國立交通大學

資訊科學與工程研究所

碩 士 論 文

CAKE – 合作式演化性知識擷取方法

CAKE - Collaborative Acquisition for Knowledge Evolution

研 究 生：鄧嘉文

指導教授：曾憲雄　教授

中 華 民 國　九 十 五　年 六 月

合作式演化性知識擷取方法

Collaborative Acquisition for Knowledge Evolution

研 究 生：鄧嘉文　　Student：Chia-Wen Teng

指導教授：曾憲雄博士　　Advisor：Dr. Shian-Shyong Tseng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

CAKE – 合作式演化性知識擷取方法

研究生：鄧嘉文　　指導教授：曾憲雄 博士

國立交通大學資訊科學與工程研究所

摘要

由於知識爆炸，知識可以分類為靜態知識和動態知識。儘管已有很多的知識擷取方法可以系統化的從領域專家那取得靜態知識的規則，卻沒有任何一種方法去深入討論發掘動態知識，其原因在於欠缺足夠的情境資訊(context information)。在本篇論文裡，我們提出一個方法命名為合作式演化性知識擷取(*CAKE*)的新知識擷取方法，其透過收集足夠的情境資訊來幫助專家發現動態物件的產生進而發掘動態知識。我們首先定義靜態個性化配置檔案(Profile)和動態行為來做為情境資訊並以服務敏感性和症狀相似性等事件來做為合作啟發(heuristic)以輔助專家意識到動態知識的發生。*CAKE* 中我們用變異物件知識擷取(*VODKA*)和趨勢事件擷取(*TEA*)來建立動態擷取表格以增進發現變異物件的效率並自動的調整屬性順序表格(AOT)內物件與屬性之間的重要程度來更進一步發現演化性知識。這對協助專家了解屬性對物件之間的改變有很大的幫助。再來，我們設計 *CAKE* 讓動態 *EMCUD* 增加新的物件或新的物件屬性來更新已存在的知識表格並進而透過動態 AOT 將其原來的隱含規則具有適應的能力。除此之外，我們開發了一個電腦蠕蟲偵測雛型系統來驗證 *CAKE* 的效能。

關鍵字：靜態知識，動態知識，知識擷取，*VODKA*，動態 *EMCUD*，*TEA*，*CAKE*

CAKE – Collaborative Acquisition for Knowledge Evolution

Student: Chia-Wen Teng Advisor: Dr. Shian-Shyong Tseng

Department of Computer Science
National Chiao Tung University

Abstract

Due to the knowledge explosion, the knowledge can be classified into static knowledge and dynamic knowledge. Although many knowledge acquisition methodologies have been proposed to systematically elicit rules of static knowledge from domain experts, none of these methods discusses the issue of discovering dynamic knowledge due to the lack of sufficient context information. In this thesis, we will propose a new collaborative knowledge acquisition methodology, Collaborative Acquisition for Knowledge Evolution (*CAKE*), to solve the issue of discovering dynamic knowledge by collecting sufficient relevant context information to help experts notice the occurrence of dynamic object. First, we define static profiles and dynamic behaviors as the context information to assist experts to be aware of the occurrence of dynamic knowledge based on several collaborative heuristics for service-sensitive and symptom-similar events. *Variant Objects Discovering Knowledge Acquisition* (*VODKA*) and Trend Event Acquisition (*TEA*) are used to construct a new dynamic acquisition table to facilitate the acquisition of variant knowledge and to automatically adjust the relative importance of each attribute to each object in the attribute ordering table (AOT) to discover the evolutional knowledge in *CAKE*. This is useful to help experts understand the changing behaviors

of attributes to each object. Furthermore, *CAKE* is designed to use *Dynamic EMCUD*, a new version of an existing knowledge acquisition system called *EMCUD* which relies on the repertory grids knowledge acquisition technique to manage object/ attribute-values tables and to produce inferences rules from these tables, to update existing tables by adding new objects or new object attributes in new acquisition for adapting the original embedded rules with the dynamic AOT. Besides, a *Worm detection prototype* system is implemented to evaluate the effectiveness of *CAKE*.

Keywords: static knowledge, dynamic knowledge, knowledge acquisition, *VODKA*, dynamic *EMCUD*, *TEA*, *CAKE*

致謝

能順利完成本篇論文，最重要的必須感謝我的指導教授，曾憲雄博士。曾教授在我碩士班兩年期間相當耐心的對我指導、與我討論；從他身上能學習到許多不只論文研究上甚至領導處事中的技巧，所有這些寶貴的經驗讓我獲益匪淺，感激不盡。同時也感謝我的口試委員，黃國禎教授、陳年興教授和胡毓志教授，他們給予了我相當多的寶貴意見，讓本篇論文更有價值。

第二位要感謝的人是林順傑學長，碩士期間讓我學會許多理論知識及研究方法，也盡心盡力與我一起討論並分享想法，兩年先後完成的會議論文、期刊論文、直到碩士論文，中間接受我的詢問和討論，並協助我論文修改工作，深表感激。

此外我必須感謝實驗室學長平日的諸多幫助，同時也感謝實驗室同窗夥伴，蔡昇翰、葉展彰、張晉璿、林永彧、林喚宇、任珍妮、施南極、羅仁杰等人在生活上和課業上互相幫忙的情誼；以及學妹陳曉涵、王芙民等人在論文實驗上的支援，深表感激。

最後我要感謝我的家人對於我的支持與鼓勵，讓我在面對挫折的時候能夠繼續向前，也讓我能夠有相當的自信完整本篇論文，深表感激。

Table of Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1 Introduction

Knowledge acquisition (KA) is a methodology of obtaining the knowledge of special domain from the domain expert, and knowledge based system (KBS) is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solutions, such as disease diagnosis, investment prediction, or computer science. Meanwhile, KA is one of the critical bottlenecks in developing a KBS for obtaining the knowledge. Traditionally, knowledge engineers retrieve knowledge from human experts by interviewing, and then a computer can be applied to perform the similar problem as human experts do in the real world. In order to facilitate the efficiency, knowledge engineers draw support from technical documents to interact with human experts; therefore, the idea of ontology helps knowledge base system overcome several bottlenecks such as knowledge representation, sharing and reusing. However, in the real world environment, there is not only static knowledge but also dynamic knowledge which is intensely fickle. With the changing environment as time goes on, new objects in many domains are incrementally evolved or developed due to the explosion of knowledge. It results in the creation of new knowledge due to the new evolved objects. Avoiding a downward efficiency spiral involves constantly hatching latest information while trying to defend an existing knowledge line. Hence, knowledge can be classified as static knowledge and dynamic knowledge according to the stability of knowledge in dynamic environment over time. The static knowledge remains the same in the changing environment as time goes on. Since the different environment can change over time, the original knowledge constructed at a time may degrade or upgrade in the near future. Therefore, the dynamic knowledge means that

the knowledge will be updated or evolved over time due to the adaptation of the changing environment. The knowledge evolution we proposed in this thesis is the iterative process to acquire evolutional knowledge in the changing environment.

Traditional KA methodologies, which are capable of acquiring static knowledge, can be classified into interviewing, machine learning and knowledge acquisition systems; however, the dynamic knowledge acquisition has hardly been discussed. Knowledge engineers directly retrieve domain knowledge by interviewing with human experts, and transform the knowledge into the computerized format to help experts solve difficult problems in the real world. In general, to acquire dynamic knowledge, the experts are required to be aware of the occurrence of new objects in the interviewing approach and knowledge acquisition systems. However, it is still difficult for experts to be aware of the new object without any additional related information. The machine learning approaches, which can learn the useful knowledge of static objects according to the selected training cases, are usually lack of the ability of discovering new objects without new cases of dynamic objects in the training process. As we know, many KA systems and tools such as *NeoETS* [5], *ACQUINS* [6], *KITTEN* [24], *EMCUD* [17], *KSSO* [14], have been proposed to rapidly build prototypes and improve the quality of the elicited static knowledge of well-known objects by domain experts with/without knowledge engineers in the past twenty years. However, most of them lack the ability of incrementally acquiring dynamic knowledge since the experts may not be aware of the occurrence of new objects without sufficient information.

EMCUD (Embedded Meaning Capturing and Uncertainty Deciding) [Hwang, 1991] was proposed to elicit the embedded meanings of knowledge (embedded rules bearing on m objects (O_1, O_2, …, O_m) and k object attributes) following repertory

grids principles, which represents the information that domain experts take for granted but are implicit to the people who are not familiar with the application domain, and guide experts to decide the certainty degree of each embedded rule for extending the coverage of the original rules generated by acquisition table. Since the relative importance of each attribute to each object could be represented as attribute ordering table (AOT) in *EMCUD*, some minor attributes can be relaxed or ignored to capture the embedded meanings with acceptable CF. Assume some objects in O_1 class, which are classified by original rules of O_1, belong to the original object class (OO_1) of O_1; the other objects in O_1 class, which are classified by embedded rules of O_1, belong to the extended object class (EO_1) of O_1. However, some embedded rules may be with marginally acceptable certainty factor (CF) values due to the weak suggestions of domain experts. In the age of the knowledge explosion, some objects might be evolved with the times and could be classified by the embedded rules of O_1 with weak CF values since some related ambiguous attributes (minor attributes) are ignored to classify these new evolved objects into O_1 class. Although *EMCUD* extends the ability of KA system to classify a new object into an original object class with the weak embedded rules with lower CF values, it still lacks the ability to incrementally discover the new objects and integrate the corresponding dynamic knowledge of new objects into original knowledge base without rerunning the whole KA procedure to generate the knowledge. Moreover, the human experts are unlikely to be aware of the occurrence of the new evolved objects due to the lack of sufficient relevant information about the new objects.

In this thesis, we will propose a new collaborative knowledge acquisition method (*CAKE* – Collaborative Acquisition for Knowledge Evolution) to collect useful evidence through monitoring the frequent inference behaviors of weak embedded

rules and tracing trend events of objects with time in order to assist experts to efficiently adapt the certainty factor of dynamic knowledge of new objects according to the sufficient context information. In order to discover the new object, the *VODKA* (Variant Object Discovering Knowledge Acquisition) is proposed to facilitate the acquisition of new knowledge of variant objects by monitoring the frequently fired weak embedded rules. Since the evidence of object evolution may appear diversely in unpredictable time, a time interval tracing oriented mechanism, Trend Event Acquisition (*TEA*) [22], for constructing dynamic knowledge of new objects is proposed to adapt knowledge to current time by recording each interested attribute's information in each time interval and update evolutional knowledge base. The *VODKA* generates a new acquisition table of new object, and the *TEA* generates a dynamic AOT table for capturing the evolutional embedded meaning of these objects. However, some dynamic knowledge might be invisible in each KBS with *VODKA* and *TEA*. Several heuristics are proposed to assist experts in discovering the new evolved objects, including service-sensitive and symptom-similar heuristics. The context information, including static profile and dynamic behaviors, is designed to assist experts to be aware of the occurrence of dynamic objects according to service-sensitive heuristic and symptom-similar heuristic by collecting sufficient information of multiple sensors. The service-sensitive heuristic can help experts to discover similar behaviors result in different symptoms due to different individual or environment configurations. The symptom-similar heuristic can assist experts to recognize different behaviors result in similar symptoms between multiple sensors due to the polymorphic configurations. We think the legacy knowledge acquisition methods are inefficient to acquire the evolutional knowledge because it is unable to learn the evidences of variation and development. Finally, we will propose the

Dynamic EMCUD (*D-EMCUD*) to integrate new dynamic knowledge into original knowledge base.

Based upon the *CAKE* framework, a worm detection prototype system with the CAKE module is implemented to evaluate the effectiveness of integrating the new evolved knowledge into original knowledge base. Based upon the collaborative framework, the dynamic knowledge of new evolved objects could be elicited to discover the new variant worms generated by the attacking traffic generator in the experimental environment.

Chapter 2 Related Work

Several knowledge acquisition methodologies and related systems are introduced in this chapter. Here we also introduce repertory grid, one of the popular indirect knowledge acquisition techniques. Finally, the elicitation of embedded meaning and some problems of traditional knowledge acquisition methodologies are discussed.

2.1 Knowledge Acquisition Review

In general, there are three approaches for knowledge acquisition (Mcgraw et al., 1989; Hwang and Tseng, 1990; Crowther and Hartnett, 1996):

(1) Interviewing experts by experienced knowledge engineers: interviewing experts 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.

Many researches like *ETS* [4], *AQUINAS* [6], *RuleCons* [23], *MOLE* [12] and *KSSO* [14] were developed to build rapid prototypes (knowledge acquisition systems) and to improve the quality of the elicited static knowledge. Most of them employed the repertory grid test originally developed by George Kelly [19], and had discovered

unexpected interesting results. The repertory grid could be used as efficient knowledge acquisition technique in identifying different objects and distinguishing these objects in a domain. A single repertory grid represented as a matrix whose columns have element objects (labels) and whose rows have construct 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 the 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 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. Also, several researches focused on building an intelligent expert system like *MYCIN* project [1], a well-known medical expert system for diagnosing infectious diseases, encouraged the advent of expert system studies. Moreover, ontologies and information sharing have a major role to play in the development of knowledge-based agents and the overcome of the knowledge acquisition bottleneck. Then, in 2003, an intelligent DNS management system [9], a web-based DNS expert system, was proposed to assist DNS administrators in managing their DNS by efficiently eliciting rules from experts or cases based on an efficient DNS ontology construction algorithm.

In summary, the knowledge acquisition bottleneck can be described as follows [26].

(1) Narrow bandwidth:

The existing channels converting organizational knowledge from its source (either experts or documents, or transactions) are relatively narrow. Knowledge engineers can

only focus on a few key applications, but not the bulk of all organizational knowledge. The uses of data and text mining are limited by cost and mining effort. End user experts are slow in capturing their own knowledge.

(2) Acquisition latency:

The slow speed of acquisition is frequently accompanied by a delay between the time when knowledge (or the underlying data) is created and when the acquired knowledge becomes available to be shared. This is especially a concern in dynamic environments where knowledge changes quickly and therefore the knowledge repositories always appear outdated. This challenge is related to the method of knowledge acquisition as well as the incentive systems, which often do not encourage experts to freely share their newest, most innovative, and most personal or tacit knowledge.

(3) Knowledge inaccuracy:

Experts make mistakes and so do data mining technologies. Furthermore, maintenance can introduce inaccuracies or inconsistencies into previously correct knowledge bases. With little available bandwidth to create new knowledge, even fewer resources are likely available to check the accuracy of knowledge already in the system. Furthermore, correction procedures can be difficult and cumbersome (Who is permitted to correct errors? What is the procedure? What incentives are there to report errors?).

(4) Maintenance Trap:

As the knowledge in the knowledge base grows, so does the requirement for maintenance. Furthermore, previous updates that were made with sufficient care and foresight (“hacks”) will accumulate and render future maintenance.

It appears there are few opportunities for breaking the knowledge acquisition bottleneck. Several possible remedies have been discussed, for example, the work of Boicu [7] described a practical approach, methodology and tool, for the development of static knowledge bases and agents by subject matter experts, with limited assistance from knowledge engineers. And the work of Wagner [26] described the use of collaborative, conversational knowledge management to demonstrate the opportunity for more effective knowledge acquisition. Obviously, the idea of collaboratively and systematically constructing the dynamic knowledge bases is launched. However, the dynamic environment limits all their efficiencies for the knowledge acquisition methodologies. The real world knowledge in a dynamic environment is often considered evolutional, because the knowledge can change or evolve over time due to the advent of information century. The speed changing in knowledge is too fast to manually accumulate the information by experts, and it usually results in the ritualistic batch process analysis which is always tough and time consuming.

Although many methodologies are proposed to extend the ability of uncertain reasoning to classify the objects, none of them discusses the issue of discovering dynamic knowledge of the new objects. It is also difficult for experts to notice the occurrence of new object, which is evolved with dynamic environment as time goes on.

2.2 EMCUD

The "*embedded meanings*" referred in this thesis represent the information that domain experts take for granted but are implicit to people who may not feel 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 and Weiss, 1987) and *SEEK2* (Ginsberg et al., 1988) have been proposed to obtain embedded meanings by some efficient refinement processes. *EMCUD* is proposed to elicit the embedded meanings of knowledge from the existing hierarchical repertory grids given by experts (Hwang and Tseng, 1990). *EMCUD*, a table-based knowledge acquisition method, is proposed to be able to elicit the embedded meanings of rules and guide experts to decide certainty factors from the existing hierarchical repertory grids. Additionally, it interacts with experts and guides them to decide the certainty degree with the embedded meaning. To capture the embedded meanings of the resulting grids, the Attribute-Ordering Table (AOT), recording the relative importance of each attribute to each object, is employed. There are three kinds of values in each AOT attribute/object entry: "X", "D" or an integer. "X" represents "don't care" which means that the attribute does not relate to the object, "D" represents "Dominate" which means that the object cannot be supported without the attribute, and an integer represents the relative degree of importance for the attributes to the object.

It is noticed that the integer ranges from 0 to 5, and 0 is treated as "X" while 5 is treated as "D". Besides the original rules, embedded rules can be generated by negating the minor (non-dominate) attributes recorded in AOT. Each embedded rule is assigned a certainty sequence (CS) by formula (1) and the certainty factor (CF) calculated by formula (2) which is between 0 and 1 can represent the degree of certainty for each embedded rule. Each of them is assigned a certainty factor (CF) between 0 and 1 while the value approaches to 1 means more important; otherwise, the value approaches to 0 means less important,

$$CS(R_i) = SUM(AOT[Att_k, Obj_i]) \quad (1)$$

where Att_k belongs to the attribute set of R_i, and Obj_i is the object of R_i.

$$CF(R_i) = UB(R_a) - \left(\frac{CS(R_i)}{MAX(CS_i)} \times (UB(R_a) - LB(R_a)) \right) \quad (2)$$

$MAX(CS_i)$ is the maximum CS value in all embedded rules generated from the original R_a with the same object. The *EMCUD* algorithm is listed in Algorithm 2.1: *EMCUD* algorithm.

Algorithm 2.1: EMCUD algorithm

Input: The hierarchical grids.

Output: The guiding rules with embedded meaning.

Step1: Build the corresponding AOT with each grid of the hierarchical multiple grids.

Step2: Generate the possible rules with embedded meaning.

Step3: Select the accepted rules with embedded meaning through the interaction with experts.

Step4: Generate automatically the CF of each rule with embedded meaning.

To decide the CF of each embedded rule, the upper and the lower bounds values are defined for accepted embedded rules. Then CF values of each rule can be automatically determined by mapping function, formula (2). Thus, the useful embedded rules with corresponding CF values could be used to cover more uncertainty cases.

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.

2.3 Variant Object Discovering Knowledge Acquisition

Although *EMCUD* successfully solves the problems of the conventional

repertory grids, including knowledge representation and embedded meaning, it might still exist several problems such as hard to explain the rules with lower CF value, difficulty in deciding attribute ordering, and infeasible to elicit the knowledge of a new variant object due to the knowledge explosion of the changing environment over time. Hence, *VODKA* [25] is proposed to help experts be aware of the occurrence the occurrence of the new objects by monitoring the inference behaviors of the weak embedded rules with the lower CF values and incrementally extract the knowledge of the new discovered objects to enhance the explanation power of the original embedded knowledge base.

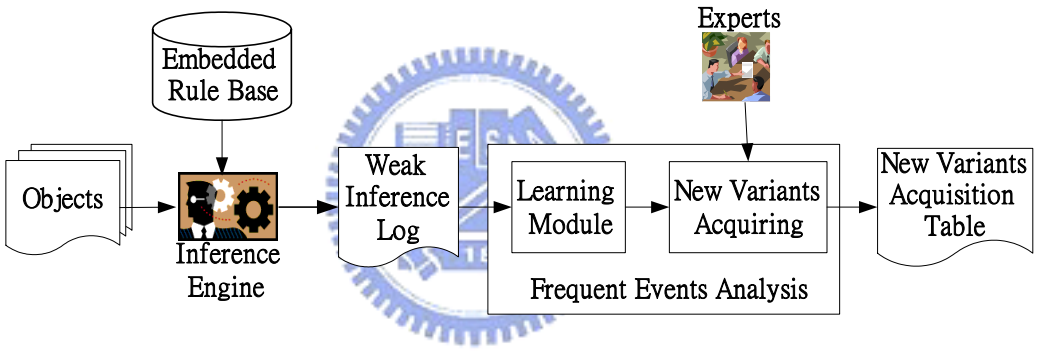


Figure 2.1: The Variant Object Discovering Knowledge Acquisition Flow

The novelty of *VODKA* approach, shown in Figure 2.1, collects the inference logs of weak embedded rules from KBS to learn the candidates of new evolved objects. Since the new evolved objects may derive from well-known objects, some of them can be classified into well-known object class with lower certainty factor according to the nature of embedded rules in *EMCUD*. The learning module can be customized according to the different applications. In this thesis, we use the frequent based heuristic in learning module to discover the frequent attribute-value pairs between collected inference logs since the new object could be fired frequently when it is evolved. Hence, the discovered frequent attribute-value pairs could be used to help

experts find out the new objects using the new variant acquiring module includes three recommendations. Furthermore, the input facts which are different from original acquisition table may be considered as the possible new attributes or attribute values of a variant object. Therefore, if the variants could be detected and recommended as the new objects by experts, the related ambiguous attributes (minor attributes) which might result in the marginally acceptable CF values of original rules suggested by experts could be refined or new attributes could be added to improve the classification ability. According to the complexity of relations between objects and attributes or even each relation between different tables, it is hard for experts to cooperate with each other in building every column and every row for each table. The acquired new objects can be used to construct the new variant acquisition table. The algorithm of *VODKA* [25] is shown as follows.

Algorithm 2.2: VODKA algorithm

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'*.
Step1: Discover large itemsets *L* using the inference log.
Step2: Generate *T'* using *L* and additional attributes provided by experts.
Stage III: Use *E-EMCUD* to generate rules of new variants.
Step1: Generate rules according to *T'*.
Step2: Merge *T'* into original main acquisition table *T*.

VODKA has been implemented based upon *DRAMA* [21], an object-oriented inference engine with *NORM* (*New Oriented-original Rule-based Model*) knowledge representation providing high maintainability, reusability, sharability, and abstraction for rule-based system, and the *E-EMCUD* (*Extended EMCUD*) has also been

implemented to refine the embedded rule base.

However, it is still hard to have a certain command of the evolutional knowledge, so we proposed CAKE to efficiently discover the evolutional knowledge by context information analysis and collaborative heuristics. On the other hand, the repertory grid-oriented method to construct acquisition table is somehow strenuous for an expert and even more strenuous to solve the adaptive problem in a dynamic environment; therefore, we proposed *TEA* to cope with the problems above by further enhancing the original *EMCUD* method to become *D-EMCUD* (*Dynamic EMCUD*) method [22].

Chapter 3 Context Representations

We use XML, a simplified dialect of SGML (Standard Generalized Markup Language), to model the behavior as a kind of context information for several reasons. First, XML-based applications and tools are widely used and developed, and it has been standard in World Wide Web Consortium (W3C) already. Seconds, software tools for processing XML documents are easily obtained since file of XML is also machine-readable. It provides a common descriptive specification framework that can be used to enhance the reusability of the software documents. The heart of an XML application is a file called the DTD (Document Type Definition), which describes the hierarchical structure of a class of documents. Note that DTD is an XML document on which we can reuse the content of the document.

3.1 Behavior Description Markup Language

In this thesis, a Behavior Description Markup Language (BDML) based upon XML protocol is proposed to provide a general model for expressing computer's network behavior. But, there are two categories of context information should be considered. First, static profile including environment configuration and individual configuration represents the location information and the basic information of the object. Second, dynamic behavior including individual trend and environment trend records the historical behaviors of the objects and objects' environment to further help experts analyze the variant trend information and evolutional behaviors.

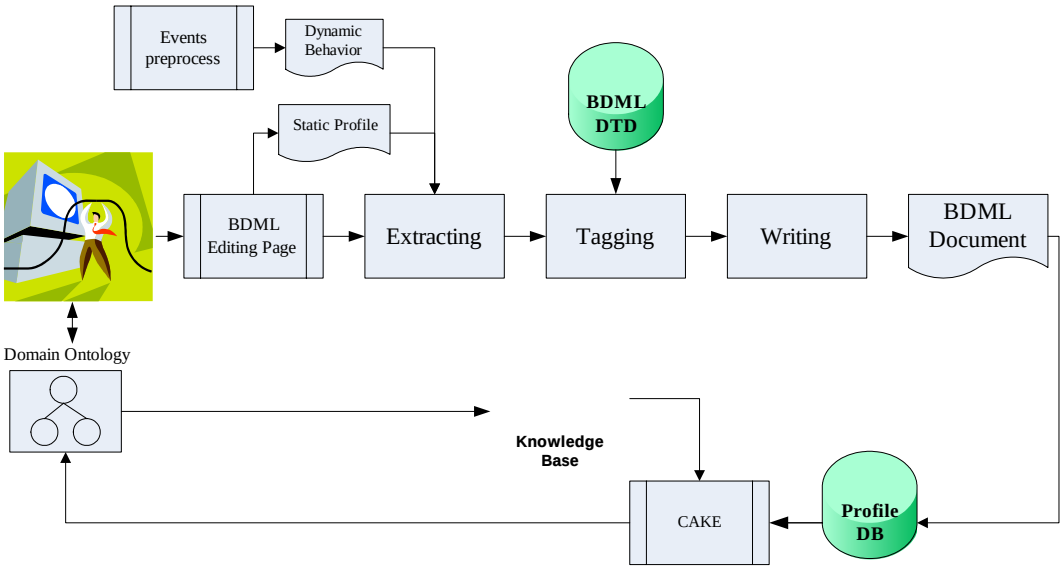


Figure 3.1: The Structure of BDML

It is noticed that the dynamic context information includes static profile, dynamic behaviors, the frequency, time based trend, and the correlations information between individual and environmental behaviors to assist experts to be aware of the occurrence of dynamic knowledge according to several collaborative heuristics for service-sensitive and symptom-similar events. The structure of BDML is shown in Figure 3.1. Since knowledge can be evolved with the dynamic environment as time goes on, how to acquire and represent the dynamic context information becomes an important issue. The context information can be classified into two categories: static profiles and dynamic behaviors information.

(1) Static Profile:

In the real world, the environment includes individuals, the relationships between individuals, and the related configurations. The environment could be considered as a collection of network properties and each individual has its own properties in the environment. Therefore, the static profile can be considered as environment configuration and individual configuration. The environment configurations describe the description of the environment, members in the environment, the status of the

environment, and other relative properties rely on domain. The individual configurations describe the individual ID, Location, Role of individual, and other relative properties depending on the domain. Through the static profile, we could classify the knowledge occurred in similar configuration.

(2) Dynamic Behaviors:

Some knowledge will be evolved to adapt the dynamic environment due to the natural of knowledge evolution. Owing to clear representing the behaviors of individual and environment, the trend of individual and environment can be easily acquired. The individual trend consists of the sequence of status of time and the occurrence of events pair. Also, the other relative properties should be also considered in each domain. Like the individual trend, the evolutional trend also combines the sequence of environment status and other properties to analyze the trend of environment for capturing evolutional knowledge.

Figure 3.2 shows the analysis of historical context information which is considered as dynamic. First it receives the information of uncertain variant frequency from *VODKA*, and then processes the trend analysis by *TEA*. Finally, the BDML can be constructed by combining all the static and dynamic components.

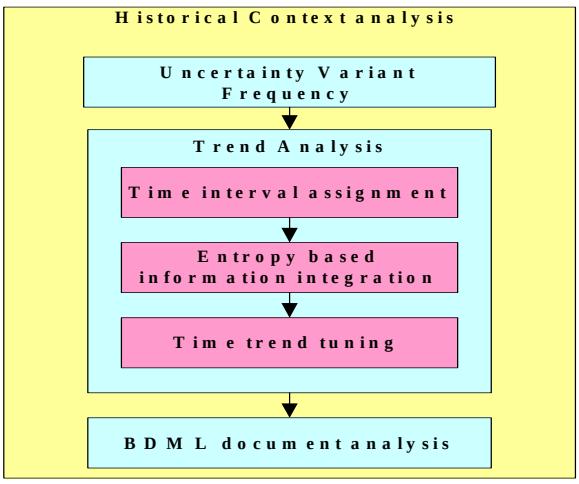


Figure 3.2: The Historical Context analysis

We design an XML based language that facilitates the machine readability for the KA framework to model the context information. In this model, not only the static profile but also the dynamic behaviors can be modeled using XML based description because the structure of XML is regular expression. Furthermore, the stored context can be easily reused, and the representation can be extended to describe new added properties of individual or environmental profile and behaviors due to the standardized property of XML.

Modeling the internet behavior by BDML, static profile DTD and dynamic behavior DTD can be shown in Table 3.1 and Table 3.2. The corresponding DTD of static profile in BDML includes environment configuration (network configuration) and individual configuration (host configuration). The environment configuration consists of description, members, status, and set of other properties, and the individual configuration consists of ID (IP address), location (LAN address), Roles (Server or Client), status, and a set of other properties.

Table 3.1: Static Profile DTD

```
<? Xml version="1.0" encoding="Big5">
<!DOCTYPE Static Profile [
  <!ELEMENT Static Profile (Environment configuration, Individual configuration)>
      <!ELEMENT Environment configuration (Description, Members, Status, Properties)>
      <!ELEMENT Description (#PCDATA)>
      <!ELEMENT Members (#PCDATA)>
      <!ELEMENT Status (#PCDATA)>
      <!ELEMENT Properties (#PCDATA)>
      <!ELEMENT Individual configuration (ID, Location, Roles, Status, Properties)>
        <!ELEMENT ID(#PCDATA)>
        <!ELEMENT Location(#PCDATA)>
        <!ELEMENT Roles(#PCDATA)>
        <!ELEMENT Status(#PCDATA)>
        <!ELEMENT Properties(#PCDATA)>
]>
```

Table 3.2: Dynamic Behavior DTD

```
<? Xml version="1.0" encoding="Big5">
<!DOCTYPE Dynamic Behavior [
   <!ELEMENT Dynamic Behavior (Individual trend, Environment trend)>
       <!ELEMENT Individual trend (#PCDATA)>
       <!ELEMENT Environment trend (#PCDATA)>
]>
```

The corresponding DTD of dynamic behavior in BDML includes individual trend and environment trend, including information of time, environment status, individual events and some set of properties. Taking computer worms as an example, the properties mentioned above consist of the basic information of the worms, the service what the worms aim at, the exploitation what the worms use, the carrier what the worms provide, the symptoms what the worms bring and the defense instructions to the worms.

3.2 Ontology Support

According to the complexity of relations between objects and attributes, it is hard for experts to cooperate with each other in building every column and every row for each table. Therefore, an ontology strategy is firstly designed in this thesis to help experts construct the knowledge in ontology and then the ontology will be transformed into the acquisition table. Therefore, we can easily define every relation between objects and attributes, and can easily assign a new attribute ordering value.

One of the purposes of applying ontology is to provide domain of discourse that is understandable by human and computers, so ontology can be represented by machine readable markup languages such as RDF. Moreover, the reusability of ontology has become increasingly important for developing intelligent systems. In

this thesis, we proposed an ontology based knowledge acquisition method that makes the ontology not only reusable but also adaptive to the current environment. The method of constructing the ontology is based upon a concept tree consisting of several prior knowledge including BDML model and real cases provided by knowledge engineers and domain experts to further help experts to construct the original *EMCUD* method as an initial prototype of knowledge.

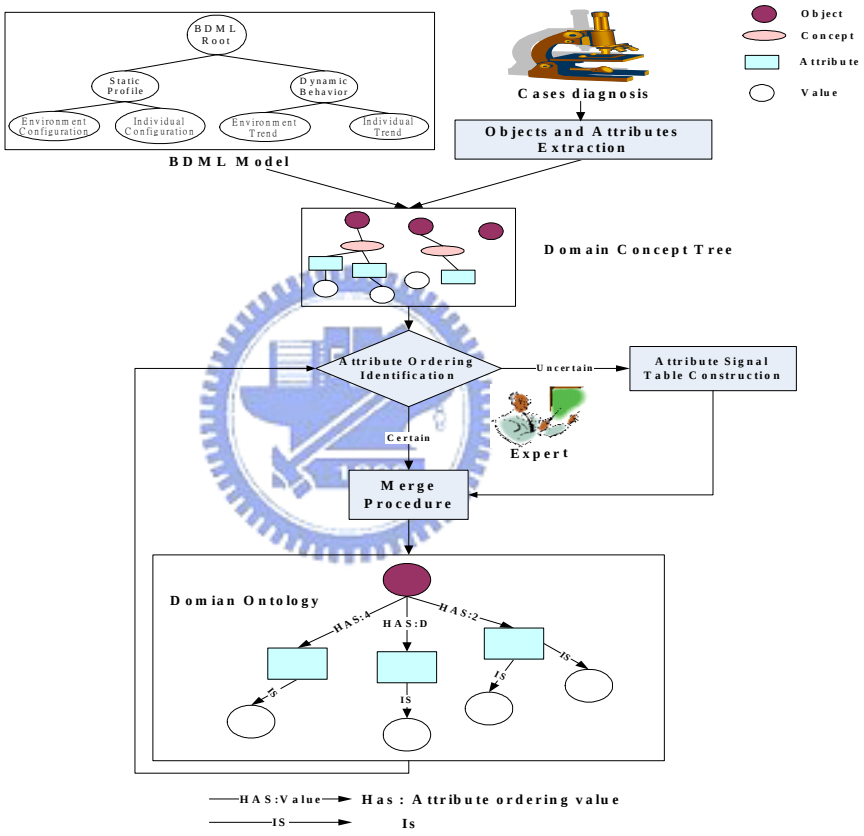


Figure 3.3: Ontology Construction Flow

Figure 3.3 illustrates the flow of constructing the domain ontology. To construct the ontology more easily, the following four steps are proposed:

Step 1, BDML model construction:

Create the BDML model according to the components, including static profile and

dynamic behavior. For example, we can model a worm BDML by identifying each worm with six general attributes, including the basic information and the service for static profile, then the exploitation, the carrier, the symptoms and the defense instruction for dynamic behavior.

Step 2, Concept tree construction:

Since it is often easier and more accurate for experts to provide critical cases rather than domain ontology, the power of critical cases described in terms of relevant objects and attributes to build domain ontology is remarkable. Therefore, after case diagnosing a concept tree is created based upon the skeletal model in Step 1. The example of CodeRed worm, Nimda worm and Blaster worm concept trees are shown in Figure 3.4, Figure 3.5 and Figure 3.6, respectively.

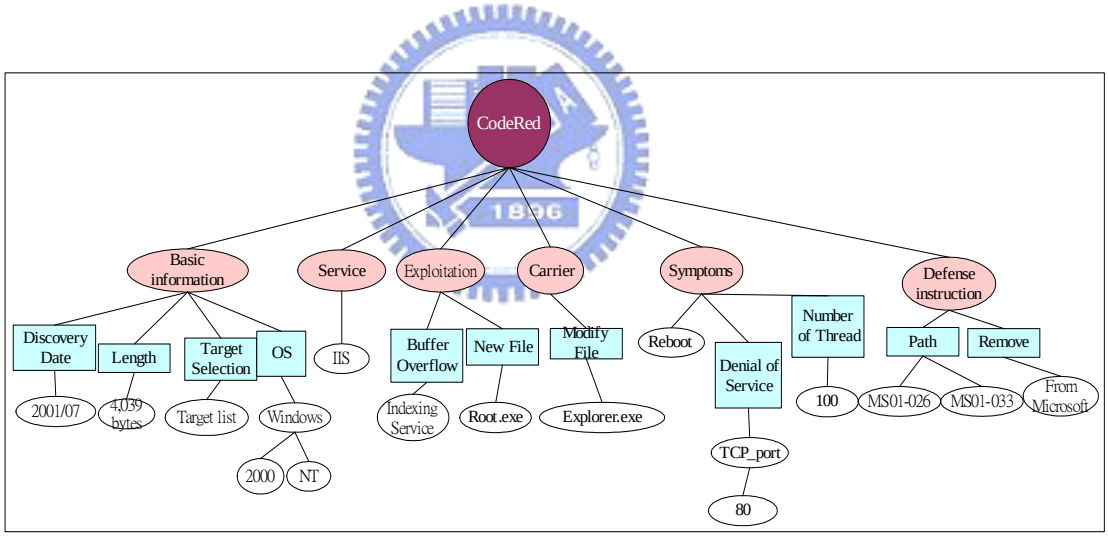


Figure 3.4: Example of CodeRed Concept Tree

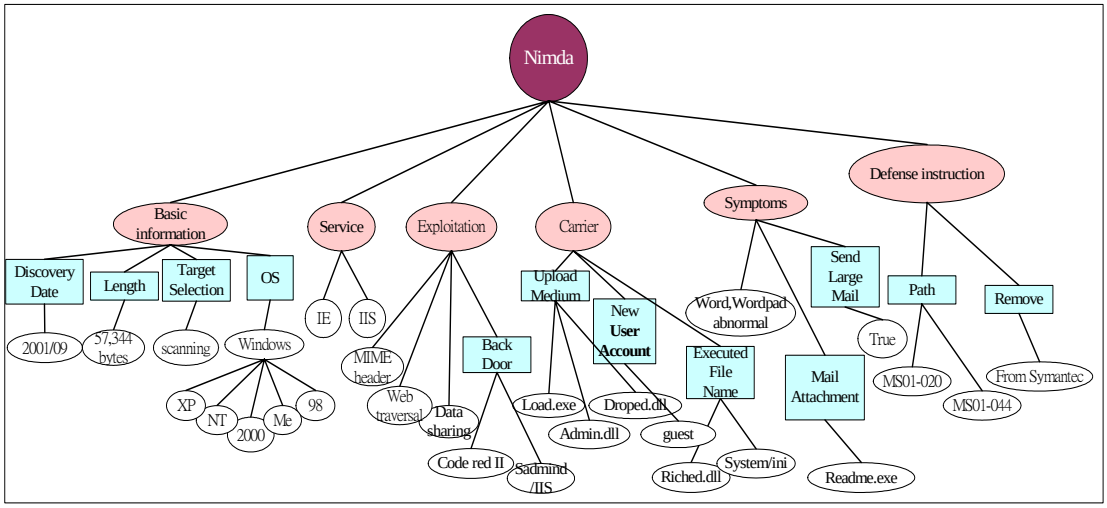


Figure 3.5: Example of Nimda Concept Tree

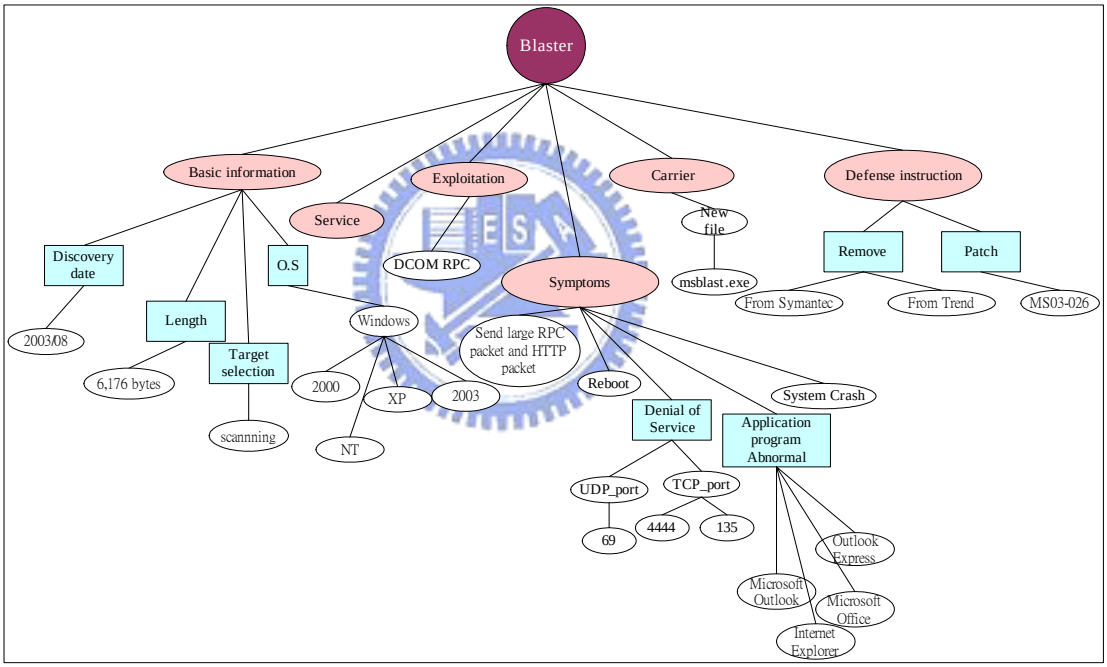


Figure 3.6: Example of Blaster Concept Tree

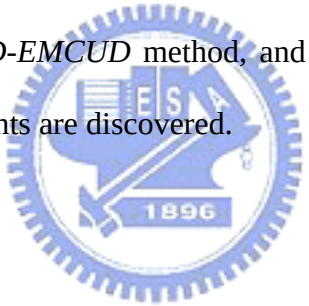
Step 3, Concept tree transformation:

The concept tree is transformed into acquisition table, and attribute ordering will be next acquired from experts. Then the original *EMCUD* method can be processed to generate the initial rules. However, because it is not easy to identify some attribute ordering values precisely, the attribute which is uncertain to identify the ordering

value should be traced and analyzed with time by constructing an AST. Each attribute signal is recorded in each time interval, when the attribute appears important the signal equals one and when it appears unimportant the signal equals zero. Then the attribute ordering table will be reconstructed according to the attribute signals collected with time.

Step 4, Merge procedure:

There are only two relations for constructing worm ontology during the merge procedure, including "*has*" relation and "*is*" relation. The "*has*" relation embeds attribute ordering value, for example, when the attribute ordering value equals 3 then the relation should be "HAS:3". Therefore, from Step 3, the ordering value would be retrieved from the reconstructed AOT by AST method. Hence, the ontology can be easily transformed into *D-EMCUD* method, and be updated whenever the AOT is reconstructed or the variants are discovered.



Chapter 4 The Framework of CAKE

4.1 The Collaborative Architecture

For the evolving environment, the important evidences could occur in every location and in every minute; therefore, deploying several kinds of specific inference engines to the significant environment is designed to cope with the problem. The collaborative architecture is shown in Figure 4.1. By collecting inference results from each inference engine, some results in the same period or in the sequential periods can further approve or disprove the evidence.

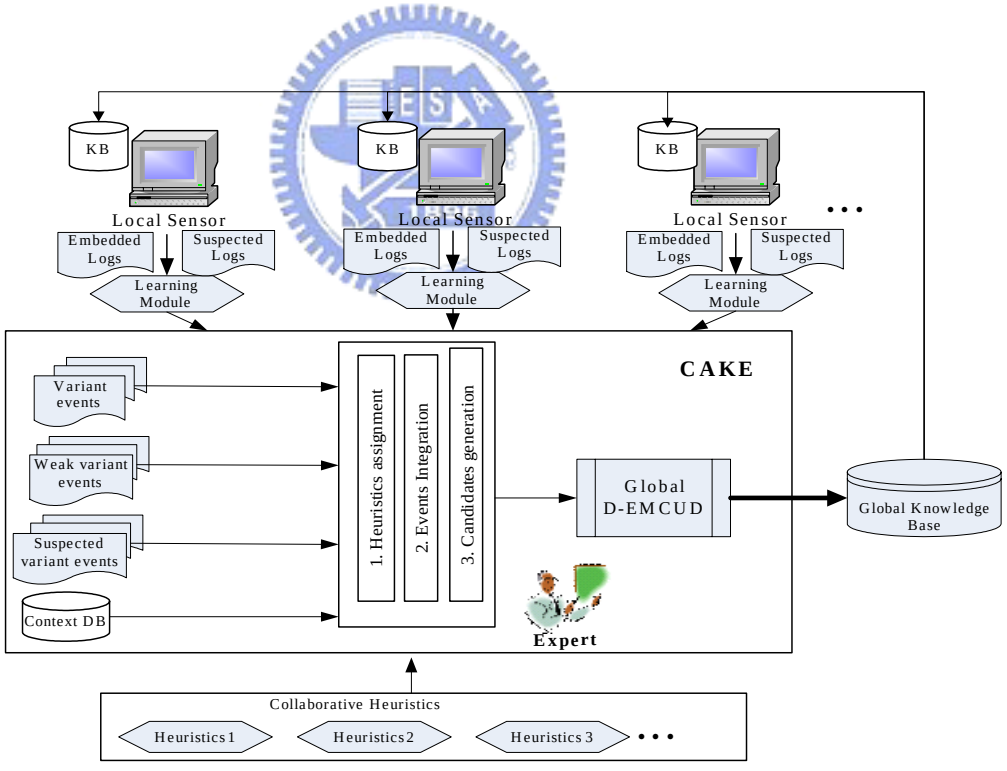


Figure 4.1: The Architecture of CAKE

For example, suppose there are smoke detectors and temperature detectors in a hotel. A simple meta rule can show that if there is a result of smoke form smoke

engine and the other result of sparkle form temperature engine then it then triggers the fire alarm service; however, if temperature engine sends back the result which is around the temperature of a human body then it disproves the evidence of fire and further approves that someone is smoking under the detectors. Obviously, the scenario described above can not be handled by single sensor environment. Hence, we use sensors with different configurations to represent such kind of expert systems or KBS with embedded rule bases. When the system is operating, some inference logs of cases, including old and new, will be recorded. By collecting and analyzing the relationships between these inference logs using the collaborative heuristics with sufficient context information, the occurrence of candidates of dynamic knowledge could be discovered. The *CAKE* consists of log collector, frequency analysis, trend analysis, collaborative analysis, dynamic *EMCUD*, and context DB to learn the dynamic behaviors and to record the configurations of different environment. The frequency-based analysis is proposed to discover significant variant knowledge by monitoring the inference behaviors of weak embedded rules. Moreover, the sequence of inference log will be considered to discover the relations between them to monitor the occurrence of evolutional knowledge as time goes on. All discovered information will be collected and further be analyzed by using collaborative analysis with consideration of the static configurations and dynamic behaviors, since some insignificant variant or evolutional knowledge might evade the frequency-based and time-based analysis. Finally, the dynamic *EMCUD* will incrementally integrate the discovered evolutional knowledge confirmed by experts, and then update the evolutional knowledge base.

In Figure 4.1, every local sensor collects logs from the inference engine based on the knowledge base (KB). Through the learning module, including components of

VODKA and *TEA*, can provide information of embedded logs and suspected logs. Embedded logs can be learned by *VODKA* to discover variants, and some of them which are weak to be identified by *VODKA* can be traced by *TEA* methodology to adjust the importance and further discover the variants. However, the logs considered suspected are logs without fired rules but are suspected of evolutional knowledge. When there is no matching rule after inferring through the inference engine, the log then contains no information of fired rule. The unexpected patterns were too uncertain to be accepted by experts at the beginning; however, they may evolve and become new knowledge as time goes on. Therefore, those suspected logs can be retrieved by context analysis based upon several collaborative heuristics and further the events of suspected variant can be identified via experts. The learning module is shown in Figure 4.2.

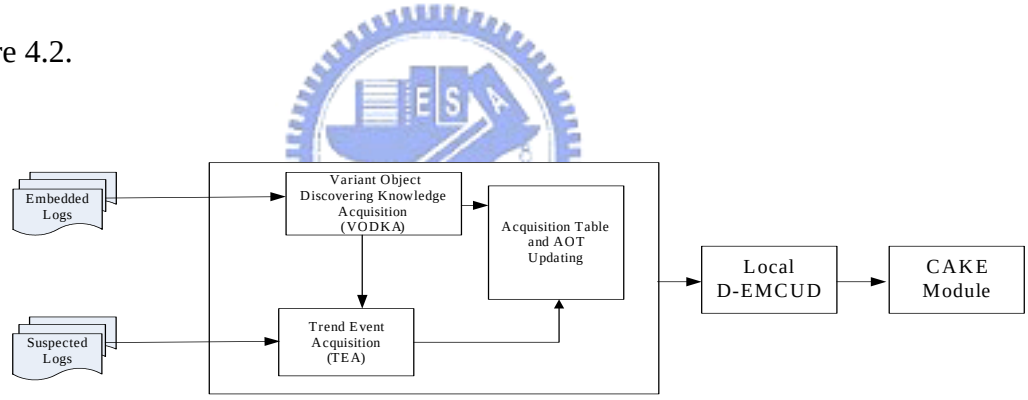


Figure 4.2: The Learning Module

We consider that the knowledge can be classified into static one and dynamic one. The static knowledge can be designed and built by following the static knowledge acquisition methodology such as *EMCUD*. But the dynamic knowledge could be variant or evolutional. For variant knowledge, it can be discovered by *VODKA* strategy when human experts themselves have enough abilities to decide the variant configuration; however, human experts have no abilities to discover the evolutional knowledge if without sufficient information, so in many situations, the evolutional

knowledge is negated easily when experts select the acceptable embedded rules. Hence, *BDML* based behavior context analysis is proposed to model ones' behavior according to their own environments. By BDML, the profile database can be constructed to support the collaborative analysis, and finally the experts can use *D-EMCUD* to construct or adjust the knowledge bases.

Taking Internet security as an example, there are many researches work on collaborative intrusion detection system (CIDS), many of them use lots of techniques such as the data mining technique to discover the intrusion; however they still have several limitations like false alarm reduction and inability to discover the variants. False alarms are too many to be handled due to lacking of environment context analysis. Different host computers may have different symptoms but suffered with the same attack. By the way, several attacks like computer worms may mutate in order to evade the detection mechanism of CIDS. Therefore, Figure 4.3 shows the example of collaborative architecture for CIDS, each of them has their own sensors and knowledge bases according to their profile configurations. The monitoring report from each one of sensors will be collected by the collaborative analysis center to further manage the security tasks in their network environment.

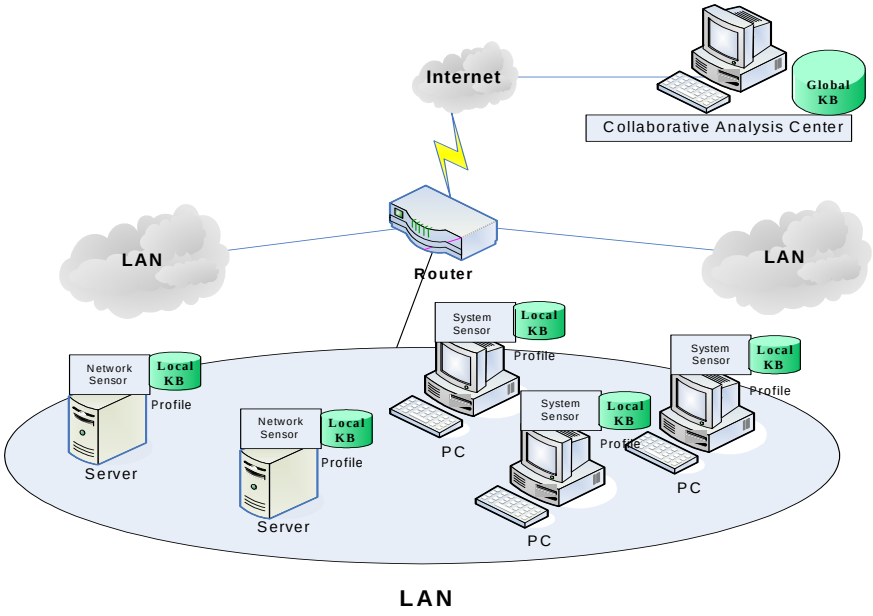


Figure 4.3: The Example of Collaborative Environment

In Internet, the environment configuration can be considered as network configuration while the individual configuration can be considered as host configuration. Hence, the network configuration can involve information like the network description, the number of members, the status information and other properties. Also, the host configuration can then involve the unique ID such as IP address, network location, the role it plays as server or client, the status information and also the other properties. For the dynamic behavior, the individual trend is retrieved from the *TEA* analysis drawn support from *VODKA*, and for the environment trend, the context information modeled by BDML is additionally retrieved to support the collaborative analysis.

4.2 Trend Capturing and Analysis

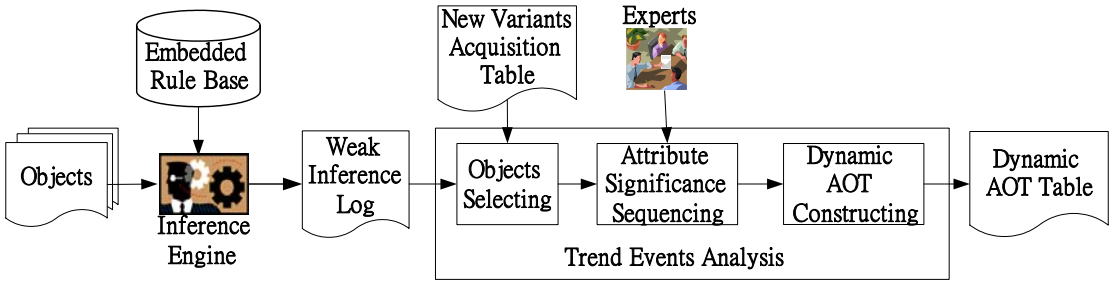


Figure 4.4: The Flow of *TEA*

The flow of Trend Event Acquisition (*TEA*) is shown in Figure 4.4. As mentioned before, *VODKA* [25] has been proposed to help experts to discover the knowledge of variant object by monitoring the inference behaviors but it lacks the ability of discovering evolutional knowledge over time. Although the original idea of constructing AOT table makes it more adaptive to elicit embedded meanings, it may not be so sure to exactly define each relative importance of each attribute to each object since the dynamic knowledge may change or evolve over time. It means that some rules, which are recommended by experts now, may become uncertain in the near future. Moreover, *VODKA* learns the behaviors of weak embedded rules with low CF values, and if the CF value is not adaptive in the past few days then it may encounter the problem of inefficiency in learning the variant object. In this thesis, decomposing each object in the AOT to record the relative importance of each attribute to the object into several time intervals is proposed. Each time interval represents in a short period according to the viewpoints of experts or the learning results of *VODKA*. It can simply assign “0” or “1” for each attribute to each object in each time interval, where “0” represents the attribute is considered as the unimportant attribute to the object and “1” represents the attribute is important to the object in this

time interval.

The signals of “0” and “1” are called the attribute signals in each time interval, and a table recording the information is called the attribute signal table (AST). There are two steps, including unfolding and reconstructing to help experts trace the evolutional behaviors. In order to capture the evolutional information of each object, the original AOT table will be decomposed into several AST tables to record the specific information in each interval with time in the first step shown in Figure 4.5. Then, we will reconstruct the AOT table by renewing the relative importance of each attribute to each object (ordering values) according to the sequence of “0” and “1” signal recording with time in the second step shown in Figure 4.6. Since the ordering values are recalculated at the present time according to all the evidences collected with time, the AOT table becomes more flexible and robust to adapt the changing environment.

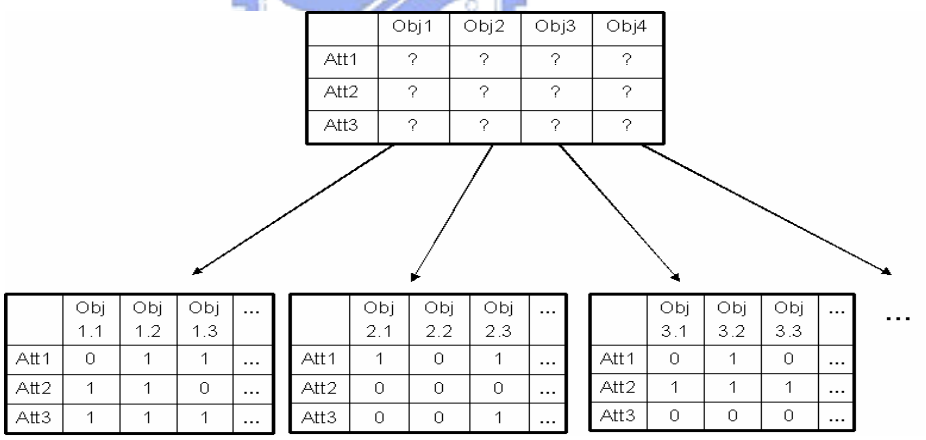


Figure 4.5: Unfolding

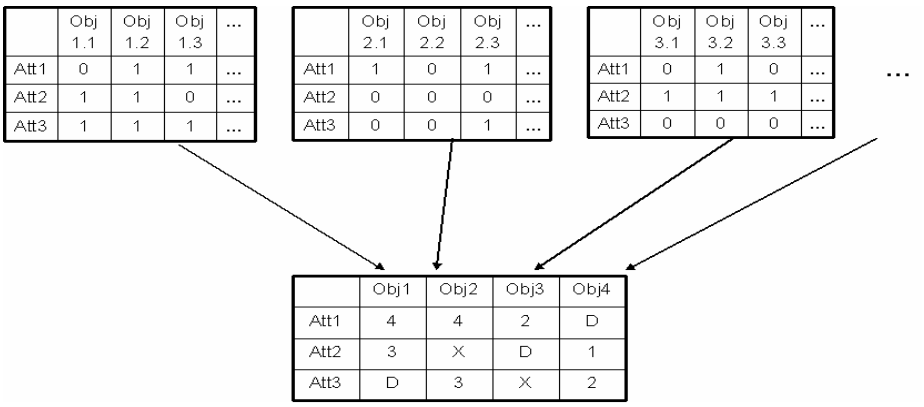


Figure 4.6: Reconstructing

4.2.1 Constructing Attribute Signal Table

In the changing environment, the knowledge will evolve over time. The relative importance of each attribute to each object may need to be adjusted. Since not all the attributes are working well after a period of time because of some characteristics of the attribute. The domain expert can decide which attributes are required to be traced with time if ordering values of the ambiguous attributes are hard to be decided immediately. There are two ways of constructing AST:

(1) Interacted with human experts:

It is designed to acquire the attribute signals from domain experts in every time interval for deciding whether the attribute is important or not. It is simpler to decide whether it is “1” or “0” rather than to decide a relative importance of each attribute to each object, the ordering value ranged from 0 to 5.

(2) Interacted with *VODKA*:

As mentioned above, *VODKA* can be used to learn and level up the CF of each weak embedded rule by monitoring their inference behaviors. It can be helpful to decide the attribute signal of importance or unimportance by directly mapping each embedded rule to the AST.

For example, assume *VODKA* generated after learning the candidates of variant and then confirmed by experts, is shown as follows:

R_1: IF A and B and not C Then Goal *2

R_2: IF A and not B and C Then Goal *3

where *2 represents that the rule R_1 has been fired two times, and R_2 is inferred after R_1. The AST can be constructed as Table 4.1 according to the inference logs collected in the five time intervals.

Table 4.1: An example of AST table

Attribute/ Object	Goal.1	Goal.2	Goal.3	Goal.4	Goal.5
A	1	1	1	1	1
B	1	1	0	0	0
C	0	0	1	1	1

Since the attribute B is not negated in R_1 according to the first two inference results, but it is negated three times after in R_2, the signal would be collected as "11000" shown in the second row of Table 4.1. Finally the scoring method is presented in the following subsection to help experts automatically renew the ordering value in the AOT table to enhance the dynamic ability of AOT table.

4.2.2 Constructing Attribute Ordering Table

To capture the adaptively relative importance of each attribute to each object from the AST table after a time period, the dynamic value adjusting method based upon entropy formula (3) [16] is applied to transform AST to AOT instead of simple leveling up the attribute's ordering value when the attribute's signal is "1" or leveling down when the attribute's signal is "0" ,

$$Entropy = -\frac{pos}{pos+neg}\log_2\left(\frac{pos}{pos+neg}\right) - \frac{neg}{pos+neg}\log_2\left(\frac{neg}{pos+neg}\right) \quad (3)$$

where “pos” represents positive case and “neg” represents negative case. In this case, the positive case means that the attribute is decided as an important attribute to this object such as signal “1” in AST table, and the negative value occurred in AST table can not be an important attribute in this time interval such as signal “0”.

As you move from perfect balance or perfect homogeneity, entropy function could be smoother between zero and one. The entropy is zero when the set of collecting instances is perfectly homogeneous like all positive or negative instances, and the entropy is one when the set is perfectly heterogeneous like half positive and half negative instances. Hence, it is obvious to decide the new adjusting value of the attribute in dynamic AOT table which should be set into a dominated attribute (D in AOT) or irrelevant attribute (X in AOT) to the object depending on number of positive and negative instances in this set when the entropy value approximates to zero. In other words, if an attribute is continuous important (unimportant) to an object in the collecting time period, then it could be a dominated (irrelevant) attribute to the object in this time moment. The strategy of dynamic *EMCUD* is shown as follows:

Step 1, Summing:

Add up each number of positives and negatives, and then apply entropy formula (3) to calculate the entropy weight of each attribute to each object.

Step 2, **Mapping:**

Transform the obtained entropy weight into relative importance of each attribute to each object according to the following three cases:

Case 1:

If the entropy weight approximates to zero and the number of positive instances is larger than that of negative instances, the ordering value of the

attribute to the object could be assigned into “D” or “4” in the dynamic AOT table.

Case 2:

In case 1, if the number of negative instances is larger than that of positive instances, the ordering value of the attribute to the object could be assigned into “X” or “1” in the dynamic AOT table.

Case 3:

If the entropy weight approximates to one, the ordering value of the attribute to the object could be set into “2” or “3” in the dynamic AOT table.

Step 3, Tuning:

If the attribute is time relevant, the trend of the attribute should be applied to adapt the obtained entropy weight according to the following three cases.

Case 1: Increasing trend.

The ordering value of this kind of attribute should be increased since the attribute seems to become more and more important in the collected period such as the sequence “001111”.

Case 2: Decreasing trend.

The ordering value of this kind of attribute should be degraded since the attribute seems to become less and less important in the collected period such as the sequence of “110000”.

Case 3: Uncertain trend.

The ordering values of this kind of attribute would not be tuned since this kind of attribute has no significant trend in the collected period such as the sequence of “010101” or “101010”.

In Step 1, it is very simple by summing up each number of “1” and “0” in AST

table for comparing the numbers of positive and negative instances. The entropy weight will be calculated to represent the distribution of “1” and “0” in each set in Step 2. When the number of positives surpasses the number of negatives, the entropy weight approximates zero and vice versa. It is obvious to assign a higher ordering value (D or 4) when most of information represent important; otherwise, a lower ordering value (X or 1) would be assigned due to the observation of unimportant attribute. However, other relative importance of each attribute to each object would be assigned when the entropy weight approximates one, which would be considered uncertain degree. That is, it is usually uncertain to decide a decision when getting half positive advices and half negative advices at a time; moreover, in Step 3, when the set is heterogeneous, but the characteristic of attribute is time relevant then it can be considered homogenous in some ways.

For example, suppose there are two sets in AST recorded as “000111” and “111000”. After Step 1, both of “000111” and “111000” in entropy method are considered uncertain because of perfectly heterogeneous. Since these two sets can be considered time relevant, it is very certain that attribute with signals “000111” should be assigned a higher ordering since it is important in the present time intervals. Therefore, after calculating with entropy method in Step 2, there is still a time bonus weighting in Step 3 to adjust each ordering value at final. Accordingly, the AOT is considered dynamic by following the time interval tracing oriented mechanism, and each CF value of a rule can now be leveled up or down automatically.

The mechanism not only applies *VODKA* to construct AST but also offers more robust information for *VODKA* to learn the variant objects since each CF value is also updated by the dynamic AOT.

4.3 Collaborative Heuristics

Furthermore, according to the specific environment for a particular domain, heuristic approach makes the collaborative analysis workable. Since some invisible or unrecognizable behaviors might be ignored without sufficient information, collaborative multiple sensors to collect more evidence of evolutional knowledge becomes more important. The static profile and dynamic behaviors of individual and environment could be used to help discover evolutional knowledge since they could assist experts to trace the changing behaviors. The static profile can be used to analyze the co-occurrence behaviors between different profiles since some behaviors might exist in similar environment. The dynamic behavior can be used to analyze the similar behaviors in different environment as time goes on. Based upon the collection of the sufficient context information including the unrecognizable or invisible behaviors in single sensor to discover the evolutional knowledge, the static profile and dynamic behaviors of individual and environment could be used to help discover evolutional knowledge. We propose two simple heuristics as followings based upon three distributed inference engines, including Service-Exploitation Engine, Carrier Engine, and Symptoms Engine.

(1) Service-sensitive event:

Different configuration may encounter with different symptoms or different degree of damage in each progress. When several Service-Exploitation Engines responds higher frequency of similar intrusion behaviors but with different behavior presentations from the corresponding Symptoms Engines, *CAKE* then triggers the trend analysis mechanism on the Symptom Knowledge Class by the aid of BDML. Consequently, by the trend from BDML we may learn that it is because the firewall

setting of some computer hosts are different from others. The central administrator can thus forward the better firewall settings to those who do not have the proper settings; furthermore, we may stumble upon a solution that the intrusion may be limited to some special configuration setting. For example, some intrusion behavior such as SQL Slammer which uses UDP port 1434 to exploit a buffer overflow in MS SQL server, then it can simply switch off this port since *CAKE* learns that others SQL servers which are unavailable to the Internet are immune.

(2) Symptom-similar event:

However, some computers come up with similar symptoms but with sort of different behavior from the Carrier sensor or even the Service-Exploitation sensor. We consider it is kind of polymorphic process for the intrusion like polymorphic worm which is considered as the variant knowledge or the evolutional knowledge in this thesis.

For heuristics above, the knowledge which is variant or evolutional, *CAKE* triggers the specific trend analysis process on special knowledge class like Carrier Knowledge Class or Service-Exploitation Class for symptom-similar events and Symptom Knowledge Class for the those events which are service-sensitive respectively. For example, D-EMCUD generates rules for Blaster and Sasser from acquisition table and AOT shown in Table 4.2 and Table 4.3. VODKA discovers variants from Log_1 and Log_2 in Table 4.4 since we suppose expert define CS value as 4 to be a threshold. According to the embedded rule represented as "*IF* A_1 *and* A_2 *and* $NotA_3$", VODKA confirms the variants named as Blaster.B and Sasser.D. However, Log_3, a suspected log in Table 4.4, is negated since both of attribute A_2 and A_3 are negative; therefore, the CS value is 7 (3+4) which is higher then the defined threshold. Unfortunately, it is the variant information of Sasser.E; hence, to enhance the

weakness, CAKE is applied and discovers the variant by Service-sensitive heuristic to find out the evolutional information such as 1022 for port number and ftplog.txt for carrier. By applying the heuristic, all information of suspected logs will be integrated by intersection. Then those suspected logs with most frequency (suspected events) will be further compared by BDML documents. Subsequently, the candidates of evolutional knowledge will be generated and further confirmed by experts.

Table 4.2: A simple acquisition table for Blaster and Sasser

Object / Attribute	Blaster	Sasser
Symptom (A_1)	System Reboot; System Crash; Word, Excel, Power point abnormal	System Reboot; System Crash; 128-threads
Service & Exploitation (A_2)	RPC Buffer Overflow (135;4444)	RPC Buffer Overflow (445;5554;9996)
Carrier (A_3)	Msblast.exe	avserve.exe; win2.log

Table 4.3: A simple AOT for Blaster and Sasser

	Blaster	Sasser
A_1	4	4
A_2	4	4
A_3	4	3

Table 4.4: Part of inference logs record

Logs #	IF Part			Then Part	CF
	A_1	A_2	A_3	Rule	
Log_1	System Reboot	Port:4444	penis32.exe	$R_{Blaster}$	0.6
Log_2	System Crash	Port:445	skynetave.exe	R_{Sasser}	0.74
Log_3	System Crash	Port:1022	ftplog.txt	-	-

Chapter 5 Case Study and Experiment

5.1 Brief of Worms

In order to understand the threat posed by computer worms, Figure 5.1 shows the life cycle of worms [13] consists of five phases. In *Initial Phase*, each worm agent begins with installing software, determining the configuration of the local machine, instantiating global variables and beginning the main worm process. To find vulnerable candidates in *Target Selection Phase*, a worm usually first discovers that the other potential victim exists. Then in *Network Reconnaissance Phase*, a worm can actively spread itself among machines or can be carried along as part of normal communication traffic through the network. Next in *Infect Phase*, system which is attacked may result in many abnormal behaviors in *Attack Phase* and may further infect other victims.

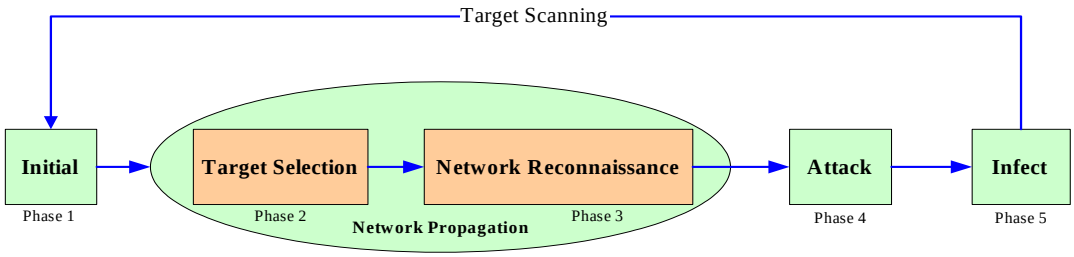


Figure 5.1: Worm Life Cycle

Since most of the worms follow the life cycle, we can construct the semi-structured document represented as a worm concept tree by interaction between domain experts and knowledge engineers with support of BDML model, then the meaningful information can be extracted according to the worms' life cycle, including

basic information of the worms, the service what the worms aim at, the exploitation what the worms use, the carrier what the worms provide, the symptoms what the worms bring and the defense instructions to the worms.

There are many companies or associations working on the research of network security such as Symantec [30], F-Secure [29], and Trend [31]. Symantec, the world leader in providing solutions to help individuals and enterprises assure the security, availability, and integrity of their information, provides Norton Antivirus, a popular software world wide, to scan the status of the host system and offer the solutions to the security problems. However, it needs to be updated by downloading the latest virus definition every period of time because of lacking self learning mechanism. Additionally, users spend lots of time finding the meaningful solution through the search engine when the report after scanning is not clear. Zero-day worms which are resulted from variants for most of the time are a serious wide-scale threat due to the monoculture problem. If there exits a standard signature-based detector which is blind to a zero-day attack, we can say that all installations of it are also blind to the same attack. The work of PAYL [27] anomalous payload detection sensor are demonstrated to accurately detect and generate signatures for zero-day worms; however, it can only be aware of the specific pattern that correlates ingress/egress payload alerts to identify the worm's initial propagation. Therefore, WISE [28] helps overcome these drawbacks by collecting all the meaningful information through *EMCUD* and then further discover more varieties of variants by *VODKA*. In advance, we make WISE integrate the BDML based technical documents of worms and real cases from experts into a tree-like structure, which is called concept tree, by knowledge engineers and domain experts. Subsequently, more meaningful information will be merged to construct the ontology in Figure 3.3 and also the learning methodology is applied to

discover the variant worms based upon the knowledge acquisition method. Finally, WISE provides tutorial guidelines to teach users to solve the problems with their own antivirus products or other solutions from the Internet. Hence, the WISE strengthens the power of general antivirus products and makes them more flexible based upon knowledge bases. However, large numbers of replicated vulnerable systems allow wide-spread infection that challenges the efficiency of WISE. We then employ WISE based upon the architecture of *CAKE* and discover more interesting and surprising knowledge to overcome WISE its' original disadvantage of signal host-based knowledge acquisition. The differences of WISE between with and without *CAKE* can be seen in the experiments in Section 5.4.

5.2 Example of Computer Worms Detection

Nimda, an incredibly sophisticated worm that made headlines worldwide, is taken as an example. Nimda is the first worm to modify existing web sites to start offering infected files for download by using Unicode exploit to infect IIS web server. It is the first worm to use normal end user machines to scan the vulnerable web sites. This technique enables Nimda to easily infect intranet web sites located behind firewalls. According to the dynamic behavior such as symptoms and carrier from BDML of worm in Figure 3.3, we may suppose after case diagnosis the Nimda concept tree is created in Figure 5.2 and can be transformed into acquisition table like Table 5.1. The three attributes, including the name of the e-mail attachment used by worms, the medium used by worms to upload themselves (the worm body), and the name of the file used by worms to start execution on servers, are used to recognize Nimda in this example. After constructing the acquisition table, the AOT table is needed to record the relative importance of each attribute to the Nimda object. Suppose Nimda is the

latest worm occurred in the recent cyber world, a worm domain expert might not have strong confidence to decide the relative importance of each attribute because the variants of Nimda may evolve quickly and the original ordering value needs to be modified in a short period. Hence, the expert can easily identify the importance of each attribute to Nimda1 and Nimda2 in each time interval according to the AST table with two time intervals, and then assign the first interval a signal which equals one in Table 5.2. Because the second time interval is pre-specified, the initial signal value is zero. Next the entropy weight of each attribute is first calculated to obtain the ordering value according to the AST. Since it is time irrelevant during the initial step of construction, the ordering value is “2” after calculation while entropy equals one because there is a positive instance and a negative instance. Therefore the initial AOT is constructed as shown in Table 5.3.

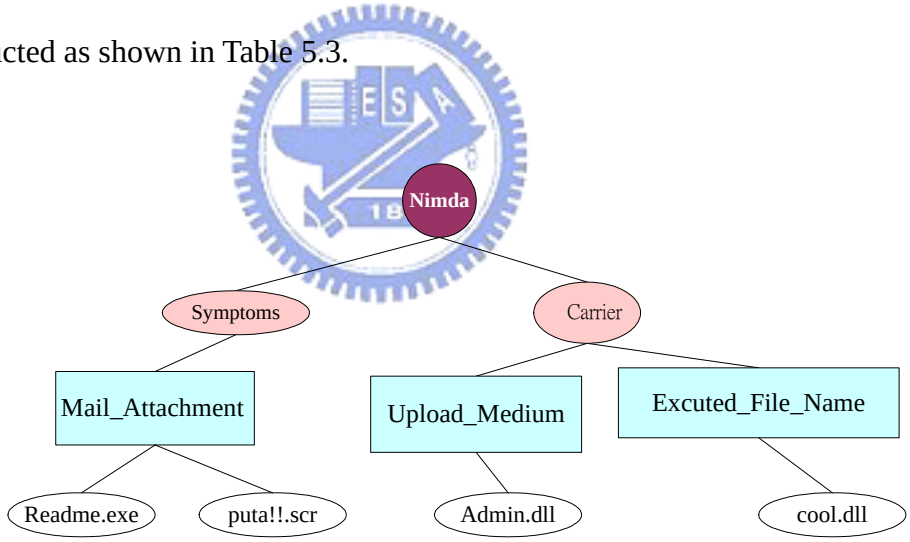


Figure 5.2: Example of Nimda Concept Tree

Table 5.1: An Example of Nimda acquisition table

Attribute/ Object	Nimda
Mail_Attachment	Readme.exe
Upload_Medium	Admin.dll
Executed_File_Name	Riched20.dll

Table 5.2: An Example of Nimda AST

Attribute/ Object	Nimda1	Nimda2
Mail_Attachment	1	0
Upload_Medium	1	0
Executed_File_Name	1	0

Table 5.3: An Example of Nimda AOT

Attribute/ Object	Nimda
Mail_Attachment	2
Upload_Medium	2
Executed_File_Name	2

Having both acquisition table and AOT, eight embedded rules are generated using *EMCUD* and some of them have low CF such as rule R_1: 'IF Not Mail_Attachment = Readme.exe and Upload_Medium = Admin.dll and Executed_File_Name = Riched20.dll Then Nimda' with CF = 0.67. Therefore, suppose that the constructing worm KBS receives several worm instances in the real world, the R_1 rule above is always fired by *VODKA* during a period, and suppose in the last two intervals the embedded rule R_2: 'IF Not Mail_ Attachment = Readme.exe and Not Upload_Medium = Admin.dll and Not Executed_File_ Name = Riched20.dll Then Nimda' with CF = 0.4 is fired by *VODKA*, the AST to record the evolutional trend is shown in Table 5.4.

In Table 5.4, the Mail_Attachment attribute is calculated by entropy-based method, and the entropy weight may lower the value because of its' decreasing trend in interval 6 and 7. Finally, the attribute is assigned a new ordering value equaling 1 since it is very possible to change again over time, subsequently, ordering value 3 is assigned for both attributes Upload_Medium and Excuted_File_Nam according to the AST. Therefore, the rule which has CF value equals 0.67 could be leveled up to 0.74.

Moreover, several new attributes' values are learned by *VODKA* with Mail_Attachment = puta!!.scr in R_1 ,then a new variant Nimda.B can be found in Table 5.5 and can be integrated into Table 5.6, and also an AOT is updated as Table 5.7, then after merge procedure, the ontology is constructed as shown in Figure 5.3.

Table 5.4: An Example of Nimda AST

Attribute/ Object	N1	N2	N3	N4	N5	N6	N7
Mail_Attachment	1	0	0	1	0	0	0
Upload_Medium	1	1	1	1	0	0	0
Executed_File_Name	1	1	1	0	1	0	0

Table 5.5: An Example of Nimda acquisition table

Attribute/ Object	Nimda.A	Nimda.B
Mail_Attachment	Readme.exe	puta!!.scr
Upload_Medium	Admin.dll	cool.dll
Executed_File_Name	Riched20.dll	httpodbc.dll

Table 5.6: An Example of Nimda acquisition table

Attribute/ Object	Nimda
Mail_Attachment	{Readme.exe; puta!!.scr }
Upload_Medium	{Admin.dll; cool.dll}
Executed_File_Name	{Riched20.dll; httpodbc.dll }

Table 5.7: An Example of Nimda AOT

Attribute/ Object	Nimda
Mail_Attachment	1
Upload_Medium	3
Executed_File_Name	3

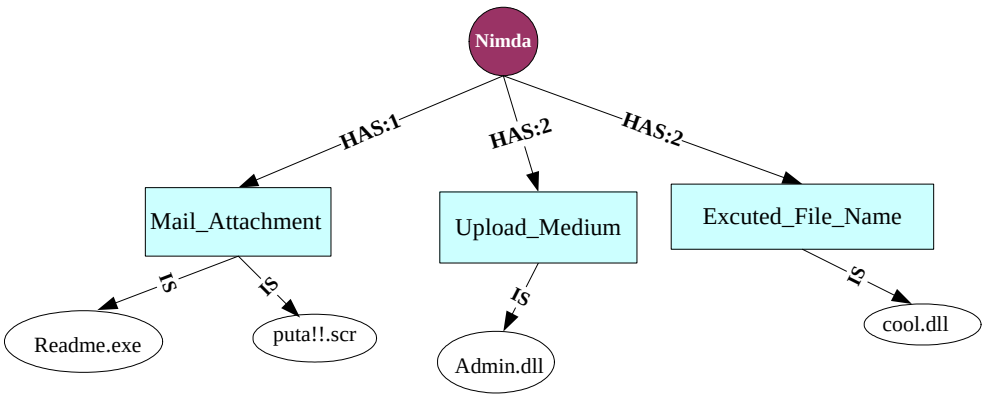


Figure 5.3: Example of Nimda Ontology

With the accumulated inference logs, *VODKA* can learn variants, and the time interval tracing oriented mechanism can be also used to incrementally update the knowledge to adapt the changing environment over time. Suppose *VODKA* learns a new attribute values, including Mail_Attachment = sample.exe, Upload_Medium = cool.dll and Executed_File_Name = httpodbc.dll in R_2 while the rule R_2 has already been fired in each time interval in a short period, another variant Nimda.E could be found. Finally the updated tables are shown in Table 5.8 and Table 5.9 and meanwhile, the ontology is also updated as Figure 5.4. Finally, the system will be set up in the current environment and guide the people who are not familiar in the domain by giving them a picture of worms for helping further preventing or removing the malicious worms.

Table 5.8: An example of Nimda acquisition table

Attribute/ Object	Nimda
Mail_Attachment	{Readme.exe; puta!!.scr; sample.exe}
Upload_Medium	{Admin.dll; cool.dll}
Executed_File_Name	{Riched20.dll; httpodbc.dll }

Table 5.9: An example of Nimda AOT

Attribute/ Object	Nimda
Mail_Attachment	1
Upload_Medium	2
Executed_File_Name	2

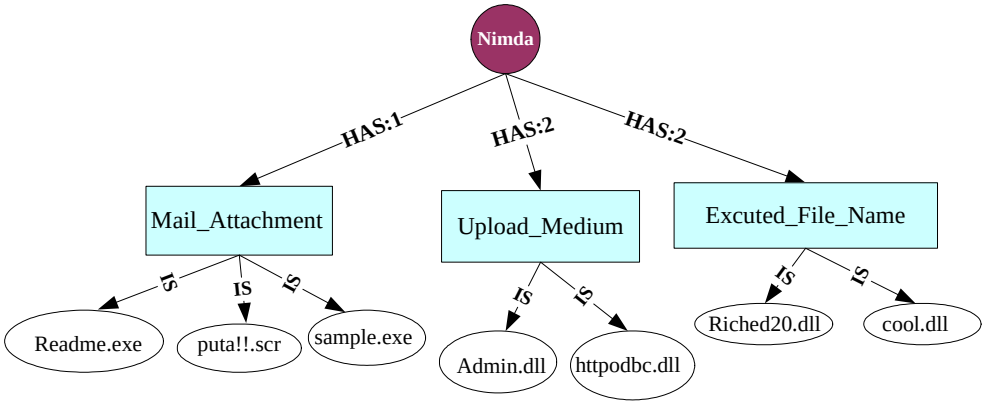


Figure 5.4: Example of Nimda ontology

5.3 Worm Immune Service Expert System

Up to now, there are many antivirus products that can discover worms, virus or Trojan horse in a computer system. However, these products are hard to automatically discover the variants of worms because the signature based approach appears fails when the signatures are changed. To overcome the weakness, the WISE is proposed to enhance the commercial antivirus products instead of replacing them. WISE, unlike pattern matching system, does not need to rewrite the program but just updates the knowledge base to modify the defense mechanism when the worms' mutation occurred; as a result, WISE can be maintained by simply updating the knowledge bases. Since the knowledge of worms accumulates in fast speed, the knowledge constructed today may be degraded in the near future. Therefore, we need to design a knowledge acquisition method to solve the adaptive problem in the dynamic changing

domain like the worm domain, and the knowledge in such domain we call it the evolutional knowledge. Moreover, since the situation of different configurations from different hosts complicates the intrusion behavior analysis, we propose a collaborative architecture for WISE. The collaborative architecture of WISE is shown in Figure 5.5.

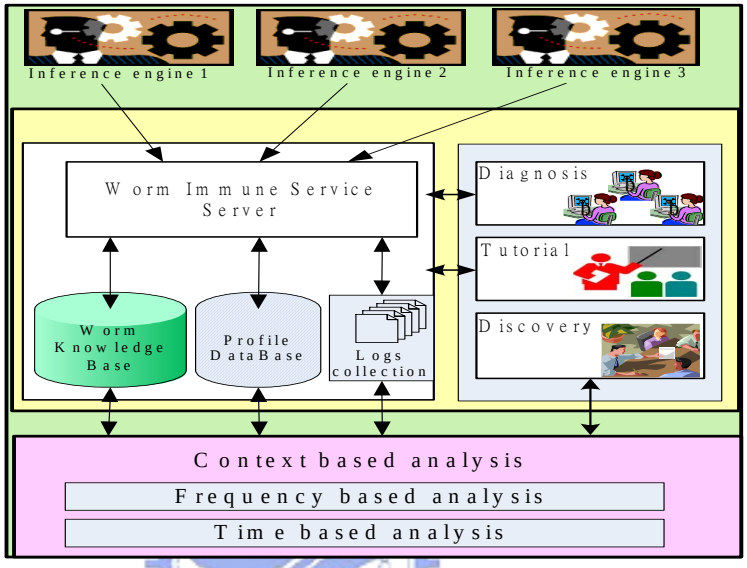


Figure 5.5: Worm Immune Service Expert System

Since there is abundant information of worms from Internet, many technological documents are easy to collect. Then the network environment can be further deployed the multiple inference engines for gathering different inference results according to the different BDML context configurations.

VODKA is applied to collect relevant information to help experts discover the occurrence of variants form the original knowledge base. Also in the architecture, *VODKA* helps *D-EMCUD* construct Attribute-Signal Table (AST) to reconstruct AOT and update acquisition table through three stages, including log collecting stage (time interval tracing stage), knowledge learning stage which would also draw support form profile database, a BDML context information, based on *CAKE* architecture and further polishes the knowledge.

The environment is to provide a web interface to interact with the users by inferring all the symptoms inputted by them or discovered by scanning tools. For example, if worm infects a system, the user can scan the host computer by some general antivirus software or can look for help from the Internet. Then WISE collects all the information from the users and infers the information based upon the knowledge base constructed by *CAKE* method. Consequently, the result after inferring will be passed to the users and can be used to teach them how to recover the system to the normal state. The example of collaborative WISE environment, shown in Figure 5.6, includes local analysis and global analysis. Frequency based analysis (*VODKA*) and time based analysis (*TEA*) are used in each local learning module, and the information after analysis are recorded as context based *BDML* documents. Then the collaborative analysis center analyzes all the context information from locals based upon several heuristics to discover the evolutional knowledge.

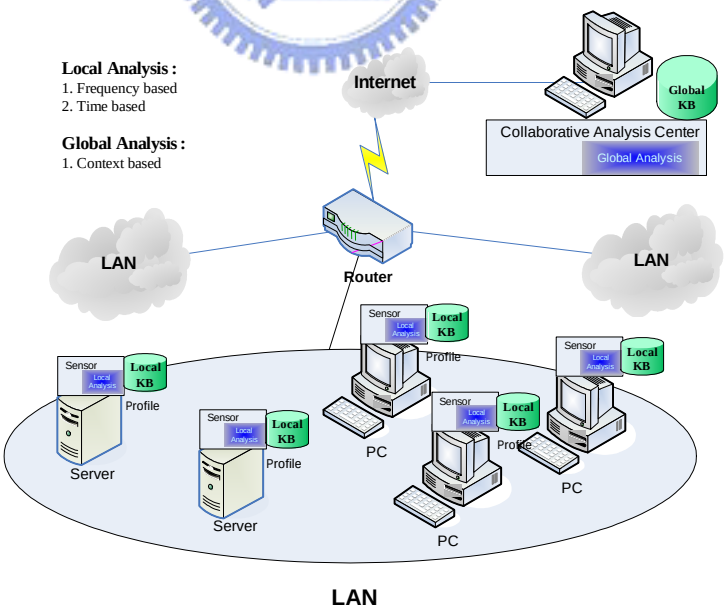


Figure 5.6: WISE environment

5.4 Experimental Results

We implemented a *D-EMCUD* web based system based upon architecture of *CAKE*, and WISE is used as an experiment domain. We set up three computers with the *D-EMCUD* system and each one is designed with the specific knowledge base according to the BDML profile. To evaluate the performance of *CAKE* architecture, we test the system to generate 100 rules, 200 rules, 400 rules, 800 rules and 1000 rules respectively. The difference between the signal host knowledge based construction and multiple hosts knowledge based construction is shown in Table 5.10 which shows the acquisition from multiple hosts based upon *CAKE* architecture is more efficient.

Notice that it does not mean the environment with multiple sensors is always the best. The collaborative heuristics between sensors can affect the whole system's performance. However, after the experiment, we do improve the performance through the environment with multiple sensors based upon *CAKE* architecture.

Table 5.10: Knowledge base construction efficiency

rules	100	200	400	800	1000
Single host	63 ms	126 ms	280 ms	656 ms	1172 ms
Multiple hosts	48 ms	93 ms	187 ms	438 ms	969 ms

Subsequently, each host or computer performs each duty of inference, and every inference process will be controlled by the collaborative heuristics to decide whether each knowledge base should be inferred. From the collection of the inference results, some evidences of worms may be further approved or disapproved. The experimental results in inference performances on designed models are sets with 100 rules, 200

rules, 400 rules, 800 rules and 1000 rules respectively to infer a goal. Examining the inference time to reach the goal, Table 5.11 shows the better performance in multiple-hosts environment.

Table 5.11: Inference efficiency

rules	100	200	400	800	1000
Single host	15142 ms	27579 ms	32807 ms	47438 ms	60697 ms
Multiple hosts	7240 ms	17485 ms	20960 ms	28611 ms	31202 ms

Finally, we did an experiment of the evaluation of discovering variants. We generated all kinds of test simples including the behaviors of original worms and variants (polymorphic worms) to randomly attack the network. Then we found that *CAKE* successfully discovers the variants those can not be discovered by *VODKA*. Since some critical weak embedded rules may be ignored in the beginning of knowledge based construction, some specific variants will never be found by *VODKA* unless the KB is reconstructed and experts accept the rules which had been ignored before, then apply VODKA once again. However, *CAKE* uses a collaborative learning mechanism based upon all information of the historical logs to reveal the rules which had been ignored at the beginning. In the experiment, we define "*inference number*", which represents the minima number of rules that inference engine has to go through, as a measure of inference efficiency. The experimental result is shown in Table 5.12.

Table 5.12: Variant Discovering ratio

	Original worms	Variant Worms	Inference number
VODKA	100%	60%	193
VODKA^{R}	100%	100%	294
CAKE	100%	100%	200

The second row of Table 5.12, VODKA^{R}, shows that the VODKA is applied

again on the knowledge which has been reconstructed. Obviously, after KB reconstruction, *VODKA* can learn all of the variants; however it needs extra inference number about 100. On the other hand, *CAKE* can also learn all of the variant information, but in the case of *CAKE*, it needs only 7 extra inference costs to reach the goal. Therefore, *CAKE* is more efficient.

Chapter 6 Concluding Remarks

We have proposed *CAKE*, a new knowledge acquisition method enhanced from legacy *EMCUD*, to make the knowledge more adaptive for the current environment. *CAKE* uses *TEA* to record important information in each time interval by interacting with human experts and also the *VODKA*'s learning strategy is used to discover the variant knowledge. Finally, *D-EMCUD* is used to generate the adaptive rules for the current environment. A case study of variants of the computer worm detection is illustrated to ease the experts' efforts from analyzing and learning and to help retrieving meaningful information for making proper decisions since the knowledge bases become more adaptive for a changing environment.

The prototype of WISE has been implemented which can enhance variants detecting and defending abilities in commercial antivirus products by using learning method based upon *CAKE*. We are enriching the worm knowledge bases of WISE by *CAKE* to provide more flexible and reliable worm immune services which can discover the variants that can not be discovered by the original WISE based upon *VODKA* and single host-based architecture.

In the near future, *CAKE* will be applied to different domains and applications. By further identifying more interesting information in BDML, it can cover as many aspects as possible to completely represent the specific domain behaviors. Moreover, designing abundant collaborative heuristics to enrich the scalabilities of CAKE is going to be an important topic.

Reference

[1] J. Adams, “Probabilistic and certainty factors”, In: B. Buchanan and E. Shortliffe, Eds, Rule-Base Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley, 1985.

[2] J. H. Boose, “Personal construct theory and the transfer of human expertise”, American Association for Artificial Intelligence, 1984.

[3] J. H. Boose, “A knowledge acquisition program for expert systems based on personal construct psychology”, International Journal of Man Machine Studies, Vol. 23, 1985, pp 495-525.

[4] J. H. Boose, “ETS: A PCP-based program for building knowledge-based systems”, IEEE Western Conference on Expert Systems, 1986.

[5] J. H. Boose and J.M. Bradshaw, “NeoETS: Capturing expert system knowledge in hierarchical rating grids”, IEEE Expert System in Government Symposium, 1986.

[6] J. H. Boose and J.M. Bradshaw, “Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems”, International Journal of Man Machine Studies, Vol. 26, 1987, pp 29-40.

[7] M. Boicu, “Automatic Knowledge Acquisition from Subject Matter Experts”, IEEE International Conference on Tool with Artificial Intelligent, 2001

[8] P. Crowther and J. Hartnett, “Using repertory grids for knowledge acquisition for spatial expert system” Processing on Intelligent Information System, November 1996.

[9] C. S. Chen, S. S. Tseng and C. L. Liu, “A Unifying Framework for Intelligent DNS Management”, International Journal of Human-Computer Studies, Vol.58, No.4, pp. 415-445, 2003.

[10] J. Diederich, I. Ruhmann and M. May, “KRITON: a knowledge-acquisition workbench for knowledge-based systems”, International Journal of Man Machine Studies, Vol 26, 1987, pp 3-27.

[11] R. Davis, “Interactive transfer of expertise”, Artificial Intelligent, Vol. 56, 1987, pp. 121-157

[12] L. Eshelman, D. Ehret, J. McDermott and M. Tan, “MOLE: a tenacious knowledge acquisition tool”, International Journal of Man Machine Studies, Vol. 26, 1987, pp 41-54.

[13] D. Ellis, Dr. McLean, “Worm Anatomy and Model” ACM CCS workshop on Rapid Malcode (WORM’03), October 2003.

[14] B. R. Gaines, “An overview of knowledge-acquisition and transfer”, International Journal of Man Machine Studies, Vol. 26, 1987, pp 453-427.

[15] W. A. Gale, “Knowledge-based knowledge acquisition for statistical consulting system”, International Journal of Man Machine Studies, Vol. 26, 1987, pp 55-64

[16] J. Han and M. Kamber, *Data Mining Concept and Techniques*, Elsevier (Singapore) Pte Ltd, 2003.

[17] G. J. Hwang and S. S. Tseng, “EMCUD: A knowledge acquisition method which captures embedded meanings under uncertainty.” International Journal of Man-Machine Studies, Vol. 33, No. 4, pp. 431-451, 1990.

[18] G. J. Hwang and S. S. Tseng, “On building a medical diagnostic system of acute exanthema.” Journal of Chinese Institute of Engineers, Vol. 14, No. 2, pp. 185-195, 1991.

[19] G. A. Kelly, *The psychology of personal construct*, New York: Norton, 1955.

[20] M. Karresand, “Separating Trojan Horses, Virus, and Worms- A Proposed Taxonomy of Software Weapons” IEEE Workshop, 2003.

[21] Y. T. Lin, S. S. Tseng and C. F. Tsai, “Design and implementation of New Object-Oriented Rule base Management system.” Journal of Expert Systems with Applications, Vol. 25, pp. 369-385. 2003.

[22] S. C. Lin, C. W. Teng and S. S. Tseng, “Capturing Evolutional Knowledge using Time Interval Tracing”, accepted by Journal of Advance Computational Intelligence and Intelligent Informatics, 2006.

[23] R. M. O’Bannon, “An Intelligent aid to assist knowledge engineers with interviewing experts”, IEEE Western Conference on Expert System, 1987.

[24] M. L. G. Shaw and B. R. Gaines, “Knowledge initiation and transfer tools for experts and novices”, International Journal of Man Machine Studies, Vol. 27, 1987, pp 251-280.

[25] S. S. Tsegn, S. C. Lin and L. H. Liu, “VODKA: Variant Objects Discovering Knowledge Acquisition”, submitted to International Journal of Human-Computer Study, 2006.

[26] C. Wagner, “Breaking the Knowledge Acquisition Bottleneck through Conversational Knowledge Management”, Information Resources Management Journal Vol. 19, Issue 1, 2006.

[27] K. Wang, G. Cretu and S. J. Stolfo, “Anomalous Payload-based Worm Detection and Signature Generation”, International Symposium on Recent Advance In Intrusion Detection, 2005

[28] C. L. Wu, ”A Worm Immune Service Expert system for denial of service attacks”, National Chiao Tung University, Master thesis, June 2005.

[29] F-Secure Corporation, 1988-2005, http://www.f-secure.com/

[30] Symantec Corporation, 1995-2005, http://www.symantec.com/corporate/

[31] Trend Corporation, 1989-2005, http://www.trendmicro.com/tw/home/enterprise.htm