Centerwide System Level Work Instruction **ISO 9001 - Ames Research Center**	**Document #:** **53.ARC.0004.1.1**	**Rev.:** **2**
Title: **Guidelines for Implementing 53.ARC.0004.1**		Page #: **1 of 10**

REVISION HISTORY			
REV	**Description of Change**	**Author**	**Effective Date**
0	Initial release based on 7/98 DNV Audit and 6/98 Internal Audit (see DCR 98-028).	M. Hines	9/25/98
1	Administrative change (DCR 98-047)	R. Serrano	12/18/98
2	Administrative change (DCR 99-070)	G. Miyahara	11/29/99

REFERENCE DOCUMENTS	
Document Number	**Document Title**
NASA-STD-2100	NASA Software Documentation Standard
NASA-STD 8719.13	NASA Software Safety Standard
ANSI/AASQC Q9000-3	Guidelines for the Application of ANSI/ASQC Q9001 to the Development, Supply, and Maintenance of Software
ANSI/ASQC Q9001	Quality Systems-Model for Quality Assurance in Design, Development, Production, Installation, and Servicing
ANSI/ASQC Q10007	Quality Management-Guidelines for Configuration Management
IEEE Standard 982.1	IEEE Standard Dictionary of Measures to Produce Reliable Software
53.ARC.0004.1	Project Management for the Design, Development, and Maintenance of Software
53.ARC.0009.4	Program and Project Management
53.ARC.0014	Corrective and Preventive Action

Documents referenced in this procedure are applicable to the extent specified herein.

1. Purpose

This work instruction provides guidance on implementing 53.ARC.0004.1.

2. Scope

This section does not apply to this document.

3. Definitions and Acronyms

There are no unique terms or acronyms in this document.

4. Flowchart

There is no flowchart required for this document.

5. Responsibilities

Responsibilities are addressed within the Procedure section of this document.

6. Procedure

6.1 Project Context

6.1.1 Project Manager should be aware of the types of problems to be addressed and have the appropriate experience to complete the project.

6.1.2 Software cannot be considered in isolation, and more commonly, software developments are sub-projects within parent projects. In this case, each software development is regarded as a software project and sections 6.3 through 6.4 of 53.ARC.0004.1 will apply.

6.1.3 Software sub-projects can be documented at various risk levels. These are reflected in the parent project's Project Plan but should follow procedures appropriate to their individual levels of risk. A parent project's risk level can be no less than the level of its riskiest sub-project. Sub-projects less risky than the project as a whole need only be documented at their own level.

6.2 Project Characterization

When the software project is to change an existing software application, the choice of control should be based upon the level applied to the original development. When the level is being increased above that used for the original development, the original application should be baselined to meet the risks being undertaken in the software project. Evidence of baselining should include proof that the application meets the users requirements and that the risks of using the software and consequences of failure have been established. See Appendix A for suggested controls of software projects.

6.3 Development Approach

6.3.1 Rapid Prototyping is most useful for projects with a strong user-interface component and an available sample of typical users.

6.3.2 Single-Release Life Cycle is most useful for projects with a minor or well-understood user interface component or with a short intended lifetime and well-defined input/output relationships.

6.3.3 Multiple-Release Life Cycle is most useful for large, complex projects with requirements that are expected to evolve with time.

6.3.4 Big Bang is most useful for projects whose result is the software's output or for a project developed to meet an emergency requirement.

6.4 Documentation

6.4.1 If the project is more than Very Low Cost and Negligible Risk, an Installation Guide will also be provided. Alternatively, for Multiple-Release Life Cycle projects, a Version Description Document will also be supplied with each release.

6.4.2 If the development approach is not Multiple-Release Life Cycle, is Simple and Very Low Cost, and carries no more than Low Risk, the Design Document may consist of comment lines in the code. In this situation, the User Reference may consist of a short on-line usage specification (such as a UNIX "man page," an MS Windows help file, or a "readme" file), and the Project Plan, the Requirements Document, and the Test Plan may consist of entries in the Project Manager's notebook.

6.4.3 If the project is not Very Complex, is no more than High Cost, and carries no more than Low Risk, the Project Plan will contain sections for a Documentation Plan and a Maintenance Plan. If the development approach chosen is Multiple-Release Life Cycle, the Project Plan will include a Release Plan section.

6.4.4 If the project is Very High Cost or Medium Risk, project documentation will also include a Risk Management Plan.

6.4.5 If the project is Complex, the Project Plan will include sections for Coding Standards, a Software Assurance Plan, and a Configuration Management Plan.

6.4.6 If the project is High Risk, Extremely High Cost, or Very Complex, project documentation will also include a Risk Management Plan, an Administrator Reference, and a User Guide. The Project Plan will contain sections for Coding Standards, a Documentation Plan, a Maintenance Plan, a Software Assurance Plan, and a Configuration Management Plan. If the development approach chosen is Multiple-Release Life Cycle, the Project Plan will include a Release Plan section.

6.4.7 If the Project is Medium Risk or High Risk, the Project Plan will contain:

? the work breakdown structure,

? the schedule,

? the project deliverables requiring sign-off and the process for their acceptance,

? the means of monitoring the implementation of the Project Plan,

? a mechanism for accommodating customer complaints, requests, comments, and compliments, and

? a mechanism for preventive action to eliminate potential problems.

6.5 Software Configuration Control

6.5.1 If the project is Simple and Low Cost and carries Negligible Risk, configuration control may consist of maintenance by the developers of master copies of the software and documentation.

6.5.2 If the project is Complex, machine-readable versions of all documents and source code will be maintained under the control of an automated configuration management system. Hard copy versions of documents and object language versions of programs will contain labels that identify the corresponding version of the controlled file.

6.5.3 If the project is at least High Cost or Medium Risk, all original source code will be peer code reviewed prior its inclusion in a release.

6.5.4 If the project is High Risk, Extremely High Cost, or Very Complex, machine-readable versions of all documents and source code will be maintained under the control of an automated configuration management system. Hard copy versions of documents and object language versions of programs will contain labels that identify the corresponding version of the controlled file.

6.6 Change Request Tracking

6.6.1 If the project does not use the Multiple-Release Life Cycle development approach, is Simple, is no more than Medium Cost, and carries no more than Low Risk, problem and change request tracking may be informal.

6.6.2 If the project uses the Multiple-Release Life Cycle development approach, is Complex and High Cost, or carries Medium Risk, formal, automated change request tracking will be used.

6.7 Metrics

6.7.1 If the project is not Very Complex, no more than Medium Cost, and carries no more than Low Risk, reported metrics and their reporting intervals may be limited to those required by 53.ARC.0009.4. See Appendix B for a description of some standard software metrics.

6.7.2 If the project uses the Multiple-Release Life Cycle development approach, is at least High Cost, or carries at least Medium Risk, the count of unresolved trouble reports over time will be maintained as a project metric in addition to any other required metrics and will be reportable on demand.

6.7.3 If the project is Very Complex, Extremely High Cost, or High Risk, automated estimates of code complexity and maintainability will be kept in addition to any other required metrics.

6.8 Reviews

6.8.1 If the project is Simple, Very Low Cost, and Negligible Risk, the Project Plan review and the Requirements review may be conducted informally by Line Management. In this case, Line Management also acts as "customer."

6.8.2 If the project is at least Medium Risk, the Project Plan will be reviewed by the System Safety and Mission Assurance Function and by applicable Directors as appropriate.

6.8.3 If the project is High Risk, Extremely High Cost, or Very Complex, initial editions of each required document and any updates to them will be given formal review. All code and code changes will be peer reviewed. Additionally, a formal Release Readiness Review will consider the full set of test results before deciding that the software or the release of the software is ready.

7. Metrics

There are no metrics required for this document.

8. Quality Records

There are no Quality Records required for this document.

9. Form(s)

There are no forms required for this document.

Appendix A Suggested Control for Software Projects

Item	Suggested Minimum Requirement	Negligible Risk	Low Risk	Medium Risk	High Risk	Comments
User Requirements	User/Purchasers Requirements Document	Required - Notebook entry Approved by customer	Required - Modified template ok Approved by customer	Required - Approved by customer	Required - Approved by customer	
Software Requirements	Software Requirements Document (SRD)	Not required	Required	Required - Approved by customer	Required - Approved by customer Verified	See NASA Std-2100
Software Design	Software Design Document (SDD), a.k.a., Architectural Design /Top Level Design/ & Preliminary Design	Not required Code comments to provide required documentation	Required if included in life cycle	Required if included in life cycle Verified against SRD	Required - Approved by customer Verified against SRD	See NASA Std-2100
Detail Design (DD)	Detailed Design Document (DDD) Software Users Manual (SUM)	Not required	Required	Required	Required - Verified against SDD	In some life cycles these two phases are combined resulting in only the DD
Implementation/ Programming	Coding Standards	Required	Required	Required	Code verified against SDD	

Appendix A Suggested Control for Software Projects, continued

Item	Suggested Minimum Requirement	Negligible Risk	Low Risk	Medium Risk	High Risk	Comments
Testing	Full test coverage	Required - Notebook entry	Required	Required	Indepen-dent Verification & Validation	See NASA Std-2100, ANSI/ASQC Q9000-3
Software Reuse	Developed to same standard	Not required	Optional	Required	Minimum or reversed engineered	
User Documentation	Full user instructions	Required	Required	Required	Required	See NASA Std-2100
Operations & Maintenance (Ongoing Support)	Software Maintenance Plan (SMP) Acceptance Test Specification (ATS)	Not required	Required if ongoing support is to be provided	Required	Required - Approved by customer	See NASA Std-2100, ANSI/ASQC Q9001, ANSI/ASQC Q9000-3
Configuration Management	Baseline definition and changes approved	Developer responsible for id & controlling items	Required	Required	Required	See NASA Std-2100, ANSI/ASQC Q9000-3 and ANSI/ASQC 10007
Source Code & Media	Held by Librarian or Project Manager Regular backup	Held by developer Backup as required	Required	Required	Required - Copy held remotely	See ANSI/ASQC Q9000-3
Problem reporting & change management	Recorded in file or electronic tool	Required - Logbook entry	Required	Required	Required - Changes verified	See 53.ARC.0014
Verification & Validation	Test schedule, test results, s/w version #s, peer reviews	Validation of User Req'ts or SRD Recorded in Logbook	Required	Required -	Required - IV&V Plan	
Software Assurance & Reviews	Quality system certified to ISO 9001	Not required	Peer checking or Software, Software Assurance	Required - SQA indepen-dent or part of project	Required - SQA indepen-dent of project	See NASA Std-2100 and ANSI/ASQC Q9000-3

			(SQA)			

Appendix A Suggested Control for Software Projects, continued

Item	Suggested Minimum Requirement	Negligible Risk	Low Risk	Medium Risk	High Risk	Comments
Software Acquisition	Baselined on receipt	Required - Logbook entry	Required	Required	Required - Conform to project standard(s)	See ANSI/ASQC Q9001 and ANSI/ASQC Q9000-3
Software Risk Management	Risk Assessment Risk Management Plan (RMP)	Risk Assessment	Risk Assessment	Both required	Both required	See NASA Std-2100
Software Safety	Hazard Analysis Report Software Safety Plan	Not required	Not required	Hazard Analysis Report required Software Safety Plan required if any software is identified as safety-critical	Required	See NASA-STD 8719.13
Metrics	Metrics Report	Not required	Metrics report every six months	Required - Each phase	Required - Each phase	
Software Project Management	Project Plan	Required - Can be document in Notebook	Required	Required	Required	See NASA Std-2100 and ANSI/ASQC Q9000-3
Project Close-out	Report	Required - Logbook entry ok	Required	Required	Required	

Appendix B Software Metrics

IEEE Standard 982.1 defines possible metrics, or measures, to be applied to the software development process and resulting product. These metrics are defined in this appendix. More detailed definitions of, and implementation procedures for, the noted examples can be found in the IEEE standard.

B.1 Process Metrics

Process metrics are applied to the activities of development, test, and maintenance. The Project Manager will identify, apply, and report on metrics as required by the appropriate project level of control. These metrics are divided into three categories:

B.1.1 Management Control

Management control metrics assess the guidance of the software development and maintenance process. Examples include man-hours per major defect detected, fault-days number, defect indices, and error distribution(s).

B.1.2 Coverage

Coverage metrics assess the presence of all necessary activities to develop or maintain the software product. Examples include requirements traceability, functionality test coverage, software documentation and source code listing, and required software reliability.

B.1.3 Risk, Benefit, and Cost Evaluation

Risk, benefit, and cost evaluation metrics assess the process trade-offs of cost, schedule, and performance. Examples include cumulative failure profile, distribution of errors in the software, Required Software Reliability, and Software Release Readiness.

B.2 Product Metrics

Product metrics are applied to the software objects produced. The Project Manager will identify, apply, and report on metrics as required by the appropriate project level of control. The metrics will address the cause and effect of the static and dynamic aspects of both projected reliability prior to operation and operational reliability. These metrics are divided into six categories.

B.2.1 Errors; Faults; Failures

These metrics measure the count of defects with respect to human cause, program bugs, and observed system malfunctions. Examples include cumulative failure profile, fault density, and fault-day number (length of time to resolve faults).

B.2.2 Mean-Time-To-Failure; Failure Rate

These metrics are derivative measures of defect occurrence and time. Examples include Mean-Time-To-Failure and Failure Rate.

B.2.3 Reliability Growth and Projection

These metrics assess the change in faultiness of the product under testing and in operation. Examples include software maturity model, reliability growth function, and combined hardware and software system operational availability.

B.2.4 Remaining Product Faults

These metrics assess the faultiness of the product in development, test, or maintenance. Examples include software science measures, estimation of number of faults remaining, failure analysis using elapsed time, and test accuracy.

B.2.5 Completeness and Consistency

These metrics assess the presence and agreement of all necessary software system parts. Examples include requirements compliance, functional test coverage, and software documentation and source listing.

B.2.6 Complexity

These metrics assess the complicating factors in a system. Examples include number of entries/exits per module, software science measures, and design structure.