GPC Project Example Numerical Simulations

Tool

Software Assurance Classification Report

Key sections are highlighted in gray.

**Prepared by:**

|  |  |
| --- | --- |
| **Name** | **Date** |
| Leslie J. Johnson, NASA LaRC  Software Assurance | 6/26/2013 |

**NASA Langley Research Center**

**Hampton, VA 23681**

**Revision and History Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision**  **No.** | **Change**  **No.** | **Description** | **Release**  **Date** |
|  |  | Initial Release | 6/26/2013 |

Table of Contents

SECTION PAGE

1. INTRODUCTION 4

1.1 Background 4

2. REFERENCE Documents 4

3. SUMMARY 5

3.1 Software Classification 5

3.2 Software Safety 5

3.3 Software Assurance Effort 6

Appendix A: ACRONYMS 7

# INTRODUCTION

This report contains the software assurance classification assessment which identifies and evaluates the characteristics of software in determining the software’s classification, software safety-criticality, and level of software assurance to be applied to a Project.

## Background

The code is a MATLAB program for performing numerical simulations of the use of Generalized Predictive Control (GPC) for active control of free-body dynamic systems.  While there are various GPC methodologies described in the literature, the subject program uses a version of GPC developed at Langley by Dr. Paul Revere and his team during the mid-1990s.  The overall objective of such numerical simulations is to assess the applicability and effectiveness of GPC for active control of dynamic systems having rigid-body modes, such as aircraft in flight.  Two possibilities present themselves.  One can determine the control inputs needed to quell undesirable vibratory responses due to disturbances acting on the system (the regulation problem) or one can determine the control inputs needed to perform a prescribed maneuver or follow a specified flight path (the tracking problem).  Both regulator (zero target response) and tracking (nonzero target response) problems can be studied with the program.

The subject code has its genesis in a series of programs developed in the late 1990s under a cooperative NASA Langley/Company X research program in tiltrotor aeroelasticity.  The present version of the program was developed during the period 5/09-10/10 during which time a state-space model defining the rigid-body flight dynamics of the NASA/Army VV-3000 Tiltrotor Research Aircraft was embedded in the program and replaced the simple three-degree-of-freedom mass/spring/dashpot system that was used as the subject system in earlier programs.  The VV-3000 model was obtained from Company X and was developed for an aircraft cruising at 150 knots in the airplane mode of flight.  The math model has four control inputs (Collective pitch, longitudinal and lateral cyclic pitches, and pedals) and four outputs (Forward velocity, pitch angle, roll angle, and sideslip angle).

Researchers knowledgeable in coding using MATLAB and having a basic understanding of the equations underlying the methodology implemented in the program should be able to modify the code as needed to meet their particular needs.  The primary changes would be to replace the VV-3000 state-space model and to adjust plotting parameters in the code to account for differences in numbers of control inputs and response outputs.

The equations and analyses underlying the implementation of GPC in the subject code are described in NASA report NASA/TM-0000-00000.

# REFERENCE Documents

The following documents were used or referenced in the development of this report:

|  |  |
| --- | --- |
| **Document No.** | **Document Title** |
| NPR 7150.2A | NASA Software Engineering Requirements |
| NASA-STD-8739.8 | NASA Software Assurance Standard |
| NASA-STD-8719.13B | NASA Software Safety Standard |
| LAPD 5300.1 | Program/Product Assurance |
| LPR 7150.2 | LaRC Software Engineering Requirements |
| LPR 5300.1 | Product Assurance Plan |
| LMS-CP-4754 | Software Assurance (SA) for Development and Acquisition |

# SUMMARY

The following paragraphs summarize the results and describe the details used to determine the software classification assessment for this report.

## Software Classification

According to LPR 7150.2, the software component for this Project is classified as Class E – Small Light Weight Design Concept and Research and Technology Software which is defined as

1. *Software developed to explore a design concept or hypothesis, but not to make decisions for an operational Class A, B, or C system or to-be built Class A, B, or C system, or*
2. *Software used to perform minor desktop analysis of science or experimental data.*

As such, the Project shall follow the instructions and complete the compliance matrix in LMS-CP-7150.6, *Class E Software*, which applies to all Class E software that is not safety-critical.

## Software Safety

The Software Safety Litmus Test below is applied to all projects with software to determine if the software is safety-critical. If the software is determined to be safety-critical, then the project must adhere to the NASA-STD-8719.13, NASA Software Safety Standard.

A software component is considered safety-critical if it meets **any** of the following criteria:

|  |  |
| --- | --- |
| **Criteria:** | **Software**  **components** |
| 1. Resides in a safety-critical system (as determined by a hazard analysis) **AND** at least one of the following apply: | No |
| 1. Causes or contributes to a hazard |  |
| 1. Provides control or mitigation for hazards |  |
| 1. Controls safety-critical functions |  |
| 1. Processes safety-critical commands or data |  |
| 1. Detects and reports, or takes corrective action, if the system reaches a specific hazardous state |  |
| 1. Mitigates damage if a hazard occurs |  |
| 1. Resides on the same system (processor) as safety-critical software |  |
| 1. Processes data or analyzes trends that lead directly to safety decisions (e.g., determining when to turn power off to a wind tunnel to prevent system destruction) | No |
| 1. Provides full or partial verification or validation of safety-critical systems, including hardware or software subsystems. | No |

The software components in this Project do not reside in a safety-critical system; process data or analyze trends that lead directly to safety decisions or provide full or partial verification or validation of safety-critical systems.

The LaRC Safety and Mission Assurance Office have determined that the software components in this Project are not safety-critical.

## Software Assurance Effort

The software assurance effort is based on the software class and impacts from potential failure. In accordance with LMS-CP-4754 software assurance is not applicable for non-safety critical Class E software developments.

# Appendix A: ACRONYMS

|  |  |
| --- | --- |
| CP | Center Process |
| LaRC | Langley Research Center |
| LAPD | Langley Policy and Directives |
| LMS | Langley Management System |
| LPR | Langley Procedural Requirements |
| NASA | National Aeronautics and Space Administration |
| NPR | NASA Procedural Requirement |
| SA | Software Assurance |
| STD | Standard |