Abstract: In today's fast paced business world, real-time information systems facilitate real-time accounting systems and real-time communication between entities. Real-time assurances can only be provided by continuous auditing technologies and what continuous auditing technologies can be utilized to facilitate continuous auditing for the next generation of accounting systems has become very important. Relying on a number of components of Web services technology, a new model based on Web services for continuously audit business processes is presented in this paper. The continuous auditing mechanism would run in the auditee's system and could be applied to provide assurance about specific business processes. The frameworks and technologies that facilitate such a Web-service-based continuous auditing mechanism are described. The features of this model are also presented in this paper.

Key-Words: Continuous auditing; Web services; XML; XBRL GL; Model

1 Introduction

With the development of information technology, human society has entered a new era of knowledge-based economy and globalization. Information technology has been widely used in companies, as accounting information systems are popularly used. Now more and more companies publish corporate information almost in real time, because timely financial or non-financial reports are in a high demand. As a result of this new demand, audit professionals are forced to implement auditing practices in a continuous time frame, and the time for continuous audit (CA) has come.

Research on continuous auditing models has blossomed in the last two decades [2,13]. There have been several continuous auditing models, such as EAM and ADM, and most are merely conceptual [10,11]. However, few seem to have been implemented in real time systems and the feasibility of implementing is also a problem.

In this study, we propose a continuous auditing model that uses web services technology. Web services technology leverages the power of the extensible markup language (XML) and related technologies. Web services have solved the interoperability of applications across operating systems, programming languages, and object models. Web services can achieve this by relying on well-supported Internet standards. Before this study, Groomer and Murthy have proposed a continuous auditing web services model (CAWS) [12]. It is a "pull" model. Based on this model, we proposed a new continuous auditing model using web services, which is a "push" model.

In the following sections, we first present some background. Section 3 introduces this new model, describing the specific components of the XML Web services framework that would be involved in such a mechanism. The demonstration and discussions are presented in Section 4 and 5, respectively.

2 Background

2.1 How Web Services Work

The official World Wide Web Consortium (W3C) definition follows: "A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols" [14].

The advent of Web services technology has changed the way computer software and the Internet. Businesses and organizations can take processes and encapsulate them in a web service and publish them on the World Wide Web. Conceptually, Web services can be thought of as a "stack" of emerging standards that describe a service-oriented, component-based application architecture.

Web services have three essential elements, as...
follows:

Simple Object Access Protocol (SOAP). It is the standard used to exchange data over the Internet using Web services and is based on XML.

Web Service Description Language (WSDL). It is an XML format to describe how a particular Web service can be called, what arguments it takes, and so on.

Universal Description, Discovery, and Integration (UDDI). It is a directory to publish business entities and the Web services they offer, and where you can find those services [1].

Thus, SOAP, WSDL, UDDI represent the core set of technologies to make Web services work. A lot of research of Web services has been done, and more details can be seen in other papers.

2.2 XBRL GL

While Web Services can achieve this by relying on well-supported Internet standards, such as Extensible Markup Language (XML), there is no standard “XML grammar” for defining and structuring business process messages. Thus, we need XBRL GL, which is based on XML, to specify how the content of business process messages should be structured.

XBRL comprises an XML-based standard for external financial reporting. This taxonomy is intended to provide a standardized format for representing the data fields found in accounting and operation systems and transactional reports that will allow organizations to tag journal entries, accounting master files, and historical status reports in XBRL [7]. The related XBRL GL specification is aimed at internal accounting systems at the transaction level. Referred to as the Journal taxonomy, the XBRL GL specification can be used for representing both financial and non-financial information.

And a XBRL GL “data hub” should be created to receive XML-based inputs from the company’s internal accounting system and provide XML-based outputs to systems within or outside the company.

3 A Web Services Based Model for Continuous Auditing

This study proposes a continuous auditing model that uses Web services technology to support auditing processes. And its continuous auditing functionality is defined as a set of Web services that reside within the auditee’s computer system rather than the auditor’s system. This model is quite different from CAWS, which lives in the auditor’s environment. CAWS only works when client systems will be fully Web services enabled, exposing not only the underlying data, but also the business practices. Thus, we built this model in the auditee’s accounting systems. It is a “push” model.

3.1 The Continuous Audit Process with This Model

The external auditors would use this model to assist in the gathering of audit evidence to support an opinion on an audit client’s financial statements. And the process happens constantly. In such a model, the external auditor could publish audit services on continuous audit Web service registry, which is quite the same as a UDDI directory, residing within the auditee’s accounting system. Users of accounting system seeking these services could invoke them that are registered by auditors on continuous audit Web service registry. Each time auditors register their services, users could invoke these services. Because after registering, auditors and users have initiated an interaction between themselves and they can bind to each other. And auditors can control the processes, because users have to use the services that are published by auditors. While users are finishing their services, these services would be provided on the continuous audit Web service registry so that auditors can invoke them. The order of the specific process can be seen in Fig.1. Thus, the services on continuous audit Web service registry are real time and auditing practices are implemented timely.

Fig.1 depicts the interaction between auditor and auditee, indicating how the continuous audit web services resident on the auditee’s system interfaces with WSDL wrappers for each business process.

In terms of XBRL GL technology, the WSDL wrapper for each business process includes specific portType mappings to facilitate SOAP communication using the HTTP protocol among the auditor, auditee and the continuous audit Web service registry. Both auditor and auditee need to convert their service description, which is prepared to publish to the registry, to XBRL GL format. WSDL is an XML format to describe how a particular Web Service can be called so that it can communicate with XBRL GL data that is also in XML format. So we need an approach to create an XBRL GL data hub for both auditor and auditee. The data hub would interface with WSDL wrapper to facilitate the continuous auditing approach proposed in this paper.
Then the model must define audit exceptions. Auditors constantly retrieve services, which are then converted to XBRL GL data. A fault would be to log the exception by continuous inspection. Data about the exceptions would be stored in data structures defined within the “a container” which is created by auditor to store exception data. However, whether exceptions exist is determined by the specific parameters, in terms of existence, completeness, and accuracy of transaction processing. It is the duty of auditors.

We have mentioned that we establish this model in the auditee’s accounting systems. It is a “push” model. Thus, the continuous audit Web service registry is created within the auditee’s accounting systems. The continuous audit Web service registry is a UDDI directory to publish business entities and the Web services they offer, and where you can find those services. Both auditor and auditee register and invoke their services from this registry.

### 3.2 Confirming the Information with A Third Party

Sometimes, auditors need to confirm specific information with a third party for validation purposes, because business documents must be authenticated and confirmed [3,15]. In business transactions, confirming the information with a third party is necessary.

Fig.2 depicts the whole model, indicating the relationship between the auditor and the third party. As the Web service registry is in auditee’s accounting system and the need for real time confirming is not essential, only a Web server is required in the third party. When auditors need to confirm with the third party, auditor’s system sends a HTTP request to the third party for the queried records. The Web server in the third party receives the request, prepares transporting the records for confirmation and then sends them [4,6]. The auditor’s system accepts the records and compares them with its own records. The end auditor’s system should be able to automatically consume the results, so the output should be returned in XML format. Indeed, it is convenient to interface with WSDL wrapper.

The third party here is quite different from investors, bankers and analysts. The third party here is a party to send the validation for auditor’s confirming. We are not sure whether investors and bankers are interested in assurance provided by auditors, and we suggest building another system for investors and bankers.

### 3.3 The Environment for Continuous Auditing

In this model, both auditor and auditee need to have frequent interactions with internal and external applications and systems for audit evidence, which subsequently will adversely impact processing...
speed of this system. Thus, building an Intranet for auditee’s system itself is essential, which obviously can speed up the auditing process and save a lot of time. Fig.2 shows the environment for continuous auditing.

The continuous audit Web service registry should be built in the auditee’s own Intranet so that it can communicate with the auditee’s accounting system directly. And the auditee’s accounting system can register or invoke the services quickly from Web service registry which is in the same network with accounting system. The communication between the auditor and auditee is via the Internet, and the interaction between the auditor and the third party is also via the Internet.

To conduct a continuous audit, a number of conditions must be present. First, the quality of both the Internet and the Intranet must be high enough so that the basic tasks, registering and invoking the services, can be finished. Second, both auditor and auditee must have highly reliable systems. Auditee’s accounting system must be able to provide the necessary subject matter to the auditor on a timely basis. The auditor must have a high degree of proficiency in information systems, computer technology, and the audited subject matter. And finally, the subject of the audit has suitable characteristics necessary to conduct the audit. For example, if the audit is focused on evaluating internal control, then the auditor must be able to electronically interrogate these controls [12].

4 Benefits and Drawbacks

4.1 Benefits

In addition to the general benefits that can be obtained with other continuous auditing models, this web services model can overcome problems that are associated with the implementation of an EAM, ADM or even CAWS, and assures that audit procedures to be implemented in a continuous way. The basic features and benefits are elaborated as follows.

4.1.1 The continuous auditing functionality can be realized by directly interfacing with the Web services attached to the auditee’s accounting system.

CAWS is created on the auditor’s system and is a “pull” model. But some professors point out that the CAWS approach works only if the client’s systems are fully Web services enable, exposing not only the underlying data, but also the business practices. Another weakness is that clients will passively finish their tasks, because the Web services are not installed within their systems.

Thus, this study proposes a continuous auditing model lodged within the auditee’s computer system rather than the auditor’s system. It is a “push” model. Clients have to actively register the services to the registry because they cannot invoke the services they need until the services have been
registered. As the registry is within every client’s own accounting system, the underlying data and the business practices will not exposed to the outside.

And this web services model requires auditee’s accounting system itself to have a continuous auditing registry. That means a continuous auditing registry has to be installed in every audittee’s accounting system, which seems to be hard to achieve in the short time. However, Web services technology is good at integration. Web services definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols. Thus, achieving the technology of creating an interface with Web services can come true soon.

4.1.2 A continuous auditing model lodging within the audittee’s computer system enhances the security and solves the data ownership issues.

A continuous auditing model lodging within the auditor’s system implies that all data will exposes to the outside. That means unauthorized access will be easy, because data will transport to the auditor’s system and unauthorized access can get the auditee’s data through entering the auditor’s system first [5]. But building the model within the audittee’s system means that all data still leaves in the auditee’s system. The clients firstly must establish an appropriate security mechanism in their Intranet for their accounting systems. Furthermore, clients have to establish other security policy to protect their systems, such as designing digital certificates into the model.

If the model resides in the auditor’s environment, the auditor would have significant responsibility for the security of the data, potentially leading to a loss of independence at least in appearance [9,12]. Thus, the model has to reside in the auditee’s system. And there is no problem of the data ownership, because all data is still in the clients’ system. At the same time, the model is still independent of the development of the system of the client.

4.1.3 As Web services are good at integrating different web applications, the realization of this web services model will be feasible.

This web services model requires auditee’s accounting system itself to have a continuous auditing registry. That means a continuous auditing registry has to be installed in every client’s system and the client’s system is fully Web services enable.

Web service technology is a universally accepted standard. Web services perform callable functions that can be anything from a simple request to complicated business processes. Once a Web service is deployed and registered, other applications can discover and invoke the deployed service. Web services make it possible for software to access documents and run applications in a general way without requiring application specific knowledge and client software. So from a technology perspective, the realization of this web services model will be feasible.

Embedded audit modules (EAM) requires auditors to engage intensively with the system developers in the design phase of a system, so the feasibility of implementing EAM is a problem [8]. Compared to EAM, this web services model needn’t auditors to engage in the design phase of a system. An interaction with the system can be established after the design time and is easy to create.

4.1.4 This model is all using XML based messages to transport.

Web services technology leverages the power of the extensible markup language (XML), and in the study we propose a continuous auditing model that uses web services technology. In this continuous auditing model, we have the WSDL wrapper for each business process includes specific portType mappings. WSDL is an XML format to describe how a particular Web Service can be called. And we establish an XBRL GL data hub for both auditor and auditee, so that it can communicate with WSDL wrapper in XML format. Besides, the output from the third party is also based on XML. Thus, we have “XML grammar” for defining and structuring business process messages.

4.2 Drawbacks

Although the model offers unique benefits for continuous auditing, the model also has some limitations in its application.

Both auditor and auditee need to have frequent interactions with the Web services registry, which subsequently will adversely impact the performance of the system. The implementation of audit procedures consumes significant computing resources. So both auditor and auditee must have highly reliable systems.

A continuous auditing registry has to be installed in every client’s system and the client’s system is fully Web services enable. It implies that corporations have to pay more money to let the client’s system fully Web services enable. The cost to achieve this model will be high.
Independent auditors face new risks in providing on-demand, real-time assurance. We don’t discuss much detail about new risks, but it is really a problem we have to face here. Maybe we will discuss it in the future research and we hope others have good issues about it.

5 Summary and Conclusion
This study proposes a continuous auditing model that adopts Web services technology to undertake the continuous execution of audit procedures. We discuss how Web services and XML can be utilized to facilitate continuous auditing for the next generation of accounting systems. The technology of XML Web services, including SOAP, WSDL, and UDDI, is described, and XBRL is then discussed. After that, a continuous auditing model using Web services technology is proposed, which resides in the auditee’s accounting system.

This paper is a fundamental exploration of the application of Web services technology in a continuous auditing model. A next step in this line of research can be to implement a prototype system demonstrating the feasibility of this model. The prototype could be developed following the XBRL GL route.

Future research could look at the security mechanisms of automatic audit practices and discuss the new risks we have to face in providing on-demand, real-time assurance. Based on our study, we could have more deep research, such as the specific design of the continuous audit Web service registry and the specific communication between auditor and auditee. The emergence of continuous audit will bring the society benefits, so the research of continuous auditing is valuable to our society. We have a lot work to propose a really appropriate approach and test it in order to achieve the goal of continuous auditing.

References: