Journal of Software Engineering and Applications, **2011, 4, 626-638**
doi:10.4236/jsea.2011.411074 Published Online November 2011 (http://www.SciRP.org/journal/jsea)

Extending Extreme Programming User Stories to Meet ISO 9001 Formality Requirements

Malik Qasaimeh, Alain Abran

École de Technologie Supérieure, University of Québec, Montréal, Canada.
Email: malik.qasaimeh.1@ens.etsmtl.ca, alain.abran@etsmtl.ca

Received August 26th, 2011; revised September 30th, 2011; accepted November 9th, 2011.

ABSTRACT

For software organizations needing ISO 9001 *certification, including those that have adopted agile methodologies, it is important that their software life cycle processes be able to manage the requirements imposed by this certification standard. However, the user stories in the XP agile methodology do not provide auditors with enough evidence that certain steps and activities have been performed in compliance with ISO* 9001. *This paper proposes an extension to the user story, based on four sub processes related to the CMMI-DEV model*: 1) *identification of the source of the user story*; 2) *categorization of the non functional requirements*; 3) *identification of the user story relationships*; *and* 4) *prioritization of the user stories. These sub processes are aligned with the XP release planning phase, and enhance the ability of user stories to accumulate the information that is mandatory for achieving ISO* 9001 *certification.*

Keywords*: *Extreme Programming*, *ISO* 90001, *Agile Process Improvement*, *Certification Process

1. Introduction

ISO 9001 was originally designed for the manufacturing sector; however, this standard is now being used in many other sectors as well, including health care and software. The development of software has become important to industry, and the ISO has developed and released a guidance document, ISO 90003, to provide a roadmap for software development organizations wishing to become ISO 9001 certified.

Those organizations wishing to do so must be audited to show evidence that they have met the ISO 9001 requirements. The advantages for software organizations of obtaining a certification such as ISO 9001 were investigated in [1], with an assessment of the efficiency of Thai software organizations that are both ISO 9001: 2000 and TQS (Thai Quality Software) certified. To obtain the required data they needed from software producers and developers, these authors carried out a field survey through interviews and a questionnaire to assess an organization's process efficiency based on four criteria: financial health, customer satisfaction, internal business process quality, and the level of learning and growth. The interviews were conducted among 56 organizations producing and developing software (23 organizations with ISO 9001:2000 certification and 33 organizations with TQS certification).

The efficiencies of the organizations having ISO 9001: 2000 and TQS certification were compared, and the following observations were made:

- The efficiency of ISO certified organizations is higher than that of TQS certified organizations, in terms of product quality and customer satisfaction.
- ISO certified organizations place greater focus on customer satisfaction than TQS certified organizations.
- The level of learning and growth in both kinds of organizations is in the middle of the comparison scale, which implies that innovation is considered effectively by the organizations studied.

The advantages of ISO 9001 certification are well understood by software organizations. Recently, however, the market penetration of the documentation-light agile software processes (e.g. extreme programming-XP) has been increasing [2,3]. "Agile development processes have a different perspective compared to traditional development processes which follow a more linear or waterfall model for performing tasks. One of the differences is that a detailed requirements specification may be missing during a large part of the project or even the whole project duration. Some other differences include the use of stories as a source for requirements. Stories include many

details and may be more ambiguous than the conventional requirements specification. A story may also be coarser grained than the traditional requirements specification" [4].

The authors of [5,6] have investigated the capability of XP to implement the software processes related to the requirements of ISO 9001 and to the guidelines in ISO 90003 based on the ISO 12207 terminology. They observed the following:

- The main means for documenting user requirements in XP is the user story technique. However, the user story provides fewer details than what is specified by ISO 9001 and ISO 90003. For exam- ple, the user story technique records a high-level description of user requirements. However, it does not record the details of face-to-face communica- tions with the user during the iterative planning process, not does it take into account the system requirements or any of the technical details need- ed during development. Also, it is not clear how XP can trace the software artifacts back to the customer requirements.
- User stories are mainly written in natural langua- ge, and they provide no formal specifications. Ra- ther, they are evaluated by prototypes and by on- site customers. Formal evaluations, such as model validations, are not supported by XP.

ISO 9001 demands of (software) organizations that a rigorous demonstration of their software processes be implemented and a set of guidelines followed at various levels of abstraction. What these organizations need to show, in other words, is that their software processes have been designed and implemented in a way that allows for a level of configuration and operation that complies with ISO 9001 requirements.

This paper proposes four sub processes (activities) aligned with the XP release planning phase. These sub processes are: 1) identification of the user story resource; 2) identification of a non functional requirements category; 3) identification of user story relationships; and 4) identification of user story priorities. The aim of these sub processes is to modify the structure of traditional user stories in order to provide the ISO 9001 auditor of XP with sufficient evidence that the data they require have been collected, and to provide traceability for the requirements throughout the earliest phases of XP (*i.e.* the release planning phase).

This paper is organized as follows. Section 2 clarifies the main terms and definitions that will be used in this paper. Section 3 presents the methodology and the objectives of the paper. Section 4 describes in detail each of the proposed sub processes. Section 5 describes the main structure of the extended user story based on the proposed sub processes. Section 6 discusses the potential benefits of this work from the ISO 9001 viewpoint.

2. Terminology

This section presents the definitions of the terms that will be used in this paper.

2.1. System

A system is defined by ISO 15288:2008 as a combination of interacting elements organized to achieve one or more stated purposes. An element is a discrete part of the system that can be implemented to fulfill specified requirements, and can be hardware, software, data, humans, or processes (e.g. processes for providing a service to users). In this context, the system is viewed as a collection of interacting elements organized to accomplish a specific function, or set of functions, within a specific environment.

2.2. System Feature and System Function from the XP Viewpoint

The differences between a system feature and a system function are poorly defined in the literature. In XP, a user story is designed to specify a goal from the user viewpoint and to specify a feature from the system viewpoint. As a result, user stories often represent user needs, which will ultimately include both essential and nice-to-have features. The collection of those features will be integrated later in the process life cycle into system elements to provide a function to the system. Every XP iteration provides the system with functionality, based on the collection of features originally implemented based on the user stories. For example, Add User, Grant Privilege to User, Delete User, and List Users are system features that can be represented at the requirements level by means of a user story. The result of implementing user stories is a system function, such as a "user administration system". **Figure 1** illustrates the concepts of feature, function, and system from the XP perspective.

3. Methodology and Objectives

The CMMI for development, version 1.2 (CMMI-DEV, v1.2), includes some process areas for identifying and managing software requirements, and contains useful guidelines and best practices for specifying them. In the context of this paper, three different CMMI process areas (*i.e.* requirement development, requirement management, and risk management) have been analyzed to derive a set of sub processes that could be aligned with the exploration phase of XP release planning—see **Figure 2**.

The objectives of these sub processes can be summarized as follows:

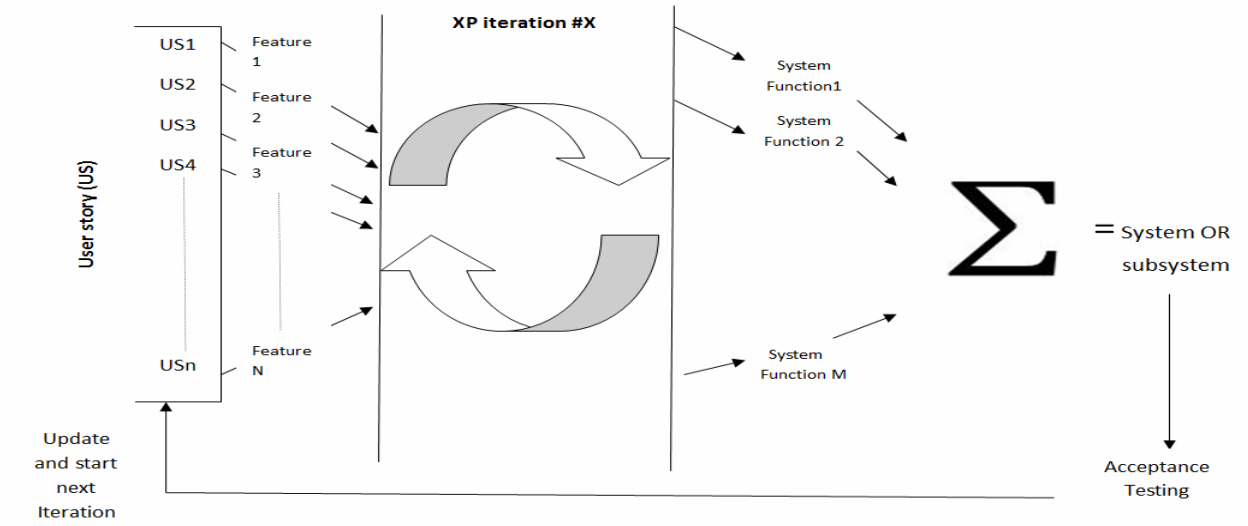

Figure 1. Relationship between system features and system functions in XP.

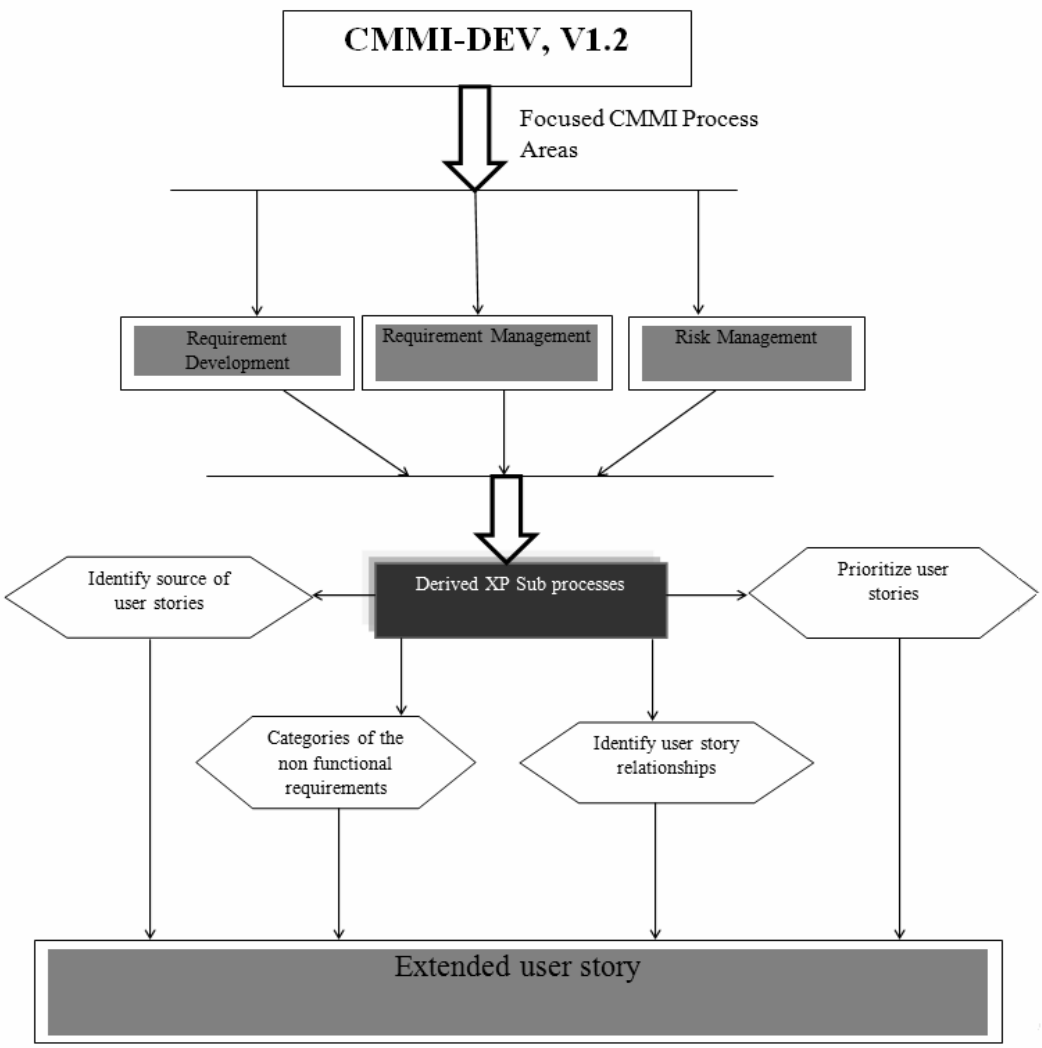

Figure 2. Methodology for deriving the XP sub-processes.

- ✓ Provide the basic metadata for managing the information gathered during XP release planning.
- ✓ Set a standard for the information and the data gathered during XP release planning; this will allow a relationship to be defined between user stories.
- ✓ Provide structured user stories that can present more information concerning dependencies between user stories and other artifacts of the XP development life cycle.
- ✓ Provide standardization across XP processes to support user story management. Standardizing user story cards, for example, will help raise the visibility of the process of capturing both functional and non functional requirements.
- ✓ Provide more information about stakeholders and the source of user stories; this will allow better decisions to be made, development times to be reduced, customer satisfaction to be improved, and the basic information for supporting XP traceability to be provided.

Table 1 shows each process area and the process goals that have been investigated, as well as the related derived XP sub processes.

4. Proposed Sub Processes

4.1. Identify the Source of the User Story

The requirements engineering process focuses on stakeholder needs. The goal is to identify all the people, organizations, and other systems that have a direct or indirect impact on the user stories elicited. "Much software has proved unsatisfactory because it has stressed the requirements of one group of stakeholders at the expense of those of others. Hence, software is delivered which is difficult to use or which subverts the cultural or political structures of the customer organization.

Table 1. CMMI-DEV.

CMMI Process Areas Investigated	Process Goal	Derived XP Sub processes
Requirement development (RD)	Elicit needs Develop customer requirements Establish a definition of required functionality Analyze requirements to achieve balance	Nonfunctional requirements categorization User story prioritization
Requirement management (REQM)	Understand requirements Obtain commitment to requirements Manage changes to requirements	Identify source of user stories User story relationships
Risk management (RM)	Determine risk sources and categories Identify risks Evaluate, categorize, and prioritize risks	Identify source of user stories Identify user story relationships Prioritize user stories

The software engineer needs to identify, represent, and manage the "viewpoints" of many different types of stakeholders [7]. Software development teams should understand the sources that directly or indirectly influence the creation of user stories, in order to be able to trace each story back to its original source in the case of an improvement or change request. Therefore, the <<STORY CONTRIBUTOR>> is defined as individuals, including the customers or clients who pay for the system, the developers who design, construct, and maintain the system, and the users who interact with the system to get their work done, as well as other systems or organizations that need to collaborate with the system. The schema proposed by [8] has been used to identify the <<STORY CONTRIBUTOR>> from the ISO 9001 perspective. The author of [8] suggests a list (provided below) of candidate stakeholders who may contribute to the progress of any software project, *i.e.* people who:

- manage, introduce, operate, or maintain the system after its deployment;
- are involved in developing the system, including architects, developers, testers, quality engineers, or project managers;
- are responsible for the business or process that the system supports;
- have a financial interest (for example, they paid for it or are responsible for selling it) Asemicolon;
- constrain the system as regulators (for example, through the laws and international software standards such as ISO 9001 that may impact the system.

Usually, the <<STORY CONTRIBUTOR>> varies according to the nature of the system being developed; for example, the system may be intended to provide special services inside the organization, such as a payroll system or documentation management system, or perhaps the system is related to public services, such as air traffic control. ISO 9001 requires these <<STORY CONTRIBUTORS>> to be clearly identified and categorized. Therefore, to improve the accuracy of the user story, it has been proposed that its source, *i.e.* the <<STORY CONTRIBUTOR>>, belong to one or more contributor types—see **Figure 3**—which have been developed based on [1,8] and ISO 9001:

- Customer side contributors,
- Development side contributors, and
- Government side contributors.

<<STORY CONTRIBUTORS>> are assumed to provide the features of their system that could affect the

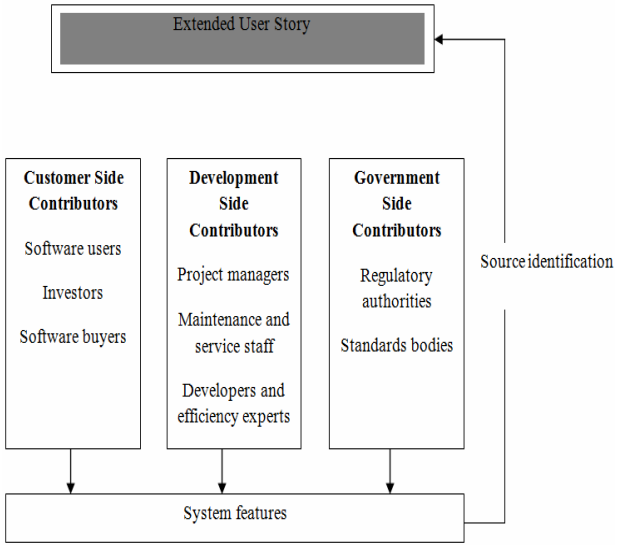

Figure 3. User story sources—the various types of contributors.

various levels of the system, such as the process level, the product level, and the project level. While this list is not exhaustive, it does provide guidance to help in identifying the source of the user stories—see **Figure 3**.

4.1.1. Customer Side Contributors

Software users: Those with a direct interest in the functions provided by the proposed new system or services. Software users are valuable sources of knowledge of the features that the system is designed to implement. They can provide insights into how the system should operate.

Investors: Those responsible for providing the required funding for the proposed system, including the organizations responsible for developing the system or an external party wishing to invest in the system. These contributors may have their own features that they consider would better implement the system's user stories. Usually, features provided by investors are related to system efficiency and to the performance of the system. The investors play an important role in balancing, and scoping, costs and perceived benefits.

Software buyers: Those who purchase large and complex software, public software, for example, such as air traffic control system or online banking system, and who could be different from the users of the software. System features from these contributors are derived from their own expectations on how to better support user needs.

4.1.2. Development Side Contributors

Project managers: Those responsible for managing the technical aspects of the project (e.g. the development process) and its non technical aspects (e.g. budget and development time). Requirements and constraints from project managers are focused as much on bringing discipline to the delivery schedule as to moving the project on to successful completion within the specified budget. Requirements from project managers are usually related to regulating the workflow of the project and focus less on system features.

Maintenance and service staff: Those whose main responsibility is to keep the system operating after it has been delivered to the system users. Requirements from these contributors are focused on a set of controls designed to better maintain the system later.

Developers and the quality assurance team: Those whose main responsibility is to design, implement, and test the system, and to verify that all the system user stories from all the story contributors have been implemented efficiently. They focus on the overview at the application level, rather than at the component level or individual programming task level. Therefore, they may contribute stories to the system concerning controls and indicators for monitoring and measuring the various characteristics and sub characteristics of system quality.

4.1.3. Government Side Contributors

Regulatory authorities and standards bodies: To ensure the compliance of organizations with codes of practice, government regulations, etc., such as Sarbanes-Oxley (SOX), the Food and Drug Ad- ministration (FDA), and the Health Insurance Portability and Accountability Act (HIPAA). It is the responsibility of every organization to develop its own business processes to address them, and the Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) recognizes that a software development process might be a part of such a business process [7]. The SWEBOK Guide also points out that there is broad acceptance that software development success is highly dependent on the software requirement activities. Therefore, user stories should be able to capture and manage the requirements (functional and non functional) of government side contributors. At the business process level, organizations react to the regulatory authorities and standards bodies by developing what are called internal controls (*i.e.* policies and procedures). "Software is often required to support a business process, the selection of which may be conditioned by the structure, culture, and internal politics of the organization" [7]. An organizational policy can be described as a formal statement that guides and steers production methodologies, and so every organization must ensure that their policies comply with the rules of the authority that governs it. An organizational procedure is a series of steps required to implement the organization's policies. It is essential, therefore, that software developers an- alyze the applicable rules for implementing the organization's internal controls. From the software engineering perspective, these internal controls are translated into application support software and control support software—see **Fig-**

ure 4.

- Application support software is software that provides a specific set of user-level functions, such as a reporting system or an employment management system.
- Control support software is software that auto- mates the organizational policies and procedures, or provides technical services to the organization.

Control support software includes control components, which can be classified as follows:

- Application level control component: a control element implemented and integrated into the sys- tem for a specific automated service; for example, services to ensure that all goods shipped are in- voiced.
- Process level control component: a control ele- ment implemented and integrated into the system to support the overall business process; it includes adequate security functionality to prevent unauthorized access to secure applications.
- Technical level control component: a control element implemented to support the organization at the operational level; for example, implement the organization's internal policies or procedures, or to ensure that policies and procedures are implemented by the operational system and business processes.

To this end, user stories should capture the sources of the requirements from the government side contributors for the regulatory authorities and standards bodies, in order to ensure that a software system is capable of meeting government and business requirements, and to provide the ISO 9001 certifying authority with evidence that data from those sources have been collected.

4.2. Categories of Non Functional Requirements

The goal of this section is to provide formal evidence that the non functional requirements have been gathered from the user stories and categorized based on their respective groups (a related work on the formal specification of non functional requirements can be found in [9,10]).

During XP release planning, the <<STORY CONTRIBUTOR>> informally states the non functional requirements that need to be considered for each user story. Every <<STORY CONTRIBUTOR>> sees the problem from a different perspective. As users often do not know which quality attributes they would like to see included, they can express their non functional requirements orally [11]. Developers must therefore be able to understand and categorize those non functional requirements and map them to the corresponding quality attribute(s) in order to comprehend the entire problem domain. To enhance the ability of user stories to capture non functional requirements during the early phases of XP, a semi structured format is proposed for defining them. This allows developers to identify the category to which the non functional requirements of each user story belong, as well as to provide a flexible format for both the functional and non functional requirements. The set of quality attributes is represented in the format {Q1,Q2,···Qn}, and the sub quality attributes associated with the non functional

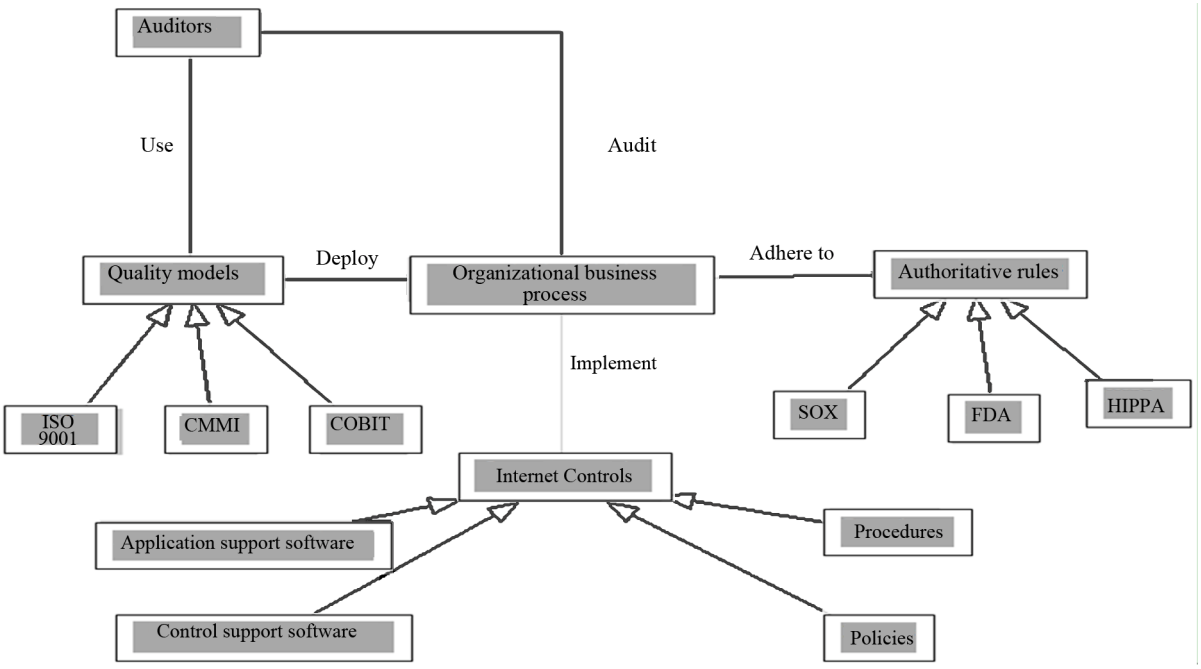

Figure 4. Government side contributors.

requirements required by a user story in the format {SQ1,SQ2,···SQn}. Also, there are many quality models that address the quality attributes and non functional requirements of software systems, such as the European Cooperation on Space Standardization (ECSS), Boehm, McCall, and ISO 9126 models. The ISO 9126 quality model refers to six quality characteristics, subdivided into twenty-seven quality sub characteristics for the internal and external quality of a software product —see **Table 2**.

AS a <<STORY CONTRIBUTOR>>, *I want the system to* <<*DO REQUIREMENTS*>>

AND incorporate <<*NON FUNCTIONAL CAPABILITIES*>>, *which belong to*

Quality characteristics {*Q*1, *Q*2···,*Qn*} *AND*

Sub quality characteristics {*SQ*1, *SQ*2,···*SQm*} *respectively*

Each story is primarily associated with a <<*NON FUNCTIONAL CAPABILITY*>> entity that represents the category of non functional requirement intended for each story. The purpose of a <<*NON FUNCTIONAL CAPABILITY*>> entity is to keep the user story as lightweight as possible, but at the same time to provide evidence for an ISO 9001 auditor that non functional requirements have been obtained during the early phases of XP. The <<NON FUNCTIONAL CAPABILITY>> category could represent one or more quality characteristics and sub quality characteristics belonging to the non functional requirements stipulated by the <<STORY CONTRIBUTOR>>. **Table 3** shows examples of non functional capability categories.

Table 2. ISO 9126 quality characteristics.

Characteristics	Sub characteristics
Functionality	*Suitability* *Accuracy* *Interoperability* *Compliance* *Security* *Functional Compliance*
Reliability	*Maturity* *Recoverability* *Fault Tolerance* *Reliability Compliance*
Usability	*Learnability* *Understandability* *Operability* *Attractiveness* *Usability Compliance*
Efficiency	*Time behavior* *Attractiveness* *Resource behavior* *Efficiency Compliance*
Maintainability	*Stability* *Analyzability* *Changeability* *Testability* *Maintainability Compliance*
Portability	*Installability* *Co-Existence* *Replaceability* *Adaptability* *Portability Compliance*

4.3. Identify the User Story Relationships

Based on the description of system features and system functionality in section 2, we next define the relationships between dependent user stories. For example, a user story "j" that depends on another user story "i" is called dependent, and is denoted <US,j>. Such a pair of dependent user stories will be read as follows: <US,j> depends on <US,i>. The dependencies between user stories are then classified into four categories: logical dependencies, data dependencies, temporal dependencies, and resource dependencies.

This classification is based on the user story features that require implementation.

- A logical dependency occurs when the feature implemented by a user story X cannot be executed before the feature implemented by user story Y, because they are logically dependent. This can be the case if user story X provides services or interfaces to user story Y. For example, in an employment management system, the employee will not be granted access to perform restricted operations unless he has

Table 3. Examples of non functional capability categories.

Example	<<*NON FUNCTIONAL CAPABILITY*>>
The customer must place an order within two minutes of registering.	Performance
The customer must be able to access their account 24 hours a day, 7 days a week.	Availability
"Update Customer" will be available to users during 98% of normal working hours.	Reliability
Up to 200 new sites per year may start to use "Update Customer".	Scalability

been approved as a legitimate employee. This can be read as follows: <US,j> logically depends on <US,i>. This relation can be represented as in **Figure 5**.

- A data dependency occurs if the feature implemented by user story X cannot be executed before the feature implemented by user story Y, because they are data-dependent. This can be the case if user story X provides input data for user story Y. For example, sorting the entries in the database should be performed after this entry has been stored. This can be read as follows: <US,j> data depend on <US,i>. This relation can be represented as in **Figure 6**.
- A temporal dependency occurs if the feature implemented by user story X cannot be executed before the feature implemented by user story Y, because they are time-dependent. In this case, feature x specifies the time frame for an event to occur, for a process to be completed, or a condition to hold true, for example, in order for feature y to start processing. Temporal dependencies can be found in designing the user stories of a real-time system, where the system features must execute respecting strict response time constraints. This can be read as follows: <US,j> depends temporally on <US,i>. This relation can be represented as in **Figure 7**.
- A resource dependency occurs if the feature implemented by user story X cannot be executed before the feature implemented by user story Y, because they are resource-dependent. In this case, the system consists of several concurrent threads (*i.e.* features) which are competing for limited resources (*i.e.* hardware resources or software resources). User stories should be analyzed first, so that precautions can be taken to ensure fairness. This can be read as follows: <US,j> resource depends on <US,i>. This relation can be represented as in **Figure 8**.

4.4. Prioritizing the User Stories

Prioritization is the process of making a choice among multiple options [12]. It is also considered an important activity in requirements engineering, as it helps developers analyze requirements in order to rank them according to their importance from the perspective of the requirements analyzer or the stakeholder who is involved in the requirements elicitation activity [13].

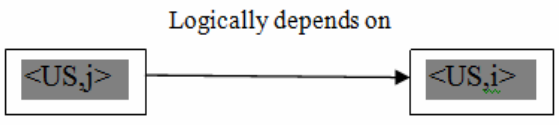

Figure 5. Logical dependency.

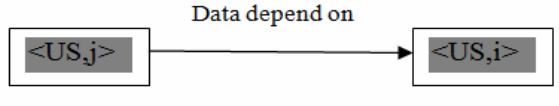

Figure 6. Data dependency.

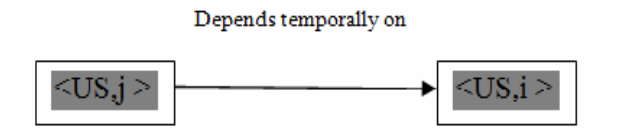

Figure 7. Temporal dependency.

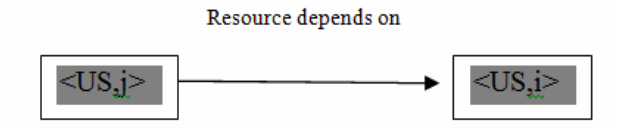

Figure 8. Resource dependency.

Requirement prioritization processes can be categorized into methods-based solutions and negotiation-based solutions. Methods-based solutions are aimed at assigning quantitative values to the requirements, such as the binary priority list methods in [14], while negotiation-based solutions focus on resolving conflicts by brokering an agreement between stakeholders on ranking requirements using a method selection framework designed for the purpose, such as the Negotiation Constellations in [15].

In XP, user stories are usually prioritized before each iteration during the exploration phase of release planning, specifically in the Planning Game activity, in which the on-site customer classifies the user stories into three groups: "those without which the system will not function," "those that are less essential, but provide significant business value," and "those [it] would be nice to have" [16]. This XP activity can be considered as a type of negotiation-based solution that is less formal from the ISO 9001 perspective and which normally provides evidence that criteria have been met by the on-site customer on sorting the user stories into their corresponding categories. Therefore, we propose that the AHP (Analytic Hierarchy Process) be integrated into the XP Planning Game, for the following reasons:

- The AHP combines the advantages of both the methods-based solutions and the negotiation-based solutions, in that the developers, along with any <<STORY CONTRIBUTORS>>, can set the criteria for ranking the user stories into "important" and "less important" stories, based on qualitative and quantitative analysis [17].
- The AHP provides formal evidence that the user stories have been evaluated using criteria which have

been determined to support the priority given by the <<STORY CONTRIBUTOR>> to the various alternatives (such as time, costs, risks, etc.).

- The result of the AHP is highly correlated to the criteria and to the <<STORY CONTRIBUTOR>> viewpoint of what is "important" and "less important". Therefore, developers should establish criteria that balance the goals of the project from different business value perspectives.

Figure 9 depicts the procedure for prioritizing the user stories in XP using the AHP method.

4.4.1. Selection of User Stories for Prioritization

The AHP process begins by defining a set of alternatives from which a decision maker wants to choose (e.g. selection of faculty members, assessment of financial management models, etc.) [18]. There is a variety of methods available for generating those alternatives, such as a brainstorming session, a literature review, or the outcome of a specific process, such as release planning in XP, where the developers, in consultation with the customer, come up with a set of user stories that need to be implemented in subsequent iterations.

At the beginning of each iteration of the exploration phase in XP release planning, the developer gets together with the customer for a planning meeting. In that meeting, they go over the features the customer wants to implement in that iteration, breaking each feature down into individual engineering tasks. In this step, the developers are required to determine the set of user stories that need to become input for AHP prioritization.

4.4.2. Building up Criteria for Comparison Purposes

The AHP allows developers to model the user story ranking as a hierarchical structure, as shown in **Figure 10**.

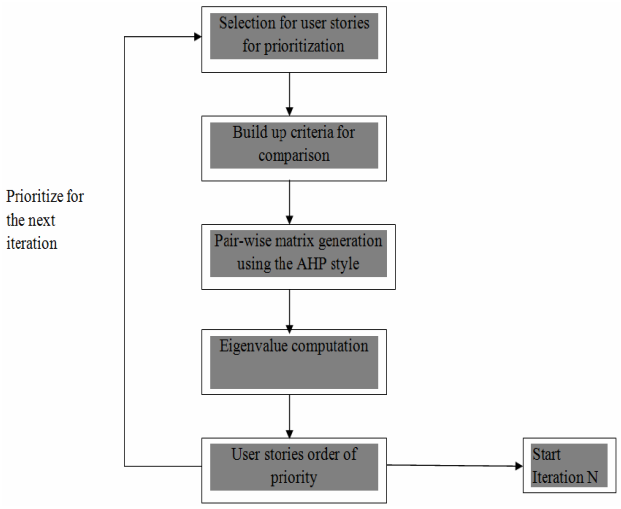

Figure 9. Procedure for prioritizing the user stories in XP using the AHP method.

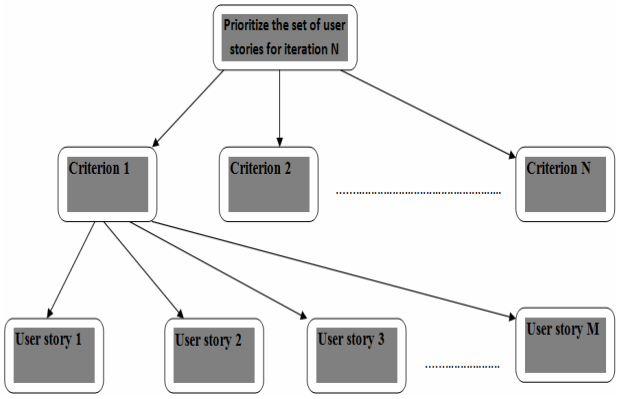

Figure 10. AHP diagram for user story selection.

Using AHP, the definition of criteria is based on the decision maker's viewpoint of what is important from his perspective in evaluating and prioritizing the alternatives. In the context of this paper, each <<STORY CONTRIBUTOR>> can generate his own criteria for ranking the set of user stories. Therefore, the customer side contributors, the development side contributors, and the government side contributors can all generate criteria that can be used to consider different aspects of user story evaluation, such as financial benefits, strategic benefits, competitors, the ability to adhere to standards or regulations, the ability to sell, etc. Next, we give some examples of criteria for developing user stories that consider cost, time, and risk:

- ✓ Cost is often expressed in terms of the number of hours spent developing the software. It is determined by considering the criticality of the requirements and the quality required [19].
- ✓ Cost is often calculated in terms of hours, which is directly related to time. Time is in turn influenced by factors such as degree of parallelism in development, training needs, the need to develop support infrastructure, the need to meet industry standards, etc. [19].
- ✓ There is a degree of risk in every project. Risk management is a process for planning ways to handle the risks that may cause difficulties in development. Among the risks that may be encountered are those related to performance, risks, and scheduling, for example. Calculating the risk per requirement enables engineers to forecast the potential risk at project level [19].

4.4.3. Pair-Wise Matrix Generation Using the AHP Style

Using the AHP pairwise comparison process, weights or priorities are assigned to a set of human judgments based on the AHP scale in **Table 4**. While it is difficult to jus-

tify weights that are arbitrarily assigned, it is relatively easy to justify judgments and the basis for those judgments [17].

The concept of pairwise comparison for prioritizing user stories works as follows: developers begin by computing the priority of their criteria, which are cost, time, and risk in this context. The first step is to generate a pairwise matrix by comparing these three criteria, according to the scale in **Table 4**.

Assume that the following relationships have been determined for these criteria:

- ✓ Cost is much more important than Time (degree of importance: 5).
- ✓ Cost is moderately more important than Risk (degree of importance: 3).
- ✓ Risk is very much more important than Time (degree of importance: 7).

Then, the following pairwise matrix will be generated —see **Table 5**.

Suppose the developers intended to rank three different user stories: <US1>, < US2>, and < US3>. The pair-wise matrix for each criterion should be generated as in **Tables 6**, **7** and **8**.

Table 4. AHP scale.

Intensity of importance	Definition
1	Equally important
3	One moderately more important than the other
5	Much more important
7	Very much more important
9	Extremely important

Table 5. Pairwise matrix for the selected criteria.

		Cost	Time	Risk
$A_{criteria}$ =	Cost	1	5	3
	Time	1/5	1	1/7
	Risk	1/3	7	1

Table 6. Pairwise matrix for the cost criterion.

		US1	US2	US3
A_{cost} =	US1	1	3	5
	US2	1/3	1	1/7
	US3	1/5	7	1

Table 7. Pairwise matrix for the time criterion.

		US1	US2	US3
A_{time} =	US1	1	9	3
	US2	1/9	1	1/5
	US3	1/3	5	1

Table 8. Pairwise matrix for the risk criterion.

		US1	US2	US3
A_{risk} =	US1	1	3	5
	US2	1/3	1	1/3
	US3	1/5	3	1

4.4.4. Eigenvalue Computation

The AHP obtains the weight vector (priority vector) by calculating the eigenvector for the largest eigenvalue of matrix A. This can be obtained using Formula (1).

$$Aw = \lambda w \quad (1)$$

By solving (1) for A criteria, A cost, A time, and A risk, the priority hierarchy will be generated as in **Figure 11**.

$US1_{priority}$, $US2_{priority}$, and $US3_{priority}$ can be obtained as of **Figure 12**.

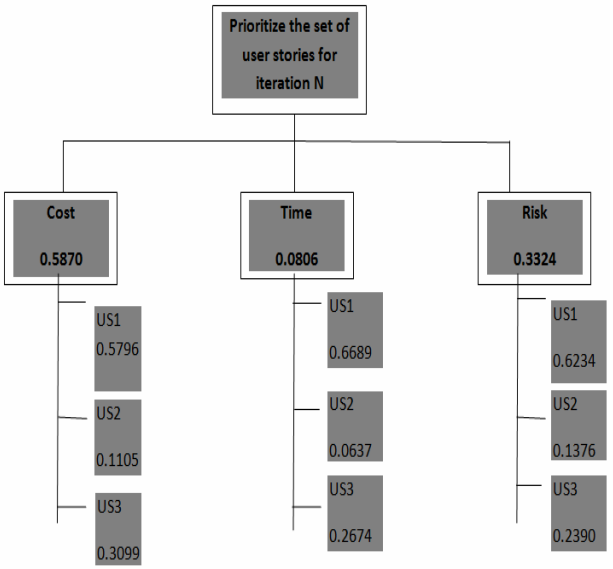

Figure 11. A priority hierarchy.

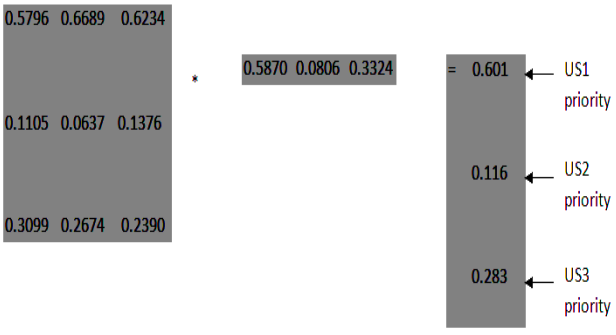

Figure 12. Calculation of user stories priority.

5. Extended User Story for XP

This paper has introduced an extension to the user story to help XP software developers in specifying important information for the ISO 9001 requirements that should be gathered in the earlier phases of software process development. The main content of the extended user story will be as follows—see **Figure 13**.

- **Requirements:** Identification of the user's functional requirements.
- **User story sources:** Identification of user story sources: <<customer side contributor>>, <<development side contributor>>, and/or <<government side contributor>>.
- **Non functional capability:** Identification of the non functional category that represents one or more quality attributes and sub quality attributes belonging to the non functional requirements needed by the <<STORY CONTRIBUTOR>>.
- **Story relationships:** Dependencies between the user stories are identified and classified into logical dependencies, data dependencies, temporal dependencies, and resource dependencies.
- **Priority ranking:** The priority of each user story is calculated based on the AHP method. The <<STORY CONTRIBUTOR>> can generate a priority list for user stories based on predefined criteria.

6. Discussion

The main contribution of this paper is the proposed sub process, aligned with XP release planning, for deriving the extended user story. The following comments illustrate the advantages of the proposed extended user story from the ISO 9001 perspective:

- **Formality:** ISO 9001 auditors need documented evidence at every phase of the development process to clarify that processes are compliant with ISO 9001. The extended user story that we propose here will provide formal evidence that the sources of each user story have been identified. It will also provide formal evidence that each user story has been prioritized from the <<STORY CONTRIBUTOR>> viewpoint. This can be supported by showing documented evidence that every <<STORY CONTRIBUTOR>> generated comparison criteria and pairwise matrices, as well as documented evidence of the final numerical values of the priorities assigned for each user story.

Extended user story	
Requirement	A plan text to indicate the user functional requirements
<<STORY CONTRIBUTOR>>	Identification of user story sources: <<customer sides contributor >><< development sides contributor >><< government sides contributor >>
Non Functional capability	Category could represent one or more a quality characteristics and sub-quality characteristics belong to the non-functional requirements required by the << STORY CONTRIBUTOR >>. Quality characteristics {Q1, Q2.....,Qn} AND Sub-Quality characteristics {SQ1, SQ2,.....SQm}
Story Relations	Identification of user story dependencies. The pair of user stories <US,i> and <<US,j>> is called dependent and will be read as <US,j > depends on <US,i >, if there exists a dependency relation that belongs to one or more of the following categories:. Logical dependencies, Data dependencies, Temporal dependencies, and Resource dependencies.
Priority ranking	Assign of numerical value that indicate the importance of the user story from the <<STORY CONTRIBUTOR>> point view using AHP method.

Figure 13. Extended user story.

- **Change management:** The extended user story can also provide support for better XP change management. For example, the identification of user story relationships and dependencies will im- prove the developer's ability to specify the impact of change requests on the system. Developers will be able to understand what types of dependencies exist between user stories: a change in <US,i> will generate a change in <US,j>, based on the kind of relationship that has been identified.
- **Process visibility:** The visible process has been characterized as the ability to define contact points between customers and organizations, wh- ere customers are allowed, or even required, to interact with the process activities [20]. The the- ory of visibility claims that organizations can im- prove their competitive advantage by deliberately managing the degree of visibility of their processes. Also, XP supports process visibility by mandating that on-site customers participate dur- ing the XP life cycle. The proposed sub processes allow for process visibility from the development perspective by allowing the developers to trace back every user story to its source and allowing the development team to rank user stories from the <<STORY CONTRIBUTOR>> viewpoint. This will enhance process visibility for both customers and developers.
- **Traceability:** The implementation of traceability requires software developers to identify the deliverables and artifacts of the software life cycle and provide information about the relationships between those deliverables at an early stage of the software project. This can be accomplished once the system has been divided into modules and the information flow (interaction) between these modules has been determined. The extended user story can support traceability by providing early information about the interaction of user stories based on the defined relationships of user stories (*i.e.* logical dependencies, data dependencies, temporal dependencies, and resource dependencies). Moreover, for large software systems that include multiple interrelated software modules, the developers can build a dependency graph that identifies the various types of interactions between the user stories.
- **Accountability:** Software project managers are responsible for ensuring that the software life cycle has been executed in conformity with ISO 9001, even before the software organization is audited by external ISO 9001 auditors. The proposed sub processes will allow software project managers to ensure that the software development activities are being performed in conformity with ISO 9001. For example, at any time in the software life cycle, the project manager can identify the source of user stories by referring to their <<STORY CONTRIBUTOR>> category. Moreover, the software project managers can find documented evidence about the non functional requirements that have been gathered during the software life cycle. The pair-wise matrices and the relationship of user stories can also provide documented evidence for software project managers as to how the user stories interact in the system and the priority ranking for each user story.

REFERENCES

[1] B. Makdee and P. Praneetpolgrang, "Roadmap in the Development of a Quality Model for Thai Software," 3*rd International Conference on Information and Communications Technology*, Cairo, 2005, pp. 829-836. doi:10.1109/ITICT.2005.1609669

[2] L. Vijayasarathy and D. Turk, "Agile Software Development: A Survey of Early Adopters," *Journal of Information Technology Management*, Vol. 19, No. 2, 2008, pp. 1-8.

[3] C. Schindler, "Agile Software Development Methods and Practices in Austrian IT—Industry Results of an Empirical Study," *International Conference on Computational Intelligence for Modeling, Control and Automation*, Vienna, Austria, 2008, pp. 321-326. doi:10.1109/CIMCA.2008.100

[4] A. Espinoza and J. Garbajosa, "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. doi:10.1007/s11334-011-0144-5

[5] M. Qasaimeh and A. Abran, "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. doi:10.1109/SERA.2010.38

[6] G. Wright, "Achieving ISO 9001 Certification for an XP Company," *Lecture Notes in Computer Science, Extreme Programming and Agile Methods*, Agile Universe 2003, New Orleans, August 2003, pp. 43-50.

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

[8] M. Glinz and R. Wieringa, "Stakeholders in Requirements Engineering," *IEEE Software*, Vol. 24, No. 2, 2007, pp.18-21. doi:10.1109/MS.2007.42

[9] A. Abran, K. T. Al-Sarayreh and J. Cuadrado-Gallego, "Measurement Model of Software Requirements Derived from System Maintainability Requirements," 20*nd International Conference on Software Engineering and Knowledge Engineering*, San Francisco, 1-3 July 2010, pp.

153-158.

[10] K. T. Al-Sarayreh and A. Abran, "A Generic Model for the Specification of Software Interface Requirements and Measurement of their Functional Size," 8*th ACIS International Conference on Software Engineering Research Management and Applications*, Montreal, 2010, pp. 217-222. doi:10.1109/SERA.2010.35

[11] H. Tracy, B. Sarah, V. June and W. David, "The Impact of Staff Turnover on Software Projects: The Importance of Understanding What Makes Software Practitioners Tick," *ACM Conference on Computer Personnel Doctoral Consortium and Research*, New York, USA, 2008, pp. 30-39.

[12] L. Karlsson, P. Berander, B. Regnell and C. Wohlin, "Requirements Prioritization: An Experiment on Exhaustive Pair-Wise Comparison Versus Planning Game Partitioning," *Empirical Assessment in Software Engineering Conference*, Keele, 2008, pp. 122-131.

[13] L. Lehtola, M. Kauppinen and S. Kujala, "Requirements Prioritization Challenges in Practice," Springer-Verlag, Berlin Heidelberg, 2004, pp. 497-508.

[14] Th. Bebensee, I. Weerd and S. Brinkkemper, "Binary Priority List for Prioritizing Software Requirements," *Proceedings of the 6th International Working*, *Conference on Requirements Engineering*: *Foundation for Software Quality*, 2010.

[15] S. Fricker and P. Grünbacher, "Negotiation Constellations: Method Selection Framework for Requirements Negotiation," *International. Working Conference on Requirements Engineering*: *Foundation for Software Quality*, 2008.

[16] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, "Agile Software Development Methods: Review and Analysis," Espoo, Finland: Technical Research Centre of Finland, VTT Publications, 2000, pp. 461-478.

[17] E. Forman and M. A. Selly, "Decision by Objectives," George Washington University, 1996.

[18] J. Grandzol, "Improving the Faculty Selection Process in Higher Education: A Case for the Analytic Hierarchy," *Process Association for Institutional Research*, Vol. 6, No. 1, 2005, pp. 1-13.

[19] P. Berander and A. Andrews, "Requirements Prioritization in Engineering and Managing Software Requirements," Springer Verlag, Berlin, 2005, pp. 69-94.

[20] H.-G Yang and B.Vandenbosch, "Visibility as the Basis of a Framework for Identifying Strategic Information Systems," *Journal of Information Technology Management*, Vol. 9, No. 2, 1998, pp. 31-42.

