Beishuizen, J. J., & Stoutjesdijk, E. T. (1999). Study strategies in a computer assisted study environment. *International Journal of Learning and Instruction, 9*, 281–301.

Blythe, J., Kim, J., Ramachandran, S., Gil, Y. (2001). An integrated environment for knowledge acquisition. In *Proceedings of the international conference on intelligent user interfaces* (pp. 13–20).

Boegl, K. (1997). *Design and implementation of a web-based knowledge acquisition toolkit for medical expert consultation systems*. Doctorial thesis, Technical University of Vienna, Austria.

Boose, J. H. (1984). Personal construct theory and the transfer of human expertise. In *Proceedings of AAAI-84 conference, California* (pp. 27–33).

Boose, J. H. (1985). A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man-Machine Studies, 23*(5), 495–525.

Boose, J. H., Bradshaw, J. M. (1986). NeoETS: Capturing expert system knowledge in hierarchical rating grids. In *IEEE expert system in government symposium*.

Boose, J. H., & Bradshaw, J. M. (1987). Expertise transfer and complex problems: Using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. *International Journal of Man-Machine Studies, 26*(1), 3–28.

Cairo, O. (1998). KAMET: A comprehensive methodology for knowledge acquisition from multiple knowledge sources. *Expert Systems with Applications, 14*(1), 1–16.

Caragea, D., Silvescu, A., Pathak, J., Bao, J., Andorf, C., Yan, C., et al. (2005). Knowledge acquisition from autonomous, distributed, semantically heterogeneous data sources. In *Proceedings of the annual meeting of the international society for computational biology (ISMB 2005)*, Poster Program, Detroit, Michigan.

Caragea, D., Silvescu, A., & Honavar, V. (2001). Towards a theoretical framework for analysis and synthesis of agents that learn from distributed dynamic data sources. *Emerging neural architectures based on neuroscience*. Springer-Verlag (pp. 547–559).

Castro-Schez, J. J., Jennings, N. R., Luo, X. D., & Shadbolt, N. R. (2004). Acquiring domain knowledge for negotiating agents: A case of study. *International Journal of Human-Computer Studies, 61*(1), 3–31.

Crowther, P., Hartnett, J. (1996). Using repertory grids for knowledge acquisition for spatial expert system. In *Proceedings of Australia and New Zealand conference on intelligent information systems, Adelaide, SA, Australia, November 18–20* (pp. 14–17).

Dixit, A. K., & Pindyck, R. S. (1994). *Investment under uncertainty*. Princeton University Press.

Eriksson, H. (2003). Using JessTab to integrate Protege and Jess. *IEEE Intelligent Systems, 18*(2), 43–50.

Fensel, D., & Angele, J. (1988). The knowledge acquisition and representation language, KARL. *IEEE Transaction on Knowledge and Data Engineering, 10*(4), 527–550.

Frank, G., & Farquhar, A. (1999). Building a large knowledge base from a structured source (KA and ontology). *IEEE Intelligent Systems, 14*(1), 47–54.

Gaines, B. R. (1987). An overview of knowledge-acquisition and transfer. *International Journal of Man-Machine Studies, 26*, 453–472.

Ginsberg, A., Weiss, S. M., & Politakis, P. (1988). Automatic knowledge base refinement for classification systems. *Artificial Intelligence, 35*(2), 197–226.

Hong, T. P., & Tseng, S. S. (1997). A generalized version space learning algorithm for noisy and uncertain data. *IEEE Transaction on Knowledge and Data Engineering, 9*(2), 336–340.

Hwang, G. J. (1995). Knowledge acquisition for fuzzy expert systems. *International Journal of Intelligent Systems, 10*, 541–560.

Hwang, G. J. (1999). A knowledge-based system as an intelligent learning advisor on computer networks. *Proceedings of the IEEE Transactions on Systems Man and Cybernetics, 2*, 153–158.

Hwang, G. J., & Tseng, S. S. (1990). EMCUD: A knowledge acquisition method which captures embedded meanings under uncertainty. *International Journal of Man-Machine Studies, 33*, 431–451.

Hwang, G. J., & Tseng, S. S. (1991). On building a medical diagnostic system of acute exanthema. *Journal of Chinese Institute of Engineers, 14*(2), 185–195.

Kang, B. (1996). Multiple classification ripple down rules. Ph.D Thesis, University of New South Wales.

Kelly, G. A. (1955). *The psychology of personal constructs*. New York: Norton.

Kienzle, D. M., Elder, M. C. (2003). Recent worms: A survey and trends. In *Proceedings of the WORM'03, October 27, Washington DC, USA 2003*.

Kim, J., Gil, Y. (2001). Knowledge analysis on process models. In *Proceedings of the international joint conference on artificial intelligent, Seattle, Washington, USA*.

Kitamura, Y., & Mizoguchi, R. (2003). Ootology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications, 24*(2), 153–166.

Kolousek, G. (1997). *The system architecture of an integrated medical consultation system and its implementation based on fuzzy technology*. Doctoral thesis, Technical University of Vienna, Austria.

Leibbink, H. J., Witteman, C. L. M., Mayer, J. J. C. (2002). Ontology-based knowledge acquisition for knowledge systems. In *Proceedings of the 14th Dutch–Belgian artificial intelligence conference* (pp. 195–202).

Leitich, H., Kiener, H. P., Kolarz, G., Schuh, C., Graninger, W., & Adlassnig, K. P. (2001). A prospective evaluation of the medical consultation system CADIAG-II/RHEUMA in a rheumatological outpatient clinic. *Methods of Information in Medicine, 40*, 213–220.

Lin, S. C., Tseng, S. S., & Lin, Y. T. (2002). A new mechanism of mining network behavior. *Lecture Notes in Artificial Intelligence, 2336*, 218–223.

Lin, Y. T., Tseng, S. S., & Tsai, C. F. (2003). Design and implementation of new object-oriented rule base management system. *Journal of Expert Systems with Applications, 25*(3), 369–385.

Maedche, A., & Staab, S. (2001). Ontology learning for the Semantic Web. *IEEE Intelligent Systems, 16*(2), 72–79.

Mcgraw, K. L., & Harbison-Briggs, K. (1989). *Knowledge acquisition: Principles and guidelines*. Prentice-Hill International Editions, 1–27.

Mirkovic, J., Martin, J., Reiher, P. (2002). *A taxonomy of DDoS attacks and DDoS defense mechanisms*. TR. 020018, Computer Science Department, University of California, Los Angeles.

Moore, D., Shannon, C., Voelker, G. M., Savage, S., 2003. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of INFOCOM 2003, March 30–April 3, San Francisco, USA, 2003*.

Noy, N. F., & Sintek, M. (2001). Creating semantic web contents with Protege-2000. *IEEE Intelligent Systems, 16*(2), 60–71.

Politakis, P., & Weiss, S. M. (1984). Using empirical analysis to refine expert system knowledge bases. *Artificial Intelligence, 22*, 673–680.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Rosaci, D. (2005). Exploiting agent ontologies in B2C virtual marketplaces. *Journal of Universal Computer Science, 11*(6), 1011–1040.

Rosaci, D., Terracina, G., & Ursino, D. (2004). A framework for abstracting data sources having heterogeneous representation formats. *Data and Knowledge Engineering, 48*(4), 1–38.

Shang, Y., Shi, H. C., & Chen, S. S. (2001). An intelligent distributed environment for active learning. *ACM Journal of Education Resources in Computing, 1*(2), 4–17.

Shaw, M. L. G., Gaines, B. R. (1996). Web grid: Knowledge modeling and inference through the world wide web. In *Proceedings of the 10th knowledge acquisition workshop* (pp. 65-1–65-14).

Shaw, M. L. G., & Gaines, B. R. (1987). KITTEN: Knowledge initiation and transfer tools for experts and novices. *International Journal of Man-Machine Studies, 27*, 251–280.

Sheremetov, L., & Arenas, A. G. (2002). EVA: Am interactive web-based collaborative learning environment. *Computers & Education, 39*(2), 161–182.

Shortliffe, E. H., & Buchanan, B. G. (1975). A model of inexact reasoning in medicine. *Mathematical Bioscience, 23*, 351–379.

Singh, P., Lin, T., Mueller, E. T., Lim, Grace, Perkins, T., et al. 2002. Open mind common sense: Knowledge acquisition from general public. In *Proceedings of the first international conference on ontologies, databases, and applications of semantics for large scale information systems* (pp. 1223–1237).

Srikant, R., Aggrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *The Fifth International Conference on Extending Database Technology, 1996*.

Su, J. M., Tseng, S. S., Wang, W., Weng, J. F., Yang, David J. T., & Tsai, W. N. (2006). Learning portfolio analysis and mining for SCORM compliant environment. *Journal of Educational Technology & Society, 9*(1), 262–275.

Symantec Co., 2005. Symantec Products and Services. <http://www.symantec.com/product/>.

Tenable Network SecurityTM, 2005. Nessus Open Source Vulnerability Scanner Project. <http://www.nessus.org/>.

Trend Micro Co., 2005. Trend Mirco Network Viruswall. <http://www.trendmicro.com/tw/products/network/overview.htm>.

Triantafllou, E., Poportsis, A., & Demetriadis, S. (2003). The design and the formative evaluation of an adaptive educational system based on cognitive styles. *Computers & Education, 41*(1), 87–103.

Tsai, C. J., & Tseng, S. S. (2002). Building a CAL expert system based upon two-phase knowledge acquisition. *Expert Systems with Applications, 22*, 235–248.

Tsujino, K., Dabija, V., & Nishida, S. (1992). Knowledge acquisition driven by constructive and interactive induction. *Lecture Notes in Artificial Intelligence, 599*, 153–170.

Tsujino, K ., Takegaki, M ., Nishida, S., 1990. A knowledge acquisition system that aims at integrating inductive learning and explanation-based reasoning. In *Proceedings of the first Japanese knowledge acquisition for knowledge – based systems workshop, Tokyo, Japan* (pp. 175–190).

Uschold, M., King, M., Moralee, S., & Zorgios, Y. (1998). The enterprise ontology. *The Knowledge Engineering Review, 13*(1), 31–89.

Weaver, N., Paxson, V., Staniford, S., Cunningham, R., Maulik, U. (2003). A taxonomy of computer worms. In *Proceedings of WORM'03, October 27, Washington DC, USA, 2003*.

Wielinga, B., Schreiber, A., & Breuker, J. (1992). KADS: A modeling approach to knowledge engineering. *Journal of Knowledge Acquisition, 4*(1), 5–53.

Yoshikawa, A., Shintani, M., & Ohba, Y. (2000). Intelligent tutoring system for electric circuit exercising. *IEEE Transactions on Eduction, 35*, 222–225.

Zacklad, M., & Fontaine, D. (1995). Systematic building of conceptual classification systems with C-KAT. *International Journal of Human-Computer Studies, 44*(5), 603–627.

Zhang, J., Honavar, V. (2003). Learning decision tree classifiers from attribute value taxonomies and partially specified data. In *Proceedings of the international conference on machine learning (ICML-03), Washington, DC*.