The Impact of Regulatory Compliance on Agile Software Processes with a Focus on the FDA Guidelines for Medical Device Software

Hossein Mehrfard and Abdelwahab Hamou-Lhadj*
Software Behaviour Analysis Lab
Department of Electrical and Computer Engineering
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, QC, Canada

ABSTRACT

The difficulty of complying with different regulations has become more evident as a large number of regulated businesses are mandated to follow an ever-increasing set of regulations. These regulations often drive significant changes in the way organizations operate to deliver value to their customers. In this paper, we focus on the impact of the Food and Drug Administration (FDA) regulations on agile software development processes, which in many ways can be considered as just another type of organizational processes. We focus in particular on the ability for Extreme Programming (XP) to support FDA requirements. Our findings show that XP fails to meet many of the FDA guidelines for medical device software, which increases the risks of non-compliance for organizations that have adopted XP as their main software process. We believe, however, that the results of this study can lead the work towards designing an extension to XP for FDA regulations.

Keywords: Regulatory Compliance, Organizational Dynamics, Organizational Processes, Software Processes, IT Compliance.

INTRODUCTION

Recently, there has been a significant increase in attention to regulatory compliance and its impact on the way organizations are managed and controlled. This increase is driven by several factors including the recent corporate scandals such as the ones that involved some of the major U.S. organizations (e.g., Enron, WorldCom), the new challenges that Information Technology (IT) pose on protecting and securing sensitive information, and a higher need for business continuity in an ever-changing business world.

As a result, more regulations, laws, standards, and guidelines are introduced every year driving significant changes in the way companies are managed (Hamou-Lhadj & Hamou-Lhadj, 2007). These changes vary in scope and impact ranging from the introduction of new business processes to changes at the governance and strategic level. Hamou-Lhadj et al. characterize these changes in the form of a compliance support framework that can help effective handling of regulatory compliance requirements (Hamou-Lhadj & Hamou-Lhadj, 2007). The framework is composed of four main components: Governance, People, Process, and Technology. The aim of the governance component is to provide the strategic direction that will guide an effective delivery of end-to-end compliance support activities, while ensuring that these activities are aligned with the company's vision and business objectives. The people component revolves around the proper selection, training, and retention of human potential that will operate the compliance support framework. The process component (the topic of this paper) is concerned with the need to adapt existing business processes (or creating new ones) for the handling of compliance requirements at the operational level. Finally, the technology component emphasizes the need for the proper tools and techniques in order to automate the delivery of compliance support activities.

In this paper, we particularly focus on the impact of regulatory compliance on the process by which software systems, used by regulated companies, are developed, maintained, and tested. Software processes can be seen as just another type of organizational processes since they are used by software companies to carry on the development of software products. As such, the paper has the broader objective of looking into the issue of how regulatory compliance impacts organizational processes used by software companies during product development.

More specifically, we target software systems used to control medical devices. These systems are subject to heavy regulations from government organizations to ensure that their design is carried out based on sound software engineering practices. One of the most predominant set of regulations in North-America that regulate the way software systems used to control medical devices should be developed is the Food and Drug Administration (FDA) regulations.

The FDA is a U.S. government agency that protects consumers by enforcing the U.S. Federal Food, Drug, and Cosmetic Act (FDA, 2009b). It regulates more than $1 trillion worth of consumer goods, about 25% of consumer expenditures in the U.S. (FDA, 2009b). The cost of not complying with FDA regulations can be considerably high, which makes its regulations some of the most important ones that should be on the priority list of a strategic compliance management initiative of any organization subject to the FDA laws.

The FDA also regulates the design and use of medical devices. There are several guidelines that have been issued by the FDA (e.g. (FDA, 2002) on how to monitor the manufacturing of safe and reliable medical devices. This also includes the software systems that control these medical devices. Due to complexity and criticality of medical devices, the FDA sets high demands on how to develop software for medical devices. Most of the FDA requirements are directly related to the process activities (e.g., requirement analysis, design, implementation, etc.) used by an organization to develop software. In addition, the FDA expects sufficient level of auditability within the software process itself. In other words, certain aspects of the development life cycle need to be tracked to allow external auditors to assess whether the system is FDA-compliant or not.

However, the requirements imposed by the FDA on the development process are very stringent, and may not be easily attainable. These requirements have been developed to overcome the difficulty of assessing the safety and reliability of software through traditional testing techniques. Additional verification and validation techniques as part of a broader and systematic process have to be applied. As such, The FDA requirements often translate into documenting and following specific guidelines to certify that the system is built, verified, and validated in a systematic manner and according to proven software engineering practices (FDA, 2002).

Therefore, from a risk management perspective, it is important for an organization to understand whether a particular software process meets the FDA requirements or not. Areas where the process fails to meet FDA should also be clearly indicated. In this paper, we chose to study the capability of the Extreme Programming (XP) software process, an agile process, to support these requirements. We choose to focus on XP due to the fact that it embeds most values of the agile movement. As such, we believe that the results presented in this paper can be easily generalized to other agile processes.

This paper is a continuation of previous work in which we discussed how XP can be extended to support one aspect of the FDA requirements which pertains to user studies and understanding user characteristics (Mehrfard et al., 2010). In this paper, we cover all aspects of the FDA requirements ranging from requirement analysis to testing, passing by design and implementation.

More precisely, the main contributions of this paper are as follows:

- We present in detail all the FDA requirements for medical device software. These requirements cover a large spectrum of process activities including requirement analysis, design, implementation, and testing. We believe that this contribution can be used as a reference work for many organizations who struggle to meet FDA requirements due to their ambiguity.
- We study the capability of XP to meet FDA requirements for device medical software. We uncover areas of XP that do not meet FDA requirements. We do this by mapping XP practices and work products to FDA guidelines for each software process activity. Ways to extend XP to meet the FDA requirements can be derived from the mapping table.

FDA GUIDELINES FOR MEDICAL DEVICE SOFTWARE

In this section, we start by presenting our generic approach for mapping a software process to FDA regulations for medical device software. Then, we show the application of this approach for extracting the software process requirements for medical devices from the regulations and guidelines provided by the FDA. We map each of these requirements to XP practices. By doing this, we uncover places where XP lacks support for FDA guidelines.

Mapping Approach

Our approach for mapping a software methodology to FDA guidelines and requirements is shown in Figure 1, and encompasses the following steps:

1. We select among FDA guidelines, the ones that relate to software development.
2. We study these requirements from the software engineering process perspective and present typical software practices and documentation that can help developers follow the guidelines.
3. According to the suggested practices and documentation, we investigate the capabilities of our desired software development methodology, i.e., XP, for supporting these requirements.

Figure 1. The framework for mapping a development process to FDA requirements for medical device software

Step 2 is particularly important since many FDA guidelines and requirements for software development are defined in a way that is too generic to be applied to a development process, which often causes ambiguities for software developers since no specific development methodology can abide by the provided guidelines. For example, the FDA requests the medical device software developers to build safe and reliable software while no specific methods on how safety and reliability should be carried out are explicitly provided by the FDA.

Furthermore, the FDA often uses terms that are not too specific and may have different meanings depending on the context. That is, a single term can be used in more than one field while having many completely different meanings. For instance, "risk analysis" can refer to an activity in both software requirements engineering and project management. This also can cause confusion in the intended meaning of a term making it difficult for development companies to comply with these guidelines as the developers do not know what they specifically have to follow.

One of objectives of this paper is, therefore, to clarify FDA guidelines and relate them common software process engineering concepts. Once this is done, we map each FDA requirement to XP to assess whether it supports it or not.

The FDA guidelines and requirements are grouped into four categories depending on the process activity to which the guidelines apply:

- Requirement analysis
- Design
- Coding and construction
- Testing

Requirements Analysis

The FDA has issues several guidelines on how the requirement gathering and analysis phase should be carried out (FDA, 2002). These guidelines can be further described based on the following sub-phases:

- Requirements Elicitation
- Requirements Evaluation
- Requirements Traceability Analysis

- System and Acceptance Test Plan

Requirements Elicitation

FDA guidelines require a complete documentation that clarifies the software inputs, a software requirements specification (SRS) document, and the expected software outputs. For instance, developers are required to illustrate all ranges, limits and default values which are acceptable as software inputs, a detailed description of the functions of the system (part of the SRS), and the expected results of applying these inputs to the system, i.e., software outputs (FDA, 2002). Documenting software inputs and outputs helps understand the boundary of the system which is necessary for requirements elicitation (Paetsch et al., 2003).

It also important to clearly define the non-functional requirements (NFR) such as performance, reliability, security and the safety features of software. A particular emphasis is put on safety requirements since an unsafe medical device may cause loss of human lives (FDA, 1997; FDA 2002).

In addition, the FDA requires that all communication points between the software in question and other software systems, hardware, and persons be well defined. These communication points are known as interfaces. For instance, the communication point between the software and users is the user interface (FDA, 2002; FDA 2009).

Another key requirement mandated by the FDA is to identify requirements that are related to human factors such as how the system will be used by end users. This requires studying the characteristics of the various users of the system using Human Factors Engineering (HFE) concepts. In particular, the FDA guidelines suggest many activities including observations, interviews, and conducting focus groups to understand HFE requirements (FDA, 1996). To this end, the SRS should describe all the user characteristics based on user's knowledge, ability, expectation, and limitation.

XP provides a number of techniques to elicit requirements. Story cards, for example, are used to capture the software features expected by end users. User stories are written by customers with the help of an XP programmer (who plays the role of an interaction designer). User story is comparable to use cases since both aim to capture the user's needs.

The high involvement of customers in XP is another effective factor for eliciting requirements. After writing user stories, the customer is involved during the "iterations to release" phase and helps break down the user story into multiple tasks. In addition, brainstorming is another elicitation technique that is encouraged in XP with the help of customers and domain experts (Paetsch et al., 2003). This customer involvement also results in better definition of HFE requirements.

However, there is no specific activity in XP that deals with documenting the different interfaces of the system as required by the FDA. As for the functional requirements, they are described in XP by developers during the "iterations to release" phase based on user stories. The non-functional requirements, on the other hand, are dealt with in XP the same way as the functional requirements, i.e., during the "iteration to release" phase. There is a common belief that XP tends to neglect the representation of non-functional requirements during the requirement gathering phase. This is due to the fact that XP tends to focus more on the functionality requested by the customers and less on non-functional requirements. Most of these requirements are only dealt with during the implementation phase. The level of documentation of the requirements in XP is less than what is expected by the FDA. XP, for example, does not require the presence of an SRS. The XP as an agile process tends to maximize team communication in detriment of documentation. The user story is the only document during the requirement elicitation stage used as the requirements document.

Requirements Evaluation

The FDA seeks from organizations to establish ways to evaluate and document requirements through written policies and procedures to resolve any incomplete, ambiguous, inconsistent and conflicting requirements. In addition, the requirements should be evaluated against possible risks. The FDA considers the possibility of applying requirements evaluation in multiple steps (i.e., incrementally) to arrive to clear functional and non-functional requirements (FDA, 2002). Requirement risk analysis focuses on potential risks that may cause the project to fail. The FDA puts a particular emphasis on risks due to misunderstanding of HFE requirements. Risk management is planned and conducted before entering the requirement phase, i.e., at the project level (FDA, 2002). The FDA also recommends a formal review of the requirements before starting extensive software design (FDA, 2002).

From the software engineering perspective, the term requirements evaluation from the FDA perspective carries many similarities with the concepts of requirement analysis and validation found in software requirements engineering. There exist many techniques to evaluate requirements including formal review meetings, risk analysis, requirements inconsistency management, requirements prioritization, evaluation of alternative options in requirements, requirement verification, and prototyping (Paetsch et al., 2003; Sommerville, 2004).

Due to the incremental nature of XP, requirements are evaluated within different iterations and releases. In fact, the product resulting from an XP iteration or release is seen in XP as a prototype that can be evaluated by customers (Abrahamsson et al., 2002). Requirement prioritization, which includes risk analysis, is a constant practice in XP during the planning phase and the "iteration to release" phase. During the planning phase, the customer selects the story cards for the next release based on its business values which is documented in the release plan. He is also responsible for choosing the story cards for each iteration during the "iteration to release" phase and document these stories in an iteration plan. This practice is done with the help of developers during these phases (Larman, 2003).

A general rule for agile processes is that they deal with the most probable risks to the project during the first release and primary iteration of each release (Larman, 2003). However, there is no specific method in XP for analyzing the risks in requirements. Moreover, XP does not support formal review meetings to evaluate requirements.

The FDA guidelines also suggest the presence of a documented mechanism for evaluating requirements. However, XP does not support the presence of any documentation for requirements evaluation.

Requirements Traceability Analysis

Traceability analysis is an essential activity during the entire development process and is required by the FDA regulations. Traceability analysis defines the relationship between the software development artefacts to keep the logical order of these artefacts. This logical order becomes more evident when we are transiting from one development phase to another one (FDA, 2002, 2009).

The FDA puts an emphasis on traceability analysis during the requirement analysis phase by requiring from regulated organizations to establish, in a documented way, the following relationships:

- Software requirements and system requirements (and vice versa)
- Software requirements and the risk analysis results

In addition, the relationship of the requirements with recognized risks coming from the risk analysis results must also be determined. In software requirement engineering, requirement traceability analysis is considered as part of the requirement management activities. Requirement management is usually supported by CASE tools during software development. To achieve this, different traceability matrices are suggested such as source traceability, requirement traceability, design traceability and other traceability matrices based on the expected level of quality (Sommerville, 2004).

XP is completely blind with respect to traceability analysis. There is not practice or process artefact that account for having traceability matrices in any of the development phases.

System and Acceptance Test Plan

At this stage, the FDA requires to develop the system and acceptance test plans. According to ANSI/IEEE standard 829, a test plan is defined as "the documentation of scope, approaches, resources, and schedule of intended testing activities. This document should identify test items, the features to be tested, the testing tasks, responsibilities, and any risks requiring contingency planning" (FDA, 2002, FDA, 2009a).

An acceptance test plan is documented based on a set of acceptance criteria provided by the customer to approve the final product. It is usually created through a close collaboration between customers and developers (FDA, 2002; Pressman, 2003). A system test plan is written with respect to criteria for testing the software product on a specific operating platform to detect performance issues, and situations of stress (FDA, 2002; Pressman, 2003).

In XP, customers are asked to explain the acceptance criteria of the system before each release (Larman, 2003). These criteria are used to generate acceptance tests which are later executed either by the customer or the developer. In XP, the acceptance test plan is known as the customer test document (Larman, 2003). System testing is supported in XP at the end of each release. The XP programmer (who plays the role of an architect) is responsible for performing system testing (Beck & Andres, 2004). A System test plan is written by an architect and put in practice with the help of an XP tester.

Design Phase

The FDA defines the design phase as the process of translating the user requirements into their related logical components to be implemented. Due to the complexity of medical devices, it suggests to have both a high-level and detailed design. The design activity and associated FDA requirements can be further divided into the following sub-phases (FDA, 2002):

- Design for usability
- Software Design Evaluation
- Design Traceability Analysis
- Updating the Test Plans
- Test Design Generation

Design for Usability

The FDA guidelines highlight the importance of usability analysis during the design process to improve human performance in using medical equipments based on their abilities (FDA, 1996). It recognizes that the design for safety of medical devices should take into account human factors. The reason is that according to the FDA Center for Devices and Radiological Health (CDRH), the lack of attention to human factors during product development may lead to errors that can potentially cause serious patient injuries or even death (FDA, 1996). A number of guidelines have been proposed on how to deal with HFE during the design of software including following Human Computer Interface (HCI) guidelines, improving software usability, and performing software design coordinated with hardware design. To improve software usability, the FDA suggests a number of usability tests such as scenario-based testing, and testing the product by users per iteration of software development (FDA, 1996).

In XP, the user interface design (UI) is done during the "iterations to release" phase, but XP does not suggest any guideline for UI design. In addition there is no specific practice in XP that supports usability inspection or any other form of usability testing. However, due to the fact that XP tends to be a user-centered process by working with users throughout the process to obtain constant feedback and that it favours communication with customers, one can assume that XP considers the usable aspect of the final product although at a limited extent (Kowalczykiewicz & Weiss, 2002).

During the exploration phase, there is no specific practice in XP that mandates the use of usability design patterns or evaluate usability at the design or architectural level. To design the software system architecture, XP suggests building a system prototype during the exploration phase to evaluate different architectures. The final architecture is consolidated during the first release (Abrahamsson et al., 2002; Beck & Andres, 2004). There are two practices in XP that affect the design of the system architecture: System metaphors and simple design. System metaphors are shared stories that describe how the system works and the simple design principle aims to make easier to understand each design component (Nord et al., 2004). Despite the existence of the XP interaction designer, there is no explicit guideline in XP with respect to following specific architectural patterns and assessing the usability of the system at the architectural level.

Software Design Evaluation

The software design evaluation is considered as an integral part of the design process. The objective is to validate correctness, completeness, consistency, and maintainability of the design (FDA, 2002). The FDA guidelines define two categories of design evaluation activities: Design review, and design verification and validation (FDA, 1997). During the evaluation of the design, activities such as analysis of control flow, data flow, complexity, timing, memory allocation, and criticality analysis should be supported (FDA, 2002). Moreover, the FDA puts an emphasis on analysing component interfaces during the design evaluation to ensure that all the defined interfaces in the requirement phase suit well the proposed design.

Design review meetings are required by the FDA to support the fact that a design inspection has taken place. The focus of these meetings is to identify different concerns of software design and their potential side-effects, possible solutions, and the corresponding corrective actions in software design. During these meetings, designers present their design to the design reviewers. There are three types of design review: preliminary design review, critical design review, and system design review. The process is conducted in an iterative manner until potential problems are explored and solutions have been proposed (FDA, 1997, FDA 2002, FDA 2009).

The FDA defines design verification as a confirmation by examination that a specific requirement has been fulfilled. This requires documenting the design verification process. The FDA suggests using some verification techniques such as fault tree analysis, and worst case analysis. In design control document, the FDA references the ISO 9001:1994 standard, where activities such as prototype evaluation, demonstration, simulation and comparing the design with other similar proven designs are considered as software design verification activities (FDA, 1997; ISO, 2000).

Design validation is the examination that a design responds to user needs or the intended use of the product. The FDA requires documenting the design validation process. Design validation should be executed under actual or simulated use condition. It also needs to follow a successful verification to ensure that each user requirements is fulfilled. For this purpose, the FDA requires to provide a validation plan, validation methods and validation review. Some of the design validation techniques recommended by the FDA include analysis and inspection methods, compilation of relevant scientific literature, and provision of historical evidence that similar designs are clinically safe (FDA, 1997).

The design in XP is kept as simple as possible and is informally documented. XP does not account for activities that deal with analysis of communication links among the system interfaces, control flow, and data flow as required by FDA. Also, the design review in XP is significantly different from the FDA design review, because there is no formal design document in XP to review. Instead of having formal meetings, design review is limited to pair programming. It is recognized that an iterative cycle of pair programming provides continuous analysis and review of the design to improve and simplify it (Abrahamsson et al., 2002). The development teams using pair programming have reported good quality of design with fewer lines of code as they worked together on both the design and the implementation (Cockburn & Williams, 2001). However, XP does not support any specific practices which are required by the FDA for design verification and validation.

Design Traceability Analysis

As mentioned in the requirement phase, the FDA requires traceability analysis throughout the entire development process. In the design phase, tractability analysis is conducted to verify that the entire design components are traceable to the software requirements and that all requirements can be mapped to a software design (FDA, 2002). For this reason, there should be a design traceability matrix which relates software requirements documents to design specification. There is overlap between design traceability analysis and design verification. Both of them put an emphasis on conformance of design with the requirements while traceability aims only to show the relations between the requirements and the design. XP does not support any traceability matrix during the design to show this relation.

Updating the Test Plans

The FDA requires updating existing test plans by generating module and integration test plans during the design phase. A module test plan should be created to test specific units of the system. On the other hand, the integration test plan should be updated to test the flow of control and data between program units (FDA, 2002).

In addition, the acceptance and system test plans, which are created during the requirements phase, should also be updated considering HFE criteria defined during the software requirements and design phases.

Both unit testing and integration testing are performed within iteration in XP. But there is no mention in XP about having a specific test plan. The nature of testing in XP is explained in the testing section.

Test Design Generation

After preparing the test plan, the FDA requires from the development team to start generating test procedures and test cases for unit, integration, system and acceptance test based on the results of requirement and design phases. These tests should be completed and finally executed during the coding and test phases (FDA, 2002).

In XP, unit tests cases are generated after the design within an iteration. In addition, acceptance tests which are already generated by customers before the iteration starts are automated by an XP tester to be executed after an iteration is completed. Furthermore, integration tests executed by an XP programmer (who assumes the role of an integrator) at the end of an iteration before acceptance testing so as to integrate the pieces of the code developed during the iteration (Abrahamsson et al., 2002; Larman, 2003).

System test is done per release and supported by an XP programmer (architect). As XP architect is the person who is charge of designing the system structure. He is responsible for developing system test cases during multiple iterations (Beck & Andres, 2004). These activities show that XP supports developing test cases before coding as requested by the FDA. However, XP does not require documenting the test procedures.

Coding and Construction Activities

In this phase, the detailed design specification should be implemented as a computer program. The construction is done either by directly start programming or assembling code components. The selection of the programming language and builder tools (i.e. assembler, linker or compiler) should be based on the availability of debugging and testing tools (FDA, 2002). The coding and construction activities regulated by the FDA are:

- Source Code Evaluation
- Source Code Documentation Evaluation
- Code Traceability Analysis
- Source code Interface Analysis

Source Code Evaluation

Source code is evaluated before compilation to make sure that it follows design specification and coding standards. The FDA recommends using desk checking techniques to evaluate the software code. Desk checking methods include code audit, code inspection, code walkthrough, and code review (FDA, 2009). During code inspection, the author of the code explains statement by statement what the code is supposed to do in a meeting convened to analyze the program logic and its conformance to coding standards. During code walkthrough, developers manually trace the source code with small set of test cases. Code audit is a review of the source code by an independent person, or team to make sure that the source code follows software design and programming standards. A code review consists of organizing meetings, where the software code is presented to project personnel, managers, and customers for feedback and approval (FDA, 2002, FDA 2009).

XP does not support any of the formal desk checking practices required by FDA. Instead, XP claims that pair programming is more successful than any inspection and formal review methods. According to experimental studies, the pair programming technique has been shown to be effective in uncovering errors in the code while programming, saving costs since errors are discovered before compilation (Cockburn & Williams, 2001). Therefore, it is reasonable to assume that pair programming satisfies FDA requirements with respect to source code evaluation without the need of having formal desk checking techniques.

Source Code Documentation Evaluation

The FDA requires documenting the coding and the construction process. In most software projects, the commented code and the generated html or text files from the source code by tools are considered sufficient for documenting the code. However, the FDA requires documentation for each implemented module or function to show its agreement with coding standards and quality policies, defined within the organization. In addition, the existing errors after coding and construction have to be documented. Moreover, the whole process of compilation should be documented (errors found, solutions and unsolved errors and warnings) (FDA, 2002).
XP does not support the production of any documentation needed to satisfy the FDA recommendations such as documenting modules, errors, the compilation process, and the used tools and techniques.

Code Traceability Analysis

The FDA requires having a traceability matrix to show the relation between the source code modules and the design specification and vice versa (FDA, 2002). In addition, the traceability matrices to show the relation of the test cases and the source code as well as the test cases and the design specification is also needed (FDA, 2002).

XP does not support the development of traceability matrices from source code-to-design, from test cases-to-source code, from test cases-to-design, from test cases-to-risk analysis, and from source code-to-risk analysis as required by the FDA.

Source code Interface Analysis

The implementation of the interfaces between the system modules should be clearly specified in the source code to ensure that the implemented communication links are well integrated with the software implementation. This aims to increase the safety of the final software product.
There is no specific guideline in XP on how to analyze different interfaces of the subsystems for implementing and integrating the various parts of the code.

Test Generation

Besides the test procedures and test cases created to test the software design, the new test cases and their corresponding test procedures are generating based on the implementation. The new test cases can be unit, integration, acceptance, and system testing.

There are two types of test cases in XP that are possible to map to the design: unit and integration test cases. XP follows the test-driven development method (TDD), in which test cases are design before coding starts. Therefore, unit test cases are generated as the result of TDD after the design and before coding. Integration test cases are generated at the end of an iteration once a new piece of code is adding to the collective codebase. These two test cases have the potential to map to the code and the design. For the acceptance test, it is not possible to map it to elements of the design or code. In addition, dividing XP into small iterations and having simple design enables developers to relate the elements of a design to code and therefore create a code-to-design traceability matrix. System test cases are generated during the whole release to be executed during the productionizing phase.

The Testing Phase

The FDA focuses extensively on software testing during the development process to ensure the reliability and safety of the software product. It lists a number of software testing principles to examine software effectively such as the importance of what is to be tested rather than how to apply the test, anticipating the results expected from the testing process, ensuring the independence of the testing process from coding, and documenting the tests. The FDA guidelines highlight four types of testing activities that need to be supported: structural testing, functional testing, statistical testing, and regression testing (FDA, 2002; FDA, 2009a).

Structural or white box testing evaluates the internal code structure. The amount of structural test coverage is defined based on common metrics such as coverage of statements, branches, loops, conditions, and data flows (FDA, 2002).

Functional testing is a black box testing technique which is conducted to evaluate the program functionality and program interfaces. The FDA divides functional testing into four different types: normal case, output forcing, robustness, and combinations of inputs (FDA, 2002).

Regression testing is another type of testing technique to manage the changes during the software development life cycle. Regression testing ensures that changes to the system do not negatively impact the other parts of the system (FDA, 2002; FDA 2009).

Test Documentation

Documenting test activities is an important concern for the FDA during the testing process. The documents that the FDA requires during the testing process include a test plan, test procedures, test cases, test reports, and test logs (ANSI/IEEE, 1983; FDA, 2002; FDA 2009).

Test plan, as defined in the requirement phase, should be created early in the process to identify the testing tasks during each development stage. A test procedures document is generated from the test plan. It contains instructions about each test on how to setup the test and evaluate the results. Test cases are designed and implemented depending on the type of testing (i.e. structural, functional, statistic, and regression). This document should identify system inputs, expected results, and a set of execution conditions for test. A test log is defined as a record of the test execution (FDA, 2009). For instance, all detected errors during test execution should be logged. Once test execution is finished, the direction and results of the test should be recorded (FDA, 2002; FDA 2009).

Except a limited number of documents that are created to reach a running product, XP attempts to minimize the amount of efforts on documentation as one of its values. Among the testing documents created in XP, the test cases exist usually in a form that is readable by automatic testing tools.

Test Execution

The FDA requires to perform the following testing activities: unit testing, integration testing, system testing and acceptance (FDA, 2002).

In unit testing, the program is divided to smaller components (modules). Then, the structure and functionality of each component is examined early in program testing (FDA, 2002). Integration testing is one level higher than unit testing. Integration testing concentrates on the flow of control and data between units (FDA, 2002). The system level testing, all aspects of functionality and performance of the software product are tested. This test is done on working software products and developers should consider the requirements that exist at the level of the operating environment. The FDA highlights some aspects of software to examine during system testing such as performance issues, responses to stress conditions, security features, effectiveness of software recovery, HFE and usability, accuracy of documentation, and compatibility with other software products (FDA, 2002).

Finally, for user site testing, the FDA requires the conduct of user site testing as the last step of the testing activity of the software product. It defines user site testing as any testing through actual or simulated use of software as the part of installed system configuration at the user's site. As mentioned in the requirement phase, the FDA assumes that user site testing is the same as the installation testing, beta testing, site validation, installation verification, and user acceptance test.

XP supports unit testing. Unit tests are generated in each iteration based on the design to test subsequent code. Then, after writing the code, the unit tests are executed to find probable faults (Maximilien & Laurie, 2003). In addition, XP supports integration testing by continuous integration whenever new code is written, added to the collective codebase, and unit tested (Abrahamsson et al., 2002).

There is system test in XP with a scope limited to testing the system structure. An XP programmer (architect sub-role) is responsible for providing the system tests to examine the architecture during the productionizing phase for each release (Beck & Andres, 2004).

Test Traceability Analysis

The FDA requires several traceability matrices to link unit tests to detailed design, integration tests to high-level design, and system tests to software requirements (FDA, 2002).
There is no support in XP for traceability matrices from unit tests to detailed design, from integration tests to high-level design, and from system tests to software requirements. But developing such traceability matrix for unit test to detailed design should be straightforward since the unit tests are developed based on the design and before coding starts.

Summary

Tables 1 to 4 summarize for each process activity the recommended practices according to the FDA guidelines for medical device software. The tables also show the documentation that is required to be generated throughout the process for the system to be FDA compliant. Areas where XP fails to meet the FDA requirements as shown in bold. As shown in these tables, many projects that adopt XP run high-risk of non-compliance with the FDA regulations because of the inability of XP to meet several FDA required practices.

To address this issue, there is a need to extend XP for projects that requires FDA compliance by explicitly addressing the missing requirements. This extension will require adding new roles, practices and work products (documentation). However, we believe that any extension to XP should consider the following points:

- The XP values should not be affected by the extension. These include increased communication among team members, pair programming, collective ownership of the code, rapid iteration, light-weight documentation, etc. These practices have been shown to be useful in many software projects.
- There should be a compromise between keeping the process agile and meeting the FDA requirements. This is particularly difficult to achieve since XP roles are defined in such a way that optimizes the time it takes to produce a release. Adding new practices to XP may XP time to market norms. Tradeoffs that balance agility and auditability need to be investigated.

Table 1. Mapping between FDA and XP – Requirement Phase

Requirement Phase	**FDA Recommended Practices**	**FDA Required Documentation**	**XP Practices**	**XP Documentation**
Requirements Elicitation	Interviews, use cases, observation and social analysis, focus group, brainstorming, and prototyping	Software Requirements Specification (SRS)	User story card writing, High customer involvement, Eliciting requirements in number of iterations	**User stories are the only documentation of requirements in XP**
Requirements Evaluation	Formal review meetings, risk analysis, requirements inconsistency management, requirements prioritization, evaluation of alternative options in requirements, requirement verification, prototyping, requirements risk analysis	Result of the evaluation needs to be documented	System prototype, Building software functionality in number of iterations, Handling possible risks early in the development process	**The process of evaluating requirements is not documented in XP**
Requirements Traceability Analysis	Create traceability matrices	Software requirements and system requirements traceability matrix, software requirements and the risk analysis result traceability matrix	**There is no practice in XP for traceability analysis**	**There is no documentation that relates different artefacts**
Test Plan	Working on acceptance and system test plans	Acceptance test plan, system test plan	Customers are involved in the writing of acceptance tests, The XP architect is responsible for creating a system test plan	Both system and acceptance test plans are documented

Table 2. Mapping between FDA and XP – Design Phase

Design Phase	**FDA Recommended Practices**	**FDA Required Documentation**	**XP Practices**	**XP Documentation**
Design for usability	Usability testing, usability inspection, usability inquiry, usability design patterns, scenario-based assessment of architecture	Documentation on design decisions that relate to making the system more usable	System prototyping to obtain feedback for the end users.	The prototype itself is the only evidence that prototyping took place
Software Design Evaluation	prototype evaluation, demonstration, simulation, comparing the design with other similar proven designs, analysis and inspection methods, compilation of relevant scientific literature, provision of historical evidence that similar designs are clinically safe	design review document, design verification document, design validation document, Software Design Specification (SDS)	Pair Programming, Refactoring	**Design evaluation is not documented**
Design Traceability Analysis	Create traceability matrices	Requirement-to-design traceability matrix	**There is no practice in XP for traceability analysis**	**There is no documentation that relates different artefacts**
Update Test Plan	Working on unit and integration test plans	Unit and integration test plans	Updating test plans taking into account design elements	Unit and integration test plans
Test Design Generation	Generating test cases for unit, integration, acceptance and system testing	Test cases and test procedures	Generating test cases for unit, integration, acceptance and system testing	Unit test cases, integration test cases, acceptance test cases, and system test cases

Table 3. Mapping between FDA and XP – Coding Phase

Coding Phase	**FDA Recommended Practices**	**FDA Required Documentation**	**XP Practices**	**XP Documentation**
Source Code Evaluation	code audit, code inspection, code walkthrough, code review	Documentation that shows that code has been reviewed	Pair programming	**No documentation is produced**
Source Code Documentation Evaluation	Although there is no specific practices defined in the FDA guidelines, the FDA requires that the source code be documented and that the evaluation of this documentation should be performed	Source code document	**There is not support for this activity**	**Since the code does not need to be documented, the evaluation of the documentation does not apply.**
Code Traceability Analysis	Create traceability matrices	Traceability matrices for: source code to design specification, test cases to source code, test cases to design specification, test cases to risk analysis results, source code to risk analysis results	**There is no practice in XP for traceability analysis**	**There is no documentation that relates different artefacts**
Source code Interface Analysis	Interface checking	Documents that show that interfaces between the system components have bee verified	**There is no support for this activity**	**No documentation is created**
Test Generation	Updating test cases for unit, integration, acceptance and system testing	Document that describes test cases and test procedures	Updating test cases for unit, integration, acceptance and system testing	Unit test case, Integration test case, Acceptance test case, System test case

Table 4. Mapping between FDA and XP – Testing Phase

Testing Phase	**FDA Recommended Practices**	**FDA Required Documentation**	**XP Practices**	**XP Documentation**
Test Documentation	The FDA requires documenting the testing process	Test plan, test procedures, test cases, test report, and test logs	XP is a test-driven approach and there are many practices and roles that are dedicated to testing	Test plans, test procedures, test logs, and test cases
Test Execution	Execution of unit tests, integration tests, system tests, and user site testing	Document that describes the results of executing the tests	Tests are executed	Test logs are kept for debugging purposes
Test Traceability Analysis	Create traceability matrices	Traceability matrices for: unit tests to detailed design, integration tests to high level design, and system tests to software requirements	**There is no practice in XP for traceability analysis**	**There is no documentation that relates different artefacts**

CONCLUSION AND FUTURE DIRECTIONS

In this paper, we discussed the changes that regulatory compliance can have on software processes with a particular focus on agile practices such as XP. We particularly looked into the requirements imposed by the FDA on the way medical device software is built, tested, and maintained. Some of the contributions of the paper include extracting the requirements that FDA imposes on software processes by focusing on process activities, and analyzing the capability for XP (an agile process) to support FDA requirements. We uncovered areas where XP fails to support many of the FDA key requirements. We suggested that one way of meeting the FDA requirements is to extend XP by adding new roles, practices, and artefacts. However, this extension should be carefully designed so as to (1) minimize the impact on the XP values, and (2) balance the agility of XP with the need to satisfy the FDA requirements.

The immediate future work would be to investigate ways to extend XP to meet the FDA requirement and experiment with this extension in practice. We anticipate that designing an extension to XP while keeping its agility could be a challenging task. Another future direction would be to apply the techniques presented in this paper to other agile processes such as Scrum, FDD, and others.

REFERENCES

Abdeen, M. M., Kahl, W., & Maibaum, T. (2007). FDA: Between Process and Product Evaluation. In J. M. Goldman, I. Lee, O. Sokolsky, S. Whitehead (Ed.), *The joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability* (pp. 181-186). Cambridge, MA, USA: IEEE Computer Society Press.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile Software Development Methods: Review and Analysis*. Kajaan, Finland: VTT Publications.

ANSI/IEEE. (1983). IEEE Standard for Software Test Documentation. Retrieved August 9, 2009, from http://standards.ieee.org/reading/ieee/std_public/description/se/829-1983_desc.html

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional.

Cockburn, A., & Williams, L. (2001). The Costs and Benefits of Pair Programming. In G. Succi, M. Marchesi (Ed.), *Extreme Programming Examined* (pp. 223-248). Boston, MA, USA: Addison-Wesley Longman Publishing.

FDA. (1996). Do It by Design: An Introduction to Human Factors in Medical Devices. Retrieved

September 6, 2009, from http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM095061.pdf.

FDA. (1997). Design Control Guidance for Medical Device Manufacturers. Retrieved August 10, 2009, from http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM070642.pdf.

FDA. (2002). General Principles of Software Validation; Final Guidance for Industry and FDA Staff. Retrieved August 3, 2009, from http://www.pacontrol.com/download/General-Principles-of-Software-Validation.pdf.

FDA. (2009a). Glossary of Computer Systems Software Development Terminology. Retrieved March 19, 2010, from http://www.fda.gov/iceci/inspections/inspectionguides/ucm074875.htm

FDA. (2009b). Guidelines, Regulatory Information. Retrieved April 3, 2010, from http://www.fda.gov/RegulatoryInformation/Legislation/default.htm

Forsström, J. (1997). Why Certification of Medical Software Would Be Useful? *International Journal of Medical Informatics, 47*(3), 143-151.

Hamou-Lhadj, A.K, & Hamou-Lhadj, A (2007). Towards a Compliance Support Framework for Global Software Companies. *The IASTED International Conference on Software Engineering and Applications* (pp 31-36). Cambridge, MA, USA: ACTA Press.

Kowalczykiewicz, K., & Weiss, D. (2002). Traceability: Taming uncontrolled change in software development. *Foundations of Computing and Decision Sciences, 27*(4), 239-248.

Larman, C. (2003). *Agile and Iterative Development: A Manager's Guide.* Addison-Wesley Professional.

Maximilien, E. M., & Laurie, W. (2003). Assessing test-driven development at IBM. In L. Clarke, L. Dillon, W. Tichy (Ed.), *International Conference on Software Engineering* (pp. 564 - 569). Portland, Oregon, USA: IEEE Computer Society Press.

Mehrfard, H., Pirzadeh, H., & Hamou-Lhadj, A. (2010). Investigating the Capability of Agile Processes to Support Life-Science Regulations: The Case of XP and FDA Regulations with a Focus on Human Factor Requirements. In R. Lee, O. Ormandjieva, A. Abran, C. Constantinides (Ed.), *Software Engineering Research, Management and Applications* (pp. 241-255), Berlin / Heidelberg, Germany: Springer Studies in Computational Intelligence.

Nord, R. L., Tomayko, J. E., & Wojcik, R. (2004). *Integrating Software-Architecture-Centric Methods into Extreme Programming (XP)* (Tech. Rep. CMU/SEI-2004-TN-036). Pittsburgh PA: Carnegie-Mellon University, Software Engineering Institute.

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements Engineering and Agile Software Development. *The International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (pp. 308 - 313). Linz, Austria: IEEE Computer Society Press.

Pressman, R. (2003). *Software Engineering: A Practitioner's Approach (6th Edition).* McGraw-Hill.

Sommerville, I. (2004). *Software Engineering (7th Edition)*. Pearson Addison Wesley.