VERIFICATION PROCESS OF THE ST. JUDE MEDICAL ATRIAL FIBRILLATION
IMPLANTABLE CARDIAC MONITOR DEVICE

A Thesis

Presented to

The Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Engineering, with Specialization in Biomedical Engineering

by

Jimmy Quoc Hy Duong

December 2010

© 2010
Jimmy Quoc Hy Duong
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: VERIFICATION PROCESS OF THE ST. JUDE
MEDICAL ATRIAL FIBRILLATION
IMPLANTABLE CARDIAC MONITOR DEVICE

AUTHOR: Jimmy Quoc Hy Duong

DATE SUBMITTED: December 2010

COMMITTEE CHAIR: Lily Laiho, Dr. and Professor

COMMITTEE MEMBER: Robert Crockett, Dr. and Professor

COMMITTEE MEMBER: Scott J. Hazelwood, Dr. and Professor

ABSTRACT

VERIFICATION PROCESS OF THE ST. JUDE MEDICAL ATRIAL FIBRILLATION IMPLANTABLE CARDIAC MONITOR DEVICE

By: Jimmy Quoc Hy Duong

The St. Jude Medical SJM Confirm™ Implantable Cardiac Monitor (ICM) is a small implantable device that is used to detect arrhythmias and stores the electrogram (EGMs) records for physicians to verify the arrhythmia. The objective of this device is to provide physicians with the technology to monitor patients who are suspected of having arrhythmias but do not exhibit any symptoms of this heart condition. This device allows for long term continuous monitoring of patients and provides recordings for physicians to prove the existence of an arrhythmia. With the help of this device, doctors can make better decisions on determining what type of treatment patients would need and provide care for those who otherwise may be diagnosed too late.

The effort that goes into creating an ICM device is a strict and stringent process. The reason is to ensure that the device is of high quality. The software itself is quite complicated and the verification of the software is critical. Reviews of all verification test designs and test implementations are conducted to ensure complete coverage of software requirements. As part of the submission process for approval of these devices, a traceability report of the requirements to passed test cases must be provided, along with evidence that the software poses no harm to the patient.

Keywords: St. Jude Medical, Implantable Cardiac Monitor, Arrhythmias, Electrogram Records, Software Verification

ACKNOWLEDGMENTS

I would like thank my thesis committee – Dr. Lily Laiho, committee chair, for your continual advice and support throughout this entire process; Dr. Scott J. Hazelwood and Dr. Robert Crockett, for your support as committee members.

I would also like to say a special thank you to my wife, Tina, for her constant support and encouragement (and also countless hours as my chief editor) as I worked on getting my Master's degree.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
<i>I. INTRODUCTION.....</i>	<i>1</i>
<i>Background on Atrial Fibrillation</i>	<i>1</i>
The Human Heart	1
Arrhythmias	3
Definition of Atrial Fibrillation	4
Symptoms of Atrial Fibrillation.....	5
Diagnosing Atrial Fibrillation.....	6
Treatment of Atrial Fibrillation	8
<i>St. Jude Medical SJM ConfirmTM Implantable Cardiac Monitor</i>	<i>9</i>
<i>II. Objective of the Software Verification</i>	<i>12</i>
<i>Software Verification Process</i>	<i>12</i>
Software Requirement Specifications	13
Test Strategies	14
Test Designs.....	14
Test Implementation	15
Verification Test Review Process	15
Regression Runs.....	18
Formal Run	18
Master Verification Matrix Report.....	19
Software Verification Report.....	20
Design History File.....	20
<i>III. Methods and Material.....</i>	<i>21</i>
<i>SAINTS</i>	<i>21</i>
<i>Unity Test Library</i>	<i>22</i>
<i>SMART.....</i>	<i>22</i>
SMART State Diagram.....	24
<i>SMART vs. Unity Test Library</i>	<i>26</i>
<i>IV. Results</i>	<i>30</i>
<i>Master Verification Matrix Report.....</i>	<i>30</i>
<i>Software Verification Report.....</i>	<i>34</i>
<i>V. Conclusion</i>	<i>35</i>
<i>AFM 1.1</i>	<i>35</i>
<i>Future Work</i>	<i>36</i>

<i>References</i>	38
-------------------------	----

LIST OF TABLES

Table	Page
<i>Table 1: AFM 1.0 Firmware Master Verification Matrix Report</i>	30
<i>Table 2: AFM 1.0 Requirements Traceability Report</i>	32

LIST OF FIGURES

Figure	Page
<i>Figure 1: Anatomy of a Human Heart and the flow of blood through the heart [1].....</i>	1
<i>Figure 2: Action Potential by Cardiac Region [2]</i>	2
<i>Figure 3: Annotated Electrocardiogram</i>	3
<i>Figure 4: ECG recording of Sinus Rhythm.....</i>	5
<i>Figure 5: ECG recording of Atrial Fibrillation</i>	5
<i>Figure 6: St. Jude Medical SJM ConfirmTM Implantable Cardiac Monitor.....</i>	9
<i>Figure 7: Flowchart of the Software Verification Process.....</i>	12
<i>Figure 8: Sample SMART Test Case</i>	23
<i>Figure 9: Initial State.....</i>	24
<i>Figure 10: Ordinary State</i>	25
<i>Figure 11: Decision State</i>	25
<i>Figure 12: Failure State</i>	26
<i>Figure 13: Success State</i>	26

I. INTRODUCTION

Background on Atrial Fibrillation

The Human Heart

The human heart is a muscle that is made up of four chambers, each of which works together to circulate oxygenated blood from the lungs into all the parts of the body and also circulates oxygen depleted blood from the body into the lungs.

Figure 1: Anatomy of a Human Heart and the flow of blood through the heart [1]

In Figure 1, the pink and blue arrows illustrate the flow of blood through the anatomy of the human heart. The pink arrow represents oxygenated blood, while the blue arrow represents deoxygenated blood. This process begins when the atriums first relax, and then, deoxygenated blood enters the right atrium and oxygenated blood enters the left atrium. Once the atrium chambers are filled, the muscle contracts and pumps the blood into its respective ventricles. The atrium then relaxes and the chamber fills up with

blood again. Likewise, the ventricles also contract once they are filled and thus push the blood out to either the lungs (right ventricle) or the rest of the body (left ventricle). The sequence of this process is very well-timed where an adult sized heart can effectively move 2,000 gallons of blood through the body in a single day.

Figure 2: Action Potential by Cardiac Region [2]

The relaxation and contraction of the heart muscle is controlled through electrical impulses in the muscle of the wall. Figure 2 illustrates the key components that help the heart beat, along with their action potential [2]. The colors of the action potential are color coded with the paths that the electrical signal takes as it goes through the respective parts of the heart. Within the heart, there is a natural pacemaker called the Sinoatrial (SA) node. In figure 2, the green lines show the path of the electrical impulse that the SA node initiates through the atrial muscle. As this electrical signal passes through each muscle cell, the muscle cell contracts, and this contraction squeezes blood through the atriums. As the electrical signal travels down the pathways in the atriums, the signal

reaches the Atrioventricular (AV) node, which regulates the transmission of the signal into the ventricles and through the Purkinje fibers. This signal contracts the muscle cells in the ventricles and squeezes blood into the body.

Figure 3: Annotated Electrocardiogram

Figure 3 illustrates a strip of an electrocardiograph with a series of QRS complexes. Each QRS complex represents a single heart beat. The P-wave represents the contraction of the atriums, and the QRS or simply, the R-wave, represents the contraction of the ventricles. Lastly, the T-wave represents the relaxation of the ventricles.

A normal heart beat is classified as a Sinus rhythm. For an average adult, the heart at rest will beat about 60-80 beats per minute. Each P-wave would be followed by an R-wave.

Arrhythmias

When a heart rhythm is not regular, this arrhythmia can be classified into different categories. Slow heart rate rhythms are classified as bradycardia while fast heart rhythms

are classified as tachycardia. For the most part, bradycardia and tachycardia rhythms still have the basic shape of the QRS complex, just with different rate intervals.

There are many things that can contribute to the cause of arrhythmias. Some possible causes [3] are scarring of the heart tissue from a heart attack, heart disease, stress, alcohol abuse, caffeine, and hypertension. Stimulants such as alcohol and caffeine can raise the heart rate, causing a mild tachycardia. Scar tissue causes a disturbance in the electrical pathways, and the signal deflects in different directions instead of following the correct pathway. This is also known as a re-entrant rhythm where the electrical impulse deflects in a different direction causing electrical impulses flowing in multiple directions. This leads to fibrillation or flutter because the atriums can start to quiver uncontrollably due to the multiple sources of electrical impulses.

Definition of Atrial Fibrillation

When the atriums contract at fast abnormal rates, this condition is also known as Atrial Fibrillation (AF). AF is a condition that affects about 2.2 million Americans and is projected to increase to 5.6 million Americans by the year 2050 [4].

AF is characterized with the condition where the atrial chambers quiver irregularly at rates often greater than 300 beats per minute. Since AF occurs in the atrial chambers, it is possible that the ventricular chambers are beating at a normal rate, which means that AF can be asymptomatic. Thus, by itself, AF is generally not life threatening.

Figure 4: ECG recording of Sinus Rhythm

Figure 5: ECG recording of Atrial Fibrillation

Figure 4 provides an electrocardiogram recording of a normal sinus rhythm. As you can see from the figure, each P wave has a corresponding QRS wave. This implies a one-to-one pairing of the heart beats in the atrium to the ventricle chambers. In Figure 5, this one-to-one pairing is non-existent. There are many P waves for each QRS and this is classified as Atrial Fibrillation since the atrium is out of sync with the ventricles and is beating more rapidly and inconsistently than the ventricles.

Symptoms of Atrial Fibrillation

If left untreated, AF can lead to other serious conditions such as chronic fatigue and congestive heart failure. If symptoms do present themselves, some of the common symptoms [5] are:

Chest pain

Decreased blood pressure

Dizziness/Lightheadedness

Fainting (syncope)

Palpitations, sensation of heart beating

Shortness of breath

Sweating

Weakness and difficulty exercising

It is possible for an individual to be asymptomatic, and this can be potentially hazardous since the individual may not be able to get help until it is too late.

Diagnosing Atrial Fibrillation

If symptoms are present, a doctor can easily diagnose a patient with AF and confirm this by collecting an electrocardiogram record. But since AF can be asymptomatic and people can have spontaneous episodes at random times, this poses a higher degree of difficulty since a doctor might not be able to detect the arrhythmia while the patient is in the doctor's office. Without being able to get an electrocardiogram recording, the doctor can not verify or accurately diagnose a patient with AF. So if the patient is not exhibiting any signs in the office, the doctor can only monitor the electrocardiogram recordings outside the office through a cardiac monitoring device. These devices are portable and are relatively small, which allows patients to be monitored for an extended period of time outside of the doctor's office. Studies show that cardiac monitors diagnose arrhythmias in up to 88% of the cases studied, a percentage that is much higher than traditional methods [6].

Some of the most common types of cardiac monitors are:

- **Holter Monitors:** A holter monitor is a small external recorder that is worn for a short period of time, generally between one to three days. Electrodes are placed on the chest and connected to the monitoring device. Much like an EKG, these electrode patches record the electric impulses of the heart muscles.
- **Portable Event Monitors:** Patients who are symptomatic and have less frequent episodes may need a cardiac monitor that lasts longer. Just like holter monitors, electrode patches are placed on the patient to capture the electrical signals. These portable event monitors, however, need the user to initiate a recording whenever they feel the onset of symptoms.
- **Transtelephonic Monitors:** Transtelephonic Monitors are devices that also record the heart rhythm and transmits it to the doctor's office over the telephone. Once the doctor receives the recordings, he or she can analyze the EGMs and verify if the patient indeed has AF. This method provides a quicker turn around than the portable event monitors or holter monitors since the patient does not need to travel to the doctor's office in order for the doctor to check if the device detects anything.

The intent of these monitors is to provide continuous monitoring of the patient in the hopes that they can capture electrocardiogram recordings of anything substantial for the doctor to determine if the patient has AF or not.

As helpful as these devices seem, there are still some major deficiencies with them. First, having external patches will cause some level of discomfort over time and will inconvenience a patient who wants to do certain activities such as swimming. If a person is asymptomatic, they would also not know when to start the recording, and this may lead

to misdiagnosis. There are also those individuals who may have infrequent AF episodes and would need a long-term monitoring device that would constantly record the heart rhythm.

Treatment of Atrial Fibrillation

There are three main treatments [7] for AF:

- a) medication to slow the heart down,
- b) electrical cardioversion to help regulate the heart beats, and
- c) ablation of the muscle cells.

Doctors use an assortment of medications to help control AF. Drugs commonly used can range from beta blockers to blood thinners in the attempt to help control the heart rate and the heart rhythm.

Electrical cardioversion [7] helps regulate the heartbeat through electrical shock, such as those created by a defibrillator. This type of treatment aims at trying to reset the heart through the electrical shock to the chest or to the heart muscle. If the patient suffers from long term AF, electrical cardioversion may not be as effective since they are likely to experience AF again after cardioversion.

Ablation attempts to treat AF by removing or killing the muscle cells that are triggering the re-entrant nodes. The doctor can perform a procedure to obtain a mapping of the electrical signals of the heart muscle cells and determine the location of the re-entrant nodes. The physician can then go and ablate that area so that the electrical pathways of the muscle cells return to normal.

St. Jude Medical SJM Confirm™ Implantable Cardiac Monitor

An Implantable Cardiac Monitor (ICM) is a cardiac monitoring device that is implanted subcutaneously in a patient to help capture electrocardiogram recordings of the heart beats and store them for a physician to review at a later time. Since this device is implantable, it helps eliminate some of the deficiencies of an external cardiac monitor. This implantable device allows patients to maintain their level of quality of life. Patients do not suffer from the discomfort of having to attach external patches to their skin, or the inconvenience of having to take off their monitoring device when they take a shower. An ICM provides continuous long-term monitoring, something that their external counterpart cannot provide.

Figure 6: St. Jude Medical SJM Confirm™ Implantable Cardiac Monitor

The St. Jude Medical SJM Confirm™, seen above in figure 6, is an ICM that is implanted subcutaneously near the collar bone. This ICM can be activated to start recording ECG signals by detecting specific heart rhythms, and it can also be activated by the patient through an external activator that is placed over the device. The electrodes reside on the surface of the actual device, which makes this procedure less invasive since the device is leadless.

The SJM Confirm™ ICM device is intended for:

- a) Patients with clinical syndromes or situations at increased risk of cardiac arrhythmias
- b) Patients who experience transient symptoms that may suggest a cardiac arrhythmia

It is the smallest ICM on the market today at 6.5 cc, weighs 12 grams, and has longevity of three years. The device can also store up to 48 minutes of EGM recordings, or 147 episodes [8].

From a software perspective, the features implemented [9] on the device are:

- a) Detection of Tachycardia, Bradycardia, and Asystole Episodes
- b) Detection of Atrial Fibrillation Episodes
- c) Distinguishing the difference between noise, physical activity, and a tachycardia episode
- d) Compatibility with Merlin™ Patient Care System
- e) Patient-triggered activation option for EGM storage
- f) SenseAbility® technology for greater sensitivity
- g) Data storage options for flexibility
- h) Receipt of Vital Information through Extensive Data Reports
- i) Comprehensive diagnostic data to assist in patient management
- j) Trans-telephonic monitoring for timely and accurate data transmission

The Trans-telephonic feature is also an interesting feature because it provides patients the ability to upload data collected by the implanted devices directly into a database from the comfort of their own home telephone system. This is provided through the Patient

Activator, which is an external device that is used to trigger patient-activated recordings as well as download data from the SJM Confirm™ ICM device to transfer it over the telephone system. Thus, doctors can closely monitor their patients in a more effective and timely manner instead of having all their patients set up in-office appointments.

II. Objective of the Software Verification

As the Software Verification Organization of a medical device company, our main goal is to ensure that our products are designed according to their specifications and most importantly, to ensure that the device is safe for patients and that all potential hazards are mitigated.

Thus the software verification process [10] is a well-defined process with many checks and balances along the way to ensure that all conditions are properly handled.

Software Verification Process

Figure 7: Flowchart of the Software Verification Process

The Software Verification Process consists of multiple phases. In figure 7, the flowchart illustrates that the cornerstone of the verification process is the Software Requirements Specifications, also known as the SRS. With these requirements in place, the verification engineers conduct test strategy meetings to plan the testing approach of verifying each requirement in the SRS. Once a strategy is in place, the verification test

design and test implementation is created to ensure that the software meets all the requirements in the SRS. All designs and implementations must go through a formal review, and once the documents are accepted testing of the device begins. The testing portion of the flowchart consists of the Formal Runs. According to the figure above, the detection of any issues in this phase requires a SWR and may require a re-test. Once the Formal Runs are completed, the results are packaged as part of the submission packet to prove that the software is safe for human use and meets the specified requirements. The last phase of the project cycle consists of the traceability report along with the software verification report. As you can see, each phase plays a critical role in producing a well-tested product, and will be further detailed below.

Software Requirement Specifications

The SRS is the building block to the entire software development. The SRS tries to detail the functional behavior of the product, but at the same time, it does not impose onto the developer a particular implementation. Throughout the SRS, the document varies in complexity and style from feature to feature. However, regardless of the style of the requirements, the goal of the SRS is to provide the software developers with the specifications on how to design the software code. Independently, the software verification engineers design test cases to test the functionality of the software through these same requirements. This results in a double redundancy in which the implementation of the software and its testing goes through two separate processes. If the verification test cases pass, this assures us that the software code was designed to the specifications described in the SRS. Software verification engineers participate in

analyzing these requirements to make sure that they are complete, thorough, and also testable.

Test Strategies

After analyzing the SRS, the first step in the software verification process is to use the SRS to develop a test strategy that will help guide the software engineer in planning the test design. Although test strategies are not required for every feature, it is highly recommended that new feature development, and features that are complex, should go through a phase of strategy planning.

During this phase of testing, the tester studies the requirements and gains a general understanding of how to test their features. The tester then organizes the different test scenarios using state diagrams, outlines, bullet points, or charts and tables. After organizing this data, the tester holds a strategy workshop meeting to discuss the test strategy. Individuals from other groups then provide assistance during this meeting to make sure that the test strategy is complete and that all the scenarios to fully test the requirements are identified.

Test Designs

After the test strategy review, the next phase is the test design which provides details regarding each test case. Details are placed into each test case to describe the requirement coverage, the configuration parameters, the test setup, the sequence of steps in the test case, and the expected results. This document, along with the requirements and test strategy, goes through a series of reviews.

Each test case indicates which requirements it covers and whether it is a positive coverage or a negative coverage. A positive test case is a direct test, that the conditions of the requirement are met, proving the software performs what the requirement states. A negative test case is an indirect test which performs conditions that should not invoke the requirement and verifies that this requirement is still met.

Test Implementation

After the test design, the test cases are then implemented using either the Unity Test Library (UTL) tool or the System for Making Automated and Random Tests (SMART) tool. Both of these tools are designed by St. Jude Medical to assist in performing automated test verification. The UTL tool is based on a C++ framework while the SMART tool is based on flow diagrams. In addition to the actual test implementation, part of the test implementation phase is to import the test cases into our test repository system in order to facilitate scheduled test runs of the test implementation. The software verification group uses an in-house tool called Software Automated Integrated Network Test System (SAINTS) for this purpose. More details about each of the tools are given below in the Methods and Material section.

Verification Test Review Process

There are different types of reviews that take place in regards to the test strategies, test designs, and test implementations. Informal reviews are optional while formal reviews are required for test designs and test implementations. The main difference between the two reviews are that formal reviews go through a much stricter process, and this review is filed away in the Design History File for the project and is included in our

submission process. Each review consists of a pre-review packet, an issues list, and also a post-review packet which contains all the corrections to the packet that were noted in the issue list [11].

Roles and Responsibilities

In these review meetings, there are four different types of participants. The Author is the creator of the document under review. After the review, the author examines the issue list and makes all the appropriate changes to the document. The Moderator is the individual who is responsible for making sure the review meeting stays on topic and maintains the flow of the review. If discussions get lengthy, the moderator is responsible for making sure that the issue is noted. The moderator is also required to verify that issues in the issue list are properly addressed. If the moderator does not agree with a resolution to an issue, the author must go back and investigate this issue again. Only when the moderator is satisfied with the disposition of the issue, is the review considered complete. The Recorder is the individual who collects all issues, questions, and concerns that are noted in a review and compiles it into an issue list. Since the role of the recorder is only present during the actual meeting, it is possible that this person could have multiple roles such as author in addition to their role as recorder. Lastly, there are also Key Reviewers who are responsible to ensure that the review has been thoroughly inspected and that they have intimate knowledge of the feature that is being reviewed. Key reviewers can be engineers from the requirements, software development, and/or system engineering groups.

Informal Reviews

An informal review is an internal review held among the software verification group. The purpose of this meeting is to review the completed review packet, to catch minor and recurring issues, and to look for missing test scenarios and discrepancies between the SRS and the test design. This type of review is optional, but it is always encouraged because it can prevent engineers from other groups from having to sit through a review that is full of minor and obvious issues. Although issues from this review are recorded, it is not necessary to track the resolution of the issues since these reviews are informal and are not part of our regulatory submission process.

Formal Reviews

A formal review is mandatory for all test designs and test implementations. All issues are recorded in the Review Database, and the moderator of the review verifies all resolutions in the issue list. The review database helps keep track of issues and is used as a formal method to document that the outlined processes in our guidelines have been followed. Formal Reviews that have too many issues may be changed to an Informal Review. By doing so, this allows the author to go back and address all the issues instead of making massive changes which do not get reviewed by the extended group. Once a Formal Review is accepted, the author then works on resolving all the issues that were noted. The moderator then verifies that the resolution was correctly resolved before the review is considered complete.

After a formal review is complete, all the issues that arise are addressed and tracked through a Software Work Request (SWR) order. The verification test designs and test implementations do not need to be reviewed again since the SWRs will document any changes.

Regression Runs

Throughout the software development process, there are several internal software releases. These internal software releases are iterative releases of key features in the product. Breaking the feature set into subsets, allows the software verification group to start their test implementation phase prior to the final completion of the software development cycle. With each internal software release, the features that are part of that subset go through a regression test. As more features are added, so is the number of test cases that are added to the regression run. The purpose of this is to ensure that with each release, which includes modifications and additional features, the software does not introduce new bugs. The software verification team tests these releases and executes the most up-to-date set of test cases. An additional benefit of running these regression tests is to provide a confidence in uncovering issues before the final run. Issues uncovered by the regression runs can be filed against the actual software itself or with the verification test cases. During this phase of the software test process, issues are reported to the respective groups so that they can be resolved in iterative releases of the software.

Formal Run

The Formal Run is the final test run that will have its results reported to the different regulatory agencies of each country in which the company is applying for approval of the device. An example of such an agency is the US Food and Drug Administration (FDA). This final run is done at the very end of the project life cycle. At this time, all requirements, software code, and software test cases have been formally reviewed. The formal test run consists of the execution of the full software verification test cases on the final software code and also on the final hardware configuration. Any issues or

discrepancies that arise during the formal test run require a SWR to be analyzed and disposed of properly. Possible dispositions of a SWR may be:

- to address the issue, which would lead to a new release of software and a re-run of affected test cases, or
- to postpone the issue if it is deemed to be non-critical and non-hazardous, or
- to consider the SWR as a non-issue.

The results of the Formal Test Run are stored into the test repository and these results are used to generate all the necessary reports.

Master Verification Matrix Report

Once all the test cases are either passing or have a SWR disposition of either postponed or non-issue, a Master Verification Matrix (MVM) report is generated. The SAINTS tool generates the MVM report, which is used to provide a traceability report linking each requirement to a particular passing test case. This is used to ensure that full coverage of the SRS was performed and that there are no missing test cases. The MVM report lists all the feature names, with their associated number of test cases, number of manual test cases, number of test cases which passed, number of test cases which failed, number of test cases with errors, number of test cases with warnings, number of tests without any results, and any SWRs linked to the features. Ideally, there should not be any test cases with failures, errors, warnings, or no results. However, if there are, there needs to be an associated SWR that describes the discrepancy, and this SWR must be in a terminal state as described above.

Software Verification Report

Finally, the final phase of the software verification process is the Software Verification Report (SVR). The SVR is a report that is generated to document the entire testing results. It is submitted to the different agencies when applying for approval of the device. The results that get reported in this document are the test results from the Formal Run along with the results from the MVM report.

The SVR lists all the feature names, with their associated Test Design Document name, test design baseline version, firmware version that was used for testing, and the pass/failure result. In addition to this, the SVR also includes a summary of the test results, a list of the issues and the corresponding SWR, and a statement that the verification test results demonstrate that the testing of the firmware requirements have been met and that there are no known significant issues in terms of patient safety or clinical efficacy. This statement indicates that the software version has successfully met the qualification requirements for human clinical use.

Design History File

The Design History File (DHF) is the centralized file system that stores all the final documents for a project. This repository consists of all plans, reviews, and final documents from all the different groups. These documents are the signed-off formal copies of the review packets, plans, and reports. Each item has an item number that is used to reference that particular document. In previous projects, the DHF was a physical file cabinet that held hard copies of these documents, but presently, these documents are stored electronically in an online database.

III. Methods and Material

SAINTS

Software Automated Integrated Network Test System [12], also known as the SAINTS tool, provides a system to run automated tests and also provides detailed reports about the status of test runs. This tool was originally designed for the purpose of allowing individuals the ability to log into this system and run test cases from their own computer and distribute the execution of the test cases over a network of test stations that are controlled by the SAINTS tool. However, as the tool matured, it took on more roles, such as adopting the ability to create test designs. Previously, test designs were created and maintained in a Microsoft Word document. Since the SAINTS tool has knowledge of each test case, a test designer was developed within the SAINTS tool to replace the older Microsoft Word documents. This Test Designer provided the verification engineer the ability to create new test cases and populate the following fields for each test case:

- Description of the objective of the test case
- Requirement coverage
- Test Procedure with expected results

Since SAINTS has knowledge of the requirements for each test case, the execution of the test case, and the results of the test case, the MVM reporting tool was also added to the SAINTS tool.

In regards to test execution, the SAINTS tool reduced the time it took to run test cases since the tool itself acted as a database where testers would be able to schedule test runs on stations that were available through the SAINTS tool. This meant that more time could be spent on analyzing results than having to manually schedule test runs. Also, by

combining all the test stations into a pool, this allowed for multiple testers to share a subset of custom test stations.

The SAINTS tool also provided real-time result reporting, execution results, test station availability, and MVM reports instantaneously since all these components were integrated into the SAINTS tool.

Unity Test Library

The verification test cases were then implemented using the Unity Test Library (UTL). The UTL is a collection of C++ system function calls that serves as an interface between the verification test cases and the software code. This method falls more in line with the typical software testing methodology and uses the C++ programming language in developing the test cases. These test cases would be linear test cases which provide some input conditions and pass and fail criteria through verification points in the code that a particular output has occurred.

Testers using this approach code C++ test scripts that execute commands on the test stations. An example of such commands is to generate a heart beat on the Heart Simulator to follow a particular pattern. Using the test library functions, the test script would then monitor if the software code on the device performs as specified by the software requirements. If these status checks are met, the test library would report a success, but if a certain time elapses, the test would be marked as a failure.

SMART

The System for Making Automated and Random Tests (SMART) tool [14] was another in-house tool that was designed as an alternative to traditional test designs and implementations though C++ test scripts. The ultimate goal of the SMART tool was to

combine the test design phase with the test implementation phase of the verification cycle. Combining the test design phase with the implementation phase optimized the testing approach and met the needs of a tool for easy maintenance across different projects as well as different iterations of the same project.

Figure 8: Sample SMART Test Case

The SMART tool has two components – the SMART Test Designer and the SMART Test Engine. The SMART testing methodology is centered on test designs that consist of a set of state diagrams, as shown above in figure 8. Using the SMART Test Designer, the tester constructs the state diagram by laying out the desired states and the

transition paths between states. Thus, the general flow of the test case would be to start at the initial state and then take a particular transition path when events occur until the test either reaches a success state or a failure state. The state diagram approach allows for multithreaded test designs as more than one state diagram can be executed at a given time. The SMART Test Engine optimizes the verification testing cycle by taking these same state diagrams and executing them on the test station, thus combining the test design and test implementation phases together. Another benefit of using the state diagram approach is that it allows for others who may not have a strong computer programming background, such as biomedical engineers, to be contributors to the software verification group.

SMART State Diagram

The basic structure of the state diagrams that make up the Test Design is composed of the following different states:

- Initial State
- Ordinary State
- Decision State
- Failure State
- Success State

Figure 9: Initial State

The Initial State, seen in figure 9 above, is the origin of a particular test design. In a single test flow, there is only one Initial State and it is where the test begins. This state is represented by a black-filled circle.

Figure 10: Ordinary State

The Ordinary State, seen in figure 10 above, may or may not contain an action item in it. Action items can consist of setting values to variables, programming parameters, or generating heart rhythms. This state is represented by a rectangle.

Figure 11: Decision State

The Decision State, seen in figure 11 above, contains expressions. Depending on the outcome of those expressions, a particular decision path will be taken as the result directs the flow of the test case in that direction. The decision state is represented as a diamond.

16 - Failure State

Figure 12: Failure State

The Failure State, seen in figure 12 above, is a terminal state that indicates a particular test case has failed. This state is represented by red circles with an X in it.

14 - Success State

Figure 13: Success State

Conversely, the Success State, seen in figure 13 above, indicates that a particular test case has passed. This state is represented by green circle with a check mark in it.

SMART vs. Unity Test Library

The SMART and UTL verification tools were both available for the SJM Confirm™ ICM project, also known as the AFM 1.0 Firmware project. As a software organization, we were looking at ways to streamline our testing, and this led us to decide upon a single test methodology for the Software Verification Organization. An objective of the AFM 1.0 Firmware project was to evaluate the different testing methodologies that existed at that time and to determine the direction that the Software Verification Organization would follow for future projects. There were other ongoing projects that were already using the UTL approach but the Software Verification Organization had yet to implement test cases using the SMART approach.

During planning for the feature testing of the AFM 1.0 Firmware project, the test features were divided into two groups – one group using the UTL approach and the other

group using the SMART approach. The test engineer assigned to each feature determined which testing approach to use. At the completion of the project, we compared the two tools based on these criteria:

- Did the tools ensure a high level of quality in the software?
- How did the two tools compare in terms of time it took to develop a test case?
- How easy was it to maintain the same test cases so that they can be reused for future releases?

Comparison Between the Two Tools

At the completion of the Software Verification phase of the AFM 1.0 Firmware project, the Software Verification Organization held a summit to review the results [14] of the SMART vs. UTL evaluation. For the AFM 1.0 Firmware project, there were 29 different test features consisting of the following:

- 12 features used the SMART test implementation method
- 13 features used the UTL testing method
- 1 feature used a combination of both testing methods

Breaking this down further, there were 301 test cases developed in SMART and 184 test cases developed in UTL (note 18 test cases were manual test cases and were not counted in this total).

The first criterion that we discussed was the level of quality of the two approaches. To evaluate the level of quality, the number of defects/SWRs filed against the software, the requirements, and the verification test cases was counted. The UTL approach recorded 47 SWRs, and the SMART approach recorded 41 SWRs. These figures were pretty close and indicated that overall the SMART tool performed similarly

to the UTL tool in terms of SWRs. This indicated that each method verified the software without any major differences in the number of defects, thus showing that both tools fared evenly for this first criterion.

The second criterion dealt with improving the development time it takes to implement a project. There were differences in the test features that were developed using the SMART tool and the UTL tool and their level of difficulties, as well as differences in the verification engineer's skill level and level of familiarity with the features. Unless we were able to take two equally skilled engineers and give them the same feature to develop test cases – one using the SMART tool and one using the UTL tool – it would be hard to evaluate this criterion because there is no real “apples to apples” comparison to determine which test method was faster in developing test cases. Because the UTL approach separates the design and implementation into two separate phases there is more potential for human error since the traceability between Test Design to Test Implementation is a manual process. There is a slight advantage to the SMART approach given that there are fewer reviews since the test design and implementation are combined.

Lastly, in the area of maintainability, the goal here was to determine how easy it would be to reuse test cases from one software release to the next. This is important because in between each software code release there needs to be some changes done by the testing tools to accommodate for changes to the software code. This maintainability criterion evaluated the speed in which it tools to be ready for regression test runs. In terms of the SMART tool, the SMART Tools Team generally can have it ready for testing within a few minutes after the new software code is released. The Verification &

Validation (V&V) Library Team performs the updates to the UTL, and it generally takes a few minutes to a day to get the UTL ready for testing after the new software code is released. Although there seems to be quite a difference in preparation time for the two tools, the V&V Library Team explained that the that they were able to generally have a turnaround of a few minutes too, but due to a lack of understaffing, had to push the release out the next day. So the actual amount of time it took for the V&V Library Team and the SMART Tools Team to make the changes to release and update the tool was about even. So this criterion ended up equal between the two tools.

Conclusion of the Evaluation

At the Software Verification Organization summit, the results of the evaluation between the SMART and UTL tools from the AFM 1.0 Firmware project were published [14]. The SMART tool provided some benefits over the UTL tool since it cut down the development time by combining the Test Design phase with Test Implementation phase. The SMART test cases also found similar and comparable issues as those found in the UTL test cases. However, after analyzing the results of the comparison, the overall objective of streamlining the verification process indicated that future verification projects should migrate to the UTL methodology since 3 other projects were already using this approach. Although the SMART tool was proven to be a viable substitute to the UTL tool, the foreseeable projects were already using the UTL test methodology, so there was no reason for the Software Verification Organization to continue using the SMART methodology.

IV. Results

The software verification portion of the SJM Confirm™ ICM project started in May of 2007 and concluded with the submission of the project to the FDA in May of 2008. The SJM Confirm™ ICM device received FDA approval four months later in September of 2008, and by the end of that year, there were well over 800 implants of this device [15].

Leading a group of 10 engineers, the team successfully executed the software verification for the SJM Confirm™ ICM device. Throughout this process there were milestones defined by the Software Verification Organization which helped track the completion of the project and helped ensure that the software requirements were thoroughly tested and that the device was safe for human use. These milestones included the test strategies, the test designs, the test implementations, and the test result report. Upon the completion of the project, all the individual components of each milestone were placed into the DHF, and these documents signaled the completion of the development phase of this project.

Master Verification Matrix Report

*Table 1: AFM 1.0 Firmware Master Verification Matrix Report
P/N – 40005718 Rev. A*

Feature	Test Cases	Manual Test Cases Total	Pass	Fail	Error	Warning	No Result	SWRs
ACTD	12	2	12	0	0	0	0	
AFD	65	2	65	0	0	0	0	
BAQ	5	0	5	0	0	0	0	
CC	7	0	7	0	0	0	0	
Conn	63	3	63	0	0	0	0	

Feature	Test Cases	Manual Test Cases Total	Pass	Fail	Error	Warning	No Result	SWRs
Diag	20	0	20	0	0	0	0	
Diag_AF	26	4	26	0	0	0	0	
Diag_LT	5	0	5	0	0	0	0	
Disc	19	0	19	0	0	0	0	
EGM	58	0	58	0	0	0	0	
FSDT	17	0	17	0	0	0	0	
IE	25	0	25	0	0	0	0	
IS	6	0	6	0	0	0	0	
MDbm	10	2	10	0	0	0	0	
MDsat	8	0	8	0	0	0	0	
MED	20	0	20	0	0	0	0	
Mrkr	10	0	10	0	0	0	0	
NSE	13	1	13	0	0	0	0	
Prog	10	0	10	0	0	0	0	
Rst	7	1	7	0	0	0	0	
SC	20	1	20	0	0	0	0	
SO	8	0	8	0	0	0	0	
TBA	27	1	27	0	0	0	0	
Tel	32	1	32	0	0	0	0	
UED	10	0	10	0	0	0	0	
	503	18	503	0	0	0	0	

Note: All SWRs were determined by CCB to have no effect on patient safety or clinical efficacy

From above, Table 1 is the MVM report [16] summary from the AFM 1.0 Firmware project. This report lists out each feature in column one of the table and then lists the following information for that feature in the subsequent columns:

- Total number of test cases
- Number of manual test cases

- Number of passed test cases
- Number of failed test cases
- Number of test cases with errors
- Number of test cases with warnings
- Number of test cases without any results, and
- SWRs that were filed during the formal run.

For example, for Activity Detection (ACTD) there were 26 total test cases with 4 of them being manual test cases. The rest of the columns beside the PASS column are zero or empty, which indicates that all 26 test cases passed without any issues. From Table 1, the total number of test cases executed in AFM 1.0 Firmware project is 503, with 18 of them designated as manual test cases. During Test Design reviews, test cases identified to be difficult to test are classified as manual test cases and are covered by firmware bench or unit testing. Many times these requirements are corner cases or scenarios that are difficult to generate on the test stations. As a result, manual test cases were created in order to achieve 100% coverage of software requirements. So in total there were 485 automated test cases in AFM 1.0 Firmware project.

*Table 2: AFM 1.0 Requirements Traceability Report
P/N – 40005718 Rev. A*

Feature	Requirements	Test Coverage	Pass	Fail	Error	Warning	No Result
ACTD	36	36	36	0	0	0	0
AFD	52	52	52	0	0	0	0
BAQ	23	23	23	0	0	0	0
CC	3	3	3	0	0	0	0
Conn	19	19	19	0	0	0	0
Diag	1	1	1	0	0	0	0
Diag_AF	80	80	80	0	0	0	0
Diag_LT	8	8	8	0	0	0	0

Feature	Requirements	Test Coverage	Pass	Fail	Error	Warning	No Result
Diag_SUM	55	55	55	0	0	0	0
Diag_TBAM	16	16	16	0	0	0	0
Diag_VHR	15	15	15	0	0	0	0
Disc	6	6	6	0	0	0	0
EGM	145	145	145	0	0	0	0
FSDT	5	5	5	0	0	0	0
IE	16	16	16	0	0	0	0
IS	5	5	5	0	0	0	0
MD	10	10	10	0	0	0	0
MDbm	38	38	38	0	0	0	0
MDsat	13	13	13	0	0	0	0
MED	33	33	33	0	0	0	0
Mrkr	35	35	35	0	0	0	0
NSE	36	36	36	0	0	0	0
Prog	9	9	9	0	0	0	0
Rst	92	92	92	0	0	0	0
SC	19	19	19	0	0	0	0
SO	10	10	10	0	0	0	0
TBA	109	109	109	0	0	0	0
Tel	14	14	14	0	0	0	0
UED	10	10	10	0	0	0	0

In addition to the MVM summary report [16], table 2 above displays the requirement coverage traceability report of each feature. This report lists all the features in AFM 1.0 Firmware project in column one of the table and then lists the following information for each feature in the subsequent columns:

- Total number of requirements in the SRS
- Number of requirements covered
- Number of passed requirements
- Number of failed requirements
- Number of requirements with errors
- Number of requirements with warnings, and

- Number of requirements without any results.

For example, for Atrial Fibrillation Detection (AFD) there were 52 total requirements in the SRS. All 52 requirements are covered by test cases that passed formal runs. In total there were 913 requirements for the AFM 1.0 Firmware project.

Software Verification Report

The Software Verification Report [17] for the AFM 1.0 Firmware project was signed off on April 29th, 2008. This report documented the results of the final test run on the final software code. The Software Verification Organization reported that all test requirements were covered with passing test cases. The report also found that there were zero software-related issues found during the final run. These results prompted the completion of the software verification portion of AFM 1.0 Firmware project.

V. Conclusion

The SJM Confirm™ ICM project was a highly successful project. It was the first cross-divisional project in SJM that leveraged the technology from the pacemaker and ICD platforms to develop a device that entered a whole new market. This device increased St. Jude Medical's portfolio by providing a minimally invasive diagnostic device. This device allows physicians to monitor individuals continuously and then determine if they need further treatment such as a pacemaker, defibrillator, or ablation.

AFM 1.1

Due to the popularity of the SJM Confirm™ ICM, work on the next software release of this device has already been completed and submitted to the FDA. The changes from the first release were minimal, and focused mainly in the area of improving the algorithms for detection and the storage of EGM records. Many of these enhancements were based off data from actual devices that were implanted in patients.

For the AFM 1.0 Firmware project, the sensing technology with the electrodes on the surface of the device was new. So after the initial release of the device, the software developers optimized the detection algorithms for the AFM 1.1 Firmware project using actual data collected from patients.

Patient data also indicated that there were a few instances where too many episodes were recorded due to the over detection of noise. Once the device detected the occurrence of noise, it ended the current arrhythmia episode recording because the noise detection considered the rhythm a false detection. This was ending the episode recording too early. Once the detection of noise ended, the arrhythmia would be detected again,

and this would trigger a new episode recording. As a result, this cycle increased the number of episodes recorded by the device, leading to a potential of lost episode records when the memory of the device filled up before the doctor got a chance to analyze it.

Future Work

Improvements are constantly part of the lifespan of any product. In fact, improvements are necessary in order to stay competitive in this type of market. The future of the SJM Confirm™ ICM is very broad, and future aspirations can be classified into the following major categories which are all somewhat dependent on each other:

- Performance and power efficiency
- Additional features
- Increased memory for storage

For future releases the performance and power improvements can be made to both the hardware and software components. An example of this type of improvement is with battery efficiency. Improving the battery efficiency allows for a longer longevity of the device as well as the ability to support more sensor features or even radio frequency communication to the device. Also, from a software standpoint, code optimization would free up some memory which can be in turn used to increase the amount of EGM records a device can store.

Improving the performance and power efficiencies of the device will allow for improvements of additional software features. Many of these algorithms already exist in SJM pacemakers and implantable cardiac defibrillators (ICDs). Features that may be in the pipeline are the addition of Radio Frequency (RF) communication, Morphology Scoring, and Exercise Compliance. Improvements to the battery efficiency will allow for

the RF communication while preserving the longevity of the device. RF communication allows for more frequent data collection via a bedside monitor without the knowledge of the patient, thus improving the quality of life. In addition to RF, other potential features are algorithms to help improve the detection of certain heart rhythms in order to better filter out muscle and external noise. One such feature used by ICDs is the detection algorithm called Morphology Scoring. The concept behind this algorithm is to use the shape of a normal heart rate as a template and compare each heart beat to this template. If the shape of the heart rate is not a close match, it helps to indicate that either there is noise or that the patient is in an arrhythmia. Another algorithm that can be migrated over is Exercise Compliance. This algorithm detects if a patient is exercising by monitoring the heart rate over a period of time. If exercise is detected, the algorithm will choose to ignore tachycardia rhythms (fast rhythms) and not record these since they are expected due to the exercise.

Lastly, increasing the physical memory on these devices will provide the user with the ability to store even more records. This allows doctors to retrieve even more episodes. More memory would also allow for other diagnostic data, such as body temperature or respiration rate, to be collected too. More information being captured allows for more data, which could potentially help develop new algorithms or improvements to current algorithms.

References

- 1) A drawing of the human heart. Subtle differences in shape affect blood flow. [Instructional Diagram]. (2009). Retrieved December 5, 2010, from: <http://planetearth.nerc.ac.uk/news/story.aspx?id=369>
- 2) Action Potential by Cardiac Region [SJM Training Document]
- 3) Mayo Clinic staff. (2009). Heart arrhythmias: Causes. Retrieved December 5, 2010, from: <http://www.mayoclinic.com/health/heart-arrhythmias/DS00290/DSECTION=causes>
- 4) Bernard J. Gersh, Teresa S.M. Tsang, Marion E. Barnes, and James B. Seward. (2005). The changing epidemiology of non-valvular atrial fibrillation: the role of novel risk factors. *European Heart Journal Supplements*, 7, C5-C11
- 5) Mayo Clinic staff. (2009). Atrial fibrillation: Symptoms. Retrieved December 5, 2010, from: <http://www.mayoclinic.com/health/atrial-fibrillation/DS00291/DSECTION=symptoms>
- 6) Andrew D. Krahn, MD, George J. Klein, MD, Raymond Yee, MD, and Caro Norris, RN. (1998). Final Results From a Pilot Study With an Implantable Loop Recorder to Determine the Etiology of Syncope in Patients With Negative Noninvasive and Invasive Testing. *The American Journal of Cardiology*, 82, 117-119
- 7) Fred Morady, MD, Hakan Oral MD, Aman Chugh, MD, Frank Pelosi MD, Frank Bogun, MD, Eric Good, DO, Krit Jongnarangsin, MD, Matthew Ebinger, DO, and Thomas Crawford, MD. (2009). The Treatment of Atrial Fibrillation. Retrieved December 5, 2010, from: http://www.med.umich.edu/cvc/services/site_electrophys/Treatment%20of%20Atrial%20Fibrillation.pdf
- 8) St. Jude Medical. (2008). *SJM Confirm ICM Specs sheet*. [Product Specification Sheet]
- 9) St. Jude Medical. (2008). *AFM 1.0 Software Verification Plan*. [SJM Part Number 40005313] Rev B
- 10) St. Jude Medical CRMD. (2006). *Unity Software Verification Test Design Guidelines*. [Internal organization guidelines]
- 11) St. Jude Medical CRMD. (2000). *Product Development Design Review*. [Departmental Work Instruction DWI 2.1.1.4] Rev A

- 12) St. Jude Medical CRMD. *Software Automated Integrated Networked Test System (SAINTS)*. [SJM internal website for the tool]
- 13) St. Jude Medical CRMD. *System for Making Automated and Random Tests (SMART)*. [SJM internal website for the tool]
- 14) Jimmy Duong. (2008). *FW V&V Test Process Analysis – SMART vs. UTL*. [SJM Presentation to the Software Verification Organization] Sylmar, CA
- 15) St. Jude Medical. (2008). *SJM Confirm Implants* [Internal SJM email]
- 16) St. Jude Medical. (2008). *AFM 1.0 Firmware Master Verification Matrix Report*. [SJM Part Number 40005718] Rev A
- 17) St. Jude Medical. (2008). *AFM 1.0 Software Verification Report*. [SJM Part Number 40005261] Rev A