

222-TP-010-001

Release B IDR Design Development Plan

Technical Paper

August 1995

*Technical paper - Not intended for formal review
or Government approval.*

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

<u>David Turanski /s/</u>	<u>10/19/95</u>
David Turanski, Architect - Release B EOSDIS Core System Project	Date

SUBMITTED BY

<u>Craig Schillhahn /s/</u>	<u>10/19/95</u>
Craig Schillhahn, Release B Manager EOSDIS Core System Project	Date

Hughes Information Technology Corporation
Upper Marlboro, Maryland

This page intentionally left blank.

Abstract

This document presents the design development plan for the preliminary design of ECS Release B, in preparation for the Release B Incremental Design Review (IDR). It includes a discussion of the Release B preliminary design approach and success criteria, the application of object-oriented methodology, and requirements tracing.

Keywords: IDR, design, OMT, RTM, object-oriented, scenario

This page intentionally left blank.

Contents

Abstract

1. Introduction

1.1	Purpose	1
1.2	Summary of Design Approach	1
1.3	Overview of IDR Deliverables	2

2. Design Methodology

2.1	Object Modeling Technique (OMT)	4
2.1.1	Static Model	4
2.1.2	Dynamic Model	5
2.1.3	Functional Model	5
2.2	Delta Design Model	5
2.3	CSCI Interfaces	6
2.4	Use Case Modeling	6
2.5	Scenario Development	7
2.5.1	Key System Activities and Scenarios	7
2.5.2	CSCI Interface Scenarios	7
2.5.3	Scenario Primitives and Key Mechanisms	10
2.5.4	CSCI Internal Scenarios	11

3. Design Repository

3.1	OMT Tool	12
3.1.1	Repository Organization	12
3.1.2	Access Control	13
3.2	General Procedures	13

3.2.1	Revision Control	13
3.2.2	Document Generation	13

4. Requirements Traceability

4.1	Requirements Mapping Tables	14
-----	-----------------------------------	----

5. Release Coordination

5.1	Release A Baseline	16
5.2	Design Synchronization	16

6. Document Development

6.1	Design Overview (Vol. 0 305) Development	17
6.2	DID 313 Development	17
6.3	Data Dictionary Development	17
6.4	DID 305 Development	17

Figures

1-1.	Release B Organization	3
2-1.	Sample Event Trace Diagram showing CSCI Interfaces for the Data Production Scenario	10
2-2.	Sample Scenario Primitive Event Trace	11

Tables

2-1.	CSCI Interface Scenarios	8
------	--------------------------------	---

Appendix A. IDR Design Checklist

1. Introduction

1.1 Purpose

This paper discusses the design approach and plan for the Release B Team to develop the necessary design products for the Release B Incremental Design Review (IDR-B). This plan was prepared by the Release B Architect and the Release B Chief Engineering Group (CEG-B) as part of the overall Release B IDR Engineering Plan, and is referenced in the document entitled Release B IDR Engineering Plan dated April, 1995.

This Technical Paper is an informal document approved at the Release Manager level. It does not require formal Government review or approval. Questions regarding technical information contained within this Paper should be addressed to David Turanski, (301)925-1045, dturansk@eos.hitc.com.

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Corporation
1616 McCormick Dr.
Upper Marlboro, MD 20774

1.2 Summary of Design Approach

The main objectives of the IDR design are to:

- Ensure that ECS Release B meets all Level 4 requirements
- Reuse as much of the Release A design as possible in Release B
- Focus on incremental development (i.e., new or modified in Release B)
- Adhere to fundamental software engineering principles
- Effectively communicate system design to the user community and system developers
- Complete required contract deliverables

In order to accomplish these objectives, a design approach has been adapted from two well-accepted object-oriented methodologies: The Object Modeling Technique (OMT) and Use Case Modeling. These design methodologies are presented in *Object-Oriented Modeling and Design*, Rumbaugh, et al., 1991 and *Object-Oriented Software Engineering (A Use Case Driven Approach)*, Ivar Jacobson, 1993.

The object-oriented CASE tool, OMT/StP, in conjunction with the RTM (Requirement Traceability and Management) tool, will assist in automating the design process. The OMT design repository is discussed in Section 3.

The key components of the IDR design are:

- Scenarios
- Delta Design Model
- Requirements Traceability

This approach focuses on developing a set of scenarios, or use cases, which describe normal system operation from the user's point of view. Each scenario is comprised of a logical sequence of events that depict a specific aspect of a user's interaction with the system. Object models can be developed independently of scenarios, but must be shown to implement the system behavior described by the scenarios. This is accomplished by developing event trace diagrams which depict the sequence of interactions (i.e., messages or operations) between objects necessary to implement individual scenario steps.

The scenarios used to drive IDR design will be taken from those developed for the ECS Release B Operations Concept document (DID 604) (operations concept scenarios). Thus, there will be a natural progression from the Key System Activities covered in the 604 document to the object interactions shown in the system design. Additional scenarios may be developed if necessary to ensure coverage of all CSCI interfaces, and key subsystem interactions. Section 2.5 covers this topic in more detail.

In tandem with scenario development, an delta design model will be developed using OMT for each CSCI. The delta model will be derived from the Release A CDR design. The expectation is that a significant part of the Release A design can be serve as a baseline for Release B. The delta model will focus on modifications to the Release A baseline rather than the cumulative design for Release B. To accomplish this, we will use a Release B naming convention to make a clear delineation between the Release A baseline and the Release B delta design. The delta design model will be the focus of the IDR presentation, however the design documents will include CDR level detail of the Release A baseline. Section 2.4 and Section 5 cover this topic in more detail.

Requirement tracing will be done by creating a mapping between Level 4 requirements and the most appropriate design component(CSCI, CSC, object, HWCI, HW component). The main focus here will also be incremental, since this mapping has already been developed for the Release A baseline. Since there is a many-to-many relationship between components and L4 requirements, two cross-reference tables will be created. One will show Level 4 requirements by component name. The other will show component names by Level 4 requirement.

1.3 Overview of IDR Deliverables

The IDR B system design will be presented in a series of documents, as specified in the CDRL (Contract Data Requirements List) and presented at IDR, October 31- November 3,1995. Each Release B development team will be responsible for developing a DID 305 design document for each of its subsystems. The Release B Chief Engineering Group (CEG-B) will develop an ECS Design Overview document (305 Volume 0), the Data Dictionary (DID 305), and the Internal Interface Control Document (ICD DID 313).

Figure 1-1 shows the Release B organizational structure. The Architect is responsible for coordinating the IDR design effort. CEG-B coordinates all IDR Release B activities, including requirements analysis, operations concept development, and design development. The complete list of CDRLs required for IDR-B, including delivery dates for each document is maintained by CEG-B. Each document will be subject to the following reviews and milestones:

- Internal Review
- ECS CCB Review
- Delivery to DMO
- Delivery to NASA

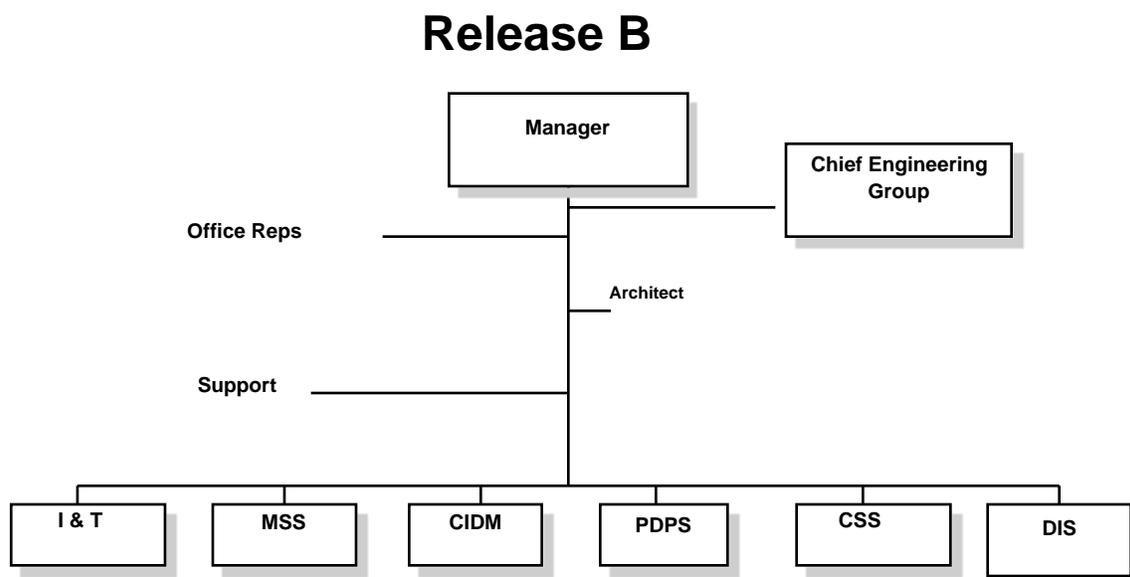


Figure 1-1. Release B Organization

2. Design Methodology

2.1 Object Modeling Technique (OMT)

Object Modeling Technique (OMT) is a software design methodology first presented in Object-Oriented Modeling and Design, Rumbaugh, et al., 1991. ECS has selected OMT as its primary design methodology. OMT offers three design models that together form a complete view of a system. These are the *static model*, in which objects, attributes, and operations are defined; the *dynamic model* represented by event trace diagrams (ETDs), and state-transition diagrams (STDs); and the *functional model*, represented by data flow diagrams (DFDs) to show data flow between processes.

Since publishing this work, Rumbaugh has made several modifications and enhancements to OMT. New developments often appear in Rumbaugh's column in the periodical, Journal of Object Oriented Programming (JOOP). Most notably, Rumbaugh has stated his support for incorporating Use Case Modeling, as presented in Object-Oriented Software Engineering (A Use Case Driven Approach), Ivar Jacobson, 1993.

The works of Rumbaugh and Jacobson provide a formal basis for the design approach adopted for Release B. The system design will be modeled using OMT, and will be driven by a well chosen set of use cases (scenarios). The emphasis on scenarios has several advantages. Each scenario starts with a high level description of some aspect of the system that is readily accessible to both the user community and system developers. Selected scenarios will be developed to a level of detail sufficient to show specific object interaction. This will provide a good basis for requirements definition and traceability, the discovery of reusable design components, and object model validation.

2.1.1 Static Model

The object model describes the static structure of objects in a system and is represented by an *object diagram*. The object diagram defines classes and relationships (associations) between classes. Each class contains a set of attributes and operations. For IDR B, the object model will be largely complete. Each development group will develop object diagrams for all CSCIs in its domain. Each object model will be developed in OMT and will satisfy the following requirements:

- All objects will be identified.*
- All attributes and operations will be identified.*
- Naming of objects, attributes, and operations must comply with standards defined in the ECS Development Plan.
- All associations, aggregation hierarchies, and generalization hierarchies will be defined.
- All CSCI interfaces will be implemented as distributed objects.

- Release B modifications will be implemented via inheritance from Release A classes where possible.

* *In this case "all" is defined in the context of Level 4 requirements. Objects should be identified and defined to the level of detail necessary to ensure that all Level 4 requirements are satisfied.*

2.1.2 Dynamic Model

The dynamic model describes the aspects of the system that change over time and is used to specify control aspects of a system. The OMT dynamic model employs *state-transition diagrams* and *event trace diagrams*. IDR B will emphasize event trace diagrams, and use state-transition diagrams only where complexity dictates. Scenarios can be easily represented as event trace diagrams. At the lowest level of detail, event traces show a logical ordered sequence of interactions (i.e., messages or operations) between objects. At higher levels, event traces can be used to model interfaces between CSCIs, or interactions between users and subsystems. This topic is covered in more detail in sections 2.4 and 2.5.

2.1.3 Functional Model

The functional model describes the data value transformations within a system. In OMT, the functional model contains *data flow diagrams*. A data flow diagram represents a computation or process model. In addition, this information is provided at varying levels of detail by textual descriptions, operation signatures (showing operation calling arguments and return values), and program design language (PDL).

Since the emphasis placed on the functional model is relatively small compared to traditional A&D methodologies, and since the level of detail provided in the functional model is beyond the scope of preliminary design, no formal functional model will be presented at IDR B. However, textual descriptions of operations will be provided, and data contents will be itemized as part of the level 4 CSCI interface requirements.

2.2 Delta Design Model

It is extremely important to separate the Release A baseline from the incremental Release B design in the IDR design model (Note: the term *incremental* in this context refers to additions and modifications required for Release B functionality and should not be confused with ECS incremental track development). This will help to effectively manage the Release B work flow, determine level of effort, maximize reusability, and facilitate requirements traceability. The delta design model will satisfy this objective.

The delta model will use a Release B naming convention to identify design changes that are needed to satisfy Release B requirements. The naming convention consists of appending the letter **B** to each new or modified attribute or operation within existing Release A classes. In addition, new classes defined for Release B will follow the same naming convention. This approach assumes that very little will be subtracted from the Release A models. Since subtraction incurs little or no development cost, it is not critical to show it in the delta model.

It is important to note that the inheritance technique used in the delta model will not be used for implementation. Prior to CDR-B the Release A baseline will be entirely maintained by Release B. At this point, the above naming convention will be dropped and Release B extensions will be directly incorporated into the Release A baseline classes to form a Release B baseline.

2.3 CSCI Interfaces

An extremely important aspect of the ECS system design is to define subsystem interfaces at the CSCI level. Each CSCI must provide a public interface through which its services must be accessed. The principles of encapsulation and weak coupling dictate the public interface should be defined only by the services provided by the CSCI without any a priori knowledge of the users of that service (client). In this way the design of each CSCI is completely independent of the design of its clients.

On the other hand, each client CSCI that invokes the services of another CSCI must specify an interface through which those services will be accessed. This requires knowledge of the server CSCI interface. Each client CSCI must specify a set of interface objects which will interact with external CSCIs. The specification of the interface should include the class names and associated operations, but not operation signatures.

CSCI interfaces consist of a set of distributed objects representing the server CSCI's public interface, and the client side objects which use the public interface. In general, the client side objects are internal to the client CSCI. For IDR B CSCI public interfaces will be illustrated by selected operations concept scenarios and will be represented as event trace diagrams in the Internal ICD (DID 313). The client side of the interface, specific to each CSCI will be documented in the corresponding subsystem design document (DID 305).

An addition to inter-CSCI interfaces, ECS-external interfaces will be implemented as distributed objects. These external interface classes will be presented in the Internal ICD and will be incorporated into the CSCI interface scenarios.

2.4 Use Case Modeling

The terms "use case" and "scenario" are often interchanged, but actually represent slightly different concepts. Rumbaugh defines *scenario* as "a sequence of events that occurs during one particular execution of a system." Jacobson's use case model does not make reference to scenarios, but extends the use case concept to enable generalization hierarchies which support reusability and encapsulation. Thus, one use case may "extend" or "use" another. In some sense, the relationship between "use case" and "scenario" can be thought of as a class - instance relationship where a scenario is a specific instance of a use case; and a use case represents a generalization of functionally related scenarios. To be consistent with the current ECS semantics, no formal distinction will be made between the terms "use case" and "scenario".

Use cases describe system behavior from the point of view of the user. Users are represented as *actors*. Actors are external to the system and interact with the system by sending and receiving events. A use case starts with an actor sends an event to the system. Actors are normally thought of as users, but may also represent external systems, or any object capable of sending an

event to the system. Interactions between objects within the system also constitute events. Object interactions are implemented by one object invoking another object's operation. Therefore, use cases can be decomposed, through successive iterations, to the detailed design level.

2.5 Scenario Development

The role of scenarios in IDR B design is based on Jacobson's use case driven approach. The advantage of this approach, is that use cases provide a natural way for users and developers to communicate and validate functional requirements. The generic process is summarized as follows:

- Identify a set of scenarios that sufficiently cover normal system operations
- Provide a high level text description for each scenario
- Define each scenario as a sequence of enumerated steps
- Map key scenario steps to event trace diagrams

The implementation of this process for IDR-B is the topic of this section.

2.5.1 Key System Activities and Scenarios

The Operations Concept Document 604 Part 2B developed for IDR B will provide scenarios based on Key System Activities which span a wide range of normal ECS operations. The operations concept scenarios will be described in narrative form in the 604 document. In addition the 604 will trace each scenario to a set of Release B Level 3 requirements. These scenarios will provide an excellent starting point for system design. Additional scenarios will be provided in the Internal ICD 313, if the 604 scenarios do not sufficiently cover CSCI Interfaces.

2.5.2 CSCI Interface Scenarios

Table 2-1 contains the CSCI interface scenarios identified so far. They include scenarios from the Operations Concept plus additional scenarios which illustrate key CSCI interfaces new or modified in Release B. Some or all of these will be included in the Internal ICD 313.

Table 2-1. CSCI Interface Scenarios

313 Paragraph	Ops Con Paragraph	Scenario Name
	4.1.7.7	Operational System and Test Activity Run in Parallel Scenario
	4.1.7.8	New Advertising Service Transitioning in Operational SDPS Scenario
4.1.2	N/A	Simple User Access Modified For Rel. B
4.1.3	N/A	Standard Production Modified For Rel. B
4.1.4	TBD	Higher Level Product Ingest Scenario (L1A - L3 from TSDIS)
4.1.5	4.2.1.9	L7 Suspend and Resume Scenario
4.1.6	4.2.3.1	Network Data Distribution Push Scenario
4.1.7	N/A	Data Acquisition Request (DAR) Scenario
4.1.8	N/A	User Pull from External Source (CSA)
4.1.9	4.2.4.3	On-Demand Request Scenario
4.1.10	4.2.8.7	Cross DAAC Search Scenario
4.1.11	4.2.8.8	Cross DAAC Coincident Search Scenario
4.1.12	N/A	Complex Subsetting
4.1.13	4.2.4.4	Planning Reprocessing Scenario
4.1.14	4.2.4.5	Replan to add Hot Job Scenario
4.1.15	4.2.5.3	PGE Involving Inter-DAAC Processing Scenario
	4.2.8.4	Metadata Produced from Data Production and Ingest Scenario
	N/A	User Services Placing Order on Behalf of User

Once the CSCI interface scenarios have been finalized and described in narrative form, the next step is to decompose each scenario such that each step consists of a logical CSCI interaction. This could be either an interface between to CSCIs, an external interface to a CSCI, or a user action. These scenarios will be represented in narrative and tabular form. In addition, they will be represented as event trace diagrams.

The following is an example of a CSCI scenario in tabular form based on a preliminary draft of the Release A 313 ICD. It is shown here to illustrate the table structure. The contents have not been verified. The Primitive References column contains a reference to a *scenario primitive* (SP). Each scenario primitive describes a particular interface at the class interface level.

Example CSCI Interface Scenario: Data Production

Preconditions:

In order to notify science teams whenever there is a new plan, PLANG will provide a script to send e-mail to a pre-prepared list of addressees when a new plan is published. This scenario assumes the list has already been created (see step 3).

Step #	Ename	Primitive References	Interface Client	Interface Server	Description
1	Activate Subscriptions	N/A	OPER	PLANG	Operations activates subscriptions to enable notification to PLANG of data arrival.
2	Activate Candidate Plan	SP 25	PLANG	PRONG	Planning activates a candidate processing plan
3	Publish Plan	SP 46	PLANG	DDSRV	PLANG creates a metadata file describing the plan. This file is referenced by a hyper link in an HTML document. The script is run to send e-mail to a pre prepared list of addressees (See preconditions).
4	Receive Subscription Notification	SP 8	PLANG	SDSRV	SDSRV notifies PLANG that required L0 data is available
5	Release DPR Job	SP 26	PLANG	PRONG	PLANG releases DPR to start execution of job stream
6	Acquire L0 Data for Processing	SP 11	PRONG	SDSRV	Processing requests SDSRV to retrieve L0 data from archives. SDSRV retrieves L0 data from archive. PRONG receives confirmation
7	Acquire Other Data for Processing	SP 11	PRONG	SDSRV	Processing requests SDSRV to retrieve non-L0 data from archives. SDSRV retrieves non-L0 data from archive. PRONG receives confirmation
8	Run PGE Job	COTS	PLANG	PRONG	PGE is a component of the job stream managed by the PLANG COTS.
9	Archive Outputs	SP 12	PRONG	SDSRV	PGE output data is stored in SDSRV archive. PRONG receives confirmation.

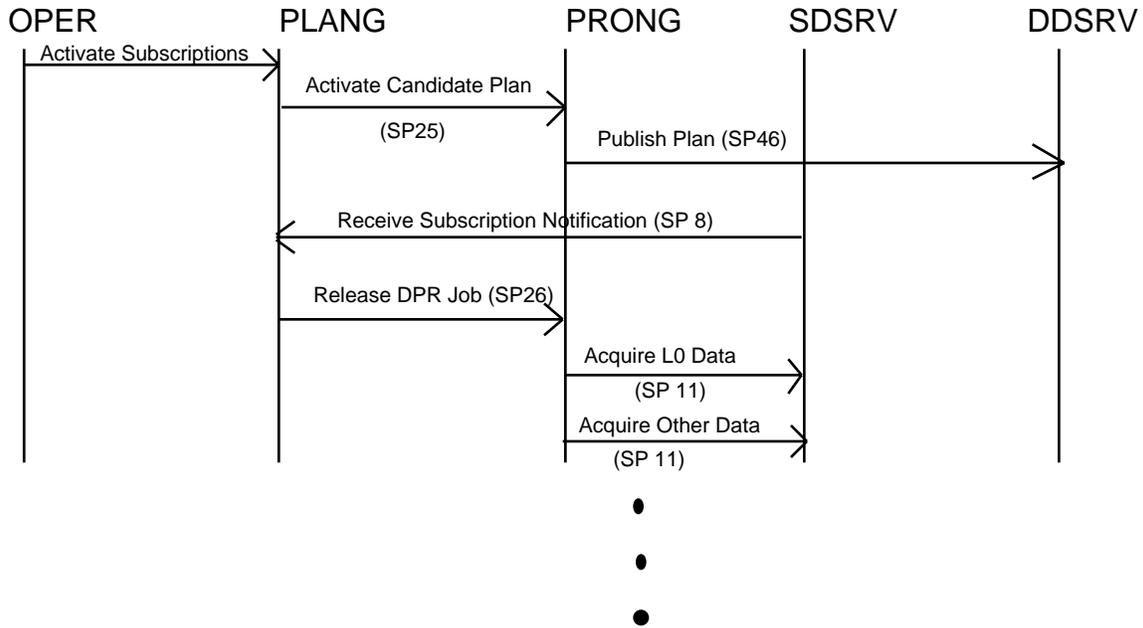


Figure 2-1. Sample Event Trace Diagram showing CSCI Interfaces for the Data Production Scenario

2.5.3 Scenario Primitives and Key Mechanisms

Scenario primitives are used to illustrate the use of distributed objects to implement a CSCI interface within ECS. Each scenario primitive constitutes a reusable design component. Scenario primitives focus on the public interfaces provided by CSCIs and not the internal workings of CSCIs. Scenario primitives will be included in the 313 ICD, but will be developed by the design team responsible for the CSCI involved.

Key Mechanisms are similar to Scenario Primitives in that they also represent reusable design components, but are considered part of the general software infrastructure and are more widely used. In ECS, key mechanisms include all CSS interfaces (e.g., asynchronous distributed objects, message passing); a significant portion of MSS interfaces (e.g., event logging, fault handling, and user profiles); global utility objects (e.g., parameter lists, callbacks, universal references); and widely used Data Server interfaces. Key Mechanisms will be included in the 313 ICD, but will be developed by the design team responsible for the CSCI involved.

The CSCI interfaces provide a context for scenario primitives and key mechanisms. However, these interfaces are designed as reusable software components. As such, they should be context independent. In other words, any client object may invoke the services provided by a public interface or key mechanism. This is shown in the event trace diagram by designating the client as "Calling Object." This is illustrated in Figure 2.2.

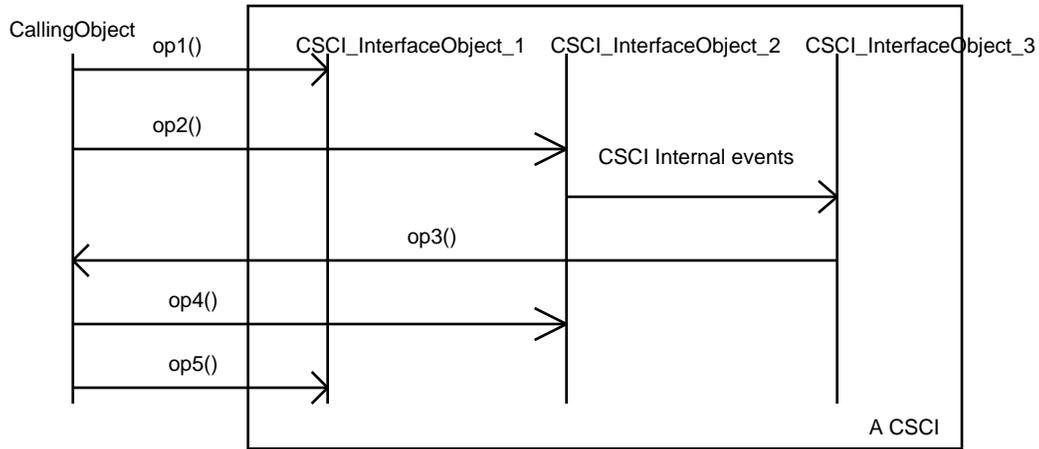


Figure 2-2. Sample Scenario Primitive Event Trace

2.5.4 CSCI Internal Scenarios

CSCI internal scenarios illustrate the object interfaces internal to the CSCI. These will be developed by the development team responsible for the CSCI design. Event trace diagrams and text descriptions will be included in the subsystem 305 documents, and will be provided to complete the scenario primitives and key mechanisms where necessary. Additional internal scenarios may be required to provide a complete dynamic model for each CSCI. A certain amount of overlap between the internal scenarios and the scenario primitives and key mechanisms is expected.

3. Design Repository

3.1 OMT Tool

OMT/StP (Object Modeling Tool/ Software Through Pictures) is a object-oriented CASE tool for the UNIX platform that supports the OMT methodology and basic Use Case Modeling. OMT includes a series of graphical editors and a central repository for developing and maintaining use case models and event trace diagrams, object models, DFDs, and STDs. In addition, various annotation capabilities are included. OMT also includes a scripting language, QRL, which allows for the development of custom applications and interfaces to other tools such as RTM (Requirement Traceability and Management).

A number of QRL scripts were developed for Release A to support automated document generation. Release B will use these scripts, and make modifications as needed. Many of these scripts generate document text from diagram annotations. Therefore, it is important that certain annotation standards be followed to ensure consistency.

3.1.1 Repository Organization

The OMT repositories are organized as a single *project* with a separate *system* for each CSCI , and an *interfaces* repository, as shown:

Project(projdir): ReleaseB

- Systems:
 - CLS
 - CSS
 - DMS
 - DPS
 - DSS-DDIST (Data Distribution)
 - DSS-DDSRV (Document Data Server)
 - DSS-SDSRV (Science Data Server)
 - DSS-STMGT (Storage Management)
 - INS
 - interfaces
 - IOS
 - MSS
 - PLS

This organization was chosen to improve overall performance and management, since smaller repositories are easier to work with than large repositories. These repositories are initially populated with models included in the Release A baseline.

3.1.2 Access Control

All Release B developers will have read access to all repositories. Each development team will have read/write access to its CSCI repositories. The Release B Architect will have read/write access to all repositories. Users will be added as requested.

3.2 General Procedures

The IDR Design model will be created in OMT by taking a copy of the Release A CDR models (Release A baseline) and modifying them to create the delta design model, described in Section 2. A set of guidelines and standards will be developed to ensure consistency in the way information is presented in all repositories, and documents. In addition, a process to facilitate requirements traceability will be developed.

3.2.1 Revision Control

Baseline designs will be created at major project milestones (IDR, CDR, etc.), and will be placed under CM. Release B is currently evaluating two options for OMT CM. The CM solution will be chosen based on design CM requirements (versioning, history, baselines) and the effort required to implement each.

Option A

The OMT design will be placed under CM using StP's built-in version control system. The StP versioning system provides an interface to SCCS, with minimal CM functionality.

Option B

The StP-OMT design will be placed under CM using ClearCase. ClearCase is the ECS standard CM tool for software and provides a full set of CM capabilities.

3.2.2 Document Generation

Release B will follow the document production process based on the one developed for Release A. This includes utilization of the scripts, and annotation standards developed by Release A. These scripts will be tailored as necessary. The process will be verified and refined for Release B, with assistance from the OMT vendor (IDE). The output will be generated as FrameMaker documents.

4. Requirements Traceability

4.1 Requirements Mapping Tables

Level 4 requirements will be traced to the most appropriate design component (CI, CSC, object, HWCI, HW component). For functional requirements, this is typically at the level of an object, and occasionally at the level of an operation. Where the requirement testability is at a higher level (e.g., CSCI, CSC), and a trace to the class level would not support testability, the trace will be to the higher level where testing will be defined. For example, performance requirements may often fall into this category. The trace will be included in DID 305.

In general, there is a many-to-many relationship between Level 4 requirements and components. The expectation is that any modifications or enhancements to Release B must be driven by one or more L4 requirements. Therefore, any new or modified classes in IDR-B can be mapped to some L4 requirements.

The L4 requirements are entered and maintained in the RTM (Requirement Traceability and Management) repository. The requirements will be linked to components in the RTM database.

The primary purpose of mapping components to L4 requirements and vice versa is to verify that there are no unsatisfied requirements, and that every component satisfies some requirement. This will be accomplished by generating two cross-reference tables. One table will list L4 requirements by component name. The other table will list component names by L4 requirement. They will appear as appendices in the subsystem 305 documents.

The column headers of the requirements mapping tables are as follows:

Mapping of Components to Level 4 Requirements

Component name

The component (CSCI, CSC, class, HWCI, HWC) name as it appears in the RTM database.

DID 305 Paragraph

The number of the 305 paragraph in which the component is referenced.

DID 305 Paragraph Title

The title of the 305 paragraph in which the component is referenced.

DID 304 Requirement

The L4 requirement ID.

DID 304 Paragraph

The number of the 304 paragraph in which the requirement is referenced.

DID 304 Paragraph Title

The title of the 304 paragraph in which the requirement is referenced.

Mapping of Level 4 Requirements to Components

DID 304 Requirement

The L4 requirement ID.

DID 304 Paragraph

The number of the 304 paragraph in which the requirement is referenced.

DID 304 Paragraph Title

The title of the 304 paragraph in which the requirement is referenced.

Component name

The component name as it appears in the RTM database.

DID 305 Paragraph

The number of the 305 paragraph in which the component is referenced.

DID 305 Paragraph Title

The title of the 305 paragraph in which the component is referenced.

5. Release Coordination

5.1 Release A Baseline

ECS Release B will be an extension of ECS Release A. The Release B design strategy, therefore, is to build onto Release A design. The expectation is that Release A will provide an ECS framework to which Release B will add new applications. In some cases, enhancements to the framework are required as well (e.g., The Data Management Subsystem).

One of the first IDR design tasks (or perhaps the final analysis task) is to conduct a detailed analysis of the Release A design to determine what portions can be reused for Release B. This analysis should be conducted by the Release B development groups and should be done at the class level for each CSCI. This will define the Release A Baseline. The baseline is expected to include a large percentage of the Release A design. Release B design will be created by modifying and enhancing the baseline to meet the Release B requirements.

5.2 Design Synchronization

The delta design model mitigates the need for multiple synchronization points to update the Release A baseline prior to IDR. However some degree of synchronization will be required since The IDR design documentation will include detailed design for the Release A baseline.

IDR will require at least two synch points. The first synch point will be to initially capture the Release A baseline. In addition, any updates to the baseline prior to IDR must be incorporated. The synchronization process will mainly consist of migrating the Release A OMT models to the Release B repositories. Additional information stored in excel files is considered part of the baseline as well. This can be done without affecting the delta model unless the structure of the baseline design has changed (e.g., new or deleted objects, associations, hierarchies). There are two synchronization points planned. The CDR-A baseline will be used for IDR-B. Any design changes that result from CDR-A RIDS will be incorporated as part of the IDR-B follow up.

6. Document Development

6.1 Design Overview (Vol. 0 305) Development

The plan is to develop a 305 design document which is a parent to the subsystem 305 documents. The Vol. 0 305 will be titled "ECS Release B Design Overview" and will be developed by the Release B Architect and CEG-B with support from the SMO System Support group and the development groups.

The Vol. 0 305 document will extend topics covered in the Release A version. An outline for this document is currently under development.

6.2 DID 313 Development

The Internal Interface Control Document (ICD), DID 313 will be developed by the Release B Architect and CEG-B with support from the Release B development groups. This will be an update to the Release A CDR version, and will include descriptions of all CSCI interfaces, the CSCI Interface Scenarios, Scenario Primitives and Key Mechanisms. This document will include (virtually) all of the Release A 313 plus scenarios added or modified for Release B. The majority of this document will be generated from OMT output.

6.3 Data Dictionary Development

The data dictionary will represent all objects and attributes in tabular form. The required tables will be automatically generated by OMT scripts and will be contained in a separate DID 305 volume. The CEG-B will be responsible for generating the data dictionary. The development groups will be responsible for maintaining the required information in OMT.

6.4 DID 305 Development

The 305 documents cover subsystem design and DAAC specific implementations. There will be one 305 per subsystem. These will be developed by the Release B development groups with support from the Release B Architect and CEG-B. These will contain the following:

- SDPS architecture overview
- Subsystem introduction and overview
- CSCI introductions and overviews
- CSCI descriptions including object models and Internal CSCI scenarios
- All HWCI information

This document will include (virtually) all of the Release A 305 plus additions or modifications for Release B. The majority of this document will be generated from OMT output.

Additional 305 documents will be developed for each DAAC implementation. These will be provided by the ECS Multi- Release Support (MRS) organization.

This page intentionally left blank.

Appendix A. IDR Design Checklist

1 General

- 1.1 Design model focuses on incremental modifications for Release B
- 1.2 All scenarios in which a subsystem participates are represented
- 1.3 All interfaces external to a CSCI are implemented with at least one interface class
- 1.4 Interfaces external to all CSCIs have been identified and coordinated with the other side of the interface
- 1.5 The Data Dictionary is complete

2 Object Models

- 2.1 All classes have been identified
- 2.2 All attributes and operations have been identified
- 2.3 All aggregation hierarchies have been defined
- 2.4 All generalization (inheritance) hierarchies have been defined
- 2.5 All associations have been defined and reflect semantic relationships
- 2.6 All CSCI public interface classes have been identified
- 2.7 All distributed objects have been identified

3 Dynamic Models

- 3.1 Selected operations concept scenarios which emphasize custom developed software are mapped to CSCI Interface scenarios
- 3.2 CSCI Interface scenarios are mapped to Scenario Primitives and Key Mechanisms
- 3.3 Scenario Primitives, Key Mechanisms, and CSCI internal scenarios map to classes and operations via event trace diagrams
- 3.4 Sequence of events is correct and complete

4 Requirements Traceability

- 4.1 Every component (CSCI, CSC, class, HWCI, HWC) maps to one or more L4 requirements
- 4.2 Every L4 requirement maps to one or more components
- 4.3 L4 requirements map to L3 requirements and L3 requirements map to L4 requirements (DID 304)
- 4.4 Operations concept scenarios map to L3 requirements (DID 604)