OSW TN002 - TMT GUIDELINES FOR SOFTWARE SAFETY

TMT.SFT.TEC.11.022.REL07

August 22, 2012
# TABLE OF CONTENTS

1 INTRODUCTION ................................................................. 3
   1.1 Audience ........................................................................ 3
   1.2 Scope ............................................................................ 3
   1.3 OSW Technical Notes ....................................................... 3
   1.4 Applicable Documents ..................................................... 3
   1.5 Reference Documents ...................................................... 3
   1.6 Change Record .................................................................. 4
   1.7 Acronyms ........................................................................ 4

2 BACKGROUND ....................................................................... 5

3 SAFETY-CRITICAL SOFTWARE .............................................. 5

4 SOFTWARE AS PART OF THE TMT SAFETY PROCESS .............. 8

5 SOFTWARE SYSTEM HAZARD ANALYSIS ................................ 8
   5.1 Software Component Analysis ......................................... 9
   5.2 Assigning Control Categories .......................................... 9
   5.3 Determining the Software Hazard Risk Index ....................... 11
   5.4 Software Development Activities ..................................... 12
      5.4.1 Software Design Activities ....................................... 13
      5.4.2 Safety-related Requirements or User Stories and Tests .... 13
      5.4.3 Simulation of Hazards .............................................. 13
      5.4.4 Code Walkthroughs ............................................... 13
      5.4.5 Functional Tests ..................................................... 13

6 SOFTWARE HAZARD CONTROLS .......................................... 13

7 SOFTWARE SYSTEM HAZARD ANALYSIS DOCUMENTATION ..... 14
1 INTRODUCTION

Safety is important within TMT, and all systems/subsystems must consider safety issues throughout all project phases including design, construction, integration and testing, and operations. The TMT Environmental, Safety and Health Process Guidelines document (AP01) defines the process for reducing potential hazards to acceptable levels for all TMT systems using identification, assessment, mitigation, and reassessment.

This document provides more focused information on software safety. There are a wide range of opinions on how to tackle software safety and what can be done to manage risk associated with software. This document is meant to supplement AP01 and explain how it applies to software as well as to suggest what is expected from software groups with regard to software safety.

This document uses information from the NASA Software Safety Guidebook (RD01) and to a lesser extent the NASA Software Safety Standard (RD02). Both contain useful and informative software-specific safety information and are worth reading. Both these documents are free and easily available over the Internet. However, this document and AP01 are the sources for details of the TMT process for safety.

1.1 AUDIENCE

This document is targeted primarily towards software developers who are designing and implementing software within the TMT construction phase, but may also serve as the foundation for software safety procedures during operations.

This document assumes the reader is familiar with AP01 as well as the safety section of AP02 (note that the Software Management Plan will not be updated for safety until the next revision during 2012).

1.2 SCOPE

This document provides guidelines for applying the TMT safety process to software systems.

1.3 OSW TECHNICAL NOTES

OSW Technical Notes are short papers documenting studies, decisions or practices regarding OSW and TMT software that end up providing input to the TMT and OSW software designs or clarification on focused topics.

1.4 APPLICABLE DOCUMENTS

| AP01 | TMT Environmental, Safety and Health Process Guidelines, TMT.PMO.MGT.12.009. |
| AP02 | TMT Software Management Plan, TMT.SEN.SPE.08.002. |

1.5 REFERENCE DOCUMENTS

1.6 CHANGE RECORD

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Who</th>
<th>Modifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRF01</td>
<td>Dec, 2011</td>
<td>KG</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>DRF02</td>
<td>Jan 4, 2012</td>
<td>KG</td>
<td>Integrated C. Carter comments.</td>
</tr>
<tr>
<td>DRF03</td>
<td>Jan 9, 2012</td>
<td>KG</td>
<td>Changed title to reflect role as guideline.</td>
</tr>
<tr>
<td>DRF04</td>
<td>Mar 15, 2012</td>
<td>KG</td>
<td>Changes to support new guidelines document.</td>
</tr>
<tr>
<td>DRF05</td>
<td>Mar 20, 2012</td>
<td>KG</td>
<td>Document modified based on V. Ford comments.</td>
</tr>
<tr>
<td>DRF06</td>
<td>Mar 21, 2012</td>
<td>KG</td>
<td>Document modified based on B. Tyler comments. Figure updates and document formatting.</td>
</tr>
</tbody>
</table>

1.7 ACRONYMS

ES&H  Environmental, Safety and Health
HA    Hazard Analysis
HL    Hazard List
OSW   Observatory Software
SCr   Safety-Critical
SHRI  Software Hazard Risk Index
SSHA  Software System Hazard Analysis
TMT   Thirty Meter Telescope
2 BACKGROUND

The following safety related definitions used within this document are copied from AP01. Other software-related text is rewritten from RD01.

**Hazard:** A condition that poses a threat of injury or damage to life, health, equipment, or the environment. Each hazard has at least one cause, which in turn can lead to a number of effects (e.g., damage, illness, failure).

A **hazard cause** may be a defect in hardware or software, a human operator error, or an unexpected input or event which results in a hazard. A **hazard control** is a method for preventing the hazard, reducing the likelihood of the hazard occurring, or the reduction of the impact of that hazard. Hazard controls use hardware (e.g. pressure relief valve), software (e.g. detection of stuck valve and automatic response to open secondary valve), operator procedures, or a combination of methods to avert the hazard.

For every hazard cause there must be at least one control method, usually a design feature (hardware and/or software) or a procedural step. … Each hazard control will require verification, which may be via design, test, analysis, inspection, or demonstration…

Software can be used to detect and control hazards, but software failures can also contribute to the occurrence of hazards.

Other familiar concepts in software are the ideas of a fault, failure, and fault tolerance. These terms are used in the subsequent text. The following definitions are from AP01 via RD01, page 21.

A **fault** is any change in the state of an item which is considered anomalous and may warrant some type of corrective action.

A **failure** is the inability of a system or component to perform its required functions within specified performance requirements.

- A fault may or may not lead to a failure.
- One or more faults can become a failure.
- All failures are the result of one or more faults.

**Fault tolerance** is the ability of the system to withstand an unwanted event and maintain a safe and operational condition. It is determined by the number of faults that can occur in a system or subsystem without the occurrence of a failure…

Fault tolerance is often described as the number of failures or faults the system can handle and continue functioning at some level. A one failure tolerant system will continue functioning after a single failure has occurred; but a second failure would lead to an unsafe or failed system.

A software system can be failed and can still be a safe software system. However, a failure of a hazard control—whether it is software or hardware is a matter of concern.

3 SAFETY-CRITICAL SOFTWARE

It may seem odd that software is involved in safety. How do we know if software is “safe” or “unsafe”? The TMT safety process is focused on identifying hazards and the prevention of hazardous situations. Software that is directly involved in hazard control (such as within a dedicated safety system) as well as any software that can impact hazardous software or hardware is called safety-critical software, and a TMT software subsystem that includes safety-critical software is a safety-critical system.
The TMT Observatory Safety System is one approach to hazard control. The use of the TMT Observatory Safety System is treated as a hardware-based hazard control mechanism.

The following describes what it means for software to be safety-critical. More detail can be found in RD01, page 18:

**Safety-critical software** is software that can cause a hazard or software that is directly involved in a hazard control.

Software is considered safety-critical if it controls or monitors safety-critical hardware or software. Such software may reside on remote, embedded, and/or real-time systems. For example, software that controls an airlock or operates a high-powered laser safety-critical. Software that monitors a fire-detection system is also safety-critical.

Software that provides information required for a safety-related decision falls into the safety-critical category. If a human must shut down a piece of hardware when the temperature goes over a threshold, the software that reads the temperature and displays it for the human operator is safety-critical. All the software along the chain, from reading the hardware temperature sensor, converting the value to appropriate units, to displaying the data on the screen are safety-critical.

Software that performs off-line processes may be considered safety-critical in some cases. For example, software that verifies a software or hardware hazard control must operate correctly. Failure of the verification software may allow a hazardous condition to be missed. In addition, software used in analyses that verify hazard controls or safety-critical software must also function correctly, to prevent inadvertently overlooking a hazard. Modeling and simulation programs are two types of off-line software that may be safety-critical. Very often we rely on our software models and simulators to predict how part or all of a system may react. The system may be modeled to represent stressed or “normal” operations. Based on those modeled reactions, changes may be made in the design of the hardware, software, and/or operator procedures. If the system model fails to properly depict safety critical situations, design errors may go undetected.

If the software resides with safety-critical software on the same physical platform, it must also be considered safety-critical unless adequately partitioned from the safety-critical portion. Non-safety-critical software (such as a data processing algorithm) could lock up the computer or write over critical memory areas when sharing a CPU or any routines with the safety-critical software.

The following text, adapted from page 30 of RD01, describes software’s potential involvement in hazards.

Software may impact a hazard in several ways.

- **Software failure may cause a hazard.** For example, software may incorrectly command a mechanical arm to move past its operational limit, resulting in the arm damaging nearby equipment or causing injury. A failure in a data conversion function could incorrectly report a temperature value, allowing a furnace door to be opened when the temperature inside is at a dangerous level.

- **Failure of a software hazard control may allow a hazard to occur.** If software is used as a hazard control to monitor pressure and open a valve at a threshold, failure of that software would allow an over-pressurization of the vessel and a potential for a hazardous rupture.
• **Failure of software safing, which transitions the system state from hazardous to non-hazardous.** Failure of the software to detect and shut down a runaway electromechanical device (such as a robotic arm or scan platform) is an example of a safing mode failure.

• **Software used to mitigate the consequences of accident may fail.** For example, software controlling the purging of toxic gases (which resulted from a failure in some other portion of the system) may fail, allowing the gases to remain in the chamber or be vented inappropriately to the outside air.

When conducting a software safety analysis, it is important to consider many types of failures that may lead to hazards. Examples of failures to consider are:

• Sensors or actuators stuck at some value (all zeros, all ones, some other value) or other sensor failure
• Value above or below range
• Value in range but incorrect
• Physical units incorrect
• Wrong data type or data size
• Incorrect operator input
• Overflow or underflow in calculation
• Algorithm incorrect
• Shared data corrupted
• Out of sequence event
• Failure to meet timing requirement
• Memory usage problems
• Data overflow due to inappropriate data packet or data arrives too quickly
• Data sampling rate not sufficient to detect changes
• One task failing to release shared resource
• Deadlocking in a multitasking system
• Effects of either system or computer hardware failures on the software
• Temperature control. How is the software system involved in setting or monitoring temperature? Can software failure lead to catastrophic damage to detectors or other hardware? Is software expected to “close the loop” when temperature is out of range?

• Consideration of interfaces. Does the system under evaluation receive an input from another safety-critical system that results in the system under evaluation having safety-critical considerations? Is the system under evaluation depending on another system to provide its hazard controls?

• What happens upon power failure to a motor or other device?
• What can happen upon network failure? What needs to be restarted?
• Operations involvement – what software failures can lead to loss of observing time?
• Can the software fail causing damage to equipment?
• Can motors be driven to inappropriate or damaging positions due to software fault?
The following bullets summarize attributes of safety-critical software:

- Causes or contributes to a hazard (i.e. software controls high-cost or difficult to replace hardware and inadvertent control or response failure may damage hardware or create another hazard).
- Provides control or mitigation for hazards.
- Controls hazardous or safety-critical hardware.
- Monitors safety-critical hardware or software as part of a hazard control.
- Provides information upon which a safety-related decision is made.
- Performs analysis that impacts automatic or manual hazardous operations.
- Verifies hardware or software hazard controls.
- Can prevent safety-critical hardware or software from functioning properly.

### 4 SOFTWARE AS PART OF THE TMT SAFETY PROCESS

The TMT safety process for suppliers is defined in AP01. This document supplements that document and tailors the process for software.

The goal of the TMT software safety process is to identify safety-critical software and determine what additional development process activities may be needed to verify and validate the safety-critical software.

The safety process for software is part of the TMT software safety process that includes:

1. **Hazard Analysis.** Evaluation of the hardware and software system to determine a list of system hazards (hazard analysis) associated with current and future activities (see Section 2.2.2 of AP01).
2. **Determination of Hazard Risk Approval Code.** The risk assessment of each hazard results in a Hazard Risk Approval Code (see Section 2.2.3.4 of AP01 and sections leading up to the section).
3. **Hazard Control.** Every hazard needs at least one hazard control. Each hazard control will require verification (see Section 2.2.4).

The Software System Hazard Analysis (see Section 5) is part of the system hazard analysis for systems that include a software component.

At the conclusion of the hazard analysis phase the TMT Software Hazard/Risk Assessment Worksheet may contain software in two ways: as a hazard control or as software that can cause a hazard through its failure (requiring hazard control).

At this point there are new steps for software.

1. **Determine software involvement.** This analysis covers both the case when software is involved in hazard control and when software itself can cause a hazard.
2. **Determine software hazard risk and associated mitigation effort.** By analyzing how software is involved in each hazard control, additional software development effort is determined to mitigate the hazard likelihood.

Section 5 describes the Software System Hazard Analysis. Section 7 describes the additional documentation needed for software safety.

### 5 SOFTWARE SYSTEM HAZARD ANALYSIS

The Software System Hazard Analysis (SSHA) is an analysis of the entire software system to identify software failures that can create hazards and the risks associated with the hazards. The SSHA is part of the system hazard analysis, but it uses the risk assessment procedure described here.
5.1 Software Component Analysis

Most TMT software systems will have multiple subsystems each of which may have its own software system safety classification. Each software component should be analyzed to determine if it is safety-critical.

Software does not break over time; its failures are due to logic or design errors. There are two common cases.

- The software has no coding errors, but is written from incorrect or incomplete specifications.
- The specifications are correct, but the software has coding errors that result in deviation from specifications or failures.

More complex software will generally have more coding errors and more requirements issues; therefore, it is more probable that complex software will have failures that are causal factors of a hazard.

The TMT software architecture is well-suited to safety analysis in that software subsystems are composed of smaller (and hopefully less-complex) task-focused components. For example, a detector temperature assembly could be safety-critical, but a filter motion control assembly is probably not safety-critical. (It is advantageous to design the software system to isolate safety-critical and non-safety-critical components as discussed in the quotation from RD01 in Section 3 of this document.)

A Software Failure Modes and Effects Analysis (SFMEA) for each component is a useful, bottoms-up approach for investigating the potential effects of possible software failures in its environment (http://en.wikipedia.org/wiki/Failure_mode_and_effects_analysis). Section 3 includes possible failure considerations.

The product of this step is documentation demonstrating an understanding of how each software component is or isn’t involved in the hazards and hazard controls of the system. See Section 7 for documentation guidance.

5.2 Assigning Control Categories

Regardless of the cause of a hazard (hardware, software, human error, or environment), the severity of the hazard remains constant. Therefore the hazard severity categories from AP01 are just as applicable for software and are repeated in Table 1.

<table>
<thead>
<tr>
<th>Hazard Severity Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic (1)</td>
<td>Death or permanent total disability, system loss, major property damage or severe environmental damage.</td>
</tr>
<tr>
<td>Critical (2)</td>
<td>Severe injury, severe occupational illness, major system or environmental damage.</td>
</tr>
<tr>
<td>Marginal (3)</td>
<td>Minor injury, lost workday accident, minor occupational illness, or minor system or environmental damage.</td>
</tr>
<tr>
<td>Minor or Negligible (4)</td>
<td>Less than minor injury, first aid or minor supportive medical treatment type of occupational illness, or less than minor system or environmental damage.</td>
</tr>
</tbody>
</table>

Table 1: Safety-Critical Hazard Severity Categories.

Many hazards have some level of software involvement. The next step is to determine the risk associated with each hazard that involves software. Software requires its own process for
determining hazard risk because failure of software is not predictable using the probabilistic models used for hardware.

TMT is using the approach documented in MIL-STD-882 (RD03). Software’s involvement in a hazard is related to the software’s “control capability” within the context of each hazard’s software causal factors.

The software control category definitions are shown in Table 2. Software is most involved in control category I and has no involvement in a hazard in control category IV.

<table>
<thead>
<tr>
<th>Safety Critical Software Control Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (total)</td>
<td>Software exercises autonomous control over potentially hazardous hardware systems, subsystems or components without the possibility of intervention to preclude the occurrence of a mishap. Failure of the software or a failure to prevent an event leads directly to a mishap occurrence.</td>
</tr>
<tr>
<td>IIA (direct)</td>
<td>Software exercises control over potentially hazardous hardware systems, subsystems, or components allowing time for intervention by independent safety systems to mitigate the hazard. However, these systems by themselves are not considered adequate.</td>
</tr>
<tr>
<td>IIB</td>
<td>Software item displays information requiring immediate operator action to mitigate a hazard. Software failures will allow or fail to prevent the mishap occurrence.</td>
</tr>
<tr>
<td>IIIA (indirect)</td>
<td>Software item issues commands over potentially hazardous software systems, subsystems or components requiring human action to complete the control function. There are several redundant, independent safety measures for each hazardous event.</td>
</tr>
<tr>
<td>IIIB</td>
<td>Software generates information of a safety-critical nature used to make safety-critical decisions. There are several, redundant, independent safety measures for each hazardous event.</td>
</tr>
<tr>
<td>IV (none)</td>
<td>Software does not directly control safety-critical hardware systems, subsystems or components and does not provide safety-critical information.</td>
</tr>
</tbody>
</table>

Table 2: Safety-Critical Software Control Categories.

The control category descriptions may require some additional explanation.

I (total). In category I, the only control preventing a hazard is totally software-based. The other possibility is that a software failure by itself can cause a mishap. This category indicates the most software risk.

IIA, IIB (direct). Categories IIA and IIB are called direct, because the software is directly involved in control of hazardous systems or directly involved in displaying information for a human who must mitigate the hazard. Category II has more risk than III because if the software has failures, the independent safety systems are not considered adequate.

IIIA, IIIB (indirect). In categories IIA and IIIB, software interacts with potentially hazardous hardware as in II, but it is indirect because a human action is required to complete the control function (and hopefully in a safe manner). In both the control and monitoring situation there are adequate independent, redundant safety systems. Therefore, category III is less risky than category II.

IV (none). There is no software involvement in controlling or monitoring a safety-critical system.

Take for example a somewhat hypothetical laser system similar to TMT’s laser system. In this example, a software component is reading out a detector image that under normal, safe operation shows six spots. If one or more spots is missing something may be wrong, and the software determines that the laser should be shuttered. In this example, the software component is safety-critical. If the software component fails, the laser may not be shuttered in a hazardous
situation and system damage may occur. Failure of the detector software causes the occurrence of the hazard. Based on the information given, this is a category I hazard since correct behavior of the software component alone must prevent the hazard.

A second example is the operation of slewing the telescope. Slewing the telescope mount is likely a safety-critical function. If a telescope slew must be “okayed” by the operator of the telescope and there are adequate safety systems, the function is a category II A operation. If the software can determine on its own to slew the telescope and the safety systems may not be adequate, the operation is a IIA operation.

5.3 Determining the Software Hazard Risk Index

The software risk of a hazard is determined by combining its Control Category and Hazard Severity. Clearly an autonomous software system (control category I) that can fail resulting in a catastrophic mishap has more risk and is of more concern than a software component that is one of several (control category IIIb) involved in notifying system operators of the occurrence of a marginal severity hazard.

The measure of software risk is called the Software Hazard Risk Index (SHRI). The higher the SHRI number, the less risk. The software associated with hazard with a smaller SHRI requires more development resources than one with a higher SHRI.

![Software Hazard Criticality Matrix](image)

Table 3: Software Hazard Criticality Matrix shows the relationship between a hazard's severity and its control category.
The SHRI relationship between hazard control category and its severity is shown in Table 3. High risk (low SHRI) is in the upper left and low risk is shown in the lower right. The SHRI value can range from 1 to 5.

An analysis of each hazard leads to its SHRI, which should be added to the TMT Software Hazard/Risk Assessment Worksheet (see Section 7).

### 5.4 SOFTWARE DEVELOPMENT ACTIVITIES

Risk associated with lower SHRI software hazards is managed through additional activities during development. Developers of a safety-critical system must work harder to demonstrate their software is correct.

The following table (Table 4) provides the minimal required activities for each SHRI level. The hazards with less risk require less effort. Each level builds on the tasks required at the lower levels. The types of required activities are described in the following sections.

In the following *user story* is a description of something a software system should do from a user's perspective. It is an alternate way of writing software requirements.

<table>
<thead>
<tr>
<th>Software Hazard Risk Index</th>
<th>Actions Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - High Risk</td>
<td>• No high-risk safety critical hazards are allowed. Hazard must be removed through redesign. Otherwise, a waiver is needed for SHRI 1 hazard with required actions determined by the situation.</td>
</tr>
<tr>
<td>2 - Medium risk</td>
<td>• Software design demonstrates how identified hazards will be eliminated or mitigated showing extent of software involvement. Design is reviewed. • Hazard control generates new software requirements or user stories that can be tracked and tested during development. • Tests associated with safety requirements simulate hazards and exercise and verify hazard controls. • Required code walk-throughs or reviews of safety-related features by team and TMT. • Tests for proper function of component under normal TMT operational scenarios. • Optional demonstration of critical software hazard controls with prototypes.</td>
</tr>
<tr>
<td>3 - Moderate High Risk</td>
<td>• Software design demonstrates how identified hazards will be eliminated or mitigated showing extent of software involvement. Design is reviewed. • Hazard control generates new software requirements or user stories that can be tracked and tested during development. • Tests associated with safety requirements simulate hazards and exercise and verify hazard controls. • Tests for proper function of component under normal TMT operational scenarios. • Optional code walk-throughs or reviews of safety-related features by team and TMT.</td>
</tr>
</tbody>
</table>
### Table 4: Actions required for hazard acceptance criterion in Safety-critical software systems.

<table>
<thead>
<tr>
<th>Risk Level</th>
<th>Actions</th>
</tr>
</thead>
</table>
| 4 - Moderate Risk | • Software design demonstrates how identified hazards will be eliminated or mitigated showing extent of software involvement. Design is reviewed  
• Hazard control generates new software requirements or user stories that can be tracked and tested during development.  
• Tests associated with safety requirements simulate hazards and exercise and verify hazard controls.  
• Tests for proper function of component under normal TMT operational scenarios. |
| 5 - Low Risk | • Tests for proper function of component under normal TMT operational scenarios. |

#### 5.4.1 Software Design Activities

For SHRI 2, 3 and 4, the Software Design Description should include a safety discussion showing how the software system is safety-critical. Any hazards created by the software and their mitigation must be discussed and reviewed. The safety treatment should be in more depth for higher risk systems.

#### 5.4.2 Safety-related Requirements or User Stories and Tests

For SHRI levels 2, 3 and 4, safety-critical software should result in new requirements and/or user stories during development that can be traced to hazards from the SSHA. These user stories will have tests allowing software behavior to be verified.

#### 5.4.3 Simulation of Hazards

For SHRI 2, 3 and 4 additional tests may be required that allow proper behavior of the software/hardware hazard control to be demonstrated with a simulation of the hazard. These tests will be based on the specifics of each situation and will be determined by the development team and TMT.

#### 5.4.4 Code Walkthroughs

For SHRI 2, safety-critical software is required to undertake a code walkthrough attended by the development team and TMT software members. This team will examine the safety-related code and recommend changes as needed. The code-walkthrough is optional for SHRI 3 hazards at the discretion of TMT.

#### 5.4.5 Functional Tests

Safety-critical software must, as with all TMT software, provide tests for user stories showing desired normal operation. This ensures that any safety-critical functionality does not influence non-safety-related functionality.

### 6 SOFTWARE HAZARD CONTROLS

The Hazard Analysis determines the list of system hazards and the Software System Hazard Analysis (SSHA) determines which of the hazards have software as a component and the extent of the software involvement in hazard control.

Section 2.2.4 in AP01 documents TMT hazard control best practices and lists the order of precedence to use when choosing a hazard control for an identified hazard. TMT prefers hardware-based hazard controls for hardware hazards or hardware-based hazard controls.
controls in conjunction with software-based hazard controls. The use of the Observatory Safety System is considered to be a hardware-based hazard control.

In some situations the software control and hardware control cooperate. The software control may be the first line of defense with a hardware-based control as a backup if the software control fails.

Note that TMT does not accept SHRI 1, high-risk hazards or hazard controls. If an analysis shows this situation it is necessary to redesign to raise the SHRI or apply for a waiver and negotiate development procedures. Software-only control of a high-risk hazard requires TMT approval.

Sections of RD01 provide guidelines and best practices for dealing with software safety which may be of help for developers.

Note: Software safety should not be confused with software security although lax software security can be a hazard that leads to a safety issue. TMT plans on addressing software security issues at the observatory level. Individual telescope systems should assume a secure environment.

7 SOFTWARE SYSTEM HAZARD ANALYSIS DOCUMENTATION

The SSHA uses the TMT Software Hazard/Risk Assessment Worksheet to document basic information for software-based hazards and software controls as with all hazards. The example for this worksheet is in Appendix A of AP01.

A second software-specific worksheet, the TMT Software Hazard Analysis Worksheet (or other similar representation), shows the results from the SSHA. An example worksheet is shown on page 15 of this document. The information needed for this worksheet is defined within the sections of this document. The Hazard ID in column 1 ties each row/entry of the SSHA worksheet to the TMT Hazard/Risk Assessment Worksheet. There should be one row for each of the hazards or hazard controls associated with a component.

It’s important that reviewers understand that all software components have been analyzed for safety even if the result of the analysis is that a component is not safety-critical. The Software Design Document (or separate Software Safety document) should contain a safety section that indicates the processes used for analyzing safety in every software component. The following table (and some associated text) is necessary to show that safety has been considered for all software systems. A component that is not safety-critical will have “none” in the component hazard ID column and there should be one row for each component in the system.

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Project Phase</th>
<th>Analysis Methods</th>
<th>Component Hazard IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Template component analysis table.

The last point is that safety analysis must be done throughout the development lifecycle. Safety analysis, refinement and progress are required for each review of TMT systems and subsystems. Software safety analysis is a mandatory part of all software reviews starting with the Preliminary Design Review. The project phase column in each of the spreadsheets can be used to indicate the phase for the hazard documentation.
<table>
<thead>
<tr>
<th>Hazard ID</th>
<th>Component Name</th>
<th>Control/Hazard (C/H)</th>
<th>Project Phase</th>
<th>Description of software’s role as a hazard or hazard control</th>
<th>Hazard Risk Index</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Safety Category</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Control Category</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software Hazard Risk Index</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Verification Method (test, analysis, inspection, demonstration)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Required Actions Included in Development Plan (Y/N)</td>
</tr>
</tbody>
</table>