NOAA NESDIS CENTER for SATELLITE APPLICATIONS and RESEARCH (STAR)

STAKEHOLDER GUIDELINE

SG-16
DEVELOPMENT PROGRAMMER GUIDELINES

Version 3.0

TITLE: SG-16: DEVELOPMENT PROGRAMMER GUIDELINES VERSION 3.0

AUTHORS:

Ken Jensen (Raytheon Information Solutions)

VERSION HISTORY SUMMARY

Version	**Description**	**Revised Sections**	**Date**
1.0	No version 1		
2.0	No version 2		
3.0	New Stakeholder Guideline adapted from CMMI guidelines by Ken Jensen (Raytheon Information Solutions)	New Document	12/31/2009

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ATBD	Algorithm Theoretical Basis Document
BB	Baseline Build
CDD	Critical Design Document
CDR	Critical Design Review
CDRR	Critical Design Review Report
CI	Cooperative Institute
CICS	Cooperative Institute for Climate Studies
CIMSS	Cooperative Institute for Meteorological Satellite Studies
CIOSS	Cooperative Institute for Oceanographic Satellite Studies
CIRA	Cooperative Institute for Research in the Atmosphere
CL	Check List
CLI	Check List Item
CoRP	Cooperative Research Program
CM/DM	Configuration Management/Data Management
CMMI	Capability Maturity Model Integration
CPI	Cost Performance Index
CREST	Cooperative Remote Sensing and Technology Center
CTD	Code Test Document
CTR	Code Test Review
CTRR	Code Test Review Report
DDD	Detailed Design Document
DG	Document Guidelines
DPP	Development Project Plan
DPR	Development Project Report
EPG	Enterprise Process Group
EPL	Enterprise Product Lifecycle
EUM	External Users Manual
EVMS	Earned Value Management System
G1R	Gate1 Review
G1RR	Gate1 Review Report
G2R	Gate 2 Review

G2RR	Gate 2 Review Report
G3R	Gate 3 Review
G3RR	Gate 3 Review Report
G4R	Gate 4 Review
G4RR	Gate 4 Review Report
G5R	Gate 5 Review
G5RR	Gate 5 Review Report
IMP	Integrated Master Plan
IMS	Integrated Master Schedule
IPT	Integrated Product Team
IT	Information Technology
IUM	Internal Users Manual
MDD	Metadata Document
MOU	Memorandum Of Understanding
NESDIS	National Environmental Satellite, Data, and Information Service
NOAA	National Oceanic and Atmospheric Administration
OCD	Operations Concept Document
PAR	Process Asset Repository
PBR	Project Baseline Report
PCOD	Pre-Operational Code
PDD	Preliminary Design Document
PDR	Preliminary Design Review
PDRR	Preliminary Design Review Report
PG	Process Guidelines
PP	Project Proposal
PRG	Peer Review Guidelines
PRD	Project Requirements Document
PRR	Project Requirements Review
PRRR	Project Requirements Review Report
PSR	Project Status Report
PTEST	Pre-Operational Test Data
QA	Quality Assurance
R&D	Research & Development
RAD	Requirements Allocation Document

RAS	Requirements Allocation Sheet
RCOD	Research Code
RNM	Requirements/Needs Matrix
RTEST	Research Test Data
SC	Steering Committee
SEI	Software Engineering Institute
SG	Stakeholder Guideline
SOW	Statement Of Work
SPI	Schedule Performance Index
SPSRB	Satellite Products and Services Review Board
SRD	System Readiness Document
SRR	System Readiness Review
SRRR	System Readiness Review Report
STAR	Center for Satellite Applications and Research
STP	System Test Plan
SWA	Software Architecture Document
TBD	To Be Determined
TD	Training Document
TG	Task Guideline
TRD	Test Readiness Document
TRR	Test Readiness Review
TRRR	Test Readiness Review Report
UTP	Unit Test Plan
UTR	Unit Test Report
VVP	Verification and Validation Plan
VVR	Verification and Validation Report
WBS	Work Breakdown Structure

1. INTRODUCTION

The NOAA/NESDIS Center for Satellite Applications and Research (STAR) develops a diverse spectrum of complex, often interrelated, environmental algorithms and software systems. These systems are developed through extensive research programs, and transitioned from research to operations when a sufficient level of maturity and end-user acceptance is achieved. Progress is often iterative, with subsequent deliveries providing additional robustness and functionality. Development and deployment is distributed, involving STAR, the Cooperative Institutes (CICS[1], CIMSS[2], CIOSS[3], CIRA[4], CREST[5]) distributed throughout the US, multiple support contractors, and NESDIS Operations.

NESDIS/STAR is implementing an increased level of process maturity to support the development of these software systems from research to operations. This document is a Stakeholder Guideline (SG) for users of this process, which has been designated as the STAR Enterprise Product Lifecycle (EPL).

1.1. Objective

The STAR Enterprise is comprised of a large number of organizations that participate and cooperate in the development and production of environmental satellite data products and services. Individual project teams are customarily composed of personnel from these organizations, supplemented by contractor personnel. These organizations and project teams are referred to as the STAR Enterprise stakeholders.

The objective of this Stakeholder Guideline (SG-16) is to provide a detailed description of the standard tasks of a **Development Programmer**. The intended users of this SG are those who have been assigned as testers on a STAR development project's Integrated Product Team (IPT).

A ***Development Programmer*** is a programmer who has been assigned to one or more of the tasks of preliminary design and detailed design of pre-operational code, writing pre-

1 Cooperative Institute for Climate Studies

2 Cooperative Institute for Meteorological Satellite Studies

3 Cooperative Institute for Oceanographic Satellite Studies

4 Cooperative Institute for Research in the Atmosphere

5 Cooperative Remote Sensing and Technology Center

operational code, integrating code into a pre-operational system, and supporting Development Testers in testing pre-operational code.

Stakeholder satisfaction is a critical component of the process. The intention is for the process to be more of a benefit that a burden to stakeholders. If stakeholders are not satisfied that this is the case, the process will require improvement.

Comments and suggestions for improvement of the process architecture, assets, artifacts and tools are always welcome. Stakeholders can provide feedback by contacting:

Ken.Jensen@noaa.gov

1.2. Version History

This is the first version of SG-14. It is identified as version 3.0 to align it with the release of the version 3.0 STAR EPL process assets.

1.3. Overview

This SG contains the following sections:

2. REFERENCE DOCUMENTS

All of the reference documents for the STAR EPL process are STAR EPL process assets that are accessible in a Process Asset Repository (PAR) on the STAR website. http://www.star.nesdis.noaa.gov/star/EPL_index.php.

Process assets include:

- Process Guidelines
- Stakeholder Guidelines
- Task Guidelines
- Peer Review Guidelines
- Review Check Lists
- Document Guidelines
- Training Documents

2.1. Process Guidelines

Process Guideline (PG) documents describe STAR's standard set of practices and guidelines for tailoring them to specific projects.

- STAR EPL Process Guidelines (PG-1)
- STAR EPL Process Guidelines Appendix (PG-1.A)

PG-1 and PG-1.A apply generally to each EPL step. Each stakeholder performing tasks during each step can benefit from a familiarity with these documents.

2.2. Stakeholder Guidelines

A Stakeholder Guideline (SG) is a description of how to perform all STAR EPL standard tasks assigned to a given type of stakeholder. For each type of stakeholder, the appropriate SG provides that stakeholder with a complete description of the standard tasks for that stakeholder role, along with references to all appropriate process assets and project artifacts. This functions as a complement to the Task Guidelines (TGs), which provide a completion description of all stakeholder tasks for a specific process step. The relevant SG for **Development Programmers** is SG-16 (this document).

2.3. Task Guidelines

The STAR EPL is designed as a sequence of 11 process steps that take a product from initial conception through delivery to operations. These steps are:

- Step 1 - Basic Research
- Step 2 - Focused R & D
- Step 3 - Project Proposal
- Step 4 - Resource Identification
- Step 5 - Project Plan
- Step 6 - Project Requirements
- Step 7 - Preliminary Design
- Step 8 - Detailed Design
- Step 9 - Code & Test Data Development
- Step 10 - Code Test And Refinement
- Step 11 - System Integration and Test

A Task Guideline (TG) is a description of how to perform the tasks of a STAR EPL process step. There is one Task Guideline for each step in the STAR EPL. Table 2.3.1 lists the Task Guidelines that are relevant for **Development Programmers**.

TABLE 2.3.1 – Relevant Task Guidelines

ID	Step
TG-5	Project Plan
TG-6	Project Requirements
TG-7	Preliminary Design
TG-8	Detailed Design
TG-9	Code and Test Data Development

TG-10	Code Test and Refinement
TG-11	System Integration and Test

2.4. Peer Review Guidelines

For each review (c.f. Section 4), there is a Peer Review Guideline (PRG) that describes the objectives of the review, the required artifacts, standards for reviewers, requirements for approval, and options other than approval. Table 2.4.1 lists the Peer Review Guidelines that are relevant for **Development Programmers**.

TABLE 2.4.1 – Relevant Peer Review Guidelines

ID	Review
PRG-5	Gate 3 Review
PRG-6	Project Requirements Review
PRG-7	Preliminary Design Review
PRG-8.1	Critical Design Review
PRG-8.2	Gate 4 Review
PRG-9	Test Readiness Review
PRG-10	Code Test Review
PRG-11.1	System Readiness Review
PRG-11.2	Gate 5 Review

2.5. Review Check Lists

For each review (c.f. Section 4), there is a Review Check List (CL) that captures all the objectives for a review as a set of check list items. Each item in the check list should have a "Disposition" column that contains "Pass", "Conditional Pass", "Defer", "Waive", or "N/A" (Not Applicable). Each item will also have columns for Risk Assessment and for Actions generated. Table 2.5.1 lists the Review Check Lists that are relevant for **Development Programmers**.

TABLE 2.5.1 – Relevant Review Check Lists

ID	Review
CL-5	Gate 3 Review
CL-6	Project Requirements Review
CL-7	Preliminary Design Review
CL-8.1	Critical Design Review
CL-8.2	Gate 4 Review
CL-9	Test Readiness Review
CL-10	Code Test Review
CL-11.1	System Readiness Review
CL-11.2	Gate 5 Review

2.6. Document Guidelines

There is a Document Guideline (DG) for each standard STAR EPL document. Each DG includes a description of the purpose for the document, a standard document outline (table of contents), a brief description of each subsection in the outline, and an Appendix containing an example document.

Table 2.6.1 lists the Document Guidelines that are relevant for **Development Programmers**.

TABLE 2.6.1 – Relevant Document Guidelines

ID	Document
DG-0.1	Document Style Guideline
DG-1.2	Software Architecture Document (SWA)
DG-5.1	Development Project Plan (DPP)
DG-5.2	Project Status Report (PSR)
DG-5.2.A	PSR Appendix
DG-5.3	Gate 3 Document (G3D)

DG-5.3.A	G3D Appendix
DG-6.2	Requirements Allocation Document (RAD)
DG-6.4	Project Requirements Document (PRD)
DG-6.4.A	PRD Appendix
DG-7.1	Preliminary Design Document (PDD)
DG-7.1.A	PDD Appendix
DG-8.1	Detailed Design Document (DDD)
DG-8.2	Critical Design Document (CDD)
DG-8.2.A	CDD Appendix
DG-8.4	Gate 4 Document (G4D)
DG-8.4.A	G4D Appendix
DG-9.1	Unit Test Plan (UTP)
DG-9.2	Test Readiness Document (TRD)
DG-9.2.A	TRD Appendix
DG-10.1	Unit Test Report (UTR)
DG-10.2	System Test Plan (STP)
DG-10.3	Code Test Document (CTD)
DG-10.3.A	CTD Appendix
DG-11.3	Metadata Document (MDD)
DG-11.4	Verification and Validation Report (VVR)
DG-11.5	System Readiness Document (SRD)
DG-11.5.A	SRD Appendix
DG-11.7	Gate 5 Document (G5D)
DG-11.7.A	G5D Appendix
DG-11.9	Development Project Report (DPR)

2.7. Training Documents

Training Documents (TD) assist the stakeholders (c.f. Section 3) in performing the process tasks. By using the TDs, the stakeholders should be able to perform the tasks more effectively.

Table 2.7.1 lists the Training Documents that are relevant for **Development Programmers**.

TABLE 2.7.1 – Relevant Training Documents

ID	Training Document
TD-9	Project Requirements
TD-11.1	FORTRAN Programming Standards and Guidelines
TD-11.1.A	Transition from Fortran 77 to Fortran 90
TD-11.2	C Programming Standards and Guidelines

3. REVIEWS

The relevant reviews for **Development Programmers** are:

- Gate 3 Review (G3R)
- Project Requirements Review (PRR)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)
- Gate 4 Review (G4R)
- Test Readiness Review (TRR)
- Code Test Review (CTR)
- System Readiness review (SRR)
- Gate 5 Review (G5R)

3.1. Gate 3 Review

Gate 3 is a STAR review of the project's readiness for development. Its purpose is to determine whether the development plan is feasible, the identified resources are available, and the identified risks are manageable. If a project passes Gate 3, the project proceeds to the Design phase.

Standard Gate 3 Review objectives:

- Identify relevant stakeholders and their planned involvement according to the project plan.
- Review the planned work tasks and Work Breakdown Structure (WBS)
- Review the planned project lifecycle
- Review the planned review objectives, entry criteria, exit criteria, and check lists
- Review the planned work products and project artifacts
- Review the Integrated Master Plan (IMP) and Integrated Master Schedule (IMS)
- Review the expected costs and funding

- Provide an initial assessment of project risks

Standard Gate 3 Review entry criteria:

- Entry # 1 - A Development Project Plan (DPP) has been written. The Gate 3 reviewers have access to the current baseline version of the DPP.
- Entry # 2 - A Project Status Report (PSR) has been written. The Gate 3 reviewers have access to the current baseline version of the PSR.
- Entry # 3 - A Gate 3 Document (G3D) has been written. The Gate 3 reviewers have access to the current baseline version of the G3D.
- Entry # 4 - A Project Baseline Report (PBR) has been written. The Gate 3 reviewers have access to the current baseline version of the PBR.

Standard Gate 3 Review exit criteria:

- Exit # 1 - Project plan and DPP are satisfactory.
- Exit # 2 - Project status and PSR are satisfactory.
- Exit # 3 - Project baseline and PBR are satisfactory.
- Exit # 4 - Project risks are acceptable.
- Exit # 5 - Status of risk mitigation actions is acceptable
- Exit # 6 - Project is ready for the Design phase

Refer to PRG-5 for a more detailed description of the Gate 3 Review. The standard Gate 3 Review Check List Items (CLI) are documented in the process asset CL-5 (c.f. Section 2).

Gate 3 Review objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the Gate 3 Review.

3.2. Project Requirements Review

Project Requirements Review (PRR) is a Design Phase Technical Review. Its purpose is to establish the requirements to be satisfied by the project and the means to validate them. Upon completion of this review, step 7 (Preliminary Design) commences.

Standard PRR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Identify changes to the project plan and project status since the Gate 3 Review
- Translate user and operator needs and expectations into an operations concept for the product processing system
- Develop and describe the initial set of project requirements, including:
  - Basic Requirements
  - Derived Requirements
  - Requirements/Needs matrix
  - Requirements Traceability matrix
  - Requirements Quality Assurance plans and methods
  - Requirements Allocation matrix
- Identify and update project risks. Make recommendations for risk mitigation plans and actions.
- Document the closing of all action items since the Gate 3 Review. Make recommendations for open actions and new actions.

Standard PRR entry criteria:

- Entry # 1 - A Development Project Plan (DPP) has been written. The PRR reviewers have access to the current baseline version of the DPP.
- Entry # 2 - A Project Status Report (PSR) Appendix has been written. The PRR reviewers have access to the current baseline version of the PSR Appendix.
- Entry # 3 - An Operations Concept Document (OCD) has been written. The PRR reviewers have access to the current baseline version of the OCD.

- Entry # 4 - A Requirements Allocation Document (RAD) has been written. The PRR reviewers have access to the current baseline version of the RAD.
- Entry # 5 – A Verification and Validation Plan (VVP) has been written. The PRR reviewers have access to the current baseline version of the VVP.
- Entry # 6 - A Project Requirements Document (PRD) has been written. The PRR reviewers have access to the current baseline version of the PRD.
- Entry # 7 - A Project Baseline Report (PBR) has been written. PRR reviewers have access to the current baseline version of the PBR.

Standard PRR exit criteria:

- Exit # 1 - Project plan and DPP are satisfactory.
- Exit # 2 – Operations concept and OCD are satisfactory.
- Exit # 3 – Requirements identification is satisfactory.
- Exit # 4 – Requirements analysis is satisfactory.
- Exit # 5 – Requirements traceability plan is satisfactory.
- Exit # 6 – Requirements tracking plan is satisfactory.
- Exit # 7 - Requirements validation plan and VVP are satisfactory.
- Exit # 8 - Requirements allocation and RAD are satisfactory.
- Exit # 9 - Project baseline and PBR are satisfactory.
- Exit # 10 - The PRR reviewers' assessment of outstanding risks and actions is documented in the PRR Report.
- Exit # 11 - Project risks and actions are acceptable.

Refer to PRG-6 for a more detailed description of the PRR. The standard PRR Check List Items (CLI) are documented in the process asset CL-6 (c.f. Section 2).

PRR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the PRR.

3.3. Preliminary Design Review

Preliminary Design Review (PDR) is a Design Phase Technical Review. Its purpose is to assess the preliminary design for the pre-operational system. Upon completion of this review, step 8 (Detailed Design) commences.

Standard PDR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Identify requirements changes since the Project Requirements Review (PRR)
- Identify a set of alternative solutions to meet the requirements.
- Provide all applicable technical data for each alternative solution, including:
    - Operations concept
    - Theoretical basis
    - Architecture, specifications, interfaces
    - Performance requirements, QA procedures, test data requirements
    - Verification and validation plans
    - Risks and benefits
- Provide an updated allocation of requirements to product components and system components of the preliminary design.
- Identify and update project risks for the selected solution. Make recommendations for risk mitigation plans and actions.
- Document the closing of all action items since PRR. Make recommendations for open actions and new actions.

Standard PDR entry criteria:

- Entry # 1 - A Project Requirements Review Report (PRRR) has been written. The PDR reviewers have access to the current baseline version of the PRRR.
- Entry # 2 - A Development Project Plan (DPP) has been written. The PDR reviewers have access to the current baseline version of the DPP.

- Entry # 3 - An Operations Concept Document (OCD) has been written. The PDR reviewers have access to the current baseline version of the OCD.
- Entry # 4 - A Requirements Allocation Document (RAD) has been written. The PDR reviewers have access to the current baseline version of the RAD.
- Entry # 5 - An Algorithm Theoretical Basis Document (ATBD v2r0) has been written. The PDR reviewers have access to the current baseline version of the ATBD.
- Entry # 6 - A Software Architecture Document (SWA) has been written. The PDR reviewers have access to the current baseline version of the SWA.
- Entry # 7 - A Verification and Validation Plan (VVP) has been written. The PDR reviewers have access to the current baseline version of the VVP.
- Entry # 8 - A Preliminary Design Document (PDD) has been written. PDR review objectives are clearly stated in the PDD.
- Entry # 9 - A Project Baseline Report (PBR) has been written. The PDR reviewers have access to the current baseline version of the PBR.

Standard PDR exit criteria:

- Exit # 1 – PRR "Conditional Pass" items have been satisfactorily disposed of.
- Exit # 2 – PRR "Defer" items have been satisfactorily disposed of.
- Exit # 3 - Project plan and DPP are satisfactory.
- Exit # 4 - Operations concept and OCD are satisfactory.
- Exit # 5 – Requirements changes since PRR are approved.
- Exit # 6 – Algorithm theoretical basis and ATBD are satisfactory.
- Exit # 7 – Software architecture and SWA are satisfactory.
- Exit # 8 – Verification and validation plan and VVP are satisfactory.
- Exit # 9 - Requirements allocation and RAD are satisfactory.
- Exit # 10 - Project baseline and PBR are satisfactory.
- Exit # 11 - A selected solution has been consistently identified in the project artifacts.
- Exit # 12 - The selected solution is approved.
- Exit # 13 - The PDR reviewers' assessment of outstanding risks and actions is documented in the PDR Report.

- Exit # 14 - Project risks and actions are acceptable.

Refer to PRG-7 for a more detailed description of the PDR. The standard PDR Check List Items (CLI) are documented in the process asset CL-7 (c.f. Section 2).

PDR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the PDR.

3.4. Critical Design Review

Critical Design Review (CDR) is the final Design Phase Technical Review. Its purpose is to assess the detailed design for the pre-operational system. Upon successful completion of this review, a Gate 4 Review is held to determine whether the project should proceed to the Build phase.

Standard CDR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Identify requirements changes since PDR
- Provide all applicable technical data for the selected solution, including:
    - Operations concept
    - Theoretical Basis
    - Architecture, specifications, interfaces, detailed design description
    - Performance requirements, QA procedures, test data requirements
    - Verification and validation plans
- Provide an updated allocation of requirements to product components and system components of the detailed design.
- Identify and update project risks. Make recommendations for risk mitigation plans and actions.

- Document the closing of all action items since PDR. Make recommendations for open actions and new actions.

Standard CDR entry criteria:

- Entry # 1 - A Preliminary Design Review Report (PDRR) has been written. The CDR reviewers have access to the current baseline version of the PDRR.
- Entry # 2 - A Development Project Plan (DPP) has been written. The CDR reviewers have access to the current baseline version of the DPP.
- Entry # 3 - An Operations Concept Document (OCD) has been written. The CDR reviewers have access to the current baseline version of the OCD.
- Entry # 4 - A Requirements Allocation Document (RAD) has been written. The CDR reviewers have access to the current baseline version of the RAD.
- Entry # 5 - An Algorithm Theoretical Basis Document (ATBD) has been written. The CDR reviewers have access to the current baseline version of the ATBD.
- Entry # 6 - A Software Architecture Document (SWA) has been written. The CDR reviewers have access to the current baseline version of the SWA.
- Entry # 7 - A Detailed Design Document (DDD) has been written for each software unit in the software architecture. The CDR reviewers have access to the current baseline version of each DDD.
- Entry # 8 - A Verification and Validation Plan (VVP) has been written. The CDR reviewers have access to the current baseline version of the VVP.
- Entry # 9 - A Critical Design Document (CDD) has been written. CDR review objectives are clearly stated in the CDD.
- Entry # 10 - A Project Baseline Report (PBR) has been written. The CDR reviewers have access to the current baseline version of the PBR.

Standard CDR exit criteria:

- Exit # 1 - PDR "Conditional Pass" items have been satisfactorily disposed of.
- Exit # 2 - PDR "Defer" items have been satisfactorily disposed of.
- Exit # 3 – Project plan and DPP are satisfactory
- Exit # 4 - Operations concept and OCD are satisfactory.

- Exit # 5 - Requirements changes since PDR are approved.
- Exit # 6 - Algorithm theoretical basis and ATBD are satisfactory.
- Exit # 7 - Software architecture and SWA are satisfactory.
- Exit # 8 – Software detailed design and DDDs are satisfactory.
- Exit # 9 - Verification and validation plan and VVP are satisfactory.
- Exit # 10 - Requirements allocation and RAD are satisfactory.
- Exit # 11 - Project baseline and PBR are satisfactory.
- Exit # 12 - The CDRR documents the current status of project risks, actions and CDR exit criteria.
- Exit # 13 - Project risks and actions are acceptable. Project is ready for the Build phase.

Refer to PRG-7 for a more detailed description of the CDR. The standard CDR entry criteria, exit criteria, and check list is documented in the process asset CL-8.1 (c.f. Section 2).

CDR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the CDR.

3.5. Gate 4 Review

Gate 4 is a review of the project status following the CDR, under the direction of STAR. Its purpose is to determine whether the project is ready to begin development of the pre-operational code and test data. If a project passes Gate 4, the project proceeds to the Build phase.

Standard Gate 4 Review objectives:

- Review the implementation of the Integrated Master Plan (IMP) and Integrated Master Schedule (IMS)
- Review the technical status and risks of the project
- Review the cost status and risks of the project

- Review the schedule status and risks of the project
- Determine whether corrective actions are needed to allow the project to proceed to the Build phase as planned
- Determine whether a re-plan and a delta Gate 4 Review are needed.

Standard Gate 4 Review entry criteria:

- Entry # 1 - A Gate 3 Review Report (G3RR) has been written. The Gate 4 reviewers have access to the current baseline version of the G3RR.
- Entry # 2 - A Critical Design Review Report (CDRR) has been written. The Gate 4 reviewers have access to the current baseline version of the CDRR.
- Entry # 3 - A Development Project Plan (DPP) has been written. The Gate 4 reviewers have access to the current baseline version of the DPP.
- Entry # 4 - A Project Status Report (PSR) has been written. The Gate 4 reviewers have access to the current baseline version of the PSR.
- Entry # 5 - A Gate 4 Document (G4D) has been written. The Gate 4 reviewers have access to the current baseline version of the G4D.
- Entry # 6 - A Project Baseline Report (PBR) has been written. The Gate 4 reviewers have access to the current baseline version of the PBR.

Standard Gate 4 Review exit criteria:

- Exit # 1 – CDR status and CDRR are satisfactory
- Exit # 2 - Project plan and DPP are satisfactory.
- Exit # 3 - Project status and PSR are satisfactory.
- Exit # 4 - Project baseline and PBR are satisfactory.
- Exit # 5 - Project risks are acceptable.
- Exit # 6 - Status of risk mitigation actions is acceptable
- Exit # 7 - Project is ready for the Build phase

Refer to PRG-8.2 for a more detailed description of the Gate 4 Review. The standard Gate 4 Review entry criteria, exit criteria, and check list is documented in the process asset CL-8.2 (c.f. Section 2).

Gate 4 Review objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the Gate 4 Review.

3.6. Test Readiness Review

Test Readiness Review (TRR) is a Build Phase Technical Review. Its purpose is to determine whether code and test data development are sufficient to allow testing of the pre-operational software components (unit testing). Upon successful completion of this review, step 10 (Code Test and Refinement) commences.

Standard TRR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Review the software architecture, including external interfaces, and identify changes since CDR.
- Identify changes to the detailed design since CDR.
- Identify changes to the verification and validation plan since CDR.
- Demonstrate the test readiness of each unit in the software architecture.
- Provide all applicable technical data to support unit testing, including:
    - Pre-operational code and test data
    - Unit test plan
- Identify and evaluate risks. Recommend risk mitigation activities.
- Document the closing of all action items since CDR. Make recommendations for open actions and new actions.

Standard TRR entry criteria:

- Entry # 1 - A Critical Design Review Report (CDRR) has been written. The TRR reviewers have access to the current baseline version of the CDRR.
- Entry # 2 - A Development Project Plan (DPP) has been written. The TRR reviewers have access to the current baseline version of the DPP.
- Entry # 3 - A Requirements Allocation Document (RAD) has been written. The TRR reviewers have access to the current baseline version of the RAD.
- Entry # 4 - A Software Architecture Document (SWA) has been written. The TRR reviewers have access to the current baseline version of the SWA.
- Entry # 5 - A Detailed Design Document (DDD) has been written for each software unit in the software architecture. The TRR reviewers have access to the current baseline version of the DDDs.
- Entry # 6 - A Verification and Validation Plan (VVP) has been written. The TRR reviewers have access to the current baseline version of the VVP.
- Entry # 7 - A Unit Test Plan (UTP v1r0) has been written. The TRR reviewers have access to the current baseline version of the UTP.
- Entry # 8 – Pre-operational code to implement the detailed design is accessible to the TRR reviewers.
- Entry # 9 - Pre-operational test data, including "truth" data is accessible to the TRR reviewers.
- Entry # 10 - A Project Baseline Report (PBR v2r0) has been written. The TRR reviewers have access to the current baseline version of the PBR.
- Entry # 11 - A Test Readiness Document (TRD) has been written. The TRR reviewers have access to the current baseline version of the TRD.

Standard TRR exit criteria:

- Exit # 1 - CDR "Conditional Pass" items have been satisfactorily disposed of.
- Exit # 2 - CDR "Defer" items have been satisfactorily disposed of.
- Exit # 3 - Changes to the project plan since Gate 4 Review are approved.
- Exit # 4 - Requirements allocation changes since CDR are approved.
- Exit # 5 - Changes to external interfaces since CDR are approved.
- Exit # 6 - Changes to the software architecture since CDR are approved.

- Exit # 7 - Changes to the detailed design since CDR are approved.
- Exit # 8 - Changes to the verification and validation plan since CDR are approved.
- Exit # 9 - The unit test plan and UTP are satisfactory
- Exit # 10 - Pre-operational code to implement the detailed design has been written according to standards and has been built into executable units.
- Exit # 11 - Pre-operational test data, including "truth" data, are satisfactory.
- Exit # 12 - The project baseline and PBR are satisfactory.
- Exit # 13 - The project artifacts document all approved changes to requirements, requirements allocation, external interfaces, software architecture, detailed design, and verification and validation plan since the CDR.
- Exit # 14 - The TRRR documents the current status of project risks, actions and TRR exit criteria.
- Exit # 15 - Project risks and actions are acceptable. Project is ready for unit testing.

Refer to PRG-9 for a more detailed description of the TRR. The standard TRR entry criteria, exit criteria, and check list is documented in the process asset CL-9 (c.f. Section 2).

TRR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the TRR.

3.7. Code Test Review

Code Test Review (CTR) is a Build Phase Technical Review. Its purpose is to determine whether the pre-operational software units are ready for integration unto a pre-operational system. Upon successful completion of this review, step 11 (System Integration and Test) commences.

Standard CTR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Provide all applicable technical data, including:
    - Refined pre-operational code and test data
    - Unit test plan
    - Unit test report
    - System test plan
- Review the unit test plan, focusing on changes since the TRR.
- Review the unit test results
- Review the system test plan
- Identify and evaluate risks. Recommend risk mitigation activities.
- Document the closing of all action items since TRR. Make recommendations for open actions and new actions.

Standard CTR entry criteria:

- Entry # 1 - A Test Readiness Report (TRRR) has been written. The CTR reviewers have access to the current baseline version of the TRRR.
- Entry # 2 -A Development Project Plan (DPP) has been written. The CTR reviewers have access to the current baseline version of the DPP.
- Entry # 3 - A Requirements Allocation Document (RAD) has been written. The CTR reviewers have access to the current baseline version of the RAD.
- Entry # 4 - A Software Architecture Document (SWA) has been written. The CTR reviewers have access to the current baseline version of the SWA.
- Entry # 5 - Detailed Design Documents (DDDs) have been written for each software unit in the software architecture. The CTR reviewers have access to the current baseline version of each DDD.
- Entry # 6 – A Unit Test Plan (UTP) has been written. The CTR reviewers have access to the current baseline version of the UTP.

- Entry # 7 – Pre-operational code units, external interfaces, ancillary data, unit test data and unit test results are in the development test environment. The CTR reviewers have access to this code, test data and test results.
- Entry # 8 – A Unit Test Report (UTR) has been written. The CTR reviewers have access to the current baseline version of the UTR.
- Entry # 9 - A Verification and Validation Plan (VVP) has been written. The CTR reviewers have access to the current baseline version of the VVP.
- Entry # 10 - A System Test Plan (STP) has been written. The CTR reviewers have access to the current baseline version of the STP.
- Entry # 11 - A Project Baseline Report (PBR) has been written. The CTR reviewers have access to the current baseline version of the PBR.
- Entry # 12 - A Code Test Document (CTD) has been written. The CTR reviewers have access to the current baseline version of the CTD.

Standard CTR exit criteria:

- Exit # 1 - TRR "Conditional Pass" items have been satisfactorily disposed of.
- Exit # 2 - TRR "Defer" items have been satisfactorily disposed of.
- Exit # 3 – Changes to the project plan since TRR are approved.
- Exit # 4 - Requirements allocation changes since TRR are approved.
- Exit # 5 - Changes to external interfaces since TRR are approved.
- Exit # 6 - Changes to the software architecture since TRR are approved.
- Exit # 7 - Changes to the detailed design since TRR are approved.
- Exit # 8 - Changes to the verification and validation plan since TRR are approved.
- Exit # 9 – Code units and unit test data are satisfactory
- Exit # 10 – Unit test results and UTR are satisfactory
- Exit # 11 - The system test plan and STP are satisfactory
- Exit # 12 - The project baseline and PBR are satisfactory.
- Exit # 13 - The CTRR documents updated status of project risks and actions.
- Exit # 14 - Project risks and actions are acceptable. The project is ready for system integration and system testing.

Refer to PRG-10 for a more detailed description of the CTR. The standard CTR entry criteria, exit criteria, and check list is documented in the process asset CL-10 (c.f. Section 2).

CTR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the CTR.

3.8. System Readiness Review

System Readiness Review (SRR) is the final Build Phase Technical Review prior to Gate 5. Its purpose is to determine whether the pre-operational product system satisfies its functional and performance requirements, and is ready for installation in the operations environment. Upon successful completion of SRR, preparations are made for a Gate 5 review of readiness for transition to operations.

Standard SRR objectives:

- Identify relevant stakeholders and document their involvement according to the project plan.
- Review the CTRR, identifying risks and actions to be addressed
- Review the system requirements, identifying requirements and requirements allocation changes since CTR.
- Review the system description, including external interfaces, software architecture and detailed design, identifying changes since CTR.
- Review and confirm the system readiness for operations and maintenance, based on the results of system testing and the availability of required code and operations documentation.
- Review and confirm the system readiness for users, based on the results of system testing and the availability of required user documentation.
- Identify and evaluate risks. Recommend risk mitigation activities.
- Review the status of all actions identified to mitigate risks. Make recommendations for open actions and new actions.

Standard SRR entry criteria:

- Entry # 1 - A Code Test Review Report (CTRR) has been written. The SRR reviewers have access to the current baseline version of the CTRR.
- Entry # 2 - A Development Project Plan (DPP) has been written. The SRR reviewers have access to the current baseline version of the DPP.
- Entry # 3 - An Operations Concept Document (OCD) has been written. The SRR reviewers have access to the current baseline version of the OCD.
- Entry # 4 - A Requirements Allocation Document (RAD) has been written. The SRR reviewers have access to the current baseline version of the RAD.
- Entry # 5 - An Algorithm Theoretical Basis Document (ATBD) has been written. The SRR reviewers have access to the current baseline version of the ATBD.
- Entry # 6 -A Software Architecture Document (SWA) has been written. The SRR reviewers have access to the current baseline version of the SWA.
- Entry # 7 - A Detailed Design Document (DDD) for each software unit has been written. The SRR reviewers have access to the current baseline version of the DDDs.
- Entry # 8 -An Internal Users Manual (IUM) has been written. The SRR reviewers have access to the current baseline version of the IUM.
- Entry # 9 - An External Users Manual (EUM) has been written. The SRR reviewers have access to the current baseline version of the EUM.
- Entry # 10 - A Metadata Document (MDD) has been written. The SRR reviewers have access to the current baseline version of the MDD.
- Entry # 11 - Pre-operational code units, external interfaces, ancillary data, and system test data have been integrated into a product processing system in the development test environment. The SRR reviewers have access to the product processing system.
- Entry # 12 – A Verification and Validation Plan (VVP) has been written. The SRR reviewers have access to the current baseline version of the VVP.
- Entry # 13 - A System Test Plan (STP) has been written. The SRR reviewers have access to the current baseline version of the STP.
- Entry # 14 - A Verification and Validation Report (VVR) has been written. The SRR reviewers have access to the current baseline version of the VVR.

- Entry # 15 - A System Readiness Document (SRD) has been written. The SRR reviewers have access to the current baseline version of the SRD.
- Entry # 16 - A Project Baseline Report (PBR) has been written. The SRR reviewers have access to the current baseline version of the PBR.

Standard SRR exit criteria:

- Exit # 1 - CTR "Conditional Pass" items have been satisfactorily disposed of.
- Exit # 2 - CTR "Defer" items have been satisfactorily disposed of.
- Exit # 3 - The project plan and DPP are satisfactory
- Exit # 4 - The requirements allocation and RAD are satisfactory.
- Exit # 5 - The algorithm and ATBD are satisfactory.
- Exit # 6 - The design documents (SWA and DDDs) are satisfactory.
- Exit # 7 - The metadata and MDD are satisfactory.
- Exit # 8 - The delivery procedures, tools, training, support services, and documentation available to the users are satisfactory.
- Exit # 9 - System test results and VVR are satisfactory.
- Exit # 10 - The project baseline and PBR are satisfactory.
- Exit # 11 - The SRRR documents updated status of project risks and actions. The risk status is acceptable.
- Exit # 12 - The integrated product processing system is ready for delivery to operations.

Refer to PRG-11.1 for a more detailed description of the SRR. The standard SRR entry criteria, exit criteria, and check list is documented in the process asset CL-11.1 (c.f. Section 2).

SRR objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the SRR.

3.9. Gate 5 Review

Gate 5 is the final review of the project status readiness before it is transitioned to operations, under the joint direction of STAR and SPSRB. Its purpose is to determine whether operations is ready to receive the pre-operational system from the developers. If a project passes Gate 5, the pre-operational system and all associated artifacts are delivered to operations.

Standard Gate 5 Review objectives:

- Review the implementation of the Integrated Master Plan (IMP) and Integrated Master Schedule (IMS)
- Review the technical status and risks of the project
- Review the cost status and risks of the project
- Review the schedule status and risks of the project
- Determine whether corrective actions are needed to allow the project to proceed to operations as planned.
- Determine whether a re-plan and a delta Gate 5 Review are needed.

Standard Gate 5 Review entry criteria:

- Entry # 1 - A System Readiness Review Report (SRRR) has been written. The Gate 5 reviewers have access to the current baseline version of the SRRR.
- Entry # 2 - A Development Project Report (DPR) has been written. The Gate 5 reviewers have access to the current baseline version of the DPR.
- Entry # 3 - A Project Status Report (PSR) has been written. The Gate 5 reviewers have access to the current baseline version of the PSR.

- Entry # 4 - A Gate 5 Document (G5D) has been written. The Gate 5 reviewers have access to the current baseline version of the G5D.
- Entry # 5 - A Project Baseline Report (PBR) has been written. The Gate 5 reviewers have access to the current baseline version of the PBR.

Standard Gate 5 Review exit criteria:

- Exit # 1 – SRR status and SRRR are satisfactory
- Exit # 2 – DPR is satisfactory.
- Exit # 3 - Project status and PSR are satisfactory.
- Exit # 4 - Project baseline and PBR are satisfactory.
- Exit # 5 - Project risks are acceptable.
- Exit # 6 - Status of risk mitigation actions is acceptable
- Exit # 7 - Project is ready for delivery to operations

Refer to PRG-11.2 for a more detailed description of the Gate 5 Review. The standard Gate 5 Review entry criteria, exit criteria, and check list is documented in the process asset CL-11.2 (c.f. Section 2).

Gate 5 Review objectives, entry criteria, exit criteria, and check list may be tailored. Tailoring guidelines are provided in the process asset PG-2 (c.f. Section 2). Refer to the Development Project Plan (DPP) Section 5 to determine whether there has been any project-specific tailoring for the Gate 5 Review.

4. PROJECT ARTIFACTS

Project Artifacts are a set of items that must be produced by the appropriate stakeholders during the product life cycle to support the reviews. They are established and maintained under Configuration Management (CM) by an Enterprise Process Group (EPG) under the direction of a Steering Committee.

The project artifacts are maintained in a project artifact repository. This is a complete set of configuration-managed artifacts developed by each project in accordance with STAR standards. When a project artifact has been approved at a Technical Review or Gate Review, it is placed in the project artifact repository under CM.

Responsibility for producing project artifacts is assigned to stakeholders during the Plan phase, and may be tailored from the standard assignment. The project artifacts that are usually the responsibility of **Development Testers** are listed in Table 4.1.

TABLE 4.1 – Development Programmer Artifacts

Artifact	Type
Software Architecture Document	Document
Development Project Plan	Document
Project Status Report	Report
Gate 3 Document	Document
Requirements Allocation Document	Document
Verification and Validation Plan	Document
Project Requirements Document	Document
Preliminary Design Document	Document
Detailed Design Document	Document
Critical Design Document	Document
Gate 4 Document	Document
Pre-Operational Code	Code
Unit Test Plan	Document
Test Readiness Document	Document
Refined Pre-Operational Code	Code

Unit Test Report	Document
System Test Plan	Document
Code Test Document	Document
Integrated Pre-Operational Code	Code
Verification and Validation Report	Document
System Readiness Document	Document
Gate 5 Document	Report
Development Project Report	Report

Software Architecture Document: The Software Architecture Document (SWA) complements the ATBD by providing the software architecture for the processing code that will implement the algorithm. Refer to DG-1.2 for detailed SWA guidelines.

Development Project Plan: The Development Project Plan (DPP) documents the plan for the development, testing, review, and transition to operations for the project, including stakeholders, tasks, Work Breakdown Structure (WBS), schedule and resources. Refer to DG-5.1 for detailed DPP guidelines.

Project Status Report: The Project Status Report (PSR) is used to manage and control the execution of the project. It complements the DPP by noting the current status of the project tasks, work products, cost, and schedule. Refer to DG-5.2 for detailed PSR guidelines.

Project Status Report Appendix: The PSR Appendix is a Microsoft Excel workbook that provides the current status of project risks and risk mitigation actions. Refer to DG-5.2.A for detailed PSR Appendix guidelines.

Gate 3 Document: The Gate 3 Document (G3D) consists of the presentation slides for the Gate 3 Review. Refer to DG-5.3 and DG-5.3.A for detailed G3D guidelines.

Requirements Allocation Document: The Requirements Allocation Document (RAD) contains the basic and derived requirements for the work products and the allocation of the requirements to system components and product components. Refer to DG-6.2 for detailed RAD guidelines.

Verification and Validation Plan: The Verification and Validation Plan (VVP) describes the work products to be verified and validated, the requirements for each selected work product and the verification and validation methods for each selected work product. Refer to DG-6.3 for detailed VVP guidelines.

Project Requirements Document: The Project Requirements Document (PRD) consists of the presentation slides for the Project Requirements Review (PRR). Refer to DG-6.4 and DG-6.4.A for detailed PRD guidelines.

Preliminary Design Document: The Preliminary Design Document (PDD) consists of the presentation slides for the Preliminary Design Review (PDR). Refer to DG-7.1 and DG-7.1.A for detailed PDD guidelines.

Detailed Design Document: The purpose of the Detailed Design Document (DDD) is to describe the product design at a level of detail that is sufficient for the development programmers to write fully functional pre-operational code. A separate DDD is produced for each software unit that is part of the product processing system. The software units are the Layer-2 elements that are defined in the system layer product software architecture, as described in the SWA. Refer to DG-8.1 for detailed DDD guidelines.

Critical Design Document: The Critical Design Document (CDD) consists of the presentation slides for the Critical Design Review (CDR). Refer to DG-8.2 and DG-8.2.A for detailed CDD guidelines.

Gate 4 Document: The Gate 4 Document (G4D) consists of the presentation slides for the Gate 4 Review. Refer to DG-8.4 and DG-8.4.A for detailed G4D guidelines.

Pre-Operational Code: The Pre-Operational Code (PCOD v1.x) consists of all software components of the detailed design that was approved at the CDR, ready for unit testing. It is expected that SPSRB coding standards will be applied to the pre-operational code. Currently, coding standards exist for FORTRAN, C, and C++ code, and general programming standards exist for all code. These standards are found on the SPSRB web site at http://projects.osd.noaa.gov/spsrb/standards_prog.htm. This requirement may be waived if the circumstances of a specific project provide a compelling reason for a waiver. Waivers should be agreed to as early as possible, included in the project plan, and accepted by operations prior to unit testing.

Unit Test Plan: The Unit Test Plan (UTP) contains the test plan for each software unit in the project's product processing system. The UTP, a complement to the project's Verification and Validation Plan (VVP), focuses on the specifics of the software units and the testing of their functionality and performance. The UTP should document the purpose and function of each unit, its traceability to the project requirements, unit data flows, unit components and unit functions to be tested, a test data description, planned test methods and test sequences and identified test risks. Refer to DG-9.1 for detailed UTP guidelines.

Test Readiness Document: The Test Readiness Document (TRD) consists of the presentation slides for the Test Readiness Review (TRR). Refer to DG-9.2 and DG-9.2.A for detailed TRD guidelines.

Refined Pre-Operational Code: The Refined Pre-Operational Code (PCOD v2.x) consists of all software components of the detailed design that was approved at the CDR (step 8) and unit tested in step 10. The code is a post-unit test refinement of the Pre-Operational Code, refined to correct bugs and other deficiencies that are revealed by unit testing. It is expected that SPSRB coding standards will be applied to the pre-operational code. Currently, coding standards exist for Fortran, C, and C++ code, and general programming standards exist for all code. These standards are found on the SPSRB web site at http://projects.osd.noaa.gov/spsrb/standards_prog.htm. This requirement may be waived if the circumstances of a specific project provide a compelling reason for a waiver. Waivers should be agreed to as early as possible, included in the project plan, and accepted by operations prior to unit testing.

Unit Test Report: The Unit Test Report (UTR) documents the results of testing of each software unit to verify that the requirements allocated to the unit's software components are satisfied. The UTR should describe the results of each unit test in a way that demonstrates the verification of the requirements allocated to components of the software unit, show how the results demonstrate that the requirements allocated to the software units are satisfied, and note any requirements allocations whose verification is incomplete or questionable. Refer to DG-10.1 for detailed UTR guidelines.

System Test Plan: The System Test Plan (STP) contains the plan for testing to ensure that the requirements specified for the product processing system are satisfied by the completed system (Verification) and that the final developed system will satisfy the users' needs and expectations (Validation). The purpose of the system test is to demonstrate, using verification and validation methods, system readiness for operations. Refer to DG-10.2 for detailed STP guidelines.

Code Test Document: The Code Test Document (CTD) consists of the presentation slides for the Code Test Review (CTR). Refer to DG-10.3 and DG-10.3.A for detailed CTD guidelines.

Integrated Pre-Operational Code: The Integrated Pre-Operational Code (PCOD v3.x) consists of all software components of the detailed design that was approved at the CDR (step 8), unit tested in step 10, integrated into an end-to-end pre-operational product processing system, and system tested.

Verification and Validation Report: The Verification and Validation Report (VVR) documents the results of unit testing and system testing to ensure that the requirements specified for the product processing system are satisfied by the completed system (Verification) and that the final developed system will satisfy the users' needs and expectations (Validation). Refer to DG-11.4 for detailed VVR guidelines.

System Readiness Document: The System Readiness Document (SRD) consists of the presentation slides for the System Readiness Review (SRR). Refer to DG-11.5 and DG-11.5.A for detailed SRD guidelines.

Gate 5 Document: The Gate 5 Document (G5D) consists of the presentation slides for the Gate 5 Review. Refer to DG-11.7 and DG-11.7.A for detailed G5D guidelines.

Development Project Report: The Development Project Report (DPR) provides the development team's assessment of their experience in implementing the project, including lessons learned and recommendations for process improvement. Refer to DG-11.9 for detailed DPR guidelines.

5. TASK DESCRIPTION

Development Scientists participate in the following process steps:

- Step 5 - Project Plan (TG-5)
- Step 6 - Project Requirements (TG-6)
- Step 7 - Preliminary Design (TG-7)
- Step 8 - Detailed Design (TG-8)
- Step 9 - Code & Test Data Development (TG-9)
- Step 10 - Code Test And Refinement (TG-10)
- Step 11 - System Integration and Test (TG-11)

The standard **Development Programmer** tasks for each of these steps are described below. **Development Programmers** may also refer to the relevant TGs for a complementary task description.

5.1 Project Plan Tasks

Figure 5.1 shows the process flow for step 5.

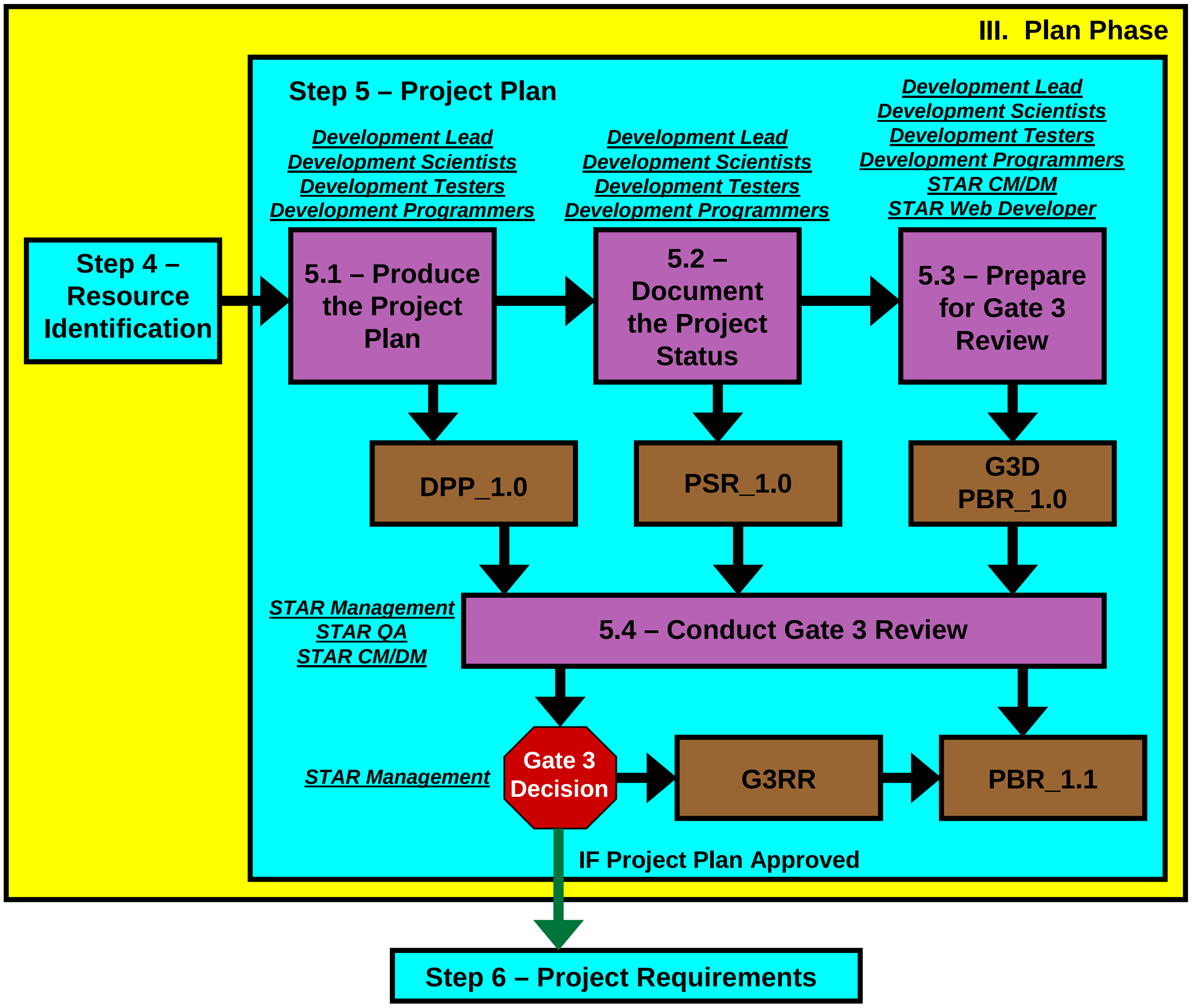

Figure 5.1 – Step 5 Process Flow

5.1.1 Expected BEGIN State

- REQUIRED: A project proposal (PP) that includes a User Request has been reviewed at a Gate 2 Review
- REQUIRED: The project has been approved for development by SPSRB and STAR.

- REQUIRED: A STAR Division and Branch has been selected to implement Development, and a Development Lead has been identified.
- REQUIRED: Required and available resources (hardware, software, personnel, and training) have been identified.
- REQUIRED: An SPSRB Project Plan that identifies these resources has been written.
- EXPECTED: The research algorithm has been matured and documented in an Algorithm Theoretical Basis Document (ATBD).
- EXPECTED: A software architecture has been matured and documented in a Software Architecture Document (SWA).
- EXPECTED: Research and Development (R&D) code has been written.
- EXPECTED: R&D code has been run with research test data to produce proxy data products.
- EXPECTED: R&D code test results are documented in the ATBD.

5.1.2 Task Inputs

- Algorithm Theoretical Basis Document
- Software Architecture Document
- R&D Code
- R&D Test Data
- Project Proposal
- Gate 2 Review Report
- SPSRB Project Plan

5.1.3 Desired END State

- Project objectives and concept of operations have been derived from user/customer needs and expectations
- Project stakeholders have been identified

- The project's process has been defined, by tailoring the STAR EPL set of standard processes. The defined process includes the project lifecycle steps, project reviews, review artifacts, work products, and Baseline Builds (BB).
- The planned work has been organized into an Integrated Master Plan (IMP) and Integrated Master Schedule (IMS).
- Expected project costs and cost schedule have been identified
- Project risks have been identified and assessed
- Risk mitigation actions have been identified
- The initial version of the DPP has been written
- Project status has been documented in the initial version of the PSR
- Risks and actions have been documented in an Appendix to the PSR
- A Gate 3 Review of the project plan and project status has been conducted
- A Gate 3 Review Report (G3RR) has been written, approving the project for the Design phase.
- Baseline Build 1.1 has placed the required items in the project artifact repository
- PBR_1.1 documents the status of the BB 1.1 project baseline

5.1.4 Task Outputs

Task outputs consist of the following BB 1.1 items:

- Development Project Plan (DPP)
- Project Status Report (PSR)
- Project Risks and Actions (PSR Appendix)
- Gate 3 Document (G3D)
- Gate 3 Review Report (G3RR)
- Project Baseline Report (PBR)

5.1.5 Stakeholder Activities

Step 5 activities include:

1) Produce the project plan
2) Document the project status
3) Prepare for Gate 3 Review
4) Conduct Gate 3 Review

5.1.5.1 Produce Project Plan

Development Programmers assist the **Development Lead** in the preparation of a DPP. The DPP is a required artifact for the Gate 3 Review. The process flow for producing the project plan is shown in Figure 5.2.

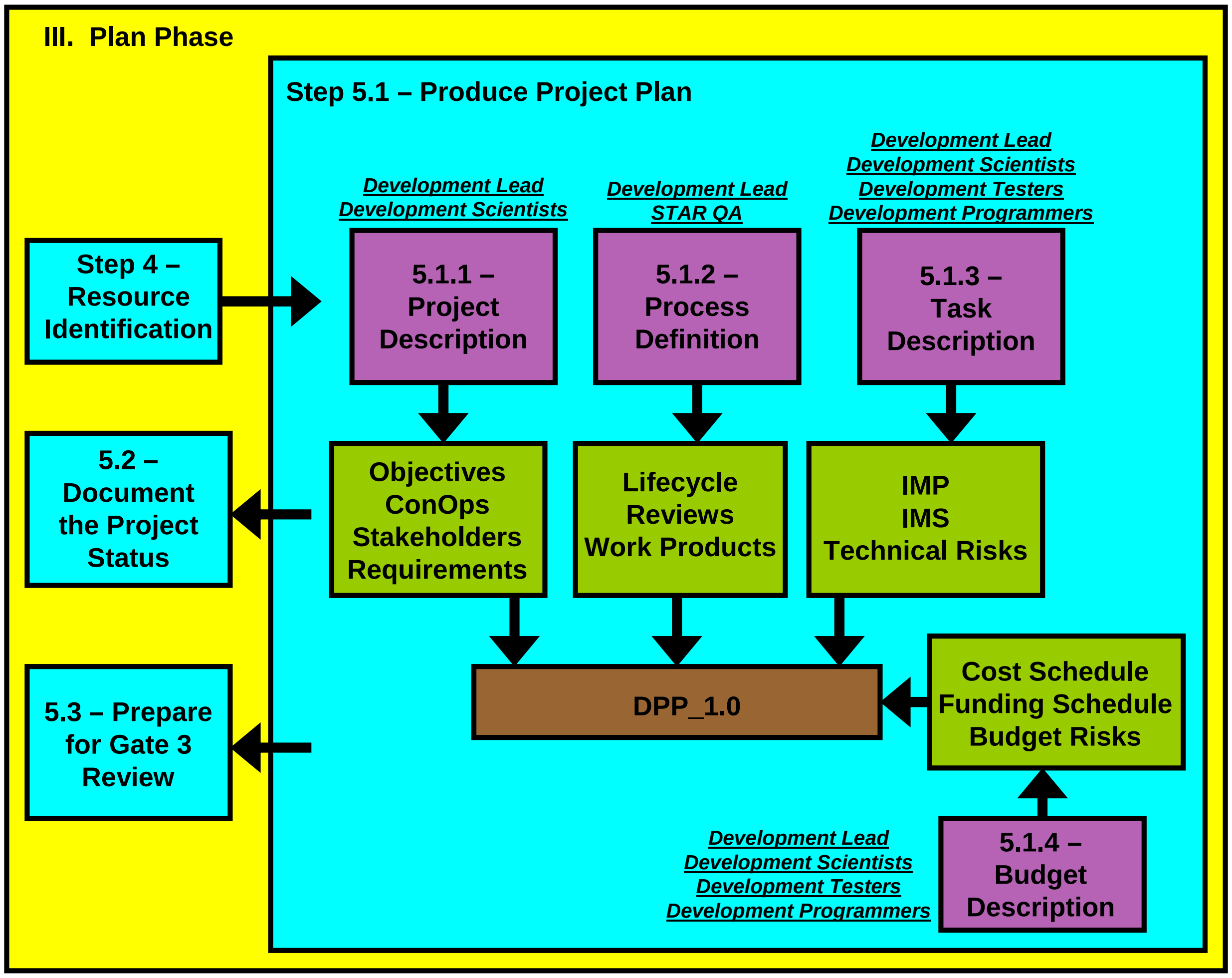

Figure 5.2 – "Produce Project Plan" Process Flow

Step 5.1 activities include:

1) Project description
2) Process definition
3) Task description
4) Budget description

The **Development Lead** provides the project description, with assistance from **Development Scientists**.

The project's process is defined by tailoring the STAR EPL set of standard processes. The **Development Lead** provides the description of the tasks that have been identified to accomplish the defined process, with assistance from **Development Scientists, Development Testers,** and **Development Programmers**.

The task description begins with the listing of tasks that are identifiable from the point of view of customers and end users. High-level tasks from the user's point of view are typically oriented to creation, delivery and maintenance of products. They can also include validation of products. These should be obtainable from the PP and from customer requirements documents. These tasks are called the "work tasks", as they are typically the kind of tasks that are stated in a Statement of Work (SOW).

The work tasks should be translated into "major tasks". These are the highest-level tasks that will be included in the Integrated Master Plan (IMP) and entered into a Microsoft Project file. Examples of high level tasks: "Develop requirements", "Develop interfaces", "Develop software units".

Once the major tasks have been identified, they can be organized into the IMP. The IMP provides a detailed roadmap for meeting project requirements. For each major task:

- Identify the project objective, project requirement and/or SOW item that is satisfied, completely or partially, by the accomplishment of the task.
- Identify the stakeholders that are affected by the activity and those who have expertise that is needed to conduct the activity
- Identify predecessor tasks and successor tasks.
- Identify the criteria for initiating the task. Typically, this will consist of satisfying the accomplishment criteria for predecessor tasks.

- Identify subtasks, to the extent possible.
- Identify the task's work products
- Identify the criteria for task accomplishment
- Identify the review or reviews at which the task's accomplishment's will be verified

Create an Integrated Master Schedule (IMS) by mapping the IMP to a calendar-based schedule, based on the estimate of effort and available resources for each task and its sub-tasks. The IMS is used to track day-to-day progress and includes the continual assessment of the technical parameters required to support each IMP task/event.

The IMS should be in a separate file that is an Appendix to the DPP. It is very useful to translate the IMS into a resource-loaded schedule in a Microsoft Project file. This file should include all of the major tasks, subtasks and milestones that were identified in the IMP, with their identified linkages. An alternative to a Project file is a Microsoft Excel file that puts each task in a row on a spreadsheet and includes associated data (predecessor tasks, successor tasks, assigned stakeholders, start data, end date) in distinct columns. An Excel file can be more accessible, but lacks the project control features of a Project file.

Identify potential risks to the successful technical implementation of the tasks. For each identified risk, provide a plan for managing the risk. A detailed assessment of risks and risk mitigation actions will be provided in the Project Status Report (PSR) Appendix (c.f. Section 6.4.2).

The DPP Document Guidelines (DG-5.1) are strongly recommended to the DPP writers. DG-5.1 provides the standard DPP Table of Contents and guidelines for each standard DPP section.

In addition to DG-5.1, the STAR PAR should include examples of DPPs from other projects. These will be very helpful to DPP writers who have not previously written a DPP.

DPP writers should also use the SPSRB Project Plan and the User request as resources for the DPP. These artifacts typically include information that can be adopted for the DPP.

5.1.5.2 Document Project Status

Development Testers assist the **Development Lead** in the preparation of a PSR in accordance with PSR guidelines DG-5.2 and DG-5.2.A. The PSR is a required artifact for the Gate 3 Review.

Project status includes:

1) Stakeholder Involvement
2) Task Progress
3) Schedule
4) Budget
5) Risks

The PSR should report the status of task progress for each task that is documented in the DPP. The **Development Lead** should request task status from the **Development Programmers** on a regular basis. If task progress is falling behind schedule, determine what factors are responsible (e.g., unexpected technical difficulty, delays in predecessor tasks, non-involvement of stakeholders, training gaps).

The PSR includes an Appendix that reports the current status of project risks and associated risk mitigation actions. **Development Programmers** should assist the **Development Lead** in documenting this status. Risk status includes the identification of risks, quantitative risk assessment, identification of actions to mitigate the risks, action closure criteria, assignment of responsibility for closing the action, and an action closure plan.

The PSR guidelines (DG-5.2 and DG-5.2.A) are strongly recommended to the PSR writers. DG-5.2 provides the standard PSR Table of Contents and guidelines for each standard PSR section. DG-5.2.A provides the guidelines for the PSR Appendix, which is a Microsoft Excel workbook that documents the status of project risks and risk mitigation actions.

In addition to DG-5.2 and DG-5.2.A, the STAR PAR should include examples of PSRs from other projects. These will be very helpful to PSR writers who have not previously written a PSR. Note that the final project PSR will reflect status at the completion of a project, when most issues have been resolved, most risks have been closed, and most actions have been completed. Examine the STAR PAR for examples of PSR versions that were produced for a Gate 3 Review. These are more indicative of what a PSR should look like at this stage of the project lifecycle.

PSR writers should also use the G2RR as a resource for the PSR Appendix. The G2RR typically documents risks and actions that are identified at the Gate 2 Review. The risks and actions identified in the PSR Appendix should be built from these.

5.1.5.3 Prepare For Gate 3 Review

Development Programmers assist in the preparation of the Gate 3 Review presentation. The presentation slide package is the Gate 3 Document (G3D). The G3D is prepared in accordance with G3D guidelines DG-5.3. DG-5.3.A provides G3D slide templates that can be adapted for the project's G3D. The G3D developers should examine the DPP to determine whether the Gate 3 Review objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the G3D slide templates must be adapted to accommodate the tailoring.

5.1.5.4 Conduct Gate 3 Review

The "Project Plan" step culminates with a Gate 3 Review.

The Gate 3 Review consists of the presentation of the project plan and project status by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**STAR Management**).

Each stakeholder who performed activities during step 5 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved. At the conclusion of Development (step 11), the **Development Lead** will collect the final edited personal stakeholder records and incorporate them into a Development Project Report (DPR).

5.2 Requirement Development Process

Requirements development is an iterative process that occurs throughout the Design phase of the product lifecycle. This phase includes three steps that produce a detailed requirements allocation through an iterative (spiral) development of requirements, solutions, and design:

- Project Requirements (step 6 of the STAR EPL)
- Preliminary Design (step 7 of the STAR EPL)
- Detailed Design (step 8 of the STAR EPL)

Figure 5.3 illustrates the Requirements Development process, with step 6 highlighted.

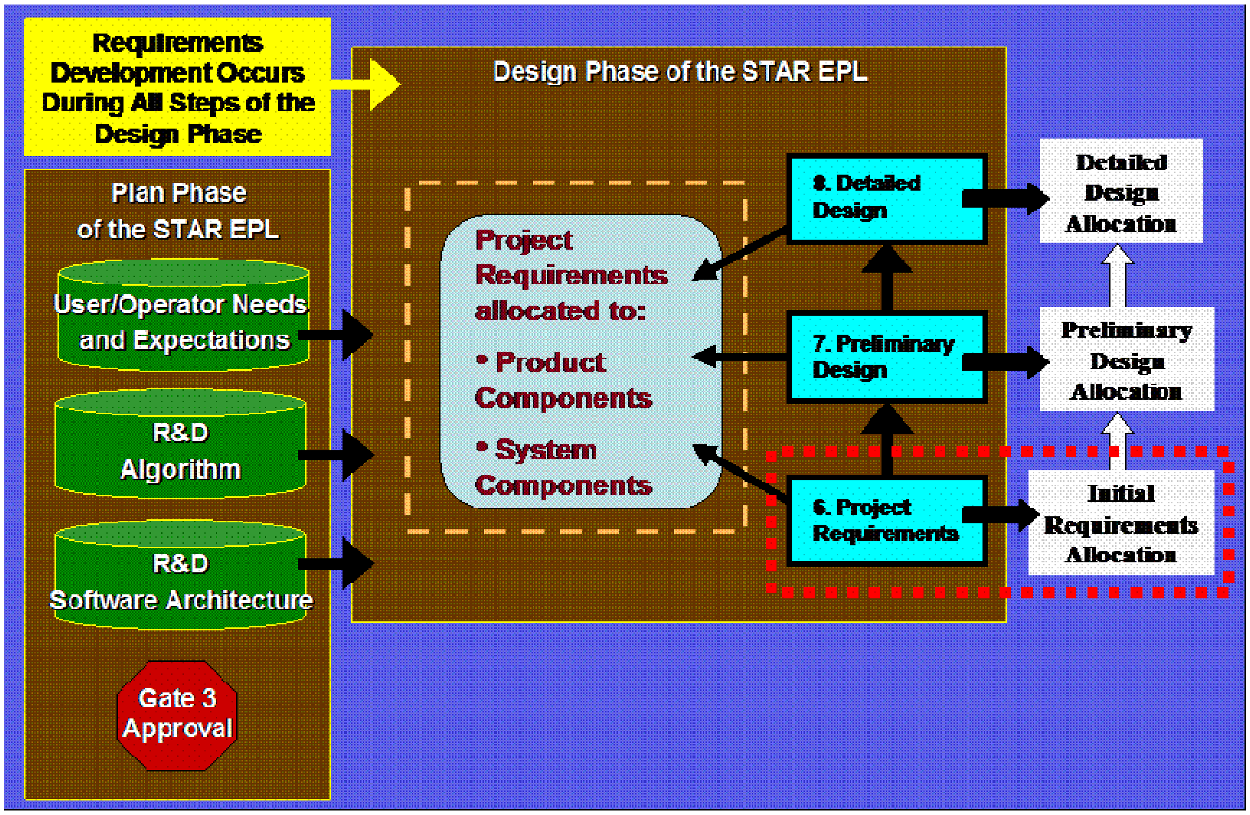
Requirements Development Occurs During All Steps of the Design Phase
Design Phase of the STAR EPL
Plan Phase of the STAR EPL
User/Operator Needs and Expectations
R&D Algorithm
R&D Software Architecture
Gate 3 Approval
Project Requirements allocated to:
• Product Components
• System Components
8. Detailed Design
7. Preliminary Design
6. Project Requirements
Detailed Design Allocation
Preliminary Design Allocation
Initial Requirements Allocation

Figure 5.3 – Requirements Development Process

As Figure 5.3 shows, the objective of step 6 is to produce an initial requirements allocation that consists of requirements derived from user/operator needs and expectations and the allocation of these requirements to product components and system components that have been identified in the Research and Development (R&D) algorithm and software architecture.

Note that steps 7 and 8 continue the requirements development process. This is because the requirements development process produces the requirements statements ***and*** their allocation to product components and system components of a design that is matured to an increasing amount of detail and completeness throughout the Design phase.

The process of producing an increasingly mature and complete requirements allocation involves an iterative development of the requirements, solution, design, and requirements allocation. Figure 5.4 illustrates this.

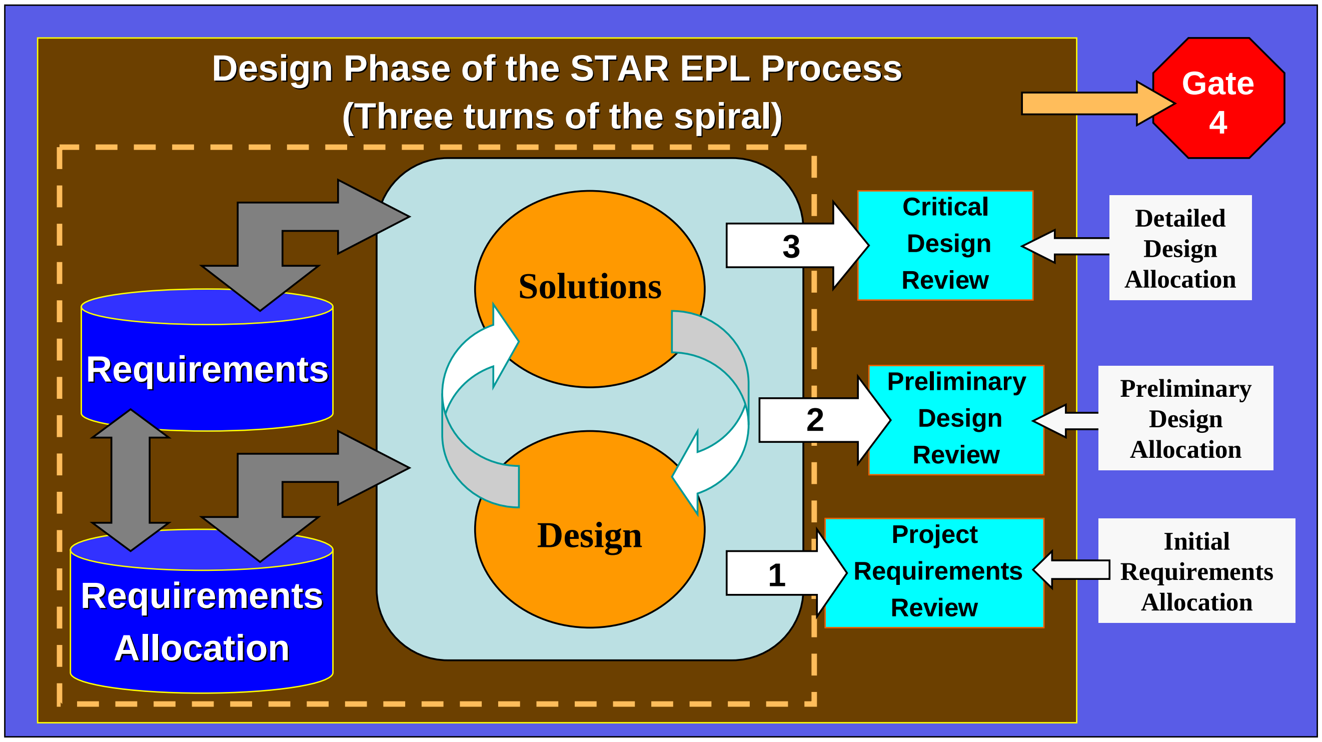
Design Phase of the STAR EPL Process
(Three turns of the spiral)
Gate 4
Requirements
Requirements Allocation
Solutions
Design
3
Critical Design Review
Detailed Design Allocation
2
Preliminary Design Review
Preliminary Design Allocation
1
Project Requirements Review
Initial Requirements Allocation

Figure 5.4 – Iterative (Spiral) Development of Requirements Allocation

As shown in Figure 5.4, requirements drive solutions, solutions drive design, and design determines requirements allocation. Gaps and/or inconsistencies between the requirements and the requirements allocation will then drive revisions to solutions and design. Revised solutions and design then drive revisions to requirements and/or requirements allocation, etc.

As the project matures throughout the Design phase, an increasingly comprehensive and mature requirements allocation is reviewed at each of the three technical reviews of this phase (PRR, Preliminary Design Review (PDR), and Critical Design Review (CDR)).

This process is continuous and iterative, but is also characterized by three distinct milestones:

1) The Initial Requirements Allocation is achieved when it is determined that the set of stated requirements is complete. That is, it is not expected that additional maturation will result in additional requirements. At that point, a PRR is conducted to complete step 6.

2) The Preliminary Design Allocation is achieved when it is determined that a preferred solution has been identified to meet the set of requirements that were approved at the PRR. That is, it is not expected that additional maturation will result in a different solution. At that point, a PDR is conducted to complete step 7. This does not preclude the possibility that the set of requirements will be revised during step 7, as a result of issues discovered during the preliminary design development.

3) The Detailed Design Allocation is achieved when it is determined that a complete design has been developed to implement the preferred solution that was approved at the PDR. At that point, a CDR is conducted to complete step 8. This does not preclude the possibility that the set of requirements will be revised during step 8, as a result of issues discovered during the detailed design development..

The iterative nature of this development means that requirements are not expected to be finalized until the complete convergence of requirements, solution, and design is finalized at the end of step 8, resulting in the detailed design allocation. Once this is accomplished, the project is ready to proceed to a Gate 4 Review and the Build phase.

5.3 Project Requirements Tasks

Figure 5.5 shows the process flow for step 6.

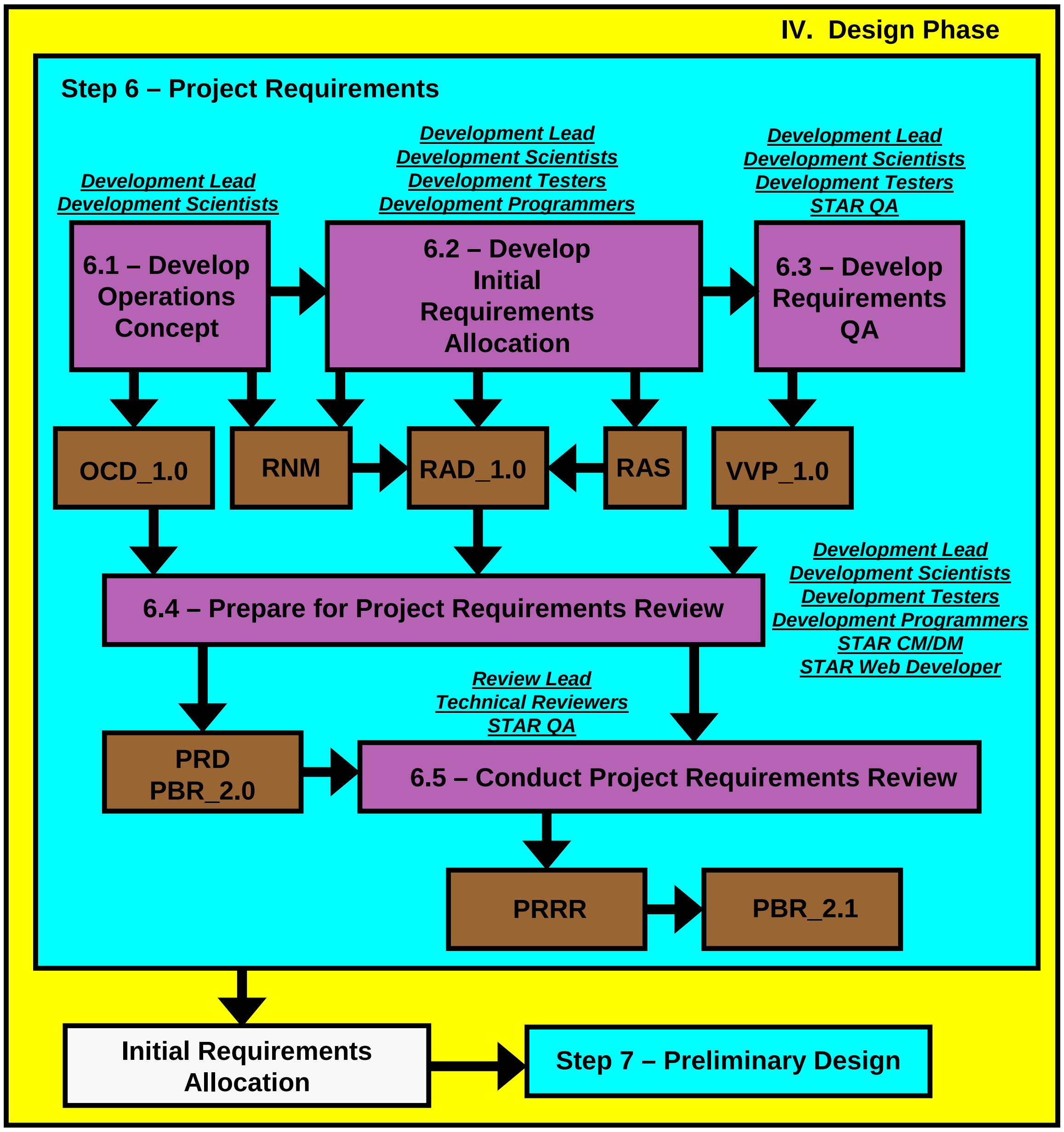

Figure 5.5 – Step 6 Process Flow

5.3.1 Expected BEGIN State

- REQUIRED: A Gate 3 Review of the DPP and PSR has been conducted
- REQUIRED: Baseline Build (BB) 1.1 has placed the following items in the project artifact repository:
    - DPP, including Appendices
    - PSR, including Appendix
    - Gate 3 Document (G3D)
    - Gate 3 Review Report (G3RR)
- EXPECTED: BB 1.1 has placed the following items in the project artifact repository:
    - R&D code
    - R&D test data
    - Algorithm Theoretical Basis Document (ATBD)
    - Software Architecture Document (SWA)
    - PP
    - Gate 2 Review Report (G2RR)
- REQUIRED: PBR_1.1 documents the status of the BB 1.1 project baseline
- REQUIRED: Gate 3 Reviewers have approved the project to proceed to the Design phase.

5.3.2 Task Inputs

Task inputs consist of the following BB 1.1 items:

- PP, including User Request
- DPP_1.0
- PSR_1.0
- Project Risks and Actions (PSR_1.0 Appendix)
- G3RR
- Project Baseline Report (PBR_1.1)

5.3.3 Desired END State

- An operations concept, developed from user/customer needs and expectations, explains what products are to be produced, why they are being produced, and how they will be produced in an operational environment,
- Basic project requirements have been developed from the operations concept
- Requirements have been analyzed in light of the customer's needs, mission objectives, system constraints, and design constraints to develop more specific product, system, and process requirements for the system.
- Derived project requirements have been developed from analysis of the basic requirements and other derived requirements
- An initial allocation of the requirements identifies product and system components and traces each component to one or more requirement so that a system architecture that will meet all project requirements can be designed.
- A plan has been developed for monitoring the status of the requirements and their allocation to ensure that the integrity of the requirements allocation is preserved as the solutions, design and implementation matures through the Design and Build phases.
- A plan has been developed to verify the identified work products, validate the identified requirements, and validate the identified products.
- The project plan has been updated as necessary
- The status of project risks and actions has been updated
- A PRR of the project plan, operations concept, requirements, and requirements allocation has been conducted
- A PRRR has been written
- Baseline Build 2.1 has placed the required items in the project artifact repository
- PBR_2.1 documents the status of the BB 2.1 project baseline

5.3.4 Task Outputs

Task outputs consist of the following BB 2.1 items:

- DPP_1.x

- OCD_1.0
- RAD_1.0, including Requirements/Needs Matrix (RNM) and Requirements Allocation Sheet (RAS)
- VVP_1.0
- Project Risks and Actions (PSR_1.x Appendix)
- PRD
- PRRR
- PBR_2.1

5.3.5 Stakeholder Activities

Step 6 activities include:

1) Develop operations concept
2) Develop initial requirements allocation
3) Develop requirements QA
4) Prepare for PRR
5) Conduct PRR

5.3.5.1 Develop Operations Concept

Development Scientists assist the **Development Lead** in the development of the operations concept**.**

5.3.5.2 Develop Initial Requirements Allocation

Development Programmers assist the **Development Lead** in the development of the initial requirements allocation. Figure 5.6 shows the process flow for developing the initial requirements allocation.

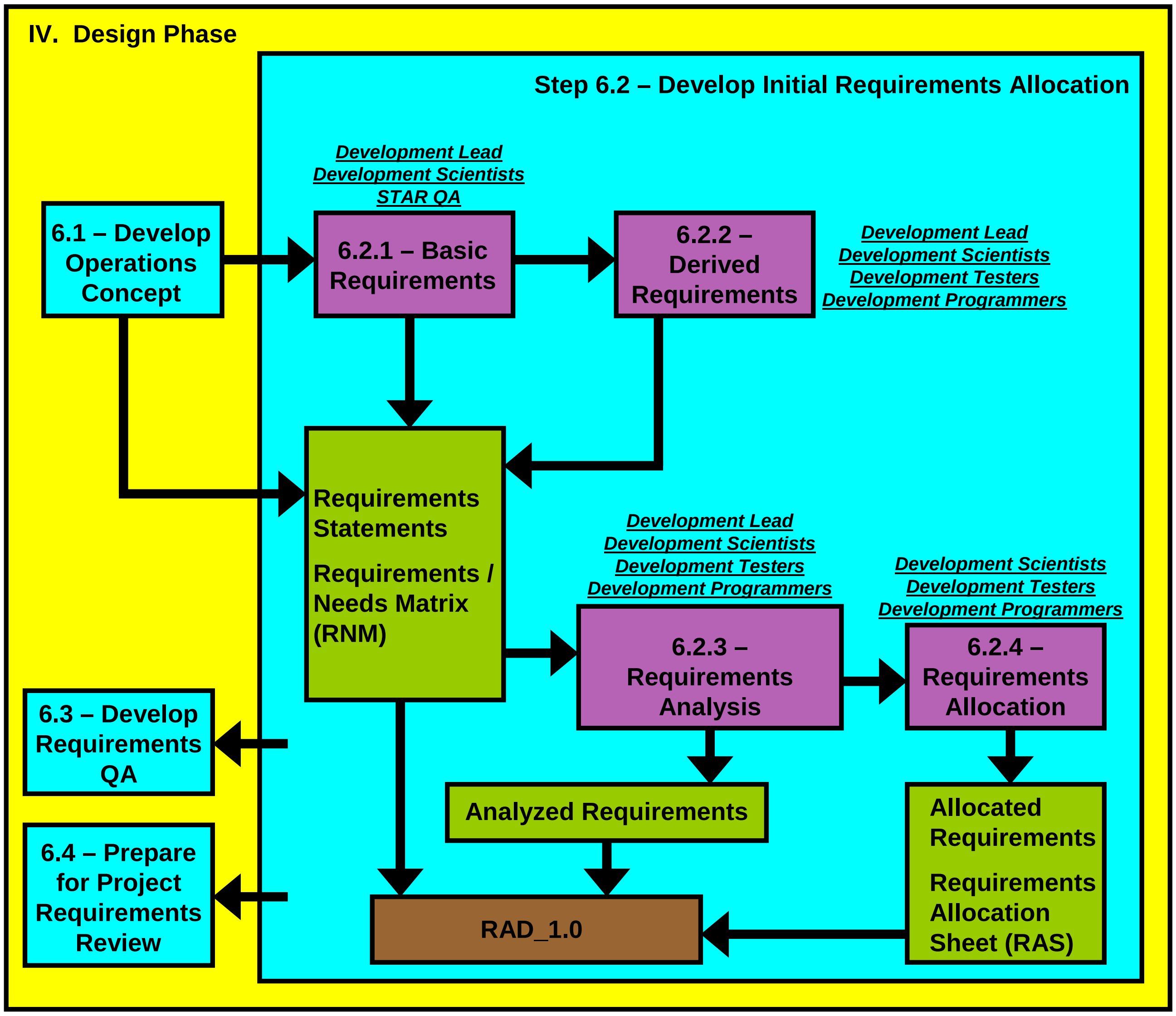

Figure 5.6 – "Develop Initial Requirements Allocation" Process Flow

Step 6.2 activities include:

1) Basic requirements
2) Derived requirements
3) Requirements analysis

4) Requirements allocation

Development Programmers assist with the identification of the project's derived requirements. Derived requirements are those requirements that are not directly traceable to a customer/user need or expectation, or a NESDIS mission goal, but instead are directly traceable to a basic requirement or to another derived requirement.

Figure 5.7 illustrates the relation between basic requirements and derived requirements.

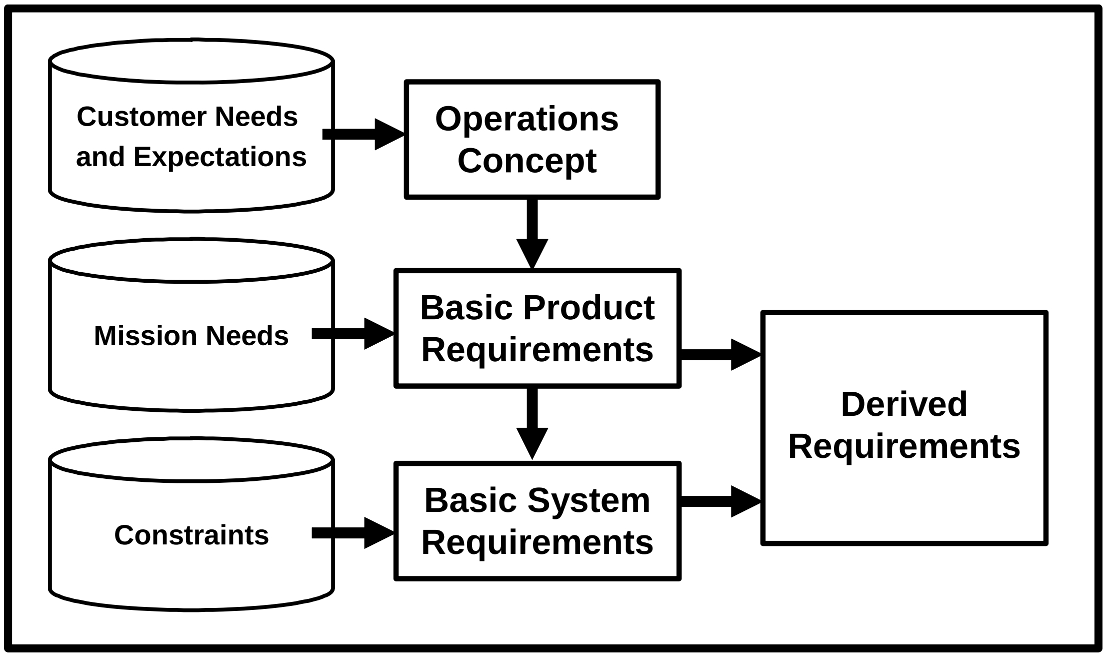
Customer Needs and Expectations
Operations Concept
Mission Needs
Basic Product Requirements
Derived Requirements
Constraints
Basic System Requirements

Figure 5.7 – Basic and Derived Requirements

Derived requirements are typically determined by analysis of basic requirements.

Derived requirements traceable to a basic product requirement are derived product requirements. Derived product requirements address the cost and performance of other life-cycle phases (e.g., production, operations, and disposal) to the extent compatible with business objectives.

Derived requirements traceable to a basic system requirement are derived system requirements.

The structure of derived requirements may include multiple levels. That is, a derived requirement may be directly traceable to a basic requirement or another derived requirement. This trace should be documented in the RAD (c.f. DG-6.2).

The derived requirements statements are documented in RAD (v1r0), following guidelines in DG-6.2.

Development Programmers assist the **Development Lead** in the analysis of the project requirements. Requirements analysis follows the identification of requirements. Requirements identifiers should provide a list of identified requirements to the analysts. Requirements analysts should work from this list and other relevant project artifacts (e.g. DPP, OCD).

Conduct analyses of the requirements with the requirements provider(s) to ensure that a compatible, shared understanding is reached on the meaning of the requirements so the project participants can commit to them.

Requirements analysis includes:

- Acceptance analysis
- Technical analysis
- Quantitative analysis
- Functional analysis

Perform an ***acceptance analysis*** of the requirements, using standard acceptance quality criteria. Requirements should be clearly and properly stated, complete with respect to customer needs and project goals, consistent with the NESDIS strategic and mission plan, internally consistent with each other, uniquely identified, traceable to their sources, and completely traceable to higher level requirements.

Perform a ***technical analysis*** of the requirements to ensure that they are feasible and verifiable. While design determines the feasibility of a particular solution, technical requirements analysis addresses knowing which requirements affect feasibility. Identify key requirements that have a strong influence on cost, schedule, functionality, risk, or performance. Identify technical performance measures that will be tracked during the development effort. Technical analysis requires an in-depth understanding of not just the customer requirements, but also the capabilities and limitations of hardware and software from which the product will be developed.

Requirements technical analysis is closely linked with the development of basic and derived requirements, also known as requirements identification. The links between the two are intended to be iterative, with analysis refining identification, until a satisfactory convergence is reached on a set of requirements that balances customer needs and NESDIS mission needs with constraints, including the capabilities and limitations of hardware and software from which the product will be developed.

Perform a ***quantitative analysis*** of the requirements. Quantitative analysis is a specialized subset of technical analysis that is focused on performance requirements. Performance requirements must be specific and quantitative. Analysis should strike a balance between customer needs and expectations, whether quantitative or qualitative, and anticipated constraints. Consider cost, schedule and technical constraints. Consider the importance of the product performance to the NESDIS strategic and mission plan.

Quantitative analysis of performance requirements may require testing of the performance of solutions, and therefore may need to be extended into the Build phase (steps 9-11) of the STAR EPL. In that case, the versions of the RAD developed during the Design phase (RAD v1r0 and its revisions) should explicitly state that the quantitative analysis of the performance requirements is provisional. This provisional status should be noted as a project risk that will require careful monitoring as coding and testing proceed.

Perform a ***functional analysis*** of the requirements. The purpose of functional analysis is to identify, describe, and relate the functions a system (or subsystem) must perform. The definition of functionality can include actions, sequence, inputs, outputs, or other information that communicates the manner in which a product will be produced and used. This is needed to allow for an effective allocation of requirements to product components and system components.

For PRR, it is expected that functional analysis will result in a decomposition of basic requirements into derived functional requirements in sufficient detail so that preliminary design solutions can be synthesized during the next step of the product lifecycle. Functional requirements describe “what” the system must do independent of the physical or actual implementation. It is important to maintain this independence in order to objectively evaluate alternative solutions during synthesis.

The definition of functions, their logical groupings, and their association with requirements is referred to as a functional architecture. Functional architecture is the hierarchical arrangement of functions, their internal and external (external to the aggregation itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and their design constraints. Functional architecture serves as the bridge between the operations concept and the system architecture of design components that will be developed during the next step of the product lifecycle.

Development Scientists, Development Testers, and **Development Programmers** allocate each project requirement to product components and system components.

Allocate requirements to product components that have been identified in the system architecture, as documented in the latest version of the SWA. Product components are defined as any item that will be integrated to form the end-use product, i.e. these are the deliverable items.

Allocate requirements to system components that have been identified in the system architecture, as documented in the latest version of the SWA. System components are defined as any item that is necessary or useful for building the end-use product, but will not be delivered to customers and/or end users.

The version 1 system architecture is developed prior to requirements development solely for the purpose of supporting research coding and typically will not be mature enough for a complete allocation. For PRR, it is sufficient to identify those product and system components in the architecture that are likely to be retained in the version 2 architecture, and allocate pertinent requirements to those components. As previously noted, requirements allocation will be developed iteratively with design development (version 2 system architecture).

Make a complete requirements allocation for each alternative approach. Establish the requirements associated with the selected set of alternatives as the set of allocated requirements to those product components. Selecting product components that best satisfy the criteria establishes the requirement allocations to product components. Lower level requirements are generated from the selected alternative and used to develop the product-component design. Interface requirements among product components are described, primarily functionally.

There may be cases where a project does not wish to analyze alternative solutions. In that case, the PRR should decide whether a trade study of alternative solutions should be conducted for PDR. If a project wishes to bypass the analysis of alternative solutions, it must provide a convincing rationale (e.g., strong algorithm heritage).

Summarize the requirements allocations in a Requirements Allocation Sheet (RAS). The RAS is a matrix. The rows consist of the requirements, with one row for each requirement. It is recommended that the requirements be listed in numerical order, with derived requirements listed after their basic requirements, as follows:

Requirement 0.0

Requirement 0.1

Requirement 0.1.1

Requirement 0.1.2

Requirement 0.2

Requirement 0.2.1

…….

Requirement 1.0

Requirement 1.1

etc.

The columns consist of the product and system components of the system architecture. It is helpful to number the components. In fact, it is the standard practice to number each component in the system architecture. The component numbers can be obtained from the latest version of the SWA. The requirements developers should consult with the developers of the system architecture to ensure that the correct component numbers are used in the RAS. The RAS should be included as an Appendix document to the RAD. The RAS can be created as a table or imported from a Microsoft Excel spreadsheet to a Microsoft Word Object.

5.3.5.3 Develop Requirements QA

Development Testers assist the **Development Lead** in the development of the requirements QA plan.

5.3.5.4 Prepare For PRR

Development Programmers assist in the preparation of the PRR presentation. The PRR slide package is the PRD. The PRD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with PRD guidelines DG-6.4. DG-6.4.A provides PRD slide templates that can be adapted for the project's PRD. The PRD developers should examine the DPP to determine whether the PRR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the PRD slide templates must be adapted to accommodate the tailoring.

Development Programmers assist the **Development Lead** in updating the status of the project risks and associated risk mitigation actions for inclusion in the PRD and the PSR Appendix. Risk management guidelines can be found in PG-1.

5.3.5.5 Conduct PRR

The "Project Requirements" step culminates with a PRR. The PRR consists of the presentation of the Initial Requirements Allocation by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the PRR to determine whether the PRR artifacts have established the requirements to be satisfied by the project and the means to validate them. Reviewers should be familiar with the PRR guidelines (PRG-6) and check list (CL-6).

Each stakeholder who performed activities during step 6 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

5.4 Preliminary Design Tasks

Figure 5.8 shows the process flow for step 7.

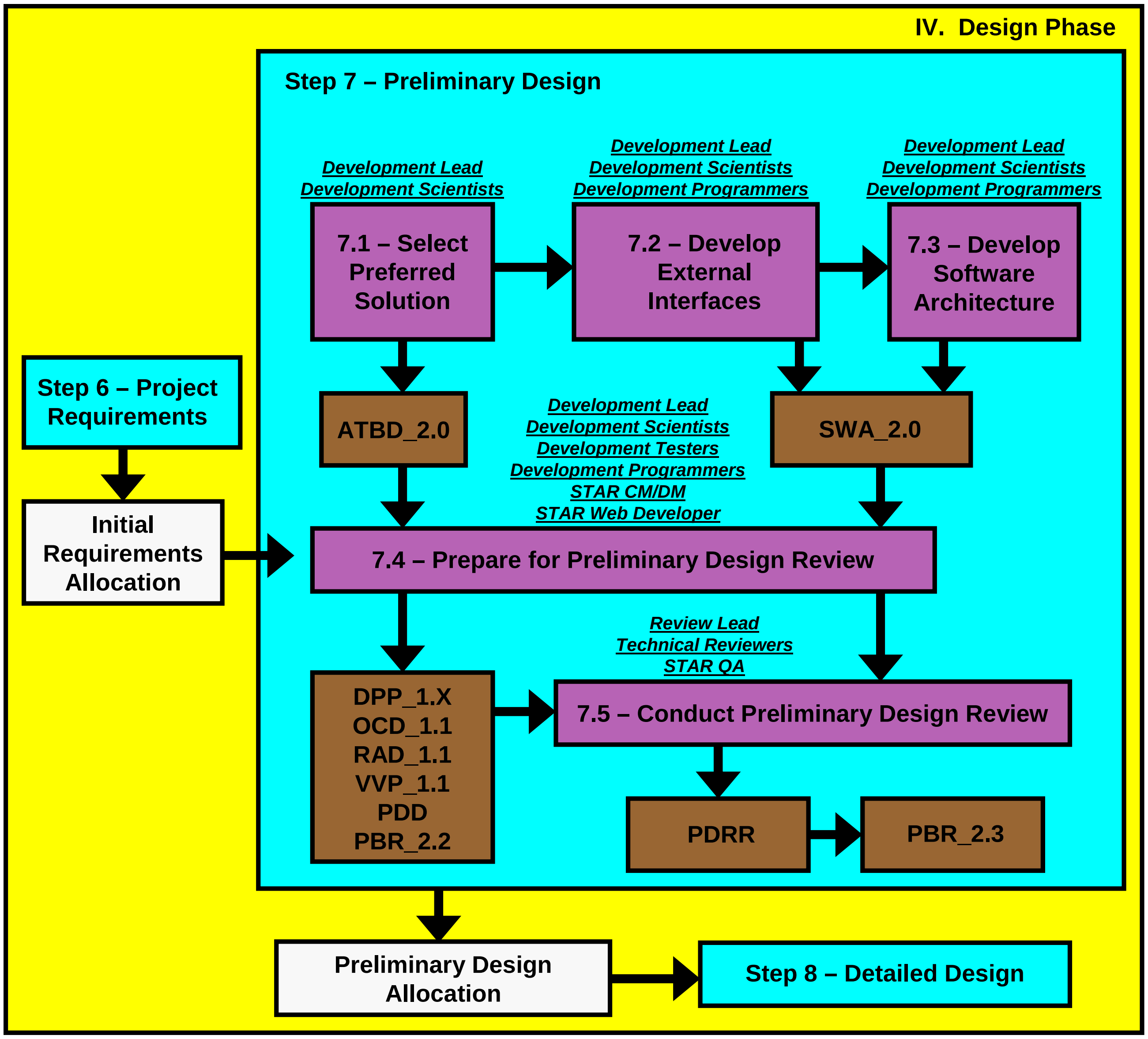

Figure 5.8 – Step 7 Process Flow

5.4.1 Expected BEGIN State

- REQUIRED: A PRR has been conducted
- REQUIRED: An Initial requirements Allocation has been developed and approved

- REQUIRED: Baseline Build (BB) 2.1 has placed the following items in the project artifact repository:
    - DPP, including Appendices
    - OCD
    - RAD, including Appendices
    - VVP
    - Project Requirements Document (PRD)
    - Project Requirements Review Report (PRRR)
- EXPECTED: BB 2.1 has placed the following items in the project artifact repository:
    - R&D code
    - R&D test data
    - ATBD
    - SWA
    - PP
    - Gate 2 Review Report (G2RR)
    - Gate 3 Review Report (G3RR)
- REQUIRED: PBR_2.1 documents the status of the BB 2.1 project baseline
- REQUIRED: PRR reviewers have approved the project to proceed to the Preliminary Design step, and have documented this approval in the PRRR.

5.4.2 Task Inputs

Task inputs consist of the following BB 2.1 items:

- DPP_1.x,
- OCD_1.0
- RAD_1.0, including Requirements/Needs Matrix (RNM) and Requirements Allocation Sheet (RAS)
- VVP_1.0
- ATBD_1.1
- SWA_1.1
- Project Risks and Actions (PSR_1.x Appendix)
- PRD
- PRRR

- PBR_2.1

5.4.3 Desired END State

- An operations concept, developed from user/customer needs and expectations, explains what products are to be produced, why they are being produced, and how they will be produced in an operational environment,
- Basic project requirements have been developed from the operations concept
- Requirements have been analyzed in light of the customer's needs, mission objectives, system constraints, and design constraints to develop more specific product, system, and process requirements for the system.
- Derived project requirements have been developed from analysis of the basic requirements and other derived requirements
- A preferred solution to meet the requirements has been identified and approved.
- A Context-Layer software architecture has been developed.
- A System-Layer software architecture has been developed.
- A preliminary design allocation of the requirements identifies product and system components down to the System-Layer, and traces each component to one or more requirement so that a detailed system architecture that will meet all project requirements can be designed.
- A plan has been developed for monitoring the status of the requirements and their allocation to ensure that the integrity of the requirements allocation is preserved as the detailed design is developed.
- A plan has been developed to verify the identified work products, validate the identified requirements, and validate the identified products.
- The project plan has been updated as necessary
- The status of project risks and actions has been updated
- A PDR of the project plan, operations concept, requirements, software architecture, and requirements allocation has been conducted
- A PDRR has been written
- Baseline Build 2.3 has placed the required items in the project artifact repository
- PBR_2.3 documents the status of the BB 2.3 project baseline

5.4.4 Task Outputs

Task outputs consist of the following BB 2.3 items:

- DPP_1.x
- ATBD_2.0
- SWA_2.0
- OCD_1.1
- RAD_1.1, including RNM and RAS
- VVP_1.1
- Project Risks and Actions (PSR_1.x Appendix)
- PDD
- PDRR
- PBR_2.3

5.4.5 Stakeholder Activities

Step 7 activities include:

1) Select preferred solution
2) Develop external interfaces
3) Develop software architecture
4) Prepare for PDR
5) Conduct PDR

5.4.5.1 Select Preferred Solution

The **Development Lead** selects a preferred solution, assisted by **Development Scientists**.

5.4.5.2 Develop External Interfaces

Development Scientists assist the **Development Lead** in the identification of the external interfaces to the product processing system.

5.4.5.3 Develop Software Architecture

Development Scientists usually develop the preliminary design software architecture for the product processing system, possibly assisted by **Development Programmers**. The software system is an integrated collection of software elements, or code, that implements a solution, producing well-defined output products from a well-defined set of input data. The software architecture describes the structure of the system software elements and the external and internal data flows between software elements.

The software architecture is structured in four layers, as illustrated in Figure 5.9.

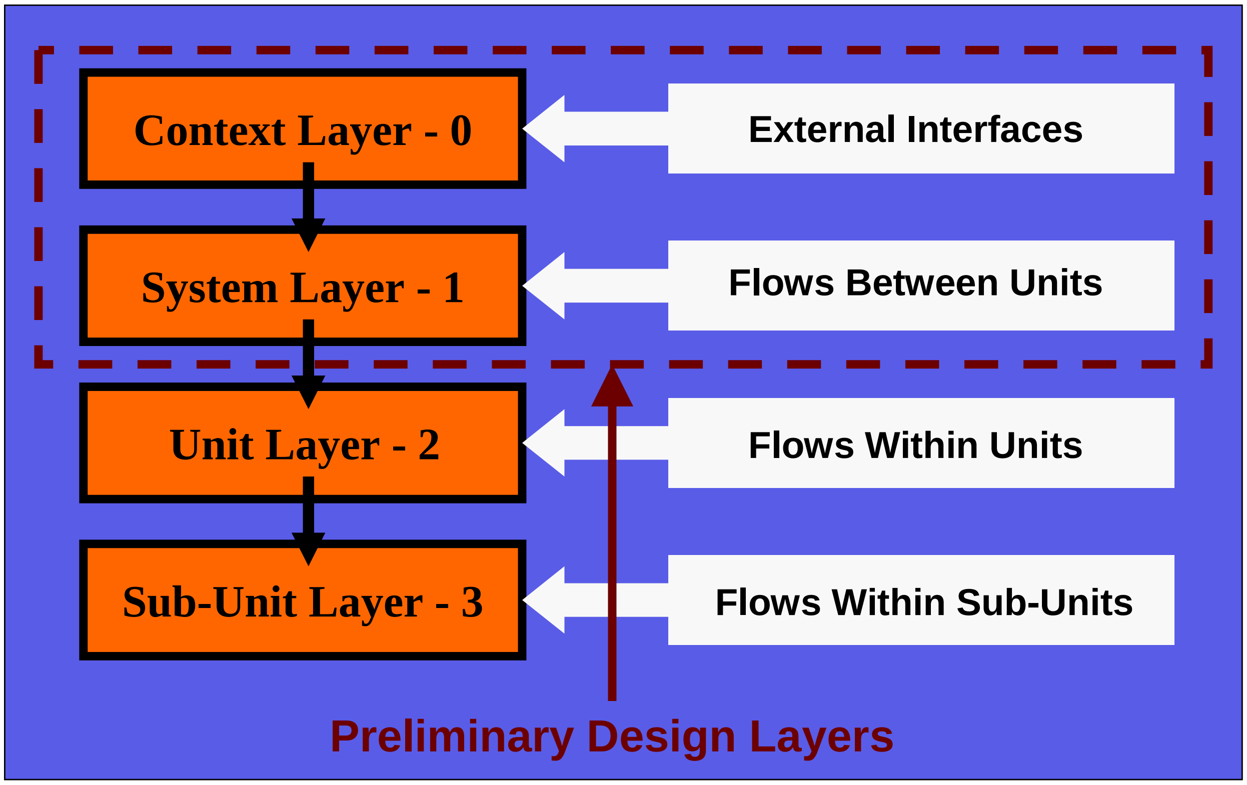

Figure 5.9 – Software Architecture Layers

As shown in Figure 5.9, the preliminary design software architecture consists of the Context Layer and the System Layer.

The Context Layer describes the flows between the system and its external interfaces.

An external input is defined as a data source needed by the system that is produced or made available by a process external to the system. Examples are raw sensor data, ancillary data, etc.

External output is defined as a data sink that is produced by the system for an external user; for example, archived environmental products (e.g. Sea Surface Temperature).

Develop a Context Layer flow diagram that illustrates these flows. An example is shown as Figure 5.10.

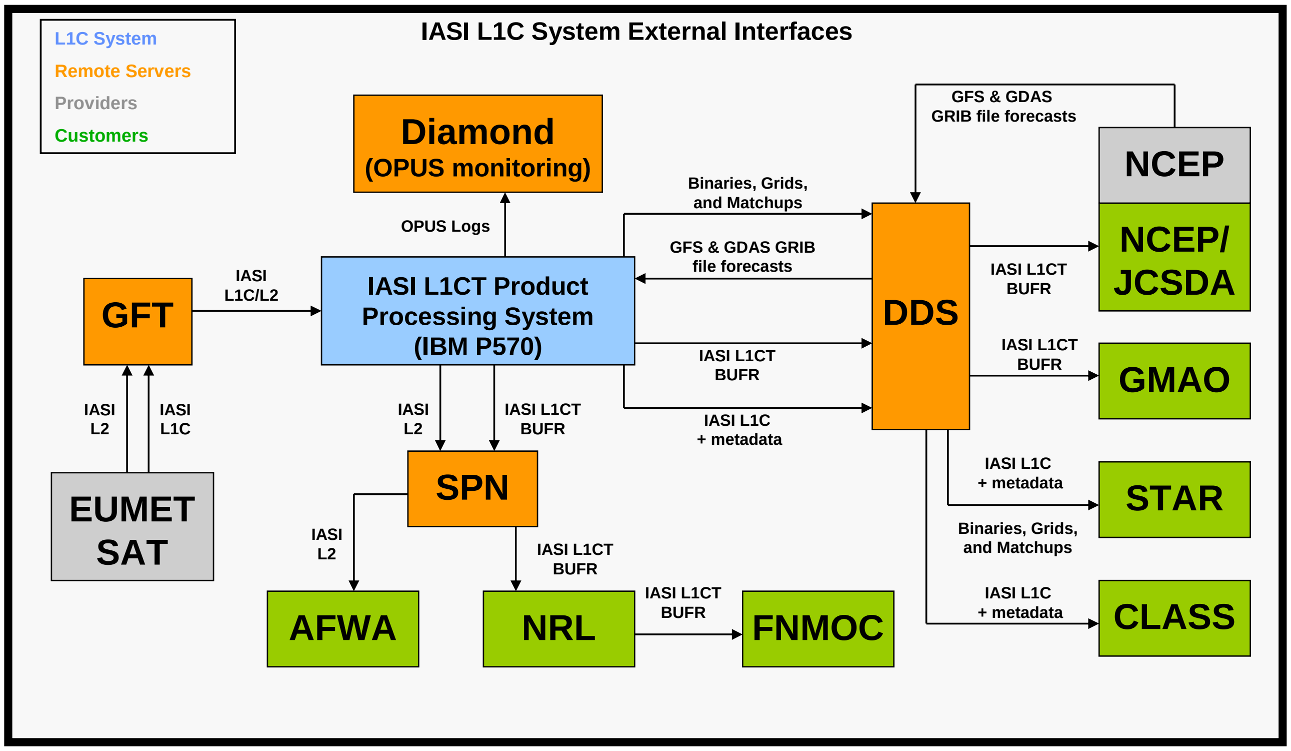

Figure 5.10 – Context Layer Data Flows

The System Layer expands upon the Context Layer, describing the first layer of decomposition. In addition to the System Layer inputs and outputs, the major processing units are identified along with their inputs and outputs.

The identification of the software units should be made with care. A software unit should contain a set of functions that meet the functional requirements. Functional requirements should have been developed during step 6 (c.f. TG-6) and may be refined iteratively with the solution and the preliminary design during step 7. Examine the functional requirements to determine which can be grouped together in a common software function. Determine which software functions can be grouped together in a common stand-alone software program that has well-defined inputs and outputs that are conducive to unit testing and integration into a System Layer scheduler. This group of functions will constitute an identified software unit.

This process of identifying the software units should result in well-defined System Layer data flows. Develop a System Layer flow diagram that illustrates these flows. An example is shown as Figure 5.11.

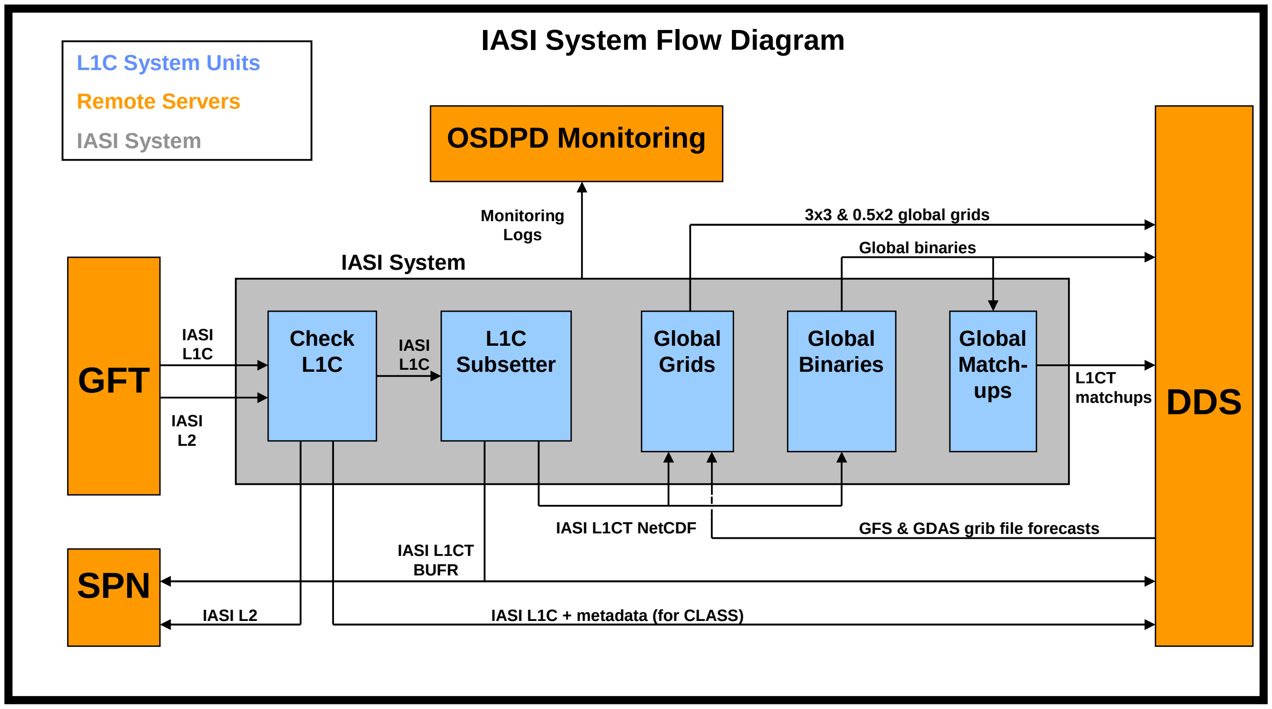

Figure 5.11 – System Layer Data Flows

When the software units are identified and traced to the functional requirements, the Preliminary Design Allocation can be completed by tracing the functional requirements to the other system requirements. The Requirements Allocation Sheet (RAS) should match each requirement to a system component. The highest layer of system components consists of the software units. Allocation of the requirements to the software units completes the Preliminary Design Allocation.

Once the Preliminary Design Allocation is completed, the **Development Scientists** and **Development Programmers** produce version 2 of the project SWA, in accordance with DG-1.2. This version of the SWA should provide a complete description of the preliminary design software architecture, in accordance with DG-1.2.

5.4.5.4 Prepare For PDR

Development Programmers assist in a revision of the RAD, following the guidelines in DG-6.2. RAD v1r1 adds to v1r0 by updating the allocation of requirements to system and product components, based on the maturing of solutions and design since PRR, as documented in SWA v2r0. It is possible that the requirements themselves must be changed by addition, deletion, or modification, based on feedback from the development of solutions and design during step 7. In that case, the RAD update should document the changes and the PDD should note what has been changed and provide a rationale for the changes.

Development Programmers assist in a revision of the VVP, following the guidelines in DG-6.3. VVP v1r1 adds to v1r0 by updating the listing and description of verification and validation items and plans, based on the maturing of the requirements allocation, solutions and design since PRR, as documented in RAD v1r1 and SWA v2r0.

The PDR slide package is the PDD. The PDD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with PDD guidelines DG-7.1. DG-7.1.A provides PDD slide templates that can be adapted for the project's PDD. The PDD developers should examine the DPP to determine whether the PDR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the PDD slide templates must be adapted to accommodate the tailoring. The PDD developers should use the project's PRD as a source for PDD slides, as many PRD slides can be re-used or adapted.

Development Programmers assist the **Development Lead** in updating the status of the project risks and associated risk mitigation actions for inclusion in the PDD and the PSR Appendix. Risk management guidelines can be found in PG-1.

5.4.5.5 Conduct PDR

The "Preliminary Design" step culminates with a PDR. The PDR consists of the presentation of the Preliminary Design Allocation by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development**

Programmers) and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the PDR to determine whether the project preliminary design is complete and sufficiently mature to proceed to detailed design. Reviewers should be familiar with the PDR guidelines (PRG-7) and check list (CL-7).

Each stakeholder who performed activities during step 7 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

5.5 Detailed Design Tasks

Figure 5.12 shows the process flow for step 8.

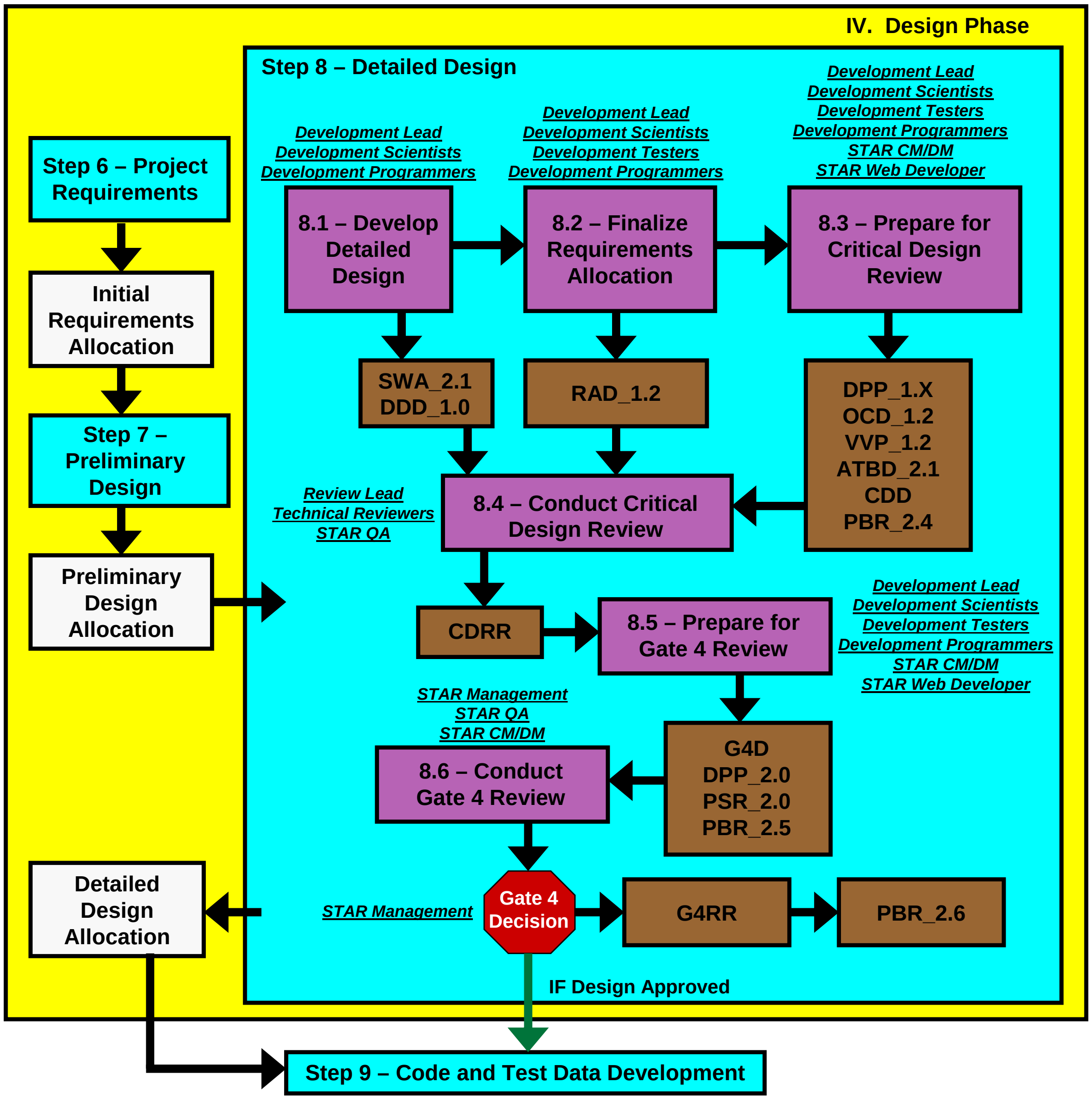

Figure 5.12 – Step 8 Process Flow

5.5.1 Expected BEGIN State

- REQUIRED: A PDR has been conducted
- REQUIRED: A preferred solution to meet the requirements has been selected and approved.

- REQUIRED: A Preliminary Design Allocation for the selected solution has been developed and approved
- REQUIRED: Baseline Build (BB) 2.3 has placed the following items in the project artifact repository:
    - DPP, including Appendices
    - OCD
    - RAD, including Appendices
    - VVP
    - ATBD
    - SWA
    - Preliminary Design Document (PDD)
    - Preliminary Design Review Report (PDRR)
- EXPECTED: BB 2.3 has placed the following items in the project artifact repository:
    - R&D code
    - R&D test data
    - PP
    - Gate 2 Review Report (G2RR)
    - Gate 3 Review Report (G3RR)
    - Project Requirements Document (PRD)
    - Project Requirements Review Report (PRRR)
- REQUIRED: PBR_2.3 documents the status of the BB 2.3 project baseline
- REQUIRED: PDR reviewers have approved the project to proceed to the Detailed Design step, and have documented this approval in the PDRR.

5.5.2 Task Inputs

Task inputs consist of the following BB 2.3 items:

- DPP_1.x,
- OCD_1.1
- RAD_1.1, including Requirements/Needs Matrix (RNM) and Requirements Allocation Sheet (RAS)
- VVP_1.1
- ATBD_2.0

- SWA_2.0
- Project Risks and Actions (PSR_1.x Appendix)
- PDD
- PDRR
- PBR_2.3

5.5.3 Desired END State

- An operations concept, developed from user/customer needs and expectations, explains what products are to be produced, why they are being produced, and how they will be produced in an operational environment,
- Basic project requirements have been developed from the operations concept
- Requirements have been analyzed in light of the customer's needs, mission objectives, system constraints, and design constraints to develop more specific product, system, and process requirements for the system.
- Derived project requirements have been developed from analysis of the basic requirements and other derived requirements
- A detailed software architecture has been developed.
- A Detailed Design Allocation of the requirements identifies product and system components down to the Sub-Unit-Layer, and traces each component to one or more requirement.
- A plan has been developed for monitoring the status of the requirements and their allocation to ensure that the integrity of the requirements allocation is preserved as the implementation of the detailed design proceeds through the Build phase.
- A plan has been developed to verify the identified work products, validate the identified requirements, and validate the identified products.
- The project plan has been updated as necessary
- The status of project risks and actions has been updated
- A CDR of the project plan, operations concept, requirements, software architecture, and requirements allocation has been conducted
- A CDRR has been written
- A Gate 4 Review of the project plan and project status has been conducted.

- A Gate 4 Review Report (G4RR) has been written, approving the project for the Build phase.
- Baseline Build 2.6 has placed the required items in the project artifact repository
- PBR_2.6 documents the status of the BB 2.6 project baseline

5.5.4 Task Outputs

Task outputs consist of the following BB 2.6 items:

- DPP_2.0
- ATBD_2.1
- SWA_2.1
- OCD_1.2
- RAD_1.2, including RNM and RAS
- VVP_1.2
- DDD_1.0
- CDD
- CDRR
- PSR_2.0, including Appendix
- G4D
- G4RR
- PBR_2.6

5.5.5 Detailed Design Activities

Step 8 activities include:

1) Develop detailed design
2) Finalize requirements allocation
3) Prepare for CDR
4) Conduct CDR
5) Prepare for Gate 4 Review
6) Conduct Gate 4 Review

5.5.5.1 Develop Detailed Design

The development of the detailed design for the product processing system is usually a collaboration between **Development Scientists** and **Development Programmers**, with **Development Scientists** usually taking the lead.

The detailed design consists of the detailed software architecture, developed by the **Development Scientists**, and a detailed code description, developed by the **Development Programmers**. The software system is an integrated collection of software elements, or code, that implements a solution, producing well-defined output products from a well-defined set of input data. The software architecture describes the structure of the system software elements and the external and internal data flows between software elements. The software architecture is structured in four layers, as illustrated in Figure 5.13.

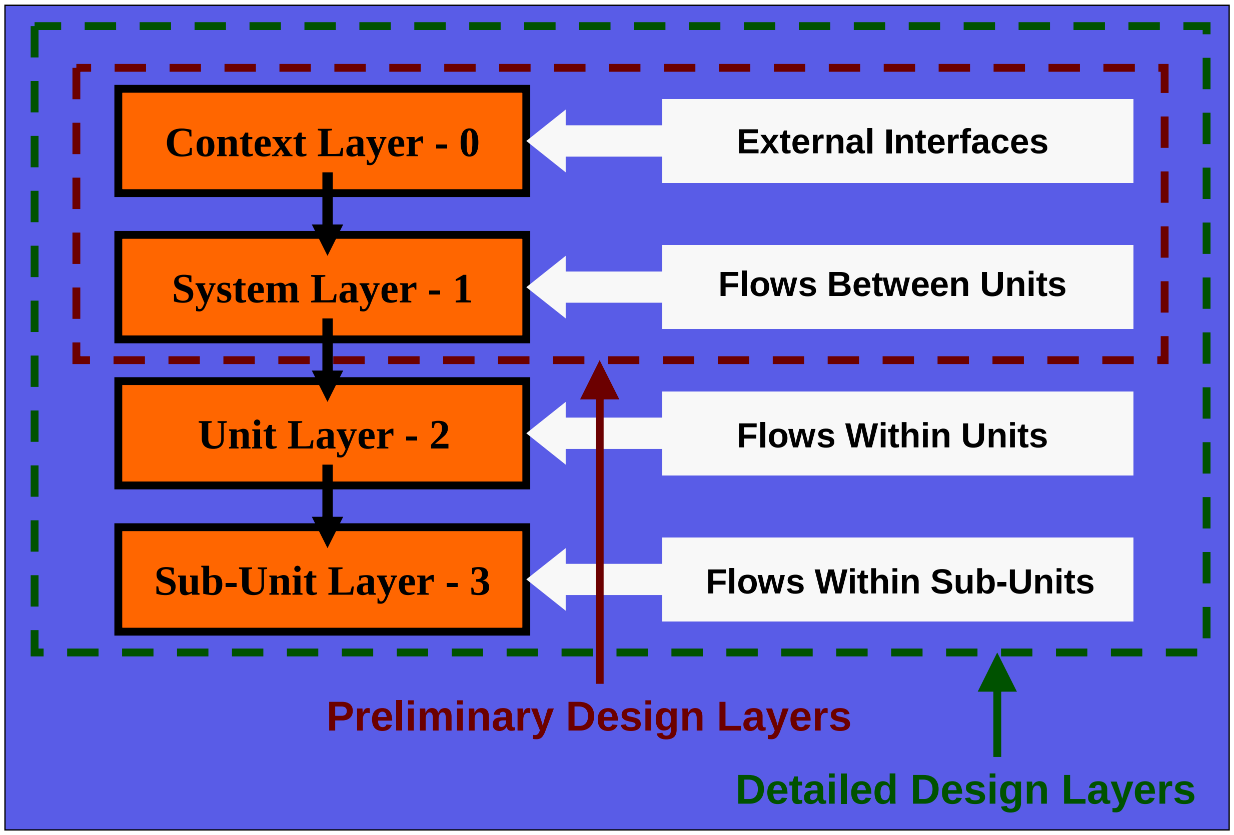

Figure 5.13 – Detailed Design Software Architecture

The Context Layer describes the flows between the system and its external interfaces (inputs and outputs).

External inputs should have been developed during step 7 and documented in SWA_2.0 (c.f. TG-7). Additional external inputs may be identified during the process of detailed design. This will occur as the functional requirements and detailed functional architecture uncover the need for additional input to support the functionality of the product processing system.

External outputs should have been developed during step 7 and documented in SWA_2.0 (c.f. TG-7). Additional external outputs may be identified during the process of detailed design. This will occur if additional requirements are identified to respond to additional requests from approved end users. It is important that risks associated with these additional requests be identified and evaluated as soon as possible. This is essential to the containment of requirements creep.

The System Layer expands upon the Context Layer, describing the first layer of decomposition. In addition to the System Layer inputs and outputs, the major processing units are identified along with their inputs and outputs.

The software units should have been established and the flows between the units developed during step 7, resulting in a well-defined System Layer architecture that was approved at the PDR (c.f. TG-7). This architecture may need to be revised if the external interfaces are changed during step 8.

The Unit Layer expands upon the System Layer, describing the second layer of decomposition. In this layer, the data flows within units are described.

The process of establishing the software units in step 7 should also have resulted in the development of software functions that meet the functional requirements, and the grouping of these functions into the software units. Complete this process by establishing the major functions of each unit. These should constitute the major elements of the Unit Layer architecture, also known as the Sub-Units. The Sub-Units constitute, the third, most detailed layer of decomposition.

To develop the Unit Layer, identify the data flows into and out of each function. This should establish a sequential order for the Sub-Units within each Unit. Once the Sub-Unit data flows and sequence are established, a Unit Layer flow diagram can be constructed. An example is shown as Figure 5.14.

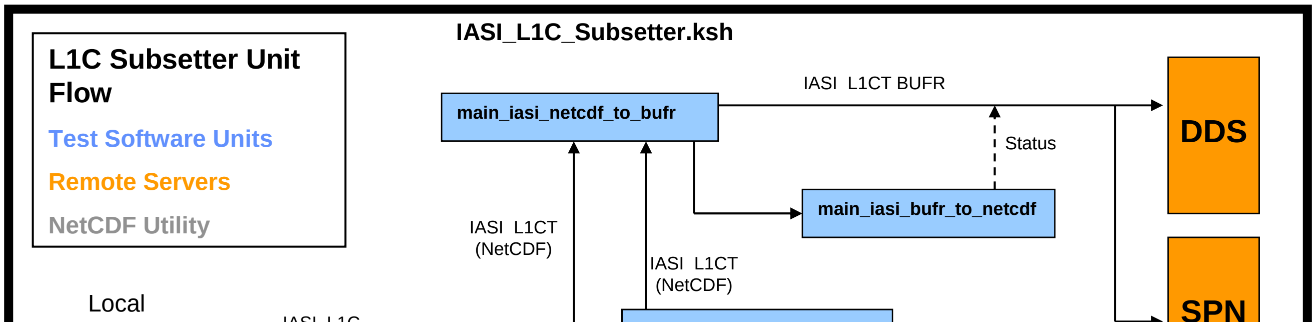

Figure 5.14 – Unit Layer Data Flows

To complete the detailed design, develop and describe the functions of each Sub-Unit. These may become software functions or may be components of a software function. The level of detail in this description should be sufficient to enable the Development Programmers to write the pre-operational code.

Upon completion of the Unit Layer and Sub-Unit Layer software architecture, the SWA should be updated to version 2.1. This update will include any revisions to the Context Layer and System Layer that were made during step 8 and add the Unit Layer and Sub-Unit Layer descriptions. Refer to DG-1.2 for SWA guidelines.

The detailed design for each software unit should be documented in its own Detailed Design Document (DDD). The DDD should provide the information needed for **Development Programmers** to write fully functional pre-operational code. Refer to DG-8.1 for DDD guidelines. The DDD should describe the unit's software functionality and design characteristics. In particular, each DDD should include design language and file descriptions.

<u>Design Language:</u> Upon completion of the detailed software architecture, a detailed code description that implements the detailed functionality can be developed. It is recommended that the software functionality be expressed in Pseudo Design Language (PDL). PDL is an exposition of the data flows and functional sequences in a style that resembles code. The idea is to use the software architecture to begin to visualize how the code will look. It is here that the design begins to be translated into a "pseudo-code" in a way that is understandable by both the designer and the programmer. In this way, the designer verifies that the programmer understands how to implement the design and the programmer is assured that the pseudo-code is a satisfactory basis for programming. The **Development Programmers** should take the lead in writing the PDL, in close consultation with the **Development Scientists** who developed the detailed software architecture.

File Descriptions: The DDD complements the SWA by providing detailed descriptions of the input files, intermediate files, and output files, including control files, parameter files, look up tables, input data files, ancillary data files, intermediate data files, and output data files. The **Development Programmers** should take the lead in writing the file descriptions, in close consultation with the **Development Scientists** who developed the detailed software architecture.

Control files are typically scripts that define run control parameters. For each file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

Parameter files contain the values of variables that are fed into the unit program or sub-programs. For each file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

Look up tables typically contain the values of variables binned by a range of conditions, or stratifications. For each file, indicate the variables it contains, how they are binned, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each ancillary data file, indicate the source of the file, how the file is obtained, references to relevant file description documentation by the file provider, the variables contained in the file, how they are binned (if they are binned), the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each input data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each intermediate data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

For each output data file, indicate the variables it contains, the file format, and how the values of the variables are read into the program or subprograms that use them. The file format is usually best described by a table.

5.5.5.2 Finalize Requirements Allocation

The Detailed Design Allocation represents the culmination of the iterative development of requirements, solutions, and design during the Design phase. The Detailed Design Allocation is achieved when it is determined that a complete design has been developed to implement the preferred solution that was approved at the PDR, including all four layers of the software architecture, and a detailed code description.

Development Programmers assist in a revision of the RAD, following the guidelines in DG-6.2. RAD v1r2 adds to v1r1 by updating the allocation of requirements to system and product components, based on the maturing of solutions and design since PDR, as documented in SWA v2r1. It is possible that the requirements themselves must be changed by addition, deletion, or modification, based on feedback from the development of solutions and design during step 8. In that case, the RAD update should document the changes and the CDD should note what has been changed and provide a rationale for the changes.

5.5.5.3 Prepare for CDR

Development Programmers assist in a revision of the VVP, following the guidelines in DG-6.3. VVP v1r2 adds to v1r1 by updating the listing and description of verification and validation items and plans, based on the maturing of the requirements allocation, solutions and design since PDR, as documented in RAD v1r2 and SWA v2r1.

The CDR slide package is the CDD. The CDD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with CDD guidelines DG-8.2. DG-8.2.A provides CDD slide templates that can be adapted for the project's CDD. The CDD developers should examine the DPP to determine whether the CDR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the CDD slide templates must be adapted to accommodate the tailoring. The CDD developers should use the project's PDD as a source for CDD slides, as many PDD slides can be re-used or adapted.

Development Programmers assisted in updating the status of the project risks and associated risk mitigation actions for inclusion in the CDD and the PSR Appendix. Risk management guidelines can be found in PG-1.

5.5.5.4 Conduct CDR

The CDR consists of the presentation of the Detailed Design Allocation by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the CDR to determine whether the project detailed design is complete and sufficiently mature to proceed to the Build phase. Reviewers should be familiar with the CDR guidelines (PRG-8.1) and check list (CL-8.1).

5.5.5.5 Prepare Gate 4 Review

Once the project passes its CDR, it is referred to the Gate 4 Review. The Gate 4 review is included in step 8 because the project status and plan will usually be modified significantly during the Design phase, so a management review of the project plan and status is typically desirable.

Development Programmers assist the **Development Lead** in updating the PSR to version 2. Version 2 of the PSR, along with its Appendix, documents the status of project tasks, cost, schedule, risks, and actions at the conclusion of the Design phase. Refer to PSR guidelines in DG-5.2 and DG-5.2.A.

The **Development Lead** leads the preparation of the Gate 4 Review presentation. The presentation slide package is the Gate 4 Document (G4D). The G4D is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with G4D guidelines DG-8.4. DG-8.4.A provides G4D slide templates that can be adapted for the project's G4D. The G4D developers should examine the DPP to determine whether the Gate 4 Review objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the G4D slide templates must be adapted to accommodate the tailoring.

5.5.5.6 Conduct Gate 4 Review

The “Detailed Design” step culminates with a Gate 4 Review. The Gate 4 Review consists of the presentation of the project plan and project status by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**STAR Management**).

Each stakeholder who performed activities during step 8 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

The **Development Lead** should remind the stakeholders to do this. At the conclusion of Development (step 11), the **Development Lead** will collect the final edited personal stakeholder records and incorporate them into a Development Project Report (DPR).

5.6 Pre-Operational System Development Process

Pre-operational system development is an iterative process that occurs throughout the Build phase of the product lifecycle. This phase includes three steps that produce an integrated product processing system through an iterative (spiral) development of code, test data and test plans.

- Code & Test Data Development (step 9 of the STAR EPL)
- Code Test & Refinement (step 10 of the STAR EPL)
- System Integration & Test (step 11 of the STAR EPL)

Figure 5.15 illustrates the pre-operational system development process, with step 9 highlighted.

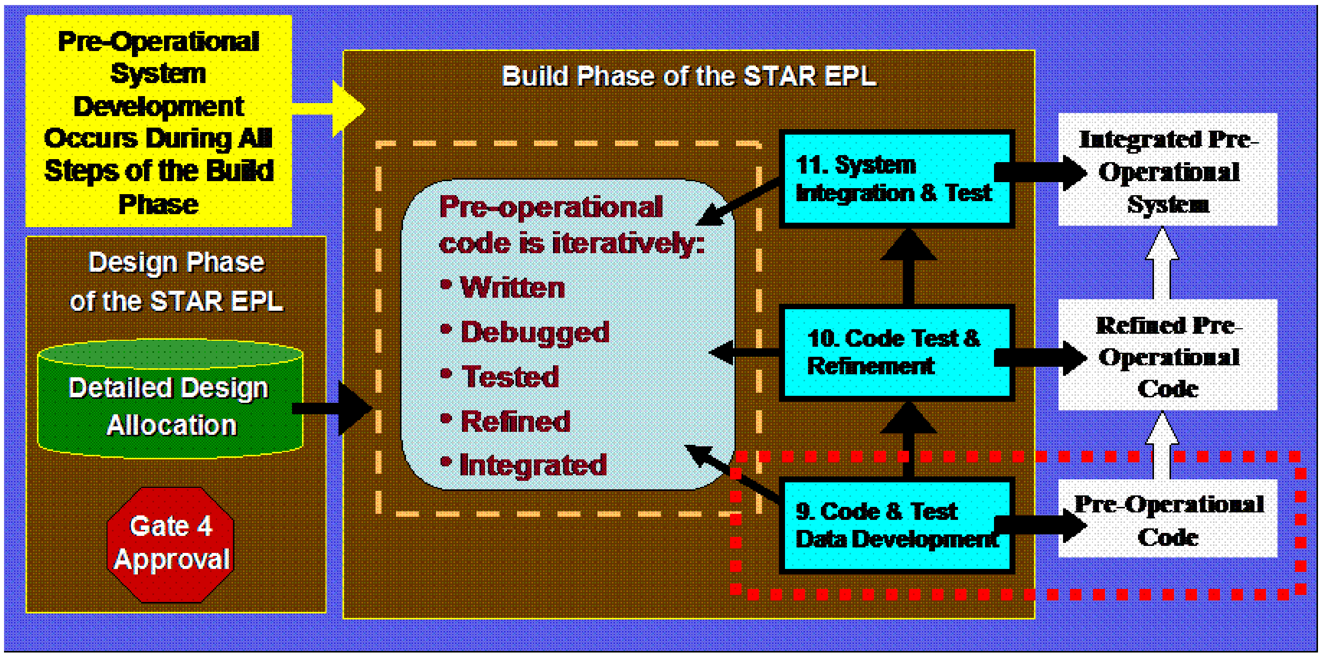
Pre-Operational System Development Occurs During All Steps of the Build Phase
Design Phase of the STAR EPL
Detailed Design Allocation
Gate 4 Approval
Build Phase of the STAR EPL
Pre-operational code is iteratively:
• Written
• Debugged
• Tested
• Refined
• Integrated
11. System Integration & Test
10. Code Test & Refinement
9. Code & Test Data Development
Integrated Pre-Operational System
Refined Pre-Operational Code
Pre-Operational Code

Figure 5.15 – Pre-Operational System Development Process

As Figure 5.15 shows, the objective of step 9 is to produce pre-operational code. Pre-operational code consists of the system components in the detailed design (software units and sub-units). This code will be written and debugged, and ready for unit testing, but not formally tested.

The process of producing a complete pre-operational product processing system involves writing, debugging, testing, refining, and integrating the code. Because these functions affect each other, the process is inherently iterative. Figure 5.16 illustrates this.

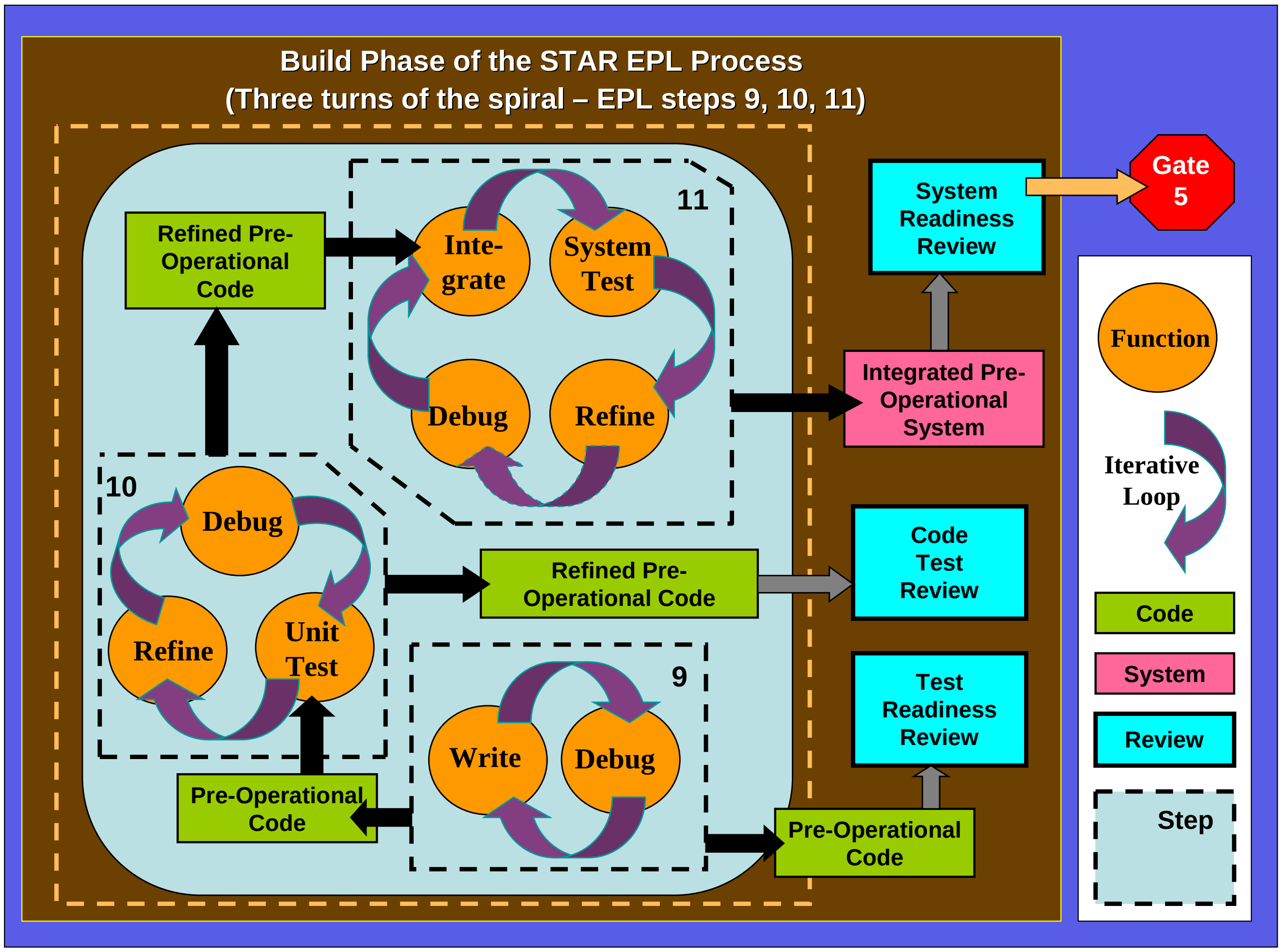

Figure 5.16 – Iterative (Spiral) Development of Pre-Operational System

Note that steps 10 and 11 continue pre-operational system development. The refinement of the step 9 pre-operational code and integration into a complete pre-operational product processing system is expected during these steps. Therefore, the objective of the step 9 activities (write and debug code) is limited to producing code that is sufficiently mature and complete to allow unit testing. This is illustrated in Figure 5.16 as the output from step 9 is pre-operational code that is input to the Unit Test function of step 10.

5.7 Code & Test Data Development Tasks

Figure 5.17 shows the process flow for step 9.

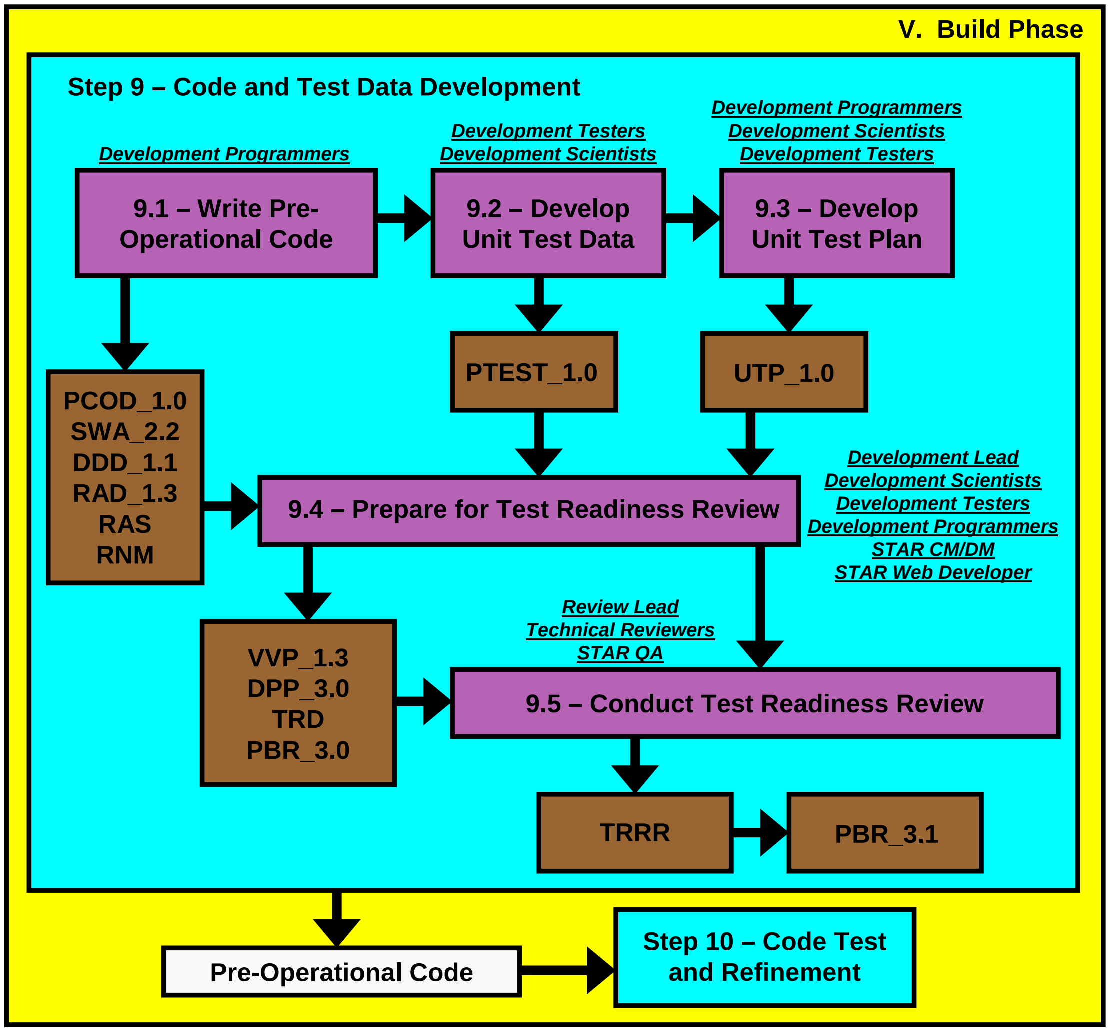

Figure 5.17 – Step 9 Process Flow

5.7.1 Expected BEGIN State

- REQUIRED: A CDR has been conducted

- REQUIRED: A Gate 4 Review has been conducted
- REQUIRED: A Detailed Design Allocation has been developed and approved
- REQUIRED: Baseline Build (BB) 2.6 has placed the following items in the project artifact repository:
  - DPP, including Appendices
  - RAD, including Appendices
  - VVP
  - SWA
  - DDD
  - Critical Design Document (CDD)
  - Critical Design Review Report (CDRR)
  - Gate 4 Document (G4D)
  - Gate 4 Review Report (G4RR)
  - PSR, including Appendix
  - PBR
- EXPECTED: BB 2.6 has placed the following items in the project artifact repository:
  - R&D code
  - R&D test data
  - Project Proposal (PP)
  - Gate 2 Review Report (G2RR)
  - Gate 3 Review Report (G3RR)
  - Operations Concept Document (OCD)
  - ATBD
  - Project Requirements Document (PRD)
  - Project Requirements Review Report (PRRR)
  - Preliminary Design Document (PDD)
  - Preliminary Design Review Report (PDRR)
- REQUIRED: PBR_2.6 documents the status of the BB 2.6 project baseline
- REQUIRED: Gate 4 reviewers have approved the project to proceed to the Code and Test Data Development step, and have documented this approval in the G4RR.

5.7.2 Task Inputs

Task inputs consist of the following BB 2.6 items:

- DPP_2.0
- SWA_2.1
- RAD_1.2
- Requirements/Needs Matrix (RNM)
- Requirements Allocation Sheet (RAS)
- VVP_1.2
- DDD_1.0
- CDD
- CDRR
- PSR_2.0, including Appendix
- G4D
- G4RR
- PBR_2.6

5.7.3 Desired END State

- A Detailed Design Allocation of the requirements identifies product and system components down to the Sub-Unit-Layer, and traces each component to one or more requirement.
- A plan has been developed for monitoring the status of the requirements and their allocation to ensure that the integrity of the requirements allocation is preserved as the implementation of the detailed design proceeds through the Build phase.
- The functionality of all system components in the detailed design (software units and sub-units) has been implemented in pre-operational code that meets coding standards.
- Pre-operational code has been debugged, compiled, and built into executable software units, ready for unit testing in the designated test environment.

- A plan for unit testing has been developed. The plan ensures that the unit tests will address all required code functionality and code outputs.
- All data required for implementation of the unit test plan has been acquired or developed, and is available in the designated test environment.
- The project plan has been updated as necessary
- The status of project risks and actions has been updated
- A TRR of the project plan, software architecture, and unit test readiness has been conducted
- A TRRR has been written. The TRRR approves the project to proceed to the next step, Code Test and Refinement.
- Baseline Build 3.1 has placed the required items in the project artifact repository
- PBR_3.1 documents the status of the BB 3.1 project baseline

5.7.4 Task Outputs

Task outputs consist of the following BB 3.1 items:

- Pre-Operational Code
- Pre-Operational Test Data
- DPP_3.x
- SWA_2.2
- RAD_1.3, including RNM and RAS
- VVP_1.3
- DDD_1.1
- UTP_1.0
- TRD
- TRRR
- PBR_3.1

5.7.5 Stakeholder Activities

Step 9 activities include:

1) Write pre-operational code
2) Develop unit test data
3) Develop unit test plan
4) Prepare for TRR
5) Conduct TRR

5.7.5.1 Write Pre-Operational Code

Development Programmers write the pre-operational code to implement the detailed design, following coding standards provided in TD-11.1 (FORTRAN code) and TD-11.2 (C code). Pre-operational code consists of the system components in the detailed design (software units and sub-units). This code will be written and debugged, and ready for unit testing, but not formally tested.

Use the SWA and DDDs (PDL and file descriptions) as a blueprint for writing the code.

It is recommended that diagnostic code be added to the unit code to facilitate the verification of functionality and outputs. If this is done, two versions of the unit code should be created. The first version will contain the diagnostic code. The second version, with all diagnostic code stripped out, is the baseline pre-operational code.

The code should be debugged until it runs without errors and produces the designed output. Debugging should be accomplished by reviewing compiler error messages and making the indicated corrections. When the code compiles without errors, it should be built into executables in the test environment and run to verify that there are no runtime errors. For this reason, it is recommended that the pre-operational code and test data (c.f. Section 6.5.2) be developed concurrently.

In the process of writing and debugging the pre-operational code, problems with the detailed design may be discovered. In that case, the **Development Scientists** and **Development Programmers** should determine whether the detailed design needs revision to correct the problems. If the revisions are relatively minor and do not affect the Detailed Design Allocation, as is usually the case, document the revisions in updates of the SWA (v2r2) and/or the DDD (v1r1). If the revisions do affect the Detailed Design Allocation, trace the affected system components or product components back to the driving requirements to ensure that these are not compromised by the design revisions.

Update the RAD and its Appendices (RNM and RAS) to v1r3 to capture any changes to the Detailed Design Allocation that have occurred as a result of design revisions during step 9.

When the procedure to build the test configuration into a test executable has been determined (c.f. Section 6.5.3), implement this procedure to verify its correct functionality. A set of unit test executables should be built to verify unit test readiness. At the discretion of the development team, these executables may be run as a unit test “dry run”. A dry run can be useful in identifying problems that require a revision of the unit test plan and/or test items. The process of dry runs plus revision should not be extensive, though. Most of the effort to test and refine the code should occur during step 10. In particular, the TRR should not be delayed by this process.

5.7.5.2 Develop Unit Test Data

Development Testers and **Development Scientists** collaborate to produce test data for unit testing of the pre-operational code.

5.7.5.3 Develop Unit Test Plan

Development Programmers, Development Scientists and Development Testers collaborate in the development of a unit test plan, documented in UTP v1r0, following guidelines in DG-9.1.

Explain the purpose of each unit, the role of the unit in the product processing system, the major functional steps, and how these steps satisfy the purpose of the unit. This information should be obtained from the SWA and DDD.

Identify all unit components that have been selected for testing. These items should be part of the system architecture as documented in the SWA. Typically, they are sub-processes of the unit’s process flow. They should also be identified as verification items in the project’s VVP. It is important that the development team confirm that the planned unit test items are documented as verification items in the VVP. Discrepancies must be resolved and either the UTP or the VVP revised as necessary.

Identify the requirements allocated to each test item, consistent with the Detailed Design Allocation.

Determine and describe the environment in which the unit tests will be performed. A project may use the development environment to test the pre-operational system or it may choose to establish a separate test environment. Project constraints will usually determine this choice. For example, operations may request that the test environment be a clone of the operational environment, but cost factors may exclude establishing the development environment as an operational clone. In that case, the best solution may be to use a small operational clone environment as a separate test environment. A project may also choose to perform its pre-operational code unit tests in the development environment and then perform its system integration tests in an operational clone environment. In any case, these choices should be explicitly stated as requirements for the test environment.

Identify all configuration items that will be used in the unit test, including code modules, test data sets, utilities, libraries, etc. The set of configuration items is called the test configuration. Identify the procedure to build the test configuration into a test executable.

Determine and describe the methods that will be used to test each test item. Test methods should be closely related to the verification methods that are documented in the VVP. It is permissible to insert relevant material from the VVP into the UTP, provided it is referenced appropriately. Alternatively, the UTP developers may choose to leave the specific material out of the UTP and refer to specific sections of the VVP that pertain to the test methods for a unit. The latter choice is recommended if the TRR reviewers are already familiar with the material in the project's VVP. It is recommended that the UTP developers consult with the TRR reviewers before deciding how to document test methods in the UTP.

Describe the planned sequence of test actions in sufficient detail that a reviewer can confirm that all test items are exercised, all test data are utilized, all planned test methods are used as planned, and the planned output will allow a reviewer to confirm that the requirements will be satisfied. Note which sequence steps exercise which test items, utilize which test data sets, and use which test methods.

Describe the expected output from each sequence step. The expected output should be described in sufficient detail for unit test reviewers to be able to confirm that the actual unit test output matches the expected output. Output includes runtime messages, diagnostic messages and the content of data files.

State the success criteria for each unit test. Success criteria should include the following:

- All test sequence steps run as planned in the UTP
- Program run time meets requirements
- All runtime message and diagnostic messages are written as expected in the UTP
- The contents of all diagnostic files are as expected in the UTP
- Unit test output satisfies all quantitative performance requirements

For each unit, determine and note which other units must be run to supply input data for the unit test. It is common for the software units to be linked in a sequential order. The SWA and DDD will contain this information.

5.7.5.4 Prepare for TRR

Development Programmers assist in a revision of the VVP, following the guidelines in DG-6.3. VVP v1r3 adds to v1r2 by updating the listing and description of verification and validation items and plans, based on changes to the Detailed Design Allocation since CDR, as documented in RAD v1r3.

The TRR slide package is the Test Readiness Document (TRD). The TRD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with TRD guidelines DG-9.2. DG-9.2.A provides TRD slide templates that can be adapted for the project's TRD. The TRD developers should examine the DPP to determine whether the TRR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the TRD slide templates must be adapted to accommodate the tailoring. The TRD developers should use the project's CDD as a source for TRD slides, as many CDD slides can be re-used or adapted.

Development Programmers assist in updating the status of the project risks and associated risk mitigation actions for inclusion in the TRD. Risk management guidelines can be found in PG-1.

5.7.5.5 Conduct TRR

The TRR consists of the presentation of the pre-operational code, test data, and test plan by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the TRR to determine whether the pre-operational code conforms to coding standards and is ready for unit testing. Reviewers should be familiar with the TRR guidelines (PRG-9) and check list (CL-9).

Each stakeholder who performed activities during step 9 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

The **Development Lead** should remind the stakeholders to do this. At the conclusion of Development (step 11), the **Development Lead** will collect the final edited personal stakeholder records and incorporate them into a Development Project Report (DPR).

5.8 Code Test and Refinement Tasks

Figure 5.18 shows the process flow for step 10.

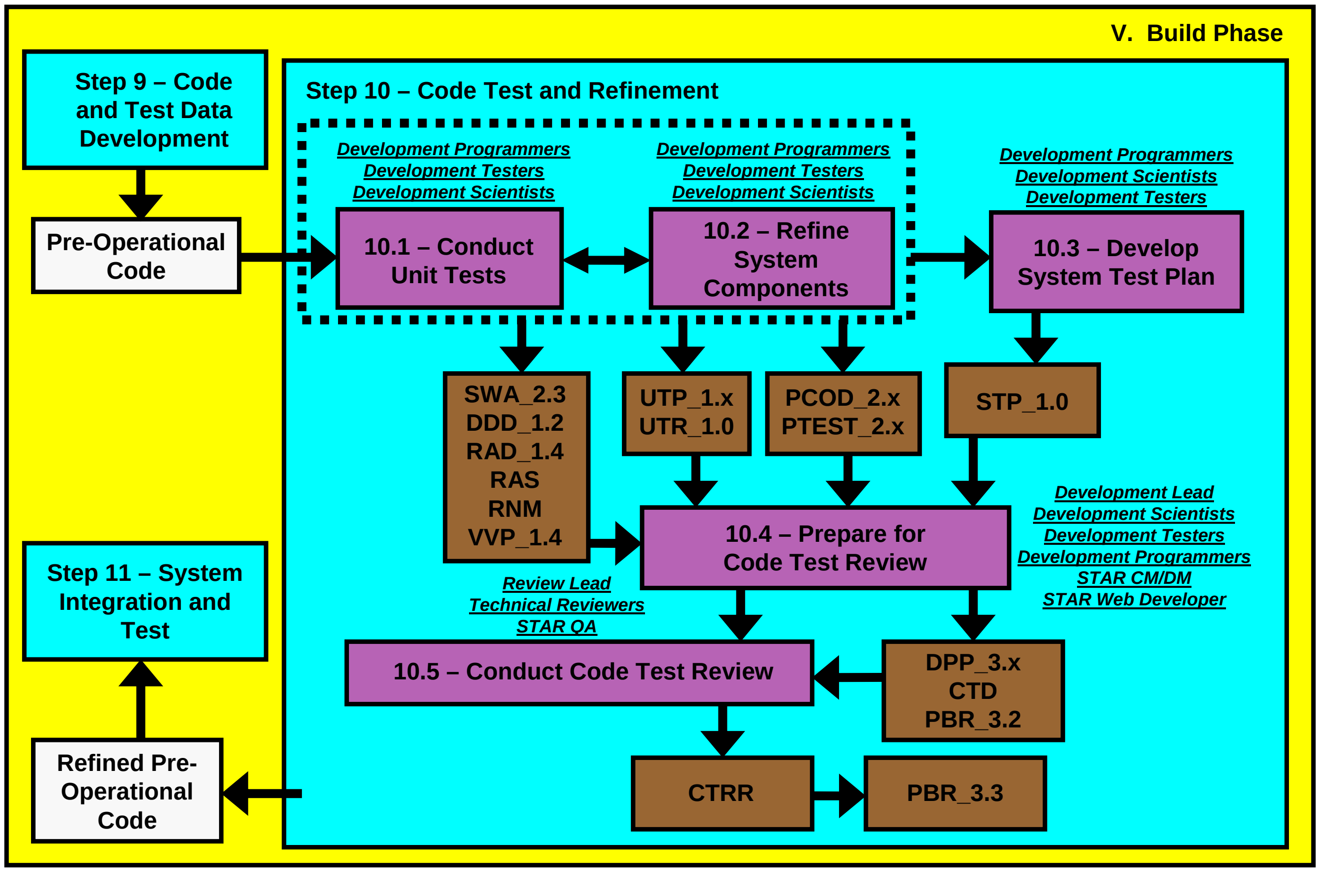

Figure 5.18 – Step 10 Process Flow

Note that processes 10.1 and 10.2 are enclosed by a common dashed border. This is to indicate that the processes are iterative, as explained in Section 5.6.

5.8.1 Expected BEGIN State

- REQUIRED: Pre-operational code has been debugged, compiled, and built into executable software units, ready for unit testing in the designated test environment.
- REQUIRED: A plan for unit testing has been developed. The plan ensures that the unit tests will address all required code functionality and code outputs.
- REQUIRED: All data required for implementation of the unit test plan has been acquired or developed, and is available in the designated test environment.
- REQUIRED: A TRR has been conducted
- REQUIRED: TRR reviewers have approved the project to proceed to the Code Test and Refinement step, and have documented this approval in the Test Readiness Review Report (TRRR).
- REQUIRED: Baseline Build (BB) 3.1 has placed the following items in the project artifact repository:
    - Pre-operational code
    - Unit test data
    - DPP, including Appendices
    - RAD, including Appendices
    - VVP
    - ATBD
    - SWA
    - DDD
    - UTP
    - Test Readiness Document (TRD)
    - TRRR
- EXPECTED: BB 3.1 has placed the following items in the project artifact repository:
    - R&D code
    - R&D test data
    - Project Proposal (PP)
    - Gate 2 Review Report (G2RR)

- Gate 3 Review Report (G3RR)
- Operations Concept Document (OCD)
- Project Requirements Document (PRD)
- Project Requirements Review Report (PRRR)
- Preliminary Design Document (PDD)
- Preliminary Design Review Report (PDRR)
- Critical Design Document (CDD)
- Critical Design Review Report (CDRR)
- Gate 4 Document (G4D)
- Gate 4 Review Report (G4RR)
- Project Status Report (PSR), including Appendix

- REQUIRED: PBR_3.1 documents the status of the BB 3.1 project baseline

5.8.2 Task Inputs

Task inputs consist of the following BB 3.1 items:

- Pre-operational code (PCOD_1.x)
- Pre-operational test data (PTEST_1.x)
- SWA_2.2
- DDD_1.1
- UTP_1.0
- DPP_3.x
- RAD_1.3
- Requirements/Needs Matrix (RNM)
- Requirements Allocation Sheet (RAS)
- VVP_1.3
- TRD
- TRRR
- PSR_2.x Appendix
- PBR_3.1

5.8.3 Desired END State

- A Detailed Design Allocation of the requirements identifies product and system components down to the Sub-Unit-Layer, and traces each component to one or more requirement.
- A plan has been developed for monitoring the status of the requirements and their allocation to ensure that the integrity of the requirements allocation is preserved as the implementation of the detailed design proceeds through the Build phase.
- The functionality of all system components in the detailed design (software units and sub-units) has been implemented in pre-operational code that meets coding standards.
- A plan for unit testing has been developed. The plan ensures that the unit tests will address all required code functionality and code outputs.
- Pre-operational code has been tested in accordance with the unit test plan.
- Pre-operational code has been refined and debugged as necessary until it passes all unit tests.
- Unit test results have been documented in a report.
- A plan for system testing has been developed. The plan ensures that the system test will address all system requirements and product requirements.
- The project plan has been updated as necessary
- The status of project risks and actions has been updated
- A CTR of the project plan, unit test results, and system test plan has been conducted
- A CTRR has been written. The CTRR approves the project to proceed to the next step, System Integration and Test.
- Baseline Build 3.3 has placed the required items in the project artifact repository
- PBR_3.3 documents the status of the BB 3.3 project baseline

5.8.4 Task Outputs

Task outputs consist of the following BB 3.3 items:

- Refined Pre-Operational Code

- Refined Pre-Operational Test Data
- DPP_3.x
- SWA_2.3
- DDD_1.2
- RAD_1.4, including RNM and RAS
- VVP_1.4
- UTP_1.x
- STP_1.0
- CTD
- CTRR
- PBR_3.3

5.8.5 Stakeholder Activities

Step 10 activities include:

1) Conduct unit tests
2) Refine system components
3) Develop system test plan
4) Prepare for CTR
5) Conduct CTR

5.8.5.1 Conduct Unit Tests

Development Testers run the unit tests, assisted by **Development Programmers**. Examine the unit test output, including runtime messages, diagnostic messages and the content of intermediate and output data files.

Runtime messages are messages written by the operating system to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). These may occur if the unit code is written to generate such messages as a way to test functionality.

Diagnostic messages are messages written by the unit program to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). The nominal purpose of a diagnostic message is to report a functional result (e.g., 'subroutine X called') or the quantitative value of an input, intermediate, or output variable (e.g., 'X(50) = 7').

Data files include the output data sets that are designed to be produced by the unit program. In addition, a diagnostic program may write intermediate data sets to diagnostic files.

If the unit test output does not satisfy all success criteria, iteratively refine and debug the code (c.f. Section 6.5.2), and repeat unit testing until success criteria are satisfied. When success criteria are satisfied, document the results in the UTR, following guidelines in DG-10.1. **Development Testers** lead in the development of the UTR. **Development Programmers** and **Development Scientists** assist with the UTR. The purpose of UTR v1r0 is to document the results of testing of each software unit to verify that the requirements allocated to the unit's software components are satisfied.

5.8.5.2 Refine System Components

Development Programmers iteratively refine, debug and re-test the pre-operational code as needed, based on the unit test results. Code refinements should conform to coding standards provided in TD-11.1 (FORTRAN code) and TD-11.2 (C code).

Development Programmers assist in revising the UTP to account for changes to the unit test plan since the TRR. UTP v1r1 updates and refines the test data description, test methods, test sequences and test risks, based on the refinement of the code and test data.

In the process of refining and debugging the pre-operational code, problems with the detailed design may be discovered. In that case, the **Development Scientists** and **Development Programmers** should determine whether the detailed design needs revision to correct the problems. If the revisions are relatively minor and do not affect the Detailed Design Allocation, as is usually the case, document the revisions in updates of the SWA (v2r3) and/or the DDD (v1r2). If the revisions do affect the Detailed Design Allocation, trace the affected system components or product components back to the driving requirements to ensure that these are not compromised by the design revisions.

Development Programmers assist in updating the RAD and its Appendices (RNM and RAS) to v1r4 to capture any changes to the Detailed Design Allocation that have occurred as a result of design revisions during step 10.

Development Programmers assist in a revision of the VVP, following the guidelines in DG-6.3. VVP v1r4 adds to v1r3 by updating the listing and description of verification and validation items and plans, based on changes to the Detailed Design Allocation since TRR, as documented in RAD v1r4.

5.8.5.3 Develop System Test Plan

Development Programmers, Development Scientists and Development Testers collaborate in the development of a system test plan, documented in STP v1r0, following guidelines in DG-10.2. The purpose of STP v1r0 is to present the plan for testing to ensure that the requirements specified for the product processing system (PPS) are satisfied by the completed system (Verification) and that the final developed system will satisfy the users' needs and expectations (Validation). The purpose of the system test is to demonstrate, using verification and validation methods, system readiness for operations. The STP builds on the project's VVP and UTP.

Identify the system requirements and product requirements that will be tested. Trace these to user needs and operator needs. These should be documented in the RAD, RNM, and OCD.

Identify all items that have been selected for testing. These items are system components and product components that should have been identified as verification items in the project's VVP. It is important that the development team confirm that the planned system test items are documented as verification items in the VVP. Discrepancies must be resolved and either the STP or the VVP revised as necessary.

System components are defined as any item that is necessary or useful for building the end-use product, but will not be delivered to customers and/or end users. System components include the elements of the software architecture, as described in the UTP. Typically, they are elements of the process flow of the software units that have been tested in the unit tests, as described in the UTP. Typically, the system test will include the end-to-end execution of the software units.

Product components are defined as any item that will be integrated to form the end-use product, i.e. these are the deliverable items. Typically, the product components are outputs from the end-to-end execution of the software units.

List and describe all data files that will be used as input files for the system test. Files to be listed here include: “Test data” includes sensor data (real, proxy, or simulated), ancillary data, control files, parameter files, and look up tables. Files to be listed here include:

- “Test data”. These data sets include the sensor data (real, proxy, or simulated), ancillary data, control files, parameter files, and look up tables that are needed to run the system test.
- “Truth” data. These are data sets that will be used to assess the quality of the system output. Truth data sets typically contain the values of environmental or weather products that are traceable to performance requirements. Truth data sets may be real, proxy or simulated data. Explain how each real or proxy truth data set has been obtained. Explain how each simulated truth data set has been constructed.

Typically, the system test will use some or all of the same test data and truth data that was used for the unit tests. In that case, information on this data can be obtained from the UTP. If the system test includes new test data in addition to the unit test data, add a description of this data to the STP.

Determine and describe the environment in which the system test will be performed. A project may use the development environment to test the pre-operational system or it may choose to establish a separate test environment. Project constraints will usually determine this choice. For example, operations may request that the test environment be a clone of the operational environment, but cost factors may exclude establishing the development environment as an operational clone. In that case, the best solution may be to use a small operational clone environment as a separate test environment. A project may also choose to perform its pre-operational code unit tests in the development environment and then perform its system integration tests in an operational clone environment. In any case, these choices should be explicitly stated as requirements for the test environment

It is preferable to use the same test environment for the system test as was used for all unit tests. For cases where the test environments differ, the system test should include verification that the same inputs to each software unit results in identical outputs when the unit is run in the system test environment.

Determine and describe the test methods that will be used. Test methods should be closely related to the verification methods that are documented in the VVP. The standard methods include Analysis, Demonstration, Inspection, and Test. Note that unit test methods are not restricted to “Test”. The “Test” verification method refers specifically to a procedure to quantitatively demonstrate compliance with performance specifications. Although test methods will often include “Test” methods for verifying quantitative requirements, they can also include, and usually will include, other methods for verifying other requirements. Note which test items will be verified with each method or combination of methods.

It is permissible to insert relevant material from the VVP into the STP, provided it is referenced appropriately. Alternatively, the STP developers may choose to leave the specific material out of the STP and refer to specific sections of the VVP that pertain to the test methods. The latter choice is recommended if the CTR reviewers are already familiar with the material in the project's VVP. It is recommended that the STP developers consult with the CTR reviewers before deciding how document test methods in the STP.

Identify all configuration items that will be used in the unit test, including code modules, test data sets, utilities, libraries, etc. The set of configuration items is called the test configuration. Identify the procedure to build the test configuration into a test executable.

Describe the planned sequence of test actions in sufficient detail that a reviewer can confirm that all test items are exercised, all test data are utilized, all planned test methods are used as planned, and the planned output will allow a reviewer to confirm that the requirements will be satisfied. Note which sequence steps exercise which test items, utilize which test data sets, and use which test methods.

Describe the expected output from each sequence step. The expected output should be described in sufficient detail for system test reviewers to be able to confirm that the actual system test output matches the expected output. Output includes runtime messages, diagnostic messages and the content of data files.

State the success criteria. Success criteria should include the following:

- All test sequence steps run as planned in the STP
- Program run time meets requirements
- All runtime message and diagnostic messages are written as expected in the STP
- The contents of all diagnostic files are as expected in the STP
- System test output satisfies all quantitative performance requirements

5.8.5.4 Prepare for CTR

The CTR slide package is the Code Test Document (CTD). The CTD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with CTD guidelines DG-10.3. DG-10.3.A provides CTD slide templates that can be adapted for the project's CTD. The CTD developers should examine the DPP to determine whether the CTR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the CTD slide templates must be adapted to accommodate the tailoring. The CTD developers should use the project's TRD as a source for CTD slides, as many TRD slides can be re-used or adapted.

Development Programmers assist in updating the status of the project risks and associated risk mitigation actions for inclusion in the CTD. Risk management guidelines can be found in PG-1.

5.8.5.5 Conduct CTR

The CTR consists of the presentation of the pre-operational code, test data, and test plan by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the CTR to determine whether the refined pre-operational code has satisfied unit test criteria and is ready for integration into a pre-operational product processing system. Reviewers should be familiar with the CTR guidelines (PRG-10) and check list (CL-10).

Each stakeholder who performed activities during step 10 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

The **Development Lead** should remind the stakeholders to do this. At the conclusion of Development (step 11), the **Development Lead** will collect the final edited personal stakeholder records and incorporate them into a Development Project Report (DPR).

5.9 System Integration and Test Tasks

Figure 5.19 shows the process flow for step 11.

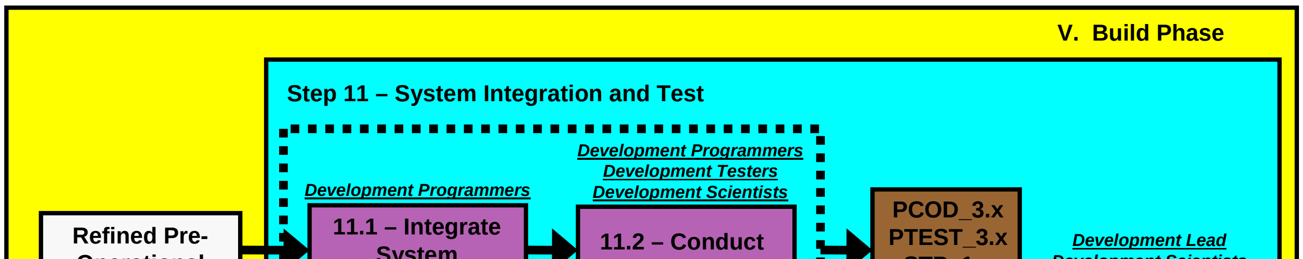

Figure 5.19 – Step 11 Process Flow

Note that processes 11.1, 11.2, and 11.3 are enclosed by a common dashed border. This is to indicate that the processes are iterative, as explained in Section 5.6.

5.9.1 Expected BEGIN State

- REQUIRED: Pre-operational code has been refined and debugged as necessary until it passes all unit tests.

- REQUIRED: Unit test results have been documented in a report.
- REQUIRED: A plan for system testing has been developed. The plan ensures that the system test will address all system requirements and product requirements.
- REQUIRED: All data required for implementation of the system test plan has been acquired or developed, and is available in the designated test environment.
- REQUIRED: A CTR has been conducted
- REQUIRED: CTR reviewers have approved the project to proceed to the System Integration and Test step, and have documented this approval in the Code Test Review Report (CTRR).
- REQUIRED: Baseline Build (BB) 3.3 has placed the following items in the project artifact repository:
    - Refined pre-operational code
    - System test data
    - DPP, including Appendices
    - RAD, including Appendices
    - VVP
    - ATBD
    - SWA
    - DDD
    - UTP
    - UTR
    - STP
    - Code Test Document (CTD)
    - CTRR
- EXPECTED: BB 3.3 has placed the following items in the project artifact repository:
    - R&D code
    - R&D test data
    - Project Proposal (PP)
    - Gate 2 Review Report (G2RR)
    - Gate 3 Review Report (G3RR)
    - Operations Concept Document (OCD)
    - Project Requirements Document (PRD)
    - Project Requirements Review Report (PRRR)

  - Preliminary Design Document (PDD)
  - Preliminary Design Review Report (PDRR)
  - Critical Design Document (CDD)
  - Critical Design Review Report (CDRR)
  - Gate 4 Document (G4D)
  - Gate 4 Review Report (G4RR)
  - Test Readiness Document (TRD)
  - Test Readiness Review Report (TRRR)
  - Project Status Report (PSR), including Appendix
- REQUIRED: PBR_3.3 documents the status of the BB 3.3 project baseline

5.9.2 Task Inputs

Task inputs consist of the following BB 3.3 items:

- Refined pre-operational code (PCOD_2.x)
- Refined pre-operational test data (PTEST_2.x)
- SWA_2.3
- DDD_1.2
- UTP_1.1
- UTR_1.0
- STP_1.0
- DPP_3.x
- RAD_1.4
- Requirements/Needs Matrix (RNM)
- Requirements Allocation Sheet (RAS)
- VVP_1.4
- CTD
- CTRR
- PSR_2.x Appendix

- PBR_3.3

5.9.3 Desired END State

- The Detailed Design Allocation of the requirements that identifies product and system components down to the Sub-Unit-Layer, and traces each component to one or more requirement, has been verified.
- The functionality of all system components in the detailed design (software units and sub-units) has been implemented in pre-operational code that meets coding standards.
- Unit testing of the code has ensured that all required code functionality and code outputs have been satisfied.
- The code and system test data have been integrated into a complete pre-operational product processing system.
- The pre-operational system has been refined and debugged as necessary until it satisfies all system requirements and product requirements, as determined by system testing.
- System test results have been documented in a report.
- All required documentation has been produced.
- The project plan has been updated as necessary
- Project status, including project risks and actions, has been updated
- An SRR of the project plan, system test results, and supporting documentation has been conducted
- An SRRR has been written. The SRRR approves the readiness of the product processing system and supporting documentation to be delivered to operations.
- A Gate 5 Review of project status has been conducted.
- A Gate 5 Review Report (G5RR) has been written. The G5RR approves the project for transition to operations.
- Baseline Build 3.6 has placed the required items in the project artifact repository
- PBR_3.6 documents the status of the BB 3.6 project baseline

5.9.4 Task Outputs

Task outputs consist of the following BB 3.6 items:

- Integrated Pre-Operational Code
- System Test Data
- DPP_3.x
- ATBD_2.2
- STP_1.1
- IUM_1.0
- EUM_1.0
- MDD_1.0
- SRD
- SRRR
- PSR_3.0
- G5D
- G5RR
- PBR_3.6

5.9.5 Stakeholder Activities

Step 11 activities include:

1) Integrate system components
2) Conduct system test
3) Refine system
4) Prepare for SRR
5) Conduct SRR
6) Prepare for Gate 5 Review
7) Conduct Gate 5 Review

5.9.5.1 Integrate System Components

The pre-operational system is produced by integrating the system components that have passed unit testing into a complete end-to-end product processing system. This should be done in the system test environment that was designated in the system test plan.

Development Programmers prepare the system test environment, in accordance with the system test plan. The system test environment should replicate the planned operational environment as closely as possible.

Development Programmers should perform the system integration. This is typically done by creating scripts that link the units in an order that follows the software architecture. This is usually a straightforward process, since each individual unit has already been built into its unit test configuration. System integration then entails the linking of the unit test configurations by scripts that ensure the proper interfaces between the units and with external data sources and data sinks. The composition of the integrated pre-operational system should include all control files, scripts, pre-operational code modules, test data sets, utilities, and libraries needed to run the product processing system

Note that the system integration function is iterative with system testing and refinement. If system testing uncovers problems that require refinement of the code, test data, and/or scripts, these will have to be re-integrated and re-tested.

5.9.5.2 Conduct System Test

Development Programmers build the system test configuration, in accordance with the system test plan. The system test configuration includes all items that will be used in the system test, including control files, scripts, code modules, test data sets, utilities, libraries, etc. If the full functionality of the system will be tested in the system test, the system test configuration will consist of the entire integrated pre-operational system (Section 6.5.1).

If there is a specific reason not to test the full functionality of the system in the system test, **Development Programmers** will build a system test configuration that is a subset of the complete system.

It is recommended that diagnostic code be used to facilitate the verification of functionality and outputs. Diagnostic code may be added to control files and scripts to verify their functionality. Diagnostic code will typically be added to the unit code during step 9. If this is done, two versions of the unit code will have been created. The first version will contain the diagnostic code. The second version, with all diagnostic code stripped out, is the baseline integrated pre-operational system.

In that case, the system test configuration should include both versions. The version with diagnostic code should be run first, with all diagnostic messages and outputs captured for verification. This provides a complete verification of functionality and performance. Then, the baseline version should be run. Verification of the baseline system is then achieved by confirming that its outputs are identical to the outputs from the version with diagnostic code. The test sequence should include this final verification step.

Development Testers run the system test, assisted by **Development Programmers**. **Development Scientists** assist in evaluating the system test results. Examine the system test output, including runtime messages, diagnostic messages and the content of intermediate and output data files.

Runtime messages are messages written by the operating system to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). These may occur if the code is written to generate such messages as a way to test functionality.

Diagnostic messages are messages written to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). The nominal purpose of a diagnostic message is to report a functional result (e.g., 'subroutine X called') or the quantitative value of an input, intermediate, or output variable (e.g., 'X(50) = 7').

Data files include the output data sets that are designed to be produced by the system. In addition, a diagnostic program may write intermediate data sets to diagnostic files.

If the system test output does not satisfy all success criteria, iteratively refine and debug the code, test data, and/or scripts (c.f. Section 6.5.3), and repeat system integration (Section 6.5.1) and system testing until success criteria are satisfied. When success criteria are satisfied, document the results in the VVR, following guidelines in DG-11.4. **Development Scientists** assist **Development Testers** in the development of the VVR. The purpose of VVR v1r0 is to document the results of system testing to ensure that the requirements specified for the product processing system are satisfied by the completed system (Verification) and that the final developed system will satisfy the users' needs and expectations (Validation).

5.9.5.3 Refine System

Development Programmers iteratively refine, debug and re-test the integrated pre-operational system as needed, based on the system test results. Code refinements should conform to coding standards provided in TD-11.1 (FORTRAN code) and TD-11.2 (C code).

Development Programmers assist **Development Testers** in refining the system test data as necessary until the system test requirements are satisfied. The STP is revised to account for changes to the system test plan since the CTR. STP v1r1 updates and refines the test data description, test methods, test sequences and test risks, based on any refinement of the code and test data that has occurred during step 11.

Note that the refine system function is iterative with system integration (Section 6.5.1) and system testing (Section 6.5.2). If system testing uncovers problems that require refinement of the code, test data, and/or scripts, these will have to be re-integrated and re-tested until all system test requirements are satisfied. At that point, refinement of the system is completed.

5.9.5.4 Prepare for SRR

The SRR slide package is the System Readiness Document (SRD). The SRD is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with SRD guidelines DG-11.5. DG-11.5.A provides SRD slide templates that can be adapted for the project's SRD. The SRD developers should examine the DPP to determine whether the SRR objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the SRD slide templates must be adapted to accommodate the tailoring. The SRD developers should use the project's CTD as a source for SRD slides, as many CTD slides can be re-used or adapted.

Development Programmers assist in updating the status of the project risks and associated risk mitigation actions for inclusion in the SRD. Risk management guidelines can be found in PG-1.

5.9.5.5 Conduct SRR

The SRR consists of the presentation of the integrated pre-operational product processing system and supporting documentation by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**Technical Review Lead** and **Technical Reviewers**).

The **Technical Review Lead** and the **Technical Reviewers** conduct the SRR to determine whether the integrated pre-operational system has satisfied system test success criteria and is ready for delivery to operations. Reviewers should be familiar with the SRR guidelines (PRG-11.1) and check list (CL-11.1).

5.9.5.6 Prepare Gate 5 Review

Once the project passes its SRR, it is referred to the Gate 5 Review, the final STAR review prior to delivery of the pre-operational system to operations. The purpose of the Gate 5 Review is to ensure **STAR Management** approval of the project status prior to delivery.

Development Programmers assist the **Development Lead** in updating the PSR to version 3. Version 3 of the PSR, along with its Appendix, documents the status of project tasks, cost, schedule, risks, and actions at the conclusion of the Build phase. Refer to PSR guidelines in DG-5.2 and DG-5.2.A.

The **Development Lead** leads the preparation of the Gate 5 Review presentation. The presentation slide package is the Gate 5 Document (G5D). The G5D is prepared by the **Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers,** in accordance with G5D guidelines DG-11.7. DG-11.7.A provides G5D slide templates that can be adapted for the project's G5D. The G5D developers should examine the DPP to determine whether the Gate 5 Review objectives, entry criteria, exit criteria and/or CLI have been tailored. If so, the G5D slide templates must be adapted to accommodate the tailoring.

5.9.5.7 Conduct Gate 5 Review

The “System Integration and Test” step culminates with a Gate 5 Review. The Gate 5 Review consists of the presentation of the project plan and project status at the conclusion of the Build phase by the development team (**Development Lead**, **Development Scientists, Development Testers,** and **Development Programmers)** and the disposition of the review CLI, including entry and exit criteria, by the reviewers (**STAR Management**).

Each stakeholder who performed activities during step 11 is encouraged to document an assessment of the experience in a personal record. This assessment should include: what was good, what was bad, what worked, what did not work, what can be improved, how it can be improved.

The **Development Lead** should remind the stakeholders to do this. At the conclusion of Development (step 11), the **Development Lead** will collect the final edited personal stakeholder records and incorporate them into a Development Project Report (DPR).

____

END OF DOCUMENT