NASA Contractor Report 181972

GCS PLAN FOR SOFTWARE ASPECTS OF CERTIFICATION

A. M. Shagnea, D. S. Lowman, and B. E. Withers

Research Triangle Institute
P.O. Box 12194
Research Triangle Park, NC 27709

Contract NAS1-17964
February 1990

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Preface

The GCS Plan for Software Aspects of Certification is document # 14 in a series of fifteen documents which fulfill the Radio Technical Commission for Aeronautics RTCA/DO-178A guidelines, "Software Considerations in Airborne Systems and Equipment Certification [3]." The documents are numbered as specified in the DO-178A guidelines. The documents in the series are used to demonstrate compliance with the DO-178A guidelines by describing the application of the procedures and techniques used during the development of flight software. These documents were prepared under contract with NASA-Langley Research Center as a part of their long term research program addressing the fundamentals of the software failure process.

This project consists of two complementary goals: first, to develop software for use by the Research Triangle Institute (RTI) in the software error studies research program sponsored by NASA-Langley Research Center [7]; second, to use and assess the RTCA/DO-178A guidelines for the Federal Aviation Administration (FAA). The two goals are complementary in that the use of the structured DO-178A guidelines in the development of the software will ensure that the test specimens of software have been developed according to the industry standards for flight critical software. The error studies research analyses will then be conducted using high quality software specimens.

The implementations will be subjected to two different software testing environments: verification of each implementation according to the RTCA/DO-178A guidelines and replicated random testing in a configuration which runs more than one test specimen at a time. The term *implementations* refers to bodies of code written by different programmers, while a *version* is a piece of code at a particular state (i.e., version 2.0 is the result of code review). This research effort involves the gathering of product and process data from every phase of software development for later analysis. More information on the goals of the Guidance and Control Software (GCS) project are available in the *GCS Plan for Software Aspects of Certification.*

The series consists of the following documents:

GCS Configuration Index Document no. 1

- *GCS Development Specification* Document no. 2

- *GCS Design Descriptions* One for each software implementation. Document no. 3

- *GCS Programmer's Manual* Document no. 4, includes Software Design Standards, document no. 12.

- *GCS Configuration Management Plan* Document no. 5A

- *Software Quality Assurance Plan for GCS* Document no. 5B

- *GCS Source Listing* One for each software implementation. Document no. 6

- *GCS Source Code* One for each software implementation. Document no. 7

GCS Executable Object Code One for each software implementation. Not available on hardcopy. Document no. 8

- *GCS Support/Development System Configuration Description* Document no. 9

- *GCS Accomplishment Summary* Document no. 10

- *Software Verification Plan for GCS* Document no. 11

- *GCS Development Specification Review Description* Document no. 11A

- *GCS Simulator (GCS_SIM) System Description* Document no. 13

- *GCS Simulator (GCS_SIM) Certification Plan* Document no. 13A

- *GCS Plan for Software Aspects of Certification* Document no. 14

Contents

List of Figures

List of Tables

1 Introduction

This document provides the framework for certification of the Guidance and Control Software (GCS) implementations developed by the Research Triangle Institute under contract with NASA-Langley Research Center. The purpose of the *GCS Plan for Software Aspects of Certification* is to describe the overall project plans to the Federal Aviation Administration (FAA) while still in the early stages of the project. At the end of the project, another document entitled the *GCS Accomplishment Summary* will be produced. The accomplishment summary will address each of the goals for the project listed in the *GCS Plan for Software Aspects of Certification* and show that each plan was carried out and each goal met. In the *GCS Plan for Software Aspects of Certification*, the project organization is discussed, and the overall GCS System is described. Each DO-178A document being developed in support of this project is described briefly in Section 5, and the schedules of software and document development are included in Section 4.2. While the details of each phase of the project are contained in other documents, this document discusses *why* the various design and development decisions were made. It should become clear, while reading this document, that while every attempt was made to have this project mirror one in industry, there is an experiment being performed as well. This forced the GCS team to consider every decision from at least two angles: whether the decision will help to produce high quality code, and whether the decision will help to bring about clear results to the experiment. The interaction of these goals coupled with the decisions necessary for a software development project is non-trivial. The *GCS Plan for Software Aspects of Certification* explains many of the decisions made and shows how both the development and experiment goals were taken into account.

1.1 Overview of the Research Triangle Institute and the Center for Digital Systems Research

The Research Triangle Institute (RTI) performs interdisciplinary research in the engineering, physical, chemical, life, environmental, statistical, social, and policy sciences under contract to clients in business, industry, and government. The Center for Digital Systems Research (CDSR) con

ducts technical research with respect to digital systems. Research projects range from software research and development to VLSI circuit design and computer architecture research. The GCS project is the responsibility of the Software Research and Development Department (SRDD), managed by Janet Dunham. The major focus of SRDD is on achieving safe and reliable software through research and development in new methods for software specification and design, program verification, software reliability, software safety and software estimation, and emulation and simulation tools. These areas of research are crucial where the consequences of failure are costly, as is the case in the current project, guidance and control software. SRDD also focuses on the research and development of parallel processing.

1.2 Overview of GCS Project Goals

As was stated in the preface, there are multiple goals for this project [1].

2 Description of System to be Certified

2.1 Software Description

The Guidance and Control Software (GCS)

1. provides guidance and engine control of the planetary landing vehicle during its terminal phase of descent onto a surface and

2. communicates sensory information about the vehicle and its descent to some other receiving device.

GCS is designed to control a planetary lander during its final descent. After the vehicle has dropped from orbit, the software will control the engines of the vehicle to the surface of a planet. The controlling software reads data about its surroundings from six sensors that relay information about the vehicle's acceleration, altitude, velocity, and rotation rate as well as the atmospheric temperature and the touch-down state. From the information provided by the sensors, an on-board navigator determines

[1] These goals are described in [7].

both the current state of the vehicle and the desired state of the vehicle. The state information is passed from the navigator to engine controlling modules that determine the appropriate commands to the axial and roll engines on the lander.

During the course of this experiment, three implementations of the specification will be developed. The Guidance and Control Software implementations will be executed in a software simulator, the Guidance and Control System Simulator, GCS_SIM. The simulator is a software tool which takes the place of the hardware system for the purposes of this project, and takes the place of the hardware system referred to in the DO-178A requirements. More detail on the simulator can be found in Section 6.6 of this document and in the *GCS Simulator System Description*.

2.2 Criticality of Software Levels

The RTCA/DO-178A guidelines use *Levels* to classify the criticality category of functions. Level 1 is associated with the critical category, Level 2 with the essential category, and Level 3 with the non-essential category [3]. The level of each piece of software is dictated by requirements for reliability and safety. For example, flight control systems would be classified as Level 1 or critical, and a toilet flush system on board an airplane could be classified as Level 3 or non-essential. The criticality of each function within the project is listed in Table 1.

Two functions (CP – Communications Processing and TSP – Temperature Sensor Processing) were originally asserted to be Essential but now are classified as Critical. This change was made because the possibility exists that these processes could corrupt the memory and thus corrupt critical processes. The FAA requires a justification of any partitions between processes of different criticality levels. Because all functions are critical in this project, there is no partition.

3 Project Organization

3.1 NASA/RTI Communication

Figure 1 shows the GCS project organization. The project organization is divided into two independent components: software quality assurance and the development of the implementations and simulator (GCS_SIM). In this way software quality assurance is independent from the development teams. The Project Leader, Janet Dunham, and Software Quality Assurance Manager, Elizabeth Bailey, report to the Contract Monitor, George Finelli. The Project Leader and Contract Monitor communicate via telephone and meet for a status meeting once per month. The Assistant Project Leader (Anita Shagnea) has weekly meetings with the RTI GCS group and reports the results to the Project Leader. The task leader for software verification (Leslie Dent) is the RTI contact for the Software Quality Assurance Manager and the NASA LaRC member of the verification team (Kelly Hayhurst). Edward Withers, as task leader for configuration management, works closely with Leslie Dent and Stephen Duncan to maintain versions of code and documents which are sent to NASA LaRC. The task leader for the simulator development (Douglas Lowman) is the main RTI contact for NASA LaRC team members Bernice Becher and Carlos Liceaga. More explanation about each group within the structure is found in Section 6.

3.2 Communication within RTI

Each large task within the project is managed by a task leader who not only manages the task, but does technical work on it as well. Because the GCS project is relatively small, staff members work on a variety of tasks. The leaders of the tasks (Edward Withers, Douglas Lowman and

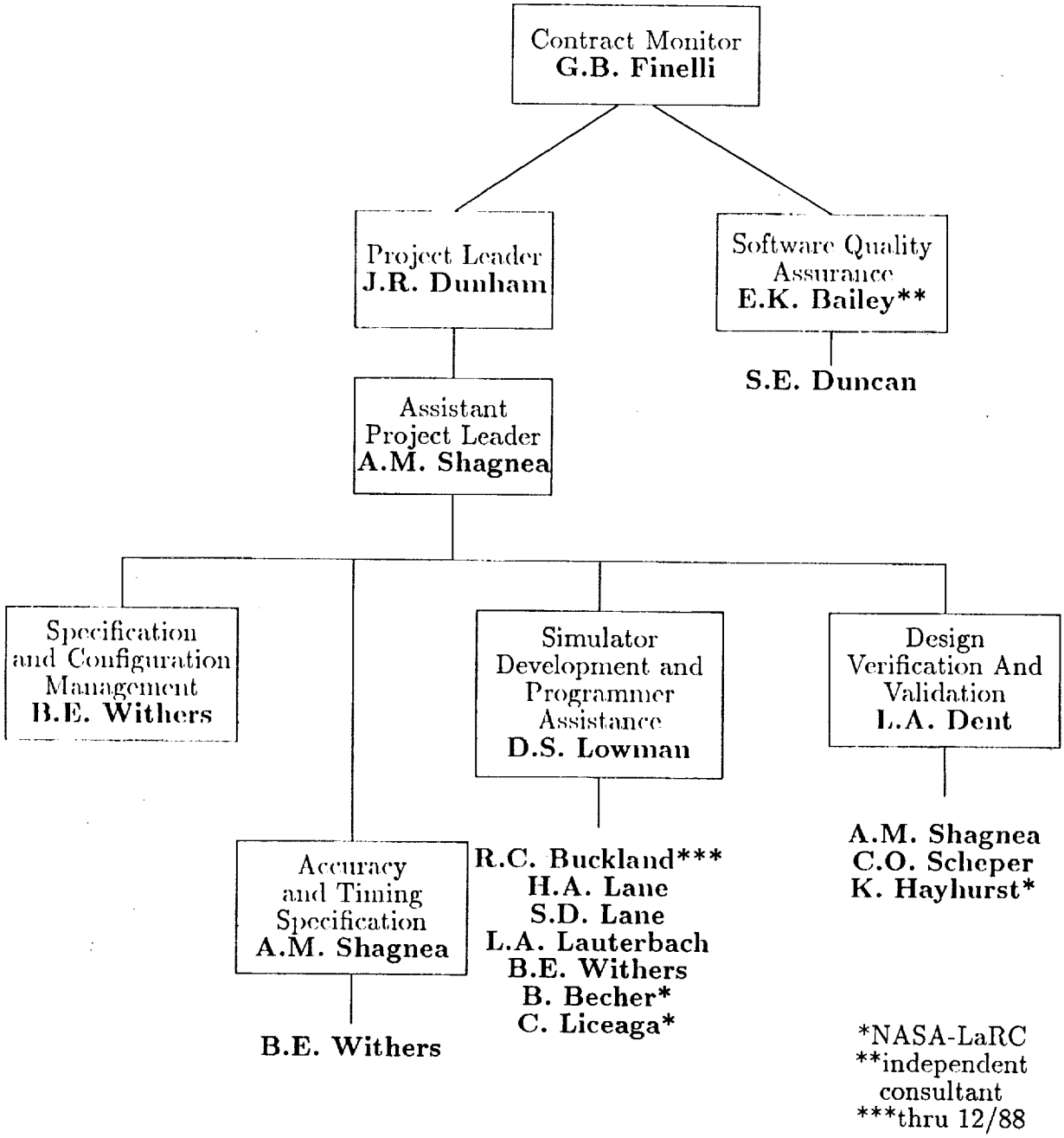


Figure 1: GCS Project Organization

Process	Criticality
2.1 AECLP - Axial Engine Control Law Processing	Critical
2.2 ARSP - Altimeter Radar Sensor Processing	Critical
2.3 ASP - Accelerometer Sensor Processing	Critical
2.4 CP - Communications Processing	Critical
2.5 CRCP - Chute Release Control Processing	Critical
2.6 GSP - Gyroscope Sensor Processing	Critical
2.7 GP - Guidance Processing	Critical
2.8 RECLP - Roll Engine Control Law Processing	Critical
2.9 TDLRSP - Touch Down Landing Radar Sensor Processing	Critical
2.10 TDSP - Touch Down Sensor Processing	Critical
2.11 TSP - Temperature Sensor Processing	Critical

Table 1: Criticality of Functions

Leslie Dent), the SQA representative (Stephen Duncan), and the Assistant Project Leader (Anita Shagnea) meet once per week to maintain open communication throughout the project.

Other staff members (including programmers) are kept informed by the Assistant Project Leader and their task leader(s).

3.3 Project Management

The Project Leader, Janet Dunham, reports to the Contract Monitor, George Finelli. She oversees the activities of the project within RTI, and, with the Contract Monitor, has final say on project decisions. The Assistant Project Leader, Anita Shagnea, oversees the day-to-day management on the project. She reports to the Project Leader and makes decisions which are subject to the Project Leader's approval. The Assistant Project Leader tracks effort and cost information, which is passed to the Contract Monitor and Project Leader on a monthly basis, and creates the monthly reports which are required for NASA projects. The Project Leader and Assistant Project Leader meet once per week to go over the activities of the past week, talk over future efforts, and discuss the general goals of the project.

3.4 Management Debriefings

Following each design review, set of code reviews, and test completion/readiness review, the review team will participate in a short debriefing with the Project Leader. Copies of the checklists and traceability matrices will be given to the Project Leader and discussed. Major problems with the design, code, or test cases will also be discussed and problems which trace back to the development specification will be noted for the Project Leader's information. The purpose of the debriefing is to apprise management of the progress of the implementations and relate any major problems that arise.

3.5 Software Quality Assurance

The Software Quality Assurance (SQA) Team is independent from the other participants in order to have a team which can audit the project freely. The

SQA Manager, Elizabeth Bailey, is an independent consultant and does not report to the Project Leader, Janet Dunham. The SQA Manager has responsibility for the *Software Quality Assurance Plan for GCS*. The onsite SQA representative, Stephen Duncan, is an RTI employee. He reports to the SQA Manager for the GCS project, and his RTI manager is outside of Janet Dunham's department. The SQA representative performs the SQA functions specified in the *Software Quality Assurance Plan for GCS* and attends the weekly GCS communication meetings.

4 Software Lifecycle and Certification Activity Milestones

4.1 Certification Activities to Support Software Aspects of Certification

4.1.1 Document Delivery Dates

Table 2 shows each document with release dates. The release of the *GCS Source Code*, *GCS Source Listing* and *GCS Executable Object Code* is described in the *GCS Configuration Management Plan* and in Figure 4 in this document.

4.1.2 Document Responsibility

Table 3 lists each document with the person(s) who have responsibility for that document. All are GCS participants who have worked directly on the task associated with the document. Most documents have more than one author and several reviewers. The reviewers always include the Contract Monitor, Project Leader and SQA Representative.

4.2 Software Lifecycle Milestones

Figures 2, 3 and 4 show the development schedule for the GCS implementations, simulator and documents.

Table 2: DO-178A Documents and Release Dates

#	Document	Release #	Completion
1.	GCS Configuration Index	1.0	5/31/89
2.	GCS Development Specification	2.0	Complete
		Mods	Periodic
3.	GCS Design Description	Mercury1.0	See
	(Contains team*work* Model)	Earth1.0	Development
		Pluto1.0	Schedule
4.	GCS Programmer's Manual	Draft	2/28/89
	(Contains Doc. 12)	1.0	4/31/89
5A.	GCS CM Plan	Draft	12/20/88
		1.0	1/31/89
5B.	SQA Plan for GCS	Draft	12/20/88
		1.0	4/31/89
6,7,8.	GCS Source Listing, Code	Mercury1.0	See
		Earth1.0	Development
		Pluto1.0	Schedule
9.	GCS Support/Development	Draft	4/31/89
	System Configuration	1.0	5/31/89
10.	GCS Accomplishment	Draft	7/1/89
	Summary	1.0	7/31/89
		2.0	8/31/89
11.	Software Verification	Draft	12/20/88
	Plan for GCS	Draft	2/15/89
		1.0	3/20/89
11A.	GCS Development Specification	Draft	
	Review Description	1.0	
13.	GCS Simulator (GCS_SIM)	Prelim	3/10/89
	System Description	Draft	5/31/89
13A.	GCS Simulator (GCS_SIM)	Draft	NASA
	Certification Plan	1.0	NASA
14.	GCS Plan for Software	Draft	1/20/89
	Aspects of Certification	1.0	4/15/89
		2.0	6/31/89

Table 3: DO-178A Documents and Responsibilities

#	**Document**	**Responsibility**
1.	GCS Configuration Index	Anita M. Shagnea
2.	GCS Development Specification	B. Edward Withers
3.	GCS Design Description	Programmers
4.	GCS Programmer's Manual	Douglas S. Lowman
5A.	GCS Configuration Management Plan	B. Edward Withers
5B.	Software Quality Assurance Plan for GCS	Elizabeth K. Bailey
6,7,8.	GCS Source Listing, Code	Programmers
9.	GCS Support/Development System Configuration	Douglas S. Lowman
10.	GCS Accomplishment Summary	Anita M. Shagnea
11.	Software Verification Plan for GCS	Leslie A. Dent Kelly Hayhurst
11A.	GCS Development Specification Review Description	Janet R. Dunham
13.	GCS Simulator (GCS_SIM) System Description	Douglas S. Lowman
13A.	GCS Simulator (GCS_SIM) Certification Plan	Carlos Liceaga Douglas S. Lowman
14.	GCS Plan for Software Aspects of Certification	Anita M. Shagnea

Figure 2: GCS Simulator and Test Planning Schedule

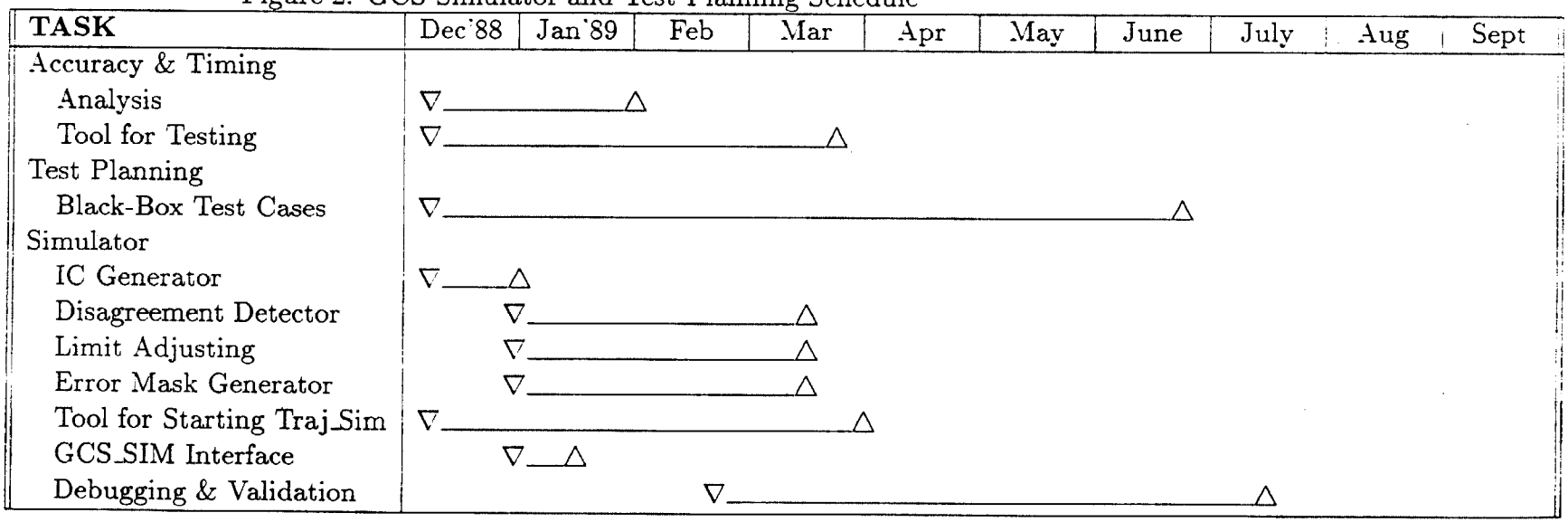
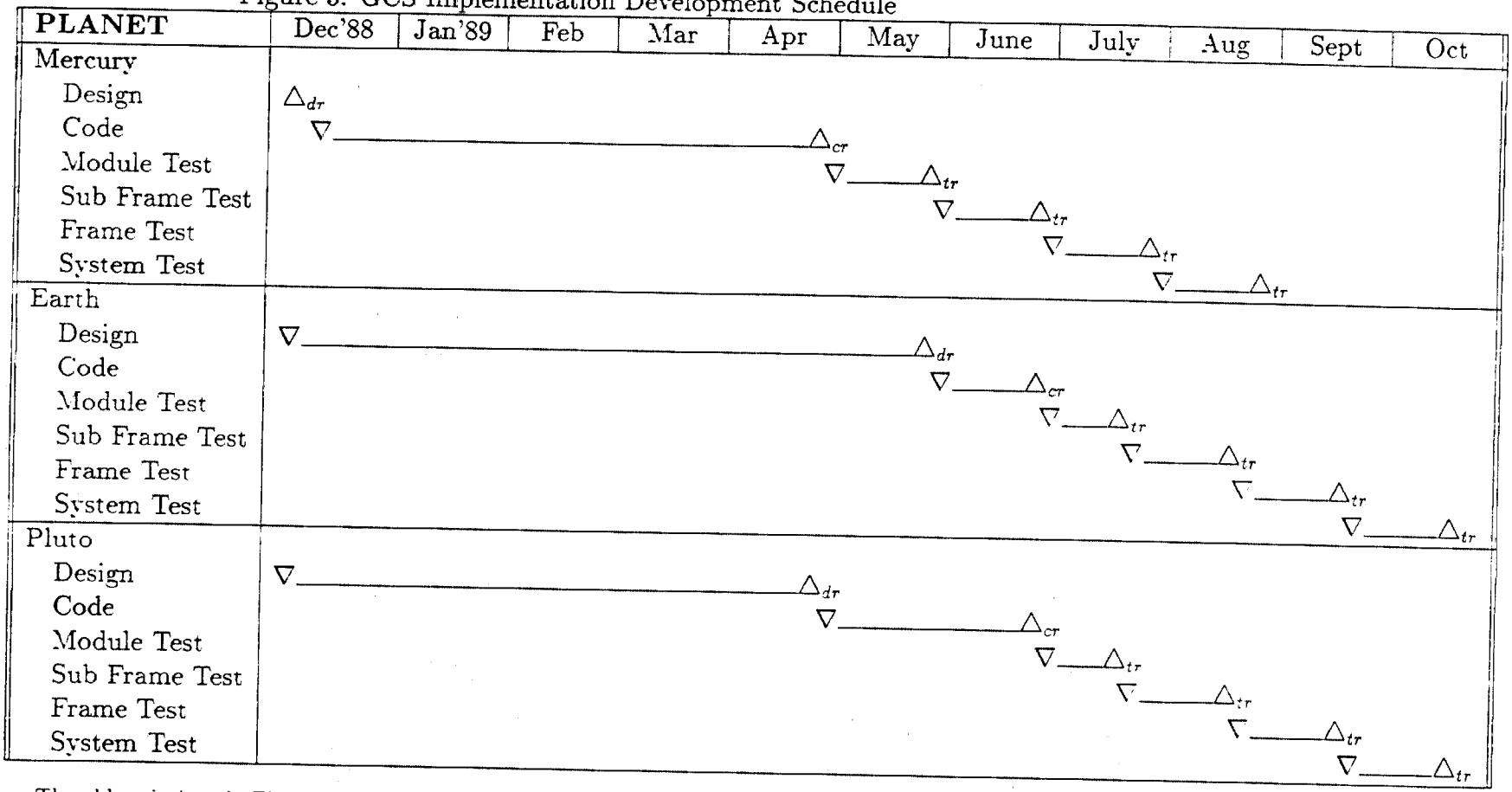
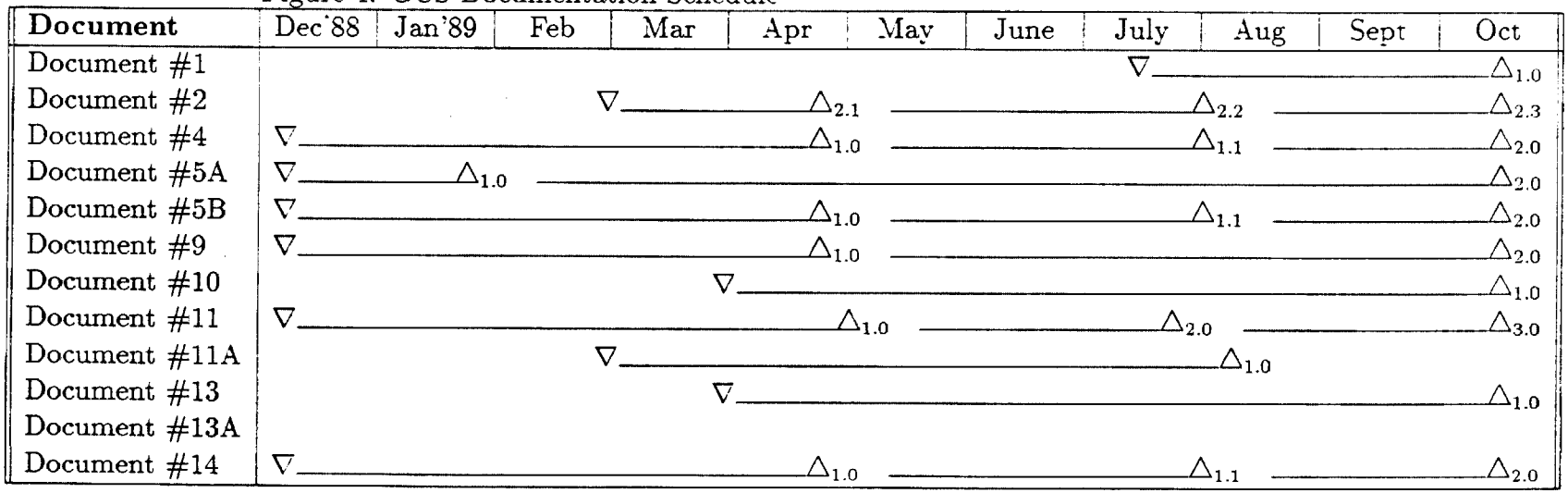


Figure 3: GCS Implementation Development Schedule



The abbreviations in Figure 3 refer to the following: dr – design review, cr – code review, and tr – test completion/readiness review.

Figure 4: GCS Documentation Schedule



5 Documentation Plan

For a project with Criticality Level 1,[2] the DO-178A guidelines require the assembly of a number of documents which together describe the entire GCS project–what has been done and why it was done in that particular manner. The coverage of the documents overlaps, and documents often reference other documents. All documents will be put under configuration control. Detailed information regarding configuration control for each document is in the *GCS Configuration Management Plan*. Figures 2 and 3 show the number of releases for each document, the deadlines for those releases, and the main author for each document. These figures are in Section 4.1.1. The following paragraphs give a brief explanation of each document.

5.1 Configuration Index Document (CID)– Document #1

The *CID* is the document which records the change history for all GCS documentation. There will be one *CID* which will list the documentation for the three GCS implementations, including not only all the documents required under the project by the DO-178A guidelines, but any other documents produced by the GCS project. Each version of every document will be uniquely identified and listed in the *CID*. The *CID* will be produced only at the conclusion of the GCS development, due to the fact that the GCS configuration management procedure collects and records change history information, points to the area where each document is kept during development, and records the exact commands used to link and compile the code. The *CID* will incorporate the information recorded by the configuration manager into a document format.

5.2 GCS Development Specification – Document #2

The current specification document contains the software requirements. The *GCS Development Specification* was put under configuration control before the design of the three implementations was begun. Modifications

[2]Section 2.2 defines the FAA Criticality Levels and discusses the criticality of the GCS project.

to the document have gone through appropriate approval channels and are also under configuration control. The *GCS Development Specification* was reviewed, and the results of this review are in the *GCS Development Specification Review Description.*

5.3 GCS Design Description – Document #3

The *GCS Design Description* for each implementation (Mercury, Earth, Pluto) is created by the programmer responsible for that implementation. The *Software Verification Plan for GCS* specifies the procedure used to review the design. The design is reviewed by a team consisting of the programmer, the tester responsible for testing that implementation, the user, and the Software Quality Assurance (SQA) representative. After the design has been reviewed, the *GCS Design Description* is put under configuration control such that one programmer/tester pair may not review an implementation different from their own.

5.4 GCS Programmer's Manual – Document #4 (Includes Software Design Standards)

The *GCS Programmer's Manual* consists of the Programmer Instructions for the GCS Experiment. These are communications to the programmers regarding different aspects of the programmer responsibilities. The instructions, prior to being contained in the *GCS Programmer's Manual*, are routed through approval channels and placed under configuration control. The material which was to be covered in the Software Design Standards document (#12) is included in this document because the standards were issued to the programmers as Programmer Instructions. The subjects of the instructions include not only the design and coding standards, but formatting for documents, information regarding the use of the software problem report forms, and a listing of the required tasks which the programmer performs for testing and SQA approval.

5.5 GCS Configuration Management Plan – Document #5A

The *GCS Configuration Management Plan* covers configuration management for all GCS documentation, source listings/code, and team*work* [3] design descriptions. The *GCS Configuration Management Plan* is intended for use by the programmers and management team. It includes change control procedures and release numbering conventions. It explains the use of CMS (Code Management System) [1], the electronic configuration management system that is being used. Detailed information about CMS is available in the *GCS Support/Development System Configuration Description.*

5.6 Software Quality Assurance Plan for GCS – Document #5B

The *Software Quality Assurance Plan for GCS* contains procedures for independent software quality assurance. This includes procedures for test completion/readiness reviews and explanations of the role the SQA representative plays in each verification and approval activity in the project. The detail of the procedures for the Test Completion/Readiness Reviews are contained in this document.

5.7 GCS Programmer Documents – Documents #6,#7,#8

The *GCS Source Code* (Document #7) will be the only one of these documents kept under configuration management. According to the DO-178A guidelines, the source code is "code in a machine-readable form" [3, Section 8.1.7]. For GCS, this is the set of FORTRAN statements for each implementation. The other documents can be retrieved from the source code. The *GCS Source Listing* (Document #6) will contain the source code, compiled with the /LIST option, and the linker map associated with the source code. Creating the *GCS Source Listing* is dependent only on the *GCS Source Code* and the compiler, so the *GCS Source Listing* for a specific version of source code can be recreated to exactly the same state at any time. The *GCS Executable Object Code* (Document #9) is also retrievable

[3] Team*work* is a registered trademark of Cadre Technologies Inc.

from the *GCS Source Code*. The *GCS Configuration Management Plan* discusses the configuration control of these documents. The commands and options used to link and compile the source code will also be under configuration control and will be listed in the *GCS Configuration Index*.

5.8 GCS Support/Development System Configuration – Document #9

The *GCS Support/Development System Configuration* specifies all software and hardware used in the development of the implementations. This will include testing tools, debugging tools, and all other tools used in software engineering on this project. It will also include a description of the software and hardware environment of the project and the release numbers for each tool.

5.9 GCS Accomplishment Summary – Document #10

The *GCS Accomplishment Summary* is a summary of all aspects of the GCS project. It will point to certification information in other documents, most notably the results of testing, which will be included in a later release of the *Software Verification Plan for GCS*, and the *CID*, which will specify where each document (including the completed implementations) resides. It is the final document created and the primary document used by the FAA for certification, in that it supports the claim that the GCS implementations are certifiable.

5.10 Software Verification Plan for GCS –Document #11

The first release of the *Software Verification Plan for GCS* explains the methodology used in testing the GCS implementations and outlines the procedures for testing. Successive releases will include the actual test cases, with expected results, and finally the testing results. More detail on the verification procedures is found in Section 6.5. The *Software Verification Plan for GCS* specifies the reviews and procedures that are the responsibility of the verification team, while the *Software Quality Assurance Plan for GCS*

outlines the responsibilities of the SQA representative. As is explained in Section 6.5, the design review, code review, and all test activities are the responsibility of the verification team, while the test completion/readiness reviews are the responsibility of SQA. Because some verification activities include both the verification team and SQA representative, the documents frequently reference each other.

5.11 GCS Development Specification Review Description – Document #11A

This document describes the procedures used to review the *GCS Development Specification*, including the review criteria, the participants, and the review results. The *GCS Development Specification* was subjected to extensive peer review, tested through the use of prototype implementations, and modelled using a CASE tool. Any errors discovered during the coding and execution of the prototypes were logged and the specification was modified. The test planning for the GCS implementations[4] incorporates the possible instance of specification error into the test procedures. Suggested modifications are logged and must be approved by the onsite Software Quality Assurance representative.

5.12 GCS Design Standards – Document #12

The design standards were originally written as a Programmer Instruction, and have thus been included in the *GCS Programmer's Manual*, described in Section 5.4.

5.13 GCS Simulator (GCS_SIM) System Description – Document #13

The *GCS Simulator System Description* describes the system to be submitted for certification. The DO-178A recommendations state that the hardware environment for the software should be described in the system requirements document [3]. Because the GCS project requires the use of a

[4]See the *Software Verification Plan for GCS* for more information on the test process.

software simulator in lieu of a hardware system, the simulator (GCS_SIM) will be described in this document. The descriptions will include specifications and descriptions of the development of the simulator. A brief description of the simulator is found in this document, in Section 6.6.

5.14 GCS Simulator (GCS_SIM) Certification Plan – Document #13A

The *GCS Simulator Certification Plan* will contain a test plan for the simulator, written jointly by NASA LaRC and RTI. The document will contain procedures for any reviews held, as well as test procedures and results.

5.15 GCS Plan for Software Aspects of Certification – Document #14

This document specifies in brief the plans for all aspects of the GCS project which relate to the certification of the GCS implementations. It includes proposed schedules and plans for execution of the tasks involved in the overall project. The *GCS Accomplishment Summary* uses this document as a specification to demonstrate to the FAA that the goals proposed in this document have been attained.

6 Activities to Support Software Aspects of Certification

The Guidance and Control Software project has a two-fold goal, as was explained in the Preface and Section 1. Several decisions were made regarding development and verification issues to accomplish various aspects of the overall goal. An attempt has been made to resolve issues in such a way that both the NASA and FAA goals have been met. This Section contains detailed explanations of these decisions.

6.1 Specification Development and Analysis

The *GCS Development Specification* was created using structured analysis methods and has been through an extensive peer review. Two prototypes were created to help with the debugging of the specification, and errors have been carefully tracked. The specification was also modelled using a CASE tool. This process gave feedback which was used in the specification. More information on the analysis and review of the *GCS Development Specification* can be found in the *GCS Development Specification Review Description*.

6.2 Accuracy Requirements Analysis and Specification

The *GCS Development Specification* accuracy requirements analysis was necessitated by the large number of real variables in the *GCS Development Specification* and by the presence of numerical operations that may limit the accuracy of computed values. The analysis utilized backgrounds in applied mathematics and aerospace engineering as well as a thorough knowledge of the specification. The following texts were consulted while completing the accuracy specifications: *Numerical Analysis* by R.Burden, J.Faires, and A.Reynolds [5] and *Calculus and Analytic Geometry* by G.Thomas and R.Finney [17]. The results of the analysis are listed in the *GCS Development Specification* and are described in *GCS Specification Accuracy Analysis Plan* [16]

6.3 Configuration Management

Configuration management on the GCS project is being carried out in a procedure sufficient for a small software project. Each version of code or documentation is recorded in the Code Management System software tool (CMS), which can provide a history of all requests for files and changes made to files. The numbering of the different versions of code is specified in the *GCS Configuration Management Plan* in order to have versions appropriate for further research into software errors.

6.4 Programmer Implementation Development

6.4.1 Number of Implementations

The NASA research goal involves different error detection mechanisms, and GCS_SIM will execute implementations in parallel. RTI is developing three implementations[5] which will execute in the simulator either in parallel or individually. Each of the implementations is being developed using the DO-178A guidelines per the requirements of the FAA. The use of the DO-178A guidelines should help the programmers create industry quality code, which should produce quality code specimens for the NASA experiment.

6.4.2 Programmer Experience

All three GCS programmers have previous programming experience. The reason for choosing experienced programmers is the supposition that experienced programmers can produce better code than inexperienced programmers and are on par with programmers working in industry. This is extremely important, because in order to compare results of the three implementations either when they are run in parallel or through some other analysis, high quality code will lend itself to high quality experiment results. These results will be representative of similar projects in industry. Software Engineering Experience Questionnaires which list each programmer's programming experience, system experience, and university coursework in software engineering have been completed. This information is in the project files and is available for inspection.

6.4.3 Team*work* Design

The design for each implementation will be created using team*work*[6]. Team*work* is a CASE tool which captures component relationships of a design. It also enables analysts to create and verify functional system specifications [2]. Team*work* uses a structured analysis methodology defined by

[5] Because the word *versions* is being used for configuration management of code and documents, the term *implementations* is being used to denote the different bodies of code produced by different programmers.

[6] Team*work* is a registered trademark of Cadre Technologies, Inc.

Derek Hatley and Imtiaz Pirbhai [8]. The design includes data flow diagrams, a data dictionary, process specifications, and control specifications.

The decision was made to use a structured design methodology and a CASE tool which supports it in order to give the programmers a tool which will help them create a structured design, and thus, structured code. Tools support modularity and structure and give the programmers a common model from which to work [10]. It is hoped that this decision will enable the programmers to create code on par with that of similar projects in industry.

6.5 Design Verification and Validation

Design verification and validation is covered in detail in the *Software Verification Plan for GCS*. Many verification decisions listed below were made to reflect the research goals of GCS. To maintain equally high standards of testing and ensure that testers will not be influenced by another programmer's code, all black-box test cases (sub-frame, frame, and system) will be written before the sub-frame testing has begun. All testers will use the same black-box test cases, so that only the white-box sub-frame test cases will vary between implementations.

6.5.1 Testing Divisions

The testing of the GCS implementations is divided into categories (these are listed in reverse chronological order); system testing, frame testing, sub-frame testing, module testing, and reviews.

System testing is done by the testers and consists of black-box testing, using only high level inputs and looking at the overall trajectory, final conditions, and certain state parameters for the outputs. This type of testing is also called hardware/software integration testing or system validation testing, but since there is no physical hardware associated with the project, the term system testing is used to refer to the testing of the implementations integrated with the simulator.

Frame testing is also executed by the testers. A frame is one time step in the trajectory, and is explained in the *GCS Development Specification*.

Here the testers will use black box test cases which will have inputs for the frame and look at the resulting output for certain variables.

Sub-frame testing is performed by the testers. A sub-frame is the smallest unit actually required by the *GCS Development Specification.* Each programmer may modularize the code within a sub-frame as finely as he/she likes, or may code the entire sub-frame as one module. This latitude is allowed in order to leave room for diversity between the three implementations. Because of this, sub-frames are the smallest unit examined by the testers. The testing consists of black-box testing, similar to that of the frame testing, and white-box testing, written by the individual testers. The white-box tests are written based on the code itself, not just on the inputs and outputs to the code unit. The GCS testers will use McCabe's structured testing methodology [11] and the McCabe tool ACT to help create the white-box test cases.[7]

Module Testing is done completely by the programmer. Because the size of the modules is not specified in the software requirements, test cases for use with all three implementations cannot be created beforehand; so the programmers will create, log, and execute their own test cases. Because the programmers are restricted from running the code before the module test phase, module testing also exists to give the programmers a chance to test their own code. There is a requirement in the *Software Verification Plan for GCS* for a minimum number of test cases.

Reviews that are the responsibility of the verification team include the design and code reviews. The verification team takes part in the test completion/readiness reviews, but these are the responsibility of SQA, and are described in detail in the *Software Quality Assurance Plan for GCS.*

[7]The Analysis of Complexity Tool (ACT) created by McCabe and Associates, Inc. is an analysis tool which facilitates white-box testing and uses a methodology which gives 100% multiple condition coverage. The tool has been widely used in industry and government work. More on the tool and method of testing is covered in the *Software Verification Plan for GCS.*

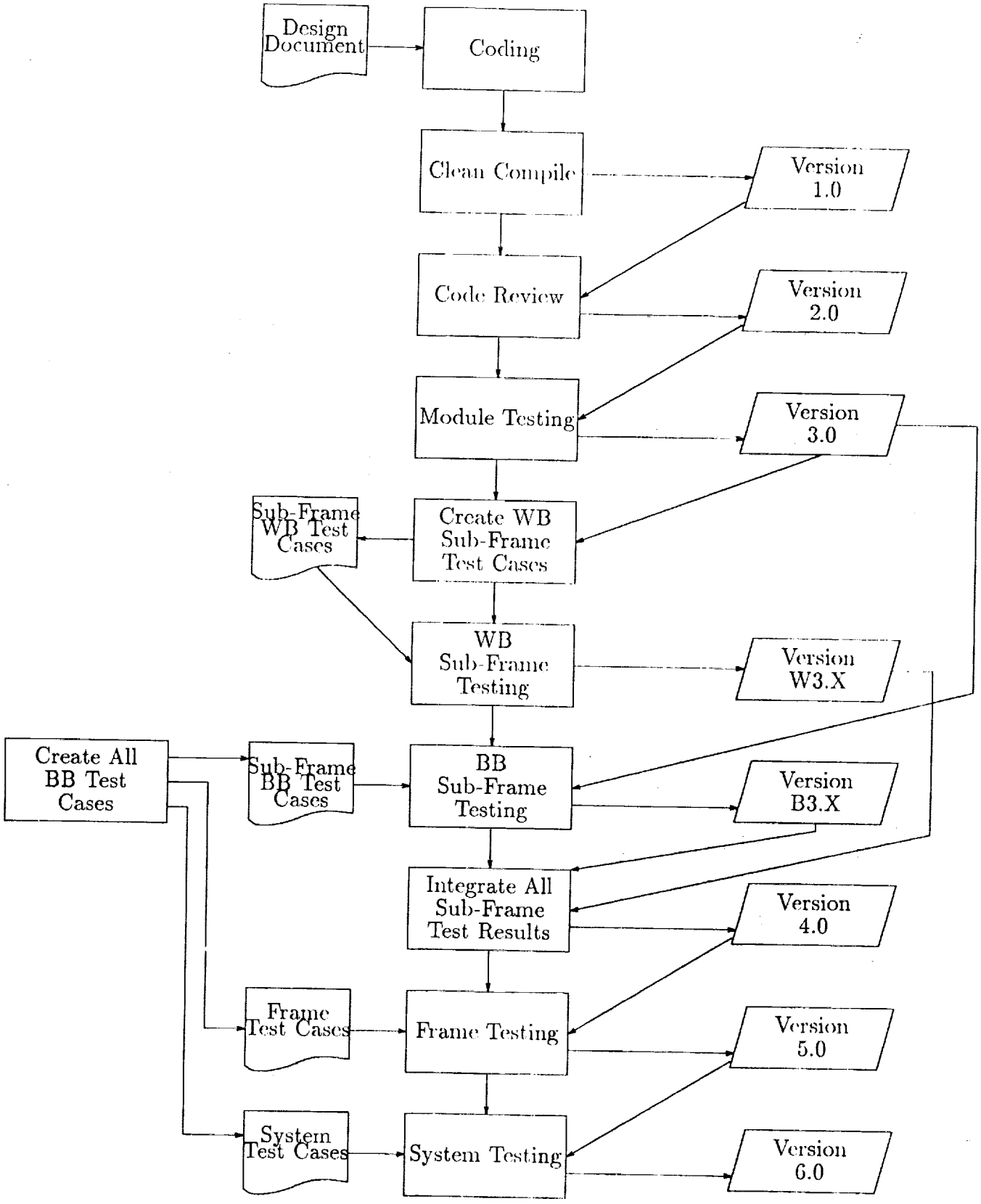


Figure 5: Software Development Phases

6.5.2 Order of Development Phases

Figure 5 shows the phases of software development, including verification activities.[8] As the diagram indicates, all of the test cases except for the sub-frame white-box test cases are created before any of the implementations reach the sub-frame test phase. The testers create the black-box test cases as a group in order to limit the variability between testers. The black-box test cases are written before any tests are executed, so that errors found in one implementation do not influence the testing of another implementation. The sub-frame white-box test cases will be created by individual testers because the code must be analyzed to produce the test cases.

Versions of code are saved at various phases during the development. These versions will be used for comparisons with versions from other development phases as well as comparisons with other methods of testing, such as repetitive run testing.

Both parts of the sub-frame testing (white- and black-box) are performed on version 3.0 of the code. The code versions produced are then direct products of either white-box or black-box testing. An interesting comparison can then be made between the results of white- and black-box testing on the same piece of code. After all sub-frame testing is done, versions W3.x and B3.x will be integrated into version 4.0. Version 4.0 will be the final product of sub-frame testing. Version 6.0 will be the result of system testing, and the final version of the code.

6.5.3 GCS Review Checklists and Problem Report Form

Review Checklists, and the GCS Problem Report Form underwent many revisions, which were reviewed by staff at RTI and NASA LaRC.[9] The Jet Propulsion Laboratory (JPL) Software Product Assurance checklists, William Hetzel's *The Complete Guide to Software Testing* [9], and Glenford Myers' *The Art of Software Testing* [14] were used as references for the GCS Design Review Checklist. The GCS Code Review Checklist used

[8] In the figure, **WB** and **BB** represent white-box and black-box testing, respectively. Although the box showing the creation of test cases is in the middle of the page, this process actually takes place before white-box sub-frame testing and is done by the testers as a group.

[9] These forms can be found in an appendix of the *Software Verification Plan for GCS.*

the above references, as well as an internal RTI FORTRAN coding standards document and the results of an experiment conducted by RTI for RADC [15]. The checklists were designed for the GCS project, and contain design and code standard information which is specific to this project. The GCS Problem Report Form used problem report forms from previous project as examples, and used a paper by Victor Basili [4] as a reference.

6.6 Simulator Development and Validation

The Guidance and Control Software Simulator (GCS_SIM) is a control-system testbed that acts as a combined test-harness, modeling system, and data collector.[10] For the purposes of the DO-178A guidelines, GCS_SIM takes the place of the hardware system. GCS_SIM is designed to allow an experimenter to test an arbitrary number of independent implementations of the Guidance and Control Software planetary lander problem in a multi tasking environment.

In the most general sense, the purpose of GCS_SIM is to run GCS implementations and to collect data on these programs as it runs. In doing so, it provides an interface between data storage and the programs. Since these programs are one side of a control-response feedback loop, GCS_SIM also provides the response model for the loop. In an overall system view, GCS_SIM can be partitioned into a modeling component and a file interfacing component. The modeling component communicates with the control programs, determines error status, and emulates the system response. The file interfacing component is responsible for loading data necessary for the current simulation, monitoring the data collected and generated by the modeling side, and recording the necessary output data.

The interface between the simulator and the applications provides a mechanism for transferring simulation inputs and outputs as well as event-driven synchronization. In the VAX/VMS environment, there are many ways of implementing these operations. GCS_SIM was designed to take advantage of some built-in VMS system features. In an effort to reduce test case execution time while still providing flexibility, the GCS_SIM design uses synchronization flags (semaphores) and global (shared) memory

[10] For a detailed description of GCS_SIM, refer to the *GCS Simulator System Description*.

between the the simulator and GCS implementation(s) in order to transfer data between the respective processes. Although the simulator can "peek" into a GCS implementation's shared memory area at any time during a trajectory, GCS_SIM only reviews information in the implementation's shared memory area at synchronization points when variable contents are stable. Note that the simulator responds to synchronization flags that are initiated by the GCS implementations and assumes control over all GCS implementations at the sub-frame synchronization points.

The test plan that is to be used to help validate GCS_SIM will be developed outside of RTI, since the GCS_SIM design is the result of the collaboration of the RTI project staff. The DO-178A guidelines do not address the need for validation of a simulator, but imply that the output of the simulator should be predictable and consistent for each run.

7 Conclusion

This document gives the reader an overview of the Guidance and Control Software project. This project is part of a larger experiment funded by NASA-Langley Research Center, "Software Error Studies Research." The GCS Project is the third and most complex project in the experiment. The data collected during the GCS project will be analyzed at the Research Triangle Institute to meet the overall goals of the Software Error Studies Experiment and to evaluate the DO-178A guidelines and draw conclusions about the effectiveness of various software development techniques. Further reading on this subject is listed in the reference Section as numbers [6], [7], [13] and [12].

References

[1] *Guide to VAX DEC/Code Management System.* Digital Equipment Corporation, Maynard, Massachusetts, April 1987.

[2] *Teamwork/SA User's Guide.* Cadre Technologies, Inc., Providence, Rhode Island, 3.0 edition, June 1988.

[3] RTCA Special Committee 152. *Software Considerations in Airborne Systems and Equipment Certification.* Technical Report RTCA/DO-178A, Radio Technical Commission for Aeronautics, March 1985.

[4] V.R. Basili and D.M. Weiss. *A Methodology for Collecting Valid Software Engineering Data.* Technical Report Computer Science Technical Report Series, TR-1235, University of Maryland, College Park, Maryland, 1982.

[5] Richard L. Burden, J. Douglas Faires, and Albert C. Reynolds. *Numerical Analysis.* Prindle, Weber and Schmidt, Boston, Massachusetts, second edition, 1981.

[6] Janet R. Dunham and Linda A. Lauterbach. Reliability analysis of a three version software system. In *Proceedings of COMPSAC'86*, 1986.

[7] George B. Finelli. Results of software error-data experiments. In *AIAA/AHS/ASEE Aircraft Design, Systems and Operations Conference*, Atlanta, GA, September 1988.

[8] Derek J. Hatley and Imtiaz A. Pirbhai. *Strategies for Real-Time System Specification.* Dorset House Publishing Company, New York, New York, 1987.

[9] William Hetzel. *The Complete Guide to Software Testing.* QED Information Sciences, Inc., Wellesley, Massachusetts, 1984.

[10] Ragui F. Kamel. Effect of modularity on system evolution. *IEEE Software*, 48-54, January 1987.

[11] Thomas J. McCabe. *Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric.* NBS Special Publication 500-99, National Bureau of Standards, December 1982.

[12] Gerard E. Migneault. On requirements for software fault tolerance for flight controls. In *AIAA Computers in Aerospace IV Conference*, October 1983.

[13] Gerard E. Migneault. Software reliability and advanced avionics. In *National Computer Conference, 1980*, 1980.

[14] Glenford J. Myers. *The Art of Software Testing.* John Wiley & Sons, New York, New York, 1979.

[15] RTI and SAIC. *Software Reliability Measurement/Test Integration Techniques: Instructions for Testers.* Contract F30602-86-C-0269, prepared by Research Triangle Institute for Science Submitted to Rome Air Development Center.

[16] Anita M. Shagnea and B. Edward Withers. *GCS Specification Accuracy Analysis Plan.* Technical Report, Research Triangle Institute, Research Triangle Park, NC 27709, 1989.

[17] George B. Thomas, Jr. and Ross L. Finney. *Calculus and Analytic Geometry.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1979.

Report Documentation Page

1. Report No	2. Government Accession No.	3. Recipient's Catalog No
NASA CR-181972		

4. Title and Subtitle
GCS Plan for Software Aspects of Certification

5. Report Date
February 1990

6. Performing Organization Code

7. Author(s)
A. M. Shagnea, D. S. Lowman, and B. E. Withers

8. Performing Organization Report No.

10. Work Unit No.
505-66-21-03

9. Performing Organization Name and Address
Research Triangle Institute
P. O. Box 12194
Research Triangle Park, NC 27709

11. Contract or Grant No.
NAS1-17964

13. Type of Report and Period Covered
Contractor Report

12. Sponsoring Agency Name and Address
NASA Langley Research Center
Hampton, VA 23665-5225

14. Sponsoring Agency Code

15. Supplementary Notes
Technical Monitor: George B. Finelli, Langley Research Center, Hampton, Virginia.
RTCA DO-178A Document Number 14

Prepared under NASA contract NAS1-17964, Task 8

16. Abstract
As part of the Guidance and Control Software (GCS) research project being sponsored by NASA to evaluate the failure processes of software, standard industry software development procedures are being employed. To ensure that these procedures are authentic, the guidelines outlined in the Radio Technical Commission for Aeronautics RTCA/DO-178A Document entitled, "Software Considerations in Airborne Systems and Equipment Certification," have been adopted. A major aspect of these guidelines is proper documentation. As such, this report, the Plan for Software Aspects of Certification, has been produced in accordance with DO-178A. This report gives an overview of the GCS research project, including the goals of the project, project organization, and project schedules. It also specifies the plans for all aspects of the project which relate to the certification of the GCS implementations developed under a NASA contract. These plans include decisions made regarding the software specification, accuaracy requirements, configuration management, implementation development and verification, and the development of the GCS simulator.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Software Certification Documentation	Unclassified-Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	35	A03

NASA FORM 1626 OCT 86