SYSTEMS TESTING

1. What is Systems Testing?

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input.

High-level requirements/design tests (Software System Tests) conducted by Independent Verification Engineers verify that the device accurately implements the Software Requirements Specification (SRS) and/or Software Architecture (SA).

2. Systems Testing at General Digital (GD)

The following describes our process of creating tests from customer or development provided documents, running those tests, collating the results and evidence for review, and implementing review changes:

- What documents to use when designing a test
- How to create and administer a test
- How to organize the results, and the evidence of those results
- How to submit the completed tests for review
- How to implement the review

2.1. Responsibilities

The responsibilities for handling the System Test Process are divided between a Project Coordinator, the System Test Engineers, and Internal Peer Reviewer. All information and work products related to module testing should flow between the three contacts through phone calls and e-mails. All e-mails will be retained on GD systems per data retention procedure.

2.1.1. Project Coordinator

The Project Coordinator is responsible for selecting the correct documentation for the System Test Engineers and for finalizing the project for delivery.

2.1.1.1. Document Selection

The Project Coordinator takes various documents from the GD Software Services Development (GDSSD), such as change orders, and System Reference and Architecture and submits them to the System Test Engineers for use.

2.1.1.2. Project Finalization

All the output deliverables are transferred to a secured location on the server upon successful release and is configured in our Quality Management System (QMS) system for formal release. The expected output deliverables are documented in the specific project software validation plan.

2.1.1.3. Change Analysis

In the event a Change Driver leads to changes in the code for a project, not every test may need to be redone. The Project Coordinator should determine which tests are affected by a change to the code base and which are not. This can be done broadly (i.e., all tests dealing with a certain screen), or narrowed down to specific tests.

2.1.2. System Test Engineer

The System Test Engineer is responsible for taking the customer documentations, identifying actionable, testable requirements, as well as creating and performing the system tests. They are also responsible for identifying possible discrepancies from the requirements while creating and performing said tests.

2.1.2.1. External Requirements

- New software release as identified by the Project Coordinator, such as Software Requirements Specification (SRS)
- Change Drivers that occur during testing such as JIRA tasks, SRS revisions or Change Orders submitted by the customer

2.1.2.2. Test Environment

- All projects should have a directory associated with them.
- This directory will house the Evidence folder, the Test Document, Requirements that the tests are derived from and a current version of the software under test.
- Tests will require a means of capturing evidence, such as by reporting log files, screen shots, or a camera to take pictures and videos.
- MS Excel spreadsheets/macros to assist with test script generation
- Text editors (XEMACS, EditPlus, etc.)
- Additional testing materials, such as USB drives or Compact Flash cards may be required for individual projects

2.1.2.3. System Test Design Methodology

The system testing technique used for this project aims to ensure that the project holistically performs as specified by the requirements identified by the customer and the project director such as the SRS documents.

While specifics of test creation will vary depending on project, the following should serve as a general guideline for creating effective test cases.

A. Identify Testable Requirements

Read through the relevant documentations and identify which requirements are fit for system testing. Requirements that do not fit under the purview of system testing should be identified, analyzed, and submitted to the Unit Testing section.

Consider the following requirements:

Req. 1: Product shall display a logo upon start-up

Req. 2: Product shall have variable *x* in *example.c* as a counter for process *y*

Req. 3: Product shall refresh *z* every 10 ms

In the above examples, Req. 1 should be identified as a testable requirement. A test case can be created and performed based on what is listed.

Req. 2 would require analysis of the source code and would be deemed under analysis, likely to be relegated to unit testing to cover.

Req. 3 would also require analysis, as it would only be feasible depending on the product and additional tools available for the system test.

B. Construct Test Cases based on Identified Requirements

Once testable requirements are identified, construct test cases aimed at verifying the identified requirement.

In general, a test case should consist of the following elements:

- ID: The test ID, useful for quick identification and for linking requirements to the correct test case.
- Test Method: the steps required to perform the test. The test method should also detail how the requirement is being verified.
- Acceptance Criteria: The results that the test case must produce for the test to pass.
- Test Date: The date the test is performed
- Test Disposition: The result of the test, typically denoted as PASS/FAIL
- Notes: A field for additional comment for the test. This will generally be used to denote the appropriate evidence.

2.1.2.4. System Test Execution

Once a test is created, the method should be attempted as written with the actual device. If there are issues that arise during the testing, the test method can be amended—if it continues to fulfill the Acceptance Criteria of all linked requirements that test was created to satisfy. During the execution of the test, evidence will need to be collected to verify the results. These include, but are not limited to, pictures, videos, and log files.

If, during the execution of the test, it is found that the results do not match the acceptance criteria, the test will be marked as a failure, and a task will be created in JIRA.

If the results do match the acceptance criteria, the test will be marked **Pass,** noted with the date and version of software used, and marked as **Awaiting Review.** All evidence will be noted, so that it can be easily found by the reviewer.

A document should be maintained cataloging test results, what machine was used to perform the test, the software version, the date of the test, what the expected results were, and if the test was passed or failed. It should also note the name and location of evidence, and if it has been reviewed or not.

2.1.2.5. Formatting and Handling of Evidence

The formatting and handling of evidence should adhere to the following:

- While any name can be used if the test files are properly linked through the Test Document to the tests they are evidence for, it is best to give evidence names evocative of at least one test they are associated with, or what the evidence contains.
- Evidence should be as concise as possible while still proving all necessary criteria.
- Any test can require multiple filetypes of evidence.
- A particular piece of evidence can be used for multiple tests.
- Evidence should be kept in a single location, or in subfolders off a single Evidence root.
- On updating to new versions, old evidence should be archived. If there are multiple small release candidates that shouldn't force a repeat of certain old tests, the evidence can be broken up into subfolders to make clear what version of the software it was performed under.

2.1.2.6. Discrepancy and Error Tracking

When the results of a test do not match the behaviors described in the documentation, there is an error. There may also be an error if there are obvious problems that occur that seem like they would prevent the successful operation of the machine. The first errors should be marked as Failed tests on the tracking sheet, and all errors should be reported via JIRA.

If an error that is not directly linked to any testable object is located, a test may need to be created for it, linking to the JIRA tag instead of the documentation.

2.1.3. Quality Assurance

Quality Assurance is responsible for reviewing all completed tests, ensuring they correctly prove the intended function, and for writing up descriptions of the failures of those that do not.

2.1.3.1. Review Process

The Reviewer will have access to the methodology of the test they are reviewing, its expected outcome, actual outcome, and all evidence. They will need to review each test, which should in some way be denoted as needing review. If an issue with the test is found, such as a discrepancy between the method and the documentation it links to, or between the listed results and the provided evidence, a review should be written, preferably in a provided space in the same document as the test information; otherwise, somewhere the System Test Engineer will have access to it.

The Reviews will be viewed by System Test Engineers who will accept or reject them. If accepted, the test in question will be altered and possibly repeated to incorporate the Quality Assurance person's findings. If rejected, a reason will be listed as to why. Either way, the Quality Assurance person must then examine the response to their review, and the test or retest if redone, to see if it now passes. If not, it will once more be marked with issues and listed as reviewed; if so, it should be marked as having passed review.

2.2. Process

- The Project Coordinator receives documents from the customer or development, and passes the necessary ones on to the System Testers.
- The System Testers create a Test Document, populated with tests, from these documents.
- The Project Coordinator passes along any Change Drivers that are not immediately visible to the System Testers throughout the process.
- The Project Coordinator should verify if any changes to the code base require the re-run of tests already performed for the upcoming release.
- Once the System Testers have their tests ready, they can begin running the tests.
- As tests are run, errors are noted using JIRA.
- Once there are some completed tests, the Reviewer can begin reviewing the tests, their evidence, and the results to determine if everything was done correctly.
- Once all tests have been completed and reviewed, and all JIRA issues addressed, the project can be submitted by the Project Coordinator.

2.3. Archival of Data

Evidence, test documents, and old versions of the documentation should be archived in its own subfolder off the directory mentioned in 2.1.2.2. Subfolders should be clearly marked for which version they correspond.

2.4. Expectations Regarding Repeatability of Tests

Tests should be repeatable by simply clearing their results, test date and review status and repeating the steps as listed. Additional steps or considerations noted during each individual test, such as preconditions that are required and not obvious, should be added to the steps.

2.5. Updating and Changing Tests

If new versions of the Product are created, new entries in the documentation will need to be created, new tests created, and old tests linked to sections changed will need to be updated. Testable objects that are removed will need to have their associated tests removed or altered. Also, Change Orders and JIRA tasks can lead to the creation of new tests beyond what is in the original documentation.

2.6. Witness Tests

Witness testing is a form of product verification where a technical specialist ("witness") confirms that a supplier follows through with the (required) factory-ordered acceptance tests. This testing is a form of quality control and surveillance.

3. Software System/Integration Test

- » Description
- » Hardware
- » Interface/Software Support
- » Results
- » Transition Criteria
- » Exit Criteria
- » Traceability

3.1. Description

The Independent Verification Engineers, as outlined below, will conduct high-level requirements/design tests (Software System Tests) to verify that the device accurately implements the SRS and/or Software Architecture (SA).

3.2. Hardware

Below is a basic example of what a high-level requirement looks like.

System	Version	Purpose	Supplier
Medical Device	Current Release Version	Product under test	Customer

3.3. Interface/Software Support

No additional software is used other than the software for the system (e.g., medical device).

3.4. Results

Test procedures will be created against every high-level requirement listed as TEST. The following is a list of all the information recorded:

- › Program Name
- › Release Version
- › Test Date(s)
- › Tester Name
- › Trace matrix between requirement and test procedure step
- › Overall UT status (Pass/Fail)
- › If issues exist, a JIRA task number
- › Reviewer Name

For each test step executed will have a PASS/FAIL status.

If errors are detected, a JIRA task will show the test procedure step that FAIL. The tester will provide a brief explanation of the error in the problem description section and, if appropriate, the design documentation can be marked and attached to indicate the nature of the error.

3.5. Transition Criteria

The Software System Tests shall begin when the high-level requirements are complete and a formal release from Software Development Team (SDT) is produced.

3.6. Exit Criteria

Software System Tests is considered complete when all high-level requirements have been tested and **PASSED.** Any outstanding failures will be noted in the Unresolved Anomalies report.

3.7. Traceability

Each high-level requirement will have a unique identifier. Each test procedure step created will trace to this unique identifier. Traceability within DOORS NG Requirements tool will be updated for each iteration of test.

4. Witness Testing

4.1. Witness Test Definition

Witness testing is a form of product verification where a Technical Specialist ("witness") confirms that a supplier follows through with the (required) factory-ordered acceptance tests. This testing is a form of quality control and surveillance.

5. Functional Testing

5.1. Functional Test Definition

Functional testing is a form of testing and is a quality assurance process that helps to validate the system or components against various functional specifications and requirements outlined. Functional testing is a type of black-box testing as the source code of the application is not considered during the testing process.

5.2. Functional Testing (GD)

5.2.1. Functional Testing Summary

For our clients, we usually get either new requirements or changes to existing requirements via dev, we then create tests to perform on the system itself to cover those requirements. We then perform the tests, take objective evidence, mark down the passes and fails, then ship it to the customer.

5.2.2. Research Testing Methods

The following spreadsheet is maintained during all functional testing for a sample client. An example of each item contained is given.

ID	Validation Item	Primary Text
99999	Operations upon detecting abnormalities	The device shall be separated from the patient upon detection of an abnormality

Test Method	Acceptance Criteria
• Operate the machine in the treatment process • Operate the machine in the treatment process using auxiliary equipment • Open the flowchart screen	• Shall emit an alarm • The indicators turn blue (OFF) on the flowchart screen

Test Results	Test Disposition	Test Date
Expected error displayed, and device stopped as it should	Pass	2022 Jun 29

Notes	Process	DUT
Screenshots as follows: 321313-01.bmp 231123-02.bmp 441234-03.bmp 224531-04.bmp	Healthcare	A

GENERAL DIGITAL
An SBA Small Business Concern
60 Prestige Park Road
East Hartford, Connecticut 06108
Phone 860.282.2900 **Toll-Free** 800.952.2535
E-mail gdc_info@generaldigital.com
Web www.generaldigital.com