ABSTRACT

In recent years, Continuous Auditing (CA) has become an inevitable trend in current business environment. Computer-aided auditing systems, such as the Generalized Audit Software (GAS), are widely used to complete this task. However, most auditors who are not equipped with IT backgrounds have great degree of difficulty in integrating computer-aided auditing system with their professional knowledge in auditing. This limitation greatly impairs the auditors’ ability to independently and continuously performs tests in the CA environment.

This study proposed a systematic analysis approach to assist auditors in bridging the “semantic gap” between the information system and auditors’ specialty. This proposed approach first provides a framework for auditors to effectively understand business processes and data flow/data structure of information systems. These ingredients are then mapped into the process of auditing including auditing objectives, key controls and auditing rules by employing information process models such as Use-case Diagram, Data Flow Diagram, and Entity-Relationship Diagram. With this approach, auditors can independently design auditing rules which can be automatically embedded in the database. Auditors can also perform auditing tasks independently on a real-time basis and reach the objectives of Continuous Auditing.

Keywords: Generalized Audit Software (GAS), Business Process, Data Flow, Conceptual Model, Continuous Auditing (CA).

INTRODUCTION

The Sarbanes-Oxley Act of 2002 addresses many areas that affect the accuracy and transparency of financial reporting. The most important proposition in the Sarbanes-Oxley is the certification of financial statements: “CEOs and CFOs are required to personally sign and certify the correctness of financial reports”. The New York Stock Exchange (NYSE) and the National Association of Securities Dealers Automated Quotation (NASDAQ) also issued more strict corporate governance procedures for listed firms to follow. This new pressure from the legislation and industry has triggered the need of improving auditing systems. Internal auditors are urgently seeking new ways to enhance auditing effectiveness and efficiency.

Recent research has discussed substantially about new auditing techniques, systems control, and Generalized Audit Software (GAS) to assist auditors improving the effectiveness of corporate governance (Shaik 2005, Brazel 2005, Debreceny et al. 2005). Although Generalized Audit Software has been developed by some auditors to obtain auditing information from the information systems, such software is generally not compatible with the complex file structures of database systems. Auditors often have some degree of difficulty in preparing the data for the first time (Braun and Davis, 2003). Auditors should not only able to understand the semantics, such as data structure, database schema, and business process, but also able to create embedded auditing rules by themselves (Frederick and Aleksandra 2000). Otherwise, auditors may not be able to locate any evidences if the audit trail is removed for data deletion or modification.

GAS also has another drawback: It can not constantly monitor the information system and provide timely warning when unusual transactions or patterns occur in the system. This drawback usually arises from the “Semantic Gap” between the information system and the auditors’ specialty. The “Semantic Gap” can be defined as:

“the large disparity between the low-level features or content descriptors that can be computed automatically by current machines and algorithms, and the richness and subjectivity of semantics in user queries and high-level human interpretations of auditing program”.

Several information system conceptual models have been proven to be able to bridge the gap between end-users and the information systems (Yourdon, 2005). These models enable end-users to better capture and analyze key data for strategic and operational decisions (Olsen, 2002). They also provide richer semantics to assist auditors understanding the data structure and the business processing capabilities of information systems.

The main purpose of this study is to develop a methodology to assist auditors implementing continuous auditing on their company’s information systems. In order to create continuous auditing capacity, it is necessary to narrow the semantic gap between high-level concepts employed by the auditor and low-level feature presentations of the system. Before auditing the information systems, auditors need to design an auditing program to satisfy transaction-related and balance-related auditing objectives. This study suggests a mechanism that auditors can employ to create an auditing program by using existing information system analysis/design documents including use-case diagram, data flow diagram, and entity-relationship diagram. Through the use of information systems semantics, this mechanism can also create auditing rules which will become an embedded auditing module for continuous auditing.

The main contribution of this paper is to make it possible for auditors, by themselves, to be able to (1) Use system analysis/design documents to understand the business process, data flow, and data structure, (2) Provide a stepwise auditing procedure to create auditing program, (3) Establish the relationship between...
auditing program and system analysis/design documents, (4) Create auditing rules for key controls automatically through semantic interface, (5) Manage and monitor the information system promptly and continuously through embedded auditing rules, (6) Preserve any data variation for key data, and (7) Preserve audit trails for future investigation.

The remainder of this paper is organized as follows: Section 2 provides some background and prior studies in this line of research. Section 3 introduces the proposed methodology, and section 4 validates the methodology used in this study. Section 5 presents the implemented system. Section 6 provides a real case to demonstrate that this methodology can practically assist auditors quickly locate problems. At last, conclusions and suggestions for future studies are discussed.

### LITERATURE REVIEW

#### Continuous Auditing

According to research reports of the Canadian Institute of Chartered Accountants (CICA) and American Institute of Certified Public Accountants (AICPA), continuous auditing is “a methodology that enables independent auditors to provide written assurance on a subject matter using a series of auditors’ reports issued simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter”. Since continuous auditing is in its embryonic stage, numerous problems and research issues are bound to arise (Wright, 2002). As a result of this interest, many symposiums were attended by academicians, corporate representatives and accounting professionals who addressed both the theoretical and practical applications of continuous auditing, continuous monitoring, and continuous reporting (Warren and Parker, 2003). The continuous auditing is gradually improved through incorporating advancements in technologies and the trend towards standardization. As a result, auditors must be proficient in the use of new auditing technologies that are required for complex systems. These technologies include embedded auditing modules (EAM), Extensible Business Reporting Language (XBRL), database technology, data warehouse, and internet technology, etc. Table 1 shows several important research issues related to the continuous auditing.

#### Design a Continuous Auditing Program

Arens et al. (2004) defined the continuous auditing as “A computerized accounting system in which auditors can perform tests of controls and substantive tests of transaction throughout the year in order to identify significant or unusual transactions and to determine whether any change have been made to the computer program”. Arens et al. (2004) proposed a step-wise procedure to design the continuous auditing environment. In this proposed continuous auditing environment, the tests of controls and substantive tests of transaction are embedded into the system. This embedded auditing module often includes several auditing rules associated with their auditing objectives. It is the auditor’s duty to design the auditing rules during the auditing procedure. We extend Arens’s stepwise auditing procedure for continuous auditing as shown in Figure 1.

### Figure 1. The stepwise of auditing procedure for continuous auditing.

#### Table 1. Summary of Recent E-auditing Studies

<table>
<thead>
<tr>
<th>TECHNOLOGY and ENVIRONMENT</th>
<th>TOPIC</th>
<th>RELATED STUDIES</th>
</tr>
</thead>
</table>
the information system applications and how they support the business processes. This task is usually done through interviews with operation managers and application administrators. Auditors must determine who has the authority to run key control functions such as approving, posting, deleting, and so on. Once the business processes are understood, auditors must then look at how applications are configured. They must also understand the structures of information system (Arens et al., 2004). Data structure within an application is a difficult process to undertake and examine, largely because the structure of a large system is very complex (Peslak, 2005). Typically only a database administrator (DBA) will have access to the tables in the applications, so auditors will have to solicit assistance from the DBA once they have defined the auditing requirements (Steven, 1999). However, DBAs are overwhelmingly trying to ensure that the system can operate efficiently, so assistance from them may be limited (Cooke, 2004). Therefore, even with the assistance of GAS, it is difficult for auditors to do the auditing work well without knowing data structure. Olsen (2002) illustrated that “Database Administrators (DBA) and auditors often struggle to understand each other’s areas. DBAs are concerned with the technology involved in database implementation and the auditors are concerned with the design of and adherence to business rules.”

The conceptual model is intended to facilitate better communication between DBAs and end-users of information system (Olsen, 2002). The conceptual model can provide effective assistance for users to learn about the existing business, come to understand its problem, define objectives for improvement, and define the detailed business requirements (Niv et al. 2002). The conceptual model currently in wide use includes Entity-Relationship Diagram (ERD), Data Flow Diagram (DFD), Unified Modeling Language (UML), the Resources/Events agents (REA) and Integrated Computer-Aided Manufacturing DEFinition (IDEF). These models offer rich semantics which can help auditors understand the data structure and the business process capabilities of database applications (Jones et al., 2005).

**SYSTEM ARCHITECTURE**

In order to implement continuous auditing for a firm, the most important thing to do is to bridge the semantic gap as defined above. The following proposes a systematic procedure for the implementation of continuous auditing. First, the information systems conceptual model is adopted to assist auditors understanding the business process of the firm and establishing the auditing program. This model can also provide an opportunity for the auditor to obtain necessary knowledge about the data flow and data structure related to the auditing program. Such knowledge is critical for the auditor to create auditing rules which is to be embedded in the auditing module. The embedded auditing module will maintain a complete and continuous audit trail, which can provide a real-time monitoring for any pattern of irregularity in the information system. Finally, these audit trails can be summarized to an audit report for auditors to monitor and control the business process flow. The continuous auditing assistance system (CAAS) architecture is described in Figure 2.

**Systematic Procedure for Building the Continuous Auditing Assistance System (CAAS)**

This section describes a procedure which can bridge the semantic gap between auditors and information system in order to design a Continuous Auditing Assistance System (CAAS). The CAAS produced by such systematic procedure is very critical for auditors to understand their business and information environments.

This systematic procedure for continuous auditing is illustrated in the Figure 3. Auditors first need to establish an auditing program, which includes business processes to be audited and the auditing objectives involved in each process, key controls for each auditing objective, and auditing rules for each key control. This auditing program is established with assistance from system analysis/design documents and conceptual models which can effectively bridge the semantic gap. Auditors can use Use-Case Diagrams (UCD) to understand the company’s business processes and to map the auditing plan into key business processes promptly. Next, the UCD mechanism maps these key business processes into data processes via the Data Flow Diagrams (DFD). Finally, the DFD passes data processes to an Entity-Relationship Diagram (ERD) that can help identify related data model and establish auditing rules. Finally, these rules will be compared to the corresponding system operations and an embedded auditing module will be created. This embedded auditing module can generate trigger commands which would be stored in the database to help auditors continuously monitor critical processes and generate audit reports.

**Pre-step-Understand Business Process**

The auditor must have a thorough understanding of the firm’s business operations in order to establish an effective auditing program. The auditor should carefully reviews the regulatory/operating environments and business processes of the firm. Appointments need to be arranged to observe operations personally and discuss with key personnel of each unit. It is also very important for the auditor to refer to documents for information System analysis and design to obtain a better picture of all the planned business processes.

Traditionally, system analysts employ (1) Use-Case modeling to perform the requirement analysis, (2) Process Modeling for organizing and documenting the structure and flow of system’s process, and (3) Data Modeling for defining business requirements for a database (Whitten et al., 2004). Conceptual models have several levels of complication. In order to reveal more details, the auditor must get to all layers of the details of conceptual models.

![Figure 2. Continuous Auditing Assistance System Architecture](image-url)
For example, the data flow diagram may be expanded into a primitive data flow diagram.

**Step 1 — Set Auditing Objectives**

The auditor’s primary responsibility is to determine whether management assertions about financial statements are justified. There are six general transaction-related auditing objectives (Arens et al. 2004):

- Existence — Recorded transactions indeed exist,
- Completeness — Existing transactions are all recorded,
- Accuracy — Recorded transactions are stated at the correct amounts,
- Classification — Transactions included in the journals are properly classified,
- Timing — transactions are recorded on the correct dates,
- Posting and summarization — recorded transactions are properly included in the master files and are correctly summarized.

Auditing is usually started with dividing the enterprise’s business processes into small segments such that related transactions are included in the same segment. This is called the cycle approach (Fang Z., 2004). For example, customer order, sales order, sales return, and cash receipts are the related transactions that comprehend sales cycle. Auditors can refer to the system requirement documents to classify business processes into various cycles.

Use-case diagram is usually used to develop system requirement. Booch et al. (1998) illustrated that “A use-case diagram is quite straightforward and depicts two types of elements that either represents business roles or the business processes”. The adoption of a use-case diagram for requirement gathering seems most practical as the gap between the developer’s background knowledge of the domain and that of the users need to be bridged in order to tackle the problem of architectural and interaction design (Odeh et al., 2003). A use-case diagram captures the functional aspects of a system — who does what with the system, for what purpose (goal), without dealing with system internals. Furthermore, a use-case diagram documents how the business works and what the business goals are for each interaction with the system (Eriksson et al. 1999, Booch et al. 1998). Therefore, auditor can consult the IT department and use the use-case Diagrams and the use-case narratives that were used during the system development as the references. The UCD can provide great assistance in accomplishing the following four tasks:

- Modeling a business of which all participants are in the system
- Capturing the system requirements
- Indicating what the system should do
- Identifying the required classes for the system

Generally, in the use-case diagram, the system is treated as a “black box”, and the interactions with system, including system responses, are as if perceived from outside the system. When the auditing objective is set, the auditor can focus on the system to uncover details using data flow diagram.

**Step 2 — Set Key Control**

Traditionally, auditors use following procedures to obtain an understanding of internal control (Arens et al., 2003):

- Make inquiries of appropriate client personnel.
- Examine manuals, documents, records, and reports.
- Observe control-related activities and operations.
- Re-perform client procedures.

Auditors usually need to spend a lot of time to understand the
enterprise’s control activities. Data Flow Diagram (DFD) can greatly save time for auditors, Data Flow Diagram (DFD) not only can provide assistance in modeling an information system, but also acts as an effective tool for business planning and strategic planning (Okrent and Vokurka, 2004, Yourdon, 2005). The auditor can specifies that some use-cases in the UCD have associated with some processes in DFD.

The DFD can support some information about the business process that assists auditors setting up key controls. The DFD can usually provide the following information:

- The overall system and its interaction with its environment.
- The major subsystems and their interactions.
- The information that the processes require as input and/or the information.
- The various individual processes that the system carries out.
- The data stores that will typically exist as files or databases.

**Step 3 — Design Auditing Rules**

Auditors identify key controls that should reduce control risk for each auditing objective.

After appropriately documenting key controls, it is critical that a system walk-through be completed to ensure that the described controls have actually been put into place.

Traditionally, auditors have two types of tests that will result in an effective and efficient auditing.

- Test of control — examining both the design and the operating effectiveness of each key control.
- Substantive tests — assuring that the recorded transactions exist and existing transaction are recorded. Auditors usually perform substantive tests to check record correctness, proper classification and summarization, and posting effectiveness to the master files.

For information systems, auditors often perform tests of controls and substantive tests throughout the year to identify significant or unusual transactions and to examine whether any irregular change has been done to the system. Auditors should be able to understand the database schema and table information, so they can do the tests in physical database system. However auditors often have some degree of difficulty in preparing the data for the first time. Fortunately, Entity Relationship Diagrams (ERD) can provide rich semantic to help auditors understand the data structure of database applications.

The data store in DFD should describe what the business wants to store. The “Datastore” symbol in DFD is equivalent to the “Entity” symbol in ERD. One table in relational database can be transformed into one or more entities. Through the mapping process, a partial ERD can be obtained while all auditing information is contained in the partial ERD. The mapping relationship is shown in Figure 5.

The ERD represents some information including:

- The business components and how they are related to one another.
- The detail of entity by its attributes or properties.
- The details about the entity-relationship which represent the business rules.

Auditor can map the data store in DFD to the entity, relationship, and attributes in ERD; the auditor is then able to analyze the captured transactions and detailed data on a real-time and continuous basis as client transactions are processed.

Auditors can use the auditing rule manipulator to generate auditing rules and store them in the database system automatically. The auditing rule command follows certain syntax rules that are easy for auditors to understand. The embedded auditing module in the database system will follow the auditing rules to capture transactions with characteristic that are of specific interest to the auditor. The embedded auditing module will be illustrated in details in section 5.

**Step 4 — Create Auditing Rules**

Based on the results in step3, all tables and attributes which are related to auditing rules are being recorded. Therefore, auditors can create auditing rules and these rules will be embedded in the database by themselves. Generally, embedded auditing rules are created using trigger command. The event-based rule language is usually used, in which a rule is triggered by an event such as the insertion, deletion or modification of data. The Event-Condition-Action (ECA) model for active database is widely used. The general form of rules in this model is as follows:

```
On event
If condition
Then action
```

The rule is triggered when the event occurs. Once the rule is triggered, the condition is then checked. If the condition is satisfied, the action is then executed. The purpose of trigger command is to (1) perform automatic monitoring of conditions.
defined over the database state, (2) to take action (possibly subject to timing constraints) when the state of the underlying database changes (transaction-triggered processing).

Suppose that we want to enforce the following constraint: when an employee record is inserted into the employee relation table, the salary of the employee must be less than that of employee’s manager. The following rule can be used to enforce the constraint:

Define trigger employee_salary
On insert employee
If employee.salary > (Select manager.salary from manager,
   employee Where manager.name = employee.manager)
Then abort

According to the data flow shown in DFD, the data flow can be divided into either “process to datastore” or “datastore to process”. As shown in Figure 6, data flowing from process to data store represents “insert”, “delete”, or “update”; as shown in Figure 7, while data flowing from data store to process represents “select”.

Therefore, we can map the “Process to datastore” of DFD to the event of the ECA. The mapping rules are shown as follows:

- “Insert dataflow of process to datastore” is mapped to “insert event of the ECA.”
- “Update dataflow of process to datastore” is mapped to “update event of the ECA.”
- “Delete dataflow of process to datastore” is mapped to “delete event of the ECA.”

An enhanced model of this type of mapping rule has also been found in the modern database system, such as Oracle, DB2, Sybase, and MS SQL Server. Some systems also extend the semantics into the rule body. The event-condition can be defined in more depth, such as before event-condition, after event-condition, etc. For example:

- Check point before the dataflow is mapped to “before event-condition”
- Check point after the dataflow is mapped to “after event-condition”

SYSTEM VERIFICATION

To verify our mechanism, we use the case study analysis approach. This study is based on the order cycle for SoundStage Entertainment Club [Whitten et al., 2004]. The related UCD, DFD, and ERD are shown in Figure 8. The auditor’s obligation in the order cycle includes obtaining an understanding of internal control, assessment of control risk, tests of controls, and substantive tests of transactions.

Step 1 — Setting Auditing Objectives

The member order entry is selected as the main business process to be audited. In the Use-Case diagram shown in Figure 8, there are several important processes in the member order entry:

- Club member submitting an order for SoundStage products.
- Club member revising an order previously placed.
- Club member canceling an order previously placed.
- Club member viewing products for possible purchase.

There are four auditing objectives related to member order as follows:

- All orders actually made from member (Existence)
All products in the member order are in the product table (Existence)
All orders are recorded (Completeness)
The amount of orders are correctly billed and recorded (Accuracy)
The order creation date is less than member expiration date (Timing)

Step 2 — Setting Key Control

Auditor can focus on the member order and map it to the data flow diagram, as shown in Figure 9. We can map the UCD and DFD, the result is shown in follow:

- UCD (Place new order) mapped to DFD (1 Member Order)

The auditor can use the DFD diagram to understand the control activities. For example, the auditing objective “All orders actually made from member (Existence)” can be mapped to two key controls:

- Check the member who exists in “Member” file
- Check the member who has been record in the “Member Orders” file

The complete key controls for each auditing objective are shown in Table 2.

Step 3 — Design Auditing Rules

Tests of customer order are based on the auditor’s understanding of the information system environment. The auditor studies the DFD and mapping of the relate data store in the ERD. The mapping process is shown in Figure 10. In this case, there are four datastores in the DFD. The mapping rules are shown in the following:

- Data store (Member Ordered Products) mapped to View (Member Ordered Products)
- Data store (Members) mapped to Entity (Member)
- Data store (Member Orders) mapped to Entity (Member Order)
- Data store (Product) mapped to Entity (Products)

The auditor can use the ERD to understand the database schema and to set the auditing rule. For example, the key control “Check the customer exists in customer file” is mapped to the auditing procedure “Customer number in customer order master...
Step 4 — Create Auditing Rules

Auditors can set the auditing rules based on the auditing program. After setting the auditing rules, auditors can identify relevant tables and fields for each auditing rules and specify the actions to be audited. Embedded Auditing Module (EAM) can then support detective control to monitor transactions that are expected to be material to the auditor. The following summarizes the process in details:

1. **<Member_No> in <Member Order> exists in <Member>**
   Insert rule 1: When inserting an order, check the data <Member.Member_No> exists
   Update rule 1: When updating an order, check the data <Member.Member_No> exists
2. `<Credit limits>` for `<Member>` is greater than 0
   Insert rule 2: When inserting an order, check the value `<Member.Credit Limit` > 0
   Update rule 2: When updating a order, check the value `<Member.Credit Limit` > 0

3. `<Product_No>` in `<Member Order>` exists in `<Product>`
   Insert rule 3: When inserting an order, check the data `<Product.Product_No>` exists
   Update rule 3: When updating an order, check the data `<Product.Product_No>` exists

4. `<Order_No>` is a sequence of `<Member Order>`
   Insert rule 4: When inserting an order, check the sequence of `<Member Order.Order_No>`

5. Check duplicate `<Order_No>` in `<Member Order>`
   Insert rule 5: When inserting an order, check the duplicate `<Member Order.Order_No>`

6. `<Order_No>` in `<Member Order>` exists in `<Member Ordered Products>`
   Insert rule 6: When inserting an order, check the value `<Member Ordered Products.Order_No>` exists in `<Member Order.Order_No>`
   Update rule 4: When updating an order, check the value `<Member Ordered Products.Order_No>` exists in `<Member Order.Order_No>`

7. Recomputed `<Order_Sub_Total>` and `<Order_Sales_Tax>` in `<Member Order>`
   Insert rule 7: When insert an order, recomputed the value `<Member Order.Order_Sales_Tax` = `<Member Order.Order_Sub_Total` * 0.05
   Update rule 5: When update an order, recomputed the value `<Member Order.Order_Sales_Tax` = `<Member Order.Order_Sub_Total` * 0.05

8. The sum of `<Order_Sub_Total>` in `<Member Order>` is less than `<Credit Limit>` in `<Member>`
   Insert rule 8: When insert an order, recomputed the value `SUM<Member Order.Order_Sub_Total` _ `<Member.Credit_Limit`
   Update rule 6: When update an order, recomputed the value `SUM<Member Order.Order_Sub_Total` _ `<Member.Credit_Limit`

9. The `<Unit_Price>` in `<Member Order>` is greater than with the `<Default_Unit_Price>` in `<Products>`
   Insert rule 9: When insert an order, check the value `<Member Order.Unit_Price` > `<Product.Default_Unit_Price`
   Update rule 7: When update an order, check the value `<Member Order.Unit_Price` > `<Product.Default_Unit_Price`

10. Check the `<Order_Creation_Date>` in `<Member Order>` is greater than `<Expiration_Date>` in `<Member>`
    Insert rule 10: When insert an order, check the date `<Member Order.Order_Creation_Date` > `<Member.Expiration_Date`

Auditors can generate trigger commands and store them in the database system. Trigger command follows certain syntax rules that are easy for auditors to write simple programs, or generate trigger command automatically through a systematized approach (as shown in section 5, system implementation).

**SYSTEM IMPLEMENTATION**

This paper implemented a real system to prove that this mechanism is feasible. The system is an assisting tool which helps IT auditors define the auditing rules and store them in the database. This system was built on “Windows 2003 server”, used “Centura 1.51” to design the user interface, and was connected to “Microsoft SQL Server 2000” as its database.
Auditors can build the auditing rules by the systematized interface design. The interface is shown in Figure 11.

When auditors use the “rule define” function to build auditing rules, it will automatically generate the corresponding trigger command and starts using the auditing rule in real time. The rule generator is shown in Figure 12. Auditors can check the audit trail file at anytime in order to implement continuous auditing or understand what unusual situation is occurring in the currently-operating information system.

SYSTEM VALIDATION — F COMPANY

The F company, a leading European industrial group, is a global company with about 13,000 people and more than 30 production facilities. The company operates in 30 countries throughout the world in 15 different languages. F company is organized in 4 fully integrated Business Divisions controlling its own sales marketing design and manufacturing activities.

The F company uses a global ERP system (MFG/PRO), but each local company needs to develop some local systems that connected with MFG/PRO in order to meet different local policies and business model. In Taiwan, the local branch out-sources a Tax Reporting System (TRS) to meet different tax reporting requirements. Financial staffs export daily operation data into the TRS. Because daily operation data is usually aggregate information, it is required to divide the data into several detailed information through the system or by financial staffs. It is also required to generate invoice for proof and declaration.

The MFG/PRO system performs internal control and auditing features. However, the internal control and tax report examination functions is built in the tax reporting system as well. As a result, the data involves cross-system, cross-platform and manual operation. These have been the blind spot for auditing purpose. Since this information is related to tax issues, any error could result in great penalty and damage the company’s reputation. In addition, traditional auditing framework cannot provide timely report which has caused concerns from the F company. The model introduced by this research to assist auditors implementing the continuous auditing ability can effectively resolve this problem. The system is continuously monitored by the embedded auditing rules. Audit trail is also created for questionable data access. We create eighteen insert auditing rules, fourteen update auditing rules, and five delete auditing rules in total. Part of the audit trails are shown in Figure 13. Most of the audit trails show no irregularity pattern. However, our system detects a serious problem which is described in details in the following.

The accounting personnel need to divide the summary invoice into several independent invoices. There are three functions when doing this process:

1. Import invoice: “import original summary invoice”;
2. Separated invoice: “divide the summary invoice into several independent invoices”;
3. Delete invoice: “delete original summary invoice”.

Figure 12. Trigger Generated by Update_Rule1

Figure 13. Audit Trail File
In the normal situation, one summary invoice may either appear in “import invoice” and “separated invoice” or appear in “import invoice” and “delete invoice”, see the Figure 13 (case1). However, we find a special case, see Figure 13 (case2), one invoice has been divided twice. This is a very serious mistake. If this mistake was not found immediately, it would cause a penalty 10 times of the original amount according to local government regulation. In this case, the penalty would be about USD$350,000 for tax evasion. This real case illustrates the importance of continuous accounting. The model provided by this research can effectively and timely identify mistakes and reduce business risk caused by the information system.

CONCLUSION

The inevitability of internal control and continuous auditing is beyond any doubt. Yet, there are two problems which hinder the implementation of an effective continuous auditing. First, auditors usually cannot thoroughly carry out continuous E-auditing because of their lack of IT proficiency and inefficient communication and interaction with IT personnel and information system. As a result, many auditors give up the auditing software and resort to the traditional inefficient paper-based auditing approach. This paper-based auditing approach creates a poorly-managed internal control system, which greatly impairs the effectiveness of continuous auditing, and exposes the enterprise to high level of business risk.

Second, a database system usually only stores the most updated record of a transaction without keeping its updating trail. However, retaining all business trails is the necessary auditing procedure for future tracing, analyzing, and investigation. Many enterprises usually have set up various internal control rules. Yet their information systems which fail to record and analyze all updating trails usually find out their internal control rules are not effectively followed. Internal control rules need to be embedded into an information system by constantly monitoring all updating trails in order to assure that all operations follow these internal control rules.

This study provides a solution framework to resolve these two problems mentioned above. Using the system analysis document and the auditing objectives as inputs, through the proposed analyzing processes, this solution framework helps build the corresponding auditing rules which integrate key business processes and database model. This framework also provides important auditing information to the auditors which can help them understand business operation status without dealing with the enormous and complicated database structure. Regarding the implementation of continuous auditing, this paper used an embedded auditing module to record all the updating trails of each transaction. Auditors can also easily set up these auditing rules into the embedded auditing module, even if they are not familiar with the database query language. Record insertion, deletion, and its relationship to exist in record can all be recorded in an auditing database through this mechanism. Such complete audit trail will enable auditors to monitor records more clearly at every updating steps of a transaction record, so any violation of internal control rules or suspicious business activity can be promptly identified.

This study suggests the use of conceptual models facilitate better communication between DBAs and end-users of information system. The CAAS prototype established in this study is still in its initial stage. There is still room for progress before this prototype can become a fully automatic assistance tool. In the future, this research should continue to establish more efficient construction methods for auditing information, reduce the uncertainties caused by manual judging, enhance the process flow control, and improve the quality of auditing information.

ACKNOWLEDGMENT

The National Science Council, Taiwan, under Grant No. NSC96-2416-H-036-001-, has supported the work presented in this article. We greatly appreciate its financial support and encouragement.

REFERENCES

15. Searcy D. L. and Woodroof J. B. “Continuous auditing: