

2012

Towards a Continuous Process Auditing Framework (Case study in Healthcare Auditing and Decision Support - Infection Regime Control Survey)

Atif Zahid
University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Zahid, Atif, "Towards a Continuous Process Auditing Framework (Case study in Healthcare Auditing and Decision Support - Infection Regime Control Survey)" (2012). *Electronic Theses and Dissertations*. 346.
<https://scholar.uwindsor.ca/etd/346>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Towards a Continuous Process Auditing Framework (Case study in Healthcare Auditing
and Decision Support - Infection Regime Control Survey)

by

Atif Hasan Zahid

A Thesis
Submitted to the Faculty of Graduate Studies
through Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2012

© 2012 Atif Hasan Zahid

Towards a Continuous Process Auditing Framework (Case Study in Healthcare Auditing
and Decision Support - Infection Control Regime Survey)

by

Atif Hasan Zahid

APPROVED BY:

Dr. Gokul Bhandari
Odette School of Business

Dr. Scott Goodwin
School of Computer Science

Dr. Robert Kent, Advisor
School of Computer Science

Dr. Subir Bandyopadhyay, Chair of Defence
School of Computer Science

May 25, 2012

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The complexity of modern digital information systems is continuously increasing as a by-product of increased system functionalities in various domains. The resultant sea of information at our disposal mandates for making informed, timely and verifiable decisions. Traditional human-based audits become a liability in pursuit of timely decisions as they fail to audit the individual process modules or the entire process chain critical for determining the efficiency of an entire complex system.

In this thesis we introduce the concept of Continuous Process Auditing (CPA) in digital systems. We propose an approach that audits the methodologies (processes) in a system used to achieve results. We use a communication mechanism and employ a weighting schema that accounts for the holistic nature of process chains and provides decision support to select alternate strategies to improve system efficiency. To demonstrate our approach we provide a case study based on auditing a survey application and present our results.

DEDICATION

This thesis is dedicated to my family for the encouragement and support of a life time. It is dedicated to my wife for supporting me through the thick and thin. It is also dedicated to my loving father who has looked after me since the day I was born, who has spent his lifetime savings in order for me to be here at this moment and who still provides me with whatever I need even before I ask. It is also dedicated to my beloved mother, who has cared for me every moment of my life, who encourages me whenever I am down and whose prayers have played a big part in making me the person I am today.

ACKNOWLEDGEMENTS

I would like to express my gratitude to all the people who have helped me during my master's study. First and foremost, I am thankful to Almighty God as without His help and mercy, I would have never achieved a thing in my life. I would like to thank my supervisor, Dr. Robert Kent, for his guidance throughout the course of my studies.

Without his constant supervision, value recommendations and continuous appetite for pushing the boundaries beyond imagination, this thesis would not have been possible.

What I have learned from Dr. Kent goes beyond this thesis and I thank Dr. Kent from the bottom of my heart for teaching that would stay with me for a lifetime, for being a great mentor, teacher and a father. I would also like to express my gratitude for the committee members; Dr. Gokul Bhandari and Dr. Scott Goodwin, their time and suggestions made this thesis a better work.

I would also like to make a special mention of Paul Preney for all his help throughout my Master's study. His invaluable insights over the years have had a deep impact on this thesis and I thank Paul for his ever ready helping nature. I would also like to thank Bryan St. Amour for his valuable suggestions during the design phase of the experiments and all the unknown C++ guru's who take time out of their lives to sit on forums and help people in need. It would not have been possible to finish this work without their help and I am grateful to them for this.

Lastly, I would also like to thank every individual in this world who prayed for my success and most importantly, my family who encouraged me every step of the way during my study.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	iii
ABSTRACT	iv
DEDICATION	v
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER	
I. INTRODUCTION	12
1.1 Audit - Formal Definition	13
1.2 Continuous Auditing	15
1.3 Problem Statement	16
1.4 Contribution	16
1.5 Organization of the thesis	17
CHAPTER	
II. LITERATURE REVIEW	18
2.1 Continuous Auditing Models	18
2.2 Continuous Auditing techniques	26
2.3 Continuous Auditing Components	29
2.4 Procedure Audit vs. Process Audit	30
CHAPTER	
III. PROPOSED METHOD	32
3.1 Continuous Process Auditing	32
3.2 Proposed CPA Framework	35
3.2.1 Sticky Log	41
3.2.2 Weight Schema	45
3.2.3 Master Document	48
3.2.4 Analysis and Reporting	51

CHAPTER

IV. CASE STUDY AND DISCUSSION	53
4.1 Background	53
4.2 Implementation	54
4.2.1 Infection Regime Control Survey	54
4.2.2 Sticky Log Implementation	57
4.2.3 Master Document Implementation	60
4.2.4 Audit Process Implementation	61
4.3 Verifications and Results	64
4.3.1 Verification of Controls	65
4.3.2 Verification of Core Elements	71
4.3.3 Verification of Audit Process	74
4.4 Summary Comments	87

CHAPTER

V. CONCLUSION AND FUTURE WORK	88
5.1 Conclusions	88
5.2 Future Work	90

APPENDICES	92
Appendix A : Verification Test Results	92

BIBLIOGRAPY	161
--------------------	------------

VITA AUCTORIS	165
----------------------	------------

LIST OF TABLES

2.1 Procedure Audit vs. Process Audit	31
---	----

LIST OF FIGURES

2.1 Generic auditing architecture	22
3.1 Overall architecture of continuous process auditing model.....	36
3.2 Workflow of steps performed in CPA engine	38
3.3 Pseudo code for procedural auditing	39
3.4 Sample objective pseudo code	40
3.5 Flowchart of steps performed in CPA engine grouped as sub-processes	41
3.6 Sample predicate set	50
4.1 User Control	54
4.2 Sample survey with processes	55
4.3 Sample survey with expanded process	56
4.4 Sample response XML file.....	57
4.5 Sample slog document	58
4.6 Sticky log updating process.....	60
4.7 Sample master document XML file.....	61
4.8 Data extraction pseudo code	62
4.9 Overall audit pseudo code.....	64
4.10 User input with missing mandatory field	67
4.11 Audit report for a survey with missing mandatory field	68
4.12 CSV file with deliberate incorrect answers.....	69
4.13 Audit report for a survey with incorrect answers beyond threshold.....	70
4.14 Sticky log for survey.....	73
4.15 Master Document for survey.....	73

4.16 Sample audit result for processes and questions	76
4.17 Sample overall audit result that passed	78
4.18 Sample audit result for failed process audit and question audit	79
4.19 Sample overall audit result that failed	80
4.20 Sample audit result identifying success factors.....	82
4.21 Sample audit result for alternate strategy – Pass 1	84
4.22 Sample audit result for alternate strategy- Pass 10.....	85
4.23 Sample audit result for alternate strategy – Pass11	86

CHAPTER I

INTRODUCTION

In the electronic world of today, we are moving towards fast adaptation of technology in every field of life. This adaptation has allowed us to move from registers to computers and from offline systems to real time systems. Internet has changed the way business is conducted and digital data transmission has been facilitated by the continuous decrease in hardware prices coupled with ever increasing transmission speed. Information technology advancement has changed the way organizations conduct business and it has become imperative to make good timely decisions. Electronic data is not only timely and precise, it is also easy to store and access. These real time systems need new techniques to make sure that they are working properly. It is imperative to make sure that these systems are properly doing what they are supposed to do.

As the complexity of modern digital information systems increases as a by-product of increased system functionalities in various domains, including healthcare, financial, transportation and communication, we have a sea of information at our disposal to make informed, timely and verifiable decisions. An assessment mechanism that provides this verification is required to fully reap the benefits of this information. The

assessment of system behaviour must account for individual modules. Also, entire system functionality must be assessed to determine if the system is working desirably or inefficient. Computer systems offer sufficient power to handle the computational needs of complex system monitoring and analysis in real time. By definition, audit is a process of evaluation [1]. Hence auditing becomes an integral part of such systems.

The word 'audit' is derived from Latin word 'audire' which literally means 'to hear' [2]. During the early days, business owners would appoint an independent person who would hear verbal explanations from the book keepers. They would then judge the facts and announce the results. The aim of this audit was to find if any cash has been embezzled and if so, the person responsible.

1.1 Audit – Formal Definition

Although a precise definition of audit is difficult to provide, there are several definitions given by different authors according to whom an audit is

- an examination of accounting records undertaken with a view to establishing whether they correctly and completely reflect the transactions to which they purport to relate (Lawrence R. Dicksee)
- an examination of such records to establish their reliability and the reliability of statements drawn from them (A. W. Hanson)
- an examination intended to serve as a basis for an expression of opinion regarding the fairness, consistency and conformity with accepted accounting principles, of statements prepared by a corporation or other entity (American Institute of Accountants)

- an examination of the records and reports of an enterprise by accounting specialists other than those responsible for their preparation (Britannica Encyclopaedia)
- an evaluation of a person, organization, system, process, enterprise, project or product (Wikipedia)
- a methodical examination or review of a condition or situation (Merriam-Webster)
- a systematic review or assessment of something (Oxford)

The various definitions, although rooted in business and financial contexts, can be considered as general in nature. Therefore, they are applicable to any domain where records are taken, representative of some processes and their interactions, and, with a suitable analytical and decision making framework, opinions are rendered to support findings of the reliability and trustworthiness of outcomes with respect to objectives. The aim is to determine the validity and reliability of information and assess the system's ability to deliver according to its intended requirements. Another objective is to find errors and misuse of data whether unintended or deliberate and provide assurance about the work accomplished. Since businesses are quickly adopting real time stature, traditional auditing methods, which are carried out on a quarterly or yearly basis, kill an important objective of moving to real time systems .i.e. make a timely and informed decision. Traditional auditing methods take a long time to provide the results thus multiplying the potential loss and incur under utilization of human power during off season. A mechanism is required that allows a timely evaluation of such systems so that correct decisions can be made quickly. There is an ever increasing need for assurance

over business integrity which has been evolving constantly in our industry and mandated by Sarbanes-Oxley Act (SOX 2002) [3] which made Corporate Governance a MUST i.e. processes/policies/behaviours must also be audited in addition to financial data. Over the last decade or so, more and more scandals and corporate frauds have led to enormous changes and need for robust control systems.

1.2 Continuous Auditing

To solve the problems faced by organizations using traditional audits, the concept of continuous auditing was introduced which is defined by Canadian Institute of Chartered Accounts (CICA) and American Institute of Certified Public Accountants (AICPA) as follows.

‘A methodology that enables independent auditors to provide written assurance on a subject matter using a series of auditor reports issued simultaneously with, or a short time after, the occurrence of events underlying the subject matter.’

It is evident from the above definition that traditional auditing methods (paper based) would not live up with the pace of the electronic systems of today and would take longer to provide answers for the anomalies. Evolving need for assurance over business integrity has led to enormous changes and need for robust control systems. Continuous auditing provides this assurance by frequent testing of internal controls and conducting risk assessments in real time. Continuous Auditing leads to two main potential benefits categorized as continuous risk assessment and continuous controls assessment. Risk assessments are areas where need for audit is recognized by the monitoring system. These areas also require frequent changes to internal operations and set of procedures to qualify

for audit adherence. Control assessments on the other hand, focus on control effectiveness and identify threshold for a tolerance level and help design control tests.

1.3 Problem Statement

Continuous auditing is an emerging concept which is being adopted quickly by organizations to solve the problems faced by using traditional auditing methods. A lot of emphasis has been given to increase efficiency of internal controls and reduce risks associated with information systems and aid them in making timely decisions. However, most of the research efforts in continuous audit world have been focused around transactions in the financial information systems of organizations and very little research has been done to create solutions for non-transactional data and in particular the processes involved in the organization.

Thus, our proposed methodology is designed to: (a) create a continuous auditing framework for non-transactional data, and (b) audit the individual processes and process chain to increase the overall efficiency and effectiveness of the audited system.

1.4 Contribution

We propose a continuous process auditing framework which performs the traditional transactional auditing and also audits the individual processes in the system to minimize risks and increase the efficiency and effectiveness of the system. Continuous auditing methods previously worked only within the context of financial transactional systems and are limited to providing assurance about financial aspects of the system. Our proposed method audits the individual processes in the system and the inherent process chain to increase the overall productivity of the system. A communication mechanism is introduced to aid in sending the relevant information between different processes and

performs the auditing tasks. As an important part of the model, the notion of an information repository is proposed to work in the context of a domain expert to validate the resultant data produced by the system. A weighting schema mechanism is introduced to facilitate the process of completing the auditing tasks and to provide alternative strategies to increase system efficiency and minimize risks and errors. An analysis and reporting mechanism is also introduced to interpret the results of the auditing procedure.

1.5 Organization of the thesis

The rest of the thesis is organized as follows. Chapter II provides the background literature review and shows the different models proposed for continuous auditing and components of Continuous auditing. It provides information about some milestone papers and also provides the analysis of the different techniques currently used to achieve continuous auditing. After the literature review, our proposed methodology for achieving continuous process auditing is shown in chapter III. A case study in applying our proposed model in auditing an experimental survey application along with the discussion of findings and results is provided in Chapter IV. Finally, Chapter V concludes our contributions summarizing the advantages of our methodology and provides recommendations for future work.

CHAPTER II

LITERATURE REVIEW

Different approaches have been taken by researchers to perform continuous auditing in literature. These approaches are reviewed in detailed in this chapter. The first section of this chapter describes the different models that have been proposed in literature to perform continuous auditing. The second section discusses the different techniques used in the literature to achieve continuous auditing. The third section details the different components of continuous auditing as described in literature. The last section discusses the difference between process and procedure from auditing point of view.

2.1 Continuous Auditing Models

The first continuous auditing application was built in 1991 by Vasarhelyi et al. [4] at AT&T Bell Laboratories. The prototype application was named continuous process auditing system (CPAS) and is the first application designed to deal with auditing problems faced by real-time systems. The approach relies on placing software probes into the operational systems for the purpose of monitoring and consists of a data provisioning system and an advanced decision support system. The application provides measurement capability by transporting the copies of key management reports to an audit workstation

where the necessary data is extracted and analysis is performed. The application also provides monitoring capability by placing an auditing module in the client system which is controlled by the auditor to monitor the desired transactions. The prototype system also provides the analysis capability in which the auditor used the operational data to generate the results. This was the first attempt towards making a continuous auditing system and although it only worked for the billing department of the company, it paved way for other companies to follow to change their legacy systems according to this evolving technology.

Woodroof and Searcy [5] proposed a model for continuous auditing for implementation within a debt covenant domain. Their model focused on 'on-demand' reporting in which a request for an 'ever green' report is initiated by the auditor. The ever green report is a report which is initiated by request and is displayed on the web. The model makes use of agent technology to carry out the activities. Agents and sensors are placed inside the client system to monitor the transactions. These agents monitor the transactions based on pre-specified rules and look for transactions in which an exception to the specified rules occurs. The model is initiated by a request from the auditor to generate a report. Agents in the client system start monitoring the transactions for exceptions. A digital agent on the auditor system sends a request to the digital agent on the client system to retrieve the real time balances from account tables. Upon receiving the result from the client agent, the digital agent in the auditor system extracts the required information from the results to make sure only necessary requested information is presented to the auditor. Based on the results, an 'ever green report' is generated and displayed to the auditor. Although this model extracted the real time information and

presented the results, its scope was limited it only applied to a debt covenant system. Also it focused on on-demand-reporting which is requested by the auditor rather than continuous reporting mechanism.

Rezaee et al. [6] proposed a continuous auditing model based on specialized data marts and standardized testing. In this approach the information is downloaded from the transactional systems and transformed into an appropriate form to be stored in an audit warehouse from where it sent to audit workstations for testing or data marts for storage. This model has the capacity to run on a distributed client/server network and provides the auditing functionality by passing audit data to specialized audit workstations. In this model, data which is to be audited is gathered from the transactional systems. Once the data is collected via the web, its passes through an ETL process (extract, transform and load). In the first step, this data collected from a variety of platforms and systems is extracted. In the next step, the extracted data is transformed into a suitable form to be loaded into the data marts. In the least step of the process, the data is loaded into the data marts. Once the data is loaded into the data marts, standardized tests are created and performed on the data. These tests are either performed periodically or continuously depending upon the requirement. Although this model is not domain specific, it consumes a lot of time and incurs sufficient cost as the data needs to be extracted from the system and then transformed into a form suitable for running the tests.

Onions [7] proposed a keystroke level monitoring model to analyze the integrity of data. This model monitors the database utilities and applications for commands that can cause fraud or error thus providing a detailed protection. It provides individual protection against each transaction and combined protection against a certain pattern of

transactions. This functionality is achieved by monitoring each transaction on an individual basis thus providing transaction level data examination. These transactions are tested at the time of entry and checked against pre-specified rules for the individual transaction. After performing the real time individual testing, these transactions may be added to a data mine for further inspection. The transactions are also audited over a period of time to provide transaction pattern level data examination which looks for patterns of transactions that together lead to fraud. This model uses specialized software tools and expert systems to perform the individual and pattern level monitoring. This model consists of four steps. In the first step transactions and data is collected from various sources and then entered for processing. In the second step, the transactions and keystrokes are mapped onto a matching schema based on the transaction format. XML based schema known as XCAL (eXtensible Continuous Auditing Language) is used to define the schema in order to convert data originating from different formats into a uniform schema for later use. Once the transactions are stored in the desired schema, real time CAATT (computer assisted audit tools and techniques) processing is utilized in third step to check these transactions and keystrokes. In the last step, expert systems are used to search for patterns in data for frauds.

Hasan and Stiller [8] proposed a generic continuous auditing architecture which mainly consisted of three entities. First entity is an auditee who carries out the activities based on specifications to achieve a certain goal. Second entity in the model is an accountant whose duty is to observe these activities and records them as facts. Once the facts are recorded, the auditor; who constitutes the third entity; conducts the audit by

looking at the specification and facts thereby detecting violations. Different units which are combined together to form this generic model are shown in Figure 1 below.

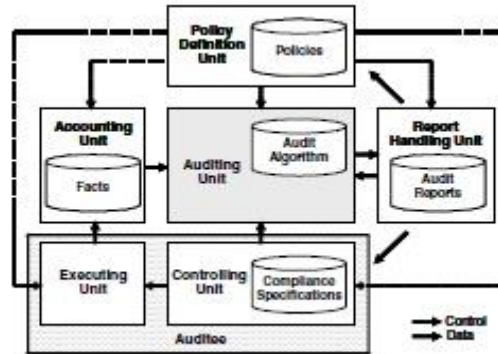


Figure 2.1: Generic auditing architecture

Different units are combined together in this generic architecture to perform continuous auditing. The Controlling unit defines the activities that are to be carried out by the user. Users, or system components, use the Executing unit to carry out the actual activities. Once the activities are carried out, an Accounting unit records the facts about the activities. These facts are then fed into the Controlling unit which contains the auditing algorithm of the application. In order to carry out the analysis, the Controlling unit requires the information from policy definition unit which dictates how the activities are to be performed. The result of the analysis is then sent to the Report Handling unit which prepares the report about the activities performed. This generic model uses policy based approach to configure the above-mentioned units and control their behaviour. This gives the benefit of making a modular structure where decision making is separated from execution. This generic architecture works well when policies and procedures are well defined and provides a step towards making a generic model which can be applicable to all parts of an auditing system.

Murthy and Groomer [9] proposed a continuous auditing web services (CAWS) model based on service oriented architecture [30] which consists of three entities: a service requestor (an auditee), a service provider (an auditor) and a service registry. They proposed a 'pull model' in which the registry mechanism resides inside the auditor system that pulls the required information from the client. The auditor uses xml-based web services framework to extract the information from the client and performs the analysis to generate the results which are passed on to the client.

Huanzhuo and Yuning [10] criticized the work by Murthy and Groomer [9] and claimed that it exposes the underlying data and business practices of the client as the service registry module resides outside the client. They proposed a 'push model' in which the service registry module resides in the client system. The client pushes the required data to the auditor that performs the analysis and sends the results back to the client via the web services framework. Since the registry module resides inside the client's system, security of data is not compromised and data ownership issue is resolved. These above mentioned continuous auditing models based on services work with applications which have same data format.

To resolve this data interoperability issue, Ruey-Shun and Chia-Ming [11] presented a model, based on services, that works with heterogeneous data formats. They proposed a Collaborative Continuous Auditing Model (CCAM) that uses agents and consists of a schema matching repository for different data formats. The model also contains a unified auditing database that gathers the information from the client and resides audit services provided by the auditor. When a request for validation of internal control arrives from the client for some specified transactions, the auditing system

invokes the data validation agent from the unified auditing database to verify the transaction data. If no data is present in the unified database, the data transformation agent is invoked to retrieve data from the client's ERP system which is then validated by the data validation agent using the schema matching repository thus reducing the complexity of data formats. It also helps in securing sensitive information as unified database is under the client's custody.

Zouming et al. [12] also proposed a similar service oriented based continuous auditing model which uses agents encapsulated by web services to communicate between them via open internet. In their model, each agent performs a specialized task and uses inherent intelligence to autonomously cooperate with other agents if required to perform a task. The use of open internet as the communication mechanism allows increasing the system performance by making it easy to add new agents as required.

Huanzahuo et al. [13] proposed a continuous auditing model based on the concept of services that employs enterprise service bus (ESB). The enterprise service bus (ESB) consists of a HUB which is used to magnify the signal and a Namespace which is used to map the corresponding services. In order to integrate a variety of enterprise applications and host business processes that run for a long time, the ESB utilizes an Adapter and Service Orchestration Engine. The service oriented based continuous auditing model used consists of a client system, an auditing system and the third parties. The client and the third parties register in the auditing system and submit an application to monitor before a transaction takes place. During the transaction, the auditing module extracts the information from the enterprise service bus (ESB) and checks if it is according to the contract agreed between the clients and third parties. This is achieved by using intelligent

agents which continuously monitor the client databases and the internet site of the third parties. At the end of the cycle, a report is generated and displayed to the client.

Chou et al. [14] an agent based continuous auditing model (ABCAM) that uses multiple software agent that are mobile to perform the auditing tasks. These software agents are able to move from one platform to another and use intelligence as required to achieve the desired goal. The aim of using intelligent mobile agents is aid in the gathering of information for auditing purposes and replace human auditor to carry out these activities. In this model, each software agent represents a specific audit procedure and assesses the audit information that are present distributed information sources. These agents come together as a group to represent the overall audit functionality. The model works for system with high degree of automation in business operations where information provided by the agents is enough for auditors to complete final analysis. The model is divided into an interface module, a procedures module and an agent invocation and execution module. The interface module captures specific information from the audit requestor to identify the specific audit procedures for the request and presents the results of the audit procedures. The procedures module maps the requests captured by the interface module to a set of activities required to gather the audit related information. The agent invocation and execution module invokes the agents to perform the audit related activities, monitors their performance during the activity and sends the results to the interface module to be displayed.

Wu et al. [15] proposed an agent-based architecture for collaborative continuous auditing. This model consists of two main entities: auditor site and auditee site. Auditor site is the master site whose duties include planning an audit service. It also has the task

of dispatching agents to auditee site to perform audits. It consists of an audit organizer which provides an interface for different functions to be called (such as planning, reporting, analyzing etc). Audit planner is the module that generates audit plans/rules (depending upon various metrics and objectives) and according to that plan agent dispatchers deploys various agents (with specific functionalities) to the auditee site. The auditee site consists of various agents. One of these agents is collaborative audit agent which provides an interface to query the audit service. It also communicates with audit organizer (in auditor site) to control the execution of tasks if deemed necessary. Analytic agent (they apply audit rules and check for exceptions and errors) and data capture agent are the agents deployed by audit dispatcher on the auditee site. These agents then match their results to make sure no errors or exceptions are performed at the auditee site and take corrective measures if required.

2.2 Continuous Auditing Techniques

Different techniques have been used to achieve the goals of continuous auditing. The techniques that have been utilized in literature [11] [16] [17] [18] [21] are as follows.

- Embedded Audit Modules (EAM)
- General Audit Software (GAS)
- EAM Ghosting
- Monitoring Control Layer

Embedded audit modules (EAM) are specialized programming modules that are inserted in the client system to achieve the purpose of continuous auditing. Murthy and Groomer [16] introduced the notion of EAM as an alternative approach to audit in order to help the companies with computer based accounting systems. This approach was

developed to solve the issues regarding the control and security in database environments. EAM modules were built into the systems in order to capture the audit related information on a continuous basis. In EAM technique, the code for performing the audit related activities is developed in the application's programming language and implemented inside the target application. This code is considered 'non-native code' since it is added to the application for performing the audit. The code allows the EAM module to evaluate the transactions against the specifications in real time and sends reports to the specified individuals by select operations already built into the module code.

Another variation of EAM technique is known as integrated test facility (ITF) [17]. This technique involves the creation of a fictitious entry in the database to process the test transaction in the live system. This fictitious entry is the test transaction which is run with the normal transactions in the system and can be implemented periodically without requiring a separate process for testing the system. EAMs and ITFs typically involve some kind of modification to the client system. Hence these are not only expensive to achieve but also highly likely to get resisted by the client as it involves modification to their system. Another problem with EAM is that they slow down the system as the audit module is checking and validating the live system.

Generic audit software (GAS) [11] is another technique used for continuous auditing. GAS involves using specialized software to assist in the auditing of the system. GAS is typically preferred by audit firms as it allows them to achieve the goal without interfering with the client system. However GAS is based on periodical auditing process model (PAPM) which means that it is not good for real time auditing and reporting. Most

widely used continuous auditing software used in the industry (according to GAIN [19]) is ACL [20]. ACL is data extraction and analysis software that provides custom solution to different organizations to provide the continuous auditing functionality.

Kuhn and Sutton [18] proposed EAM ghosting as a technique which is a variation of EAM technique and provides the benefits of EAM with the advantage that audit functionality is implemented, operated and maintained outside the production system of the client. Thus it separates the audit functionality from the production system. This segregation can be achieved by two methods. In first method, the production system (PRD) is separated from the quality assurance system (QAS) by creating partition on the server. QAS is a mirror copy of the production system but is used to house the EAM module to complete the continuous auditing process. In the second method, virtualization techniques are used again to mirror the PRD system and EAM module is embedded in the QAS server which is hosted on the virtual server. Virtualization is beneficial amongst these two methods as it requires less physical hardware space and less memory to operate. Hence, EAM ghosting retains the integrity of the system and reduces the cost of the system as well. One concern, however, is the existence of non-native code (EAM in the ghost system) affecting the transactions in the ghost system to an extent where the system trudges along and possibly fails.

Vasarhelyi et al. [21] introduced monitoring control layer (MCL) as an alternative technique to perform continuous auditing. MCL creates a bridge between the auditing system and the client system. It consists of a middleware layer which binds the auditing system with the client system. MCL not only captures and filters data, it also stores it and performs analytical actions to send alarms and create reports accordingly. Advantages of

using MCL include the separation of auditing functionality from the client system. Also it is easy to implement even if the client system is distributed and comprises of different platforms. Hence, it provides client independence as well as system design and maintenance freedom.

2.3 Continuous Auditing Components

Continuous auditing means performing assessments about the system on a continuous basis. This process consists of performing the assessments about the controls in the system to make sure it functions as desired. The system is also assessed against the risk factors to improve its efficiency. On the basis of these criteria, Alles et al. [22] divided continuous auditing into the following two main components.

- Continuous data assurance
- Continuous control monitoring

As is evident from the term itself, continuous data assurance deals with the auditing of the data itself. This means that the data itself is under investigation for auditing. For example, if it a financial company, then continuous data assurance would mean making sure that the financial information is correct. This category of continuous auditing deals with the data part of the system and makes sure that it is correct and without any errors/fraud. This implies looking at a transaction to make sure it complies with all of the controls that are in place. And this is not done for only one transaction or selective transactions or ever at all transactions at selective times; this process is done all the time. Every transaction that the system performs is checked to make sure that it is working as specified and all the data matches the expected results.

Continuous control monitoring forms the other integral part of continuous auditing. As evident from the term, it simply means to make sure that the control within the system is working correctly. For example, in a financial system such as a bank, this means to make sure that controls in place; such as maximum number of transactions allowed as per account type, is working correctly. Thus continuous control monitoring checks the settings in the system. It compares them with a given model and makes sure that the system settings are working as they are supposed to perform. For example, measuring specific attributes that if certain parameters are not met, they will trigger auditor-initiated actions. The nature of these actions may vary according to the risk or anomaly identified. Hence, the main objective of continuous control monitoring is to focus on the effectiveness of the control itself.

2.4 Procedure Audit vs. Process Audit

ISO 9000:2000 [23] defines process as a set of inter-related and interesting activities that transform the input into output. The results of these activities are examined to verify if the activities and the resources and behaviors that caused them are managed effectively and efficiently [24]. A process audit establishes that the results generated by the process are being generated by an effectively managed process whereas a transaction audit simply follows the trail from the input to the output. The effectiveness of a process is measured in terms of its objectives. In an effectively managed process the activities, resources and behaviors are organized and controlled in a way to achieve the desired objective. A process audit emphasizes on the results whereas a transaction/procedure audit focuses on the tasks. Table 2.1 summarizes the differences between a procedure audit and process audit [24].

Procedure Audit	Process Audit
Identifies what tasks are being performed	Identifies what objectives are required to be achieved
Identifies who performs the tasks	Identifies the factors affecting success
Identifies the procedures governing the tasks	Establishes what the process is for achieving the objectives
Establishes whether the procedures are being followed	Verifies that the controls in place are consistent with the success factors
Establishes whether the person is trained to perform the task	Establish the competences and capabilities required to deliver the process outputs
Verifies that the documentation is current and the equipment is calibrated	Establishes that the competency and capability is being assessed effectively
Verifies the working conditions are suitable	Establishes what results are being achieved
Establishes where the inputs come from and where the outputs go to	Establishes how outputs are being measured
Verifies that the personnel making acceptance decisions are authorized	Verifies the integrity of the results
Verifies maintenance of record	Establishes that performance, efficiency and effectiveness is reviewed and pursued

Table 2.1 Procedure Audit vs. Process Audit

Table 2.1 shows that process auditing focuses on the results generated by the auditing process and makes sure that the behaviors and resources that perform those activities are properly managed. Previous work by researchers in continuous auditing focused on continuous audit of transactions/procedures only and it is required that work be done on the continuous audit of processes to increase the compliance and efficiency of the system under discussion.

CHAPTER III

PROPOSED METHOD

In this chapter, we highlight the limitations of the methods used in previous work and present the proposed framework that is the basis of this research work. We define and present a new form of continuous auditing named Continuous Process Auditing (CPA), based in part on the pioneering work of Vasarhelyi et al. [4]. We focus on creating a framework for continuous auditing of processes in order to achieve better overall efficiency of system operation. Details of our proposed framework are introduced in this chapter.

3.1 Continuous Process Auditing

In most information systems, transactions form an integral part of the system on the micro computing level and transaction management presents a mechanism to manage critical information resources [25]. This form of computing level management has drawn much attention from researchers. In order to improve the efficiency of information systems, much work, as reviewed and presented in Chapter II, has been done by

researchers in the continuous auditing domain, in particular, by targeting the transaction level computing and auditing the systems based on transactions. However, the transaction or procedural level auditing, can provide little benefit if the overall process that utilizes that procedure is inherently inefficient or inapplicable. Therefore, to address this problem, overlooked in previous continuous auditing models, we introduce the term Continuous Process Auditing and define it as follows.

Continuous Process Auditing refers to an examination of results on a continuous basis to determine whether the activities, resources and behaviours that cause them are being managed efficiently and effectively.

We note at the outset, however, that the context of this thesis research inherently refers to discrete monitoring of events in digital domains. Hence, the use of the word “continuous” will be used interchangeably whether referring to mathematical objects with well-defined behaviour within a continuous domain, or to cases where system behaviour is defined in terms of well-defined models with defined, measurable, and discrete behaviours.

From the definition, it is evident that CPA is centered on the activities being performed in the system. These activities can be in the form of transactions, procedures or any other form depending upon the atomicity and granularity of the system. By effectively managing the resources that perform these activities, a better overall efficiency can be achieved. This can result from releasing the resource as soon as an activity is done or even sub-releasing the resource to perform a second activity while the current activity waits for an input from a third activity. Just as the resources are needed to

perform a given activity, behaviours are also an important component of the system as they define the circumstances under which a given resource performs a specified activity. Thus, behaviours are manipulated to achieve a desired result. This, in turn, makes behaviours an important aspect to consider in pursuing better efficiencies and managing behaviours that affect how systems of procedures attain definite goals. Together, an activity, the resource that carries out that activity and the behaviour that causes that resource to carry out the activity, form an important trio of process characteristics that are of immediate relevance to the efficiency and effectiveness of the process and dictate its productivity.

Thus, Continuous Process Auditing (CPA) helps in establishing whether the results, which are generated as an output by executing the process, are being generated by an effectively managed process by performing a continuous audit of processes involved in the system. Another goal of CPA is to suggest alternative strategies to achieve process efficiency. These strategies can range from changing the sequence of steps performed in the process to overlooking a few process steps during certain cases to changing the steps altogether depending upon the specific needs.

Moving from continuous auditing of transactions/procedures to continuous auditing of processes in the digital domain comes with its challenges. One such problem is bridging the gap between business processes and IT technology. The challenge is to translate the information about processes from business language into technology specifications that can be used inside an automated program. Another problem is mimicking the complex actions that human auditors take while performing an audit. The specific challenge here involves mapping the actions performed by human auditors

during an audit cycle onto digital actions that can be performed by programmable modules. Another major problem that continuous process auditing poses is minimization of associated overhead as increased testing is beneficial, but it may increase audit cost. Preserving an audit trail especially in digital world after an event has transpired poses another major problem as a revisit to audit evidence may be critical in future. Translation of analysis information into meaningful data for human understanding is another challenge posed by moving the continuous audit of process into digital domain. We researched these problems in detail and present the solutions in our proposed framework.

3.2 Proposed CPA Framework

This model described below is proposed to solve the problem of auditing processes in a system in a continuous manner. We evolve the continuous audit of transactions/procedures in financial domains and move it towards continuous audit of processes devoid of domain dependence. We propose that by performing a continuous audit of processes aided by a weighting schema and a communication mechanism, efficiency can be achieved and risks, errors and frauds can be mitigated in complex IT systems. In our model, a communication mechanism and a weighting schema is added to perform the continuous audit. By using the weighting schema and information about the processes obtained via the communication mechanism, analysis of the processes is done and results are generated. The overall architecture of our proposed model is shown in Figure 3.1.

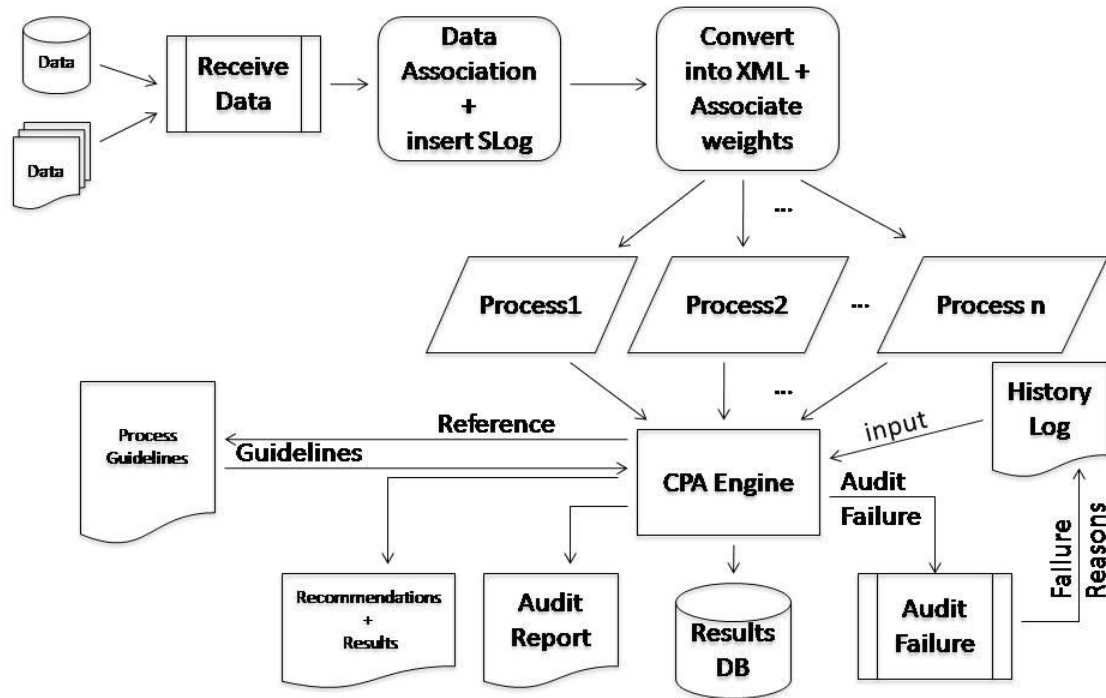


Figure 3.1: Overall architecture of continuous process auditing model

As shown in Figure 3.1, data is received from the system in the first stage. Once the data is received, it is associated with the appropriate process it belongs to. This is done by extracting the Process ID field from the data block. Once data is associated with the correct process, sticky logs (Slog) [26], which define the communication mechanism in our model, are created, populated and associated with each process data block. There is one sticky log per process so the total number of sticky logs depends upon the number of different individual processes. In the next stage, data is converted into an XML format. Using XML for formatting is widely accepted and allows for data to be transported easily if and when required. Weights, used from the weighting schema proposed in this model, are associated to different individual steps of processes.

Then the data is transmitted to the CPA engine via individual dedicated busses (bus-based approach). The use of dedicated bus for each process means that CPA only needs to interpret Process ID from the data block to know which process that data block belongs to and sends the data to appropriate handlers for performing the analysis. Another benefit of bus-based approach means data would not mix and data cleansing (an important step that takes considerable time and space) can be avoided thus increasing the efficiency of the system. It also augers well for using one sticky log per process (this saves time, complexity and overhead of I/O operation for each block of data) as all relevant data for each process is at one place and can quickly be assessed if required and can easily be moved to persistent storage as deemed necessary (periodic). In the last stage the data is fed to CPA engine and the engine performs its analysis on the data according to the steps involved [24] to generate the results. The flowchart of activities performed in the engine is shown in Figure 3.2.

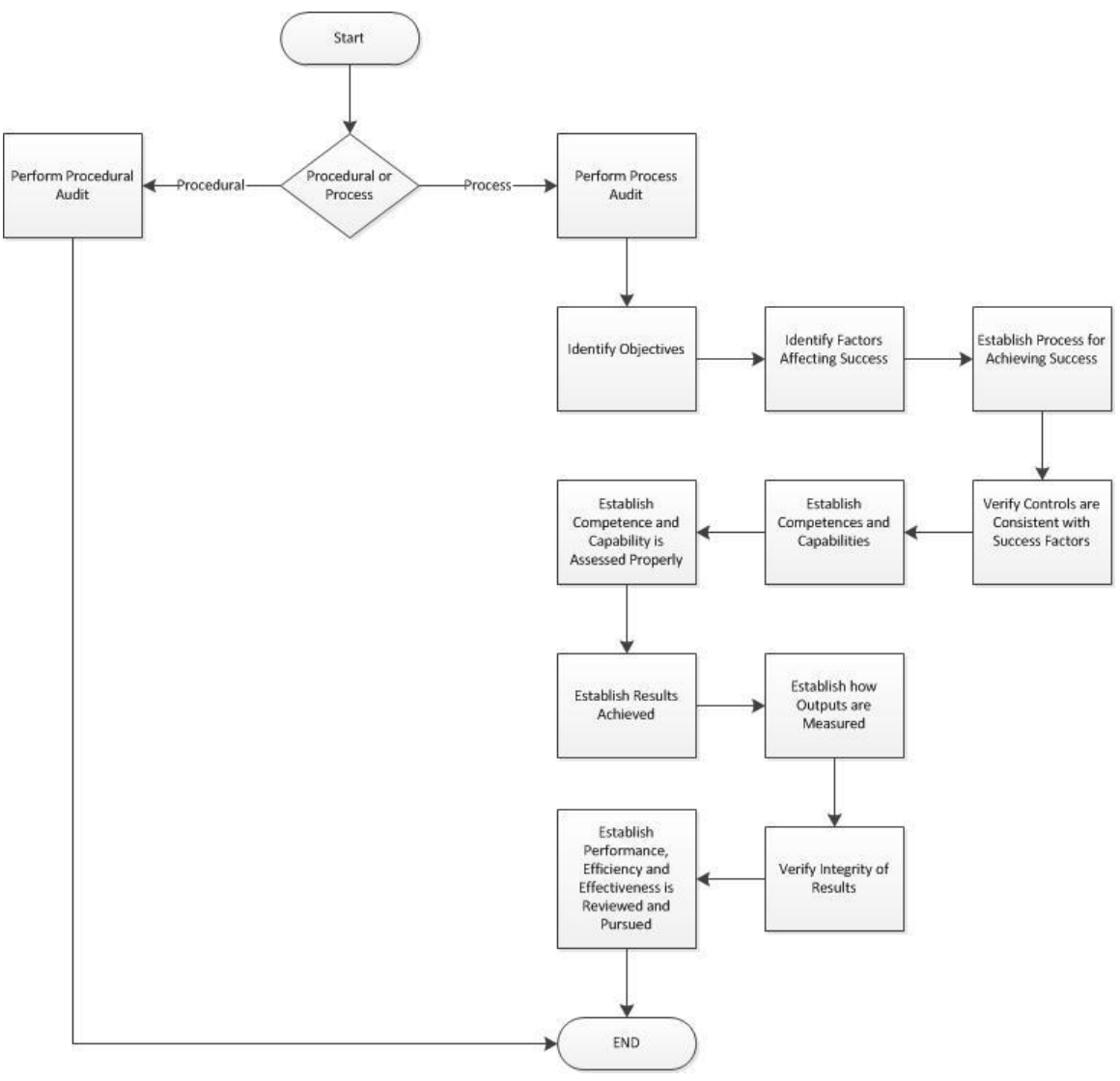


Figure 3.2: Workflow of steps performed in CPA engine

As explained in previous sections and chapters, auditing of processes means accounting for the activities, resources and behaviours that combine together to generate the results of the process. Hence, the CPA engine has two main aspects. The first aspect consists of procedural audit (also known as transactional audit) and the second aspect includes auditing the process. Procedural/transactional audit is concerned with auditing the actual steps of the process (step1, step2 etc). It deals with issues such as whether all the steps in the process were duly followed. Another important feature involves making

sure only authorized persons are allowed to manipulate the data and an audit trail is recorded for future references. In short, the procedural/transactional auditing checks every transaction to make sure it is error and fraud free and records necessary information so that every transaction can be trailed to find the initiator. The pseudo code for procedural auditing is shown in Figure 3.3.

```

Check Process ID
Consult Master Document
Retrieve # if steps for the process
Verify Steps 1/2/3 are done
Log necessary information in slog

```

Figure 3.3: Pseudo code for procedural auditing

The second aspect of CPA engine consists of auditing the process itself. Process auditing differs from procedural/transactional auditing as the aim here is the improvement in the process and hence the overall system. The objectives of Process Audit are (but not limited to) as follows:

- Identify the objectives required to be achieved
- Identify the factors affecting success
- Establish what the process is for achieving the objectives
- Verify that the controls in place are consistent with the success factors
- Establish the competences and capabilities required to deliver the process output
- Establish that competence and capability is assessed effectively
- Establish what results are being achieved
- Establish how outputs are being measured
- Verifies the integrity of the results
- Establish that performance, efficiency and effectiveness is reviewed and pursued

One of the objectives above is identifying the factors affecting success. This corresponds to finding out which factors (procedural steps) are more critical for this process and should carry more weight. This information can be gathered, for example, by checking the steps that fail mostly for a given process and giving more weightage to achieve the results. The factors (procedural steps) carrying most weight (or the steps failing mostly) are the factors affecting success. The time complexity of this algorithm is $O(n)$, where n is the number of processes found in the Process Sticky Log. The pseudo-code for this objective is shown in Figure 3.4.

```
Check Process ID  
Consult Master Document  
Factors affecting success == factors with max weight, steps that fail mostly
```

Figure 3.4: Sample objective pseudo code

A flowchart of these objectives grouped as sub-processes is shown in Figure 3.5. Based on the different requirements, our proposed architecture is divided into four main components including: the Sticky Log, the Weight Schema, the Master Document and the Analysis/Reporting. The details of these components will be discussed in following sub-sections.

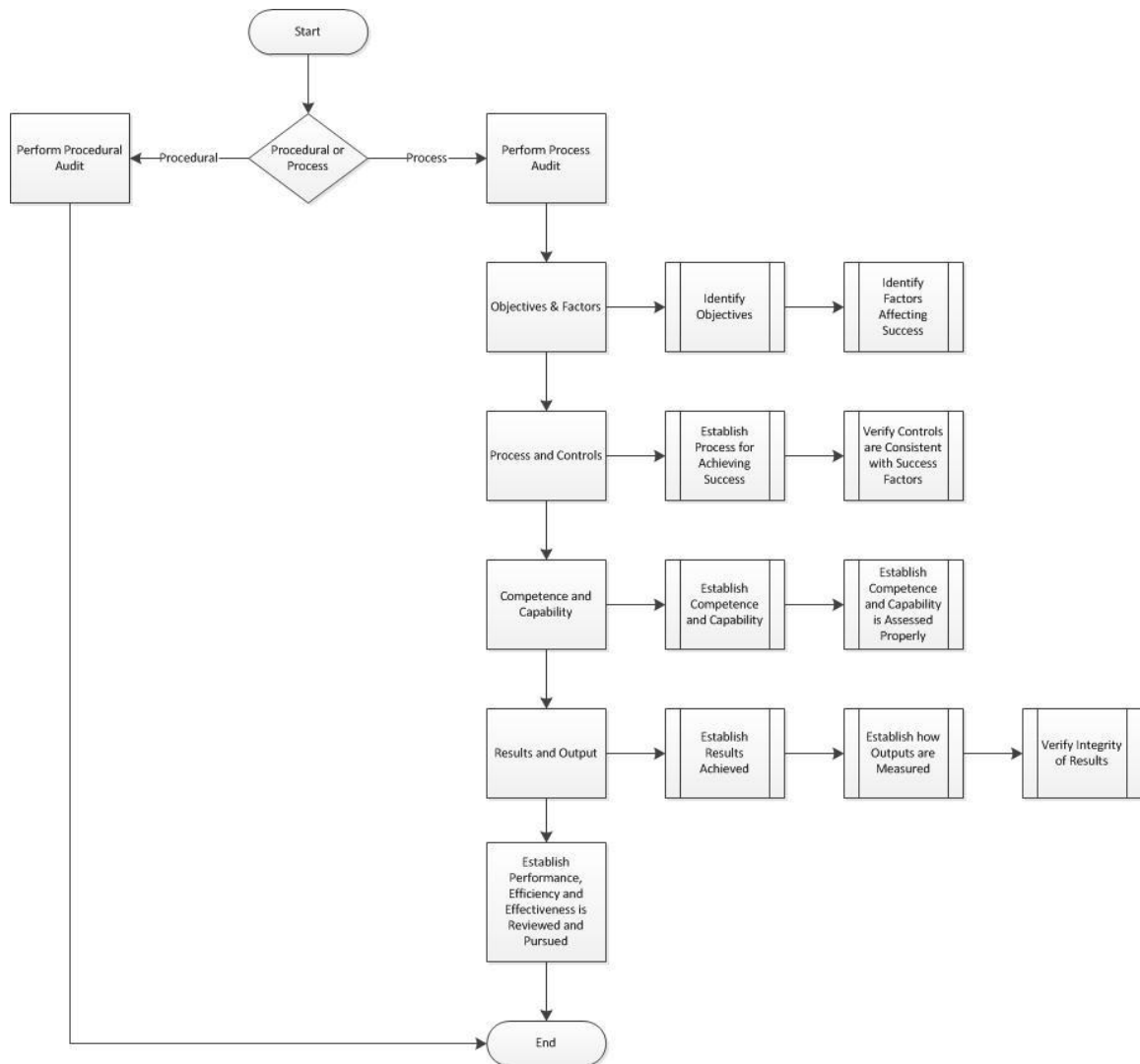


Figure 3.5: Flowchart of steps performed in CPA engine grouped as sub-processes

3.2.1 Sticky Log

Sticky log (Slog) is a document that is used for communication of important data in our model. It is an ontological framework for organizing data and is used to provide quick access to relevant data during the lifecycle of the audit process. It is inserted in the metadata of each block of process data and is written to external permanent storage at the end of the cycle. It contains the critical audit information about the process data block generated as a result of executing the process.

The Sticky Log contains identification data such as the ID of the process, ID of the data block or ID of the person who executed the process. This helps in the indentifying who performed what operation in the process and also helps in maintaining the controls in the system intact to improve the efficiency. The Slog also contains important data about the process such as the number of steps undertaken for the process and the individual value of each step. This helps in making sure whether the steps undertaken during the process work towards fulfilling the objective of the process and whether they are consistent with the success factors or not. In short, the Slog contains basic audit parameters and may contain rich metadata that provide interpretive information. The benefit of these logs is that each Slog contains information about a specific process and it can be systematically appended to form a growing document that represents the complete audit trail at any moment during the execution of the process chain. Thus, it provides a solution for the audit trail challenge as identified in the previous section. Owing to its small size, by virtue of collecting only critical data, it also helps solving the problem of minimizing the audit cost without compromising the testing coverage and solves another challenge posed in previous section. Audit cost is reduced and associated overhead is minimised as a result of lower transmission costs of these Slogs across the network.

One issue in using Slogs is designing a structure and format which is best suited and easily adoptable across application domains. It can be defined using Resource Description Framework Schema (RDFS) and Ontology Language for the Web (OWL) [27] (which is expanding quickly), Extensible Markup Language (XML) [28] (which is

preferred due to ease of use and portability) or a simple delimiter based criteria such as Comma Separated Values (CSV) [29] (which is the widely adopted approach).

We propose that, due to the imperfection of RDFS/Owl at the current time and the added inner complexity of XML manipulation, a delimiter based approach is used for defining the structure of Slog. Another aspect to consider for defining the Slog is the actual structure of the document. Should it have a predetermined size or should it be flexible? Careful consideration is required to define the structure of the Slog as it would be transmitted across the network in a message that could impose overheads and performance bottlenecks. Predetermined size with a limited space for recording extra information (e.g. error code or exception message thrown) is the approach we consider to be most suitable to minimize the overhead and increase the overall efficiency.

Another challenge in the creation of Slog is the number of Slogs that should be created. This question offers choices, including whether one Slog should be created for each cycle/component of the system, for each process in the system, or for each block of incoming data. For example, for an application about patients coming into a hospital environment for various tests/checkups (each of which is a process), should a Slog be created for each patient, one for each test that every patient goes through or one for each test that every patient undertakes? One Slog per test (process) and one Slog per process for each patient puts too much burden on the system as too many Slogs would have to be created adding a complexity layer to keep track of all the Slogs producing an undesired overhead.

We propose, therefore, that for the patient scenario, one Slog per patient is created which contains information about all the tests (processes) that the patient goes through.

For performing the analysis, the information from this Slog can be divided according to the tests (by using the process ID) and delivered to appropriate placeholders for further processing.

Transmission of Slogs is also an important issue to consider since the Slog contains important data. This transmission can be directly from the live host system to the audit engine or from the host system to a staging area where it is kept and a group of Slogs are sent to the audit engine periodically. We propose that secure transmission mechanisms, such as UNIX pipes, be used to transmit the Slog from the system to the audit engine. Either a named pipe can be used to transmit process specific data, or one pipe can be used to transmit the whole Slog which is separated into relevant data for each process by matching the Process ID during analysis. We do note the need for further work on this security issue, but we do not consider it in depth within this thesis.

In general, then, for any given domain, the sticky log structure can be divided into two main parts. The first part is fixed and contains the identification data for a given process including properties such as ID of the process, ID of the person who executed the process, the number of steps for the process etc. The properties are extracted from the Master Document and stored in the Sticky log to lower data transmission costs associated with performing the audit analysis. The second part of the sticky log structure is variable and contains data about the steps of the process that user has performed. This basic sticky log structure is same for all the processes in the given domain. However, this basic structure can be extended to include any other property of a given process which is not present in another process of the same domain. Hence, there can be different sticky logs for different process which are combined in the end to generate the overall audit result of

the application in a particular domain. This flexibility and abstraction in the structure of sticky log allows each process to create a sticky log that is tailored to accommodate its specific needs.

3.2.2 Weight Schema

A desired goal of continuous process auditing is moving the auditing realm into the digital world. A challenge that precedes this realization as described in previous section is mimicking the complex action of human auditors. During traditional auditing, human auditors perform complex tasks during the audit cycle and the mapping those actions onto digital actions performed by programmable modules is a problem. To solve this problem, we introduce the notion of weight schema in our model.

Just as the human auditors take different actions under different circumstances, we suggest that programmable modules, working under the guidance of a weight schema, can achieve the desired effect of mimicking a human action under the same scenario. In the weight schema, a specific weightage is assigned to each step in the process. The sum of all weights (adding the individual weights of each step) for a given process is equal to 1. Thus each step is given the weight according to the importance of the step in the overall process.

For example, in a hospital scenario where a patient comes and fills in a personal information form, the field (step of the ‘gather personal information’ process) of personal identification (Social Insurance Number) is given more importance than the field of personal address (such as street name) as first field is required to identify the patient uniquely. Hence the weight for a given step in the process is dictated by the importance of the step depending upon the criteria. For each process, there is a minimum threshold

score that defines the minimum score that must be achieved by adding the individual steps of the process in order to pass the audit. Again, the value of this threshold score is dictated by the definition criteria and may vary from process to process. If the total score of a process is greater than the minimum threshold score, the audit is considered to be a success (audit is passed) and if the total score of a process is less than the minimum threshold score, the audit is considered to be a failure. Depending upon the difference between the audit score achieved and the minimum threshold score, various categories of audit result can be assigned to the process. For example, for a minimum threshold score of 70, an audit score of 73 falls in the 'just passed' category, an audit score of 82 falls in the 'passed' category and an audit score of 95 falls under 'passed with flying colors' category. These categories are defined externally and can be modified as required.

Recommendations are provided by changing the weights of individual steps with the goal to maximize the difference between the audit score and minimum threshold score. The aim of providing the recommendation can vary as per the situation demand. For example, consider a hospital scenario where a patient has to undergo three separate tests to complete the process. First test has been done and he is waiting to undergo the second test and is waiting for the nurse who is busy with some other patient. Meanwhile, the third test can be performed although the patient is waiting to undergo the second test; however, the process requires finishing each test before moving on to the next one. Under this scenario, the recommendation to get the third test done before the second one results in increasing the efficiency of the process by saving valuable time. Similarly, for a survey application, recommendation by changing the weight could simply imply changing the order of the question on the survey in order to get the response for some question that are

missed due to their lower position on the questionnaire. Hence, the recommendation provided can vary according to the desired goal.

One issue in implementing weight schema is the initialization of the weight schema. How should the weights be configured at the start of the cycle? One option is to start with an equal weight schema for every process step and adapt as the learning continues. This approach, although logical and un-biased, depends on heuristics and artificial intelligence where the system can learn during its life span. Since these techniques have not matured and require much more perfection before being adopted, we hope these can be relied on in future. Another possible approach is to depend on the knowledge of the domain expert in initializing the system. This requires hiring the experts of the field and combining their background knowledge of the subject with current expertise to come up with a solution. This option, though viable in theory, can prove to be expensive and increase the overall cost of the system. Another option is to get the initial weight schema from the future system users as a part of requirement gathering process. This relies on benefiting from the knowledge of the future users and is the approach that we suggest owing to the increase in cost and lack of maturity of the other mentioned approaches.

One main challenge for implementing the weight schema is finding the correct time to implement the change in the schema. If the schema is changed in between the audits, it can lead to wrong results very quickly. A process, as it enters the execution pipe, is passing the audit with flying colors under one schema when the schema is changed as a result of recommendation provided at the end of previous process audit which just finished after this process entered the execution pipe. Under the new schema

change, the same process can possibly fail the audit. This also creates a consistency problem as the results of the new schema cannot be combined with the results from previous schema to perform the analysis.

We suggest that schema changes only be implemented at pre-defined periodic intervals, such as every Sunday midnight in a typical workplace. We also suggest that process audit should be stopped and the new schema takes effect after finishing the current execution batch; we also note that this issue is related to concurrency problems in transaction processing.

How to implement this change in weight schema is another challenge that we encountered. It can be implemented as a policy where a new policy takes effect at a certain pre-defined time in future thus allowing for the processes to finish before the new schema takes over. Another possibility is to use rule-based reasoning [31] but it requires the use of expert systems and employs extensive use of fuzzy logic and reliance on AI. Since the semantic web domain is not fully functional yet, we suggest the use of policy based approach where the new policy takes effect at a pre-determined interval. This new policy is reflected by changing the weights of the process steps in the Master Document described in the next sub section.

3.2.3 Master Document

A major challenge in moving the CPA into the digital domain as described in previous sections is bridging the gap between business processes and IT technology. The specific challenge is how to use the information about the business processes and convert that information from business language into technology specifications. In simple terms, the information regarding how the process should work is used by the human auditors to

perform auditing in a traditional audit environment. This information is present in the knowledge domain usually as a business specification in natural language and is common knowledge for most of the auditors. The auditors then use this information to perform assess if the execution of the process under audit conformed to those specifications.

The problem is to change those specifications from the business domain (or natural language) into technology specifications that can be used by the automated programs to perform the task. We propose the concept of a Master Document to resolve this problem. Master Document is a document that contains the details about a given process. These include, but are not limited to, Process ID, number of steps for process, the weight schema for each process step, the minimum threshold score, passing score, failing score, so on and so forth. Thus this document contains all the information that is required to perform the analysis of the process. This document supplies all the information that the analysis component (described in the next sub-section) uses to perform its task.

One main challenge for design of a Master Document is the format it should have. It can be an XML document with details about a given process as sub tags or it can be a delimiter based document. Until the overhead involved in processing the XML document is minimized, this option is not viable to use. Another interesting option is the uses of predicate-based approach where the knowledge in the Master Document is divided into a “subject, predicate, object” triplet. This approach can be used by employing ontological frameworks such as Resource Description Framework Schema (RDFS) [27]. The advantage of this approach is that a skeleton (based on the predicates) is created for a generic Master Document irrespective of the domain and underlying business application

used. This skeleton can just be populated with the “subjects” and “objects” supplied by the user and the Master Document can be generated without too much trouble.

After investigating this approach, we propose the following set of predicates shown in Figure 3.4 that can form the basis of a simple generic Master Document (based on our case study described in the next Chapter).

```

Every Survey "has a goal" Goal
Every Survey "has number of processes" NumProcesses
Every Process "has an ID" ProcessID
Every Process "has passing score" ProcPassScore
Every Process "has failing score" ProcFailScore
Every Process "has minimum threshold score" ProcMinThrshldScore
Every Process "has current audit score" ProcCurntScore
Every Process "has total audit score" ProcTotalAuditScore
Every Process "has number of steps" ProcNumSteps
Every Step "has weight" Stepweight

```

Figure 3.6: Sample predicate set

The list in Figure 3.6 shows predicates that we propose in order to construct a generic Master Document. For a predicate like “has number of steps”, all the user needs to provide is the Process ID and number of steps for that process to generate the Master Document rule. Such rules combine to form the business knowledge that can be used by the programmable module to perform the audit analysis. Although we provide a basic list of predicates, the use of ontology via RDFS is still in its infancy and more work needs to be done in this field before this technology can be adopted. Hence, we suggest the use of delimiters to form the Master Document at the current time with hope that our initial list will encourage researchers to join with domain experts to come up with a complete and generic predicate skeleton for future use. For the issue of updating the Master Document according to the new schema, we suggest that it should be done during a pre-determined interval in order to keep the consistency of the system intact.

In general, then, the Master Document contains the rules under which the processes in the given domain operate. These rules are defined by the stakeholders while designing the application for the particular domain and usually defined in the business specifications document for each process involved. These rules to be extracted to create the Master Document by interviewing the stakeholders and observing system behaviour (e.g. monitoring functions that take the input and produce the desired output thus constituting a step of a given process and the restrictions applied on the functions define the rules under which the given process should operate). These rules can differ for each process of the application in the given domain and can be combined together to form the Master Document. There can be several Master Documents if the application is divided into different modules (with one Master Document for each module and all Master Document kept in a library) where each modules contains completely different set of processes and depending upon the requirement for analysis, the particular Master Document can be retrieved from the library to get the required rules for the process under audit.

3.2.4 Analysis and Reporting

A major challenge in moving CPA into the digital domain involves translating the information available into actions that are to be performed based on the information. These actions can range from deploying triggers to take particular actions to displaying the results in a way that intended information is delivered properly.

Our proposed analysis and reporting mechanism takes care of these problems. It takes the information already gathered by previously mentioned components and uses it to determine the course of action necessary to be deployed. The analysis component,

takes into account the information from various resources to perform its job. It takes the input from Slogs about the actual steps that have been performed during the process and then compares them with the information available in the master document to determine if the audit has passed or failed. It uses the individual steps recorded in the Slogs and applies appropriate weights to each step by using the information from Master Document. Based on the calculations it declares the audit as a pass, failure, more complicated as desired (just pass, pass with flying colors, etc) or triggers alarms to take appropriate actions. The result of this analysis is rendered to the user.

We propose a simple text based format for displaying the information. This can be a simple display message (Pass, Fail etc) or a pre-determined text based message with the result filling in the blanks. An issue with the rendering includes displaying of partial contents of the report to the user and depending upon the need, it can be useful or unnecessary. For example displaying the individual results of a process which calculates the base salary may prove to be futile and at times incorrect as the final salary would have to include the benefits and overtime pay. On the other hand, the individual result of a patient failing a heart beat test needs to be conveyed immediately to take appropriate actions without waiting for the full result of the complete check-up involving sugar, temperature and rest. The result can be displayed in a simple text based rendering or a more complex XML based format and we leave the design and format of the rendering to the language and domain experts for future.

CHAPTER IV

CASE STUDY AND DISCUSSION

In this chapter we present a detailed implementation process for auditing a survey application, based on our proposed model as a case study. Also, some experiments are conducted to demonstrate the use of our continuous auditing model and show the benefits of using process auditing as opposed to traditional auditing.

4.1 Background

This case study is built on top of the work done towards the creation of a real-time data management and decision support system [32] [33] which is a research project at University of Windsor. The purpose of this case study is to provide continuous auditing of a survey application with different embedded processes with the aim of providing real-time assurance to increase the efficiency of the application. It consists of creating a sample survey using the automated survey building tools created by the researchers at University of Windsor [33] and performing an audit of the results on a continuous basis to improve the efficiency.

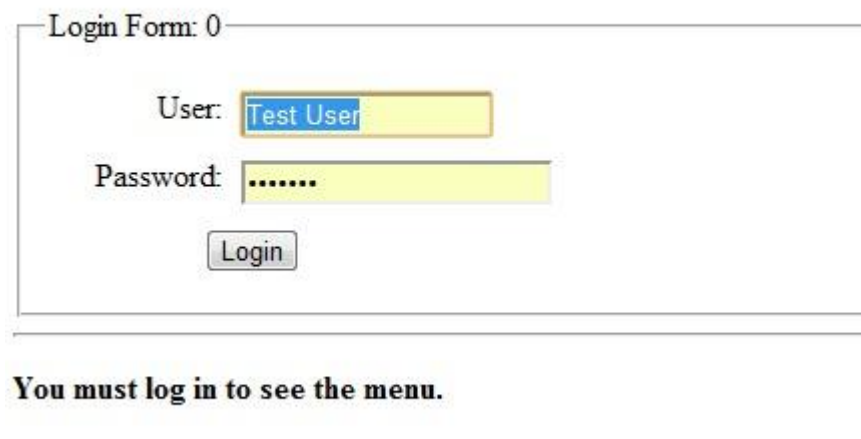
4.2 Implementation

In this section we discuss the implementation of a system, in the form of a survey for Infection Regime Control, used for a case study in applying our CPA framework and approach.

4.2.1 Infection Regime Control Survey

During the first step of this case study, a sample survey is built using the automated survey generator tool. The sample survey is used by healthcare providers to monitor the infection regime control [34] in their facilities. User is provided with the specific URL of the survey and the credentials to log in to the system as shown in Figure 4.1.

Home page



The image shows a login form titled "Home page". At the top left, it says "Login Form: 0". Below this, there are two input fields: "User:" with the text "Test User" and "Password:" with a masked password ".....". A "Login" button is positioned below the password field. Below the form, there is a message: "You must log in to see the menu." The form is enclosed in a box with a thin border.

Figure 4.1: User Control

As shown in the figure, user has to enter the credentials before accessing the sample survey. This aids in minimizing the security issues as only authorized personal can enter the system and also helps in verifying that controls are in place and are consistent with the success factors. Hence an important part of process auditing is taken

care of at this level. Once the user credentials are verified, he is taken to the next screen which displays the survey as shown in Figure 4.2.

The screenshot shows a web form titled "Continuous Audit Sample Survey". The form is enclosed in a rectangular border. At the top left, the title "Continuous Audit Sample Survey" is displayed in a bold, black font. Below the title, the first section is a question: "Are you a Staff or a Patient ?". This question is followed by two radio button options: "Staff" and "Patient". Below this section, there are six tabs, each with a label: "Add 'Visitor and Patient Information.'", "Add 'General Information: This section is based upon observation of normal staff procedures in the ward.'", "Add 'The Ward Environment: Please decide if you agree or disagree with the following statements based on your observations.'", "Add 'Waste Disposal: Please select the option that most closely corresponds to your view regarding the following statements.'", "Add 'Care of Equipment: Please select the option that reflects your observation.'", and "Add 'Hand Washing: Please select the option that most closely relates to your observation'". At the bottom right of the form, there is a "Submit" button.

Figure 4.2: Sample survey with processes

As shown in Figure 4.2, the continuous audit sample survey consists of different sections (shown as tabs) each of which represents a process. Each section can be clicked and displays inter related questions that represent the steps of the process. A sample process (section) with steps (inter related questions) is shown in Figure 4.3.

Continuous Adult Sample Survey

Are you a Staff or a Patient ?

Staff
 Patient

Visitor and Patient Information.

Leaflets and posters were available to visitors explaining correct hygiene measures when visiting

Agree
 Disagree

All patients are given information on MRSA on discharge

Agree
 Disagree

All staff are able to explain to visitors reasons for infection control procedures

Agree
 Disagree

Add 'Visitor and Patient Information.'

Add 'General Information: This section is based upon observation of normal staff procedures in the ward.'

Figure 4.3: Sample survey with expanded process

After filling all the sections appropriately, the user presses the submit button at the bottom to submit the results. The responses are then gathered and transformed into an XML format before being stored permanently [33]. The response XML document contains the question the user answered, the answers to the questions and some personal user data for identification as shown in Figure 4.4.

```

<?xml version="1.0" encoding="UTF-8" ?>
<response id="12" version="2.0.1">
<content>
  <question id="1">
    <answer>1.1</answer>
  </question>

  <question id="2">
    <answer>1</answer>
  </question>

  .
  .
  .

  <question id="1" templateid="15">
    <question id="15.1">
      <answer>15.1.1</answer>
    </question>
    <question id="15.2">
      <answer>15.2.3</answer>
    </question>
    <question id="15.3">
      <answer>15.3.1</answer>
    </question>
    <question id="15.5">
      <answer>15.5.2</answer>
    </question>
  </question>
</content>

<meta>
  <entered.by login="Test User" at="2456029.0471064816" />
  <deploy.info as="Test 1" survey.id="9" />
</meta>
</response>

```

Figure 4.4: Sample response XML file

4.2.2 Sticky Log Implementation

Sticky logs are implemented in two steps in our system. In the first step, after getting the responses from the user, we decided to apply eXtensible Style Sheets to extract the necessary data for performing the audit. The XSLT is designed as such to extract the minimum information required to perform the audit without losing any critical information. The resultant data consists of Process ID, Question ID and User Answer each separated by a period and terminated by a semi-colon (e.g. 10.1.2; means Process 10, Question 1 and user chose option 2 for the answer). These responses are stored in the

sticky log document which is in a comma separated value (csv) format for easy of portability and cost effectiveness shown (in bold) in Figure 4.5 below.

The screenshot shows a web-based XML transformation tool. The left pane displays the input XML code, which is a response document containing a list of questions and answers. The right pane displays the XSLT stylesheet used for transformation, which includes instructions for outputting the data as text and for each question and response. The bottom pane shows the resulting CSV output, which is a single line of comma-separated values representing the survey data.

```

XML Code:
<?xml version="1.0" encoding="UTF-8" ?>
<response id="12" version="2.0.1">
  <content>
    <question id="1"><answer>1.1</answer></question>
    <question id="2"><answer>1</answer></question>
    <question id="3"><answer>3.1</answer></question>
    <question id="4"><answer>4.1</answer></question>
    <question id="5"><answer>4/11/2010</answer></question>
    <question id="6"><answer></answer></question>
    <question id="9"><answer></answer></question>
    <question id="1" templateid="10">
      <question id="10.1"><answer>10.1.1</answer></question>
      <question id="10.2"><answer>10.2.1</answer></question>
      <question id="10.3"><answer>10.3.1</answer></question>
    </question>
    <question id="1" templateid="11">
      <question id="11.1"><answer>11.1.2</answer></question>
      <question id="11.2"><answer>11.2.3</answer></question>
      <question id="11.3"><answer>11.3.2</answer></question>
    </question><question id="1" templateid="12">
      <question id="12.1"><answer>12.1.1</answer></question>
      <question id="12.2"><answer>12.2.2</answer></question>
      <question id="12.3"><answer>12.3.1</answer></question>
      <question id="12.4"><answer>12.4.2</answer></question>
      <question id="12.5"><answer>12.5.1</answer></question>
  </content>
</response>
XSLT Code:
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- example with if-else printing template id -->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output
method="text"/>
  <xsl:template match="/">
    <!-- print meta data information -->
    <!-- get question survey id -->
    <xsl:for-each select="response/meta">
      <!-- <tr><td -->
        <xsl:value-of select="deploy.info/@survey.id"/>,
      <!-- </td></tr -->
    </xsl:for-each>
    <!-- get response id -->
    <xsl:for-each select="response">
      <!-- <tr><td -->
        <xsl:value-of select="@id"/>,
      <!-- </td></tr -->
    </xsl:for-each>
  </template>
</xsl:stylesheet>
Your Result:
9, 12, Test User, 2456029.0471064816, 1.1; 1; 3.1; 4.1; 4/11/2010; ; ; 10.1.1; 10.2.1; 10.3.1; 11.1.2; 11.2.3; 11.3.2; 12.1.1; 12.2.2; 12.3.1; 12.4.2; 12.5.1; 14.1.1; 14.3.2; 14.4.2; 15.1.1; 15.2.3; 15.3.1; 15.4.2; 15.5.2;

```

Figure 4.5: Sample slog document

This sticky log document is sent to the audit program where the results are generated with the help of using the Master Document. This sticky log document is also stored permanently to be used later on for audit trailing purposes.

In the second part of the sticky log implementation, the data from this sticky log is extracted and stored in relevant data structures. Survey Object sits at the top hierarchy of the data structures and contains a vector of Process Objects along with other identifying data. Vector is used as a preferred sequence container due to its consistent design, low memory usage, ease of implementation, linear time complexity for creation and good locality of reference [35]. Each Process Object in turn contains a vector of Question

Objects. Once the data is stored in these data structures (which are only used as placeholders), the second part of sticky log management is performed in which data is extracted from these data structures and stored in corresponding Survey Sticky Log, Process Sticky Log and Question Sticky Log data structures.

One reason for creating separate data structures for sticky logs is that sticky logs contain more information than the normal data structures (for survey, process and questions). This extra information is supplied from the Master Document and stored in the data structure in order to have all relevant information about the process in one place to reduce time for lookup thus increasing the efficiency and reducing cost of operation. Another reason for creating separate data structures is to make sure that core elements of the framework are defined and working so that the framework works once ported to other applications. In our case study, sticky log data is supplied in two steps due to the system design and data does not flow directly from the production system (as the user manipulates the data on front end) into the data structures.

For other systems or applications, event listeners can be placed inside the system which extract the relevant data and send them directly to the sticky log objects for auditing. The process of extracting data from Master Document and updating the sticky log objects is shown in Figure 4.6.

```

//get user data for each process in the ProcesstickyLog
proc_sl_vector = survey_sl_obj.getProcessStickyLog();
for (p_sl_pos1 = proc_sl_vector.begin(); p_sl_pos1 != proc_sl_vector.end(); ++p_sl_pos1)
{
    int num_ques_ans = (p_sl_pos1->getQuestionStickyLog()).size();
    p_sl_pos1->setNumQuestions(num_ques_ans);
    int theID = p_sl_pos1->getProcID();
    m_p_pos1 = find_if (m_proc_vector.begin(),m_proc_vector.end(),
    [theID](const MasterProcess& other)
    { return other.getProcessID() == theID; });
    if (m_p_pos1 != m_proc_vector.end())
    {
        p_sl_pos1->setMinThresholdScore(m_p_pos1->getProcessMinThresholdScore());
        p_sl_pos1->setProcessTotalScore(m_p_pos1->getProcessTotalScore());
        p_sl_pos1->setProcessTotalValue(m_p_pos1->getProcessTotalAuditValue());
    }
    else
    {
        cout << "Process not Found in Master Document " << endl;
    }
}
}

```

Figure 4.6: Sticky log updating process

4.2.3 Master Document Implementation

Master Document is implemented using the same structure for data as for the sticky logs in order to ensure consistent design pattern. Master Document object consists of a vector of Master Processes and each master Process contains a vector of Master Questions. Since a question can have multiple answer options, a new data structure is added for capturing this detail and each Master Question contains a vector of Master Answers. Master Document data is stored in an XML file format shown in Figure 4.7.

```

<?xml version="1.0" encoding="UTF-8" ?>
<masterDoc>
.
.
.
<process id="11" numQuestions = "4" minThresholdScore="0.75" totalScore="1.0" totalValue="0.1">
  <question id = "1" weight="0.25" answerOptions="2">
    <answer score="0.25">1</answer>
    <answer score="0.0">2</answer>
  </question>
  <question id="2" weight="0.25" answerOptions="3">
    <answer score="0.25">1</answer>
    <answer score="0.0">2</answer>
    <answer score="0.15">3</answer>
  </question>
  <question id="3" weight="0.25" answerOptions="2">
    <answer score="0.25">1</answer>
    <answer score="0.0">2</answer>
  </question>
  <question id="4" weight="0.25" answerOptions="2">
    <answer score="0.25">1</answer>
    <answer score="0.0">2</answer>
  </question>
</process>
.
.
.
</masterDoc>

```

Figure 4.7: Sample master document XML file

As shown in Figure 4.7, each process in the Master Document contains the necessary information about the process such as number of questions, minimum threshold score for process to pass the audit, the total value of the process in the overall audit and so on. Each question in the audit also contains the information about the question and the answer options. For ease of laboratory implementation, we have implemented this Master Document by hard coding it directly into the system. By using the data from the XML file, processes and questions are initialized using default constructors.

4.2.4 Audit Process Implementation

As mentioned in previous sections, the user response (initial sticky log) is saved in the comma separated value format (csv) with each response delimited by a semi-colon. In the first step of the audit implementation, this file is read word by word and each word is stored in a vector from where the user data is extracted to be stored in the relevant data structures. The time complexity of this algorithm is $O(n)$, where n is the number of words

in the csv file (i.e. the number of questions user has answered in the survey). The space complexity of storing the answers in the respective sticky logs is $O(n)$ also, where n is the number of elements stored in the respective sticky log. The pseudo-code, shown in Figure 4.8, summarizes this extraction process.

```

Algorithm Data_Extraction(Vector v, words_user_input{s1,s2,s3,...sn})
For all words in v do
  extract word s and put in a stringstream
  read first integer from the stream and store in integer new_proc_num
  ignore the delimiter
  read question string from the stream
  create question object and stickyLogQuestion object using the question string as id
  compare new_proc_num with already stored integer current_proc_num
  if (current_proc_num == new_proc_num)
    //process already exists
    get the last process object from process vectors of the survey object
    add the question object in the question vector of that process
    get the last stickylogprocess object from process vectors of the surveystickylog object
    add the questionstickylog object in the question vector of that processtickylog
  else
    //process does not exist
    create new process and processtickylog objects using new_proc_num as id
    add the question object to the newly created process object
    add the questionstickylog object to the newly created stickylogprocess
    add the process object to the survey object
    add the stickylogprocess to the stickylogsurvey object
    current_proc_num == new_proc_num

```

Figure 4.8: Data extraction pseudo code

After extracting the data from the user response, the next step involves getting relevant data from Master Document into respective Process Sticky Log object as shown by the sticky log updating process in figure 5.6 previously. Next step involves performing the audit operations on the Process Sticky Log objects. This is done by going through the list of all master processes in the Master Document one by one and scoring the audit result. The time complexity of getting every process from the Master Document is $O(n)$, where n is the number of processes in the Master Document. Time complexity of determining if the process exists in the Process Sticky Log is $O(p)$, where p is the number of processes found in the Process Sticky Log.

The reason for choosing the Master Process as the main criteria to perform audit is that a user may choose not to answer any given question or whole process in the survey. Even though the user did not respond to a few questions, they still have to be

scored (awarded a score of zero in this situation) and the end audit result must account for this missed question and display it properly in the audit report.

Hence, for every question in the Master Document, the sticky log is checked. If that question is present in the sticky log, it is given the appropriate score depending upon the user answer value. If the question is missing in the sticky log, it is given a score of zero, and added into the list of failed questions for the respective process. Also if the user answers a question but choose any other option other than the first, it is regarded as a failed step. The time complexity of getting every question for the given process from Master Document is $O(m)$, where m is the number of question in the current process. The time complexity of finding the current question in Question Sticky Log of the current question (i.e. finding if the user answered this particular question or not) is $O(q)$, where q is the number of questions user answered for the current process. In the end, all the scores for every question in the process are combined to generate the process score. All the process scores are added in the end (after multiplying with their respective weights for each process) to obtain the final audit score.

The algorithm for this audit scoring procedure is shown in Figure 4.9. The overall time complexity of this algorithm is $O(np+nmq)$, where n is the number of processes in Master Document, p is the number of processes in Process Sticky Log, m is the total number of questions for process n and q is the number of questions users answered for process n . The space complexity of storing the results of this algorithm in respective audit objects (question audit, process audit etc) is $O(nm)$, where n is the total number of processes found in the Master Document and m is the total number of questions in the Master Document.


```

Algorithm Audit(Vector MPV, master_process_objects{mp1,mp2,mp3,...mpn})
For all master process objects in MPV do
  extract the process ID of the object
  create audit_process object using the process ID
  check if current master process ID matches with Id of processtickyllog
  objects in the processtickyllog vector
  if process found in Process sticky log
  //user answered a question of the process in the survey
  get the list of masterquestion for the current masterprocess from
  masterdocument in vector MQV
For all the master question objects in MQV
  extract the question ID of the object
  create question_audit object using the question ID
  check if current master question ID matches with Id of
  questionstickyllog objects in the questionstickyllog vector
  if question found in Question sticky log
  //user answered this question
  extract the user answer for the current question from question sticky log
  find the score of the user answer from the vector of master answer objects
  of the current question from the master document and update the score in
  current question sticky log object
  if the user answer matched with the id of the first answer in the master
  answer vector, mark the question result as 'Pass' and add to the list of
  passed_question for the process
  else set question result as 'Fail' and add to the list of failed_questions
  for the process
  else question not found in question sticky log
  assign a score of zero to the question audit
  mark question result as 'Fail'
  add the question to the list of failed_questions for the process
  add current question audit object to the vector of questions object of the
  current process_audit object
  calculate audit score for the process (by adding scores of each
  individual questions)and store in the process_audit object
  if the process total score > min_thrshold_score
  set process audit result as 'Pass'
  else
  set process audit result as 'Fail'
  add process audit in the process_audit vector audit object
  else process not found in Process Sticky Log
  assign a score of zero to the process_audit
  add process audit in the process_audit vector audit object

```

Figure 4.9: Overall audit pseudo code

4.3 Verifications and Results

We assess our CPA framework by auditing a sample survey and generating the audit results by comparing the user responses with the data in Master Document. The user enters the website through a given username and password and is taken to the survey. Once the user fills out the survey, an XML file is generated based on the results. An XSLT schema is applied to these results in order to obtain minimum but complete information that captures all user responses accurately. The resultant comma separated value (csv) file is then sent as input to the audit program which performs the verification tests and generates audit results. Since the result of an audit is an audit report which is

text based, we decided to perform verification tests that take different survey's submitted by user to check the system behaviour under the said circumstances and generate the audit result in the form of a report which is displayed on the console.

4.3.1 Verification of Controls

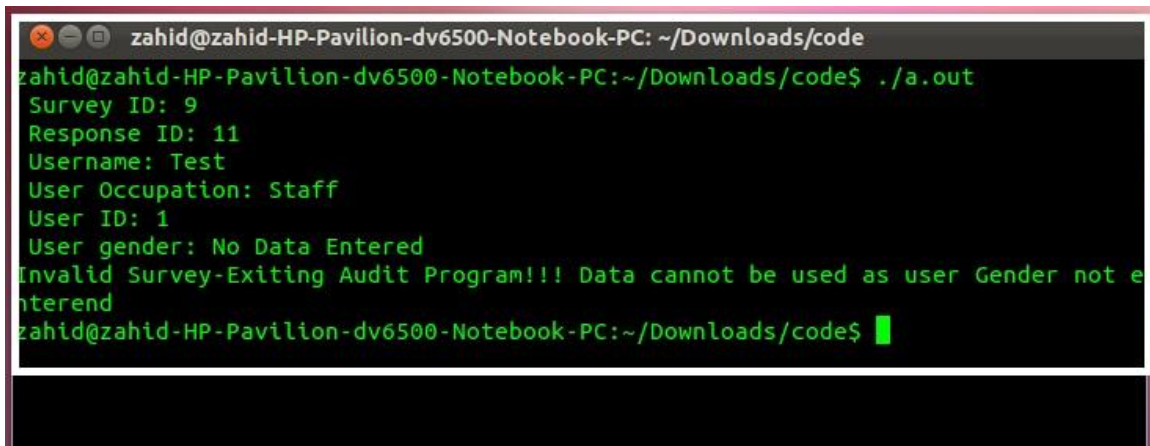
As outlined in previous chapter, in order to improve the efficiency of the process it is imperative that controls placed in the system are consistent with the success factors. This requires that system should have controls in place that invalidate any unauthorized activity/behaviour. Controls are implemented in our framework on two different levels. On the first level, a user is only allowed to access a survey by entering a valid username and password. This stops any unknown access to the survey and make sure that only authorized personal are allowed to enter the survey (such as a staff working at the hospital etc). The username and password are supplied to the user and security can be further enhanced by making sure that the username and password expires after a certain time period. Currently the username/password remains valid for the lifetime of the survey application and user can enter as many responses as they like. However, each response is given a separate ID and treated as a new survey. Once submitted, the user is not allowed to modify a survey and this adds another level of security by making sure that no genuine user data is altered.

The second level of control verification in our system is achieved by monitoring the user behaviour once they successfully log in to the system. Even though the user is an authorized personal, they might still enter incorrect data to deliberately introduce inconsistency in the data. This can range from not entering some mandatory fields to entering wrong data on purpose to forge the results. In our framework, we provide some

mandatory fields which provide useful information about the user and can be used in future for statistical purposes. These mandatory fields include the User ID, User Gender, User Shift and the Date on which the survey was entered. In our first control verification experiment, we deliberately submit the survey by missing a mandatory field and run the audit program to see if the system catches this behaviour or not. The user input for this experiment is shown in Figure 4.10. As shown in the highlighted part of Figure 4.10, a mandatory field (for question 3) is missing in the user response. This user response is fed into the audit program and the goal of the experiment is to find out if the audit program catches this missing behaviour. The framework should catch this behaviour and provide appropriate report which is displayed through the console. The result of this verification experiment is shown in Figure 4.11.

Continuous Audit Sample Survey	
Are you a Staff or a Patient ?	
<input checked="" type="radio"/> Staff <input type="radio"/> Patient	
ID	
1	
Gender	
<input type="radio"/> Male <input type="radio"/> Female	
Shift	
<input checked="" type="radio"/> Morning <input type="radio"/> Evening <input type="radio"/> Night	
Date	
4/11/2010	

Figure 4.10: User input with missing mandatory field



```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$ ./a.out
Survey ID: 9
Response ID: 11
Username: Test
User Occupation: Staff
User ID: 1
User gender: No Data Entered
Invalid Survey-Exiting Audit Program!!! Data cannot be used as user Gender not e
nterend
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$ █

```

Figure 4.11: Audit report for a survey with missing mandatory field

As shown in Figure 4.11, our CPA framework catches that the user did not enter a mandatory field in the survey response and invalidates the whole survey. For our second control verification survey, we enter a survey in which the user answers the questions incorrectly on purpose. The intent can be to forge a desired result by entering specific data to target one or other parts of the survey. A control is required in the system that catches this behaviour as it becomes obvious and moves beyond a certain threshold limit. For our framework, we decided to set the limit on number of question that can be answered incorrectly in a given survey. If the total number of question that are answered incorrectly increase beyond the given threshold, the system catches it and generates appropriate reports. The csv input file for the second control experiment is shown in Figure 4.12.

```

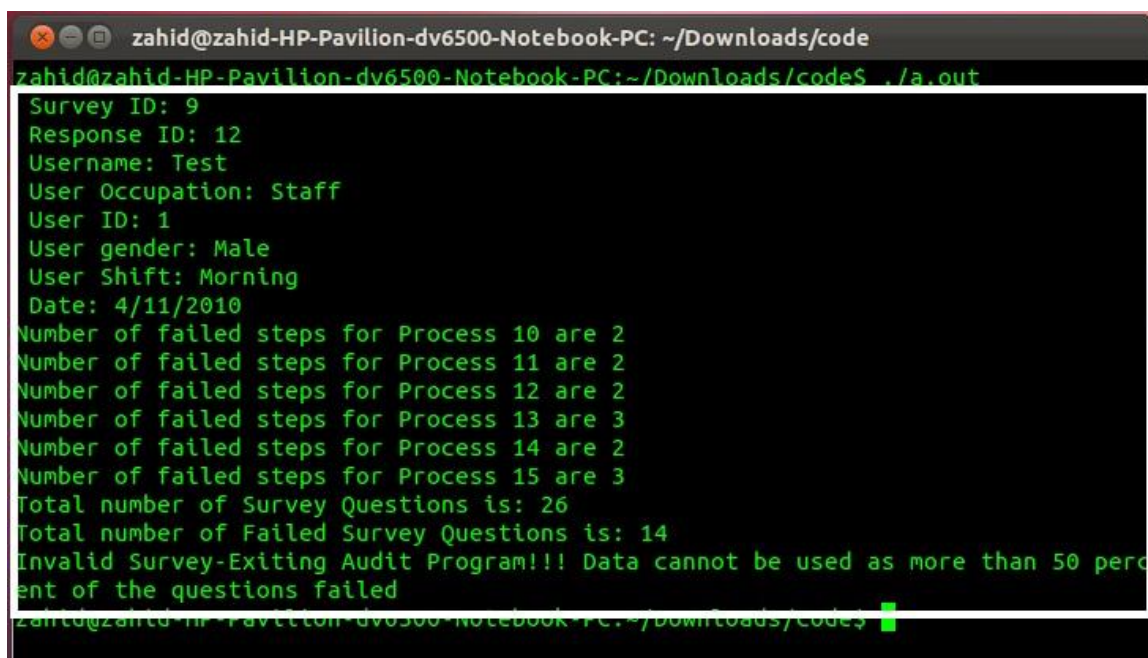
*csv-test2 ✕
9,
12,
Test User,
2457029.0471064816,
1.1;
    1;
    3.1;
    4.1;
    4/11/2010;
    ;
    ;
    10.1 1;
    10.2 2;
    10.3 2;
    11.1 1;
    11.2 1;
    11.3 2;
    11.4 2;
    12.1 1;
    12.2 1;
    12.3 1;
    12.4 2;
    12.5 2;
    13.1 1;
    13.2 1;
    13.3 3;
    13.4 3;
    13.5 3;
    14.1 1;
    14.2 2;
    14.3 1;
    14.4 2;
    15.1 1;
    15.2 3;
    15.3 1;
    15.4 2;
    15.5 2;

```

Figure 4.12: CSV file with deliberate incorrect answers

As shown in the highlighted part of Figure 4.12, incorrect answers are chosen deliberately to exceed the threshold value for incorrect answers for a given survey. We decided to keep this value at 50 % meaning that more than 50 % of the total number of

questions cannot be answered incorrectly. If the survey response contains more incorrect answers than this threshold value, the system catches this behaviour, interprets it as malicious behaviour and discards the results. This threshold value is decided by the system owners to best cater their specific business needs and should be chosen carefully to catch any anomalies and adjusted. The result of this experiment is shown in Figure 4.13.



```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$ ./a.out
Survey ID: 9
Response ID: 12
Username: Test
User Occupation: Staff
User ID: 1
User gender: Male
User Shift: Morning
Date: 4/11/2010
Number of failed steps for Process 10 are 2
Number of failed steps for Process 11 are 2
Number of failed steps for Process 12 are 2
Number of failed steps for Process 13 are 3
Number of failed steps for Process 14 are 2
Number of failed steps for Process 15 are 3
Total number of Survey Questions is: 26
Total number of Failed Survey Questions is: 14
Invalid Survey-Exiting Audit Program!!! Data cannot be used as more than 50 per
cent of the questions failed
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$

```

Figure 4.13: Audit report for a survey with incorrect answers beyond threshold

As shown in Figure 4.13, the total number of questions in the survey is 26 and user answered 14 questions incorrectly (incorrect in our case means user disagrees with the questions). Since the number of incorrect answers exceeds the minimum threshold value of 50 %, the survey is invalidated and results are discarded. As shown in Figure 5.11 and Figure 5.13, our CPA framework catches any violations of the controls placed in the system thus making sure that controls in place are consistent with the success factors. It also helps in establishing that the process (overall audit regarded as one process here) is

competent and capable of handling any anomaly produced either by error or misuse. Thus the results show that our CPA framework is capable of enforcing the controls placed in the system in order to increase the overall efficiency of the process (by discarding the results produced by violating the controls) which is the whole purpose of performing a process audit.

4.3.2 Verification of Core Elements

As discussed in section 3.2.1 and section 3.2.3, sticky logs and master document form the core elements of our CPA framework. Sticky log contains the relevant user response data and master document contains the scores for the specific user responses. Together, these form the basis of our framework and it is important that these work correctly for the system to perform as desired. The verifications tests in this section aim at making sure that these core elements are working properly and carrying the correct data as entered by the user.

There is one sticky log for each survey which contains a vector of process sticky logs. These represent the different processes (question templates) that are present in the survey. Each process contains a set of questions that are related and work towards the common goal of the process.

For example, questions regarding information provided to the user are grouped under the “Visitor and Patient Information” process as shown in Figure 4.3. The corresponding data for each question/process response is stored in the Master Document. It also contains the useful information about how many questions are in each process, the weight of each question in the process and the overall weight of the process in the survey. The sticky logs are populated by entering this important data along with the user response

data. The questions/processes which are more important are given more weightage and are critical in determining the outcome of the overall audit result.

Thus, it is imperative that these core elements work correctly and at all times and contain the correct information as provided by the user. The sticky logs contain the questions/process that are entered by the user only and thus can miss some questions/processes if the user decides not to provide an answer for them. Although some questions/processes are missing, they must be accounted for when calculating the total audit score for the process/survey. Therefore it is important that correct user data is captured in the sticky log and the information in the master document is complete to perform the necessary calculation and produce correct results. In the first core element verification test, the contents of sticky logs are printed to make sure that they contain the correct data as entered by the user.

A sample result of this test is shown in Figure 4.14. As shown in the highlighted part of Figure 4.14, the sticky log shows the correct data as entered by the user. It shows the questions and answers users provided for each process along with important process data used to perform the audit. The input csv file and the complete output for this test (showing all the process sticky logs) are attached in Appendix A. In the second test, the contents of master document are printed to show the data that they contain. The sample result of this test is shown in Figure 4.15.

```
zahid@zahid-HP-Pavilion-dv6500-Noteb
=====
Process Sticky Log ID: 15
Process Minimum Threshold Score: 0.8
Process Total Score: 1
Process Total Audit Value 0.2
Question Sticky Log ID: 1
Question Sticky Log Ans: 1
-----
Question Sticky Log ID: 2
Question Sticky Log Ans: 3
-----
Question Sticky Log ID: 3
Question Sticky Log Ans: 1
-----
Question Sticky Log ID: 4
Question Sticky Log Ans: 2
-----
Question Sticky Log ID: 5
Question Sticky Log Ans: 2
-----
=====
Survey Valid
```

Figure 4.14: Sticky log for survey

```
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Down
User Shift: Morning
Date: 4/11/2010
=====
Master Process ID :10
Number of Questions in Master Process: 3
Minimum Threshold Score for Master Process: 0.7
Total Score for Master Process: 1
Total Audit Value for Master Process: 0.1
Master Question ID: 1
Master Question Weight: 0.3
Master Question Answer Options: 2
-----
Master Answer ID: 1
Master Answer Score: 0.3
-----
Master Answer ID: 2
Master Answer Score: 0
-----
Master Question ID: 2
Master Question Weight: 0.3
Master Question Answer Options: 2
-----
Master Answer ID: 1
Master Answer Score: 0.3
-----
Master Answer ID: 2
Master Answer Score: 0
-----
Master Question ID: 3
```

Figure 4.15: Master Document for survey

As shown in the highlighted part of Figure 4.15, the master document contains the data about all the process in the survey including important information such as the number of questions for each process, the minimum threshold score for the process, the value of the process in the overall audit, etc. It also shows the individual questions for the process along with the answer options and score/weightage for each answer option. This score is applied to the user answer in the sticky log to perform the scoring procedure. Hence it is important that this data remains correct and consistent at all times. The input csv file and the complete output for this test (showing all the master processes) are attached in Appendix A. Master Document contains the same data for each audit cycle but the sticky logs are undated as the new user data comes and hence it is important to make sure that they contain the correct data otherwise the audit results would be compromised. These results show that the core elements contain the correct data and aid in generating the audit results. As per our current system design, the sticky logs are populated by the user data supplied via the csv file but they can work with any system which allows for data extraction (e.g. by placing event listeners) and has a communication mechanism for supplying the extracted data.

4.3.3 Verification of Audit Process

As discussed in chapter 3, the purpose of auditing a process is to make sure that the process achieves its desired results which are consistent with the goal of the system. As this is a survey application, the first and foremost goal of the system is to pass the audit. The processes/questions in the survey are given weights and scores (based on their individual importance) and the audit results decide if the processes are working towards

achieving the goal of the survey. The goal of the process can vary according to the application.

For a survey application such as ours, the goal can be as simple as making sure that a certain question is answered in the survey. Thus, if that question is not answered the audit can fail, meaning that the question should carry more weight. Changing the weight can either mean giving more emphasis to the question (e.g. making sure that leaflets and posters explaining correct hygiene are available for visitors as shown in Figure 4.3), or changing the order of the question in the survey (to make sure that a particular question in the survey, which is always missed out due to being the last question, is moved up in the survey so that a response can be received).

Like sticky logs and master document, audit also consists of process audits and question audits which combine their results to generate the overall audit result. Another goal of the system is to provide alternate strategies in case a certain process/question fails.

In our survey application, this alternate strategy means changing the weights of the question in a way that makes sure that questions/processes that fail consistently are given new weights so that they are emphasized and are given more care and importance. This alternate strategy helps in establishing the process for achieving success as the constantly failing question can affect the overall audit result. In the context of our survey application, the alternate strategy means giving more importance to the practise that is failing constantly (e.g. washrooms not cleaned regularly or leaflets information not available for visitors or staff not able to answer visitor queries) and making sure that the

failed practise is given precedence when doing the practises grouped together as a process.

This is achieved by identifying the factors that affect the success of the particular module/process and forms another important goal of process auditing. In the first verification test for audit process, the user supplies a regular input (user can still choose to not answer any question/process except the mandatory part which are answered to make the survey valid) and audit result is calculated based on the user responses. The result of the test is shown in Figure 4.16.

```
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Down
Question ID: 3
Question Score: 0.4
Question Audit Result: Pass

Final Audit Score of Process 10: 0.1
Process Audit Result: Aced
=====
Process ID: 11
Process Resultant Score: 0.9
Process Minimum Threshold Score: 0.75
Question ID: 1
Question Score: 0.25
Question Audit Result: Pass
Question ID: 2
Question Score: 0.15
Question Audit Result: Fail
Question ID: 3
Question Score: 0.25
Question Audit Result: Pass
Question ID: 4
Question Score: 0.25
Question Audit Result: Pass
-----
Final Audit Score of Process 11: 0.09
Process Audit Result: Passed With Flying Colors

Process ID: 12
Process Resultant Score: 0.8
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
```

Figure 4.16: Sample audit result for processes and questions

As shown in the highlighted part of Figure 4.16, audit is performed on each question resulting in either a “Pass” or a “Fail”. For our survey application, if the user agrees with the survey question, the question is passed and if the user disagrees with the question, it is regarded as a failure. Depending upon the result of the individual questions, each process is given a final score and a result.

We have used the notion of different results for different score ranges and in Figure 4.16, process 10 has a result “Aced” while process 11 has the result “Passed With Flying Colors”. A final process score of maximum is regarded as acing the result, where as a final score greater than the minimum threshold score is regarded as passing the result with flying colors. The aim of each audit is to get the maximum difference between the minimum threshold score and score generated which for our survey application means complete compliance (everything is working perfectly). However, perfection is not possible in all cases; therefore, the notion of minimum threshold score leaves an option of some parts of the audit to fail. The final results are added to generate the overall audit result as shown in Figure 4.17.

```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
Question Audit Result: Pass

Final Audit Score of Process 14: 0.1
Process Audit Result: Failed
=====
Process ID: 15
Process Resultant Score: 0.9
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
Question Audit Result: Pass
Question ID: 2
Question Score: 0.1
Question Audit Result: Fail
Question ID: 3
Question Score: 0.2
Question Audit Result: Pass
Question ID: 4
Question Score: 0.2
Question Audit Result: Pass
Question ID: 5
Question Score: 0.2
Question Audit Result: Pass
-----
Final Audit Score of Process 15: 0.18
Process Audit Result: Passed With Flying Colors
=====
Overall Audit Score: 0.81
Overall Audit Result: Passed With Flying Colors
=====
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$

```

Figure 4.17: Sample overall audit result that passed

As shown in Figure 4.17, overall audit score is calculated by calculating and adding the individual scores for each process/question along with their appropriate weights. As with the audits for questions and processes, the audit result depends upon the final audit score achieved and the minimum threshold score for the audit which for our survey application is based at 80 % mark. Thus, by choosing this minimum limit, in our survey we permit a failure of either two low priority processes (each containing an overall audit value of 0.1) for one high priority process (containing an overall audit value of 0.2). The csv input file and the detailed output of this test are attached in Appendix A.

Another test showing the results of failed individual process audit and failed overall audit are shown in Figure 4.18 and Figure 4.19, respectively, with the csv input file and detailed output attached in Appendix A.

```
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/cod
Question Audit Result: Fail
-----
Final Audit Score of Process 12: 0.16
Process Audit Result: Just Passed
=====
Process ID: 13
Process Resultant Score: 0.7
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
Question Audit Result: Pass
Question ID: 2
Question Score: 0.2
Question Audit Result: Pass
Question ID: 3
Question Score: 0.1
Question Audit Result: Fail
Question ID: 4
Question Score: 0.1
Question Audit Result: Fail
Question ID: 5
Question Score: 0.1
Question Audit Result: Fail
-----
Final Audit Score of Process 13: 0.14
Process Audit Result: Failed
=====
Process ID: 14
```

Figure 4.18: Sample audit result for failed process audit and question audit


```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
Process Audit Result: Failed
=====
Process ID: 15
Process Resultant Score: 0.7
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
Question Audit Result: Pass
Question ID: 2
Question Score: 0.1
Question Audit Result: Fail
Question ID: 3
Question Score: 0.2
Question Audit Result: Pass
Question ID: 4
Question Score: 0.2
Question Audit Result: Pass
Question ID: 5
Question Score: 0
Question Audit Result: Fail
-----
Final Audit Score of Process 15: 0.14
Process Audit Result: Failed
=====
Overall Audit Score: 0.69
Overall Audit Result: Failed
=====
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code$

```

Figure 4.19: Sample overall audit result that failed

As shown in the highlighted portion in Figure 4.19, the overall audit score is less than the minimum threshold for the process, hence the audit is failed. The detailed audit report contains the information about the individual process audits and question audits that failed. This report serves as the basis for finding out the inconsistencies in the system (practises that fail or do not help in infection control for our survey application) and works as a catalyst to change practises in order to increase the process efficiency and ultimately pass the audit.

Another important feature in process auditing involves finding the factors that are critical for success. These factors dictate the success or failure of a given module and therefore every audit needs to find the success factors for a given module in order to increase its efficiency.

For our survey application, the success factors include the question with highest weight in a process and the question that failed for that process. The question with the highest weight is included in the success factor as it is deemed most critical for the given process and is identified by the master document. Since the master document contains the perfect case scenario and is populated with careful consideration with the system stakeholders during the design process, the question with the highest weight is critical for process success.

The failed steps for a process are also included in the success factors as they dictate the passing or failure of the process audit hence affecting the overall process result. Identifying the success factors also improves the process efficiency as they are the critical elements of the process that determine its success and success of a process means higher efficiency. In our test for finding success factors, the user submits a sample survey which is sent to the audit program to perform the normal audit and identify the success factors. The sample result for this test is shown in Figure 4.20.

```
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downl
3
1
=====
Process ID: 14
Process Resultant Score: 0.5
Process Minimum Threshold Score: 0.75
Question ID: 1
Question Score: 0.25
Question Audit Result: Pass
Question ID: 2
Question Score: 0
Question Audit Result: Fail
Question ID: 3
Question Score: 0
Question Audit Result: Fail
Question ID: 4
Question Score: 0.25
Question Audit Result: Pass
-----
Final Audit Score of Process 14: 0.1
Process Audit Result: Failed
Factors Affecting Success:
2
3
1
=====
Process ID: 15
Process Resultant Score: 0.9
```

Figure 4.20: Sample audit result identifying success factors

As shown in the highlighted part in Figure 4.20, for process 14, question 2 and question 3 fail their respective question audits and hence are included in the success factors. Also, question 1 is identified as a success factor also being the first question with the highest weight. If there are two or more questions with the same high weight, the first amongst the list is included in the success factor as it is located on the top of the survey process and the priority of the questions on the survey dictate importance in case of a tie. Hence, our audit program successfully identifies the success factors for any given survey

that is valid. The csv input file and the detailed output for this test are attached in Appendix A.

The final verification test for audit process deals with an important feature of process auditing and involves establishing the process for achieving objectives. Since the objective is to pass the audit, it is imperative to have a solution in case a failure occurs. In the context of our survey application, failure means that a certain question has failed for a process. If any particular question fails, it is included in the success factors and due course of action is taken for it by the appropriate authorities. Despite the failure of a given question, the process audit can still pass depending upon the weight of the question in the process and the importance of the process in the overall audit. However, if a certain questions fails repeatedly time and again, our CPA framework takes it into account and suggests alternate strategies to make sure more care is given to that question.

In our survey application, each question that fails is accounted and a counter is stored for every question that fails. Every time that particular question fails again, the counter is incremented to keep track of the failure. After the failure of the same question goes beyond a minimum threshold value, the CPA provides an alternate strategy to make sure that more importance is given to this question.

For our testing purpose, we devised a test in which a particular question was deliberately failed every time. The threshold value for a question failure in our survey application is defined at 10. If a given question fails more than 10 times, the CPA provides an alternate weight schema for the whole process. It changes the weight of the current failed question with the question having the highest weight. For our experiment,

we entered a survey in which the question 2 of process 10 was answered incorrectly. The output of the first test is shown in Figure 4.21.

```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
Process ID: 15
Process Resultant Score: 1
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
Question Audit Result: Pass
Question ID: 2
Question Score: 0.2
Question Audit Result: Pass
Question ID: 3
Question Score: 0.2
Question Audit Result: Pass
Question ID: 4
Question Score: 0.2
Question Audit Result: Pass
Question ID: 5
Question Score: 0.2
Question Audit Result: Pass
-----
Final Audit Score of Process 15: 0.2
Process Audit Result: Aced
=====
Overall Audit Score: 0.97
Overall Audit Result: Passed With Flying Colors
=====
Process: 10
Question: 2 Number of times failed: 1
zant@zant@-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$

```

Figure 4.21: Sample audit result for alternate strategy – Pass 1

As shown in the highlighted part of Figure 4.21, question 2 for process 10 failed for the first time and this information is accurately shown in the audit result. An important point to note here is that this information is recorded despite the overall audit process passing with flying colors. We repeat the same input for the next experiment and the same result is shown expect that the number of times question 2 failed is increased due to the counter. The result of the test on the 10th pass is shown in Figure 4.22.

```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
Process ID: 15
Process Resultant Score: 1
Process Minimum Threshold Score: 0.8
Question ID: 1
Question Score: 0.2
Question Audit Result: Pass
Question ID: 2
Question Score: 0.2
Question Audit Result: Pass
Question ID: 3
Question Score: 0.2
Question Audit Result: Pass
Question ID: 4
Question Score: 0.2
Question Audit Result: Pass
Question ID: 5
Question Score: 0.2
Question Audit Result: Pass
-----
Final Audit Score of Process 15: 0.2
Process Audit Result: Aced
=====
Overall Audit Score: 0.97
Overall Audit Result: Passed With Flying Colors
=====
Process: 10
Question: 2 Number of times failed: 10
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$ █

```

Figure 4.22: Sample audit result for alternate strategy- Pass 10

As shown in the highlighted part of Figure 4.22, the audit program finds that the question 2 of process 10 has failed for the 10th time but since this failure is still within the threshold limit, no action is taken. For the action to trigger, the number of times a question fails must be greater than the minimum threshold value. In the 11th pass (with the same input from user where question 2 fails), the audit programs finds that the question has failed more than the threshold value and provides an alternate strategy shown in Figure 4.23.

```

zahid@zahid-HP-Pavilion-dv6500-Notebook-PC: ~/Downloads/code
Question Audit Result: Pass
Question ID: 5
Question Score: 0.2
Question Audit Result: Pass
-----
Final Audit Score of Process 15: 0.2
Process Audit Result: Aced
=====
Overall Audit Score: 0.97
Overall Audit Result: Passed With Flying Colors
=====
Process: 10
Question: 2 Number of times failed: 11
-----
Current Failed Question ID is 2 weight -> 0.3
Question With Maximum Weight is -> 3 Weight -> 0.4
-----
Old Weight Schema is:
Question ID: 1 Old Weight: 0.3
Question ID: 2 Old Weight: 0.3
Question ID: 3 Old Weight: 0.4
-----
Alternate Weight Schema is:
Question ID: 1 New Weight: 0.3
Question ID: 2 New Weight: 0.4
Question ID: 3 New Weight: 0.3
=====
zahid@zahid-HP-Pavilion-dv6500-Notebook-PC:~/Downloads/code$

```

Figure 4.23: Sample audit result for alternate strategy – Pass11

As shown in the highlighted part in Figure 4.23, the question 2 failed for the 11th time and an alternate strategy is provided for the current process. The weight for the current failed question is shown and the question with highest weight (retrieved from the master document) is also shown along with their respective question ID's. In the next part of the report, the old weight schema for the whole process is shown, and then the new weight schema is displayed. This new schema is however not implemented in our master document since there is currently no provision in our system to update the master

document. Also the master document is not to be updated without getting the necessary approval from the concerned authority on the new weight schema. Lastly, the purpose of the audit is to identify the failure points and recommend strategies to increase the efficiency. The implementation of the new strategy is dependent upon the system stakeholders. The csv input file and the detailed output for all the passes for this experiment are attached in Appendix A.

4.4 Summary Comments

In this Chapter we have presented and discussed a case study of an Infection Regime Control Survey and the application of our proposal Continuous Process Auditing framework and approach. Our investigations of this case study have demonstrated the consistency and effectiveness of the CPA to achieving the stated goals, namely, identification of process, or component, errors in results and behaviour, relative satisfaction or failure to achieve stated goals, and the ability to diagnose and recommend changes to improve the effectiveness of the underlying application.

Although our results are limited to a single case study, we note and emphasize that the nature of surveys, as probes for information, typically from human users of systems, or as reporting instruments, such as are used increasingly in healthcare and other contexts, is fairly well established [32,33]. Thus, the nature of our case study and the results achieved demonstrate effectively that our approach is generalizable to such process systems.

CHAPTER V

CONCLUSION AND FUTURE WORK

In this chapter we present the conclusions of our work and identify some areas for future work.

5.1 Conclusions

This thesis proposes a Continuous Process Auditing (CPA) framework that audits the processes in a complex system to increase the efficiency of the system. Based on this framework, a weight based model aided by communication mechanism is introduced to perform the audit analysis of processes in the underlying system. The framework generates audit reports to verify that our methodology catches the errors and misuse in a system and proposes alternate strategies to increase the system efficiency.

We demonstrated the applicability of the CPA framework and approach to a case study of an Infection Regime Control Survey, typical of survey applications used in modern healthcare and other contexts.

The verification results also show that our methodology is useful in auditing survey applications and can be applied to any survey application to increase its efficiency and achieve the required goal. The verification results also show that our framework

takes into account the steps required to differentiate the process audit from the procedure audit, fulfills the steps and constantly works towards achieving the system goal.

Our proposed framework is a generic framework and can be applied in any other domains which consist of processes working towards a common goal. Although our case study involved auditing a survey application, our framework can be applied straightforwardly to any other domain by extracting the process steps from the domain and getting the rules under which the processes would operate for the given domain. The steps from any given application can be used to populate the sticky logs and the underlying guidelines under which these process steps are done make up the master document.

Our framework is designed to solve a practical problem of auditing user responses in a survey; however there are still some shortcomings which limit its broad and rapid adoption. The first issue concerns design and handling of the master document which is currently hard coded into our system and has to be delivered by the system stakeholders from the onset. The downside of this is that stakeholders seldom know what they want and their requirements keep on changing with time. Having a master document that is hard coded into the system would require constant changes especially if the stakeholders are not domain experts.

Another limitation in our framework is the reliance on using csv files as input to the audit program. Although we took this approach due to the design of the current system producing the survey results, an approach where the data can be directly transferred to the audit program would be beneficial.

Lastly, our laboratory implementation and testbed does not address issues of a continuous audit program that runs constantly, as in a production environment. To achieve this would be beneficial although it would require more resources being tied down, thus increasing the cost.

5.2 Future Work

We address a few potential areas for future research work, based on the experience gained in this thesis research.

One vital area concerns the creation and usage of master documents. We have identified a list of predicates to be used to populate the master document. More research needs to be done that uses domain experts that combine their knowledge to come up with an exhaustive list of predicates that can be used irrespective of the domain. This would focus on creating a basic master document structure that is usable without the underlying domain details and can be populated as RDFS triplets by passing a list of subjects and objects for the specific domain.

Another aspect that can be improved in future regarding the master document is the implementation of the audit recommendation digitally. The recommendation currently provided have to go through a human who approves them and then has to go and manually update the master document and the introduction of a digital agent with some level of intelligence that can take a look at the recommendation and make changes (even if small) would be a challenge to look at in future.

Another area for future work would be to extend this framework in order to make it work with interoperable data gathered from different platforms and used to perform the audit analysis. Modern complex systems are distributed across large organizations that

rely on varying data formats, all working towards a common goal and an audit system that can extract and gather live data from the system and generate results on the fly, in real time.

Finally, the presentation of audit results is another aspect that should be examined further. This would require involving language and design experts in order to come up with a presentation format that is best suited for the stakeholders. It is also important to consider the realistic reporting requirements of specific users. For instance, nurses who must monitor medication dosages and schedules and duty orders may have a significantly different kind of report issued based on requirements of patient safety, than for hospital administrators looking at audit reports for cumulative events over a day, week, or month long time period of system auditing.

APPENDICES

APPENDIX A

Verification Test Results

CSV input file for Process Sticky Log Test

9,

14,

Test User,

2478019.0471064816,

1.1;

1;

3.1;

4.1;

4/11/2010;

;

;

10.1.1;

10.2.1;

- 10.3.1;
- 11.1.2;
- 11.2.3;
- 11.3.2;
- 12.1.1;
- 12.2.2;
- 12.3.1;
- 12.4.2;
- 12.5.1;
- 14.1.1;
- 14.3.2;
- 14.4.2;
- 15.1.1;
- 15.2.3;
- 15.3.1;
- 15.4.2;
- 15.5.2;

Detailed output of the audit program for Process Sticky Log Test

Response ID: 14

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

=====

Process Sticky Log ID: 10

Process Minimum Threshold Score: 0.7

Process Total Score: 1

Process Total Audit Value 0.1

Question Sticky Log ID: 1

Question Sticky Log Ans: 1

Question Sticky Log ID: 2

Question Sticky Log Ans: 1

Question Sticky Log ID: 3

Question Sticky Log Ans: 1

=====

=====

Process Sticky Log ID: 11

Process Minimum Threshold Score: 0.75

Process Total Score: 1

Process Total Audit Value 0.1

Question Sticky Log ID: 1

Question Sticky Log Ans: 2

Question Sticky Log ID: 2

Question Sticky Log Ans: 3

Question Sticky Log ID: 3

Question Sticky Log Ans: 2

=====

=====

Process Sticky Log ID: 12

Process Minimum Threshold Score: 0.8

Process Total Score: 1

Process Total Audit Value 0.2

Question Sticky Log ID: 1

Question Sticky Log Ans: 1

Question Sticky Log ID: 2

Question Sticky Log Ans: 2

Question Sticky Log ID: 3

Question Sticky Log Ans: 1

Question Sticky Log ID: 4

Question Sticky Log Ans: 2

Question Sticky Log ID: 5

Question Sticky Log Ans: 1

=====

=====

Process Sticky Log ID: 14

Process Minimum Threshold Score: 0.75

Process Total Score: 1

Process Total Audit Value 0.2

Question Sticky Log ID: 1

Question Sticky Log Ans: 1

Question Sticky Log ID: 3

Question Sticky Log Ans: 2

Question Sticky Log ID: 4

Question Sticky Log Ans: 2

=====

=====

Process Sticky Log ID: 15

Process Minimum Threshold Score: 0.8

Process Total Score: 1

Process Total Audit Value 0.2

Question Sticky Log ID: 1

Question Sticky Log Ans: 1

Question Sticky Log ID: 2

Question Sticky Log Ans: 3

Question Sticky Log ID: 3

Question Sticky Log Ans: 1

Question Sticky Log ID: 4

Question Sticky Log Ans: 2

Question Sticky Log ID: 5

Question Sticky Log Ans: 2

=====

Survey Valid

CSV input file for Master Process Test

9,
15,
Test User,
2568019.5471464216,
1.1;
1;
3.1;
4.1;
4/11/2010;
;
;
10.1.1;
10.2.1;
10.3.1;

- 11.1.2;
- 11.2.3;
- 11.3.2;
- 12.1.1;
- 12.2.2;
- 12.3.1;
- 12.4.2;
- 12.5.1;
- 14.1.1;
- 14.3.2;
- 14.4.2;
- 15.1.1;
- 15.2.3;
- 15.3.1;
- 15.4.2;
- 15.5.2;

Detailed output of the audit program for Master Process Test

Survey ID: 9

Response ID: 15

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

=====

Master Process ID :10

Number of Questions in Master Process: 3

Minimum Threshold Score for Master Process: 0.7

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.1

Master Question ID: 1

Master Question Weight: 0.3

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.3

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 2

Master Question Weight: 0.3

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.3

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 3

Master Question Weight: 0.4

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.4

Master Answer ID: 2

Master Answer Score: 0

=====

Master Process ID :11

Number of Questions in Master Process: 4

Minimum Threshold Score for Master Process: 0.75

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.1

Master Question ID: 1

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 2

Master Question Weight: 0.25

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.15

Master Question ID: 3

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 4

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

=====

Master Process ID :12

Number of Questions in Master Process: 5

Minimum Threshold Score for Master Process: 0.8

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.2

Master Question ID: 1

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 2

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 3

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 4

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 5

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

=====

Number of Questions in Master Process: 5

Minimum Threshold Score for Master Process: 0.8

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.2

Master Question ID: 1

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

Master Question ID: 2

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

Master Question ID: 3

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

Master Question ID: 4

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

Master Question ID: 5

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

=====

Master Process ID :14

Number of Questions in Master Process: 4

Minimum Threshold Score for Master Process: 0.75

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.2

Master Question ID: 1

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 2

Master Question Weight: 0.25

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.15

Master Question ID: 3

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 4

Master Question Weight: 0.25

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.25

Master Answer ID: 2

Master Answer Score: 0

=====
Master Process ID :15

Number of Questions in Master Process: 5

Minimum Threshold Score for Master Process: 0.8

Total Score for Master Process: 1

Total Audit Value for Master Process: 0.2

Master Question ID: 1

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 2

Master Question Weight: 0.2

Master Question Answer Options: 3

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Answer ID: 3

Master Answer Score: 0.1

Master Question ID: 3

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 4

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

Master Question ID: 5

Master Question Weight: 0.2

Master Question Answer Options: 2

Master Answer ID: 1

Master Answer Score: 0.2

Master Answer ID: 2

Master Answer Score: 0

CSV input file for Sample Audit result for passed Process audit and Question audit

9,

16,

Test User,

2997524.046234324561064816,

1.1;

1;

3.1;

4.1;

4/11/2010;

;

;

10.1.1;

10.2.1;

10.3.1;

11.1.1;

11.2.3;

11.3.1;

11.4.1;

12.1.1;

12.2.1;

12.3.1;

12.4.1;

12.5.2;

13.1.1;

13.2.1;

13.3.3;

13.4.1;

13.5.1;

14.1.1;

14.2.2;

14.3.2;

14.4.1;

15.1.1;

15.2.3;

15.3.1;

15.4.1;

15.5.1;

Detailed output of the audit program for passed Process audit and Question audit

Survey ID: 9

Response ID: 16

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid

=====

Process ID: 10

Process Resultant Score: 1

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0.3

Question Audit Result: Pass

Question ID: 3

Question Score: 0.4

Question Audit Result: Pass

Final Audit Score of Process 10: 0.1

Process Audit Result: Aced

=====

Process ID: 11

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.15

Question Audit Result: Fail

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.09

Process Audit Result: Passed With Flying Colors

=====

Process ID: 12

Process Resultant Score: 0.8

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0

Question Audit Result: Fail

Final Audit Score of Process 12: 0.16

Process Audit Result: Just Passed

=====

Process ID: 13

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.1

Question Audit Result: Fail

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 13: 0.18

Process Audit Result: Passed With Flying Colors

=====

Process ID: 14

Process Resultant Score: 0.5

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0

Question Audit Result: Fail

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.1

Process Audit Result: Failed

=====

Process ID: 15

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.1

Question Audit Result: Fail

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 15: 0.18

Process Audit Result: Passed With Flying Colors

=====

Overall Audit Score: 0.81

Overall Audit Result: Passed With Flying Colors

=====

CSV input file for failed Process audit and Question audit

9,

17,

Test User,

2797419.046234575871064816,

1.1;

1;

3.1;

4.1;

4/11/2010;

;

;

10.1.1;

10.2.1;

10.3.2;

11.1.1;

11.2.3;

11.3.1;

11.4.1;

12.1.1;

12.2.1;

12.3.1;

12.4.1;

12.5.2;

13.1.1;

13.2.1;

13.3.3;

13.4.3;

13.5.3;

14.1.1;

14.2.2;

14.3.2;

14.4.1;

15.1.1;

15.2.3;

15.3.1;

15.4.1;

15.5.1;

Detailed output of the audit program for failed Process audit and Question audit

Survey ID: 9

Response ID: 17

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid

=====

Process ID: 10

Process Resultant Score: 0.6

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0.3

Question Audit Result: Pass

Question ID: 3

Question Score: 0

Question Audit Result: Fail

Final Audit Score of Process 10: 0.06

Process Audit Result: Failed

=====

Process ID: 11

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.15

Question Audit Result: Fail

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.09

Process Audit Result: Passed With Flying Colors

=====

Process ID: 12

Process Resultant Score: 0.8

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0

Question Audit Result: Fail

Final Audit Score of Process 12: 0.16

Process Audit Result: Just Passed

=====

Process ID: 13

Process Resultant Score: 0.7

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.1

Question Audit Result: Fail

Question ID: 4

Question Score: 0.1

Question Audit Result: Fail

Question ID: 5

Question Score: 0.1

Question Audit Result: Fail

Final Audit Score of Process 13: 0.14

Process Audit Result: Failed

=====
Process ID: 14

Process Resultant Score: 0.5

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0

Question Audit Result: Fail

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.1

Process Audit Result: Failed

=====

Process ID: 15

Process Resultant Score: 0.7

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.1

Question Audit Result: Fail

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0

Question Audit Result: Fail

Final Audit Score of Process 15: 0.14

Process Audit Result: Failed

=====

Overall Audit Score: 0.69

Overall Audit Result: Failed

=====

CSV input file for Sample audit result identifying success factors

9,

29,

Test User,

2956419.143574575872348917,

1.1;

1;

3.1;

4.1;

4/11/2010;

;

;

10.1.1;

10.2.1;

10.3.1;

11.1.1;

11.2.3;

11.3.1;

11.4.1;

12.1.1;

12.2.1;

12.3.1;

12.4.1;

12.5.2;

13.1.1;

13.2.1;

13.3.3;

13.4.1;

13.5.1;

14.1.1;

14.2.2;

14.3.2;

14.4.1;

15.1.1;

15.2.3;

15.3.1;

15.4.1;

15.5.1;

Detailed output of the audit program for Sample audit result identifying success factors

Survey ID: 9

Response ID: 29

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid

=====

Process ID: 10

Process Resultant Score: 1

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0.3

Question Audit Result: Pass

Question ID: 3

Question Score: 0.4

Question Audit Result: Pass

Final Audit Score of Process 10: 0.1

Process Audit Result: Aced

Factors Affecting Success:

3

=====

Process ID: 11

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.15

Question Audit Result: Fail

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.09

Process Audit Result: Passed With Flying Colors

Factors Affecting Success:

2

1

=====
Process ID: 12

Process Resultant Score: 0.8

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0

Question Audit Result: Fail

Final Audit Score of Process 12: 0.16

Process Audit Result: Just Passed

Factors Affecting Success:

5

1

=====

Process ID: 13

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.1

Question Audit Result: Fail

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 13: 0.18

Process Audit Result: Passed With Flying Colors

Factors Affecting Success:

3

1

=====
Process ID: 14

Process Resultant Score: 0.5

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0

Question Audit Result: Fail

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.1

Process Audit Result: Failed

Factors Affecting Success:

2

3

1

=====

Process ID: 15

Process Resultant Score: 0.9

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.1

Question Audit Result: Fail

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 15: 0.18

Process Audit Result: Passed With Flying Colors

Factors Affecting Success:

2

1

=====

Overall Audit Score: 0.81

Overall Audit Result: Passed With Flying Colors

=====

CSV input file for Alternate Strategy test

9,

12,

Test User,

2797419.046234575871064816,

1.1;

1;

3.1;

4.1;

4/11/2010;

;

;

10.1.1;

10.2.2;

10.3.1;

11.1.1;

11.2.1;

11.3.1;

11.4.1;

12.1.1;

12.2.1;

12.3.1;

12.4.1;

12.5.1;

- 13.1.1;
- 13.2.1;
- 13.3.1;
- 13.4.1;
- 13.5.1;
- 14.1.1;
- 14.2.1;
- 14.3.1;
- 14.4.1;
- 15.1.1;
- 15.2.1;
- 15.3.1;
- 15.4.1;
- 15.5.1;

Detailed Output of the audit program for Alternate Strategy Test

Survey ID: 9

Response ID: 12

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid

=====

Process ID: 10

Process Resultant Score: 0.7

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0.4

Question Audit Result: Pass

Final Audit Score of Process 10: 0.07

Process Audit Result: Just Passed

=====

Process ID: 11

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.1

Process Audit Result: Aced
=====

Process ID: 12

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 12: 0.2

Process Audit Result: Aced

=====

Process ID: 13

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 13: 0.2

Process Audit Result: Aced

=====

Process ID: 14

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.2

Process Audit Result: Aced

=====

Process ID: 15

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 15: 0.2

Process Audit Result: Aced

=====

Overall Audit Score: 0.97

Overall Audit Result: Passed With Flying Colors

=====

Process: 10

Question: 2 Number of times failed: 1

=====

10th run

=====

Survey ID: 9

Response ID: 27

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid

=====

Process ID: 10

Process Resultant Score: 0.7

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0.4

Question Audit Result: Pass

Final Audit Score of Process 10: 0.07

Process Audit Result: Just Passed

=====

Process ID: 11

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.1

Process Audit Result: Aced

=====

Process ID: 12

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 12: 0.2

Process Audit Result: Aced

=====

Process ID: 13

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 13: 0.2

Process Audit Result: Aced

=====

Process ID: 14

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.2

Process Audit Result: Aced

=====

Process ID: 15

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 15: 0.2

Process Audit Result: Aced

=====

Overall Audit Score: 0.97

Overall Audit Result: Passed With Flying Colors

=====

Process: 10

Question: 2 Number of times failed: 10

=====
11th run
=====

Survey ID: 9

Response ID: 28

Username: Test

User Occupation: Staff

User ID: 1

User gender: Male

User Shift: Morning

Date: 4/11/2010

Survey Valid
=====

Process ID: 10

Process Resultant Score: 0.7

Process Minimum Threshold Score: 0.7

Question ID: 1

Question Score: 0.3

Question Audit Result: Pass

Question ID: 2

Question Score: 0

Question Audit Result: Fail

Question ID: 3

Question Score: 0.4

Question Audit Result: Pass

Final Audit Score of Process 10: 0.07

Process Audit Result: Just Passed

=====

Process ID: 11

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 11: 0.1

Process Audit Result: Aced

=====

Process ID: 12

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 12: 0.2

Process Audit Result: Aced

=====
Process ID: 13

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 13: 0.2

Process Audit Result: Aced

=====

Process ID: 14

Process Resultant Score: 1

Process Minimum Threshold Score: 0.75

Question ID: 1

Question Score: 0.25

Question Audit Result: Pass

Question ID: 2

Question Score: 0.25

Question Audit Result: Pass

Question ID: 3

Question Score: 0.25

Question Audit Result: Pass

Question ID: 4

Question Score: 0.25

Question Audit Result: Pass

Final Audit Score of Process 14: 0.2

Process Audit Result: Aced

=====

Process ID: 15

Process Resultant Score: 1

Process Minimum Threshold Score: 0.8

Question ID: 1

Question Score: 0.2

Question Audit Result: Pass

Question ID: 2

Question Score: 0.2

Question Audit Result: Pass

Question ID: 3

Question Score: 0.2

Question Audit Result: Pass

Question ID: 4

Question Score: 0.2

Question Audit Result: Pass

Question ID: 5

Question Score: 0.2

Question Audit Result: Pass

Final Audit Score of Process 15: 0.2

Process Audit Result: Aced

=====

Overall Audit Score: 0.97

Overall Audit Result: Passed With Flying Colors

=====

Process: 10

Question: 2 Number of times failed: 11

Current Failed Question ID is 2 weight -> 0.3

Question With Maximum Weight is -> 3 Weight -> 0.4

Old Weight Schema is:

Question ID: 1 Old Weight: 0.3

Question ID: 2 Old Weight: 0.3

Question ID: 3 Old Weight: 0.4

Alternate Weight Schema is:

Question ID: 1 New Weight: 0.3

Question ID: 2 New Weight: 0.4

Question ID: 3 New Weight: 0.3

=====

END

BIBLIOGRAPY

- [1] Available from: www.dictionary.com
- [2] <http://iamsam.hubpages.com/hub/Definition-of-Auditing>
- [3] http://en.wikipedia.org/wiki/Sarbanes%E2%80%93Oxley_Act
- [4] VASARHELYI M. A., HALPER F. B. 1991. The continuous audit of online systems. *Auditing: A Journal of Practice & Theory*, 10 (1), 110-125.
- [5] WOODROOF J., SEARCY D.W. 2001. Continuous audit: model development and implementation within a debt covenant compliance domain. *International Journal of Accounting Information Systems* Volume 3 (2), 169-191.
- [6] REZAEI Z., SHARBATOGHLIE A., ELAM R., MCMICKLE P.L. 2002. Continuous auditing: building automated auditing capability. *Auditing: A Journal of Practice & Theory*, 21 (1), 147-163.
- [7] ONIONS R.L. 2003. Towards a paradigm for continuous auditing. Available from: <http://www.auditsoftware.net/community/how/run/tools/Towards%20a%20Paradigm%20for%20continuous%20Auditing1.doc>.
- [8] HASAN, STILLER, B. 2005. A Generic Model and Architecture for Automated Auditing. *Lecture Notes in Computer Science*, Volume 3775/2005, 121-132.

- [9] MURTHY U. S., GROOMER S. M. 2004. A continuous Auditing Web Services Model for XML-based Accounting Systems. *International Journal of Accounting Information Systems*, Volume 5 (2), 139-163.
- [10] HUANZHUO YE, YUNING HE. 2008. A continuous auditing model based on web services. In *Proceedings of the 7th WSEAS International Conference on Applied Computer & Applied Computational Science (ACACOS '08)*, April 6-8.
- [11] RUEY-SHUN CHEN, CHIA-MING SUN. 2007. A Collaborative Continuous Auditing Model under Service-Oriented Architecture Environments. In *Proceedings of the 6th WSEAS International Conference on E-ACTIVITIES*, Tenerife, Spain, 47-52.
- [12] ZUOMING HUANG, QIAO-LING LIU, JI-BING HU. 2011. Web-Service Oriented Computer Audit System Based on Agents. In *Proceedings of the 2011 International Conference on Management and Service Science (MASS)*, 2011, 1-4.
- [13] HUANZHUO YE, SHUAI CHEN, FANG GAO. 2008. On application of SOA to continuous auditing. *WSEAS Transactions on Computers*, 7 (5), 532-541.
- [14] CHARLES LING-YU CHOU, TIMON DU, VINCENT S.LAI.2007. Continuous auditing with a multi-agent system. *Decision Support Systems*, Volume 42 (4), 2274-2292.
- [15] CHIEN-HO WU, YUEHJEN E. SHAO, BIH-YIH HO, TSAIR-YUAN CHANG. 2008. On an Agent-based Architecture for Collaborative Continuous Auditing. In *proceeding of the 12th International Conference on Computer Supported Cooperative Work in Design*, 355-360.

- [16] GROOMER S. M., MURTHY U. S. 1989. Continuous auditing of database applications: An embedded audit module approach. *Journal of Information Systems* 3 (2), 53–69.
- [17] Available from http://en.wikipedia.org/wiki/Integrated_test_facility
- [18] KUHN J.R., SUTTON S.G. 2010. Continuous Auditing in ERP System Environments: The Current State and Future Directions. *Journal of Information Systems*, Volume 24 (1), 91-112.
- [19] Global Audit Information Network (benchmark): Available from <http://www.theiia.org/guidance/benchmarking/>
- [20] Available from www.acl.com
- [21] VASARHELYI M. A., ALLES M., KOGAN A. 2004. Principles of analytic monitoring for continuous assurance. *Journal of Emerging Technologies in Accounting*, Volume 1 (1), 1-21.
- [22] ALLES M., KOGAN A., VASARHELYI M. A. 2002. Feasibility and economics of continuous assurance. *Auditing: A Journal of Practice & Theory*, Volume 21 (1), 125–138.
- [23] Available from http://en.wikipedia.org/wiki/ISO_9000#2000_version
- [24] Available from http://transition-support.com/Process-Auditing_Technique.htm
- [25] GRAY J., REUTER A. 1993. *Transaction processing: Concepts and techniques*. Morgan Kaufmann, 1993.
- [26] RINGELSTEIN C., STAAB S. 2010. DIALOG: A Distributed Model for Capturing Provenance and Auditing Information. *International Journal of Web Services Research*, Volume 7 (2), 1-20.

- [27] ALLEMANGD., HENDLER J. 2008. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann, 2008.
- [28] Available from <http://www.w3.org/XML/>
- [29] Available from <http://tools.ietf.org/html/rfc4180>
- [30] PAPAZOGLU M.P. 2003. Service Oriented Computing: Concepts, Characteristics and Directions. In Proceedings of the 4th International Conference on Web Information Systems Engineering, (WISE'03), 3-12.
- [31] Available from http://en.wikipedia.org/wiki/Rule-based_system
- [32] KENT R.D., KOBTI Z., SNOWDON A.W., AGGARWAL A. 2010. Towards a Unified Data Management and Decision Support System for Health Care. Intelligent Interactive Multimedia Systems and Services, Volume 6, 205-220.
- [33] KOBTI Z., SNOWDON A.W., KENT R.D., BHANDARI G, RAHAMAN S.F., PRENEY P.D., KOLGA C.A.; TIESSEN B., ZHU L. 2011. Towards a “Just-in-Time” Distributed Decision Support System in Health Care Research. Annals of Information Systems: Supporting Real Time Decision-Making, Volume 13 (3), 253-285.
- [34] Available from <http://www.speedwell.co.uk/page/145/healthcare-surveys.htm>
- [35] Available from [http://en.wikipedia.org/wiki/Sequence_container_\(C%2B%2B\)](http://en.wikipedia.org/wiki/Sequence_container_(C%2B%2B))

VITA AUCTORIS

Atif Hasan Zahid was born in 1979 in Peshawar, Pakistan. He received his B.Sc. (honours) in Computer Science from the University of Windsor in 2006. He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in Winter 2012.