UAH Fit Vault

Software Development Plan

16 October 2015

James J. Duggan IV

Glen L. Riden

Whit J. Sisulak

Timothy R. Wilkins

**Revision History**

|  |  |
| --- | --- |
| *Version* | *Author and Comments* |
| 0.1 | Glen Riden – Initial Draft |
|  |  |

Contents

[1 Scope 8](#_Toc432514533)

[1.1 Identification 8](#_Toc432514534)

[1.2 System Overview 8](#_Toc432514535)

[1.2.1 Data Collection 8](#_Toc432514536)

[1.2.2 Data Analysis 8](#_Toc432514537)

[1.3 Document Overview 9](#_Toc432514538)

[1.4 Relationship to Other Plans 9](#_Toc432514539)

[1.5 Glossary 9](#_Toc432514540)

[2 Referenced Documents 9](#_Toc432514541)

[3 Overview of Required Work 9](#_Toc432514542)

[4 Plans for performing general software development activities 10](#_Toc432514543)

[4.1 Software development process 10](#_Toc432514544)

[4.2 General plans for software development 10](#_Toc432514545)

[4.2.1 Software development methods 10](#_Toc432514546)

[4.2.2 Standards and practices for software products 10](#_Toc432514547)

[4.2.3 Traceability 10](#_Toc432514548)

[4.2.4 Reusable software products 10](#_Toc432514549)

[4.2.5 Handling of critical requirements 10](#_Toc432514550)

[4.2.6 Computer hardware resource utilization 10](#_Toc432514551)

[4.2.7 Recording rationale 11](#_Toc432514552)

[4.2.8 Access for acquirer review 11](#_Toc432514553)

[5 Plans for performing detailed software development activities 11](#_Toc432514554)

[5.1 Project planning and oversight 11](#_Toc432514555)

[5.1.1 Software development planning 11](#_Toc432514556)

[5.1.2 Software item test planning 11](#_Toc432514557)

[5.1.3 System test planning 11](#_Toc432514558)

[5.1.4 Software installation planning 11](#_Toc432514559)

[5.1.5 Software transition planning 11](#_Toc432514560)

[5.1.6 Following and updating plans, including the intervals for management review 11](#_Toc432514561)

[5.2 Establishing a software development environment 11](#_Toc432514562)

[5.2.1 Software engineering environment 11](#_Toc432514563)

[5.2.2 Software test environment 11](#_Toc432514564)

[5.2.3 Software development library 12](#_Toc432514565)

[5.2.4 Software development files 12](#_Toc432514566)

[5.2.5 Non-deliverable software 12](#_Toc432514567)

[5.3 System requirements definition 12](#_Toc432514568)

[5.3.1 Analysis of user input 12](#_Toc432514569)

[5.3.2 Operational concept 12](#_Toc432514570)

[5.3.3 System requirements 12](#_Toc432514571)

[5.4 System design 12](#_Toc432514572)

[5.4.1 System-wide design decisions 12](#_Toc432514573)

[5.4.2 System architectural design 12](#_Toc432514574)

[5.5 Software requirements definition 12](#_Toc432514575)

[5.6 Software design 12](#_Toc432514576)

[5.6.1 Software item-wide design decisions 12](#_Toc432514577)

[5.6.2 Software item architectural design 12](#_Toc432514578)

[5.6.3 Software item detailed design 13](#_Toc432514579)

[5.7 Software implementation and unit testing 13](#_Toc432514580)

[5.7.1 Software implementation 13](#_Toc432514581)

[5.7.2 Preparing for unit testing 13](#_Toc432514582)

[5.7.3 Performing unit testing 13](#_Toc432514583)

[5.7.4 Revision and retesting 13](#_Toc432514584)

[5.7.5 Analyzing and recording unit test results 13](#_Toc432514585)

[5.8 Unit integration and testing 13](#_Toc432514586)

[5.8.1 Preparing for unit integration and testing 13](#_Toc432514587)

[5.8.2 Performing unit integration and testing 13](#_Toc432514588)

[5.8.3 Revision and retesting 13](#_Toc432514589)

[5.8.4 Analyzing and recording unit integration and test results 13](#_Toc432514590)

[5.9 Software item qualification testing 13](#_Toc432514591)

[5.9.1 Independence in software item qualification testing 13](#_Toc432514592)

[5.9.2 Testing on the target computer system 13](#_Toc432514593)

[5.9.3 Preparing for software item qualification testing 14](#_Toc432514594)

[5.9.4 Dry run of software item qualification testing 14](#_Toc432514595)

[5.9.5 Performing software item qualification testing 14](#_Toc432514596)

[5.9.6 Revision and retesting 14](#_Toc432514597)

[5.9.7 Analyzing and recording software item qualification test results 14](#_Toc432514598)

[5.10 Software/hardware item integration and testing 14](#_Toc432514599)

[5.10.1 Preparing for software/hardware item integration and testing 14](#_Toc432514600)

[5.10.2 Performing software/hardware item integration and testing 14](#_Toc432514601)

[5.10.3 Revision and retesting 14](#_Toc432514602)

[5.10.4 Analyzing and recording software/hardware item integration and test results 14](#_Toc432514603)

[5.11 System qualification testing 14](#_Toc432514604)

[5.11.1 Independence in system qualification testing 14](#_Toc432514605)

[5.11.2 Testing on the target computer system 14](#_Toc432514606)

[5.11.3 Preparing for system qualification testing 14](#_Toc432514607)

[5.11.4 Dry run of system qualification testing 14](#_Toc432514608)

[5.11.5 Performing system qualification testing 14](#_Toc432514609)

[5.11.6 Revision and retesting 14](#_Toc432514610)

[5.11.7 Analyzing and recording system qualification test results 14](#_Toc432514611)

[5.12 Preparing for software use 14](#_Toc432514612)

[5.12.1 Preparing the executable software 14](#_Toc432514613)

[5.12.2 Preparing version descriptions for user sites 14](#_Toc432514614)

[5.12.3 Preparing user manuals 14](#_Toc432514615)

[5.12.4 Installation at user sites 14](#_Toc432514616)

[5.13 Preparing for software transition 15](#_Toc432514617)

[5.13.1 Preparing the executable software 15](#_Toc432514618)

[5.13.2 Preparing source files 15](#_Toc432514619)

[5.13.3 Preparing version descriptions for the maintenance site 15](#_Toc432514620)

[5.13.4 Preparing the “as built” software item design and other software maintenance information 15](#_Toc432514621)

[5.13.5 Updating the system design description 15](#_Toc432514622)

[5.13.6 Updating the software requirements specification 15](#_Toc432514623)

[5.13.7 Updating the system/subsystem specification 15](#_Toc432514624)

[5.13.8 Preparing maintenance manuals 15](#_Toc432514625)

[5.13.9 Transition to the designated maintenance site 15](#_Toc432514626)

[5.14 Software configuration management 16](#_Toc432514627)

[5.14.1 Configuration identification 16](#_Toc432514628)

[5.14.2 Configuration control 16](#_Toc432514629)

[5.14.3 Configuration status accounting 16](#_Toc432514630)

[5.14.4 Configuration audits 16](#_Toc432514631)

[5.14.5 Packaging, storage, handling, and delivery 16](#_Toc432514632)

[5.15 Software product evaluation 16](#_Toc432514633)

[5.15.1 In-process and final software product evaluations 16](#_Toc432514634)

[5.15.2 Software product evaluation records, including items to be recorded 16](#_Toc432514635)

[5.15.3 Independence in software product evaluation 16](#_Toc432514636)

[5.16 Software quality assurance 16](#_Toc432514637)

[5.16.1 Software quality assurance evaluations 16](#_Toc432514638)

[5.16.2 Software quality assurance records, including items to be recorded 16](#_Toc432514639)

[5.16.3 Independence in software quality assurance 16](#_Toc432514640)

[5.17 Corrective action 16](#_Toc432514641)

[5.17.1 Problem/change reports, including items to be recorded 16](#_Toc432514642)

[5.17.2 Corrective action system 16](#_Toc432514643)

[5.18 Joint technical and management reviews 16](#_Toc432514644)

[5.18.1 Joint technical reviews, including a proposed set of reviews 16](#_Toc432514645)

[5.18.2 Joint management reviews, including a proposed set of reviews 16](#_Toc432514646)

[5.19 Administrative security and privacy protection 16](#_Toc432514647)

[5.20 Software management indicators 16](#_Toc432514648)

[5.21 Administrative security and privacy protection 16](#_Toc432514649)

[5.22 Managing subcontractors 16](#_Toc432514650)

[5.23 Interfacing with software IV&V agents 16](#_Toc432514651)

[5.24 Coordinating with associate developers 16](#_Toc432514652)

[5.25 Project process improvement 16](#_Toc432514653)

[6 Schedules and activity network 16](#_Toc432514654)

[7 Project organization and resources 16](#_Toc432514655)

[7.1 Project organization 16](#_Toc432514656)

[7.2 Project resources 16](#_Toc432514657)

[8 Notes 16](#_Toc432514658)

[9 Annexes 17](#_Toc432514659)

*<This page intentionally left blank>*

# Scope

## Identification

*<This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s). >*

This document is designed to be applicable to the development stage of the UAH Fit Vault software system. The UAH Fit Vault software system will consist of two tools (data collection and data analysis) which will both be released (version 1.0.0.0.X) to the customer at the end of the development process. There will be no maintenance plan specified by this document, as it is not applicable to the scope.

## System Overview

### Data Collection

The data collection portion of this project will consist of the following. There are two different medical devices to be used for this project that record various types of data. The data provided by these devices consists of different file formats, and the data is different from device to device. The software will have to determine the contents of each file and how to process them. Due to how long data transfers take to download the data from a device, there may be a need to convert the data from a binary format to another format in order to speed up the process of getting data off the device. The software needs to able to take in files provided by the medical devices and be able to translate them in a way where they can be stored in a database. The software needs to run in the background of a PC and wait for files that need to be processed. The software will have to interact with a database to insert the data that has been processed in order for the data to be stored for later analysis. The software should allow for some basic configuration such as designating a folder on the PC to be a listener. Files moved or copied into this folder will be processed by the software when they are added. The software should have the ability to process multiple files if more than one is placed into the processing folder at a time.

### Data Analysis

Data analysis software needs to be created to analyze the data that is captured from the data collection tool mentioned above. This piece of software will be a separate stand-alone web application. The software needs to perform data analysis over different intervals of time such as one week, one month, etc. There will need to be some way to manage user access to the various medical data that has been inserted into the database that this software will access. Below are some proposed data analysis ideas that can be incorporated into the project.

• Simple Moving Average

• Data correlation discovery between the multiple devices.

• Possibly determine when an individual moves from walking to running or simply being able to identify the activities that were being performed while the data was being captured.

The data analysis possibilities will likely not fully be realized until the project team understands the different types of data that are available. Also, there will need to be collaboration with the customer for additions or changes to the data measurements provided by this software. The web application will have to have different levels of user access which will be defined later in this document.

## Document Overview

The purpose of this document is to detail the software development procedures to be utilized by the Medical Health group. The intended audience for this document is software developers, testers, customers, and any other stakeholders. The software will most likely require additional privacy and security protections due to the nature of the product.

## Relationship to Other Plans

*< This subclause shall describe the relationship, if any, of the SDP to other project management plans. >*

## Glossary

|  |  |
| --- | --- |
| Term | Definition |
| Software Item | An aggregation of software, such as a computer program or database, that satisfies an end use function and is designated for purposes of specification, qualification testing, interfacing, configuration management, or other purposes. Software items are selected based on trade-offs among software function, size, host or target computers, developer, support strategy, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors. A software item is made up of one or more software units. |
|  |  |

# Referenced Documents

• UAH Fit Vault Configuration Management Plan

• UAH Fit Vault Software Test Plan

• Microsoft Secure PW Guidelines (https://www.microsoft.com/security/pc-security/password-checker.aspx)

• J-STD-016-1995

# Overview of Required Work

*< This clause should be divided into subclauses as needed to establish the context for the planning described in later clauses. It shall include, as applicable, an overview of:*

*a) Requirements and constraints on the system and software to be developed*

*b) Requirements and constraints on project documentation*

*c) Position of the project in the system life cycle*

*d) The selected development/acquisition strategy; any requirements or constraints on it*

*e) Requirements and constraints on project schedules and resources*

*f) Other requirements and constraints, such as on project security, privacy protection, methods, standards, interdependencies in hardware and software development, etc.>*

# Plans for performing general software development activities

## Software development process

The software development process will follow an iterative development process. Each iteration will last approximately two weeks. At the end of each iteration, process will be documented, and plans will be updated accordingly.

The result of this project will be a final official release (version 1.0.0.0.X) to the customer consisting of data collection and data analysis tools.

## General plans for software development

### Software development methods

The software development will follow an iterative process. The software will be developed using mainly Visual Studio for C# support.

### Standards and practices for software products

The software source code written in will follow standard formatting practices for indentation.

Header comments will be located at the beginning of a file and preceding functions. The header comments at the beginning of files will consist of the date modified, author and a brief summary of the updates and/or additions. Header comments preceding functions will be brief when preceding the function prototype and consist of author, date, and a brief summary. A more detailed header comment should be provided preceding the function definition and include details such as author, date and pseudo-code (for complicated algorithms).

Naming conventions for variables, functions, and classes shall abide by standard naming practices.

### Traceability

As requirements are implemented in system design, there will be traceability that links the design item (diagrams, models, test cases, source code, etc.) to the requirement. The traceability will probably be maintained via a traceability matrix.

### Reusable software products

There are no known existing re-useable software products that will used in this project.

### Handling of critical requirements

There may be certain privacy and security critical requirements associated with this project due to the medical nature. As those specific requirements are identified, the team will conduct research in order to figure out how to satisfy the requirement.

### Computer hardware resource utilization

The SQL database storing the fitness information may need to be monitored to ensure that its size does not exceed virtual machine storage. This is out of the scope of this project, and the maintainer will be responsible for this.

### Recording rationale

The development process will be tracked through a GitHub repository. The customer will be given access to the GitHub repository in order to track the history of the project.

### Access for acquirer review

The software team will provide access to the data analysis tool and data collection tool and source code as requested by the customer.

# Plans for performing detailed software development activities

## Project planning and oversight

### Software development planning

The software development will be planned throughout the iterative development process. Each iteration will usually be two weeks long. If the software development plan needs to be changed, then those changes will be addressed at the start of an iteration.

### Software item test planning

Each software unit test should be developed and executed during the iteration that the unit is implemented.

### System test planning

The system test will be performed at the end of the project. System tests may also be performed on the data collection tool and data analysis tool individually, if necessary.

### Software installation planning

The system will be installed at the customer’s site when version 1.0.0.0.X is released.

### Software transition planning

The software will be transitioned to the customer at the end of the project. The end of the project will be signaled by the release of software version 1.0.0.0.X.

### Following and updating plans, including the intervals for management review

During development, plans will be updated at the start of each iteration.

## Establishing a software development environment

### Software engineering environment

The software engineering environment will consist of the project team’s development computers (personal computers).

### Software test environment

The test environment will consist of the customer’s web server to test the web analysis tool and the customer’s work computer to test the data collection tool. The web analysis tool test environment may be stored on a virtual machine for easy back-up and re-load.

### Software development library

There are no plans to use any software libraries for this project.

### Software development files

Software development source files, documents, models and diagrams will be stored on a GitHub repository.

### Non-deliverable software

There is no non-delivered software planned for the project.

## System requirements definition

### Analysis of user input

The project team met with the customer for two meetings to gather a list of desired features for the system. The project team as a whole translated these features in to requirements for the system. Then, the requirements were sent to the customer for review.

### Operational concept

### System requirements

After meeting with the customer, system requirements were generated by all members of the project team.

## System design

### System-wide design decisions

System-wide design decisions are usually proposed by one member of the project team and reviewed by the rest of the team. When performing system-wide design, the requirements must be reviewed to ensure that all requirements are satisfied by the proposed design decision.

### System architectural design

Architectural design decisions are made by the project team members that are associated with the subsystem that is being designed. When performing system architecture design, the requirements must be reviewed to ensure that all requirements are satisfied by the proposed architecture design.

## Software requirements definition

The software requirements were defined via the same method that the system requirements were defined. See section 5.3.

## Software design

### Software item-wide design decisions

The section shall follow the same procedures as 5.4.1.

### Software item architectural design

The section shall follow the same procedures as 5.4.2.

### Software item detailed design

The section shall follow the same procedures as 5.4.2.

## Software implementation and unit testing

### Software implementation

Software implementation should implement a single feature at a time. The CM plan details this process thoroughly.

### Preparing for unit testing

All modules should be evaluated by the author(s) to determine if unit testing should be performed on the module. All modules requiring unit testing should be designed during the software development process to support unit testing.

### Performing unit testing

Unit testing will be performed after a module has been fully developed. If the module is a large module, unit testing may also be performed throughout development.

### Revision and retesting

If a module fails unit tests, then the unit tests should be rerun on the module or the tests will be re-evaluated to ensure expected behavior is correct.

### Analyzing and recording unit test results

The UAH Fit Vault Software Test Plan details the analysis of the unit testing and the method of how the unit test results are saved.

## Unit integration and testing

### Preparing for unit integration and testing

When modules have been completed, they will be reviewed by another member of the project team. After the reviewer agrees with the module implementation and proposed testing, the module will be ready for integration and testing.

### Performing unit integration and testing

After modules have been reviewed and approved, they will be integrated in to the software by the branch master. The process of merging is further documented in the UAH Fit Vault Configuration Management Plan. The integrated module will then be tested.

### Revision and retesting

If a module fails integration testing, then the module will need to be examined by a second reviewer (not the author or initial reviewer) to ensure the implementation appears correct. If the implementation appears correct, then the expected test result will be examined to ensure it is correct. If the module appears to be implemented correctly and the expected test result is correct, then the project team will need to re-examine the proposed design.

### Analyzing and recording unit integration and test results

The UAH Fit Vault Software Test Plan details the analysis of the unit integration testing and the method of how the unit integration test results are saved.

## Software item qualification testing

### Independence in software item qualification testing

### Testing on the target computer system

### Preparing for software item qualification testing

### Dry run of software item qualification testing

### Performing software item qualification testing

### Revision and retesting

### Analyzing and recording software item qualification test results

## Software/hardware item integration and testing

### Preparing for software/hardware item integration and testing

### Performing software/hardware item integration and testing

### Revision and retesting

### Analyzing and recording software/hardware item integration and test results

## System qualification testing

The UAH Fit Vault Software Test Plan may be referenced to provide detailed information about the system qualification testing.

## Preparing for software use

### Preparing the executable software

### Preparing version descriptions for user sites

Since there is only one release planned (1.0.0.0.X) to be released to the customer, there will be a single version description provided.

### Preparing user manuals

A user manual will be provided to the customer after the software has officially been released.

### Installation at user sites

The software/system will be installed on the customer’s site when the software is complete.

## Preparing for software transition

### Preparing the executable software

### Preparing source files

Source files for the data collection tool will be provided to the customer at the end of development via a cabinet file (Windows or Winzip or 7zip). Source code for the web analysis tool will be provided to the customer in a separate cabinet file (Windows or Winzip or 7zip).

### Preparing version descriptions for the maintenance site

The version description document for the system maintainers will be prepared after software development is complete. The version description document will be delivered to the customer upon completion of the project.

### Preparing the “as built” software item design and other software maintenance information

Since maintenance is not in the scope of this project, it will be up to the customer to determine a maintenance plan.

### Updating the system design description

The Software Design Description will be updated at the start of each iteration if design changes are necessary. The Software Design Description will undergo a final review at the end of development before being given to the customer.

### Updating the software requirements specification

The Software Requirements Specification will be finalized after 6 January, 2016. The Software Requirements Specification will be given to the customer after it is finalized.

### Updating the system/subsystem specification

The system/subsystem specification will be updated at the start of each development iteration if necessary.

### Preparing maintenance manuals

Since maintenance is not in the scope of this project, it will be up to the customer to determine a maintenance plan.

### Transition to the designated maintenance site

Since maintenance is not in the scope of this project, it will be up to the customer to determine a maintenance plan. There is no plan to transition the system to any other site than the deployment system.

## Software configuration management

The configuration management for the development process is specified in the UAH Fit Vault Configuration Management plan.

## Software product evaluation

### In-process and final software product evaluations

### Software product evaluation records, including items to be recorded

### Independence in software product evaluation

## Software quality assurance

### Software quality assurance evaluations

### Software quality assurance records, including items to be recorded

### Independence in software quality assurance

## Risk management

All risks will be managed and assessed at the beginning of each development iteration.

## Administrative security and privacy protection

The software source code will be restricted to be viewable by only the project team, the customer, a maintainer when the software is transitioning in to maintenance, and the professor of the class.

# Notes

# Annexes