Journal of Software Engineering and Applications, **2013, 6, 1-13**
http://dx.doi.org/10.4236/jsea.2013.69A001 Published Online September 2013 (http://www.scirp.org/journal/jsea)

Process of Designing Robust, Dependable, Safe and Secure Software for Medical Devices: Point of Care Testing Device as a Case Study

Sivanesan Tulasidas[1], Ruth Mackay[1], Pascal Craw[1], Chris Hudson[1], Voula Gkatzidou[2], Wamadeva Balachandran[1]

[1]Centre for Electronic Systems Research, School of Engineering & Design, Brunel University, Uxbridge, UK; [2]Information Systems and Computing, Brunel University, Uxbridge, UK.
Email: sivanesan.tulasidas@brunel.ac.uk, wamadeva.balachandran@brunel.ac.uk

Received June 25th, 2013; revised July 26th, 2013; accepted August 3rd, 2013

ABSTRACT

This paper presents a holistic methodology for the design of medical device software, which encompasses a new way of eliciting requirements, system design process, security design guideline, cloud architecture design, combinatorial testing process and agile project management. The paper uses point of care diagnostics as a case study where the software and hardware must be robust, reliable to provide accurate diagnosis of diseases. As software and software intensive systems are becoming increasingly complex, the impact of failures can lead to significant property damage, or damage to the environment. Within the medical diagnostic device software domain such failures can result in misdiagnosis leading to clinical complications and in some cases death. Software faults can arise due to the interaction among the software, the hardware, third party software and the operating environment. Unanticipated environmental changes and latent coding errors lead to operation faults despite the fact that usually a significant effort has been expended in the design, verification and validation of the software system. It is becoming increasingly more apparent that one needs to adopt different approaches, which will guarantee that a complex software system meets all safety, security, and reliability requirements, in addition to complying with standards such as IEC 62304. There are many initiatives taken to develop safety and security critical systems, at different development phases and in different contexts, ranging from infrastructure design to device design. Different approaches are implemented to design error free software for safety critical systems. By adopting the strategies and processes presented in this paper one can overcome the challenges in developing error free software for medical devices (or safety critical systems).

Keywords: Point of Care Testing; System Architecture; Safety Critical Systems; Software Development

1. Introduction

In the medical devices industry, software system failures can cost lives and result in fatal consequences. Software faults can arise due to the interaction among the software, the hardware, and the operating environment. Unexpected environmental changes lead to software abnormalities that may have significant impact on the overall success of the system operation. Latent coding errors can surface at any time during the system operation and trigger faults in despite of significant effort having been expended in verification and validation (V&V) of the software system. Extra efforts and spending considerable time in V&V are not enough to guarantee that a complex software system meets all safety, security, and reliability requirements [1].

Medical devices are particularly partial to interoperability issues due to incompatible data formats that lead to fatal consequences in the interconnected network environment [2]. Point of Care testing (POCT) has been considered as a case example because it meets the definition of the Safety Critical System (SCS). POCT is becoming an increasingly popular method over standard laboratory testing for diagnosis of infectious and genetic diseases. This testing methodology has shown many advantages such as diagnosis time reduction, cost reduction, portability and better process control due to the automated nature of POCT in a hand-held or benchtop device. This reduces the risk of human error. The decrease in diagno-

sis time and rapid onset of treatment has been shown to improve patient outcome in emergency settings [3]. Diagnostic errors such as false positive results still occur, these result in poor patient outcomes due to unnecessary treatment or a delay in therapy which can result in a poor clinical outcome [4].

The POC device considered in this case uses the automated process control for measuring parameters pertained to medical diagnosis and it is used in a network connected environment, it meets the basic definition of the SCS. In this context, this paper shows how inherent complexity of software development process cycle for the POC system, can be project managed, designed and verified, using the methodologies, tools and system design concepts that are followed in prototyping the POC device. These software system developing processes will help eliminate (or minimize) fatal consequences of errors made in software coding.

In this paper, a holistic strategy of developing software (SW) for POC devices has been described. The paper contents have been divided into different sections. Each section addresses software development process of SCS with different aspects of the development cycle to produce the product. In Section 2, a requirement eliciting methodology called Value Based Requirements Gathering is described. This process eliminates ambiguity in requirements. This is essential for developing any SCS. In Section 3, a system identification process that is used in the development of the POC system end-to-end architecture is described. The system identification process will provide a basic abstraction view of the system that needs to be designed and to be implemented. In Section 4, the role of security design in the development of the SCS is explained. In the Section 5, operational scenarios used have been discussed. In Section 6, a cloud based architectural strategy for POCT devices has been presented. In Section 7, justification and benefits for adopting recommendations from IEC 62,304 at the early stage of the development cycle are given. In Section 8, a process generating testing strategy based on combinatorial design methodology has been described. This methodology shows how less number of test scenarios guarantees 99% functional coverage as being opposed to spending time and resources on exhaustive testing. In Section 9, a recommendation for project management methodology that will potentially help eliminate software errors is discussed. Finally in Section 10, conclusions drawn are presented.

2. Pyramid Requirement of Eliciting Process

2.1. Issues with Current Process

The FDA has analysed 3140 medical device recalls between 1992 and 1998, these reveal that 242 of them (7.7%) are traceable to software failures and of those software related recalls, 192 (or 79%) were linked to software defects that were introduced when changes were made to the software after its initial distribution [5]. The requirements were not fully understood by the downstream development teams when changes were made.

Figure 1 shows the traceability relationship between requirements and the downstream teams; design, software coding, testing and deployment teams. Also the figure shows those portions of system with errors are increasing as development move from requirement phase to implementation phase. Note that the errors portion at the design and SW coding level are more than the errors portion at the requirement level. Less error at the requirement level will have a cascade effect at the last stage.

It is very clear that by controlling and eliminating the errors at the requirement phase can be minimized at the SW coding level. At any given phase of development, the domain experts concentrate on their main deliverables and have limited awareness of information from previous phases.

2.2. Pyramid Requirement Model

In order to eliminate the errors that originate from the requirement phase, a methodology is presented here. It is known as “Problem Pyramid” modeling and is amalgamated prior to creating any requirements. The requirements become unclear because they are not tied to key measures. The pyramid modeling provides a way to associate the key measures that the requirement needs to fulfill as shown in **Figure 2**. Each block is identified with a flow, which will help to create a meaningful and error free requirement. The requirement can be refined methodically by having the iterative refinement process (1: 2: 3: 4: 5) until all the stakeholders are satisfied.

It is very unlikely that a SW engineer who focuses on the “Program and Test” phase will detect an error propa-

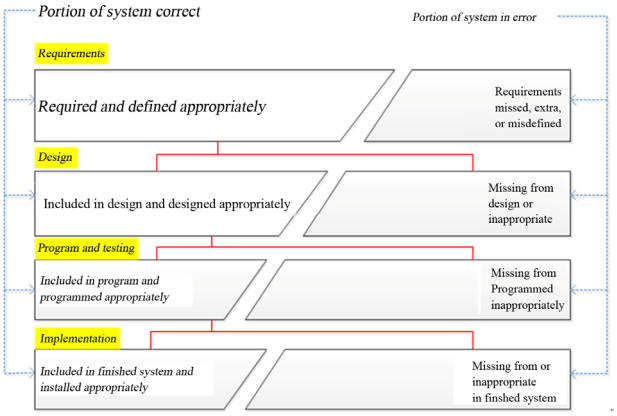

Figure 1. Error sources by development phases (layers) [6].

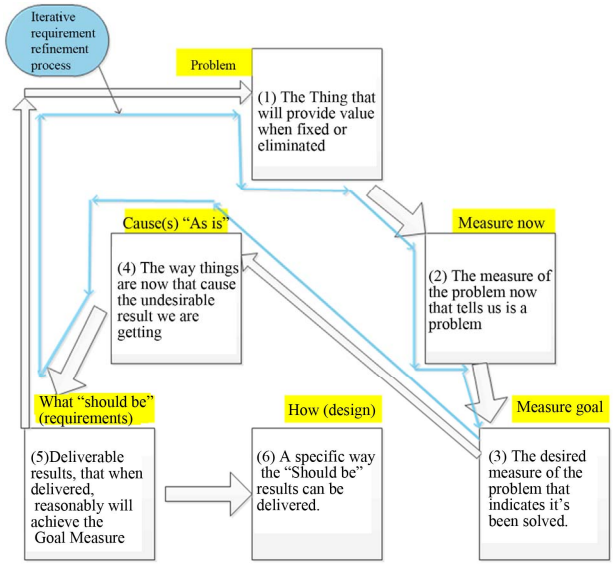

Figure 2. Problem-Pyramid [6].

gating from the "Requirement" phase. In order to have error free requirements, a methodology is presented which gives more importance to technical and business values that the requirement brings to the device development [6].

2.3. Requirement Pyramid Model Example

An example of formulating an error free requirement for the POC device is shown in **Table 1**.

In this example, the items (1) and (2) represent the problem domain; item (3) states the desired outcome; item (4) identifies the problem causes; item (5) states the requirement to accomplish the desired goal; item (6) states the design view of how the requirement can be implemented. With this kind of information it is very hard to make SW related coding errors because the requirement is tied into a measureable goal.

Also note that this process increases visibility and access to actual requirements for the downstream team (SW development and verification teams). This will encourage all the teams who are in the various vertical layers to collaborate effectively to produce error free SW for the medical devices.

3. System Design Process and Techniques

3.1. Basic Building Blocks Identification

Analysis of medical device recall reports in the FDA database in 2005-2006 show 109 software-related recall cases. The main recall reason for high risk device is design defect. Though the ratio of Class I devices with high risk is declining in 2006 compared to 2005 [7], the FDA data shows the repeatable occurrence of device recalls. There are directives and legislations have been proposed for software vendors for preventing product failure [8]. The design defects are the results of failure to understand the intent of the requirements by the design team and the lack of architectural view of the end-to-end system. By formulating the basic building blocks of the system that is being developed will help to design a fundamental DNA of the system architecture. Once the basic building blocks are understood they can be organized or connected in such a way that a service (functionality) can be implemented to meet user requirements.

The basic building block view (BBBV) of the POCT end2end system consists of P-Node, G-Node and P-Cloud (**Figure 3**). The P-Node is the representation of the POCT device. The G-Node is the representation of the gateway entity, which can be a smart-phone or a laptop which is used to send measurement data to database server. The P-cloud is the representation of the secure private data cloud which has special interface access requirements such as UK's NHS N3 [9] cloud infrastructure and Canadian infoway [10].

Table 1. Application of pyramid process.

(1)	Problem:	Lab user needs to input appropriate volume count when configuring the POC device
(2)	Current measures:	Multiple of 4 digit numbers were entered. Unable to enter 2 digits numbers
(3)	Goal measures:	Must enter only two digit number representing either mm^3 or micro liter
(4)	Causes:	Keypad entry does not differentiate entry for the volume and the process ID. One keypad input function is used for both entering the volume and process ID
(5)	Should be:	Keypad should differentiate the entry for volume and process ID
(6)	Design:	Two separate keypads functions for inputting volume and process ID

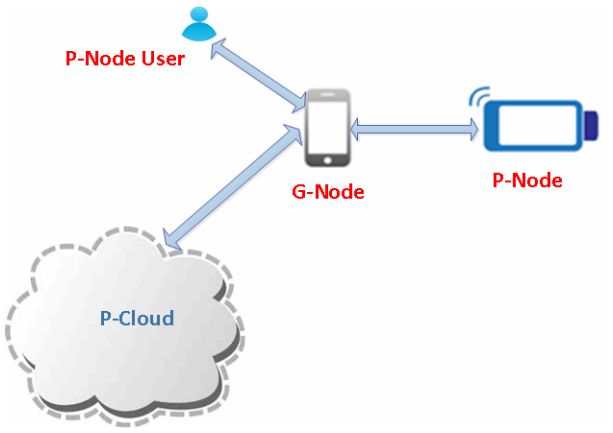

Figure 3. Basic build blocks of POC system.

In general there are two main root classes of mode of operation of the POC system. When a POCT request is initiated from the G-Node to P-Node, the mode of operation is defined as user functional mode. When the data is sent from the P-Node to the P-Cloud after the test, the mode of operation is defined as reporting mode.

User Functional (Control) mode

Request from G-Node and response from P-Node

Reporting mode

Data sent only from P-Node to G-Node

3.2. Isolating System Module Responsibilities

Once the basic building blocks have been identified, isolate the functionalities and create functionality boundaries *i.e.*, assigning responsibilities to system components or architectural modules will create responsibility zones within the system architecture. A similar idea is encouraged in USE-CASE MAPS methodology [11] which is a popular system modeling process for complex architecture of interconnected systems.

3.3. Creating Loosely Coupled Modules

The loosely coupled system has many advantages in service oriented architecture [12-14]. This helps to implement the system independently because of the nature of loosely coupled system.

In practical terms, this kind of system architecture allowed more stable and dependable products as the faults or fatal errors in one sub system will not impact the other system components. There is a vital need for these loosely coupled systems to interact and produce a desired end goal based on the system requirements. This communication between the loosely coupled system components should be of asynchronous messaging to maintain non-interdependency. The following paragraphs explain how this is implemented in the POCT architecture.

The loosely coupling implementation in POCT devices is used both in software and hardware components. I2C interface (or similar interface) is used to connect these entities to share common data. This will provide the following advantages:

- Independent systems operation (*i.e.*, each module can function on its own)
- The system has high fault tolerance or the faults can be contained within the sub system boundaries
- The safety classifications (system portioning) can be applied easily as per the IEC 62304, the standard for medical device software life cycle process.
- The system can be easily validated and verified as sub-system units and single End-2-End system
- This type of system design paves the way to expand system (encouraging scalability) whenever requirements changes or the POCT needs to interface with 3rd party systems.

When designing mission critical systems such as medical devices, the couplings (HW based or SW based) need to be minimized in order to eliminate be single point failures, which will impact whole system. This includes assigning hardware architectures to multiple HW modules or platforms.

Figure 4, shows the end2end topology of the POCT system which is being developed. The principle of isolating the functionalities and assigning the functionalities to multiple HW modules is followed in develo ping the topology. For e.g., the touch screen module and the control system module are separated and connected via “Link 3” as shown in **Figure 4**.

The other important aspect is the identification of communication links, even within the HW components (e.g. the “Link 2”: connectivity between the SD card and Ethernet module). This will help to verify the functional correctness of these links during the system integration tests, which are done by verification and validation teams.

3.4. Actuation Channels and Actuation Confirmation channels

Cross verification of activation of control outputs signals that drives external peripherals (e.g. coils and actuators) need to be verified for all the outgoing actuations scenarios. This kind of design is used in automated train control systems. There are two groups of control channels, the actuation channel (ACH) and actuation confirmation channel (ACCH). A general system construct is shown in **Figure 5** below.

The “actuation channel” is responsible for activating

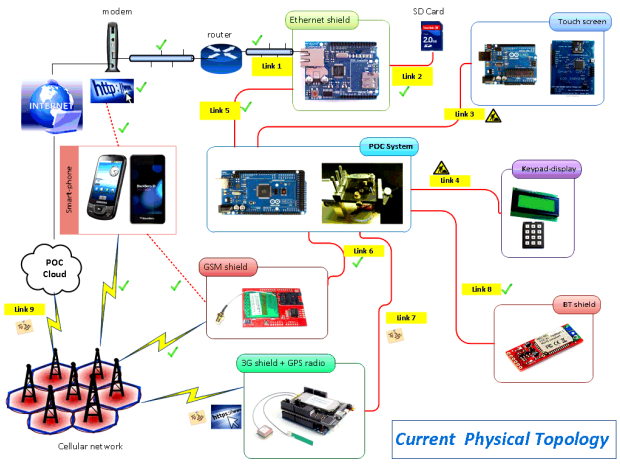

Figure 4. End 2 End Topology.

(e.g. setting a voltage to an actuator) a peripheral. The "actuation confirmation channel" is responsible for verifying if the actuation was successful. In the POCT system design, this will be accomplished by an actuation control word which is a 16 bits variable and each bit (or group of bits) corresponds to a peripheral. This is because some of the peripherals will need multiple input signals to activate them. The actuation confirmation channel will have a corresponding word (a 16 bits entity) maps to the actuation control word.

Table 2 shows an example of an 8 bit control word and an 8 bit conformation word. Bits 1 and 2 are representation of the control signals for a peripheral and bit 8 in the conformation word represents the signal generated (state change because of the activation control signal). Two other such mappings are shown in the table. The state of the bits is shown next to each bit. For e.g., the bit 8-logic LOW of the control word will activate pump 3. The conformation of the pump 3 activation will be sensed by the bit 2-logic LOW of the conformation word.

There are many benefits of having this strategy; for example, detecting component failures before starting the tests, automated way of having component maintenance, postmortem of success or failures of the device operation

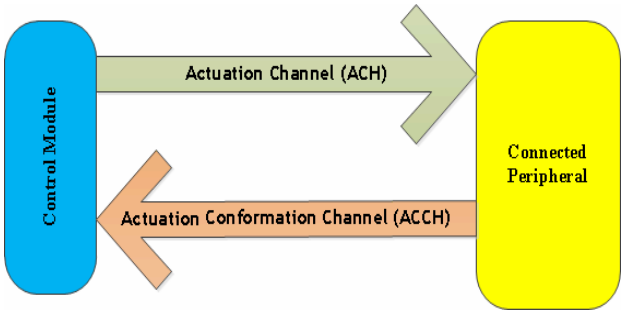

Figure 5. Actuation and conformation channels.

Table 2. Example of ACH and ACCH.

Actuation CNTL Word	*Actuation Conformation Word*	*Peripheral*
Bit 1 (H)	Bit 8 (H)	Actuator 1
Bit 2 (H)		
Bit 4	Bit 4	Not used
Bit 5 (L)	Bit 5 (L)	Pump 2
Bit 6	Bit 6 (H)	
Bit 7	Bit 7	Not used
Bit 8 (L)	Bit 2 (L)	Pump 3

and provide guarantee in collecting the accurate test measurement data information and hence accurate diagnosis.

The algorithm in **Figure 6** shows a way of implementing the concept in the SW.

The description of the each block and the data flow are shown below:

[0001, 0002, 0003]: Before the activation process is called, default state of the peripheral is read which needs to be activated. The control conformation word which represents the default state of the peripheral is acquired via the ACCH. Information pertained to the peripheral of interest, is masked out from the control conformation word (also known as status word).

[0004]: The control conformation word is verified for the expected state of the peripheral.

[0012]: Reset attempts will be made if the expected state of the control conformation word is not correct. This process will be repeated for a predefined number of times.

[0010, 0011]: If the reactivation attempts are failed, then and error log will be created, user will be alerted and the process will be aborted.

[0005]: If the status word is read and confirmed, then the peripheral will be activated (to support a biological process, e.g. a coil is turned on to generate required tem perature).

[0006, 0007, 0008, 0009]: The activation conformation

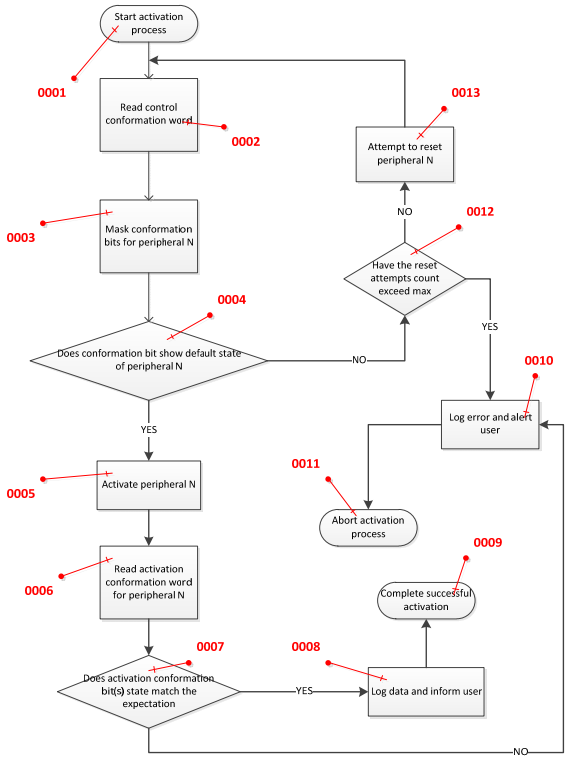

Figure 6. Algorithm for peripheral activation.

word will be read and it will be verified that the activation was successful. And the process will continue.

[0007, 0010, 0011]*:* If the activation conformation word indicates that the peripheral was not activated correctly then the process will be aborted, an error log will be created and the user will be notified.

4. Security Principles for Designing Safety Critical Software

Listed are some of the guiding principles that should be followed in build security around the POCT devices.

4.1. User Data

All access to user data (measurement data and configuration data) must be approved by the user. This means, minimally that password protection must be respected on all interfaces that can access this data (even hardware interfaces). This includes protection of data removed or copied from the device (memory dumps, cloud server access etc).

The POCT device is linked to a smart-phone (via Bluetooth) only when the smart-phone access is authorized by the user. When the measurement data is sent over to the p-cloud, the data base link (M2M) access must be password protected.

4.2. Security Threat Modeling

A security threat model needs to be developed at the system level and the device level. The security threat model lifecycle changes all the time. Security is not absolute entity, and the attack coordinates change continuously. What was considered as good security counter measures yesterday may not be applicable for tomorrow.

4.2.1. Opportunity to Improve Security

At any time there is an opportunity to improve security, at the end-to-end system level or at subsystem level or at component (SW and HW) level or at the SW API level or at data structure level, one should implement required changes. Relevant security requirements must be updated and they must be shared among all other stakeholders during device development.

If a design team found an opportunity for improving security, that information must be documented. After appropriate impact analysis all the impacted teams must be informed about the changes needed. This needs to happen regardless of bottom-up or top-down direction of the team hierarchy.

4.2.2. Balanced Approach for Security

In the POCT interconnected system, all the elements in the configured network must have individual security boundaries. All secrets stored on the device are unique to that device. This will prevent any compromises to the whole system in one device or a sub-system within the device. Development effort and cost should not be factors in deciding security protection strategies. A less costly device and an expensive device must be treated equally with respect to security protection.

4.2.3. Origin of Security Measures

All device security measures originate from the device. Activation of security agent or security sub-system or security service which is responsible for security must start from the device. It will then use the security services provided by other collaborating entities in the system. For example, data transport security is a function of the infrastructure. Whenever the data is sent to the p-cloud from the G-Node, encryption of the data must be implemented using the security services provided by the infrastructure.

4.2.4. Physical Security of on Board Data

The POCT device will contain clinical test data which belongs to the user. If a POCT device is opened physically, it should not provide any more access to the data contained within than is logically accessible through the user interface or external ports without opening the device. Circuit level attacks on the device should be unfeasible and the minimum bar to a hardware based attack on the user data should be a silicon level attack.

4.2.5. Security through Obscurity

Some security experts believe that security through obscurity as one of security measures which creates system design ambiguity to confuse security attacks. But this process of disclosing system design as insecure systems can lead to catastrophic security failure [15]. Automation systems have relied on “security through obscurity” to solve computer attacks problems. But the tools needed to conduct these attacks are easily obtainable for free and the potential consequences from an attack are large [16]. A system relying on security through obscurity may have theoretical or actual security vulnerabilities. Therefore, obscurity should not be relied upon as part of the security design. However, this can (and should) be used as a mitigation strategy against unknown SW bugs in the implementation. In general, anything that would divert any attempts of hackers in turning a SW bug into exploitation must be avoided.

5. Some General Operational Scenarios That Are Very Specific to Security in the POCT Devices

5.1. Processing Data in POCT Devices

***Scenario*:** Where will the “data process/generate results

stage" take place? The options are the POCT device or the server (p-cloud).

The data process needs to take place at the server (the other end of the SSL tunnel) domain where the computing power can be more than the POCT device.

5.2. Clinical and Surveillance Data Collection

***Scenario*:** Will the data collection (clinical & surveillance) be collected before the results of the test are presented to the user?

The data collection (clinical & surveillance) will happen when the test completion bit is set. This will be the trigger event for the smart-phone app to initiate the data collection. This means that the POCT device must have the capacity to store the collected data for during the testing process, which will simplify the SW implementation (**Table 3**).

5.3. Access from Multiple Devices (Preventing Unauthorized Access)

***Scenario*:** How can we ensure a one-to-one relationship/ connection between a particular user and the testing device? The one-to-one relationship can be guaranteed using logic similar to **Table 4**.

5.4. User authentication

5.4.1. User Validation and User Identification

There are two methodologies that can be used, "user validation" and "user identification". User validation is a process of comparing two data sets, while user identification is a process of using other data sets (data bases) to identify the user. Based on the environment the device is used, one can choose the authentication method. This selection can be set in the OMAP module. The result of this would be to set security info data (in the communication protocol) sent between the POCT device and G-Node, to reflect this configuration.

Table 3. Clinical and surveillance.

IF ((test completion bit set) **AND** (smart-phone has subscribed to the test completion event) **)**

THEN {
1-inform smart-phone about test completion
2 – provide smart-phone with a pointer to access the test data memory
}

ENDIF

Table 4. Multiple device access.

IF (an active session is present) **AND** (the user is not owner)
THEN (Reject any new session creation)
ENDIF
Note that this can be configured as to how many concurrent sessions can be established by the owner. This can be configured using the services provided by OMAP (Operation, Maintenance and Provisioning) Module.

5.4.2. Authentication Using Biometric

The biometric way of authentication process uses the user validation methodology. In this method, the user presents his or her credentials and the device compares what is presented with the user profile information stored in the device.

5.4.3. Authentication Using Active Directory

The active directory (AD) technology is an authentication mechanism developed by Microsoft. When the user tries to log on to the device, a security challenge is communicated from the AD sever to authenticate the user. The user can also log on to the device with the cached data (user profile info in the device) when there is no connection to AD. For the AD to work one need to have packet data network connectivity via cellular network or WLAN to the AD server.

5.5. Safely Dispose of the Testing Device

***Scenario*:** Will the user be advised to safely dispose of the testing device once the test is complete?

An approach to accomplish this requirement is that the smart-phone app (or the apps running on G-Node) needs to register with the POCT device for the completion event of the testing. This will enable the POCT device to send the testing event complete type packet to the smartphone. Then the smartphone app can indicate the test completion to the user by a visual or sound indication via UI (User Interface).

5.5.1. Mechanical Method

If the user attempts to use the device cartridge again, a mechanism can be implemented to prevent this by mechanical means. Some of the options that can be used are, using material properties of the cartridge and using paper microfluidics techniques that force only one time usage.

5.5.2. On Device Usage Prevention Using SW

If the whole POCT device needs to be disposed after one time use, then this can be done in SW; a usage indication bit can be stored to indicate that the device has been used once. This needs to be stored in the ROM area (non-volatile memory). Whenever the user attempts to use the

disposable device for a second time, the SW will prevent the user using the device.

5.5.3. More Complex and Fail Proof Mechanism for Preventing (or Restricting) Device Usage

Other alternative is to compare the device ID for the history (in a database) before the device is used. For this, we need to have the network connectivity at the beginning of the test. The FDA provides guidelines or usages that are outlined for the device identification process [17]. This will also prevent any illegal resale of the POCT devices.

6. P-cloud Architecture for Delivering Error Free SW System

The following section explains the concepts of the p-cloud. The principle here is that the division of data boundaries based on the usage model and it will guarantee to create an error free computing environment with respect to data handling. Each data repository can be accessible via separate gateways [**Figure 7**]. The gateways will connect to their partner entities in the G-Node which is usually a smartphone. In the case of the POCT, there are four gateways that get the access to appropriate data bases. These gateways have machine to machine (M2M) interface and user interfaces. The smartphone application developers will create apps which will provide access to the gateway entities in the p-cloud.

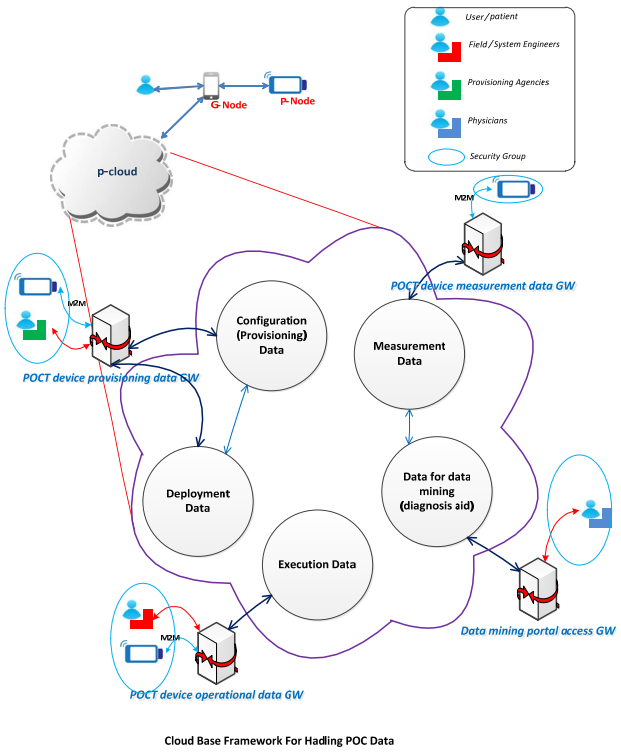

Figure 7. Cloud-Based data organization.

The p-cloud consists of mainly 5 classes of data base repositories.

6.1. Configuration, Measurement Databases

The configuration database is a place for storing POCT provisioning data. This will be accessible to the POCT device (M2M link) and to various health care providers and lab associates. The measurement data base will have the data collected during the POCT and will only be accessible to the POCT device.

6.2. Data Mining Database

The database for data mining will be used by the clinicians for diagnosing. This database is an aggregated version of the measurement data base. The data mining data base will be accessible to data mining application developers (data reporting tools) to provide suitable data portal for the clinicians.

6.3. Execution Path Database

The execution path database will collect operational data. The operational data base will help to debug field issues and it will help to log critical execution paths in the POC. The concept here is that the logged data will help to identify any computational deviation for a particular POCT process. It is very essential that all the processes and their execution sequences in a medical device (or mission critical device) should be clearly documented and assigned labels. In the POCT, a particular process or a set of processes will provide measurement data that is pertained to a test. There will be a set of documented execution paths in the design that will accomplish the result. If the documented execution paths differ during a POCT, the test will not be valid. The execution path data which will be logged in the data base will reveal any execution deviations against the expected execution for a given test. This process will not only help to remove any errors in SW coding during system acceptance testing (before launching the system), but will also be an essential tool for field engineering team. By analyzing the execution data logs, the field engineering team will be able to pin-point any root cause of any issues during actual operation of the system. The execution log data base will be accessible to the field team and the POCT devices in the system.

6.4. Deployment Database

The deployment data base will have the information about the association of the POCT device and the G-Nodes that can access the POCT device. It is possible that the association between the POCT device and the G-Node can be one-one or one-many (one POCT device and many kinds of G-Nodes such as smart-phones, PC

and any user computing devices). The configuration data base will have only the configuration that is pertained to the POCT device and the deployment data base will only have the association data.

The principle here is to segregate the data boundaries in different data bases and hence avoid any SW coding and execution errors.

7. Benefits of Adopting IEC62304 at Early Stages of Development

The IEC62304 was initially released in 2006 and it specifies the life cycle requirements for the medical device software based on the safety criticality of the software. Currently the IEC62304 is a harmonized EU standard, which is approved by the FDA as a recognized standard for SCS system development. It uses ISO14971 to do risk analysis. There are three classes of the software [IEC62304: Section 4.3] based on the level of risks or hazard presented to the patient and users.

- Class A: No injury or damage to health is possible
- Class B: Non-SERIOUS INJURY is possible
- Class C: Death or SERIOUS INJURY is possible

The standard specifies the degree of SW development process rigor based on the established classification. It places additional focus attention on the use of third party SW, known as Software with an Unknown Pedigree (SOUP).

The **Table 1** [ISO62304: Annex A.2] shows the development process requirements by the safety classification. The Class C has additional process requirements compared to Class A and Class B.

The hazard identification is done at the early stage of the development cycle by looking for opportunities to isolate and contain critical elements of the system to reduce the number of critically classed components. The effective system design (error free system) and software partitioning is driven by two main goals; building safe and effective product and minimizing development complexity and cost. Thus the architecture segregation is the key to risk isolation.

The software architecture breaks the medical device software down into smaller modules. These modules have a defined function and can be classified based on the functions' role in the system.

Figure 8 shows the architectural partitioning of the POCT device that has been developed. Furthermore the segregation of software components provided a view of the Class C functional modules from the other modules.

There are two main approaches that can be used to segregate the system, hardware approach and software approach which is tied directly to operating system parameters. The important operating system parameters are as follows: Task Isolation, Restricted memory access, CPU/time protection, Exception handling and Virtualization. If an embedded OS is used in the design, the first three parameters can be set easily. The exception handling needs to be done at the code level. The virtualization helps to create a separate partition over the functionalities.

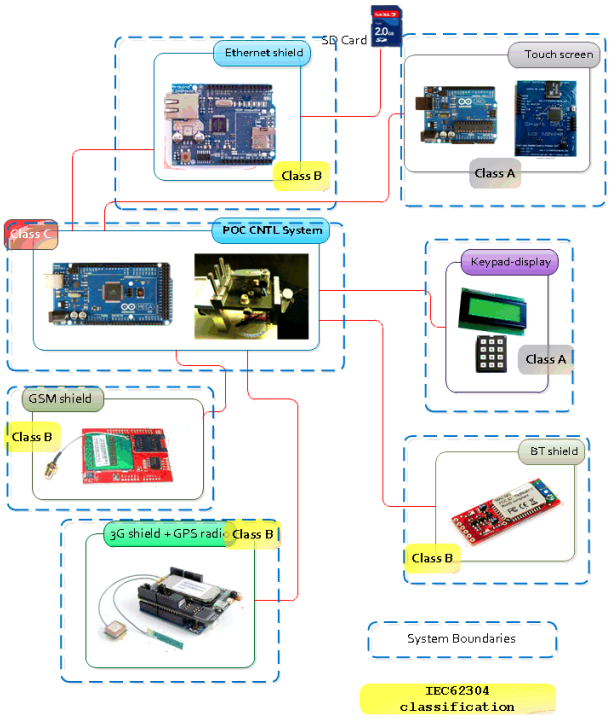

Figure 8. Architecture partitioning and IEC62304 classification.

Attempting to get compliance with the IEC62304 development process has many benefits which includes the development of error free software. During the project cycle, a software development plan must be maintained and it should reflect the changes and it must provide a real picture of the activities. All the team members must be aware of the plan and they must follow the plan. Software risk assessment must be done based on the ISO14971 and risk should be mitigated. The risk management activities should be given the importance and must be performed during the design phase.

The system integration test must be conducted at the early stages of the project to mitigate any interface issues. The device manufacturer must take full responsibility for the entire software stack (including SOUP) [IEC62304: Sections: 5.1.1, 5.1.5, 5.1.7, 5.2.2, 5.3.3 - 3.4, 5.3.6, 6.1, 7.1.2, 7.1.3, 7.4.1 - 4.2 8.1.1 - 1.2]. All the known bugs in the SOUP must be enumerated and intended use and conditions for the SOUP must be stated. The risks need to be identified (by analyzing the failure modes) and mitigated for the SOUP portion of the software stack. The configuration management for the SOUP must be implemented. The importance of evaluating SOUP mo-

dules is mentioned in the standard very strongly.

With these recommendations, delivery of the error free software for the medical device and the maintenance after the first product release can be managed.

8. Using Combinatorial Design Methodology for Generating Smart Testing Vectors

The combinatorial design methodology will help to identify [18-22] the basic and essential test vectors that are required for verifying the system. There will be many possibilities that can be validated. However it is not possible to test all the possibilities within a given short cycle of testing available. Combinatorial testing is an adoptable methodology which is useful in a wide range of situations to uncover software defects. It is based on the statistical process that while the behavior of a software system may be affected by a large number of factors, only a few factors are dominant in inducing software failures [23].

There are two main reasons for the large number of software defects in the design and implementation of Safety-Critical Systems (SCS), system with complex requirements and absence of software verification tools. Exhaustive system testing which covers all the use case scenarios is hard with in the allocated testing time. Manually identifying the highly probable complex combination of inputs of the system under test is impossible, because of the combinatorial explosion in the great number of states that an SCS can reach when it executes [24].

The combinatorial design based approach offers a way of testing the system with fewer key inputs. This process will build structured variation into testing scenarios. The structured variation focuses on creating almost all the key subsystem behaviors. Most of the issues are triggered by one or two entities in the system and this process will engage all possible two way interactions (or combinations).

Figure 9 shows a comparison between manual and the combinatorial testing process.

Note that some of the combinations are not valid. They are called invalid pairs, and these can be removed based on the subject matter expert's recommendation. Also a new pair can be inserted based on the partial knowledge of the system under test.

Manual Process	Combinatorial Design Process
Use more scenarios than needed	Right number of scenarios
Forget an important combination	Additional scenarios
Repeating the same scenario	Specialized cases
	Unexpected combinations
	Notable combinations
	Rare combinations that may occur

Figure 9. Benefits of combinatorial testing process.

There are numerous open source tools available. **Figure 10** shows the output (test vectors generated) for the basic POCT system interconnectivity using a tool called HEXAWISE [25].

Note that the tool has come out with 99 two way interaction scenarios out of 36,450 exhaustive test scenarios which are not possible to execute practically.

9. Agile Project Management Approach in Developing SCS

9.1. Project Management in IEC62304

In the IEC62304, Annex B talks about the project management strategies that are suitable for the developping medical device software. It references the ISO/IEC 12207, which describes the three main development models. They are stated below:

9.1.1. Waterfall Model

This is a "once-through" strategy where the requirements are gathered at the beginning of the project and then the system is designed, implemented, tested and delivered. This has neither multiple development cycle nor interim software delivery.

9.1.2. Incremental Model

This is an "incremental" strategy where the requirements are elicited at the beginning and then it goes through a multiple development cycle with the possibility of interim software delivery.

9.1.3. Evolutionary Model

This is an 'evolutionary' strategy as contrast to the other methodologies; all the requirements are not defined at the beginning. It goes through a multiple development cycle in partnership with the customer (stakeholders) and delivers incremental interim software versions. The requirements evolve as the software become mature.

Note that in the first and the second methods the validation and verification is done multiple times, hence the test cases become very mature. This helps to develop mature systems with virtually zero errors.

9.2. Agile Project Management Process

Agile project management process is very similar to the evolutionary model, which encourages *interaction between individuals* over useless processes and tools, interests in *working software* over comprehensive documentation and promotes *customer collaboration* over contract

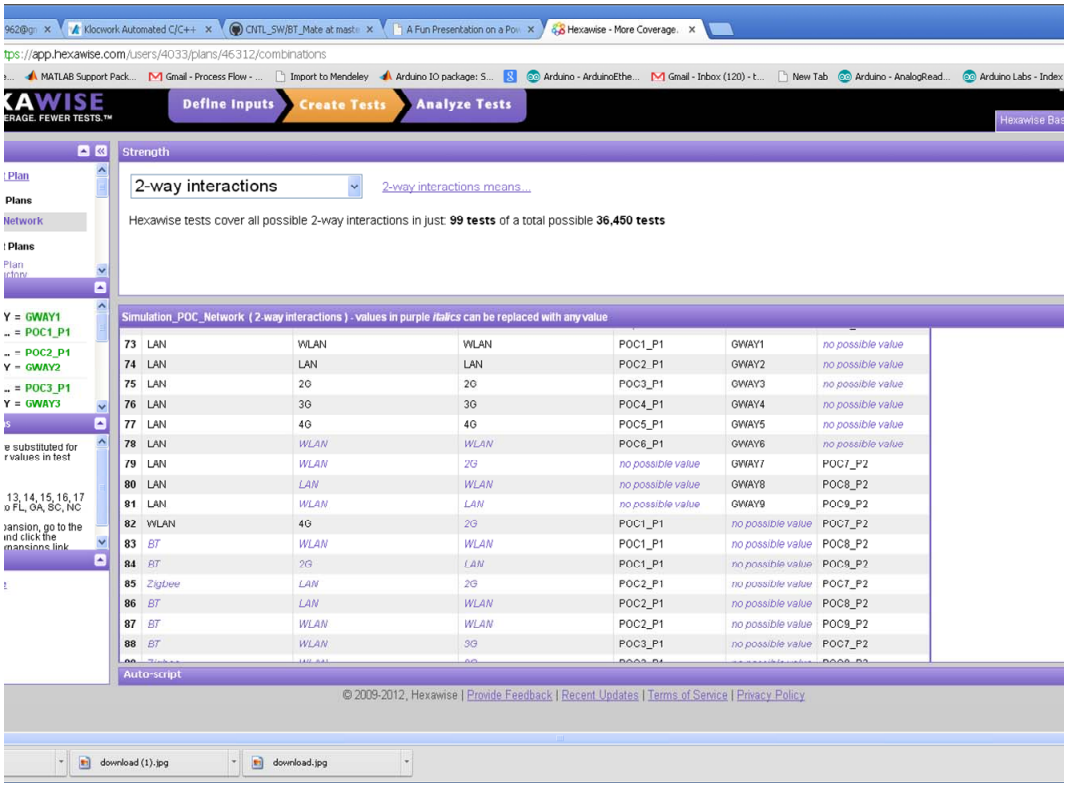

Figure 10. The test vector generated from HEXAWISE.

negotiation and *Responds to change* over, following a constant plan [26]. This methodology has 12 main principles [27], where the highest priority is to satisfy the customer through early and continuous delivery of valuable software. This recognizes the fact that the change in scope is unavoidable and it provides a framework to deliver high quality software. This is suitable for delivering medical software system with highest quality possible.

If safety-critical systems require greater emphasis on activities, such as formal specification and requirements management, then an agile process will include these as necessary activities. Furthermore, agile methods focus more on continuous process management and code-level quality than the classic software engineering process model [28]. In addition to the advantages of using Agile project management, it supports the requirement traceability in the SCS engineering domain, where agile traceability from functional and safety requirements is necessary to meet certification process [29].

10. Conclusions

In this paper, a study of software development methodologies and processes has been presented. These recognize that effectiveness in all the development phases is required in producing error free software. A requirement elicitation methodology, Value Based Requirements have been presented, which will eliminate any ambiguities and requirement errors. The key to successful error free software is to remove all the errors at the higher development phases.

A system design methodology is shown which describes the benefit of creating the basic system building blocks view of the end to end system. This process highlights the importance of having a clear architectural view of the system. There are some key strategies such as, isolation of system modules, need for creating loosely coupled system and its benefits, actuation and actuation conformation channel and an algorithm for implementation and they are presented. The goal of these approaches is to develop error free software system that can be used in medical diagnosis environment without any fear that the system is not dependable. These strategies will help produce a product with highly predictable behavior.

The role of security in SCS is presented, focusing on a few important topics; user data privacy, threat modeling, and process of improving security, using services of security infrastructure, authentication and cryptography. Some of the operational scenarios pertained to the operation of the POCT device may cause security risks and the process of mitigating those risks is presented.

A cloud-based architecture is presented for the purpose of storing five types of data that are relevant to the POCT system. A gateway for data access concept is described.

This will separate the implementation of the data base design and the associated applications. The segregation will provide data boundaries that will eliminate any software implementation errors; hence the system will be dependable for SCS (or medical system) operations.

The compliance of the IEC62304 will make the system more dependable with zero errors as discussed.

The process of smart testing based on the combinatorial design methodology is explained and its benefits are shown with an example using an open source tool.

Finally the Agile project management process that can be used for delivering software incrementally is discussed.

These strategies and methods discussed will guarantee to deliver high quality SW code for safety and mission critical system, in particular, in the development of medical device software. These strategies can be used in conjunction with IEC 62304 or similar SCS standards.

The processes and methods presented in this paper have many advantages over the traditional SW development processes. The Value Based Requirement eliciting process provides key value attributes which will directly resolve a real world problem. Because it provides a measurable attribute to the requirement, the implementtation (*i.e.* the SW code) can be validated without any ambiguities. In other words the behavior of the SW can be described in terms of measurable requirement attributes. Therefore a highly predictable system will be developed using these processes.

The actuation and actuation conformation channels in the POCT device control will guarantee that all the actuators are checked before any bio medical process is initiated. This will eliminate any measurement errors that may interfere with medical diagnosis.

The existing medical devices in the market today are sold as individual medical devices as being opposed to medical device systems. The system (Section 6) described in this paper has infrastructure entity and device entity, which is an end-2-end architecture for deploying POCT devices. Because the infrastructure is architected mainly for carrying POCT type of devices, the architecture entities are customizable to implement the POCT requirements.

The application developers who are interested in creating smartphone applications can focus on individual gateways, which are architected to deliver only one service functionality. This will eliminate any unwanted coupling that may occur between the applications.

The combinatorial testing process will help contain any software defects before the product launches or maintenance upgrades. This can be done with fewer resources and relatively less testing time than the traditional testing processes.

Finally, the agile project management process will enable device SW vendors to manage constantly changing requirements effectively, and deliver mature SW implementations.

REFERENCES

[1] A. N. Srivastava and J. Schumann, "Software Health Management: A Necessity for Safety Critical Systems," *Innovations in Systems and Software Engineering*, 2013, pp. 1-15, In Press.

[2] S. Wang, A. Ayoub, R. Ivanov, O. Sokolsky and I. Lee, "Contract-Based Blame Assignment by Trace Analysis," *Proceedings of the* 2*nd ACM International Conference on High Confidence Networked Systems*, Philadelphia, April 2013, pp. 117-126. doi:10.1145/2461446.2461463

[3] R. H. Birkhahn, E. Haines, W. Wen, L. Reddy, W. M. Briggs and P. A. Datillo, "Estimating the Clinical Impact of Bringing a Multimarker Cardiac Panel to the Bedside in the ED," *The American Journal of Emergency Medicine*, Vol. 29, No. 3, 2011, pp. 304-308. doi:10.1016/j.ajem.2009.12.007

[4] L. V. Dommelen, F. H. V. Tiel, S. Ouburg, *et al.*, "Alarmingly Poor Performance in Chlamydia Trachomatis Point of Care Testing," *Sexually Transmitted Infections*, Vol. 86, No. 5, 2010, pp. 355-359. doi:10.1136/sti.2010.042598

[5] J. M. S. Alonso and D. M. M. Pereira, "Medical Software Requirements at the New Cuban Regulations for Evaluation and State Control of Medical Devices," *IFMBE Proceedings on Biomedical Engineering CLAIB*, Habana, 16-21 May 2011, pp. 433-435.

[6] R. F. Goldsmith, "Discovering Real Business Requirements for Software Project Success," 2004 http://www.ebookmall.com/author/robin-f-goldsmith

[7] Y. Yan, S. Liu, Q. Zhang and H. Wu, "Analysis of Medical Device Recall Reports in FDA Database in 2005-2006," *IFMBE Proceedings*, 2013, pp. 766-769.

[8] S. Kierkegaard and P. Kierkegaard, "Danger to Public Health: Medical Devices, Toxicity, Virus and Fraud," *Computer Law and Security Review*, Vol. 29, No. 1, 2013, pp. 13-27. doi:10.1016/j.clsr.2012.11.006

[9] National Health Service Infrastructure UK. http://n3.nhs.uk/technicalinformation/n3networkoverview.cfm

[10] Canada Health Infoway, 2012. https://www.infoway-inforoute.ca/index.php/about-infoway

[11] A Notation to Describe Behavior of Complex and Dynamic Systems, University Of Ottawa. http://www.usecasemaps.org/aboutucms.shtml

[12] G. Holl, D. Thaller, P. Grünbacher and C. Elsner, "Managing Emerging Configuration Dependencies in Multi Product Lines," *Proceedings of the* 6*th International Workshop on Variability Modeling of Software-Intensive Systems*, Leipzig, 25-27 January 2012, pp. 3-10.

[13] B. Ostermaier, M. Kovatsch and S. Santini, "Connecting

Things to the Web Using Programmable Low-Power WiFi Modules," *Proceedings of the* 2*nd International Workshop on Web of Things*, San Francisco, 16 June 2011.

[14] C. Ruz, F. Baude and B. Sauvan, "Component-Based Generic Approach for Reconfigurable Management of Component-Based SOA Applications," *Proceedings of the* 3*rd International Workshop on Monitoring, Adaptation and Beyond*, 2010, pp. 25-32.

[15] S. Bono, A. Rubin, A. Stubblefield and M. Green, "Security through Legality," *Communications of the ACM*, Vol. 49, No. 6, 2006, pp. 41-43. doi:10.1145/1132469.1132499

[16] M. McKay, "Best Practices in Automation Security," *IEEE Cement Industry Technical Conference*, San Antonio, 14-17 May 2012, pp. 1-15.

[17] US Food and Drug Administration Certification Authority. http://www.fda.gov/MedicalDevices/DeviceRegulationandGuiance/UniqueDeviceIdentification/ucm054169.htm

[18] M. F. Johansen, Ø. Haugen and F. Fleurey, "Bow Tie Testing—A Testing Pattern for Product Lines," *Proceedings of the* 16*th European Conference on Pattern Languages of Programs*, Irsee, 13-17 July 2011.

[19] V. A. de Santiago Júnior and N. L. Vijaykumar, "Generating Model-based Test Cases from Natural Language Requirements for Space Application Software," *Software Quality Journal*, Vol. 20, No. 1, 2012, pp. 77-143. doi:10.1007/s11219-011-9155-6

[20] J. Natarajan, J. Wells, A. Chatterjee and A. Singh, "Distributed Comparison Test Driven Multiprocessor Speed-tuning: Targeting Performance Gains under Extreme Process Variations," *Proceedings of the Asian Test Symposium*, 20-23 November 2011, New Delhi, pp. 154-160.

[21] C. Nie and H. Leung, "The Minimal Failure-Causing Schema of Combinatorial Testing," *ACM Transactions on Software Engineering and Methodology*, Vol. 20, No. 4, 2011, Article No. 2. doi:10.1145/2000799.2000801

[22] I. Segall, R. Tzoref-Brill and E. Farchi, "Using Binary Decision Diagrams for Combinatorial Test Design," *Proceedings of the* 2011 *International Symposium on Software Testing and Analysis*, Toronto, 17-21 July 2011, pp. 254-264. doi:10.1145/2001420.2001451

[23] R. N. Kacker, D. R. Kuhn, Y. Lei and J. F. Lawrence, "Combinatorial Testing for Software: An Adaptation of Design of Experiments," *Measurement*, 2013, In Press. doi:10.1016/j.measurement.2013.02.021

[24] M. I. Capel and L. E. M. Morales, "A Formal Compositional Verification Approach for Safety-Critical Systems Correctness: Model-Checking Based Methodological Approach to Automatically Verify Safety Critical Systems Software," *Proceedings of the* 14*th International Conference on Enterprise Information Systems*, Wroclaw, 28 June 2012, pp. 105-112.

[25] Test Design Tool, HEXAWISE. https://app.hexawise.com/

[26] Agile Project Management Process Alliance, 2013. http://www.agilealliance.org/the-alliance/the-agile-manifesto/

[27] Agile Project Management Process, 12 Principles, 2013. http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/

[28] K. Gary, A. Enquobahrie, L. Ibanez, P. Cheng, Z. Yaniv, K. Cleary and J. Heidenreich, "Agile Methods for Open Source Safety-Critical Software," *Software*: *Practice and Experience*, Vol. 41, No. 9, 2011, pp. 945-962. doi:10.1002/spe.1075

[29] M. Taromirad and R. F. Paige, "Agile Requirements Traceability Using Domain-Specific Modelling Languages," *Proceedings of the* 2012 *Extreme Modeling Workshop*, Innsbruck, 1 October 2012, pp. 45-50.

