

THE NORTH CAROLINA STATE UNIVERSITY

DEPARTMENT OF NUCLEAR ENGINEERING

REACTOR DYNAMICS AND FUEL MODELING GROUP

**NC STATE**  
UNIVERSITY

---

# CTF Software Development Lifecycle Procedures

---

**Robert K. Salko**  
*Staff Research Scientist*  
Oak Ridge National Laboratory  
salkork@ornl.gov

March 4, 2023

Rev. 1

**Prepared By:**

**Signature**

**Date**

Robert K. Salko

\_\_\_\_\_

\_\_\_\_\_

CONTENTS

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                    | <b>1</b>  |
| <b>2</b> | <b>Work Flow</b>                       | <b>2</b>  |
| 2.1      | Sprint Planning Meeting . . . . .      | 2         |
| 2.2      | Standup Meeting . . . . .              | 3         |
| 2.3      | Review Meeting . . . . .               | 3         |
| <b>3</b> | <b>Work Item Procedures</b>            | <b>4</b>  |
| <b>4</b> | <b>Configuration Control</b>           | <b>7</b>  |
| <b>5</b> | <b>Project Management Documents</b>    | <b>8</b>  |
| <b>6</b> | <b>Software Release and Versioning</b> | <b>9</b>  |
| 6.1      | Release Requirements . . . . .         | 9         |
| <b>7</b> | <b>Software Defects</b>                | <b>10</b> |

|                               |           |
|-------------------------------|-----------|
| <b>8 Test Case Management</b> | <b>12</b> |
| <b>Bibliography</b>           | <b>13</b> |

---

---

---

LIST OF FIGURES

# CHAPTER 1

## INTRODUCTION

This document details the procedures to be followed for CTF related software development lifecycle and documentation tasks and the configuration management requirements associated with lifecycle tasks. CTF revision control is managed by Git. All changes to source code, the testing system, and code documentation are recorded in the Git software repository. Lifecycle tracking is performed using the **PHI Kanban board**. The Kanban board is a ticket tracking system that keeps track of all CTF-related development activities, including source code changes, feature additions, code documentation changes, and code testing activities. The following sections describe the iterative work flow organized into sprints, the detailed procedures associated with work items, the configuration control policies and procedures, the release procedure followed in providing versions of software outside of the development team, the procedures for managing software defects, and test case management.

## CHAPTER 2

---

## WORK FLOW

CTF has adopted an iterative and incremental approach to software development based on concepts from the Scrum agile software development framework. This iterative approach divides work into six week “sprints”. Each sprint will focus on a small collection of tickets in the PHI Kanban system. The sprint starts by selecting a collection of tickets that represent high priority activities. The number of tickets chosen is kept small enough that all tickets can be accomplished by the end of the six-week sprint. When a ticket is chosen to be worked on in a sprint, its status is set from “backlog” to “selected”. When tickets are actually being worked on, the status is set to “in-progress”. When the ticket is finished, its status is set to “review” and an independent developer is selected to review that the goal of the ticket has been accomplished in a satisfactory fashion and that sufficient testing exists to demonstrate the correct implementation of the feature and to protect that development from regressing. Documentation will also be checked to be sufficient in describing how the feature works, how the user enables the feature, and how the feature is tested. Finally, the status is set to “closed” after the work has been determined to be complete. This process represents the lifecycle for CTF development work.

### 2.1 Sprint Planning Meeting

The sprint planning meeting will focus on the selection of the highest value product backlog items for development of requirements, designs, and test cases. The number of items selected will be based on the estimate of effort for each

item. The goal is that all selected tickets should be able to be completed in the 6-week period of the sprint. The planning meeting is performed at the beginning of each sprint.

### 2.2 Standup Meeting

The sprint is divided into smaller pieces to better track progress, any changes in requirements, and any additional issues that were unexpected and require attention during the sprint. The standup meeting is held once every week and includes all member of the CTF development team. The meeting involves reviewing the PHI Kanban board and walking through all in-progress and selected tickets and briefly updating the team with the progress on the work item.

### 2.3 Review Meeting

The review meeting is performed at the end of the 6-week sprint. The focus during the meeting is to discuss lessons learned related to the software development process and lifecycle. Tickets that represent some important lessons learned are discussed. A review of the number of tickets closed and statistics on remaining outstanding tickets are discussed.



## CHAPTER 3

---

### WORK ITEM PROCEDURES

All CTF development work is tracked on the PHI Kanban ticket-tracking system. CTF tickets are organized into four categories known as “Epics”. These include:

- **CTF production activities**, which includes specific features and improvements that are required for CTF production use. These tickets should be designed such that they are short lived and able to be accomplished within a 6-week sprint.
- **CTF research activities**, which includes long-running development tasks related to CTF. These tickets are ones that generally have a looser definition of done, as the expectations of what the outcome will be are less clear than specific features that are being implemented into the code. These tasks are normally owned by developers at university organizations that do not always participate in the 6-week sprints and, therefore, will likely not complete the tickets within a 6-week period.
- **Defects**, which include any issues that track a defect in the code. These are typically high-priority tasks that limit some functionality of the code until fixed. Any defects that affect use of CTF in CASL applications shall never be in backlog status. Defects that track a problem in a CTF feature that is not used in CASL are permissible to be backlogged until developer availability is sufficient for addressing the issue.
- **Coupling activities**, which includes issues that relate specifically to CTF’s integration into CASL’s core simulator, VERA-CS, and its functioning and interoperability with other CASL codes.

Tasks that support CTF development activities are organized into tickets in the PHI Kanban system. The ticket is used as a means of capturing design requirements, acceptance criteria, a log of the work performed and changes made to the code, documentation, and tests. A ticket may have one of several statuses:

- **New**, which is the default status of a newly created ticket. Any tickets with new status shall have a different status assigned no later than the next sprint planning meeting. A new ticket can be set to a status of either backlog, selected, in-progress, or closed (won't fix or duplicate).
- **Backlog**, which means that the ticket will not be worked on during the current sprint.
- **Selected**, which means that the ticket will be worked on during the current sprint.
- **In-progress**, which means the ticket is currently being worked on. Progress on the work and any changes to requirements shall be updated regularly during the work.
- **Review**, which means the ticket is being reviewed by an independent (from the developer that did the work) CTF developer. In this stage, the independent reviewer ensures the acceptance criteria has been met, that the implementation is done in accordance with the rules laid out in the CTF Training Plan [1], that documentation has been updated, and the feature has been sufficiently tested. In addition to looking over the source code changes, the reviewer shall ensure the code builds and that all test cases (existing and newly added) pass.
- **Closed (fixed)**, which means the ticket has been completed and reviewed. The issue has met its stated definition of done.
- **Closed (won't fix)**, which means the ticket has been closed, but no work was done to fix the stated definition of done. This status should be used when it was decided that the work described in the ticket is no longer needed.
- **Closed (duplicate)**, which means that it was discovered that the work described in the ticket is already covered in another PHI Kanban ticket.

Some high-level guidance to using this system is provided:

- Every ticket shall have a designated owner that will be responsible for moving the ticket through its lifecycle and doing the development work required. The designated owner may be changed at any point throughout the ticket lifecycle. When the ticket is moved to "review", an independent developer should be selected to review the work.

### CHAPTER 3. WORK ITEM PROCEDURES

---

- The number of “in-progress” tickets that any one developer owns should be minimized to help the developer stay focused and work more efficiently.
- CTF has a product owner, who is responsible for closing tickets and moving tickets from “in-progress” back to “selected” or “backlog” when necessary. All CTF developers have permission to create new tickets as needed and move their tickets through the normal flow of the lifecycle.
- Any tickets that support CASL milestone work should be linked to the milestone number in the appropriate box of the Kanban ticket.
- All tickets shall include a clear definition of done (acceptance criteria) that states what criteria must be met to move the ticket to closed status.

The PHI Kanban lifecycle process is fully documented in detail at [https://vminfo.casl.gov/trac/casl\\_phi\\_kanban](https://vminfo.casl.gov/trac/casl_phi_kanban).

## CHAPTER 4

---

## CONFIGURATION CONTROL

Configuration control is managed using the Git version control system. Requirements for using this system are covered in the CTF Developer Training Plan [1]. CTF also has dependencies on third-party libraries (TPLs). TPLs are not included in any sort of configuration control on the CTF development and testing systems; however, a list of supported TPL versions is maintained in the Software Requirements Specification document [2]. All significant changes made to CTF are managed using the PHI Kanban system.

## CHAPTER 5

---

### PROJECT MANAGEMENT DOCUMENTS

Project management documents are those documents that control the management, QA, and work processes for the CTF project. These documents are version controlled using the Git version control system. Documentation is directly stored in the COBRA-TF source code repository in the COBRA-TF/doc directory. All documentation is written in  $\text{\LaTeX}$  and only the  $\text{\LaTeX}$  source files are stored in the source code directory. Documentation is built as part of the code build process. Project management documents include:

- Software requirements specification [2]
- Developer training plan [1]

CTF also includes end-user documentation:

- CTF User Manual [3]
- CTF Theory Manual [4]
- CTF Validation and Verification [5]

In addition to these CTF-specific documents, there are also CASL project management documents. The process for tracking and moving tickets through PHI Kanban are documented in the [PHI Kanban wiki](#). The CASL SQA document details the overall software quality assurance requirements for all CASL software products [6].

## CHAPTER 6

# SOFTWARE RELEASE AND VERSIONING

CTF is released continuously via [Github](#) and periodically as part of the CASL core simulator, VERA-CS.

### 6.1 Release Requirements

CTF is continuously released via Github. The master CTF git repository is stored on the casl-dev server. Changes shall only be pushed to the casl-dev server using the VERA-CS checkin test script. The checkin test script will determine what packages will be effected by new changes and, therefore, what tests need to be run to test that no code features have regressed. Upon successful completion of the checkin test script, new changes will be pushed to the casl-dev server.

A cron job exists on the casl-dev server that will run the entire CTF HEAVY test suite every night and, upon successful completion, will push all new changes to Github. Github is the main access point for all non-CASL CTF users. Non-CASL developers may push changes back to Github using topic branches, as documented in the CTF Developer Training Plan [1].

CTF is also released as part of VERA-CS, which is a more formal process. A formal versioning system is used and testing and documentation requirements are more comprehensive than the continual release process. The VERA-CS release process is documented in [\[casl'release\]](#).

## CHAPTER 7

---

## SOFTWARE DEFECTS

A defect is any condition in the software that does not meet the requirements as tracked through completed product backlog items or fails to meet reasonable and commonly accepted user expectations. Missing or incorrect requirements are not considered defects and instead should be managed as new product backlog items so that the specific requirement can be documented. Similarly, a failure of the software to meet measurement needs is not considered a defect unless it results from a failure to follow documented algorithms or requirements. Unanticipated measurement or administrative needs should be handled as new product backlog items so that the specific requirements can be documented. Problems reported by external users will always be entered as a defect work item for tracking purposes, but may be moved to a closed state if it is determined not to be a defect during analysis. The reason for closing will be recorded in the work item.

It is the goal of the CTF team to produce and maintain the production code base as defect free software, and to thereby ensure high quality defect free releases. In the event that defects are found in the production code base either by the CTF team or by external users they will be managed using the following procedures to ensure timely analysis of the problem and appropriate resolution. A system, CSICAT, is provided to external users for reporting defects. Users can email issues to [support@casl.gov](mailto:support@casl.gov). This process is detailed in the [CASL wiki](#).

- A ticket is created in the PHI Kanban system to track the defect.
- The summary description of the ticket shall start with “(Defect)” so as to make it easier for other developers to search on the outstanding defects in the code.

- All defects will be added to the VERA-CS defect epic.
- Defects that do not directly support CASL applications of CTF may be placed in the backlog or in selected.
- Any defects that directly impact use of CTF for CASL applications shall be placed either in selected or in-progress.

At a minimum, any defects shall be reported to the PHI focus area lead and all current CTF developers. Defects shall also be communicated to interested CASL CTF users that may be affected by the defect.



## CHAPTER 8

## TEST CASE MANAGEMENT

All testing of the CTF software is managed via CMake and TriBITs. This leads to all tests being automated. There are two categories of tests:

- **Unit tests**, which are the most fine-grained type of testing. Unit tests are created using the MPACT unit-test harness, which is available directly in the COBRA-TF source repository. Unit tests directly call specific procedures and classes in CTF and then compare procedure outputs to anticipated results.
- **Regression tests**, which are full CTF tests. The test process works by reading in CTF input files, running CTF to produce a set of output files, and doing a diff of the output file and a gold file. Gold files are version controlled in the CTF source code repository. The diff is done to within a tolerance to allow for small changes due to machine round-off error. The regression tests include tests that enable specific code features that have some impact on the tested code output as well as validation and verification tests, which have been compared to some analytical or experimental results in the Validation and Verification manual.

The regression test is determined to pass if it is within some tolerance of a “gold” output file. The CTF VTK output files are used for comparing results. The VTK file specifies the major solution variables for every solution cell in the mesh. Every solution data must match its gold counterpart in every solution cell in the model for the test to pass. There are two sets of tolerance; one for developers and one for users. The developer tolerances are tight enough that test

failures can occur just by machine round-off error encountered from switching machines. To prevent erroneous test failures when end-users install the code on their machine, a set of looser tolerances are provided. All developers who check changes into the master version of the repository shall use the tighter developer tolerances in their testing. Tolerances are defined directly in the CTF regression test harness.

It is a requirement that all tests added to the CTF testing matrix shall include the author of the test, the date the test was created, and a description of what the test is testing. This information shall be written directly in the test input file in the header of the input file using the code comment character to comment out the header information.

It is a requirement that all tests shall be sufficiently documented. Some simple unit tests may only call a correlation and demonstrate that the correct answer is returned. In-source documentation of the purpose of the test and a reference to the description of the calculation (if available) is sufficient for such tests. Larger scale verification-type tests that compare a code answer to an analytical solution shall be documented in detail in the CTF V&V manual [5].

The testing system organizes tests into three categories based on the computational demand of the test: CONTINUOUS, NIGHTLY, or HEAVY. Tests that take less than 60 seconds to run on the CASL development machines may be added to the CONTINUOUS, NIGHTLY, or HEAVY category. Tests that take less than 10 minutes to run on the CASL development machines may be added to the NIGHTLY or HEAVY category. Any tests that take longer than 10 minutes should be added to the HEAVY category. Tests that use parallel CTF may also be added to the test system, but shall be limited to using 3 processors or less.

Tests are reviewed as part of the PHI Kanban process (during the review stage of the lifecycle). Tests are run on a continual basis on the CASL testing machines and results are monitored on the CASL CDash site. If any tests fail, a notification will be sent to all developers who submitted code changes that went into the build that caused the test failure. All developers receiving the notification are required to review the failure and determine the root cause. If the test failure can quickly be remedied, it shall be fixed immediately. If the test failure cannot be quickly remedied, the test shall be deactivated and a PHI Kanban ticket shall be created to track the defect.

---

## BIBLIOGRAPHY

- [1] R. Salko. *CTF Developer Training Plan*. 2016.
- [2] R. Salko. *CTF Software Requirements Specification*. 2016.
- [3] R. Salko and M. Avramova. *CTF Preprocessor User's Manual*. The Pennsylvania State University. 2012.
- [4] R.K. Salko and M.N. Avramova. *CTF Theory Manual*. The Pennsylvania State University.
- [5] R. Salko et al. *CTF Validation and Verification*. The Pennsylvania State University. 2015.
- [6] M. Sieger. *CASL-QA-030 CASL Software Quality Assurance Requirements*. Tech. rep. CASL-U-2015-0010-000. Consortium for Advanced Simulation of Light Water Reactors, 2015.