Summer 10-6-2011

INFORMATION FUSION IN CONTINUOUS ASSURANCE

Johan Perols

Uday Murthy

INFORMATION FUSION IN CONTINUOUS ASSURANCE

Perols, Johan, University of San Diego, 5998 Alcalá Park, San Diego, CA 92110, US, jperols@sandiego.edu

Murthy, Uday, University of South Florida, 4202 East Fowler Avenue, Tampa, FL 33620, US, umurthy@coba.usf.edu

Abstract

We extend continuous assurance research by proposing a novel continuous assurance architecture grounded in information fusion research. Existing continuous assurance architectures focus primarily on methods of monitoring assurance clients' systems to detect anomalous activities and have not addressed the question of how to process the detected anomalies. Consequently, actual implementations of these systems typically detect a large number of anomalies, with the resulting information overload leading to suboptimal decision making due to human information processing limitations. The proposed architecture addresses these issues by performing anomaly detection, aggregation and evaluation. Within the proposed architecture, artifacts developed in prior continuous assurance, ontology, and artificial intelligence research are used to perform the detection, aggregation and evaluation information fusion tasks. The architecture contributes to the academic continuous assurance literature and has implications for practitioners involved in the development of more robust and useful continuous assurance systems.

Key words: Continuous assurance, information fusion, machine learning, artificial intelligence, REA.

1 Introduction

Most organizations today rely heavily on computerized information systems to automate and control their business processes. The proliferation of technologies like Enterprise Resource Planning systems and the widespread acceptance of e-commerce standards like EDI, XML and XBRL have fundamentally changed the nature of the assurance function. The real-time availability of detailed data about business processes internally and more frequent reporting of financial information externally have together increased the pressure on auditors to provide continuous assurance in some form. Additionally, recent legislation such as SoX, HIPPA and GLBA, has increased the importance of providing continuous assurance of the design and effectiveness of controls throughout the year. Organizations covered by the Sarbanes-Oxley (SOX) Act of 2002 are now required to disclose material changes in financial condition as they occur.

Prior research has demonstrated how information technology can be harnessed to change both the type and timing of assurance services provided to users, specifically demonstrating techniques of performing "continuous auditing" (Groomer and Murthy 1989; Kogan et al. 1999; Rezaee, et al. 2002; Dull, et al. 2006). The AICPA and the Canadian Institute of Chartered Accountants define continuous auditing as "…a methodology that enables independent auditors to provide written assurance on a subject matter, for which an entity's management is responsible, using a series of auditors' reports issued virtually simultaneously with, or a short period of time after, the occurrence of events underlying the subject matter" (CICA/AICPA 1999). A recent report issued by PricewaterhouseCoopers indicates that organizations are indeed moving towards continuous assurance, with 81% of firms reporting that they either had a continuous auditing or monitoring process in place or were planning to develop one (PwC 2006). However, of these firms only 3% had fully automated continuous assurance capabilities and most firms were auditing on a frequent rather than continuous basis, i.e., quarterly (57%), monthly (34%) and daily (9%) (PwC 2006). Thus, industry is far from realizing the type of continuous assurance envisioned by policy makers and researchers where assurance is provided in real-time or close to real-time, i.e., on a continuous rather than frequent basis.

Academic research on continuous assurance has largely focused on the process of detecting assurance exceptions, i.e., deviations from a desired state such as a benchmark, best practice, policy or rule. Consequently, existing continuous assurance architectures (e.g., Groomer and Murthy 1989; Vasarhelyi and Halper 1991) do not address the important problem of processing detected exceptions. Unfortunately, behavioral decision theory and empirical research show that humans generally perform poorly in tasks requiring aggregation, processing, and analysis of cues (Tversky and Kahneman 1974; Iselin 1988; Kleinmuntz 1990). As reported by Alles, et al. (2006, 2008) and Debreceny et al. (2003), although the implemented continuous assurance systems were effective in detecting anomalies, there were simply too many anomalies generated for the users to process, leading to information overload. Thus, to the extent that the task of aggregation and analysis of detected exceptions is left to humans, the overall effectiveness and efficiency of any continuous auditing system will be limited.

In this research we follow the design science paradigm[1] and present the design of a novel architecture for continuous assurance that supports exception detection, aggregation and analysis. The new architecture, called Continuous Assurance Fusion (CAF), is grounded in both prior continuous assurance literature and computer science information fusion research. This architecture contributes to the literature by using information fusion concepts to address the issue of the processing and analysis of exceptions after they have been generated. Information fusion involves gathering data about an

[1] Please refer to Section 4.5 for a description of eight design science principles and an evaluation of this research based on these principles.

object of interest from multiple sources and integrating these data in order to arrive at a holistic conclusion about the object. While the architecture itself is based on prior information fusion research, specific architectural components are based on various research streams within the accounting and information systems fields. Specifically, prior continuous assurance research informs the data gathering component of the architecture, Resources-Events-Agents (REA) ontology concepts are used in the first data integration step, and artificial intelligence and machine learning algorithms are used in the subsequent data integration steps. This research provides a new direction for future continuous assurance research and a framework for integrating continuous assurance, artificial intelligence and REA ontology research. The CAF architecture itself, and subsequent research leveraging this architecture, can ultimately benefit practitioners through the development of continuous assurance systems that can effectively deal with detected exceptions.

2 Background and Related Research

Continuous assurance research has proceeded primarily along two paths, one focusing on the potential impact of continuous assurance on various stakeholders and the other focusing on technical issues in designing and implementing CA architectures. The latter research stream informs our proposed CA architecture. The vast majority of continuous assurance design research has largely focused on the process of detecting assurance exceptions, i.e. deviations from a desired state such as a benchmark, best practice, policy or rule. The detection of exceptions can be accomplished through data- and control-oriented continuous assurance procedures (Kogan, et al. 1999). In data-oriented procedures the raw business process data are audited for exceptions using detailed testing techniques, while in control-oriented procedures the correct operation of general and application computer controls are verified. The latter approach is often more efficient because the effective operation of general and application controls also provides assurance regarding the accuracy of the underlying business process data, but this approach requires that computerized controls are amenable to automated testing.

Two classes of architectures have been proposed in prior research to accomplish the exception detection task: (1) embedded audit modules; and (2) monitoring and control layer. Both architectures focus on the detection of exceptions and do not focus on the task of disposition or resolution of the exceptions after they have been detected. In embedded audit modules the exception detection logic is integrated within the accounting system, often relying on embedded code within the application or database triggers that are fired if an exception occurs (Groomer and Murthy 1989). In the Groomer and Murthy (1989) embedded audit modules approach, subroutines within application programs examine transaction data for exceptions and log detected exceptions in real-time in the database for further examination. Woodroof (2001) extended this idea and proposed a continuous auditing approach using digital agents, i.e., software agents to perform the exception detection and communication. More recently Debreceny, et al. (2003) implemented 10 embedded audit modules, as a proof-of-concept, into Microsoft Great Plains using stored procedures and triggers in the database.

The second type of continuous assurance architecture, the monitoring and control layer architecture, was first proposed by Vasarhelyi and Halper (1991). In the monitoring and control layer, the exception detection application is implemented as a standalone system that periodically queries the accounting system for raw transaction and control setting data. This information is then examined within the standalone system and flags are raised for detected exceptions. The architecture consists of eight elements: "(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" (Vasarhelyi et al. 2004, p. 12). Related to this architecture, Kuhn and Sutton (2006) proposed using the architecture specifically for detecting financial statement fraud and discussed how a continuous analytical monitoring model of certain "key-event-transactions" can be used to alert auditors of potential management fraud, while Murthy and Groomer (2004) proposed a new Web Services based monitoring and control layer architecture for business process monitoring that would facilitate and standardize the communication between assurance providers and clients. While not

monitoring transaction level data or internal controls for exceptions, nor monitoring data in real-time, Nigrini and Johnson (2008) report on an implementation where restaurants are scored on ten key variables using data from monthly sales reports to help internal auditors decide which restaurants to investigate. Alles et al. (2006, 2008) reported the implementation of an actual continuous assurance system using the monitoring and control layer architecture. This system focused on detecting control exceptions in business processes using control information from SAP applications at Siemens. In addition to focusing on exception detection, Alles et al. (2006, 2008) also discussed problems with too many alarms being generated. They dealt with this issue by making it easy to turn on and off entire groups of controls, as well as choosing who is notified by what alarms using a role-based approach.

Assurance providers perform tasks that require collecting, aggregating and analyzing information that is often distributed in nature, for example fraud detection, test of controls, assurance of accounts payable amounts and aging, and going-concern assessments. Existing continuous assurance architectures focus on the detection of exceptions and do not address how to process these exceptions further before being presented to human users, i.e., assurance professionals like auditors. Thus, existing continuous assurance architectures focus primarily on the collection of exception data, but not on the aggregation and analysis of exception data. The aggregation and analysis of the exceptions is assumed to be performed by the users of the continuous assurance system. Behavioral decision theory and empirical research, however, show that humans generally perform poorly in tasks requiring the combining of multiple cues (Kleinmuntz 1990), can be overloaded by too much information (Iselin 1988), and use heuristics such as representativeness, availability and anchoring-and-adjustment that introduce biases in the decision-making process (Tversky and Kahneman 1974). These information processing limitations constrain the effectiveness of a CA system that relies on humans to combine and analyze audit exception data.

A consequence of not including aggregation and analysis support in the CA system is evident in Alles et al. (2006, 2008) and Debreceny et al. (2003), the only studies that have reported on the implementation of actual CA systems. In both these implementations, the authors reported problems with too many alarms being generated. The continuous assurance systems were effective in detecting anomalies, but the detected anomalies led to information overload--there were simply too many anomalies generated for the users to process. Alles et al. (2006) addressed the alarm overload problem by developing a manual solution where entire groups of controls could be turned on and off. Thus, by ignoring what was deemed to be the least severe anomalies, the auditors could analyze and respond to the most important exceptions. However, assuming that continuous assurance systems are implemented to detect exceptions that indicate potential problems, ignoring some of the exceptions can result in the loss of valuable information. Debreceny et al. (2003) deferred the alarm overload problem to future research.

We extend existing continuous assurance architectures by proposing continuous assurance fusion (CAF)—an approach that specifically addresses the problem of exception handling with the goal of increasing the efficiency and effectiveness of continuous assurance services. We next provide a description of CAF and then provide an overview of the information fusion research stream within computer science that informs the design of CAF.

3 Continuous Assurance Fusion

CAF is grounded in continuous assurance architecture research and information fusion research, but also leverages artificial intelligence research and REA ontology research in accounting, and machine learning research in computer science. The overall architecture is designed based on information fusion models that use layered approaches for combining distributed information. More specifically, CAF has four major layers: event monitoring, information aggregation, object evaluation, and decision-making, as depicted in Figure 1.

The monitoring layer of CAF is based on prior continuous assurance architectures and performs essentially the same function as in prior CA architectures, i.e., anomaly detection. The other layers of the hierarchy are unique to CAF and provide functionality such as handling alarm floods and audit trail generation, information aggregation, object evaluation and decision-making. In the information aggregation layer, exceptions detected in the monitoring layer are used to generate object features by grouping exceptions based on their association to specific objects and computing additional object features. These object features are used by classifiers, i.e. statistical and machine learning algorithms like logistic regression and artificial neural networks, in the evaluation layer to make decisions about objects' class membership. Finally, the decision layer combines the individual classifier object classifications into an overall CAF object class membership decision. The four layers are described in greater detail in Section 3.

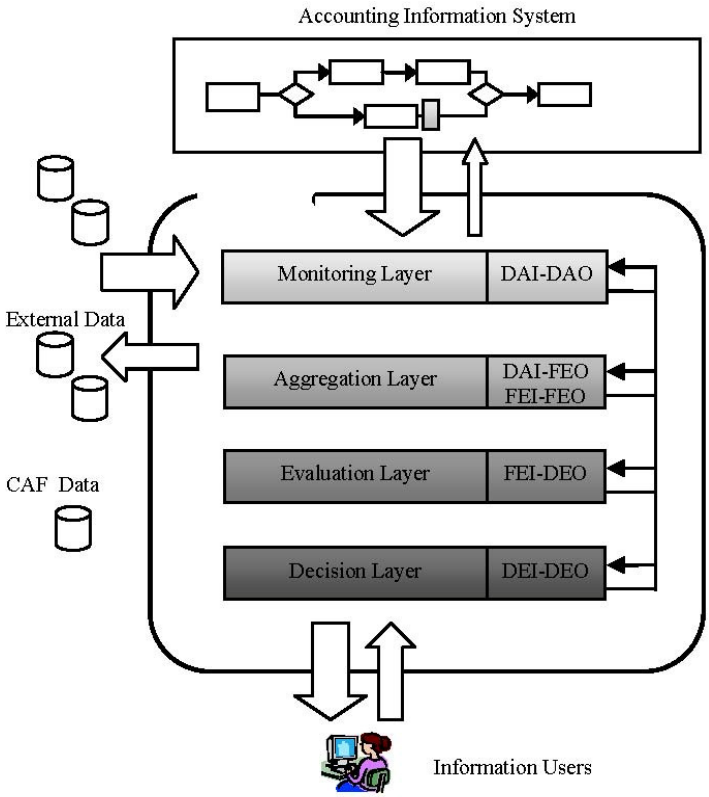

Figure 1 - CAF Conceptualization

3.1 Rationale for Using Information Fusion in Continuous Assurance

Information fusion is the process of producing estimates and knowledge about objects and situations based on data from various input sources (White 1987). The input sources are commonly sensors that capture different perspectives of the objects and situations of interest. The idea being that it is possible to get a more complete and accurate assessment of objects and situations if data from many sensors are combined and multiple models are used to evaluate these data. Although most distributed problems, where the data and/or decision makers are distributed, require some level of information fusion, the most commonly researched problem domains include: emergency response (natural catastrophe, terrorist attacks, etc); robotics; network security (intrusion detection systems); geoscience; and defense systems (Valet, et al. 2000). The three largest research areas employing information fusion concepts are defense systems, geoscience and robotics.

Assurance providers perform tasks that require collecting, aggregating and analyzing information that is often distributed in nature, for example fraud detection, test of controls, assurance of accounts payable amounts, and going-concern assessments. Within the computer science domain, information fusion research focuses specifically on the aggregation and analysis of data, features and decisions in distributed problem solving. Information fusion has been used in a wide variety of domains but has not yet been leveraged in accounting or auditing research. CAF extends prior continuous assurance research by applying ideas from the information fusion domain to the problem of dealing with large volumes of detected exceptions.

3.2 Information Fusion Models

As a consequence of information fusion having its roots in defense research, most early work in the field focused on defense systems. When information fusion started being researched and used for applications in additional domains, such as robotics and geoscience, the information fusion field became divided into two sub-communities, one broader that includes all application domains, including defense, and one that is primarily focused on defense applications (Bedworth 2000). The existing fusion models mirror this division with some models focusing more on defense systems, for example the JDL (White 1987) model, while others are framed in more general terms, notably the Dasarathy (1997) model. In our work, which falls into the latter group, we base CAF on the Dasarathy model and follow Dasarathy's definitions of the different fusion processes. Although we use the Dasarathy model to define CAF, there are also similarities between CAF and the JDL fusion model.

In the Dasarathy model, depicted in Figure 2, the fusion process is described based on the input (data, features or decisions) it takes and the output (data, features or decisions) it generates. The Dasarathy model does not, however, explain how the processes generate the output from the input. Rather, the model highlights the fact that different fusion processes exist and can be combined to improve the fusion system. Entire research streams are devoted to the development of specific fusion processes. For example, multi-classifier combination research focuses on the Decision In – Decision Out fusion process, taking decisions from an ensemble of base-classifiers as input and producing an ensemble decision as output using a combiner method.

The terms data, features, and decisions are not formally defined in Dasarathy (1997). The term data is used to refer to what is initially captured by sensors and has not yet been associated with a particular object (i.e., entity) of interest in the environment. Objects can exist at different levels of analysis, for example, in an audit where a transaction exception is detected, the object of interest might be the transaction itself, a control designed to mitigate risks in the transaction, the employee recording the transaction, or the manager approving the transaction. Thus, a single exception can be associated with multiple objects. Data can also belong to a particular object, but not yet be associated within the system to that object, and can even be processed in different ways but still be referred to as data (for example data can be combined to create new data). When the data are associated in the system with specific objects, the data become features of those objects. An object feature can be viewed as an attribute of the object, i.e., something that describes the object. For example, features of an employee recording a transaction might be the employee's assigned duties, authorization levels, name, gender, tenure, etc, while features of a transaction might be the time of occurrence, amount, employee recording the transaction, etc. Finally, decisions are opinions about an object that are based on features of that object. Decisions are often in regards to the class membership or probability of class membership of the object. For example, an object can be a member of the class clean opinion or adverse opinion, fraud or not fraud, debt covenant violated or debt covenant not violated, etc.

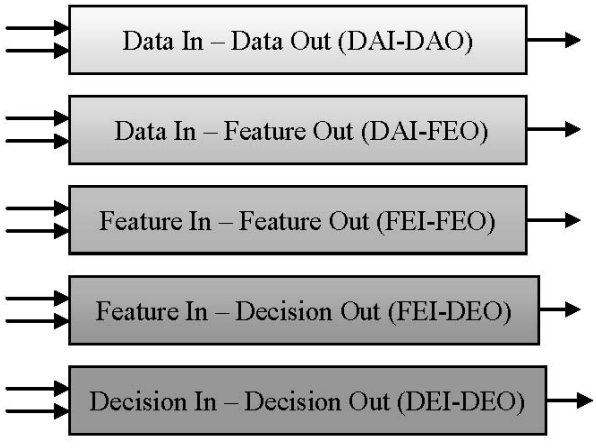

Figure 2 - Dasarathy Model

3.3 CAF and Information Fusion

From an information fusion perspective, embedded audit modules, and monitoring and control layer techniques used in prior continuous assurance research are sensors that detect anomalies in the

accounting environment by analyzing data and capturing exceptions. The exceptions, after they have been grouped and associated with a specific object of interest, become object features. However, before this association the exceptions are simply data. Thus, embedded audit modules and monitoring and control layer techniques perform data fusion and more specifically Data In – Data Out fusion. In prior research, the output data from the Data In – Data Out fusion was not processed further by the system. The main contribution of information fusion models lies in performing the additional steps to process the captured exceptions. In CAF, the additional fusion levels are defined using the Dasarathy model as Data In – Feature Out, Feature In – Feature Out, Feature In – Decision Out, and Decision In – Decision Out.

After the exceptions have been captured in the monitoring layer, the next step in CAF is to create features from the exceptions. In CAF the process of identifying features is done by aggregators that use domain-ontology or hard-coded knowledge to associate the exceptions with various objects and to define how objects at different levels of analysis are related. Aggregators can also perform Feature In – Feature Out fusion. For example, aggregate statistics like average, count, and standard deviation, which are features of an object, can be calculated based on existing object features.

Object evaluators then analyze the features output by the aggregators, and output object assessments at the decision level. Thus, object evaluators perform Feature In – Feature Out fusion. For example, artificial neural network and decision tree machine learners, and also logistic regression and discriminant analysis statistical functions, evaluate object features and output a probability assessment of object class membership. Finally, the decisions made by the different object evaluators are merged into an overall decision in a Decision In – Decision Out fusion process. The same techniques used in the Feature In – Decision Out fusion process can be used in the Decision In – Decision Out fusion process. In particular, so called meta-learners, or specialized ensemble base-classifier combiner methods can be leveraged.

The main advantage of CAF is in the automation of processes that are left to human auditors in existing CA architectures. Specifically, the automation of Data In – Feature Out and Feature In – Decision Out in CAF results in the automatic processing of audit exception data to arrive at meaningful conclusions, avoiding the "alarm flood" situations resulting from CA systems that only capture exceptions. As noted earlier, such automation is advantageous as computers generally outperform humans in processing large amounts of data. Further, by fusing the data into features and ultimately decisions, auditors' information processing load is reduced, thereby improving both the effectiveness and the efficiency of the system.[2] We now describe the CAF architecture in greater detail.

4 CAF Architecture

The CAF architecture consists of four layers: the monitoring layer where exceptions are captured, the data aggregation layer where exceptions are processed, the object evaluation layer where specific audit-relevant conclusions are drawn, and the decision making layer where an overall decision is reached, if appropriate and necessary. The functionality of each layer is now discussed.

4.1 Monitoring Layer

The monitoring layer consists of sensors that detect exceptions, anomalies and changes in the environment, such as the recording of a transaction that violates certain business rules, changes in a control settings table or news of a certain kind being released. Upon detection, the exception, anomaly

[2] There is no reason why the system could not also output the raw exceptions. This information could be used in parallel with CAF for manual review to reduce the risk of false negatives and to evaluate the effectiveness of the system.

and environment change data are processed and sent to the aggregation layer. Note that the monitoring layer does not put the detected data into any context, i.e., there is no "processing" of the exception and other audit relevant data performed at the monitoring layer. The monitoring layer thus performs only the Data In – Data Out fusion process.

For the continuous assurance to be conducted in real-time, these sensors would have to be embedded audit modules as opposed to monitoring and control layer sensors. However, when using embedded audit modules, in contrast to when the sensors are located in the monitoring and control layer, embedded audit modules require a high level of collaboration from transaction processing system owners and vendors and forces the continuous assurance system owners to cede some control over the monitoring layer (Alles et al. 2006). Thus, there is a trade-off between the two solutions and a combination of the two architectures provides the best solution. The exact combination of the two would be based on maximizing the net benefits of the assurance system, which is a function of domain and task specific costs and benefits. CAF might use database triggers (embedded audit modules) for monitoring process controls and general computer controls. For example, changes to purchase order approval thresholds and password controls could be monitored using database triggers. CAF can then query the host system (or a replicated database) on a frequent as opposed to continuous basis (monitoring and control layer) for data level monitoring. The data level monitoring could for example verify that all purchase orders have matching purchase requisitions. We refer the reader to Groomer and Murthy (1989) and Vasarhelyi and Halper (1991) for the seminal work on the two continuous assurance architectures that serve as the foundation for the monitoring layer of CAF, and Alles et al. (2006, 2008) and Debreceny et al. (2003) for examples of actual designs and implementations of these architectures.

To provide a more formal description of CAF, we use the monitoring of purchase transactions as an example. In this example a purchasing process p, i.e., processing of purchase order 1015, can be classified as in control (j=0) or out of control (j=1), where $j \in J$ given the index set of classes J={0,1}. Further, in the monitoring layer there are k monitoring modules m (software agents, database triggers, etc) in the index set M={1,…,k}. When monitoring module m detects a deviation it raises e_p, an exception e for object p, where e_p = {0,1} and $e \in E$ where E is a set of all exceptions being monitored for the process p. That is, e for p is either not raised (e_p = 0) or it is raised (e_p = 1). For example, m monitors that when p is above a certain threshold p has appropriate supervisor approval, and if this is not the case then creates exception e_p that signals that purchase order 1015 exceeds the threshold but is missing appropriate supervisory approval. Monitoring unit m can also collect i_p and i_{ep}, which is information i that describes p and e_p, respectively. Information i is not a known exception or anomaly type, i.e., $I \cap E = \emptyset$, where $i \in I$. Rather, information item i might represent non-exception information regarding the process or simply other relevant non-process information of audit relevance. For example, i can describe relevant information such as the purchase order number, order date, the purchasing agent that created the purchase order, etc and i_{ep} can describe information about the control activity such as the threshold that was violated, by how much it was exceeded, etc. Upon gathering e_p, i_p or i_{ep}, these items are sent to the aggregation layer.

4.2 Data Aggregation Layer

Within the data aggregation layer, CAF groups, aggregates and synthesizes data generated in the monitoring layer into object features that can be used in the Object Evaluation layer. The data aggregation layer determines the object type that the incoming information and exception data pertain to; assigns the input to a specific object; gathers additional missing features; computes additional object features; and establishes relations among the objects for object drill-down/roll-up. The assignment of information and exceptions to objects is a Data In – Feature Out fusion process, while the computation of additional features from existing features is a Feature In – Feature Out fusion process. Examples of the latter would be the calculation of object feature moving averages, standard deviations and financial ratios.

To aggregate and synthesize the monitoring layer exceptions into object features, the data aggregation layer can either utilize a domain ontology that defines how different domain specific objects are related and the features of these objects, or hard-code needed object-feature and object-object relations directly into each data aggregation module. The former alternative is more costly upfront, but at the same time offers greater flexibility and adaptability, and is probably preferable if the system is intended to be implemented using agent technologies. Whether a specific CAF actually needs a domain and/or CAF specific ontology is as such a cost-benefit decision that depends on the purpose of the specific CAF and the domain that it is designed for.

At a high level, an ontology consists of sets of constructs (objects, features and fusion modules are examples of constructs in CAF) and relations among these constructs that describe conceptualization of real world phenomena (Gruber 1993). Ontologies differ from theories in that they are used to describe, rather than explain or predict real world phenomena (Wand and Weber 2004) and differ from taxonomies in that they not only describe the constructs but also the relationships among the constructs. We propose that McCarthy's (1982) Resources, Events, and Agents (REA) ontology can serve as the basis for the CAF aggregation layer processing. Exceptions flagged at the monitoring layer could be grouped based on the REA ontology using the following scheme:

- Level 1: Exceptions grouped in terms of whether they relate to resources, events, or agents. That is, grouping in terms of 'R' exceptions, 'E' exceptions, or 'A' exceptions that relate to changes affecting a single R, E, or A object.
- Level 2: Exceptions grouped in terms of event-resource relationships. Specifically, exceptions involving illegal operations on resources by events (e.g., placing an order on an out-of-stock inventory item). These exceptions might stem from operations in different subsystems and hence might not be detected as exceptions within those subsystems individually.
- Level 3: Exceptions grouped in terms of event-agent relationships. Specifically, exceptions involving illegal operations by agents on events (i.e., a segregation of duties violation).
- Level 4: Exceptions grouped in terms of event-event relationships. Specifically, exceptions that violate rules of precedence between events (e.g., a payment to a vendor without a preceding receiving report and a purchase order).
- Level 5: Exceptions grouped in terms of resource-event-agent relationships. Specifically, exceptions that involve an illegal event executed by an agent that results in an invalid operation on a resource (e.g., a cashier processing a sales return that reduces cash and increases inventory).

At the aggregation layer, incoming exceptions are automatically processed by way of grouping using the five levels, as a form of "Data In – Feature Out" fusion where exceptions are associated with different R, E, or A objects. While our initial use of the REA ontology is only to group detected exceptions using a generally accepted business process ontology, we recognize that the REA ontology can support augmented intentional reasoning directed towards continuous assurance objectives (Geerts and McCarthy 2000). For such reasoning, however, the REA ontology would need to be extended with continuous assurance specific objects, i.e., definitions of a control, exception, etc, and constructs describing CAF system components, i.e., knowledge, communication, fusion process, etc. McCarthy et al. (2005) have proposed an extension to the REA ontology that addresses automatic enforcement of internal control procedures, but more research is needed in this area.

Continuing the purchase transactions monitoring example; in the aggregation layer there are s aggregators represented by indices a in the index set $A=\{1,\ldots,s\}$. Aggregator a, $a \in A$, receives e_p, i_p and i_{ep} from m, and requests additional features i_p, $i \in I$ and exception states e_p and information i_{ep}, $e \in E$, for purchasing transactions p, as needed. Using i_p, i_{ep} and e_p, aggregator a completes the Data In – Feature Out fusion, and generates feature set f_a for the specific object that the system is aggregating information about, where $f_a = i_{ep} \cup i_p \cup e_p$. For example, the aggregator a might aggregate information on different types of R, such as the type of inventory/services being purchased, the type of payment used, etc., E, such as the processing of transaction 1105, the approval process itself, the purchase order preparation process, etc, or A, such as the purchasing agent that created the purchase order without

appropriate approval, the supervisor, the vendor, etc, objects. In this example we will assume that a is aggregating information about the approval process itself. Aggregator a then sends feature set f_a to the object evaluation layer. The aggregator knows what features f_a should contain before f_a is sent to a specific evaluator c. This feature set might contain things such as the number of threshold violations reported in the past week, the trend in report violations, the time of day the violation occurred, whether the violation occurred on a weekend or a holiday, by how much the threshold was violated, how long since the control was implemented, etc.

4.3 Object Evaluation Layer

The object evaluation layer performs Feature In – Decision Out fusion. This layer receives input from the aggregation layer and provides decision output to the decision layer. To clarify, consider the example of an audit of the purchasing subsystem. The input from the aggregation layer would be various control anomaly exceptions, for example instances where payments were made for invoices without matching purchase orders (an event-event exception), or instances where an unauthorized agent executed a purchase return resulting in an invalid increase in returned merchandise inventory (a resource-event-agent exception). Classifiers, i.e., machine learning models and statistical models, would evaluate these features along with other object features and generate probability estimates about the "in control" or "out of control" state of the auditee's purchasing subsystem. For example, an artificial neural network (ANN), after being trained, would use these features as input and make a probabilistic decision about the state of the auditee's purchasing subsystem. To train the ANN, prior known "out of control" and "in control" cases would be used by the ANN along with features used to describe the purchasing subsystem. The ANN would use these data to learn complex relations between the features (input data) and the known outcomes by adjusting weights placed on different potential interneuron connections.

Various classification algorithms proposed in both the machine learning domain, for example neural networks, support vector machines and decision trees, and in the statistics domain, for example logistic regression and discriminant analysis, can be used in the evaluation layer for Feature In – Decision Out fusion. In addition to using learning algorithms (that require data to learn), artificial intelligence based expert systems using hard-coded if-then type rules or simple heuristics could also be used to perform the Feature In – Decision Out fusion. Note that these algorithms have been used in prior artificial intelligence accounting research to improve the classification performance in various accounting classification problems, i.e., going concern decisions, financial statement fraud, control risk assessments, etc. (Calderon and Cheh 2002). More directly related to continuous assurance, Koskivaara and Back (2007) implemented an ANN for continuous auditing of financial data, while Lee and Han (2000) used a fuzzy cognitive map to model the causal relationships among various control components and the effect of those components on overall system performance. However, while prior research has developed algorithms that are readily available to be used in the object evaluation layer and that have been used for related classification problems, there is a need and the time is ripe for research that integrates artificial intelligence accounting research with continuous assurance research (Kuhn and Sutton 2006). CAF fills this void by providing the architecture for directly integrating these two research streams.

Continuing with the purchasing process example; in the evaluation layer, evaluators $c \in C$ use the feature vector f_a associated with aggregated information for object a to determine the posterior probability estimate q_{caj} in [0, 1] that a belongs to class $j \in J$. For example, the system uses an already trained artificial neural network model that takes the feature vector f_a as input and outputs a likelihood q_{caj} that a, the approval process, is ineffective. Note that $q_{caj} = \{0,1\}$ for evaluators c that output crisp decisions, i.e., the decisions are in nominal rather than continuous form. These individual decisions q_{caj} are then sent to the decision making layer where a decision maker combines the individual decisions q_{caj} into CAF's overall probability estimate Q_{sj} in [0, 1] that system S, i.e., the entire accounting information system or sub-component of this system, belongs to class j.

4.4 Decision Making Layer

The decision layer is the highest layer in the CAF architecture. This layer takes object evaluation decision output as input and makes a system wide object classification decision. The decision maker in CAF performs Decision In – Decision Out fusion by combining the decisions from the evaluation layer into an overall decision. This layer has, to our knowledge, not received any attention in the accounting, auditing, or specifically the continuous auditing literature. There are, however, other research streams that focus on decision fusion, for example multi-classifier combination and meta-learning in data mining, that can be leveraged in the decision layer of CAF.

In multi-classifier combination, the outputs from a group or ensemble of classifiers (i.e., the object evaluators in CAF) are combined to improve the classification performance in various classification problems (Suen and Lam 2000). These individual classifiers, commonly referred to as base-classifiers, classify objects based on inputs consisting of object feature vectors. The base-classifiers' classifications or decisions are then combined using a combiner method into a single decision about an object's class label. In the context of CA at the business process level, the "single overall decision" could pertain to whether the auditee's information system as a whole or a specific sub-system is functioning reliably, or error-free. The basic idea behind multi-classifier combination is that different classifiers in an ensemble have different strengths and weaknesses, and therefore provide complementary information about the classification problem. These differences can be leveraged to improve classification performance by combining base-classifiers' decisions (Kittler, et al. 1998). Again, in the context of CA at the business process level, there could be exceptions of varying severity that occur in individual subsystems, but it is at the final decision layer in CAF that an overall conclusion is drawn about whether the auditee's overall information system or sub-system is functioning properly.

In the context of CAF, the purpose of (or potential advantage of) Decision In – Decision Out fusion in the decision layer is perhaps less clear. In general, base-classifier ensemble research has shown that base-classifiers, i.e., object evaluators in CAF, often make different types of errors and that it is therefore possible to improve performance by combining decisions from multiple base-classifiers (Kittler, et al. 1998). Additionally, Decision In – Decision Out fusion allows different object evaluators to specialize in different decision tasks and increases redundancy. The use of Decision In – Decision Out fusion could also reduce the possibility of a single part of the system becoming a bottleneck and could allow multiple decisions to be made in parallel (Lee et al. 2000).

Continuing the purchasing transactions example; decision maker d receives the posterior probability estimate q_{cpj} from evaluators $c \in C$ in the objective evaluation layer. Through a Decision In – Decision Out fusion process, d combines the individual q_{cpj} into an overall system reliability decision system S, i.e., probability estimate Q_{sj} that system S belongs to class j. The decision Q_{sj} can be evaluated against threshold h and if $Q_{sj} > h$ then S is classified as "out of control." The threshold can be set using Markovian or Bayesian control concepts (Dittman and Prakash 1979). When S is classified as "out of control" then CAF would notify the assurance professional and provide Q_{sj} and feature vector f_{pj}. Alternatively, Q_{sj}, and all q_p and f_{pj} can be output to the assurance professional, who would then decide which aspect of the purchasing system should be investigated further. This alternative might be preferred when for example the user wants more control over the selection process, costs associated with false positive and false negative classifications are unknown, or the assurance professional chooses to investigate as many cases as possible.

Note that in both CAF and in existing continuous assurance architectures the assurance professional has to investigate the potential fraud. However, in existing continuous assurance architectures, as opposed to CAF, the assurance professional has to additionally aggregate and evaluate raw exceptions, form opinions about the objects of interest, and select objects to investigate before actually investigating the object. These tasks require a tremendous amount of work on the part of the assurance professional, which has resulted in information overload in existing implementations (Debreceny, et

al. 2003; Alles, et al. 2006, 2008). By performing these information processing steps, CAF improves the efficiency of the overall CA system. Further, due to human information processing limitations, which lead to the use of heuristics that can introduce biases, CAF also has the potential to improve the effectiveness of the assurance.

4.5 Evaluation of CAF architecture

This research follows the design science paradigm in proposing a new architecture for CA, employing information fusion concepts. To evaluate our CAF approach, we apply the design science evaluation guidelines proposed by Hevner et al. (2004). The seven guidelines are: (1) design as an artifact; (2) problem relevance; (3) design evaluation; (4) research contribution; (5) research rigor; (6) design as a search process; and (7) communication of research. As Hevner et al. (2004) note, these guidelines are not to be used as a static set of absolute criteria, rather they are guidelines for assisting in evaluating design science research while exercising judgment in the process. In this section we use the seven Hevner et al. (2004) guidelines to provide a self-assessment of our CAF approach.

By extending existing continuous assurance research, through the integration of ideas from this research with information fusion research, and to some extent machine learning and REA ontology research, CAF is a novel IT artifact that provides a solution to the information overload problem noted in prior continuous assurance research (Debreceny, et al. 2003; Alles, et al. 2006, 2008. The CAF artifact improves the effectiveness of extant CA systems by automatically aggregating and processing detected exceptions, which is a significant advantage given that humans have a limited ability to process and integrate large volumes of data (Tversky and Kahneman 1974; Iselin 1988; Kleinmuntz 1990). The solution to this problem is of importance given the increasing need to be able to provide continuous assurance, a need that has been acknowledge by researchers over the years (Groomer and Murthy 1989; Rezaee, et al. 2002; Dull, et al. 2006), institutions (CICA/AICPA 1999) and practitioners alike (Woodroof and Searcy 2001; PwC 2006).

The proposed IT artifact, CAF, thus provides a novel solution (guideline 1: design as an artifact) and also addresses a relevant problem (guideline 2: problem relevance). The development of CAF contributes (guideline 4: research contribution) to both research and practice. CAF extends research by providing a solution to a continuous assurance problem and thus adds to the existing design science research knowledgebase. Furthermore, given that (1) the basis for the lower fusion levels of CAF have been implemented in prior continuous assurance projects (Data In – Data Out fusion) (Debreceny et al. 2003; Alles et al. 2006, 2008), (2) the basis for the upper fusion levels of CAF (Feature In – Decision Out, and Decision In – Decision Out fusion) have been implemented in prior machine learning accounting research (Suen and Lam 2000; Calderon and Cheh 2002; Lee and Han 2000), and (3) information fusion architectures have successfully been used in a wide variety of domains (Dasarathy 1997; Bedworth 2000), this research also contributes to practice by providing an implementable continuous assurance architecture that organizations can use when developing continuous assurance systems.

CAF solves an accounting problem by integrating existing architectures and models from the knowledgebase developed by continuous assurance research and information fusion research. We argue that in addition to being implementable, evidence from prior research of the utility provided by the various components of CAF, i.e., the monitoring layer (Debreceny et al. 2003; Alles et al. 2006, 2008), the object evaluation layer (Calderon and Cheh 2002; Lee and Han 2000), and the decision layer (Suen and Lam 2000), and their integration, i.e., the information fusion architecture (Dasarathy 1997), serves as evidence of the utility of CAF (guideline 3: design evaluation). The true value of CAF, however, will not be known until a system is built, implement and evaluated in a real-world setting. We consider this evaluation to be outside of the scope of this research and a valuable avenue for future research and CA practitioners. Furthermore, by leveraging existing research in information fusion and REA ontology, the research rigor is enhanced. Additionally, the descriptions of CAF

include both narratives and set theory notations to further improve the research rigor (guideline 5: research rigor).

When examining various information fusion frameworks for fit with the continuous assurance problem domain we conducted an informed search process to select a specific solution among many possible (guideline 6: design as a search process). Furthermore, by leveraging existing continuous assurance research and extending this research, our research is part of a larger well established domain of research aimed at designing and developing effective and efficient continuous assurance systems. Finally, the research has been communicated at a level that we believe is appropriate for this journal's audience without being too technical or too general. Through the use of figures, descriptions and notations, we have sought to clearly articulate the CAF architecture (guideline 7: communication of research). Accordingly, we submit that the CAF approach proposed in this paper adequately conforms to the Hevner et al. (2004) evaluation guidelines.

5 Discussion and Conclusion

In this paper, we present a novel architecture for continuous assurance that addresses one of the key outstanding problems in the CA literature, that is, a mechanism for dealing with detected exceptions. As actual CA implementations have revealed, when the systems operate as designed they can produce an overwhelming number of exceptions causing information overload for assurance professionals tasked with handling and parsing the exceptions. Put simply, there is a need for "intelligent summarization" of detected audit exceptions, such that assurance professionals have a logical basis for targeting their efforts to areas that are most in need of their attention. The architecture presented in this paper, referred to as Continuous Assurance Fusion, performs intelligent summarization of audit exceptions through algorithmic aggregation and analysis. The architecture leverages prior research in continuous assurance, computer science, and information fusion research. The main idea behind information fusion is to collect and analyze data from multiple sources regarding an object of interest to eventually draw some conclusions about the state of the object.

A strength of the CAF architecture is that it leverages prior research in accounting and information systems, in both continuous assurance and REA. The data gathering component of the architecture relies on previous continuous assurance architectures, while the data aggregation component is based on REA ontology concepts to categorize exceptions. Specifically, focusing on exceptions related to resources, events, agents, and the relationships between them, the data aggregation component is able to group exceptions in terms of the degree of severity. Beyond the initial data aggregation step, techniques in artificial intelligence and machine learning are used to make inferences about the states of assurance objects of interest. For example, if the object of interest is the accounts receivable amount, an appropriately designed CAF that monitors the credit sales, sales returns, and collections from customers subsystems could aggregate and analyze all audit exceptions dealing with these subsystems and apply appropriate machine learning algorithms to arrive at an overall conclusion regarding whether the accounts receivable account is misstated. In this manner, the CAF architecture we present moves beyond the process of merely detecting exceptions to actually process them with the goal of drawing meaningful audit-relevant conclusions.

This paper contributes to the CA literature by introducing the notion of information fusion and describing an architecture that blends concepts of information fusion, REA ontology, artificial neural networks, and machine learning. Although we do not present an actual implementation of a CAF system, we contend that the architectural details presented in this paper are of sufficient detail to guide a CAF implementation. Furthermore, there is sufficient evidence in prior research and practice regarding the successful implementation of individual components of CAF, specifically the monitoring layer (Debreceny et al. 2003; Alles et al. 2006, 2008), the object evaluation layer (Calderon and Cheh 2002; Lee and Han 2000), and the decision layer (Suen and Lam 2000). Future research in a design science vein could focus on the development of a functioning CAF system, to demonstrate the process of aggregating and analyzing exception data to arrive at an overall holistic conclusion about the target

system. Practitioners engaged in designing and implementing CA systems should be particularly interested in the CAF architecture, as it addresses the key unresolved issue in the existing CA literature, i.e., the process of handling large volumes of detected exceptions through a process of intelligent summarization to guide further audit investigation.

Future research could also investigate the design and implementation of CAF using agent-based technologies. For example, a CAF system could be described using the multi-agent-based integrative business modeling language (MibML) grammar (Kishore et al., 2006) and ontology (Zhang et al., 2007). Another avenue for future research is to investigate the relative efficacy of CAF for tasks such as employee fraud detection, assessment of internal control systems, or financial statement fraud detection. To conclude, the main contribution of this paper is in presenting a novel architecture called Continuous Assurance Fusion, which blends concepts of information fusion, REA ontology, artificial intelligence, and machine learning to address the problem of handling large volumes of detected audit exceptions.

References

Alles, M. G., 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.

Alles, M. G., 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.

Bedworth, M. (2000). The Omnibus Model: a New Model of Data Fusion? IEEE Aerospace and Electronic Systems Magazine 15 (4): 30-36.

Calderon, T. G., and J. J. Cheh. (2002). A Roadmap for Future Neural Networks Research in Auditing and Risk Assessment. International Journal of Accounting Information Systems 3 (4): 203-236.

Canadian Institute of Chartered Accountants/American Institute of Certified Public Accountants. 1999. Continuous Auditing. Research Report, Toronto, Canada.

Dasarathy, B. V. (1997). Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications. Proceedings of the IEEE 85 (1): 24-38.

Debreceny, R., 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.

Dittman, D., and P. Prakash. (1979). Cost Variance Investigation: Markovian Control versus Optimal Control. The Accounting Review 54 (2): 358-373.

Dull, R. B., D. P. Tegarden, and L. L. F. Schleifer. (2006). ACTVE: A Proposal for an Automated Continuous Transaction Verification Environment. Journal of Emerging Technologies in Accounting 3 (1): 81-96.

Geerts, G.L., and W.E. McCarthy. (2000). Augmented Intensional Reasoning in Knowledge-Based Accounting Systems. Journal of Information Systems 14 (2): 127-150.

Groomer, S. M., and U. S. Murthy. (1989). Continuous Auditing of Database Applications: An Embedded Audit Module Approach. Journal of Information Systems 3 (2): 53-69.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (2): 199-220.

Hevner, A. R., S. T. March, J. Park, and S. Ram. (2004). Design Science in Information Systems Research. MIS Quarterly (28) 1: 75-105.

Iselin, E. R. (1988). The Effects of Information Load and Information Diversity on Decision Quality in a Structured Decision Task. Accounting, Organizations and Society 13 (2): 147-164.

Kishore, R., H. Zhang, and R. Ramesh. (2006). Enterprise Integration Using the Agent Paradigm: Foundations of Multi-Agent-Based Integrative Business Information Systems. Decision Support Systems 42 (1): 48-78.

Kittler, J., M. Hatef, R. P. W. Duin, and J. Matas. (1998). On Combining Classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3): 226-239.
Kleinmuntz, B. (1990). Why we still Use Our Heads Instead of Formulas: Toward an Integrative Approach. Psychological Bulletin 107 (3): 296-310.
Kogan, A., E. F. Sudit, and M. A. Vasarhelyi. (1999). Continuous Online Auditing: A Program of Research. Journal of Information Systems 13 (2): 87-103.
Koskivaara, E., and B. Back. (2007). Artificial Neural Network Assistant (ANNA) for Continuous Auditing and Monitoring of Financial Data. Journal of Emerging Technologies in Accounting (4): 29-45.
Kuhn, J. R., and G. S. Sutton. (2006). Learning from WorldCom: Implications for Fraud Detection through Continuous Assurance. Journal of Emerging Technologies in Accounting 3 (1): 61-80.
Lee, S., and I. Han. (2000). Fuzzy Cognitive Map for the Design of EDI Controls. Information & Management 37: 37-50.
Lee, W., S. J. Stolfo, and K. W. Mok. (2000). Adaptive Intrusion Detection: A Data Mining Approach. Artificial Intelligence Review 14 (6): 533-567.
McCarthy, W. E. (1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review 57 (3): 554-578.
McCarthy, W. E., G. Gal, and G. Geerts. 2005. Semantic Specification and Automated Enforcement of Internal Control Procedures within Accounting Systems. 10th World Continuous Auditing Conference, November.
Murthy, U. S., and S. M. Groomer (2004). A Continuous Auditing Web Services (CAWS) Model for XML Based Accounting Systems. International Journal of Accounting Information Systems 5 (2): 139-163.
Nigrini, M. J. and A. J. Johnson. 2008. Using Key Performance Indicators and Risk Measures in Continuous Monitoring. Journal of Emerging Technologies in Accounting 5: 65-80.
PricewaterhouseCoopers. (2006). State of the Internal Audit Profession. New York, NY.
Rezaee, Z., A. Sharbatoghlie, R. Elam, and P. L. McMickle. (2002). Continuous Auditing: Building Automated Auditing Capability. Auditing: A Journal of Theory and Practice 21 (1): 147-164.
Suen, C. Y., and L. Lam. (2000). Multiple Classifier Combination Methodologies for Different Output Levels. Proceedings of First International Workshop on Multiple Classifier Systems pp. 52–66.
Tversky, A., and D. Kahneman. (1974). Judgment under Uncertainty: Heuristics and Biases. Science 185 (4157): 1124-1131.
Valet, L., G. Mauris, and P. Bolon. (2000). A Statistical Overview of Recent Literature in Information Fusion. Proceedings of the Third International Conference on Information Fusion 1: 22-29.
Vasarhelyi, M. A., M. G. Alles, and A. Kogan. (2004). Principles of Analytic Monitoring for Continuous Assurance. Journal of Emerging Technologies in Accounting 1 (1): 1-21.
Vasarhelyi, M. A., and F. B. Halper. (1991). The continuous audit of online systems. Auditing: A Journal of Practice and Theory 10 (1): 110-125.
Wand, Y., and R. Weber. (2004). Reflection: Ontology in Information Systems. Journal of Database Management 15 (2).
White, F. (1987). Data fusion lexicon. Joint Directors of Laboratories, Technical Panel for C3, Data Fusion Subpanel, Naval Ocean Systems Center, San Diego, CA.
Woodroof, J., and D. Searcy. (2001). Continuous Audit-Model Development and Implementation Within a Debt Covenant Compliance Domain. International Journal of Accounting Information Systems 2 (3): 169-191.
Zhang, H., R. Kishore, and R. Ramesh. (2007). Semantics of the MibML Conceptual Modeling Grammar: An Ontological Analysis Using the Bunge-Wang-Weber Framework. Journal of Database Management 18 (1): 1-19.