Starter Kit
Autodoc

Gary A. Howard

June 1, 2018

Abstract

This document demonstrates how-to use (autodoc) a LaTeX documentation framework. autodoc declares this a tool-test document type.

Contents

1 Overview

Starterkit is a document that is used to demonstrate how autodoc can be used to setup a unit test, verification test, a tool validation test document, or a test record document.

An autodoc report consists of two distinct parts: 1) text and 2) a program. Text describes requirements, test expectation, and test results, and a program automates running Test Plans, Test Suites, and Test Steps that are compliant with the test-output factory specified by the Amber driver.

The goal of this report is to demonstrate how to specialize autodoc for your specific purposes. This report does not hook into Amber . Amber has a similar report that describes the requirements for the input factory and the test-output factory .

Several sections in section 1 will seam out of place, and perhaps will read akward. I have included them so that you will see autodoc framework reuses documentation parts.

1.1 Starterkit.tex

The contents of Starterkit.tex are listed below. Page layout, headers, footers, and table of contents is driven by the LaTeX document class tlc-article .

```
\documentclass[10pt]{tlc-article}
\begin{document}
  \tlcTitlePageAndTableOfContents
    {Autodoc \\ Starter Kit}
    {Gary A. Howard}
    {This document demonstrates how-to use (\skAutoDoc) a \LaTeX\
    documentation framework. \skAutoDoc declares this a tool-test document
    type.}

  \input{\toolTestDir/main.tex}

  \makeatletter
  \tlcDebug
  \autodocDebug
  \makeatother

\end{document}
```

1.2 tlcTitleAndTableOfContents

tlc-article provides the tlcTitleAndTableOfContents to provide the document title, table of contents, author name, and document abstract that you read on the first page.

1.3 additional-layout

tlc-article provides a hook to data/additional-layout.tex to allow you to specialize your documents layout. The primary contents of this documents data/additional-layout.tex file is shown below.

```
\def\autodocDir{..}
\input{\autodocDir/boilerplate/project-directories.tex}
```

autodocDir is the location on your file system of the autodoc repository. project-directories.tex is an autodoc file that defines locations of files within the autodoc folder.

1.4 main

All text files found in autodoc/boilerplate are common to all autodoc documents. Each document consist of four major sections:

Overview: This section provide an overview of the document. It has been designed with quality and regulatory compliance in mind. The goal is to assemble documents that a suitable for your Design History File, or may be used when submitting records to the Food and Drug Administration.

Requirements: Identifies the requirements of the document. This may be user needs, products requirements, or testing requirements.

Test Plan Overview:
This section will vary based on the type of testing that is being described. This section will contain test evidence when combined with Amber .

Conclusion: This section is the conclusion you write after you have reviewed your test evidence. By default, autodoc , provides a default conclusion that you override by placing a file named Conclusion.tex in same directory as Starterkit.tex .

autodoc provides three document templates.

boilerplate/tool-test :
This template was designed to standardize Tool Validation .

boilerplate/unit-test :
This template can be used when your project requires Unit Test .

boilerplate/ver-test :
Verificaiton Test template is use do when you need to record Unit Test results.

The term Test has been abstracted to mean any test type including: unit test, verification test, validation test, integration test and tool validation test.

1.5 tlcDebug

tlcDebug is a command provided by tlc-article to assist you when you are having difficulty with LaTeX text expansion. You will notice the final sections of this document has debugging information.

1.6 autodocDebug

autodocDebug is similar to tlcDebug with a focus on debugging autodoc documents.

1.7 Purpose

This Report Package is a detailed record that provides a Configuration Item overview, a list of its IUR, one or more Test Reports, evidence the Test Reports ran along with the output produced by the Test Report including pass/fail result for each Test Step and Test Report, a statement indicating the Configuration Item has a Configuration Identification, and a conclusion that the Configuration Item has been validated for its intended use.

1.8 Scope

This Report Package applies to a Company medical device software projects that have determined a Configuration Item must be validated for its intended use. This Report Package covers activities associated with validating a Configuration Item for its intended use requirements.

1.9 Tool Validation Objectives

1. Describe the intended use of the tool.
2. Set the purpose and scope for the tool validation effort.
3. Enumerate intended use requirements.

4. Disclose compliance criteria.
5. Define Tool validation acceptance criteria.
6. Identify responsible persons and their roles.
7. Document required deliverables.
8. Define specific test steps and test steps to confirm that the Tool's intended use requirements have been met.
9. Collect test evidence.
10. Record Tool validation conclusion.

1.10 General Terms

Configuration Control The systematic process for managing changes to and established baseline.

Configuration Identification A unique identifier used to associate a collection of software artifacts.

Configuration Items Software source code, executables, build scripts, and other software development and software test artifacts relevant to creating and maintaining a software project.

Configuration Status Accounting The recording and reporting of the information needed to effectively manage the software and documentation components of a software project.

Report Package A detailed record that provides a Configuration Item overview, a list of its Intended Use Requirements (IUR), one or more Test Reports, evidence the Test Reports ran along with the output produced by Test Report including a pass/fail result for each Test Step and Test Report, and a statement indicating the Configuration Item has a Configuration Identification, and conclusion that the Configuration Item has been validated for its intended use.

Test Plan A test plan is a collection of one or more test suites a tester has determined to use to challeng requirements.

Test Suite A test suite is a collection of one or more test cases a tester has determined to use to challeng requirements.

Test Case A test case is a set of conditions under which a tester will determine whether the test is working as it was originally established for it to do.

Test Step A unique test identifier with predetermined expectation, confirmation criteria, and pass/-fail result.

Test Report A test report consists of Detailed instructions for the set-up, execution, and evaluation of results for a given test. The test protocol may include one or more test cases for which the steps of the protocol will repeat with different input data. Test cases are chosen to ensure that corner cases in the code and data structures are covered. A test protocol may be a script that is automatically run by the computer.

1.11 Configuration Item Specific Terms

TBD You should add terms that are specific to your Configuration Item or delete terms.tex.

1.12 General Acronyms

FDA Food and Drug Administration

IUR Intended Use Requirements

LMS Learning Management System

SOP Standard Operating Procedure

SOUP Software Of Unknown Provenance

1.13 Configuraiton Item Specific Acronyms

n/a Not applicable

1.14 Tool Compliance Criteria

Configuration Items are often used to assist with formal verification and validation, and for the management of configuration. All such tools must be validated for intended use or rationale provided otherwise. Throughout this Report Package the term Tool is synonymous with Configuration Item.

This Test Plan is compliant with:

1. SOP-PRC02004 Software Development Procedure
2. SOP-PRC02004 Software Development Procedure; Software Configuration Management
3. 21 CFR 820.70(i) utomated Processes, and General Principles of Software Validation

1.15 References

1. SOP-PRC02004 Software Development Procedure
2. SOP-PRC02004 Software Development Procedure; Software Configuration Management
3. 21 CFR 820.70(i) utomated Processes, and General Principles of Software Validation
4. General Principles of Software Validation; Final Guidance for Industry and FDA Staff; January 11, 2002

1.16 Training

Company's Learning Management System (LMS) is used to record all training records. Additional training is not required for one to use this Tool or run this Report Package. It is expected that the individuals conducting the tests are experienced in use of the process, subsequently not requiring prescriptive setup information in conducting the validation.

1.17 Tool Validation Test Approach

This Test Plan describes a series of Test Suites, Test Cases, and Test Steps When executed, each Test Step determines if the Configuration Item satisfies one or more software requirements. When a Test Step indicates that the software requirements are satisfied, the Test Step's result is "pass". Otherwise, the Test Step's result is "fail". The computer records all "pass" and "fail" results in the Test Plan record. The Configuration Item is considered verified when all Test Steps are executed and the Test Plan record contains no "fail" results. Each Test Step that results in a deviation, observation, incident, or failure shall be represented in the final report.

1.18 Configuration Management

When a Configuration Item is changed by its manufacturer or by a Company, a Company will review the manufacturer's release notes or a Company's design history file to determine if regression testing or adjustment to this Report Package is necessary, and re-run the Report Package to confirm the manufacturer's changes or a Company's changes do not impact product operation, product quality, or quality decisions made before upgrading to a new release of the Configuration Item.

1.19 Test Plan Instructions

This Test Plan describes Test Suits, Test Cases, and Test Steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Plan describes any setup criteria needed to conduct the Test Steps.

Each Test Plan contains a list of IUR and the Test Steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Step is marked passed or failed as it is completed. Each Test Plan is marked passed when all Test Steps pass or failed if a single Test Step fails. This serves as a record of completed Test Plans and Test Steps.

Test Plans are automatically run by the computer generating a report in PDF format. Test Plans are code-reviewed prior to use using the same code-review practice that is used for the medical device software. Test Reports are stored in the Configuration Management tool that is used by the medical device software. Test Steps that fail during computer execution or observations made during document routing, result in tickets being created to address the issue. This results in another execution of the Report Package and document routing.

When it becomes necessary to annotate a computer generated document, a Company's documentation practices must be followed. The documentation practices are summarized below:

1. When necessary to write down test values, they are written using an ink pen.
2. Any cross-outs of results or modifications to any test step shall be initialed and dated.
3. All appropriate blanks shall be filled in or annotated as "N/A" (not applicable).
4. No whiteout or other text-obscuring technique is to be used.
5. Any annotation must be signed and dated.

In most cases, a Test Plan is to be judged as "fail" if any Test Step within that Test Plan fails. If the failed Test Step does not impact proper evaluation of the safety and effectiveness of the device, the Test Plan in which the Test Step failed may be judged "pass". For such failed Test Steps a rationale will be noted in the comments section and in the final Report Package. If a failed Test Plan is re-executed a record of the failed results are included with the Test Plan when it passes and attached to the Test Plan. If a Test Plan is re-executed a new report is created and routed for approval in the document management system.

1.20 Test Plan Storage and Review

This Test Plan is part of a Company's automated validation framework. The framework consists of following parts:

LaTeX files are used to provide an Abstract, Introduction, Intended Use Requirements, Test Plan Overview, Test Equipment, Configuration Item Validation, Conclusion, and Change Summary. LaTeXfiles are converted assembled into PDF documents. PDF documents are routed using the Company's document management system for approval.

Ruby software is used to run the automated framework to collect test evidence.

Git is used as the storage repository for LaTeX& YAML files, a Git pull-request is used to review the LaTeX& YAML files prior to use.

Evidence Test Plan output includes one Test Suite, Test Plan, and Test Step, and Test Evidence.

YAML files define the Test Plan, Test Suite, and Test Steps that are processed to generate test evidence.

2 Requirements

Intended-use requirements are defined using the following story format:

As a <type of user>, I want <some goal> so that <some reason>

IUR00 As a model developer, I want to know my tool has been installed so that I can run my validation protocol.

3 Test Plan Overview

This section describes Test Plans, Test Suites, Test Cases, and Test Steps that demonstrate how a Configuration Item satisfies the IUR. Each Test Plan describes any setup criteria needed to conduct the Test Steps. Each Test Plan contains a list of IUR and the Test Steps that demonstrate how the Configuration Item satisfies the IUR. Each Test Step is marked passed or failed as it is completed. Each Test Plan is marked passed when all Test Steps pass or failed if a single Test Step fails. This serves as a record of completed Test Plans and Test Steps.

Each Test Plan is described in its own section. The order the Test Plans are listed is the order they are run. Each Test Plan defines:

name Each Plan, Suite, and Case has a unique name.

purpose Each Plan, Suite, and Case has a purpose.

Test Steps Each step has a confirmation and expectation along with the command needed to challenge the UIR.

Objective Evidence A record the Test Plan was run along with any evidence collected while the Test Steps were run.

Traceability Suites and Cases are traced to an IUR that is challenged. IUR can be traced to multiple Suites and Cases.

Each Test Plan, Test Suite, Test Case, and Test Step has been designed to be run by the computer. However, a person may choose to manually run the Test Plans, save the test results, and generate this test report as specified in the appropriate design documentation.

The example below runs two commands: 1) git help and 2) cat /gitconfig. The output from both commands are written to the console

```
plan:
  name: A Test Plan Name
  purpose: purpose of the plan

suite:
  name: A Test Suite Name
  purpose: a suite purpose
  requirement: UIR01 and UIR02

  - case:
    name: A Test Case name
    purpose: A Test Case purpose
    steps:
      - confirm: Confirm git help is written to the console output.
        expectation: Git help is displayed.
        command: git
        argument: help

      - confirm: Confirm .git config is written to the console.
        expectation: .gitconfig is written to the console output.
        command: cat
        argument: .gitconfig
```

4 Prerequisites

Note: These prerequisites are just example of what you might need to declare to the person reviewing or running this protocol.

1. A valid Git username and password. Request these from your Git administrator.
2. A private / public SSH key.
3. Your public key needs to be upload to your Git account. Request assistance from your Git administrator.
4. Administrative rights to your computer. Contact your computer systems administrator.

5. Your computer requires HTTP, HTTPS, and SSH access to the Internet. Contact your network administrator.
6. Configure your /.ssh/config file to reference a Git host. Request this information from your network administrator. Update your /.ssh/config file with information provided by your network administrator.

```
# Host alias to GitHost
Host GitHost
        HostName githost.amgen.com
        Port 12345
        User validStashUsename
        IdentifyFile ~/.ssh/css_rsa
```

5 Test Evidence

The Company's automation framework assembles the content in this section. The secion has one or more Test Plans, Test Suites, Test Cases, and Test Evidence. The evidence provided is used to conclude the Tool has meet the Intended Use Requirements.

6 Configuration Item Conclusion

This Report Package has satisfied the IUR for the Configuration Item described herein thus the Configuration Item is considered validated for its intended use.

Change Summary

Change	Justification
Initial version.	New document.

7 tlc-article Debug

7.1 tlc-article default files

tlc@location: data
tlc@additionalLayout: data/additional-layout.tex
tlc@headerFooter: data/header-footer.tex
tlc@versionFile: data/version.csv
tlc@logoFile: data/logo.png
tlc@versionFile: data/version.csv

7.2 tlc-article file hooks

tlc@additionalLayout: DEFINED
tlc@headerFooter: NOT DEFINED
tlc@logoFile: DEFINED
tlc@versionFile: DEFINED

7.3 tlc-article header and footer hooks

tlcVersion: v1.0.1
tlcDate: March 25th, 2018
tlcStatus: Released
tlcInstitution: Traap
tlcPermission: BSD-3-Clause

8 autodoc debug

8.1 autodoc definitions

autodocDir: ..
boilerplateDir: ../boilerplate
dataDir: ../data
placeholderDir: ../boilerplate/placeholder
sharedDir: ../shared
testRecordDir: ../boilerplate/test-record
toolTestDir: ../boilerplate/tool-test
unitTestDir: ../boilerplate/unit-test
verTestDir: ../boilerplate/ver-test
boilerplateSpecificTestDir: ../boilerplate/tool-test