JOURNAL OF INFORMATION SYSTEMS
Vol. 24, No. 1
Spring 2010
pp. 91–112

American Accounting Association
DOI: 10.2308/jis.2010.24.1.91

Continuous Auditing in ERP System Environments: The Current State and Future Directions

John R. Kuhn, Jr.
University of Louisville

Steve G. Sutton
University of Central Florida

ABSTRACT: Recent research has focused heavily on the practicality and feasibility of alternative architectures for supporting continuous auditing. In this paper, we explore the alternative architectures for continuous auditing that have been proposed in both the research and practice environments. We blend a focus on the practical realities of the current technological options and ERP structures with the emerging theory and research on continuous assurance models. The focus is on identifying the strengths and weaknesses of each architectural form as a basis for forming a research agenda that could allow researchers to contribute to the future evolution of both ERP system designs and auditor implementation strategies. There are substantial implications and insights that should be of interest to both researchers and practitioners interested in exploring continuous audit feasibility, capability, and organizational impact.

Keywords: continuous assurance; continuous auditing; continuous monitoring; enterprise systems; enterprise resource planning (ERP) systems; enterprise risk management; embedded audit modules; management control layer.

I. INTRODUCTION

Over the past 20 years, the discourse on the need for, and ability to deliver, continuous auditing of business information has slowly gained momentum. Today, the concepts of continuous auditing have reached the instantiation stage, becoming a key element in many internal audit departments' risk monitoring strategies. Additionally, continuous auditing is increasingly under consideration as a tool to augment the external audit. This progression has included the evolution of architecturally different methodologies for approaching continuous auditing in computerized environments, primarily embedded audit modules (EAM), which are software modules embedded in an information system (i.e., built into) to monitor activities in such systems (Groomer and Murthy 1989), and a monitoring control layer (MCL), which is an external software module that operates independently of the information system to be monitored but is linked into the system and/or its underlying database to provide a similar level of monitoring (Vasarhelyi et al. 2004). A key component of the discourse in recent years has been focused on the advantages,

Published Online: March 2010

limitations, and feasibility of the two dominant approaches (Alles et al. 2006; Groomer and Murthy 2003; Kogan et al. 1999; Kuhn and Sutton 2006; Vasarhelyi 2006; Vasarhelyi et al. 2004; Woodroof and Searcy 2001).

The constantly evolving state of corporate governance fueled by organizations' focus on strategic enterprise risk management has moved this debate to a focus on practicalities. Continuous auditing tools (EAM and MCL) are quickly becoming key components of overall corporate governance and compliance efforts. Many are designed for specific business processes, enterprise resource planning (ERP), legacy system control settings, transaction processing, and IT processes, etc., that taken together provide the foundation for a more holistic governance, risk management, and compliance (GRC) landscape supporting the entire enterprise (AMR Research, Inc. [AMR] 2008). Continuous auditing applications are being used to support GRC activities across business functions, departments, and IT platforms. As such, third-party software vendors (i.e., non-ERP developers) such as Approva have created additional applications to support enterprise-wide governance interlinking and sub-applications integration designed for monitoring of specific systems and processes. Features of the enterprise-wide governance applications include functionality that supports the identification and management of the varied GRC initiatives across a company (SOX 404 compliance, IT governance, corporate social responsibility, etc.); identification, assessment, categorization, and prioritization of risks; and tracking of key performance and risk indicators in an integrated fashion (e.g., dashboard reporting). The continuous auditing movement has evolved over time, expanded in scope and breadth, and continues to progress forward into unchartered territory.

The ensuing discussion presents a brief history of the origins of continuous auditing and continuous assurance; the development of continuous auditing applications and various alternatives available to companies and auditors; the current state of continuous monitoring and assurance of ERPs in practice; and where we see continuous auditing and overall assurance heading in the future in order to increase the clarity of the current state and provide suggestions for future research that would allow academic researchers to support and lead the growth and evolution. Our perspectives are founded in deep practical experience with ERP systems placed within the context of the evolving theory and research on continuous auditing.[1]

As the push for continuous auditing moves forward, it is important to see where we have been, where we are, and where we are headed. The current discussion contributes to the evolving literature on continuous auditing by examining the *technical* characteristics of continuous auditing through traditional EAM and MCL methods, a modified EAM approach, and newer enterprise-wide/cross-platform applications; the advantages and limitations of alternative methods; and contemporary developments. Of particular interest are the technological advances necessary to support initiatives such as XBRL data tagging and enterprise-wide GRC. Companies of varying sizes and views toward continuous auditing can and will need to choose continuous auditing functionality that supports their individual goals and desires—be it through EAM, MCL, a hybrid approach, or some new adaptation that has yet to evolve.

The remainder of the paper consists of four sections. The first section presents a brief history of the continuous auditing movement; characteristics of various architectural approaches; and advantages/limitations of the approaches from a technical and practical perspective—specifically, real-time monitoring and reporting, complexities of design and maintenance, alarm floods, client

[1] Our views are influenced in part by the first author's personal ERP audit experiences. Prior to entering academia in 2005, the author spent ten years in practice as a Big 4 IT auditor (managed the SAP audit of WorldCom during the fraud restatement) and Financial Systems Manager (managed an SAP implementation), and as an Internal Audit Manager at Siemens Corp.—auditing, evaluating, and implementing continuous auditing functionality both pre- and post-Sarbanes-Oxley 404.

independence, and external auditor legal liability. The second section provides an overview of existing continuous auditing software developed by third-party vendors and ERP developers—primarily, a look at SAP (ERP developer) and Approva (third-party) ventures into enterprise-wide continuous auditing to support overall GRC. The third section discusses continuous monitoring and auditing issues in need of research at both the application and enterprise-wide level that require a variety of research methods. The final section provides a brief summary of this review on the current status and future directions of continuous audit integration with ERP systems.

II. CONTINUOUS AUDITING BACKGROUND AND TYPES OF SYSTEMS

Beginnings of Continuous Auditing and Development of Embedded Audit Modules

ERP systems designed for the processing of business transactions traditionally have been built upon a core database management system (DBMS). Groomer and Murthy (1989) raised the awareness and interest of the accounting research community in the notion of EAM as a viable approach to supplement and enhance the audit of companies with increasingly complex computer-based accounting systems relying on DBMS. They described an approach to continuous auditing of database-driven accounting applications using EAMs to address the unique control and security aspects of database environments. They defined EAM as "modules (code) built into application programs that are designed to capture audit-related information on an ongoing basis." Braun and Davis (2003) similarly define EAM as follows:

> This technique involves the auditor inserting an audit module in the client's application that will identify transactions that meet some pre-specified criteria as they are being processed … Often, these modules are designed in such a way that they can be turned on and off, reducing costs but also reducing coverage. (Braun and Davis 2003, 726)

Vasarhelyi and Halper (1991) advanced the work of Groomer and Murthy (1989), developing the Continuous Process Audit Methodology (CPAM) and illustrating the design and functions of an EAM in a hypothetical customer billing system. The authors subsequently tested a prototype continuous auditing system they created based on CPAM for use with three large financial systems. The prototype results indicated a deeper and more reliable level of audit examination was achieved.

The early works of Groomer and Murthy (1989) and Vasarhelyi and Halper (1991), in conjunction with the joint report on continuous auditing sponsored by the Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants (CICA/AICPA 1999), spawned a stream of continuous auditing research. From an architectural standpoint, a good portion of the efforts adhered to the methodology from these two early works by focusing on EAM as the underlying technological approach (Groomer and Murthy 2003; Debreceny et al. 2003; Murthy 2004; Debreceny et al. 2005; Chen et al. 2007; Loh and Jamieson 2008).

Debreceny et al. (2005) defines a set of essential characteristics for successful continuous monitoring applications: (1) an end-user environment that allows the auditor to establish a set of queries to test transaction integrity constraints either from a pre-defined suite of queries, the modification of the attributes of pre-defined queries, or by the creation of new queries by the construction of simple scripts; (2) a process for registration and scheduling of these queries; (3) a method for running these queries against the flow of transactions for violations either continuously or temporally; (4) a capacity for reporting violations electronically; and (5) an ability to copy the transaction details of violations to secondary storage. We elaborate on these as we create a comparison of characteristics for a variety of continuous auditing application design approaches. But first, clarification of some related terms will help. *Continuous monitoring*, also often referred to as *continuous auditing*, consists of the analysis of data on a real- or near real-time basis against a set of predetermined rule sets. Henrickson (2009) suggests the delineating determinant between the two can be viewed as monitoring being a management responsibility and auditing being an auditor's responsibility. *Continuous reporting*, on the other hand, is the continuous reporting of information about defined phenomena and in a continuous auditing context includes the notification of rule violations occurring as a result of continuous monitoring. Finally, at the macro level sits *continuous assurance*, as noted by Alles et al. (2002, 128):

> [Continuous auditing] is best described as the application of modern information technologies to the standard audit products … Continuous auditing is another step in the path of the evolution of the financial audit from manual to systems-based methods … By contrast, continuous assurance sees continuous auditing as only a subset of a much wider range of new, nonstatutory products and services that will be made possible by these technologies.

All of the approaches (monitoring, auditing, assuring) require the same basic technical capabilities. There are alternative approaches to delivering these capabilities, however, other than EAM. Before considering these alternatives, we review the design and implementation characteristics of EAM.

First, with EAM the programming code for the audit procedures/tasks is developed and implemented *inside* the walls of the target application (i.e., embedded) using the programming language of the application itself. For instance, SAP developed a unique programming language called ABAP for specific use with their core ERP software. If, for instance, a company wishes to build an EAM audit check that verifies every invoice has a corresponding purchase order and goods receipt with identical quantity and dollar amounts within a certain threshold range (i.e., commonly referred to as the three-way matching process), programmers for the company can create an ABAP program and literally "plant it" inside SAP alongside the programs delivered with SAP. This new program is considered "non-native code" in the respect that it is added to SAP afterward. Second, EAM audit functionality can evaluate transactions against programmed audit criteria *live* (i.e., real-time monitoring) as they happen in the application. That does not necessarily mean the EAM module *has* to run constantly, but it *can*, if so desired. Third, EAMs include reporting functionality that notifies predetermined individuals through any of a number of options, including emails, pager notifications, system reports, etc., of any transactions violating the audit procedures. Due to the real-time monitoring functionality, EAM consequently offers real-time reporting. In our three-way matching example, if an invoice was processed and flagged for payment but the invoice amount exceeded the original purchase order by a predetermined threshold (say 1 percent), then an instantaneous notification (referred to as an *alarm*) would be sent to individuals in the internal audit department or external audit firm (or both) depending on what party relies on the EAM. Fourth, storage of the alarms for data retention purposes resides in the same databases/database structures with normal transactions so regularly scheduled application backups capture the EAM auditing data. This also allows the auditor to handle real-time alerts in a batch mode, if so desired. We explore the rationale for such a decision later.

Alternative Technical Architectures to Traditional EAM

The traditional EAM model prescribes coding of audit procedures directly into the auditee's host system to facilitate real-time monitoring and reporting of transactions and system settings. However, certain characteristics of a traditional EAM instantiation may not be the best fit for every company and situation. Therefore, the question arises as to the alternatives available to companies and auditors for continuous auditing implementation. Can the EAM model developed in extant research be modified? Are other architectures more feasible under certain circumstances? Or, are the various continuous auditing architectural strategies situational and, therefore, amenable to co-existence within organizations?

EAM Ghosting

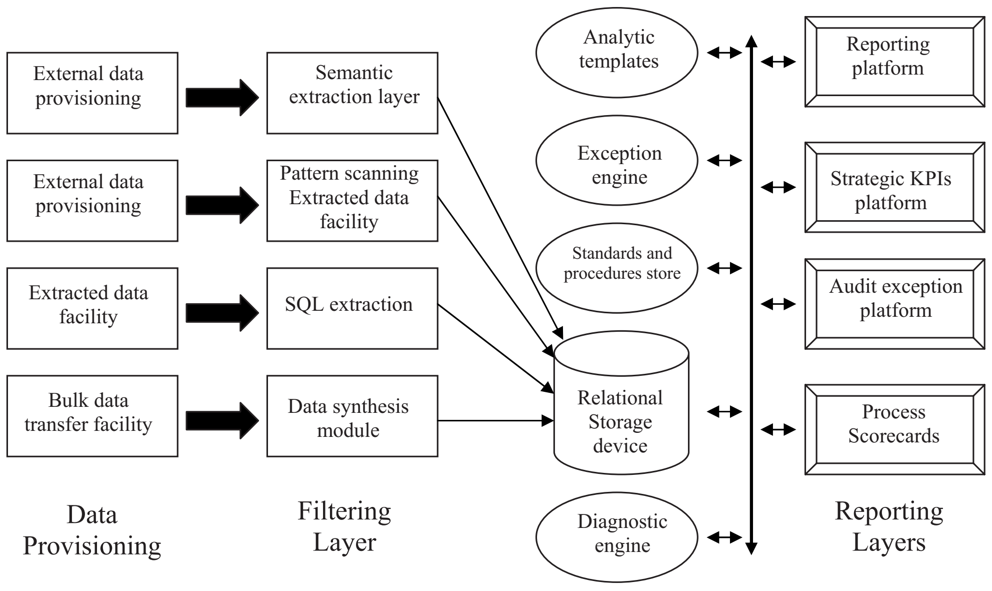
System "ghosting" offers an opportunity to benefit from the advantages of EAM yet implement, operate, and maintain audit functionality outside the production system of a company keeping the audit program separate from the live system. Ghosting entails operating a "copy" of an entire system on separate hardware, including data and system settings, in a real-time fashion similar to an instantaneous fail-over common to disaster recovery and business continuity planning procedures and redundant firewall systems. The EAM functionality would reside in the ghosted copy where there would be no risk of affecting live transaction processing occurring within the production environment.

Two slight variations of EAM Ghosting also exist. Many companies operating large-scale ERP systems utilize a stream of copies of the production environment used for development and change management. Separate systems exist for development (DEV), quality assurance (QAS), and production (PRD), generally hosted on different physical servers that communicate through transport management software (VMware 2009). Changes intended for production will first be configured and tested in DEV by IT personnel, then migrated to QAS for end-user testing before transport to PRD. Typically, the DEV instance contains production data a few days to months old, but with current systems settings. QAS data will be closer in time to PRD with the same system settings. Both variations take advantage of the fact that the DEV and QAS physical servers experience significantly less usage than PRD. For the first variation, companies can segregate a physical server such as the one housing QAS into two partitions, one supporting QAS and the other supporting a mirror-image of PRD updated nightly (rather than real-time) that contains the EAM code. The primary advantage of this approach over standard EAM Ghosting is that IT personnel do not have to worry that real-time processing is updated instantaneously in the system housing the EAM. Second, virtualization technology provided by companies like VMware can take advantage of unused physical hardware space and create an independent virtual machine on the unused QAS server that ghosts the production system but contains the EAM programs. Virtualization requires less physical hardware space and less memory to operate than creating a separate partition on an existing server.

Monitoring Control Layer

Vasarhelyi et al. (2004) introduce an alternative continuous auditing system architecture to EAM referred to as the Monitoring and Control Layer (MCL). MCL can be viewed as the next stage in the evolution of continuous auditing application design—not necessarily one subsuming EAM but more as an alternative to cater to different circumstances. The MCL approach takes the continuous auditing system and "hooks" it into the client's existing enterprise system, generally using a middleware layer to integrate loosely coupled applications such as ERP systems, legacy systems, and web-based applications (e.g., supply chain management, customer relationship management, etc.). The main elements of the MCL architecture consist of: (1) data capture layer; (2) data filtering layer; (3) relational storage; (4) measurement standards layer; (5) inference engine; (6) analytic layer; (7) alarms and alerting layer; and (8) reporting platform as depicted in Figure 1 (Vasarhelyi et al. 2004). Essentially, the continuous auditing system (designed with a simple user interface and underlying database reusable for multiple clients) resides outside the client's network environment (for external auditors) and is controlled by the auditor. The continuous auditing system receives periodic interfaces of client data as determined necessary by the auditor (i.e., not real-time) that are processed against a predefined rule-set of audit procedures inside the continuous auditing application, which is physically and virtually outside the client's audited system (unlike EAM). Any violations as defined by the rule-set trigger an automatic alert to the auditor. These alerts are stored inside the continuous auditing application under direct control by the auditor, not inside the databases supporting the client's ERP system.

FIGURE 1
The Architecture of the MCL Layer
Continuous Monitoring and Control Platform



Source: Vasarhelyi et al. (2004).

Recent research efforts by Kuhn and Sutton (2006), Alles et al. (2006), and Alles et al. (2008) offer evidence of the viability of an MCL continuous auditing approach in an ERP environment. As an example of how fraudulent activities at WorldCom could have been detected earlier, Kuhn and Sutton (2006) lay out the design specifications for integrating a specific set of metrics appropriate for the WorldCom financial reporting system that could have been used to continuously monitor transactions. The study demonstrates how financial transaction data (i.e., journal entries) can be extracted from the client database layer without any direct processing inside the client system. Both Alles et al. (2006) and Alles et al. (2008) document a functioning system prototype developed at Siemens for the continuous monitoring of business process controls and the detection of exceptions to those controls. The prototype demonstrates the use of ABAP (SAP's unique language programming) to extract the business process control data from Siemens' ERP system. The two studies differ on several key aspects. Kuhn and Sutton (2006) *design* continuous auditing procedures for the testing of *financial transactions* based on an *historical case* of financial reporting fraud. Alles et al. (2006) and Alles et al. (2008), on the other hand, *implement* a continuous auditing system for the testing of *internal controls* in a *live* environment.

Table 1 presents general characteristics for continuous auditing systems and how various architectural types compare on those traits. The remainder of this section focuses on analysis of the issues and concerns associated with each approach, including some that all approaches share.

TABLE 1
Characteristics of Continuous Auditing System Types

Characteristic	EAM	EAM Ghosting	MCL
Contains predefined audit rule-set.	Yes	Yes	Yes
Location of audit rule-set and program.	Inside target system	Outside target system, inside client network	Outside target system; outside client network if external auditor; inside client network if internal auditor
Requires installation of additional hardware to implement.	No	Possibly	Yes
Performs real-time monitoring.	Yes	Yes	No
Performs real-time reporting.	Yes	Yes	No
Automatically notifies auditor of rule violations.	Yes	Yes	Yes
Location of stored rule violations.	Target system database	Database external to target system, inside client network	Database external to target system; outside client network if external auditor; inside client network if internal auditor
Owner of system resources.	Client	Client	Client if internal auditor, Audit firm if external auditor

Limitations of Continuous Auditing Applications

Debreceny et al. (2005) provide an overview of several limitations in the use of EAM along with observations regarding the lack of perceived demand for EAM capability. In the Alles et al. (2002) discussion on the feasibility and economics of continuous assurance, the authors raise some concerns with EAM for external auditors. They question whether EAMs designed by external auditors and incorporated into ERP systems ultimately transform the ERP into an "unauditable" system due to a perceived or actual lack of independence. A technical dependence invariably exists between the assuring system (EAM) and assured system (ERP), and the auditor may well be perceived partially responsible for the client's enterprise system. As Alles et al. note, questions such as these can only be answered through theoretical and empirical research. The following discussion expands upon the limitations of EAM, EAM Ghosting, and MCL in greater depth in an effort to improve the understanding of the evolution of continuous auditing and related applications, specifically why ERP vendors were slow to develop EAM functionality.

Technical Concerns

ERP systems, by the nature of their integrated processes, require significant information system resources to operate at an optimal level. As Debreceny et al. (2005) noted, Vasarhelyi and Halper (1991) expressed concerns related to the computing resources EAMs require to monitor transactions and system settings in real-time. Those additional resource requirements risk slowing lem, that of independence. Both scenarios would result in knowledge of the audit procedures and ultimately could be subject to client manipulation. The client could potentially utilize the inside information on the auditor's monitoring procedures to conduct fraudulent activities in unaudited areas or design suspect transactions configured intentionally to not trigger alerts. In short, the existing independence requirements and prescribed procedures for unannounced audit visits are in place to assure the auditor has the opportunity to monitor and search for fraudulent activity without the client having *a priori* knowledge of what will be tested and observed. Sacrifice of this ability seems atheoretical to the underpinnings of audit independence.

Legal liability. A major deterrent to auditors assuming many of the responsibilities often seen desirable by the public (e.g., fraud) has been the risk of legal liability. While the client may be apprehensive about an auditor having access and making changes within their system or simply guiding the implementation of EAM modules within the system, the auditor would likewise be expected to have some trepidation over the risk involved in making such changes to a client's system. Systems are core to an entity's ability to continue its operations and compete in the marketplace. The threat to systems from interacting with trading partners has become severe enough that some powerful corporations such as Wal-Mart have required trading partners that interact with their systems through e-commerce to sign a contractual agreement that the trading partner is responsible for all costs related to damages and repairs to Wal-Mart's systems along with all costs incurred from lost business should their system be infected or otherwise damaged by the trading partner linkage (Gerhardt 2002). It does not seem unreasonable to assume a client organization would want a similar level of risk reduction from the auditor if the auditor is going to be put into a position where damage to the client's system may occur.

Regardless of whether the client and auditor have signed a damage responsibility agreement, should any of the modifications implemented by the auditor adversely affect the processing of client business transactions, the client would likely take legal action against the auditor in order to recover damages incurred. This threat of litigation alone would likely be sufficient to cause many public accounting firms to refuse to undertake continuous auditing of client systems using an EAM strategy.

The GRC Software Market

The massive amounts of resources (man-hours and money) expended to comply with SOX during the first years have driven companies to seek opportunities to streamline the ongoing compliance process. Continuous auditing applications offer the ability to strengthen the internal control environment and provide efficiencies in the overall compliance activities. However, as Debreceny et al. (2005, 23) noted, ERP software vendors generally "believe customers are unwilling to pay a premium for a function that is not perceived as mission-critical" and, therefore, only limited (if any) built-in functionality exists. The disconnect between ERP vendors and their customers may lie in the fact that accelerated filers initially grossly underestimated the resources required to comply in 2004 and the ERP vendors did not receive any type of substantial call for the additional functionality early in the SOX compliance process.

Companies are now looking for ways to reduce the level of manual effort and the related consulting costs incurred the early years of SOX 404 compliance through technology applications. Furthermore, recent PCAOB Auditing Standard No. 5 guidance that recommends external auditors rely more on internal audit work provides an even greater impetus for companies to find alternative solutions for performing automated testing. Independent, third-party software developers have seized the opportunity and developed SOX compliance tools that monitor the effectiveness of internal controls in the financial systems along with the system settings that facilitate control processes; the vendors target these applications directly at internal audit departments and other helping to understand how well organizations assimilate and leverage continuous auditing systems. On the other hand, the unique nature of continuous auditing systems as a control mechanism may also yield advancements in existing theories of assimilation that expand the understanding of the applicability to a broader range of systems.

- What are the organizational factors that lead an organization to adopt continuous audit technologies?
- What are the organizational factors that drive strong assimilation of continuous audit technologies?

The research challenges put forth in this paper likely only scratch the surface. Our understanding of the impact of continuous auditing systems at this point in time is rudimentary and the opportunities for research that can have a significant impact are great.

IV. CONCLUSION

The development and pervasive use of ERP systems provides the critical infrastructure necessary for the effective evolution of the assurance function from a periodic event to an ongoing process through the integration of continuous auditing applications. Two principal system architectures for the development and continuous auditing applications have emerged in research literature. The Embedded Audit Module methodology developed early on in the domain's existence integrates continuous auditing functionality internally within the system of concern and operates in a truly real-time, continuous mode. Subsequently developed as an alternative to EAM, the Monitoring and Control Layer methodology focuses on an external module that interfaces with the system of concern to retrieve scheduled interfaces of selected data. Each of the continuous auditing approaches offers distinct advantages over the other.

Debreceny et al. (2005) examine the nature of EAM functionality and the relationship with ERP systems, perceived reasons for the lack of widespread adoption of EAM, and potential directions to increase the viability of the EAM approach to continuous auditing. In order to add additional clarity to the discussion, our study expands upon the limitations of EAM noted by Debreceny et al. (2005) and also explores the limitations of alternative continuous auditing approaches such as EAM ghosting and MCL. These limitations are examined from both technical and practical perspectives, as well as also examining the issues specifically from an audit perspective which carries major concerns of design and maintenance, client independence and auditor legal liability. We examined alternative strategies for implementation of continuous auditing, considering both (1) a modified EAM approach through the use of "ghosting" technology and (2) the MCL methodology. At present, MCL methodology appears to offer the most used alternative, as evidenced by both the recent increased attention in academic research (Alles et al. 2006; Kuhn and Sutton 2006; Alles et al. 2008) and the exclusive use of MCL for continuous auditing products developed by third-party vendors (AMR 2008).

As Debreceny et al. (2005) note, no prior research existed on EAM support within ERP systems at the time of their study. Only two additional research projects have surfaced since that point that examine continuous auditing in an ERP environment (i.e., Kuhn and Sutton 2006; Alles et al. 2006; Alles et al. 2008). Given the pervasiveness of ERP environments, additional research is necessary to advance the awareness, relevance, and practicality of ERP continuous auditing. In the course of discussing the issues fundamental to the use of EAM versus MCL, we have outlined a number of key research issues. The opportunities for research are plentiful and the need even greater when considered in light of wide ranging issues related to the potential demand for continuous monitoring and continuous auditing (e.g., Daigle and Lampe 2004, 2005), to the consideration of external auditor willingness and capability to implement and utilize continuous auditing strategies, to the transference and increasing reliance of internal control work performed by internal auditors, to the need for additional instantiations of continuous auditing applications in "real" environments (e.g., Alles et al. 2006; Kuhn and Sutton 2006), and to the examination of innovative integrated IT platforms that cross technologies. Exploration has only just begun on understanding the behavioral effects on an organization's employees and managers (e.g., Hunton et al. 2008), but these issues are also critical to anticipating the impact as these systems become more widespread. Opportunities abound for research facilitating the efficient and effective implementation and utilization of continuous auditing capabilities.

REFERENCES

Alles, M., A. Kogan, and M. A. Vasarhelyi. 2002. Feasibility and economics of continuous assurance. *Auditing: A Journal of Practice & Theory* 21 (1): 125–138.

———, G. Brennan, A. Kogan, and M. A. Vasarhelyi. 2006. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at Siemens. *International Journal of Accounting Information Systems* 7 (2): 137–161.

———, A. Kogan, and M. A. Vasarhelyi. 2008. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Journal of Information Systems* 22 (2): 195–214.

AMR Research, Inc. (AMR). 2008. Enterprise performance management: 2008 landscape series. Available at: http://www.approva.net/assets/resources/AMR_Research_REPORT_21828-The_Governance,_Risk_Management,_and_Com.pdf.

Behn, B. K., D. L. Searcy, and J. B. Woodroof. 2006. A within firm analysis of current and expected future audit lag determinants. *Journal of Information Systems* 20 (1): 65–86.

Braun, R. L., and H. E. Davis. 2003. Computer-assisted audit tools and techniques: analysis and perspectives. *Managerial Auditing Journal* 18 (9): 725–731.

Canadian Institute of Chartered Accountants and American Institute of Certified Public Accountants (CICA/AICPA). 1999. *Continuous Auditing. Research Report*. Toronto, Canada: CICA.

Chatterjee, D., R. Grewal, and V. Sambamurthy. 2002. Shaping up for e-commerce: Institutional enablers of the organizational assimilation of web technologies. *Management Information Systems Quarterly* 26 (2): 65–89.

Chen, H. H., S. H. Li, S. M. Huang, and Y. C. Hung. 2007. The development and performance evaluation of a continuous auditing assistance system. *International Journal of Electronic Finance* 1 (4): 460–472.

Daigle, R. J., and J. C. Lampe. 2004. The impact of the risk of consequence on the relative demand for continuous online assurance. *International Journal of Accounting Information Systems* 5 (3): 313–340.

———, and ———. 2005. The level of assurance precision and associated cost demanded when providing continuous online assurance in an environment open to assurance competition. *International Journal of Accounting Information Systems* 6 (2): 129–156.

Debreceny, R. S., G. L. Gray, W. L. Tham, K. Y. Goh, and P. L. Tang. 2003. The development of embedded audit modules to support continuous monitoring in the electronic commerce environment. *International Journal of Auditing* 7 (2): 169–185.

Debreceny, R. S., ———, J. Jun-Jin Ng, K. Siow-Ping Lee, and W. Yau. 2005. Embedded audit modules in enterprise resource planning systems: Implementation and functionality. *Journal of Information Systems* 19 (2): 7–27.

Deloitte. 2007. Security user segregation of duties—SAP GRC. Available at: http://www.Deloitte.com/dtt/case_study.

Dowling, C. 2009. Appropriate audit support system use: The influence of auditor, audit team and audit firm factors. *The Accounting Review*.

Ernst & Young. 2006. SEC and PCAOB roundtable discussion on implementation of internal control reporting provisions: Year Two. *Emerging Trends in Internal Controls* (May): 1–12.

Fogarty, J. 2005. *Continuous Auditing: Can it Become a Reality and What is its Impact on Auditing Standards*. Rutgers Tenth Continuous Auditing and Reporting Symposia. Newark, November.

Gerhardt, K. W. 2002. *The Emerging Role of the Information Systems Function in Modern Organizations:*