An Audit Model for ISO 9001 Traceability Requirements in Agile-XP Environments

Malik Qasaimeh
King Hussein Faculty of Computing Sciences
Princess Sumaya University for Technology
Al-Jubaiha, Amman 11941, P.O. Box 1438, Jordan
Email: m.qasaimeh@psut.edu.jo

Alain Abran
École de Technologie Supérieure
University of Québec, 1100 Notre–Dame Ouest,
Montréal, Québec H3W 1T8, Canada
Email: alain.abran@etsmtl.ca

Abstract— Software organizations that develop their software products using the agile software processes such as Extreme Programming (XP) face a number of challenges in their effort to demonstrate that their process activities conform to ISO 9001 requirements, a major one being product traceability: software organizations must provide evidence of ISO 9001 conformity, and they need to develop their own procedures, tools, and methodologies to do so.
This paper proposes an auditing model for ISO 9001 traceability requirements that is applicable in agile (XP) environments. The design of our model is based on evaluation theory, and includes the use of several auditing "yardsticks" derived from the principles of engineering design, the SWEBOK Guide, and the CMMI-DEV guidelines for requirement management and traceability for each yardstick. Finally, five approaches for agile-XP traceability approaches are audited based on the proposed audit model.

Index Terms— Agile Software Certification, Extreme Programming, Software Process Improvement, ISO 9001

I. Introduction

The origins of ISO 9001 can be traced back to the manufacturing sector; however this quality standard is now being applied to many other types of organizations, even health care. At the same time, the development of software has become an important endeavor in ISO member countries, and so the ISO has developed and released a set of software engineering guidelines to serve as a roadmap to enable software development organizations to become ISO 9001certified. These guidelines are contained in the ISO 90003 publication, and those organizations that need to be ISO 9001 certified can use it when audited to show evidence that they have implemented the ISO 9001 requirements. The ISO 9001 certification requirements are not technology related but are business requirements: there is ample evidence that many small organizations have achieved ISO 9001 certification and reap business benefits from such certification [1-3].

This paper addresses this specific business issue of requirements for ISO 9001 in contexts where the software teams develop software products in an agile mode: more specifically, this paper looks at what agile teams must put in place to meet this business need.

The literature reports as well that some authors have initiated research work to address this business need. For instance, Vitoria in [4] has studied the ISO 9001 and TickIT standard and analyzed how it has been used in two case studies with agile projects. Vitoria reports for these two projects that 33% of ISO 9001 requirements could not be applied in an agile-XP project, 24% could be partially applied, 20% could be applied in full, while 23% were not relevant to the scope of the projects.

Vriens [5] has discussed CMM, ISO 9001 and their relationships to agile-XP and Scrum: he observes that most of the ISO 9001 requirements are independent of development methods used. This author reports on his experience of getting certified for both CMM Level 2 and ISO 9001:2000 on a time scale of 2 years by using agile methodologies.

Wright [6] describes a successful certification evidence for an agile-XP organization. This author describes how the organization managed the large team through the practice of agile-XP and highlights the tools used to support the project team to handle the ISO 9001 requirements: this author's focus is only on some selected ISO 9001 requirements and he highlights their corresponding XP support activities.

Extreme programming (agile-XP) has been selected in the research reported here for improvement as a candidate agile process. This selection was based on the literature indicating a higher adoption of agile-XP over other agile software processes. The higher adoption of agile-XP over other agile software processes can be supported by the literature with different case studies from both academia and industry, such as [7, 8].

The traceability of the user requirements during the development process is among the important auditing

doi:10.4304/jsw.8.7.1556-1567

challenges reported in the agile literature [9-11]. Software traceability is defined in ISO 12207:2008 as "the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another." Ramesh in [12] defines requirement traceability as "a characteristic of a system in which the requirements are clearly linked to their sources and to the artifacts created during the system development life cycle based on these requirements." In agile development, verifying that the requirements have been implemented, designed, and tested in the final product depends mainly on lightweight artifacts, such as test cases and user accepted tests, without documented evidence on how these requirements have been traced through the project life cycle. This creates challenges for software auditors, in terms of ensuring that the processes are in conformity with a specific standard, such as ISO 9001. For example, according to [13] a manager cannot track progress in agile projects in the same way as in plan-driven projects, where a manager simply asks whether or not the necessary documents have been produced.

Software development-related documents constitute valuable audit evidence for Information Systems (IS) auditors. However, this is not the only type of evidence that can be obtained by the auditors: the IT Standards, Guidelines, and Tools, and the Techniques for Audit and Assurance and Control Professionals [14] point out that other audit evidence types are also important, such as observed processes and the existence of physical items, activity and control logs, and system flowcharts. In addition, analysis of the information through comparisons, simulations, calculations, and reasoning can also be used as audit evidence.

Developers in agile environment can adopt agile modeling (AM) for the modeling and documentation for the software development processes: agile modeling (AM) is a collection of practices, guided by values and principles for application in a day to day basis. AM include practices such as: active stakeholder participation, group work to create suitable models, verification, iterative modeling, parallel model creation, application of standards and documentation improvement. Agile modeling has some common values with existing agile processes, such as XP and SCRUM, like communication with team members, simplicity, and feedback. Agile modeling puts an emphasis on humility, briefly defined as the openness for different ideas and perspectives. Ambler in [15] mentions that " What makes AM a catalyst for improvement aren't the modeling techniques themselves—such as use case models, class models, data models, or user interface models—but how to apply them". However, agile modeling does not come with detailed procedures on how to create a software documentation process; rather, agile modeling is closer to an overall high level understanding of the whole system. This will of course provides the software development team with facilities to create modeling artifacts to their agile process but will provide less evidences for IS auditors to identify the auditing evidences necessary to assess the conformity of the agile process to a specific standard such as ISO 9001.

An analysis of several auditing standard and guideline documents, such as ISACA and the International Standard of Auditing [16] reveals that the term evaluation has been considered as an integral part of the auditing process. Although no clear definition of the term has been found in either document, it has been noted that both refer to static or dynamic analysis, review, and/or observation of the organization's business processes, internal control methods, and software processes as evaluation activities. According to [17], other synonyms for can be found in the literature, such as analysis, appraisal, audit, review, and examination (because evaluation is an activity that causes anxiety in most people). This indicates that evaluation and audit are closely related terms: the connection between the evaluation theory and the objectives of the paper will be discussed in the methodology section.

This paper proposes a design of an auditing model for agile software processes (e.g. XP) based on evaluation theory, which can provide IS auditors with a methodological approach to the auditing process. The motivation for this work is to help auditors obtain evidence in conformity with ISO 9001. The proposed model is aimed at providing evidence of process traceability based on the observation of techniques and mechanisms intended to implement the traceability requirements. Our model is designed from an engineering perspective: it is based on evaluation theory and on the investigation of the principles of engineering design [18-20]. Several best practice software engineering models such as CMMI-DEV and SWEBOK will be used to design the traceability auditing yardsticks.

This paper is organized as follows. Section 2 presents an overview of auditing practices in software organizations. Section 3 presents an analysis of traceability requirements in ISO 9001 and their potential advantages in software organizations. Section 4 presents the methodology and reviews the evaluation theory. Section 5 presents the formulation of the auditing criteria and the yardsticks. Section 6 presents a case study for each of five agile-XP traceability approaches, and discusses the auditing evidence collected for software process traceability. Finally, section 7 presents the conclusion of the paper.

II. Overview of Auditing Practices

Auditing is a systematic and independent examination for determining whether or not an organization's activities (i.e. business processes) are in conformity with the requirements of a specific standard or set of rules, and whether or not those activities have been effectively implemented and are suitable for achieving their predefined objectives [21]. The activities may be carried out at various levels, such as: organization, system, process, project, or product. This paper focuses on ISO 9001 auditing activities conducted at the software process level.

The above generic definition of auditing embodies several important points:

- It is a systematic examination, which means that it is carried out in a methodical manner. It is also a planned activity performed systemically on the organization's activities.
- It is an independent process, in that auditors collect and evaluate evidence, and the results, based on their findings, are unaffected by the client or the organization. The role of an auditor in this case is similar to that of a judge who collects and evaluates evidence based on the law.
- It is conducted to evaluate whether or not an organization's internal controls are performing as they are supposed to. The primary objective of the auditing process is to establish whether or not these internal controls have been implemented effectively.
- Auditors examine, analyze, and judge the internal controls to determine whether or not they are suitable for achieving their predefined objectives.

Auditors begin by extracting from ISO 9001 the specific information that will be considered later as the basis for the auditing process. This basis corresponds to the set of recognized best practices that the organization should implement in order to comply with ISO 9001 requirements. The evidence is a set of facts that objectively confirms how those best practices have been implemented and to what extent they have achieved their objective. The results of comparing the audit basis to the evidence are called observations. Those observations should be subjected to several analysis cycles before they are summarized into what are called findings – see Figure 1.

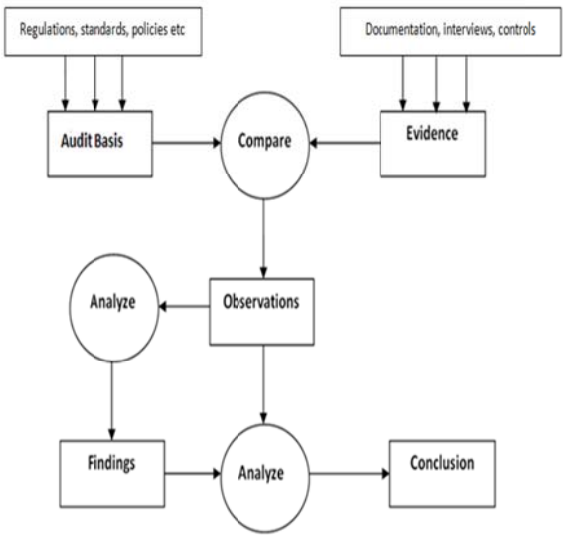

Figure 1: Generic auditing model [12].

The difference between an observation and a finding is that an observation consists of raw data which need exhaustive examination and analysis before they are useful to stakeholders, governments, or senior management. A finding is the result of investigating observations: it is the most important piece of information, and constitutes the final result of the auditing process. Finally, an audit conclusion is prepared and reported to all interested parties.

The term audit came into general use after World War II, when the military issued a standard and specifications for developing complex products and systems. The term auditing was introduced to refer to a set of inspection activities conducted in large manufacturing companies (in the electronics industry, for example) and in high risk manufacturing sectors (in the nuclear, food, and pharmaceutical industries, for example) [22].

In 1970, the United States Government Accountability Office (GAO) indicated that auditing in federal agencies needed to be conducted in a more comprehensive manner. Moreover, the GAO was advocating entirely different auditing practices, addressing companies and organizations from various perspectives. For example, according to the GAO, auditing practices should not be limited to the review or examination of financial statements by accountants, and should include investigation of:

- The organization's level of compliance with laws and regulations;
- The efficiency of all the activities conducted within the organization;
- The effectiveness of the activities in achieving their objectives.

In 1980, special standards and new laws were created to ensure more frequent and better auditing practices to cover all organizational sectors and their related activities. Later, in 1990, the amount of federal government auditing increased, and new laws and regulations were mandated to focus on additional issues, including performance, management, compliance, and the effectiveness of the auditing activities themselves. As a result, federal government audit practices have become a key element in meeting the government's responsibilities and providing a degree of confidence that is understood by all parties.

Recently, auditors have begun to scrutinize business process controls to determine the level of adherence of organizations to industrial standards and federal laws. The premise is that, although a financial statement audit is important, it provides incomplete information, since software systems can also affect the organization's business processes. IT auditing should therefore be initiated that covers all aspects of IT practices, with a view to examining the organization in terms of its adherence to industrial standards and federal laws [23].

IT auditing should not be confused with financial auditing, even though there may be some overlaps in the work of the two groups of auditors. IT auditing provides an examination of computers, databases, and software systems. It is a professional discipline involving several

different techniques for independently reviewing IT processes (e.g. software processes), as well as IT applications (e.g. financial records databases).

III. Analysis of Traceability Requirements in ISO 9001

ISO 9001 is a quality management standard that identifies a set of requirements designed to ensure consistency and proficiency in terms of the activities, techniques, and methods used in the organization. As a result, it provides a set of requirements for the process of gathering customer needs and for creating a product that achieves customer satisfaction.

In non software organizations, such as pressure vessel manufacturers, for example, it is common for a particular material to be monitored throughout all the manufacturing stages, and for the changes it undergoes to be recorded. In this way, the final component can be traced back to the original material. For ISO 9001, the material must be uniquely identified and the changes recorded to show evidence of traceability1.

For software systems, traceability of the software process is a major requirement that has been described in ISO 9001 and in ISO 90003 in clause 7.5. Even though ISO 90003 does not elaborate on the techniques for achieving the traceability of a software process, nor does it recommend a specific method for doing so, the ISO 90003 guidelines for the application of ISO 9001 for software state that traceability is usually implemented through configuration management: “Throughout the product life cycle, there should be a process to trace the components of a software item or product, and this process may vary in scope, according to contract or marketplace requirements, from being able to place a certain change request in a specific release, to recording the destination and usage of each variant of the product.”

The reasons for implementing traceability analysis are not discussed in either ISO 9001 or in the guidelines document. However, we know that the advantages of doing so for a quality management system are the following:

A. Support for Change Management

Software projects are subject to dynamic changes at the technical level, such as changing software project requirements or replacing development tools, or at the managerial level, such as changing the development schedule or making changes because of budget constraints. According to [15], for larger and more complex software projects, change management practices are challenging without a traceability mechanism in place, because, at some point, the increasing number of people involved in the project and its growing size will significantly aggravate the communication difficulties between project management and developers.

The process of change management should be formalized, so that every change request follows a sequence of activities, starting with the initiation of a request for a change (assignment of a number to the change process and acceptance of the change by the team manager) and ending with the implementation and testing of the change request. Kowalczykiewicz in [24] maintains that the change management process should be supported with tracking techniques, so that every change request can be tracked throughout the project life cycle. From a development team point of view, the traceability mechanism will allow the team to keep the development baseline updated., because every requested change will be handled individually, and all the related artifacts that have been affected by the change request will be updated at the same time; for example, for instance, when a change has been made to improve a module N, then developers should ensure that all the related artifacts that have a relationship with module N are modified if appropriate, including a modification to the associated test cases and to the requirements related to module N.

From the ISO 9001 point view, support of traceability at the project level implies support of software maintainability, because project and maintenance teams will easily understand the relationships and dependencies between the project components and artifacts, and they will have the opportunity to more effectively modify the software system based on updated customer requirements.

B. Cost Management

Software traceability techniques can support the software development team in their efforts for change impact analysis [25]. In this context the developers need first to analyze and translate the change request into software terms and then to identify the potential links between requirements, specifications, design elements, and tests. These links can be analyzed to determine the scope of an initiating change. The objectives of change impact analysis are [19]:

- Determination of the scope of the change, in order to plan and implement work.
- Development of accurate estimates of the resources needed to perform the work.
- Analysis of the costs/benefits of the requested change.
- Communication to others of the complexity of the change.

A quality management system requires project managers to perform an impact analysis when a change is requested by the customer. The impact analysis statement will help the development team estimate the budget needed to implement the change request before beginning the change process. The statement will be analyzed by both the project manager and the customer. According to [19], the software change request is impacted by many factors, such as:

- Application type;
- Novelty of the software;
- Software maintenance staff availability;
- life span of the software;
- Hardware characteristics;
- Quality of the software design, construction, the documentation, and testing.

The SWEBOK Guide [19] also point out that the software development team should have knowledge of the structure and content of the software system before they begin implementing the requested change. They gain this knowledge by identifying all the systems and software products affected by a software change request, and estimating the resources needed to accomplish the change. This initial knowledge will be enhanced by the availability of traceability mechanisms that will enable developers and software managers to better estimate the cost of changing the content of the system. It will also make it easier to determine the risk associated with implementing the change.

E. Process Improvements

Organizations are complex systems with processes that run concurrently and interact. Improving those processes requires discipline on the part of organizations and a defined reference model to systematically consider their process and project management strategies, as shown in Table I.

The focus of ISO 9001:2008 is on process quality improvement, and a set of requirements and guidelines (in ISO 90003) is defined to help organizations set up their improvement program goals in alignment with their business objectives. Table 1 set out the improvement areas in ISO 9001 at both the process and project levels, and their corresponding CMMI key process areas (KPAs).

TABLE I:
ISO 9001 OBLIGATIONS AND CMMI KPAS CORRESPONDING TO PROCESS AND PROJECT IMPROVEMENT AREAS

ISO 9001 and ISO 90003 obligations at the process and project levels	
• Organizational process planning • Defined team responsibilities, authority, and communication procedures • Project resource management • Product realization planning • Production and service provision • Process control and monitoring • Project measurement and data analysis for improvement purposes	
CMMI Process management KPAs	**CMMI Project management KPAs**
• Organizational Process Focus • Organizational Process Definition • Organizational Training • Organizational Process Performance • Organizational Innovation and Deployment	• Project Planning • Project Monitoring and Control • Supplier Agreement Management • Integrated Project Management • Risk Management • Integrated Teaming • Integrated Supplier Management • Quantitative Project Management

In terms of the relationships between software process improvement and traceability techniques, the SWEBOK Guide [19] points out that the tools and techniques intended to manage the tracking of software documentation and that of software releases can also contribute to improving software process. Briefly stated, traceability for process improvement can:

- Positively impact the communication procedures shared by the process improvement team members, and improve the availability of the software project status throughout all the development phases.
- Facilitate tracking of the sources and causes of defects arising during the software process life cycle, and help address them in a timely manner.
- Help to quickly determine the requirements affected by potential changes to the source code and to any associated test cases.

IV. METHODOLOGY

In this section, we present our design for an audit model for software process traceability, focusing on ISO 9001 and the agile software processes. The methodology for this design is based on the work of [26]: An Evaluation Theory Perspective of the Architecture Tradeoff Analysis Method – ATAM. The use of evaluation theory in the domain of software engineering has been investigated by [27] and [28], with a view to helping software engineering researchers develop their evaluation criteria, procedures, and conclusions. We have used those concepts in this paper for developing our auditing model for ISO 9001 traceability requirements.

A. Evaluation Fundamentals

To design an evaluation procedure, the researcher should consider the components proposed in [26] and presented in Figure 2. We use these components to design an audit model to evaluate ISO 9001 traceability and to select a case study that demonstrates the applicability of our audit model – see Figure 2.

The components of an evaluation procedure are highly interrelated with the target, and the delimitation of the target is the first evaluation component that could impact the selection of the evaluation method. López in [16] has classified the evaluation methods into objective-oriented evaluation, management-oriented evaluation, consumer-oriented evaluation, expertise-oriented evaluation, adversary-oriented evaluation, and participant-oriented evaluation.

The design of our audit model considers the steps of an evaluation procedure as described by [26]:

- Target: the object under evaluation;
- Criteria: the characteristics of the target that are to be evaluated;
- Yardstick: the ideal target against which the real target is to be compared;
- Data gathering techniques: the techniques needed to assess each criterion under analysis;
- Synthesis techniques: the techniques used to organize and synthesize the information obtained with the assessment techniques, the results of which are compared to the yardstick.

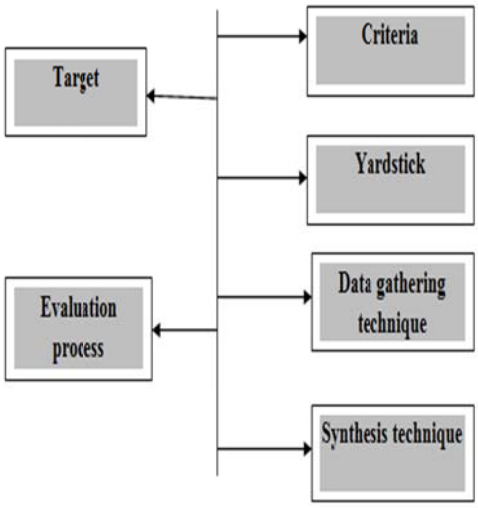

Figure 2: Components of an evaluation procedure [16]

Evaluation process: A series of activities and tasks by means of which an evaluation is performed.

For our purposes, the design of an audit model can be considered as a type of hybrid approach that combines the principles of management-oriented evaluation and adversary-oriented evaluation [26], because it is aimed at providing useful information to aid in decision making and at providing a balanced examination of all sides of controversial issues.

Once the target is known and delimited, its characteristics must be identified for evaluation purposes [26,28]. All the characteristics and their ideal values, which indicate the nature of the target under ideal conditions, make up what is known as the yardstick or standard.

Data about the real target should be obtained using particular data gathering techniques, and assigning a value (data, information set, numerical, etc.) to each criterion. The data, once collected, are organized into an appropriate structure and compared against the yardstick by applying synthesis techniques. This comparison yields the results of the evaluation. Finally, all the above components above are linked through the evaluation process [26].

Figure 3 presents the main process for designing an audit model for IS0 9001 traceability requirements based on the evaluation described in [26].

V. Design of the Auditing Model

A. Scope Delimitation

For agile software processes (e.g. agile-XP), implementing a traceability technique can help software developers and project managers in tracking the status of the software project and responding efficiently to change requests. The objective of this paper is to design an auditing model for traceability requirements using evaluation theory. ISO 9001 is the main target standard for deriving the auditing model. The process for designing this auditing model takes as its inputs the guidelines of CMMI and the SWEBOK, as well as Vincenti's engineering design concepts for identifying audit criteria.

The aim of the traceability auditing model is to help ISO 9001 software auditors to audit the agile software processes for traceability requirements. It can also be useful for auditing traditional software processes.

B. Design of the Audit Criteria and Yardsticks

As stated by [26], criteria can be elicited either using an obligatory standard that implicitly contains the criteria to be applied in the evaluation, or, if no such standard has been defined, the auditors should refer to any relevant study of targets, relevant standards, or ideals that might be relevant to the target in question. In our work here, the obligatory standard is ISO 9001.

The development of an audit model for agile process traceability could not be achieved without support from other relevant software engineering standards, such as CMMI and the engineering design concepts in [18]. The structure of the proposed auditing model is presented in Figure 4.

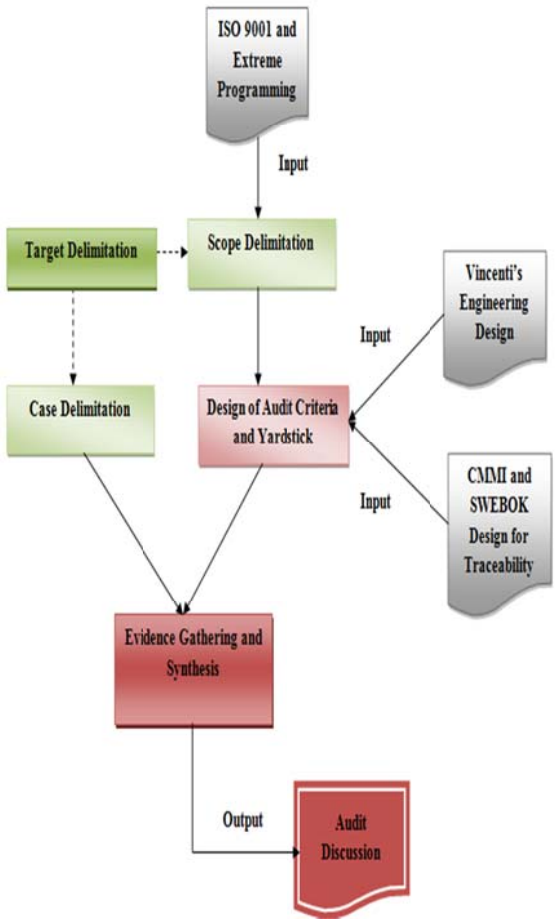

Figure 3: Design methodology for the audit model

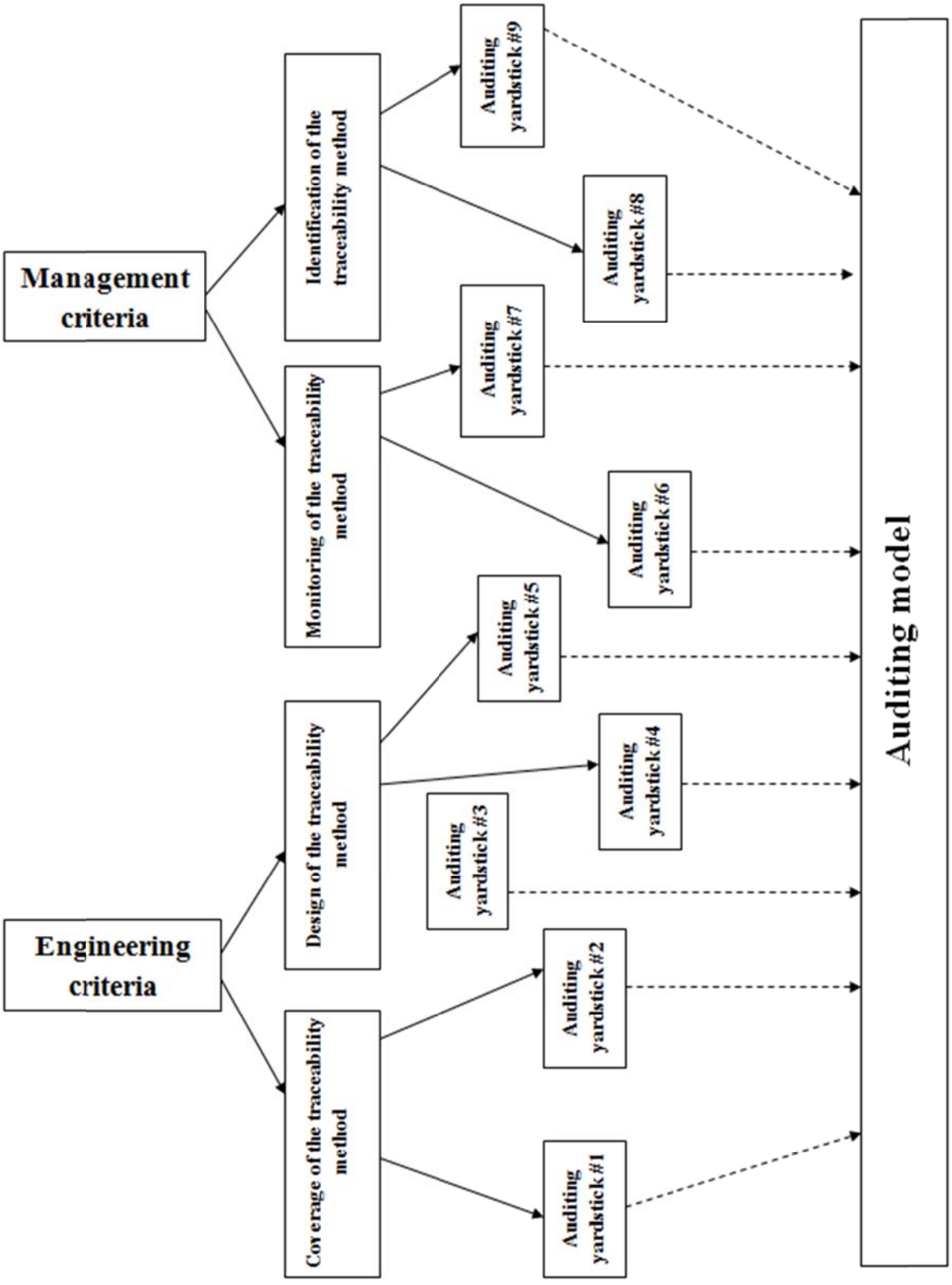

Figure 4: The structure of the proposed auditing model.

✓ **Engineering criteria**

The list of the audit criteria presented next is based on the concepts of theoretical tools and the operational principles of engineering in [18,20,28].

- **Design of the traceability method for agile**

The main objective of an agile software traceability method is to provide the software developers and project managers with a tool that supports their development tasks. Vincenti's classifications of theoretical engineering tools have enabled us to see what kinds of engineering tools have been used in the design of traceability methods. In [18], these tools are used by engineers to help them with the design process. They include intellectual concepts for thinking about designs, as well as mathematical methods, theories, and formulas, which can be simple or complex, for performing design calculations. The following are the audit yardsticks for these criteria:

Yardstick #1

Intellectual concepts, which represent the design ideas people have in mind, are expressed in natural language. These concepts are subject to the qualitative reasoning of engineers, before quantitative analysis and design calculations are performed.

Yardstick #2

Mathematical models, which are useful for quantitative analysis and design, can be either simple or complex. This scientific knowledge must be reformulated to make it applicable to providing engineering knowledge about the design.

- **Coverage of the traceability method**

The set of operational principles underlying an engineering design is classified as a fundamental design concept in [18]. These principles define the essential fundamental concept of a device (in this context, a traceability method) and provide a high-level description of the design objectives, either of the whole design or of each design component. Thus, designers provide either a complete engineering design for the problem in the domain, or a design component that partially addresses the problem in the domain based on the objectives of the operational principles.

The following are the audit yardsticks for this criterion:

Yardstick #3

Full operational principles: The engineering design of a traceability method considers different life cycle iterations, such as requirement specifications, architecture, detailed design, source code, and testing phases.

Yardstick #4:

Partial operational principles: The engineering design of the traceability method focuses on the relationships between entities developed in the same iteration of the process life cycle; for example, the artifacts produced during the requirements elicitation process (e.g. the user stories in agile-XP).

✓ **Management Criteria**

In both CMMI and the SWEBOK Guide, traceability management activities are described as a part of the configuration management process area. The SWEBOK Guide describes configuration management as a software engineering knowledge area focused on systematically controlling changes to the configuration, and on maintaining the integrity and traceability of the configuration throughout the system life cycle. The viewpoint of a configuration management system in the SWEBOK Guide is not limited to a software product, but rather covers the functional and/or physical characteristics of hardware, firmware, or software.

CMMI describes configuration management as a supporting process at maturity level 2, which focuses on identification, control, status reporting, and auditing for the traceability items. These items are intended to describe any artifact produced during the software life cycle, such as requirements specifications, architectural design, source code, test cases, and so on.

The audit criteria presented below are based on the concepts of configuration management described in the SWEBOK Guide and CMMI.

- **Identification of the traceability method**

In the SWEBOK Guide, identification of a software traceability item is considered a fundamental step in the construction of a software system that can be controlled and traced during the software process life cycle. At the same time, both the SWEBOK Guide and CMMI stress the importance of assigning unique identifiers to traceability items and developing a strategy for labeling software items and describing their relationships.

The following are the audit yardsticks for this criterion:

Yardstick #5:

Traceability item identification: The traceability method should consider the related traceability identification activities, which include mechanisms for identifying and labeling the traceability items and/or establishing identification schemes that automatically assign unique identifiers to each traceability item.

Yardstick #6:

Traceability item relationships: The proposed schemas for the identification of the relationships and dependencies between the traceability items are considered within a specific development phase or within the entire software life cycle.

Yardstick #7:

Traceability role identification: The traceability method assigns privileges to the software project stakeholders to access or modify the software items in the project baseline or to monitor the status of the software project according to their role in the project. The aim is to comply with the best practices for building a traceability management system in CMMI, and identifying the owner responsible for each traceability item is one of those practices.

- **Monitoring of the traceability method**

Status monitoring and updating of the software project is a requirement for designing a software life cycle traceability mechanism. As discussed in section 3, it helps software developers and project managers

determine the status of the software project and gauge the impact of changes to the cost, resources, and duration of the project.

The following are the audit yardsticks for this criterion:

Yardstick #8:

Traceability documentation: The information produced during the software life cycle to support the traceability method is reported. The documentation in this case is different from that produced during the software process life cycle, such as software requirements or test cases.

The traceability method produces the required documentation, which covers the entire software life cycle and provides project stakeholders with useful information regarding project status. This information can take the form of ad hoc queries to answer specific questions, or the periodic production of design reports. Examples of such documentation are traceability logs, the history of traceability items, and the relationship of traceability items, and so on.

Yardstick #9:

Documentation access: Traceability documentation and items should be stored in repositories in such a way that software stakeholders are able to access and retrieve them at any stage of the development process. The storage and retrieval mechanisms are evaluated, and the right of access that has been granted based on the role of the traceability stakeholders to assess them is monitored.

VI. Case Study

A. Context and Scope

To study the applicability of the auditing model proposed, we have selected five case studies which are compatible with the scope defined in section 5.1. The selection of these case studies is based on the following steps and criteria – see Table II:

- A search was conducted of IEEE Xplore and ScienceDirect-Elsevier for any paper proposing a methodology, technique, or framework to enhance the traceability of an agile software process.
- The following terms were used in browsing the content of IEEE Xplore, ScienceDirect-Elsevier:
  - ✓ Agile AND traceability
  - ✓ XP AND traceability
  - ✓ Agile AND configuration management
  - ✓ XP AND configuration management
- A title and abstract analysis was performed to select the papers that discuss XP traceability. Note that some authors discuss the principles of XP, but refer to them as agile principles (i.e. they do not specify which agile process they are improving, and selected XP as a candidate process without referring to it by name).
- All the papers that discuss agile traceability in general, without proposing a methodology, technique, or framework for managing traceability, were discarded.
- As some authors discuss the same proposed traceability approach in a number of different research articles, only their most recently published article was selected.

XP was proposed by Kent Beck in 1999, and very few papers discuss agile software process traceability or XP traceability prior to 2003. As a result, the papers we identify were published between 2003 and 2011.

TABLE II
SELECTED CASE STUDIES FOR THE AGILE (XP) TRACEABILITY AUDIT

	Authors	Case title	Case objective	Case Date
Case 1	Angelina Espinoza, Juan Garbajosa	A Study to Support Agile Methods More Effectively Through Traceability	To propose a model, called the traceability meta model (TmM) to support the traceability of XP by developing three features of the TmM which are user-definable traceability links, roles, and linkage rules. The proposed model is aimed at improving and enhancing XP maintainability processes.	March, 2011
Case 2	Christopher Lee, Luigi Guadagno, Xiaoping Jia	An Agile Approach to Capturing Requirements and Traceability	To propose a tool, called Echo, to capture user stories, and any informal information generated during the development phases, and restructure that information to better support XP traceability and change management.	October, 2003
Case 3	Arbi Ghazarian	Traceability Patterns: An Approach to Requirement-Component Traceability in Agile Software Development	To propose a conceptual model that captures a traceability pattern in XP. The approach focuses on providing traceability through mapping the user stories to the source code components after defining certain design constraints, such as the location, naming, and content constraints.	November, 2008
Case 4	Sukanya Ratanotayanon, Susan Elliott Sim, Rosalva Gallardo-Valencia	Supporting Program Comprehension in Agile with Links to User Stories	To propose Zelda, which is an Eclipse plug-in tool designed to work with XP to support the traceability of source codes generated using agile software processes by helping developers create links from user stories to source code, and test cases.	August, 2009
Case 5	Yaser Ghanam, Frank Maurer	Extreme Product Line Engineering: Managing Variability & Traceability via Executable Specifications	To propose an approach for managing XP variability and traceability using executable specifications.	August, 2009

B. Discussion of the Evidence Gathered

The international standard of auditing [16] defines audit evidence as "all the information used by the auditor in arriving at conclusions on which the audit opinion is based." The audit evidence is described by the ISA as "proofs, facts and information about something to convince the auditors that something is true, fair or false. It gives auditors reasonable assurance and not absolute assurance about something." This standard was designed for auditing financial systems and financial records, and

examples of auditing evidence are counting records, internal and external documents, and physical observations.

For information systems, auditors usually look for evidence of the existence of internal controls. The COBIT (Control Objectives for Information and related Technology) [19] defines internal IT controls as specific activities performed by persons or systems designed to ensure that business objectives are met. As indicated by COBIT, internal IT controls can be implemented at different levels (organization, process, and product) to support business objectives, such as process activity integrity, reliability, and compliance.

In our paper here, we look at the set of mechanisms, techniques, approaches, and documentation implemented to support the traceability method pertaining to internal agile software process control. We have audited the case studies described in Table II, based on our proposed auditing model, to determine whether or not they can provide evidence of the implementation of the audit yardsticks – see Table III. For all the case studies in this paper, the evidence was gathered using the information system audit procedure described by the Information Systems Audit and Control Association (ISACA), as follows:

• Observation of traceability processes and the existence of the components of the traceability method.

• Documentary audit evidence, such as results of the traceability method execution, and records of the method performance.

• Representations of the method, such as written analyses, and descriptions of the traceability method and traceability method flowcharts.

The following comments can be made based on the evidence gathered:

- The traceability method in Case A implements a meta model for agile process traceability based on the ISO-24744:2007 meta model, which was designed based on the UML architecture and notation. Case B was also designed based on the UML architecture and notation. Both cases therefore provide evidence of intellectual concept design rather than mathematical model design, and similarly for Cases C, D, and E.
- Case B shows partial evidence of operational principles, as the traceability approach only covers the requirements phase, and similarly for Case C, since it shows support for traceability for the requirements, design, and coding phases. No evidence was found that the traceability approach is supported in the planning, testing, validation, and verification phases.
- For Case B, there is partial support for the documentation access audit yardstick, since a mechanism was implemented in this case for accessing and retrieving the traceability items produced during the requirements phase, but there is no evidence of right of access mechanisms. The same is true for Case D.
- No evidence was found for traceability role identification in Case B, and the project stakeholders have the same right to access, modify, and retrieve the traceability items. The same is true for Case D.
- Little evidence was found of support for the audit model in Case E. The approach presented in Case E was implemented to support traceability between the coding and testing phases in XP.
- For Case E, no evidence was found for traceability item relationship identification or traceability role identification. Nor was evidence found of traceability documentation, such as traceability logs, the history of traceability items, and the relationships among traceability items, and so on. No evidence was found supporting documentation access or access rights either.

TABLE III:
EXISTENCE OF EVIDENCE IN THE SELECTED CASE STUDIES

	Case A (Espinoza, Garbajosa)	Case B (Lee *et al.*)	Case C (Ghazarian)	Case D (Ratanotayanon *et al.*)	Case E (Ghanam, Maurer)
Engineering criteria					
	Design of the traceability method				
Yardstick #1 Identification of design intellectual concepts)	Evidence exists	Evidence exists	Evidence exists	Evidence exists	Evidence exists
Yardstick #2 (Identification of design mathematical models)	Evidence does not exist	Evidence does not exist	Evidence does not exist	Evidence does not exist	Evidence does not exist
	Coverage of the traceability method				
Yardstick #3 (Full operational principles)	Evidence exists	Evidence does not exist	Evidence does not exist	Evidence exists	Evidence does not exist
Yardstick #4 (Partial operational principles)	Evidence does not exist	Evidence exists	Evidence exists	Evidence does not exist	Evidence exists
Management criteria					
	Identification of the traceability method				
Yardstick #5 (Traceability item identification)	Evidence exists	Evidence exists	Evidence exists	Evidence exists	Evidence exists
Yardstick #6 (Traceability item relationships)	Evidence exists	Evidence exists	Evidence exists	Evidence exists	Evidence does not exist
Yardstick #7 (Traceability role identification)	Evidence exists	Evidence does not exist	Evidence does not exist	Evidence does not exist	Evidence does not exist
	Monitoring of the traceability method				
Yardstick #8 (Traceability documentation)	Evidence exists	Evidence exists	Evidence exists	Evidence exists	Evidence does not exist
Yardstick #9 (Documentation access)	Evidence exists	Evidence partially exists	Evidence does not exist	Evidence partially exists	Evidence does not exist

VII. Conclusion and Future Work

Auditing is an important process in a software organization which needs ISO 9001 certification. This means that the organization must demonstrate with documented evidence that their processes have been executed in conformity with ISO 9001 [20, 21]. However, software organizations that adopt lightweight documentation processes such as XP find it a challenge to demonstrate that they meet ISO 9001 requirements by providing such documentation.

This paper proposes an auditing model for ISO 9001 traceability requirements for agile software processes, in particular for XP. This model can help software organizations in their effort to achieve ISO 9001 certification and help software auditors to extract auditing evidence that demonstrates the ability of a software organization to implement the ISO 9001 traceability requirements. The design methodology for the proposed auditing model is based on evaluation theory. The model consists of two major categories of auditing criteria: engineering criteria and management criteria. Each auditing criterion consists of several auditing yardsticks, which focus on the evidence that can be extracted to demonstrate process conformity to the ISO 9001 traceability requirements. Five different case studies have been audited based on the proposed model to investigate whether or not they conform to the ISO 9001 traceability requirements. The evidence gathered shows at least partial support for the requirements in each case study, however no case study has demonstrated full support for the auditing yardsticks.

This paper has focused solely on designing an auditing model for the traceability requirements of ISO 9001. Future work is required to extend this model to include other ISO 9001 requirements, such as the control of design and development changes, as well as measurement analysis and improvement. This would allow the auditing model to cover all the mandatory ISO 9001 requirements for both software organizations and software auditors.

References

[1] Griesemer, J. (1999). “A Field Study of the Impact of ISO 9001 on Software Development in the United States”, PhD thesis, Pace University, United State of America.

[2] Fuller, G. K. (2006). “Antecedents and Consequences of Certification of Software Engineering Processes”, PhD Thesis, University of British Columbia, Canada.

[3] Ferreira, A., G. Santos , G. Santos, R.Cerqueira, M. Montoni, A. Barreto, A. Oliveira, S. Barreto, A. Rocha. (2007). “Applying ISO 9001:2000, MPS, BR and CMMI to Achieve Software Process Maturity: BL informatica’s pathway”. 29th Int. Conference on Software Engineering, Minneapolis, USA, pp. 642-651.

[4] Vitoria, D. (2004). “Aligning XP with ISO 9001:2000-TickIT guide 5.0”, Master Thesis Software Engineering”, Blekinge Institute of Technology, Sweden.

[5] Vriens, C.H. (2003). “Certifying for CMM level 2 and ISO9001 with XP@Scrum”, Conference on Agile Development, IEEE Computer Society, Washington DC, USA, pp. 120-124.

[6] Wright, G. (2003). “Achieving ISO 9001 Certification for an XP Company", Lecture Notes in Computer Science, Extreme programming and Agile Methods, agile universe, Springer Berlin / Heidelberg, New Orleans, pp. 43-50.

[7] Vijayasarathy, L. and Turk D. (2008). “Agile Software Development: A Survey of Early Adopters”. Journal of Information Technology Management, 19(2), pp. 1-8.

[8] Schindler, C. (2008). “Agile Software Development Methods and Practices in Austrian it Industry Results of an Empirical Study”, International Conferences on Computational Intelligence for Modeling, Control and Automation, Vienna, Austria, pp. 321-326.

[9] Espinoza A. and Garbajosa J., “Study to Support Agile Methods More Effectively through Traceability”, Computer Science Innovations in Systems and Software Engineering, Vol. 7, No. 1, 2011, pp. 53-69.

[10] Ghazarian A., “Traceability Patterns: An Approach to Requirement Component Traceability in Agile Software Development”, 8th WSEAS International Conference on Applied Computer Science, Venice, Italy, 2008, pp. 236-241.

[11] Lee C., Guadagno L., Jia X., "An Agile Approach to Capturing Requirements Traceability", 2nd International Workshop on Traceability in Emerging Forms of Software Engineering , Canada, October, 2003, pp 104-110.

[12] Ramesh B., Jarke M., “Towards Reference Models for Requirements Traceability”, IEEE Transactions on Software Engineering, Vol. 27, No. 1, 2011, pp. 58-93.

[13] Cohn M. and Ford D. "Introducing an Agile Process to an Organization", IEEE Computer, 36(6), 2003, pp.74-78.

[14] Systems Audit and Control Association (ISACA), “Standards, Guidelines and Procedures for information system auditing”, 2010, http://www.isaca.org/Knowledge Center/Standards/Documents/ALL-IT-Standards-Guidelines- and-Tools.pdf.

[15] Scott Ambler, "Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process", John Wiley & Sons, 1st edition, pp. 1-400.

[16] International Federation of Accountants, "International Standard on Auditing 500", 2009. http://www.ifac.org/sites/default/files/downloads/a022-2010-iaasb-handbook-isa-500.pdf

[17] Scriven M., "Evaluation in the New Millennium: The Transdisciplinary Vision", in S. I. Evaluating Social Programs and Problems: Visions for the new millennium Donaldson & M. Scriven (Eds.). Mahwah, NJ: Lawrence Erlbaum Associates. 2003 pp.19-42.

[18] Vincenti W. G., “What Engineers Know and How They Know It”, The John Hopkins University Press, Baltimore, London, 1990.

[19] Abran A., Bourque P., Dupuis R., Moore J., Tripp L., “Guide to the Software Engineering Body of Knowledge”, IEEE Computer Society Press, 2004, pp. 1-228.

[20] Meridji. K, "Analysis of Software Engineering Principles From an Engineering Perspective", Ph.D. dissertation, École de technologie supérieure, Montréal (Canada), 2010.

[21] Paul C., "Process Driven Comprehensive auditing: a New Way to Conduct ISO 9001:2008 Internal Audits", ASQ Quality Press, 2nd edition, June 24, 2009.

[22] Chorafas N., "IT auditing and Sarbanes-Oxley Compliance: Key Strategies for Business Improvement", Auerbach Publications, 1st edition, October 29, 2008.

[23] Chambers A., and Rand G, "Operational Auditing: Auditing Business and IT Processes”, Wiley, 2nd Edition, 2010.

[24] Kowalczykiewicz K., and Weiss D., “Traceability: Taming Uncontrolled Change in Software Development”, 4th

National Software Engineering Conference, Tarnowo Podgorne, Poland, 2002.
[25] Kilpinen, M.S. (2008) 'The Emergence of Change at the Interface of Systems Engineering and Software Design: An Investigation of Impact Analysis', PhD-thesis, Cambridge University Engineering Department.
[26] Lopez M., "An Evaluation Theory Perspective of the Architecture Tradeoff Analysis Method (ATAM)", CMU/SEI-20Q0-TR-012, Pittsburgh, PA, CMU/SEI, 2000.
[27] Lopez M., "Application of an Evaluation Framework for Analyzing the Architecture Tradeoff Analysis Method", Journal of Systems and Software, Vol.68, No.3, 2003, pp. 233-241.
[28] Zarour M., "Methods to evaluate lightweight software process assessment methods based on evaluation theory and engineering design principles", Ph.D. dissertation, École de technologie supérieure, Montréal (Canada), 2010.
[29] IT Governance Institute, Cobit 4.1, Information Systems Audit and Control Association (ISACA), ISBN:1933284722 9781933284729,2007. "http://dl.acm.org/citation.cfm?id=1534415".
[30] Qasaimeh M., and Abran A., "Investigation of the Capability of XP to Support the Requirements of ISO 9001 Software Process Certification", Eighth ACIS International Conference on Software Engineering Research Management and Applications, Montreal, Canada, 2010, pp. 239-247.
[31] Qasaimeh M., and Abran A., "Extending Extreme Programming User Stories to Meet ISO 9001 Formality Requirements", Journal of Software Engineering and Applications, Vol.4, No.11, 2011, pp.626-638.

Malik Qasaimeh is an assistant professor of software engineering in Princess Sumaya University for Technology-Jordan. He received his PhD degree in software engineering form University of Quebec (Canada, 2012). He has a Master's degree in Information Systems Security from Concordia University (Canada, 2007), and a Bachelor's degree in Computer Science from Jordan University of Science & Technology, (Jordan, 2003). He has published several papers in a reputed journals and international conferences. His research interests include agile software processes certification and compliance, software process and product improvement, software engineering ISO standards and software engineering principles.

Alain Abran is a Professor and the Director of the Software Engineering Research Laboratory at (ETS) university of Quebec (Montréal, Canada). He Holds a Ph.D. in Electrical and Computer Engineering (1994) from the École Polytechnique de Montréal (Canada) and Master's degrees in Management Sciences (1974) and Electrical Engineering (1975) from the University of Ottawa. He has over 15 years of experience in teaching in a university environment, and more than 20 years of industry experience in information systems development and software engineering. His research interests include software productivity and estimation models, software engineering foundations, software quality, software functional size measurement, software risk management, and software maintenance management. He has published over 300 peer-reviewed publications and he is the author of the book "Software Metrics and Software Metrology" and a co-author of the book "Software Maintenance Management" (Wiley Interscience, Ed., & IEEE-CS Press). Dr. Abran is co-editor of the Guide to the Software Engineering Body of Knowledge – SWEBOK, and he is the chairman of the Common Software Measurement International Consortium (COSMIC).