FIRST/ESA/D/0025.01

| | | | |
|---|---|---|---|
| **Customer** | : **ESA/ESTEC** | **Document Ref:** | **FDS.PWN.001** |
| **Contract No** | : **12628/97/NL/RE** | **Issue Date** | : **26/5/98** |
| **WP No** | : **1110/1120** | **Issue** | : **Draft C** |

**Title** : **FINDAS Concept and Architecture: Technical Report**

**Abstract** : This document identifies the motivations for the design of a new ground segment system for the FIRST mission. It describes the anticipated functionality in terms of data flows and processes, provides a preliminary database schema and data dictionary, discusses architectural aspects, and highlights the risk areas and cost drivers. The document is intended to provide a common frame of reference and vocabulary as a basis for further work.

**Author** : _____
Derek Crockford

**Approval** : _____

**Accepted** : _____

**Distribution** : P. Estaria ESTEC
J. J. Mathieu ESTEC

FINDAS Project Documentation Library

**Hard Copy File: FDS005**
Network ID: P:\FDS\PWN\PWN001_C.DOC


**2 Falcon Way, Shire Park, Welwyn Garden City, Herts, AL7 1TW, UK.**
**Tel : +44 (0)1707 391999**
**Fax: +44 (0)1707 393909**

FIRST/ESA/D/0025.01

| | | | |
|---|---|---|---|
| **Customer** | : **ESA/ESTEC** | **Document Ref:** | **FDS.PWN.001** |
| **Contract No** | : **12628/97/NL/RE** | **Issue Date** | : **26/5/98** |
| **WP No** | : **1110/1120** | **Issue** | : **Draft C** |

**Title** : **FINDAS Concept and Architecture: Technical Report**

**Abstract** : This document identifies the motivations for the design of a new ground segment system for the FIRST mission. It describes the anticipated functionality in terms of data flows and processes, provides a preliminary database schema and data dictionary, discusses architectural aspects, and highlights the risk areas and cost drivers. The document is intended to provide a common frame of reference and vocabulary as a basis for further work.

**Author** : _____
Derek Crockford

**Approval** : _____

**Accepted** : _____

**Distribution** : P. Estaria ESTEC
J. J. Mathieu ESTEC

FINDAS Project Documentation Library

**Hard Copy File: FDS005**
Network ID: P:\FDS\PWN\PWN001_C.DOC


**2 Falcon Way, Shire Park, Welwyn Garden City, Herts, AL7 1TW, UK.**
**Tel : +44 (0)1707 391999**
**Fax: +44 (0)1707 393909**

# TABLE OF CONTENTS

# AMENDMENT POLICY

This document shall be amended by releasing a new edition of the document in its entirety. The Amendment Record Sheet below records the history and issue status of this document.

**AMENDMENT RECORD SHEET**

| ISSUE | DATE | DCI No | REASON |
|---|---|---|---|
| A | 24/3/98 | N/A | Draft |
| B | 12/5/98 | N/A | Draft revised and architecture sections added |
| C | 26/5/98 | N/A | Revised draft with ESTEC modifications |

# 1. INTRODUCTION

## 1.1. Purpose and Scope

This document is a technical report on the FIRST Integrated Data Archive System (FINDAS) ground segment concept and architecture, and is an output of work packages 1110/1120 (RD.12).

The purpose of this document is to provide a unified perspective of the proposed FINDAS system based on inputs from ESTEC documents, COTS tool vendors, relevant computing texts, and discussions with a number of subject experts. The main themes covered are *concept* and *architecture*, where concept concerns the system from a domain point of view, and architecture concerns the practicalities of implementing the system on a network of computers.

The FINDAS concept has been studied using a number of ESTEC documents, but RD.1 and RD.2 are the principal sources. Some effort has been expended in presenting a single, consistent view of the required system. The overall ground segment process, involving both people and computer systems, has been expressed in terms of dataflow diagrams as well as text. The emphasis has been on the stages that lead from the entry of proposals at one end to the production of products at the other. The dataflow diagrams are supplemented with tables of process inputs and outputs. It should be emphasised that these diagrams are not final, but are intended to provoke further discussion.

The principal data types to be stored in FINDAS are represented in class diagrams. The class diagrams given do not contain attributes and operations, and so present an entity-relationship view rather than a strict object view. For this reason they are supplemented with a data dictionary in which the intended responsibilities of the classes are clarified. Support classes and user interface classes have not been given detailed treatment. All of these aspects will be addressed further during future work. The class diagrams are not set in stone, and are very likely to change during the course of prototyping and beyond.

The choice of FINDAS architecture involves the assessment of a number of computing technologies in an effort to provide a high level blueprint of a working system. The emphasis here has been placed on providing solutions to meet selected key requirements. How can the architecture insulate one part of the system from changes made in another and preserve common interfaces across mission phases? How can we achieve scalability and performance with respect to a number of simultaneous criteria? How can we provide security and reliable configuration management? The document discusses the possibilities and provides recommendations and suggestions for further investigation. The treatment is not a comprehensive description of system architecture, but does address the major points of concern.

Certain important aspects of the FINDAS system have not been addressed in this document because they are dealt with elsewhere. In particular, the reader should refer to RD.11 for a review of User Requirements and to RD.7 for a justification of the adoption of the CORBA network objects standard.

A description of the FINDAS Prototype is given in RD.6. The Prototype Phase will demonstrate, test and refine many of the ideas presented in this document.

## 1.2. Overview

In section 2 we identify the motivations and ideas behind the development of FINDAS. In particular we look at the pre-existing science ground segment, as exemplified by ISO. In the light of the new requirements for FIRST we consider the areas in which some fundamental redesign would bring significant benefits. The relationship between FINDAS and the Planck Integrated Data and Information System (IDIS) is also discussed.

Section 3 deals with the functionality of the FIRST ground segment. The basic flow of data during two representative mission phases is described in words and by means of dataflow diagrams. In order to avoid cluttering the diagrams a more complete description of the inputs and outputs of each process (with respect to the database) is given separately in Appendix B.

In section 4 a complementary view of the ground segment is given by means of class diagrams, and these represent a first draft of the database schema. The purpose is to identify the principal data types and the relationships between them. All classes mentioned in section 4 are described in the data dictionary, Appendix C.

Section 5 deals with architectural issues. Access to the FINDAS database is provided by middleware, which is implemented as object servers. The object servers provide abstract views of the data tailored to the needs of particular clients, as well as access control and configuration management. The object servers decouple the database from its clients and render the system more adaptable and maintainable. The database itself will be distributed over many machines and volumes, and some data may be stored locally to improve performance.

Our conclusions and recommendations are given in section 6. The emphasis of this section is on cost drivers and areas of risk.

Appendix A is a glossary of terminology not covered in the data dictionary.

## 1.3. Referenced Documents

The following is a list of documents with a direct bearing on the content of this report. Where referenced in the text, these are identified as RD.n, where 'n' is the number in the list below:

1. FINDAS Preliminary Conceptual Model, Version 1.0, July 1997
2. FIRST Science Operations Concept and Ground Segment Document, PT-03056, Draft 2, September 1996
3. ISO Ground Segment, SOC System Design Summary, SAI/94-221/Dc, Version 1.1, May 1995
4. FINDAS Development Plans, FIRST-EMS-CON002, 15/4/98

5. Notes taken by P. Winder at ILT Meeting at ESTEC on 8/4/98
6. FINDAS Prototype Specification, VEGA Ref. FDS.PWN.003
7. Review of the use of CORBA in FINDAS, VEGA Ref. FDS.PWN.002
8. Implementation of a Prototype Integrated Data Archive System for the FIRST Ground Segment - Technical Proposal. VEGA Ref. FUT.PRO.845
9. Object Databases: An ODMG Approach,
R. Cooper, Thompson Computer Press, 1997
10. C++ Object Databases: Programming with the ODMG Standard,
D. Jordan, Addison-Wesley, 1998
11. Review of FINDAS User Requirements, VEGA Ref. FDS.PWN.004
12. Software Project Management Plan for the Implementation of a Prototype Integrated Data Archive System (FINDAS). FDS.PLN.001 Issue 1, 20 March 1998

# 2. FINDAS CONCEPT OVERVIEW

FINDAS is intended to be the backbone of a decentralised science ground segment architecture for the FIRST mission. In this role it must link the ESA science operations centre and the geographically distributed Instrument Control Centres throughout all phases of the programme. In the case of Planck, FINDAS will distribute the Planck TM received from the MOC to the Planck Data Processing Centre (DPC) entry point (Level 1). The distribution mechanism is not yet fully defined (see Figure 1).

In this section the overall FINDAS concept and its major features are examined. In particular the benefits of undertaking a new approach to ground segment design are highlighted.

## 2.1. FINDAS and IDIS

When the Statement of Work for the implementation of the FINDAS prototype was issued it was not clear if the two missions (FIRST and Planck) would be combined in some way or would be kept separate. The possibility of FINDAS providing some services to Planck was envisaged but due to the uncertainty of the overall scenario and in order to bound the overall effort in was decided by ESA that, in a combined scenario, FINDAS would only provide the following functions:

- Distribution of the Planck Telemetry from the MOC to the DPC entry point
- Communications between DPC entry point and MOC
- Storage of the Planck "final products" into FINDAS (TBC)

Furthermore, it was assumed that, as far as the FINDAS *prototype* was concerned, **no specific functionality** was required to support Planck beyond the ability to segregate between Planck and FIRST data. In particular, the FINDAS *prototype* **does not need** to support a Planck data model.

Within the framework of a Planck-alone mission the Planck community, led by the two Planck PI teams, had already started definition of the Integrated Data and Information System (IDIS). IDIS and FINDAS share many common characteristics:

- Providing a link between various, geographically separated centres
- Handling of a large volume of data
- Controlled exchange of data and software between centres
- Implementation following an O-O approach
- Maximum use of COTS products

However, due to the different natures of the two missions (FIRST is an observatory, Planck is a survey-type PI mission) the FIRST and Planck data

models are very different. This implies in principle that the FIRST and Planck applications which represent the major part of the respective FINDAS and IDIS efforts will be specific to each mission.

It is agreed that a common approach to the implementation of the infrastructure elements (e.g. COTS, DMS, configuration control, communications protocols, etc.) in the operational versions of FINDAS and IDIS is desirable. Steps are currently being taken by ESA to ensure the necessary exchange of information between the two communities and to provide visibility to the Planck community into the FINDAS prototyping effort. However, **no Planck specific** requirements have been introduced into the prototype implementation. The remaining part of this document will therefore no longer make reference to the Planck mission although many of the issues addressed could be of relevance to Planck and the IDIS effort.

## 2.2. Pre-Existing Science Ground Segment

The underlying functional capabilities, mission phases and data types of the FIRST ground segment are similar to those of the existing Infrared Space Observatory (ISO). The ISO science ground segment supported all aspects of the mission's core science tasks in particular during the operational phase. These include the following set of major tasks which were implemented by separate software applications:

- Entry of observation proposals
- Handling of received proposals
- Construction of observation schedules with associated command file
- Receipt and initial analysis of real-time telemetry
- Processing of the scientific data
- Archival and delivery of the (processed) scientific data to the observers

As FIRST is also an observatory style mission the majority of the above functionality is also needed for FIRST. Hence at a glance it would seem reasonable to reuse the centralised ISO architecture model, during development and operations, since a large proportion of the task functionality is unchanged. However experience has shown several key disadvantages to the ISO architectural model, including:

- interfaces had to be changed (sometimes extensively) between different mission phases – leading to respecification and reimplementation of parts of the ground segment
- software was developed separately and then had to be integrated into a centralised operations centre – leading to changes and inconsistencies
- the heterogeneous software systems had their own separate databases, which could not be kept consistent

- changes to one subsystem or data structure often required matching changes to other subsystems hence causing a ripple effect

- considerable effort was required from the instrument groups throughout all mission phases – the centralised operations required collocated instrument expertise during the operational mission phase, which could have been more effectively utilised at the home institute

Therefore the adoption of a new science ground segment architecture must build upon the successful aspects of pre-existing ground segment, but bring additional enhanced benefits to both the ESA Science Centre and the FIRST instrument centres.

## 2.3. FIRST Science Ground Segment Concept

In order to address the aforementioned pre-existing ground segment shortcomings, the concept advanced for the FIRST ground segment is that of a:

- homogeneous,

- consistent,

- evolving and

- distributed architecture,

which will allow the re-use/re-engineering/development of the applications required to perform the science tasks for the FIRST mission.

As illustrated in Figure 1 the major components of the ground segment are:

- Mission Operations Centre (MOC)

- FIRST Science Centre (FSC)

- Instrument Control Centres (ICCs) for FIRST

- and the pre-operational phase role of the FIRST Project Team

which are connected by and share information and data through the FIRST Integrated Data Archive System (FINDAS). The Integrated Data and Information System (IDIS) and Data Processing Centres (DPCs) of the Planck mission may also make use of FINDAS facilities (see section 2.1).

In order to achieve the benefits which are promised by adopting this new ground segment architecture the critical component is FINDAS. FINDAS consists of the backbone which connects together:

- people
  - the various teams working on FIRST
  - the external community

  via access to documentation and data;
- application software
  - at each centre

  via access to stored data and associated data transfer.

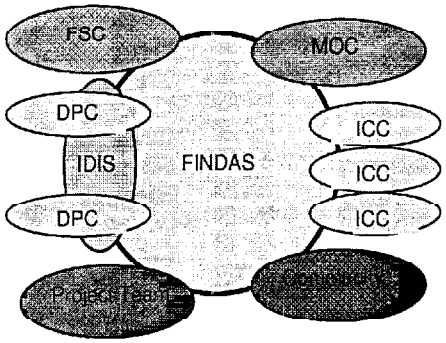


**Figure 1 - FIRST Ground Segment Concept**

## 2.4. FINDAS Conceptual Design

FINDAS has the following conceptual design:

- A solution based upon a COTS Object-Oriented Database Management System
- User applications which connect to the database by means of a 3-tier client server architecture
- A distributed architecture in which the master database is present at the FIRST Science Centre (FSC) with replicated/selected data storage at each Instrument Control Centre (ICC)

The implementation of this FINDAS design within the overall FIRST ground segment architecture has the following advantages over pre-existing science ground segments as typified by ISO:

| Issue with Pre-Existing Ground Segment (ISO) | FIRST Comparison | Resultant Feature on FINDAS | Derived Benefit |
|---|---|---|---|
| separate systems for entry of calibrations and observations | unified system with all uplink requests entered as part of proposals | new system has different levels of flexibility according to usage | simple consistent interface which eases workload on both users and developers |
| interface (& hence data or s/w) changes between phases | seamless transition between phases | common homogeneous data interfaces in all phases | minimise effort wastage due to unnecessary interface changes |
| centralised architecture | distributed architecture | master database with replicated or selected data at ICC and good communications between locations | increase effectiveness of ICC by enabling work to be performed at home institute in all phases |
| multiple separate databases | one database | master database with replicated or selected data at ICC | consistent data availability for all users |
| separate development phase software applications which had to be integrated at central location for operations | applications to be developed and operated locally through all phases | 3-tier client server architecture | minimise "big bang" integration issues and increase cost effectiveness during operational phase |
| small data structure changes requiring changes to multiple subsystems | flexibility and adaptability | 3-tier client server architecture | object servers hide the complexity of the database from applications and localise effects of interface changes during development hence providing inherent flexibility and adaptability |
| complex data items but difficult to maintain relationships between data items | complex data items with sophisticated relationships and configuration management facilities | use of object oriented database - these were developed to handle complex data schemas with configuration management | implementation directly reflects desired organisation of data<br>reliable configuration management |

| Issue with Pre-Existing Ground Segment (ISO) | FIRST Comparison | Resultant Feature on FINDAS | Derived Benefit |
|---|---|---|---|
| not initially designed for long life | design lifetime of 20 years | use of COTS object oriented database and COTS network protocols | ODMG standard databases and CORBA are being developed vigorously by several vendors, hence long term durability and adaptability are assured |
| disparate data storage/retrieval | centralised control of storage/retrieval of data | access rights and configuration management are coordinated across system | data privacy and security are assured: reliable version control becomes feasible |
| original archive needed redesign for full post mission science requirements | data growth and external access expected | object oriented databases are being used in high volume situations<br>external web access is part of architecture | minimise cost to provide a post mission archive separate from the initial archive |
| no document management system | document management included | provides external access to DMS<br>provides internal "simplified" document system | enables simple and consistent access by FIRST teams to all relevant documents |
| pipeline processing of observation data to a suitable standard | no pipeline for FIRST | able to provide raw data bundled with necessary processing s/w and auxiliary data | observers can be given all necessary data and software<br>no expensive processing burden on FIRST |
| near continuous telecommanding | limited TC window so onboard storage of commands (3 day), macro commands | possible definition of new data objects<br>no CCS execution in real time | limited effect |

| Issue with Pre-Existing Ground Segment (ISO) | FIRST Comparison | Resultant Feature on FINDAS | Derived Benefit |
|---|---|---|---|
| near continuous telemetry stream | all TM received in 2 hour window (22 hour dump & 2 hour real-time) | differentiates between dump and real-time stream (via MOC)<br>real-time stream delivered (as required) to ICC quickly<br>remainder fed down lines to ICC after real-time finished<br>all data stored | real-time telemetry available quickly at ICC, hence ICC may function as if it were co-located at FSC |
| telemetry in a fixed TDF format | packet telemetry on basis of spacecraft subsystem or instrument | distinguishes originating source for each packet<br>delivers only requested source packets to each ICC | minimises traffic flow over network<br>each ICC can request packet sources they are interested in, hence maximum flexibility |
| limited and predetermined querying access to data from external users (prior to post mission) | different types of user | support to novice and expert, local and remote, internal and external users<br>powerful query access can be given using OQL query language | access control limits unauthorised access<br>novice users given structured access<br>expert users given freer access to query database |

**Table 1 – Benefits of FINDAS Compared to ISO Ground Segment**

# 3. A FUNCTIONAL VIEW OF THE GROUND SEGMENT

This functional analysis of the FIRST ground segment concept has been performed by attempting to unify the descriptions given in RD.1 and RD.2. Concentrating only on the parts relevant to FINDAS we have used RD.3 to anticipate what is likely to carry over from the ISO mission. The intention is to capture the data items and data processing interactions which will form the FIRST science ground segment. Rather than describe all mission phases we have chosen the operational and instrument level test (ILT) phases as being representative of the mission as a whole. For the ILT phase we have also drawn on RD.4 and RD.5. The architectural aspects of the FINDAS concept are dealt with in section 5.

The dataflow diagrams given in the following sections are not intended to be exhaustive but rather to capture the key aspects of the phases. A more complete listing of process inputs, process outputs, and general data types is given in Appendix B. There is a glossary of terminology in Appendix A.

## 3.1. The Operational Phase

The data types of the "uplink request to product chain" form the basis of our analysis of the operational phase and their use is summarised in dataflow diagrams: Figure 2 deals with the uplink, Figure 3 with the downlink. The notation used in the diagrams is standard. The rectangular boxes are *actors*, that is, entities at the boundary of the system that produce or consume data. The ellipses represent *processes*, where data is manipulated and transformed. The heavy parallel lines are *data stores*. The various data stores depicted in the figures are all part of the single FINDAS database and represent either database segments or different views of the same data.

The usage of a FIRST instrument begins with the creation of a proposal, which groups together a number of uplink requests together with associated data and documentation. Each successful uplink request ends up being converted into telecommands (TC). The processing of the telecommands on the spacecraft generates telemetry (TM) which is processed on the ground to yield products. This will be the case whether the initial request is for an observation, a calibration, or a test, and whether the final product is a standard astronomical product, calibration data, or a test product. The flow of data along the many stages that form the "uplink request to product chain" is implemented by means of many different files, which require version control, access control, and to be linked in sophisticated ways.

The unit of instrument exposure is the uplink request, of which there are three types: observation, calibration and test. An observation is an uplink request from an astronomer or survey consortium, a calibration is an uplink request from a calibration scientist, and a test is an uplink request from a test team. All uplink requests are entered into the system as part of proposals using common software called Proposal Generation. Observations of astronomical objects may be defined by filling out an astronomical observation template (AOT), which restricts the instrument settings to those of a valid observing mode. The Proposal Generation software uses these filled AOTs to generate ordinary

observations. Calibrations and tests may be entered by the calibration and test staff and allow free access to the instrument. They are entered as part of a special type of proposal called a test procedure.

The dataflow in the uplink is depicted in Figure 2. An observation is not eligible for scheduling until its proposal has been checked and approved by the FSC staff and OTAC using privileged software called Proposal Handling. Calibration and test staff generally have direct access to Proposal Handling, and do not need OTAC approval. Some types of test bypass scheduling altogether and are passed directly to flight control, but a copy of the test procedure is always lodged with FINDAS for archive purposes.

The first stage of the scheduling process is performed by the Planning System and involves the placement of uplink requests into a plan. A plan defines the instrument activity over a unit observing period and has an associated file containing the corresponding instrument commands. The time available for observing during an observing period is defined by the planning template, which is provided by the flight control system. An uplink request may be rejected by the planner if it cannot be planned for some reason.

The second stage of scheduling involves the processing of the plan and instrument commands by the Scheduling System to produce a central command schedule (CCS). The CCS defines all spacecraft and instrument activities in the proper sequence with start and end times. It applies to a single planning period and is used by the flight control system to produce the telecommand sequence transmitted to the spacecraft. A plan may be rejected if it is found to be unsatisfactory.

The dataflow in the downlink is depicted in Figure 3. The flight control system manages telemetry, auxiliary data, the telecommand history and an operations log, all of which must be stored on FINDAS. The process of inputting telemetry into FINDAS will involve the association of each TM packet resulting from an uplink request with the request.

The initial online assessment of downlinked data is performed by a system called RTA/QLA, which is part of a physical system called an Instrument Station (IS). The recorded products of RTA/QLA are the IS report and RTA/QLA products, which relate to the uplink request, and the log of the IS activities.

The offline processing of data is performed for each instrument by two systems, the pipeline and interactive analysis. The pipeline produces standard astronomical products and may be implemented either as a local semi-automated system or as a bundle of telemetry and processing software for astronomers to make use of at their own institutions. Interactive analysis is implemented as part of the instrument station. It is performed by the calibration scientists and results in calibration products. An instrument test team may make use of its own specialised software to produce test products. Both calibration and test products may be associated with a further level of processing to produce the calibration data which characterise the behaviour of an instrument.

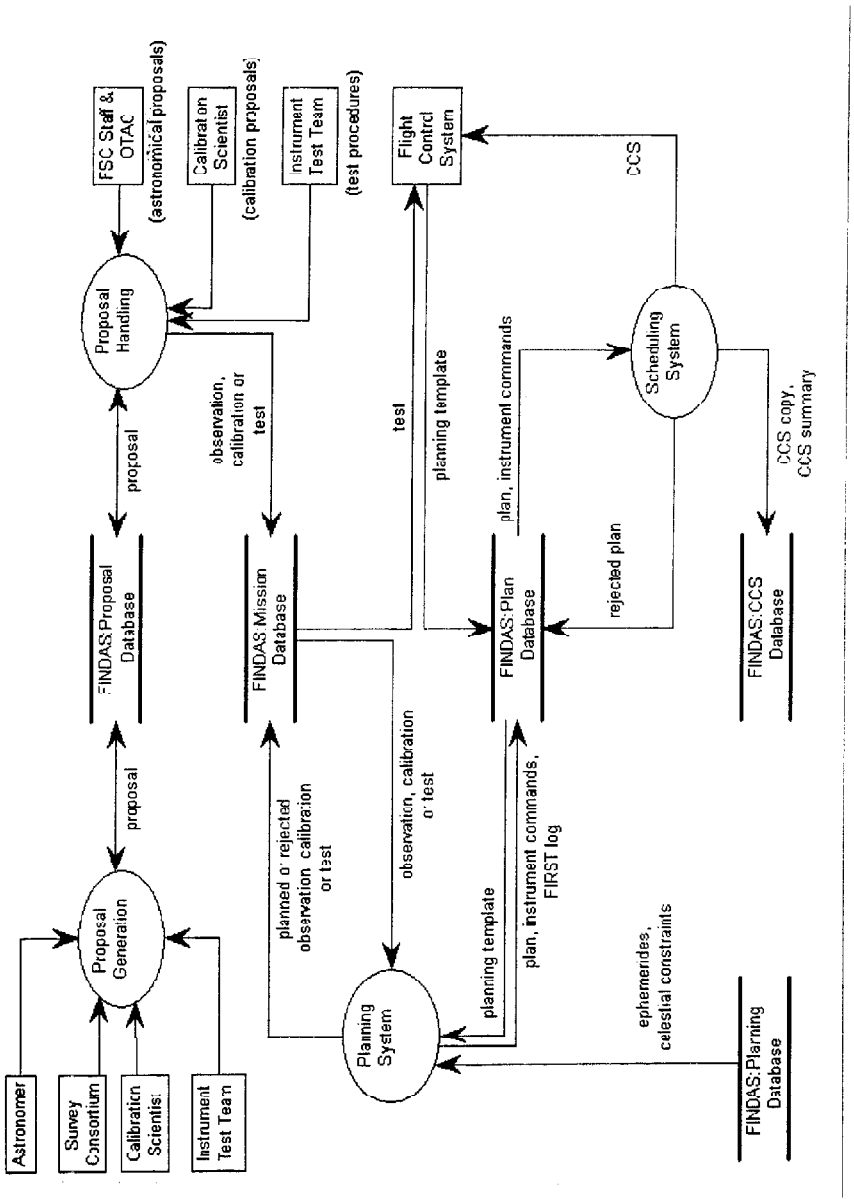


**Figure 2 - Uplink Dataflow for the Operational Phase**

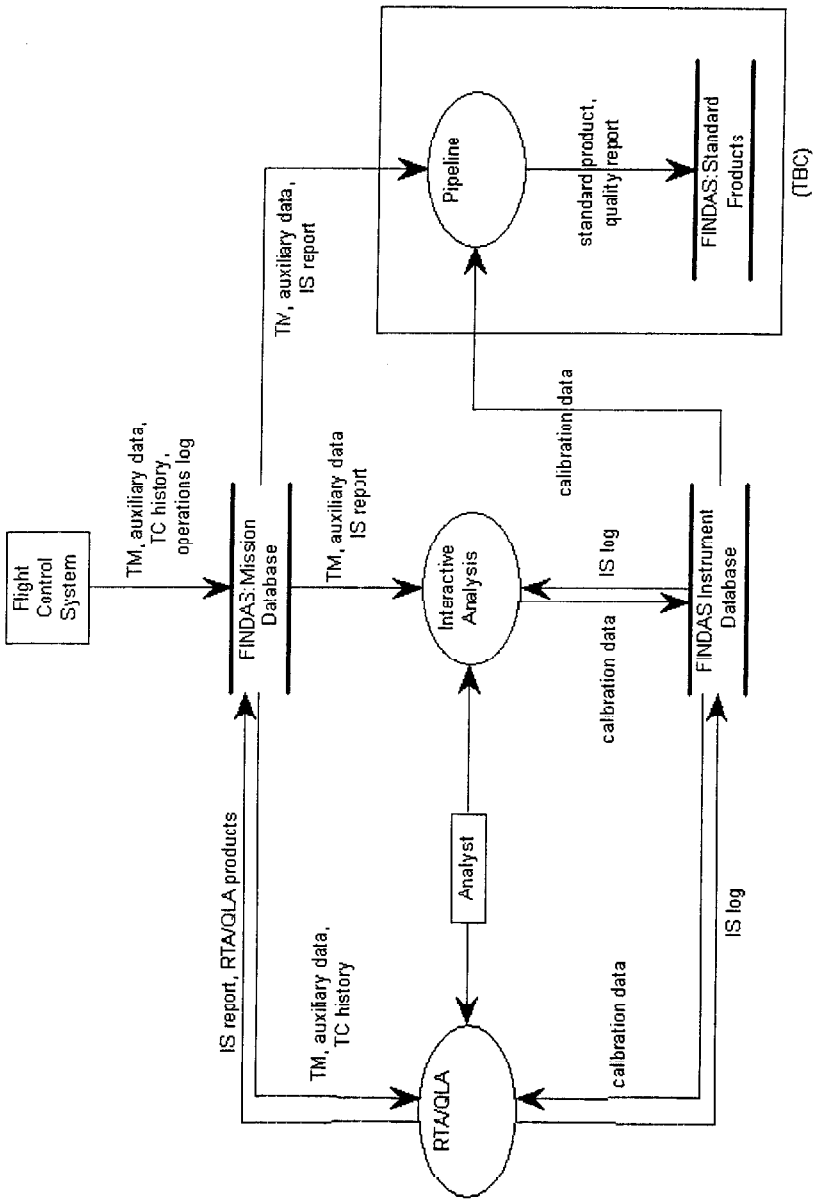


**Figure 3 - Downlink Dataflow for the Operational Phase**

## 3.2. The Instrument Level Test Phase

The instrument level test (ILT) phase is a pre-operational phase in which the structure of the instrument telecommands and telemetry is determined, the AOTs for an instrument are defined, and the instrument is characterised through calibration and testing. The ILT phase generates considerable software and documentation. The storage of data files such as software elements and document components, which may have sophisticated attributes and inter-relations, will be a basic FINDAS service. In fact, this is one of the main reasons that the FINDAS architecture has been based on the use of an object oriented database (see section 4). The initial requirements of a FINDAS document management system are dealt with in RD.6. Hence we prefer to concentrate on a different aspect of ILT, that of the testing and calibration of the instrument. It should be noted that the ILT phase is very similar to the check-out phase, the principal difference being that in check-out the instrument has been integrated into the spacecraft.

During ILT, instrument testing takes place through the use of a test rig which simulates the rest of the system, i.e. spacecraft, flight control and external stimuli. Figure 4 is the dataflow diagram for the ILT phase, which is basically a simplified version of Figures 2 and 3. Here the test control system takes on the roles of the spacecraft and the flight control system. Instrument test teams enter tests and calibrations into the FINDAS ILT database. The test controller executes them according to the test procedure defined by the test teams (these are an analogue of the proposals in the operational scenario and are also stored in FINDAS). The instrument generates telemetry which is stored in FINDAS and passed in real time to an analysis system (the Instrument Station, running RTA/QLA).

There is an architectural decision to be made as to whether telemetry passes to the instrument station via FINDAS or is split into duplicate streams for the two destinations. However, in order to achieve a smooth transition to the check-out and operational phases it is important that the interface between FINDAS and the instrument station is stable. The test control system coordinates the overall test process. An analyst working at the instrument station may wish to rerun a test from any phase based on telemetry stored in FINDAS. Equally, during the check-out or operational phase, a link to the central check-out equipment (CCE) or flight control system should replace the spacecraft simulator without causing disruption to the mission schedule.

The data stores shown in Figure 4 are a suggestion for the way in which the data from the ILT phase might be organised in FINDAS. It seems sensible to keep ILT bulk data, chiefly telemetry, together with those products directly derived from it in a separate database segment. On the other hand, those data types that are applicable across phases should probably be placed in a common segment (compare Figure 3).

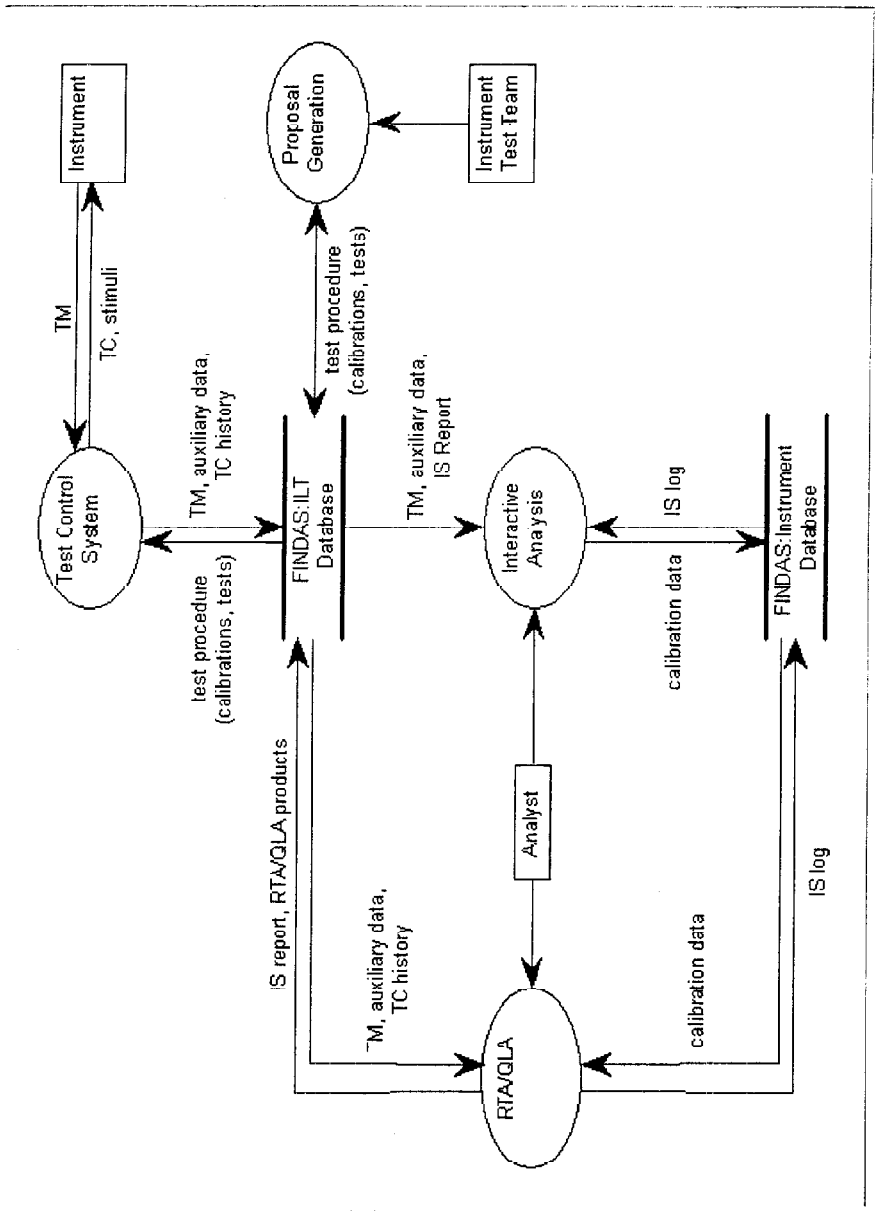


**Figure 4 - Dataflow for the ILT Phase**

# 4. A CONCEPTUAL DATA SCHEMA

While the previous section was a functional analysis, concentrating on the processes in the ground segment and the data items they manipulate, we now look at the data items in the context of the relationships between them. The FINDAS database will contain a variety of data types but its fundamental role is to provide sophisticated file management. Many database objects will provide boxes that hold the various types of file along with supplementary information in the form of attributes. Formal associations between objects will capture the relationships between files and will facilitate the grouping of data into meaningful units. It is important when designing a system to ensure that it is possible to navigate easily between related items. The class diagrams that follow represent different views of FINDAS objects and their relationships: objects can be organised in many different ways at the same time, subject to consistency.

In object oriented terminology a class is a data type that it is meaningful to consider as atomic, or in database terminology, an entity. A class diagram is an entity-relationship diagram with some extensions in notation to support concepts such as inheritance. A class embodies both the data values or attributes that compose an entity and the operations that may be applied to the entity. An object is an instance of a class. In a class diagram, a square box represents a class. The box is divided into three sections: the top one gives the class name, the middle the attributes, and the bottom the operations. In the diagrams given we concentrate on the identification of classes and their associations, which is the step performed first in most object oriented methodologies. The types of association represented are inheritance, aggregation and relation. Inheritance means *is a*: anything that can be done with an A can be done with a C provided C *is an* A. Objects A and B are related if we need to be able to navigate to one given the other. Aggregation signifies the *part of* association, which is like relation but semantically stronger.

Figure 5 is a class diagram centred on the associations important in the operational phase (compare Figures 2 and 3). The most fundamental classes in FINDAS are the uplink and downlink. An observation by an astronomer *is an* uplink, and similarly for calibrations and tests. There is a relation between an uplink (or downlink) and all classes in the corresponding "uplink request to product chain". This means that given an uplink (or downlink) object we can find any object in the chain quickly and straightforwardly, and vice versa (depending on whether the associations are implemented as bidirectional). The uplink class gathers together all of the files used in the generation of the telecommands; the downlink class gathers the telemetry packets themselves and everything derived from them. The concept of an uplink unifies the different types of "observation" required during the mission.

Every request is *part of* a proposal. The proposal will be the unifying concept in the entry of requests into the system. A proposal will correspond to a single user or consortium and will have associations with all documents involved in the proposal process. A test procedure is a special type of proposal entered by an instrument test team.

The other classes below uplink have already been touched on in section 3, as have the classes below downlink. The instrument commands class associates

instrument commands with an uplink. A plan is associated with a CCS for the planning period. By associating the auxiliary data and other files with the CCS we have assumed that these refer to a single planning period. This also might have to be revised in the final system, but the intention behind the association should be clear.

The class diagram for the ILT phase is given in Figure 6 (compare Figure 4). It can be seen that Figure 6 is really a subset of the class diagram for the operational phase (Figure 5). This is no accident, as a smooth transition between mission phases can only be obtained if there is a consistent core in the data schema which is common between phases.

The instrument class provides an alternative view of our schema in Figure 7. It is associated with many of the classes that are not directly related to the "uplink request to product chain". An instrument object will exist for each of the instruments on the spacecraft. It is the repository for all information relating to the instrument, such as manuals and contact details of responsible individuals. Associated with an instrument are the ICS, PCS and macro lists, which provide the commands recognised by the instrument, the instrument logic and characteristics, which describe its behaviour and limits, and memory images, which are the software that actually control the instrument on the spacecraft. The standard observing modes of the instrument are defined by its blank AOTs. Calibration data characterises the instrument. Telemetry or products that cannot be associated with an uplink object, such as serendipity telemetry or trend analysis, can be conveniently associated with an instrument.

It seems sensible to provide a spacecraft class as an additional starting point for navigating the database (Figure 8), as it forms a natural point from which to reach a number of classes that have not been included elsewhere. The spacecraft class is a singleton: there will be only a single instance of spacecraft in the model.

A comprehensive set of class descriptions is to be found in the data dictionary (Appendix C). The class diagrams and data dictionary are not exhaustive but are intended to capture the key classes and their relationships. Other classes and class diagrams can be added to the model as needed. The class diagrams do not contain information about the directionality or cardinality of associations, which is appropriate to a more detailed level of analysis.

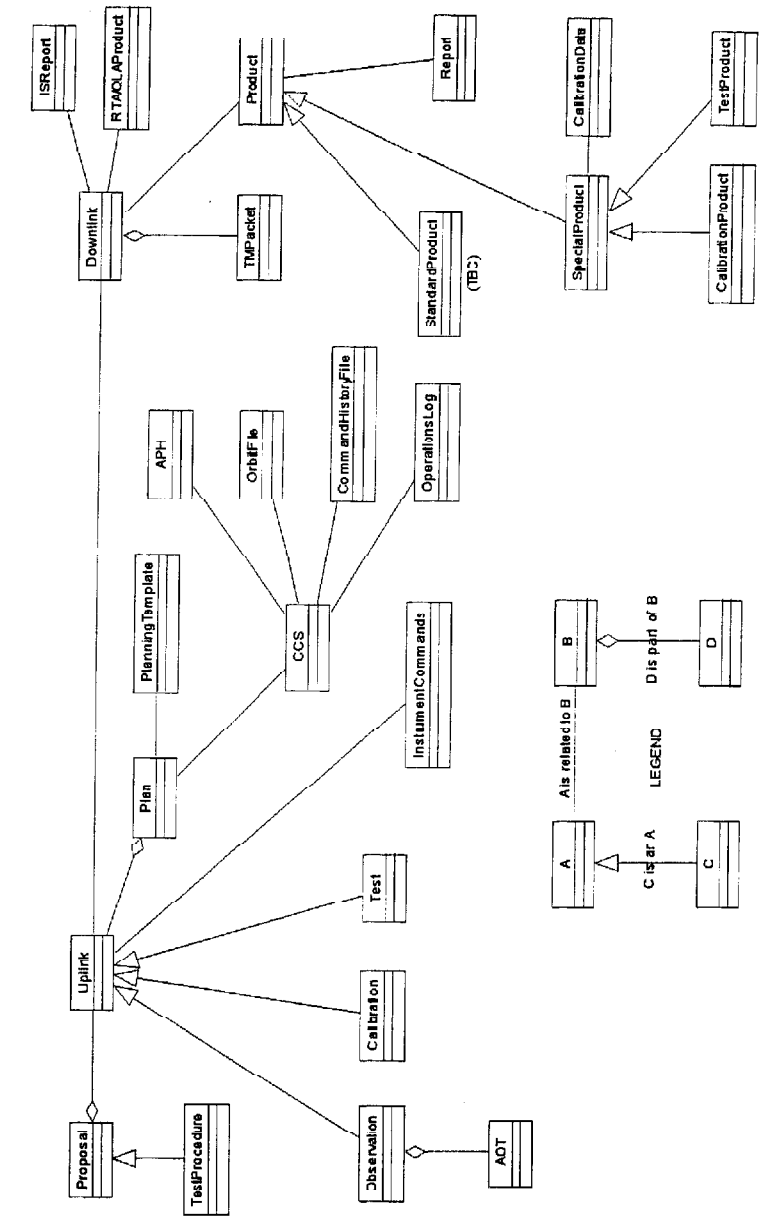


**Figure 5 - Class Diagram for the Operational Phase**

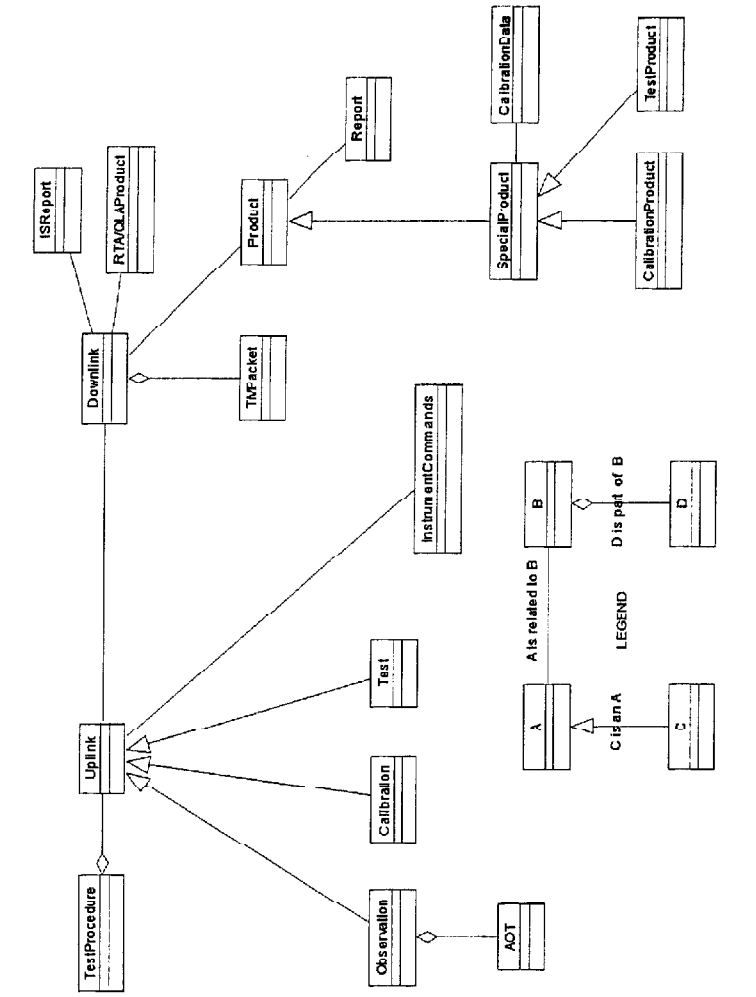


**Figure 6 - Class Diagram for the ILT Phase**

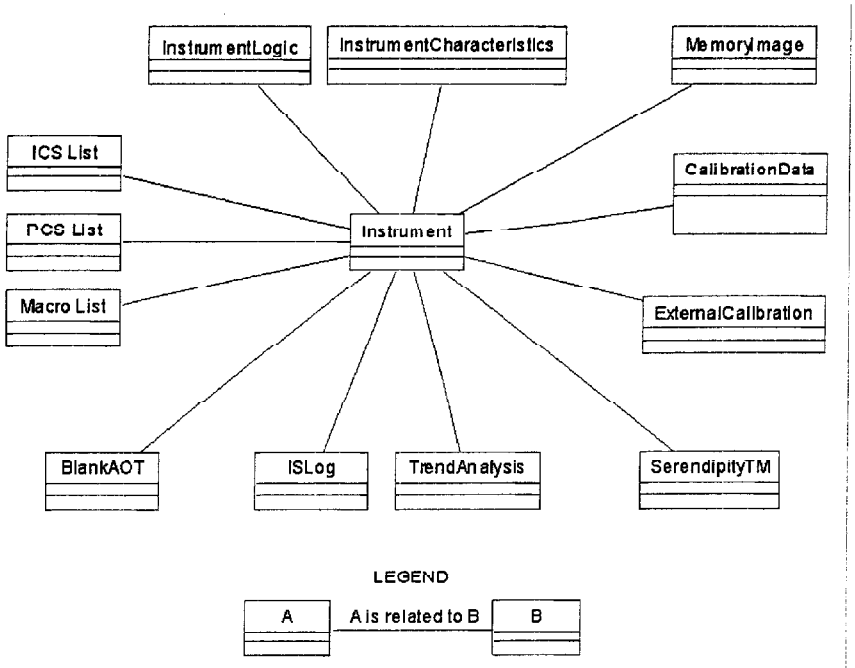


**Figure 7 - Class Diagram Centred on the Instrument**

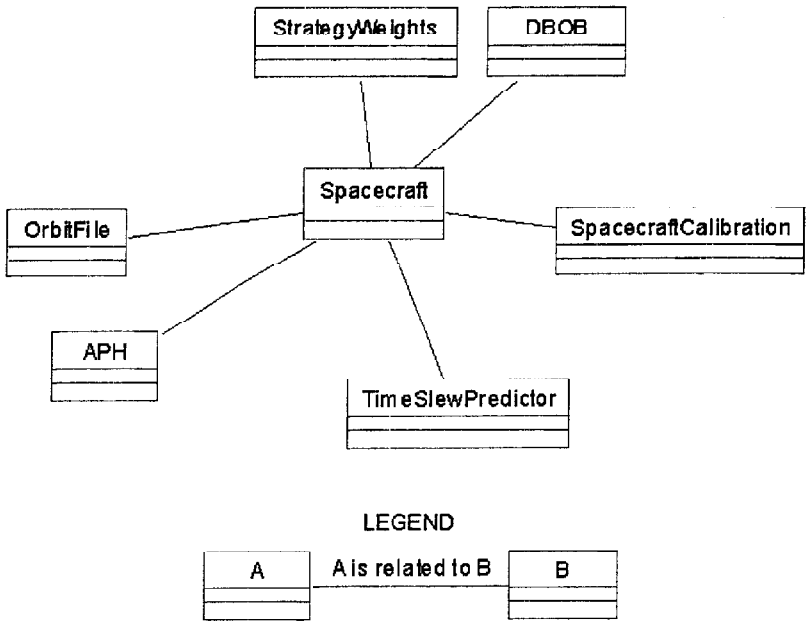


**Figure 8 - Class Diagram Centred on the Spacecraft**

# 5. AN ARCHITECTURAL VIEW OF THE GROUND SEGMENT

## 5.1. COTS Framework: CORBA and ODMG

Given that FINDAS is to be developed using object oriented techniques it is evident that system development will be done using mainstream object oriented programming languages. We know that elements of FINDAS will be distributed across a network, and that data must exist beyond the lifetime of any particular process. When searching for a COTS framework on which to base our architecture we are therefore looking for support for the interoperation of networked objects and for a database technology that integrates as seamlessly as possible with the use of object oriented programming languages.

There are two chief technologies available to support object communication across a network: CORBA and DCOM. Whereas CORBA is an OMG (Object Management Group) standard supported by virtually the whole software industry and is available on nearly all platforms, DCOM is a Microsoft proprietary product which is being vigorously developed for Windows platforms. We therefore recommend that FINDAS client networking be based on CORBA (see RD.7).

Objects that can maintain their state independently of being in the memory of a process are called persistent. While it is possible to implement persistent storage using the file system it is less effort in the long term to adopt a database: when the quantity of data stored becomes significant the use of a COTS client server database becomes essential. The difficulties in mapping an object model onto a relational database have been discussed elsewhere (RD.8), but in brief the process involves considerable development work and can require numerous table joins, which hit performance. In addition, the results of ad-hoc queries using embedded SQL require explicit conversion into programming language types. We have therefore advocated the use of an object database in FINDAS.

Object databases can provide near seamless persistence to the objects manipulated by a programming language and their support for client server architecture makes the presence of a network almost transparent. The ODMG (Object Database Management Group) language bindings (RD.9, RD.10) and the object query language, OQL, mean there are now standards that allow applications to be developed independently of database vendor, and for ad-hoc queries to be performed with the same ease as in relational databases.

The ODMG base standard prescribes a minimum set of facilities that a database must provide in order to be compliant. The ODMG language bindings define the interface to ODMG facilities in object oriented languages such as C++, Smalltalk and Java. OQL provides an SQL-like query language that can be used standalone or embedded in code. The ODMG bindings and OQL introduce no new language syntax and use the type system provided by the language itself: ODMG facilities and OQL access are provided by a class library. In C++, a program using persistent objects is distinguished by the use of a special pointer class, a common base class, and an overloaded new operator. In this document, we shall just assume that all objects in the example code are persistent. ODMG facilities and OQL access can be made available across the network using CORBA, as an alternative to using the database client server protocol.

## 5.2. Three-Tier Client Server System

### 5.2.1. Historical Perspective

The original database systems were designed to run on mainframes connected to dumb terminals. The mainframe had to be large and powerful, as it had to handle the combined workloads of database access, application logic and user interface management (a relatively small overhead in the case of dumb terminals). With the ubiquity of the personal computer came the possibility of providing a more sophisticated windowing interface requiring considerable machine resources, and some or all of the application processing, on the user's local machine. The ease of the workload on the database host allowed smaller, cheaper machines to be used to house databases: often individual departments could afford their own database machine. This combination of the use of personal computer clients with smaller database hosts is the traditional client-server architecture.

The client-server architecture was very popular but itself suffers from a basic flaw: by breaking apart the database and processing across the system the resulting parts are forced to be very dependent. Any change in the database access interface, application logic, or even the database itself required the deployment of new versions of client components, and would usually break all previous client versions. This problem was addressed through the three-tier client-server architecture, in which a third set of machines, called application servers, takes on much of the application logic and serves to insulate clients from changes made in the database. The current trend towards object oriented methods and component based systems is driving the creation of the distributed architecture. In this architecture the roles of database server and application server are merged, their functions being provided by a collection of object servers. An object unifies a data structure with its associated processing. Object servers are therefore classified in terms of the services provided by the objects they host, and the functionality of a distributed system can be extended simply by the addition of a new object server.

### 5.2.2. Applicability to FINDAS

The FINDAS requirement that all data be held in a single unified database which presents a common interface to all users through all phases mitigates against the adoption of a true distributed architecture. Instead we shall tend towards a hybrid distributed/three-tier approach. A recommended scheme is illustrated in Figure 9. Here the middleware is be provided by object servers, but the object servers share a common object database, rather than storing all objects locally. Modern object databases can distribute data over a number of machines while maintaining a single interface, allowing the database to scale as needs develop. Object based middleware is intrinsically scalable: to increase performance with respect to a particular object simply add more server processes for that object, either on the existing servers or on additional machines.

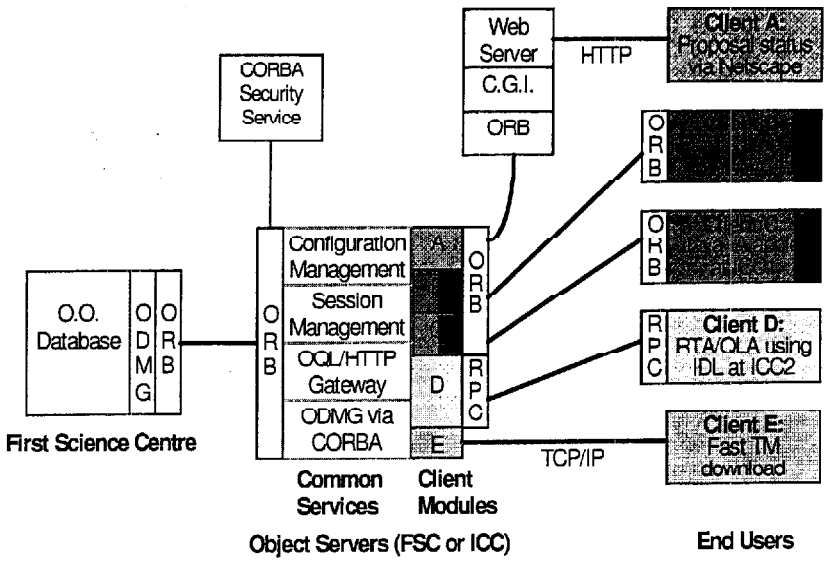


**Figure 9 - FINDAS Three Tier Client-Server Architecture**

In Figure 9, a CORBA representation of the ODMG interface provides access to the objects contained in the database, as well as facilities for transaction management and locking. Object server client modules will provide customised views of the data, and could be written and maintained by those responsible for the corresponding clients. This would give considerable autonomy and control over the development schedule. A client module developer can make use of the common services, such as configuration and session management, provided by a common set of libraries. The developer can define new classes that exist solely in the object server for the purpose of providing a convenient view of the database to their clients. These new classes are instantiated by gathering and manipulating raw database objects. Any change to the database schema is accommodated by making changes to a relatively small number of object servers, rather than a large number of diverse clients scattered over a multitude of machines.

The connection between object servers and clients is standardised on CORBA and HTTP. This provides a standard interface across both development groups and technologies. Some example possibilities are shown in Figure 9. Traditional clients based on C++/Motif can make use of ORB software provided by their CORBA vendor. Web clients implemented as java applets may download (perhaps transparently) a mobile java ORBlet, or use the built-in ORB provided with more modern web browsers. In CGI based web applications it is the CGI program that communicates with the object server. The development of web based user interfaces will be greatly eased by the OQL/HTTP gateway library,

which supports the generation of HTML pages in response to OQL queries. Of course, in special circumstances it may be desirable to use protocols other than CORBA to connect to clients, and the architecture does not prevent this. Hence, the download of bulk telemetry might use a raw TCP/IP socket interface. Similarly, an application developed in a visualisation environment such as IDL, which might be based on Fortran or C, could make use of CORBA with a restricted interface definition but could equally employ a procedural protocol such as RPC if need be.

In the design of Figure 9 the database/object server connection is achieved using CORBA, and so access control can be provided transparently using an implementation of the CORBA Security Service. While modification of the client modules is under the direct control of development groups, modification of the database schema will have to be more closely controlled. If it is necessary to subclass from an existing class that is represented in the database, and if the new class involves the storage of additional data items, then the database schema will have to be altered. This will involve reinitialising the ORBs with the CORBA IDL (Interface Definition Language, see RD.7) for the new schema, and getting the object servers to recompile against the new header files. The necessary CORBA IDL and header files can be generated from the schema using database tools but the amount of work involved may still be significant. It should be stressed that the subclassing of a class in the database schema will not cause existing object servers to cease to work, as their interfaces will be unchanged. The clients, of course, are not affected by a change to the database.

### 5.2.3. An Alternative Design

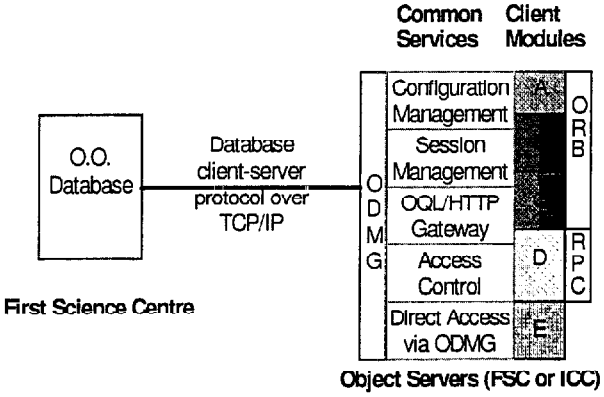


**Figure 10 - Alternative Architecture Using a Direct ODMG Back-End**

Figure 10 presents an alternative design which differs from Figure 9 only in the detail of the back-end. Here, the object servers are presented with the raw ODMG language binding rather than a CORBA implementation of the ODMG

interface. The connectivity between database and object servers relies on the native client-server protocol of the database vendor. The chief advantage of this solution is that an ODMG language binding is a much more straightforward interface to an object database than a CORBA implementation of ODMG, and the recompilation of object servers following a database change is likely to be achieved more quickly as a result. Another advantage is that the database client-server protocol is likely to give better performance than CORBA in the transfer of bulk data such telemetry. In addition, an ODMG language binding is a standard interface, whereas a CORBA representation of ODMG gives a standard set of facilities, the precise interface of which is likely to be database dependent.

The disadvantage of the Figure 10 scheme is that the CORBA Security Service cannot be used to protect objects at the database level. Instead it becomes the responsibility of the client module developers to make sure that end users do not gain unauthorised access to data. This would be supported by an additional library which provides access to the sort of classes described in section 5.4. It could also be rendered easier by careful design, ensuring that related objects in the database schema share the same access permissions as far as possible, as much database access will consist of traversal of the relations between objects. Regulation of associative access via OQL is more problematic, but this will be the case however access control is enforced (see section 5.4).

### 5.2.4. Recommendations

The final choice between the architectures of Figures 9 and 10 can only be made in the light of the results obtained from the Prototype Phase (RD.6).

## 5.3. Distribution of Data Storage

A key difference between FINDAS and previous ground segment systems is that FINDAS must hold the master copy of all data at the FSC but must support users with demanding access requirements situated at diverse locations. In particular, it is envisaged that the ICCs for the FIRST mission will not be geographically close to the FSC, but will be networked to the FSC using leased lines. There is therefore a trade-off to be considered between the bandwidth of the leased lines used and the provision of local storage at the ICC. Local storage generally provides superior performance for the access of a selected subset of the full database but can give rise to higher administration and development costs. It is the purpose of this section to discuss the various options and their balance of advantages.

### 5.3.1. Option 1: No Local Storage

Probably the simplest arrangement is to have no local storage within FINDAS at the ICC (Figure 11). In this scheme all client communication with FINDAS must occur via an object server at the FSC. Performance is entirely dependent on the loading and reliability of the leased line but the system requires little infrastructure or support at the ICC.

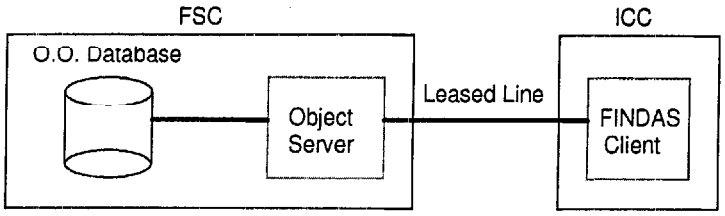


**Figure 11 - No Local Storage at ICC**

## 5.3.2. Option 2: Selective Local Storage

There are a number of possibilities for providing local storage of data at the ICC should this be preferred, and these differ in the implementation of the local store and the mechanism used for its management. In all cases client communication with FINDAS occurs via local object servers, which make the source of data, FSC master database or ICC local store, transparent to the user.

Given that the object servers give transparent access to the local store it would be sensible to provide the software for its management as part of the object servers as well (Figure 12). This software could selectively store and remove data obtained from the FSC according to a policy to be decided by the ICC. This might be based on explicit requests by users, first in - first out, frequency of access, or other strategies. It should be remembered, however, that the development cost of the system is likely to depend on the sophistication of the policy chosen.

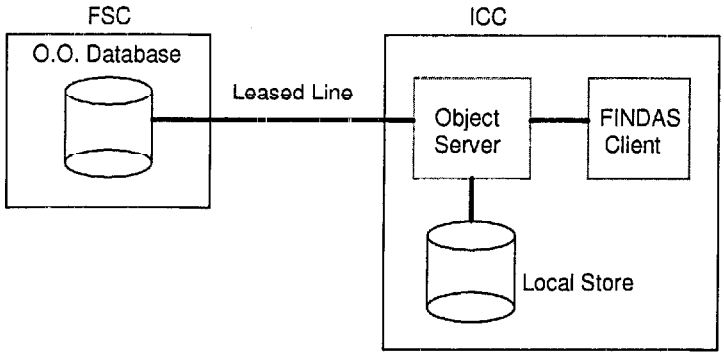


**Figure 12 - Selective Local Storage at the ICC**

In general, all FINDAS data locking and updates must occur at the FSC, in order to maintain the consistency of data across the system. However, for data produced in bulk at an ICC, such as ILT telemetry, a selective local store could be provided with a direct update. Of course, the object server would be responsible for ensuring that a copy of the data is provided to the FSC as well, but this arrangement could reduce the loading of the leased line by avoiding the

situation in which bulk data is uploaded to the FSC only to be downloaded a few days later into an ICC local store following a request for a rerun.

The simplest storage implementation for a selective local store would be to have the local data written as files on disk. This would require only standard system administration skills but is likely to be suitable only for simple types such as telemetry. A better implementation would be to use a local copy of the object database for storage. This would require greater initial expense but could provide a much more flexible system.

### 5.3.3. Option 3: Replication

Modern object database systems allow a database to be split into segments across many database servers and volumes while maintaining a single interface to the user. Some allow database segments to be replicated automatically to shadow databases, and this provides another possible means of local storage management. By careful design of the way the database is segmented at the FSC it would be possible to maintain shadow ICC data stores automatically by replicating the appropriate database segments to the ICC (Figure 13). The chief advantage of this approach is that the management of local storage would be obtained with little internal development effort. However, the database replication mechanisms are currently designed to maintain an exact, read-only copy of a database segment. Hence it might be impossible to delete old data from local storage without risking the integrity of the local database. In addition, database updates would have to occur strictly at the FSC, and in a real system there is likely be some delay for an update to propagate back to the ICC.

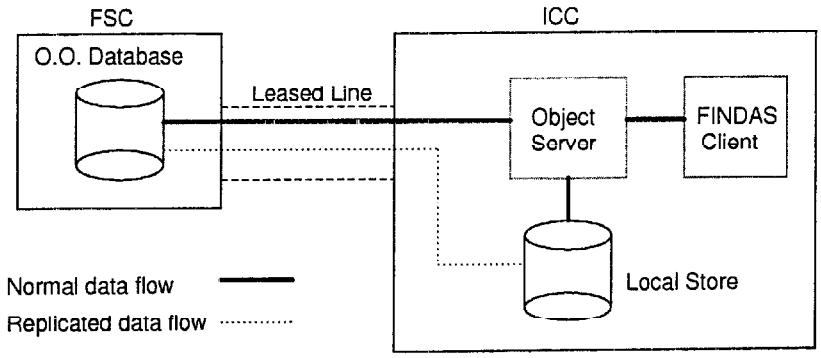


**Figure 13 - Local ICC Storage by Replication**

### 5.3.4. Recommendations

The techniques employed at each ICC are not fixed for the duration of the project, and the choice may well be affected by the way the relative costs of leased line bandwidth and large-scale hard disk storage evolve over time. It might be possible for ICCs to have high bandwidth connections and dispense with local storage altogether (Figure 11). However, we would recommend that the ICCs start with selective local storage (Figure 12) implemented on a local copy of the object database. This would be particularly convenient for the ILT phase and would offer the most flexibility for future changes.

It should be emphasised that the system will be constructed in such a way that architectural decisions will not affect the user. The implications of these choices will be further investigated during the Prototype Phase.

## 5.4. Access Control

### 5.4.1. Introduction

The FINDAS database must be protected: some data will be proprietary or sensitive and should only be viewed by authorised users. In general, the ability to alter data should be very strictly controlled. There are two aspects to FINDAS security. One aspect is that of gross access to the database and involves issues such as network and host security, and user authentication, for which there are well-known solutions and products. The second aspect concerns what fine-grained access a user should have, given that gross access to the system has been obtained, and it is this aspect that we deal with here. The basic issue is: which operations should a specific user be allowed to perform on a specific object? This is the subject of access control. The information required to enforce access control must be held within FINDAS in a manner which is both simple and efficient. There follows a candidate scheme for the provision of access control in FINDAS.

### 5.4.2. System Design

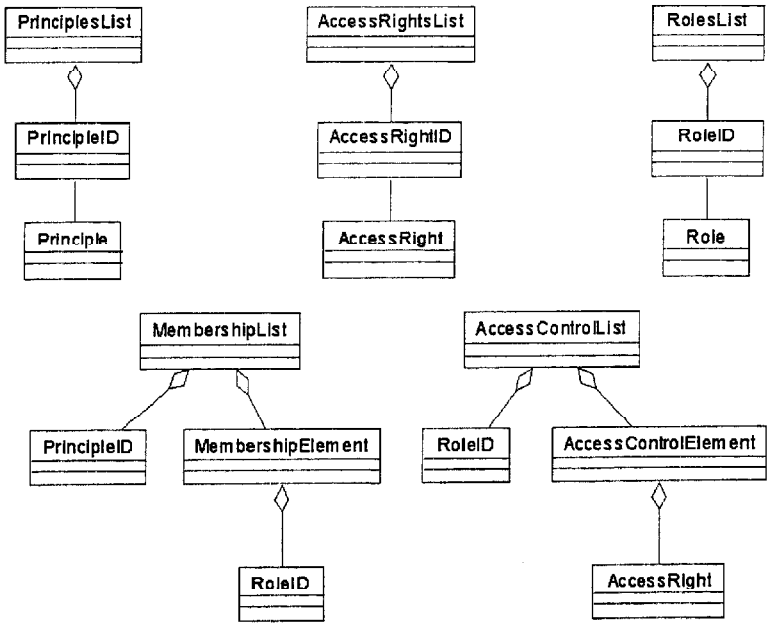


**Figure 14 - Access Control Classes**

There are five basic types of access control data that need to be stored in FINDAS (Figure 14). The first is the set of principals, which correspond to named authorised users. The second is the set of access rights, where each right defines a capability to perform an operation or group of operations on a database object. Examples of access rights are read, update and remove. The third is the set of roles. A role denotes a user or group of users: access rights to an object are conferred or denied according to the roles held by the principal (i.e. named user) requesting access. Examples of roles include administrator, scheduler and external user. The fourth is the set of roles and their corresponding membership, in terms of principals. The fifth is the set of roles and their corresponding access rights, which must be defined in some manner for every object in the database.

The first three access control structures are represented in the database by classes PrincipalsList, AccessRightsList and RolesList. These all have a very similar structure, and each one will have only a single instance in the database. In order to minimise the transfer of data around the system each member of these lists will generally be represented by a simple ID object, from which more detailed information may be extracted from the database if required. The last two structures are represented by classes MembershipList and AccessControlList. These two classes also have a similar structure: in both cases there is an intermediate Element class which enables their representation as a list of lists. However, while MembershipList will have only a single instance in the database AccessControlList will be instantiated for every non-lightweight object in the database. All of these classes will be assigned appropriate attributes and operations, the precise details to be determined during the design phase. The arrangement of classes given, with all data broken down into small elements held in lists, will allow the sets of roles and access rights supported to be modified with minimal impact on the system.

The control of access to an individual database object is determined by means of the access control list (ACL). An ACL is a list of access control elements, where each access control element is a list of the access rights corresponding to a given role. In general, every class will be associated with an ACL through inheritance from the base class FINDAS_Object. Where a class must be kept lightweight for performance reasons access may be defined by a parent container object. An example of this is the TMPacket class. Access to telemetry will be mostly controlled through the parent Downlink object.

It is envisaged that the global lists, given by unique instances of the first four classes above, will be cached wherever access control is enforced (i.e. the object servers). These objects will be available at global program scope: for example *::membershipList* might represent the cached copy of MembershipList. When an attempt is made to access a database object the object's ACL comes too, owing to the inheritance from FINDAS_Object. The decision to allow or deny the attached principal the requested access to the given object will then be decided according to an algorithm like that represented in Figure 15. The implementation of this algorithm could be achieved either directly in code or through the use of a COTS security framework, such as the CORBA Security Service.

```
bool accessAllowed( MembershipList membershipList,
                    PrincipalID principalID,
                    FINDAS_Object object
                    AccessRightID accessRequested)
{
        MembershipElement membershipElement;
        membershipElement = membershipList.membershipElement(principalID);

        if(principalID==object.getOwner()) membershipElement.add(principalID);

        RoleID roleID;
        AccessControlList acl = object.accessControlList();

        AccessControlElement ace;

        for(roleID in membershipElement)
                if(roleID in acl){
                        ace = acl. accessControlElement(roleID);
                        if(accessRequested in ace) return true;
                }

        return false;
}
```

**Figure 15 - Basic Access Control Algorithm**

Figure 15 says look for each RoleID contained in both the MembershipElement for the principal and the AccessControlElement for the role, and whenever there is a match grant access if the requested access is present in the AccessControlElement. The search over the MembershipElement forms the outer loop, as the Membership element is likely to have less members than the AccessControlList. The algorithm could be optimised in a number of ways, such as arranging the roles in the MembershipElement in order of decreasing privilege, so that the roles most likely to give access are searched first. The MembershipElement is first augmented with the owner role if the principalID of the requester matches that of the object's owner.

The provisional list of access rights for FINDAS objects is: create, remove, read, update, lock, promote and freeze. The meanings of create, remove and read are straightforward. The update access right is linked to lock, in that a user must hold a lock on an object in order to update it. The system may need to support different species of lock, in order to allow for competitive and collaborative working modes. The promote operation sets an object to be the default object of its class, so that any request not accompanied by a specific version number receives that object. The promote access right will have to be very strictly controlled. The freeze operation will render an object read-only. It is likely that a promote would always be accompanied by a freeze.

The provisional list of roles in FINDAS might include: Administrator, General Management, Instrument Calibrator, Scheduler, Real Time Operator, External User, Owner and Everyone. An administrator will have full access to the system. The Owner role cannot be held in the MembershipElement for a principal, but is

added on the fly during the access decision process (see above). The Everyone role effectively removes access control for the access right specified, so far instance we would expect a "public" object to be readable by Everyone. The External User role is distinguished from Everyone because an External User is a registered FINDAS user whereas Everyone really does mean anyone.

The discussion of access control has so far covered only the object level of granularity. At a coarser level of granularity the subject becomes that of gross access to the database (see first paragraph). as any collection of objects is still considered to be an object. The finer level of granularity is defined by the level of access allowed to the attributes and operations of objects. Good object oriented practice demands that direct access to attributes not be allowed, but be controlled through the access operations for the object, that is, the collection of operations of the form set_attribute and get_attribute. Access to operations can be achieved either by adding code to the sensitive operations of a class in order to force access control, or through the use of a suitable COTS security framework. It should be pointed out, however, that fine grained security should be enforced with discretion, as its overuse in a complex system can render the effective security administration difficult.

Related to the issue of fine grained security is the provision of completely general database queries via OQL. Object databases do not yet have built in security, in general, and OQL implementations allow an OQL query full access to the attributes of an object. The subject of encapsulation breaking by OQL has given rise to considerable debate, but it is generally held that enforcing object encapsulation for OQL would defeat the purpose of a query language. The job of ensuring the security of an QQL query therefore rests with the system designer. If a COTS security framework is used it might well be expected to apply the system security policy regarding object and operation access transparently at a lower level than the application. If a COTS product is not used access to attributes via OQL could be controlled by using the convention that all attributes begin with an underscore, and filtering all queries for underscores before allowing them to proceed. A further layer of filtration would also be required to ensure that a query does return objects to which a user should not have access.

In conclusion, to completely enforce security and allow OQL queries is difficult, and careful thought should be given to the question of raw OQL access and who gets it. We may well need greater detail in the set of requirements for access control.

## 5.5. Configuration Management

### 5.5.1. Introduction

The provision of flexible configuration control is a very important requirement for FINDAS. We need the ability to attach version information to any object or group of objects in the database. The version information might include the author, modification date and reason for the version. We need to be able to manage a tree of related versions, to make specific versions read-only, and to designate a default version. It is also highly desirable that an object can be placed under configuration control without having to modify its class structure, and that applications access versioned and non-versioned objects in exactly the same way, i.e. the ability to be part of a version should be an orthogonal property of an object. This type of facility can best be achieved using a specialised collection class, which we shall call Version. The Version class will form the basic tool for manipulating versions of objects: FINDAS clients may use it to construct new classes that enforce specific version management policies. either by direct delegation or by inheriting from Version and redefining its operations where required. There follows a explanation of a possible Version class (Figure 16) and its use (Figures 17 and 18), as well as an example of how it might be employed to enforce a version management policy for a real client (Figure 19).

### 5.5.2. System Design

The Version class is basically a collection to which objects may be added and removed. However, it is a collection that can exist in multiple simultaneous incarnations, each labelled with a version name. Effectively, each incarnation contains a separate copy of a contained object, and changes can be made to an object in one incarnation without affecting the copies in the others. When a programmer selects a working branch any reference to an object in the collection is made to point to the appropriate copy. It is envisaged that the interface to Version will include the following operations, most of which are defined to be capable of being overridden in a derived class:

```
class Version{
public:
        Version(String version_name)
        virtual bool add_object(void *new_object)
        virtual void *get_object(void *object_to_find)
        virtual Version *create_branch(String version_name)
        virtual Version *retrieve_branch(String version_name)
        virtual bool working_branch()
        virtual bool set_readonly(bool)
        virtual bool set_default(bool)
};
```

**Figure 16 - Interface to the Version Class**

A new Version object is created in C++ using the constructor *Version()*.The first version created is the root of the version tree. An object may be added to a version using *add_object*, which returns false if the object already exists in the version. A new branch may be created with *create_branch* and is initially an exact copy of its parent. Operations exist that enable a version to be made read only or default. The default version of an object is accessed automatically in any software where configuration control is not explicitly used. There follows in Figure 17 some code which creates a simple version tree in a very contrived fashion but illustrates well the use of the various operations. For clarity, the ODMG database access syntax has been omitted. Figure 18 depicts the state of the version tree at various points during the execution of the code.

```
1    Int A(123), B(10);
2    Version *root = new Version("v1.1");
3    root.add_object(A);
4    Version *v12 = root.create_branch("v1.2");
5    Version *v21 = root.create_branch("v2.1");
6    v12->working_branch();
7    Int *a21 = (Int *) v21->get_object(A);
9    v21->create_branch("2.2");
8    *a21 = 47;
10   v21->add_object(B);
11   A = 246;
12   {       Version *v11 = v12->retrieve_branch("v1.1");
13           Int *a11 = (Int *) v11->get_object(a);
14           *a11 = 369;
15   }
```

**Figure 17 - Example Code to Illustrate the Use of Class Version**

In line 1 we define two simple objects (of a class implementing integers), and in line 2 we instantiate a Version object. This first version forms the root of the version graph and we call it "v1.1". As the root is the only version that currently exists it must be the working version. In line 3 we place an initial copy of object A (=123) into the version, and in lines 4 and 5 we create two new versions, "v1.2" and "v2.1". The root being the working version means that both versions v1.2 and v2.1 branch from the root of the tree, and start off identical to the root, i.e. containing one object A with value 123. The current state of the tree is given in Figure 18(a). In line 6 we set the working branch to v1.2. Lines 7-9 illustrate how we can manipulate v2.1 even though v1.2 is the working branch, and they create a new branch "v2.2", change the value of object A to 47 and add object B. Version v2.2 is explicitly created under v2.1, and as its creation precedes the changes made to v2.1 it is not subject to those changes (Figure 18(b)). Line 11 changes the value of A in the working version (v1.2) to 246. Finally, lines 12-15 illustrate how, given any version in the graph and an object present in the tree, one can retrieve another version and make a change to its copy of the object. In this case we change the value of A to 369 in the root version (Figure 18(c)).

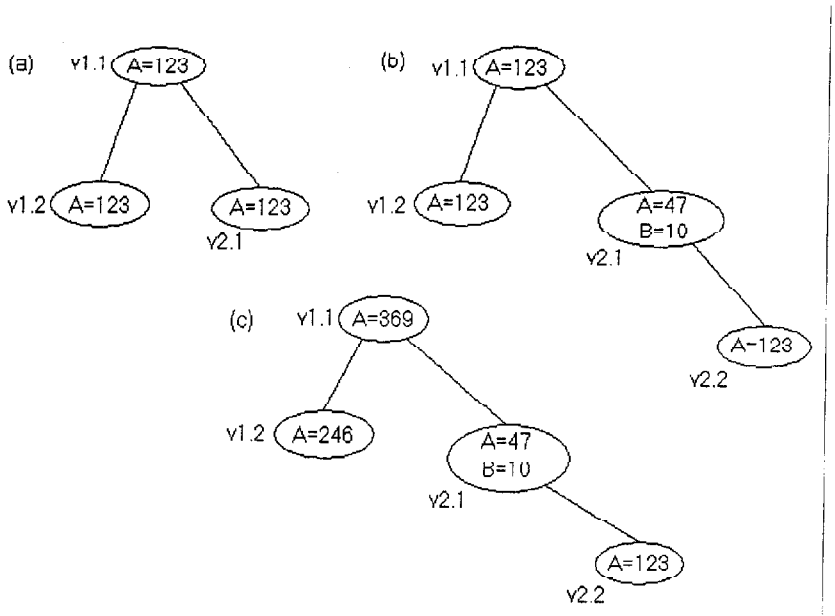


**Figure 18 - Some Intermediate Stages in the Version Control Example**

In the FINDAS system we may often require to enforce a particular configuration control policy on a given class of objects. For example, we might wish to enforce that the creator and creation time of a branch be recorded, that only an authorised user can define the default branch, and that the default branch is read only. This can be achieved quite simply by creating a new class MyVersion in the database, which inherits from Version and overrides some of its operations (Figure 19).

As before, this code relies on an underlying object persistence, provided by the database, but we have not explicitly shown the ODMG syntax. The *accessAllowed* function and its arguments will be available in any process where access control is enforced (see previous section). The *promote* right exists specifically to control the setting of default versions in the database. The first three parameters to *accessAllowed* enable the function to determine whether this specific user is allowed the requested access to *this* object (in this case, the Version object).

```
class MyVersion : public Version{
        PrincipalID creator;
        Time creation_time;
public:
        MyVersion(String version_name) : Version(version_name){}
        Version *create_branch(String version_name);
        bool set_default(bool);
};


MyVersion *Version::create_branch(String version_name){
        MyVersion *new_version = new MyVersion(version_name);
        *((Version*) new_version)
                = *Version::create_branch(version_name);

        new_version->owner = getCurrentUser();
        new_version->creation_time = getCurrentTime();

        return new_version;
}


bool MyVersion::set_default(bool switch){
        if(accessAllowed(membershipList,user(),this,AccessRightsList::promote){
                Version::set_default(switch);
                Version::set_readonly(switch);
                return true;
        }
        else return false;
}
```

**Figure 19 - Implementation of a Sample Configuration Control Policy**

Although this is only a basic example the same mechanism could be used to enforce the much more sophisticated configuration control policies that will be required for different FINDAS objects.

# 6. CONCLUSIONS AND RECOMMENDATIONS

## 6.1. Overview

As a result of the analysis done since the submission of our original proposal we conclude that the overall FINDAS design is feasible and will be able to deliver the benefits enumerated in Table 1 on page 13. The work performed will provide a solid basis for rapid progress during the development of the FINDAS Prototype system.

In this document we have given a unified perspective of the proposed FINDAS system based on diverse inputs. The main themes covered have been the concept and the architecture. Sections 3 and 4 have provided a thorough description of the FINDAS concept from a domain point of view. Section 5 has covered the FINDAS architecture with the aim of realising certain key requirements. Alternative design options have been dealt with and recommendations have been made, both for the final system and for issues that need further investigation during the Prototype Phase. A summary of the proposed architecture, with expanded detail for the FSC and ICC3, is given in Figure 20, which corresponds to the recommendations of section 6.4.

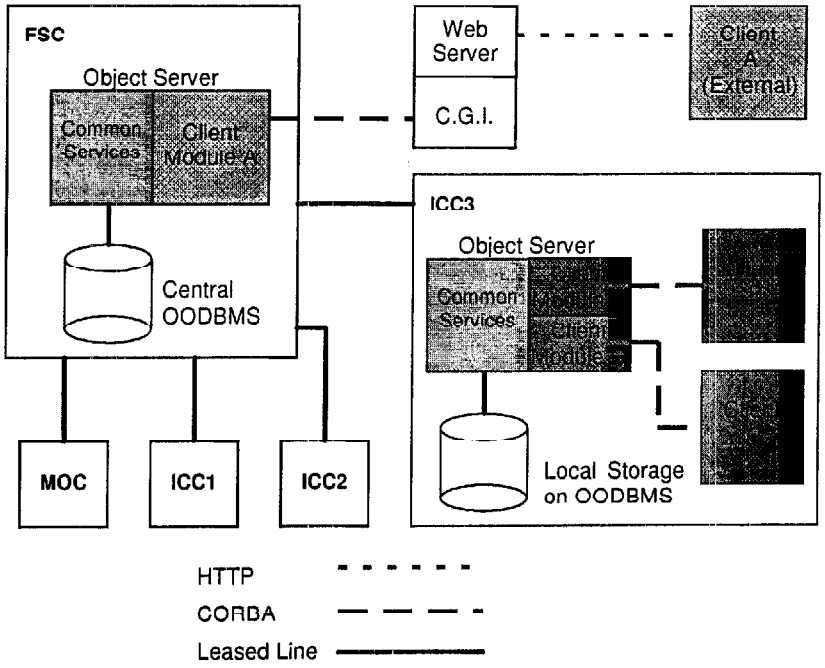


**Figure 20 - Summary of FINDAS Architecture**

In the following sections we recapitulate the key points arrived at in the main text concerning the specific areas of risk, cost drivers and recommendations. It should be pointed out that it may be necessary to trade off performance against cost in some areas, e.g. real time data availability against provision of local storage.

## 6.2. Areas of Risk

- No significant benefit over re engineering of existing ground segment systems - this is extremely unlikely considering that FINDAS offers a single, unified system making use of object oriented techniques and the latest COTS tools
- No seamless transition between phases - this will be mitigated by having a well designed database schema, possessing a significant core element in which is captured the entities/relationships common to all mission phases
- Lack of database scalability - object database technology can already handle the predicted final archive volume (2.5 Terabytes, including bulk data) and its capabilities can only increase over the lifetime of the project
- No adaptability or flexibility - the use of object oriented techniques and middleware, and the ability to make changes to the database schema during all project phases, ensure that FINDAS will meet evolving requirements
- No durability - the adoption of standard interfaces such as CORBA, ODMG and HTML ensures that no part of the system depends on a single vendor and that migration paths will exist should new standards emerge
- No robustness - the system must be designed to provide resilience to failure and misconfiguration, and to provide sanity checking prior to update
- Inadequate data transfer performance - the transfer of data between the FSC and other centres must be sufficiently rapid: this concerns the performance of the COTS framework used to build the system as much as crude network data rate and will be investigated during the Prototype Phase

## 6.3. Cost Drivers

- Guaranteed system availability - this can only be bought by duplicating facilities, potentially a major cost driver
- CORBA versus ODMG - choice of middleware to database interface is not clear-cut: development time is expensive, so it is important to determine which interface allows the object servers to be updated with minimum effort following a change to the database schema
- Provision of local storage - this is essential to providing a system which provides flexible access throughout all mission phases and is not totally dependent on the availability of network links between centres

## 6.4. Recommendations

We recommend:

- that FINDAS be based on a leading object oriented database: this will allow the development of a persistent object system in leading object oriented languages with the minimum of additional effort and training costs
- the adoption of a three-tier client server architecture
- the use of either CORBA or ODMG as the interface standard between middleware and database: these are the only mainstream standards available for *cross-platform* persistent object development
- that the choice between CORBA and ODMG be investigated during the Prototype Phase
- the use of CORBA as the interface standard between middleware and clients
- that the user interface standard be HTML
- that each ICC implements local storage on its own copy of the object oriented database: this provides maximum flexibility in the face of changing future demands
- that the issue of how to apply access control to OQL database queries be addressed during the Prototype Phase

# APPENDIX A – GLOSSARY OF TERMINOLOGY

Where a term corresponds to a FINDAS class a more detailed description will be found in Appendix C.

AOT - a template for a valid operating mode of an instrument

APH - spacecraft attitude pointing history

Auxiliary Data - files necessary to correctly process TM, e.g. APH, orbit file

CCS - the commands to be sent to the spacecraft for one planning period

Celestial Constraints - see DBOB

DBOB - the parts of the sky observable in a planning period

Downlink - a class that gives access to all downlink data from a single uplink

Instrument Characteristics - structure of the TC/TM for an instrument

Instrument Logic - a software module that emulates an instrument

Interactive Analysis - detailed analysis of instrument telemetry, a process

IS Report - report on an uplink from RTA/QLA

Orbit File - details of the spacecraft orbit

Pipeline - software for producing standard products from observation telemetry

Plan - a set of uplinks for one planning period, with start and stop times

Planning - allocation of start and stop times to a set of uplinks, a process

Proposal - the unit of access to an instrument

Proposal Generation - software for entering proposals

Proposal Handling - software to check proposals and make them schedulable

RTA/QLA - real time and "quick look" analysis of instrument telemetry, a process

Scheduling - conversion of a plan into a CCS, a process

Strategy Weights - a coordinating policy for observations

Uplink - the unit of instrument usage; a single exposure of an instrument

# APPENDIX B – DATA FLOW IN FINDAS

## B.1 Process Inputs Required from FINDAS

| | |
|---|---|
| **Proposal Generation**<br>Proposal / test procedure<br>Blank AOTs<br>Instrument logic | **Proposal Handling**<br>Proposal / test procedure<br>Instrument logic |
| **Planning**<br>Uplink<br>Planning template<br>Rejected plan<br>ICS & macro lists<br>Instrument logic<br>Strategy weights<br>DBOB<br>Time slew predictor | **Pipeline**<br>Downlink<br>Auxiliary data<br>IS report<br>Calibration data<br>Instrument characteristics |
| **Scheduling**<br>Plan<br>Instrument commands | **RTA/QLA**<br>Downlink<br>Auxiliary data<br>TC history<br>Calibration data<br>Instrument characteristics |
| **Flight Control System**<br>Test (unscheduled)<br>Memory image<br>PCS list & CCS | **Test Control System**<br>Test procedure<br>Instrument characteristics<br>Memory image |
| **Instrument Logic**<br>Calibration data | **Interactive Analysis**<br>as Pipeline, but also IS log |

## B.2 Process Outputs Stored on FINDAS

| | |
|---|---|
| **Proposal Generation/Handling**<br>Proposal / test procedure<br>Uplink | **Interactive Analysis**<br>Calibration product<br>Calibration data<br>Test product |
| **Planning**<br>Uplink (planned or rejected)<br>Plan<br>Instrument commands | **Pipeline**<br>Standard product<br>Report |
| **Scheduling**<br>CCS (copy of file sent to FCS)<br>Plan (rejected) | **RTA/QLA**<br>IS report<br>RTA/QLA product<br>IS log |
| **Flight Control System**<br>Planning template<br>Telemetry<br>Auxiliary data<br>Command history<br>Operations log | **Test Control System**<br>Telemetry<br>Auxiliary data<br>Command history |

## B.3 General Data Types Stored on FINDAS

| | |
|---|---|
| **Documents**<br>e.g. ILT documents | **Software**<br>e.g. pipeline |
| **Catalogues**<br>e.g. astronomical catalogues | **Trend Analysis**<br>e.g. trend analysis results |

# APPENDIX C – FINDAS DATA DICTIONARY

In the following the names of classes begin in upper case.

It should be noted that every class (with the possible exception of TM Packet) inherits from FINDAS Object and therefore contains the attributes and operations of FINDAS Object implicitly. In addition, every class contains access operations (get/set) for its specific attributes, as well as operations allowing navigation to all related objects (see figures 5-8).

### Astronomical Observation Template (AOT)

A template for a valid operating mode of an Instrument. There is an AOT corresponding to each Standard Product.

An instantiated AOT is an AOT object that has been completed and is available for planning. It is either part of a Test Procedure or relates to a Proposal that has been checked and approved by the FSC staff and OTAC.

Proposal Generation is a tool to allow astronomers to complete or manipulate an AOT in order to define or refine an Observation. The mode attribute determines the type of an AOT. The other attributes of an AOT define an Observation in astronomical and instrument mode terms. It is the job of Proposal Generation to verify the sanity of entered attributes.

### Attitude Pointing History

Contains a File object giving the record of spacecraft attitude for one planning period.

### Calibration

A privileged type of Uplink object used by an calibration scientist to calibrate an instrument. A Calibration contains a File object which holds the instructions for the calibration, defined using the instrument scripting language. Its attributes include: time of required execution.

### Calibration Data

Contains a File object which may be generated as a result of the analysis of a Calibration or Test Product. It is used to update the uplink and downlink client software. Its attributes include: the identity of the Product from which it was generated, the version of interactive analysis software used, and cautionary comments from the calibration scientist.

### Calibration Product

See Special Product.

### Catalogue

Contains a File object that holds data in a specialised external format. A catalogue object contains attributes that describe the structure of the File and the software required to manipulate it.

### Central Command Schedule (CCS)

Contains a File object which consists of the commands to be sent to the spacecraft to cover a single planning period. It is produced by the scheduling system and used by the flight control system where it is converted into the telecommand stream. Its attributes include the planning period ID and the version of the scheduling software used to generate it.

### Database of Observable Bins (DBOB or Celestial Constraints)

Contains a File object giving the accessible region of sky for a given planning period divided up into bins. The DBOB is produced by the flight dynamics system and is used for planning purposes. It contains an attribute giving the flight dynamics version identifier.

### Document

A Document object holds source and presentation formats of a document in the form of File objects. The presentation format is PDF. The Document attributes include: document reference, document type, source institute, source format, and a list of keywords to be used for indexing. The document type could be one of: email, minutes, memo, user manual, procedure, table etc. The source institute could be one of: General, ICC1, ICC2, ICC3, SOC or MOC. The source format gives the software used to generate the source, e.g. Microsoft Word 95.

### Downlink

A Downlink object implements a relationship between an Uplink object and the telemetry packets generated as a result of executing it, and provides a navigation path to everything derived from the telemetry.

### File

A basic class to hold files. All documents, catalogues, data sets etc. entered into the database will be held in File objects. Its attributes include: title/name, format, date of issue, source, and version. Its operations include: store, update and retrieve. The store operation stores the actual file in FINDAS. The update operation creates a new version of the File object containing the modified file. The retrieve operation transparently retrieves the file, either from the central database at the FSC or from local storage.

### FINDAS Object

The base class for all FINDAS objects. Contains a reference to an access control list, a version number, date of generation, owner, and a number of operations that may be overridden in derived classes. The display operation produces an HTML page representing the object and its relations to other objects expressed as HTML links. The access allowed operation determines whether object access should be granted to the given user (see section 5.4).

### ICS List

An object which defines the instrument command keywords and their parameters.

#### Instrument Characteristics

Contains a File object listing telemetry parameters, their limits, and response lookup tables. It is produced by the instrument teams. During ILT it is required by the RTA/QLA and test control systems. During the operational phase it is required by RTA/QLA and the flight control system.

### Instrument Commands

Contains a File object giving a sequence of instrument commands (and any necessary spacecraft keywords) in a format acceptable to the ILT test control system and the scheduling system.

### Instrument Logic

Contains a Software object which simulates the behaviour of an instrument. The Instrument Logic is used in the following processes: proposal generation, proposal handling and planning. Its attributes include: calibration data ID.

#### Instrument Station (IS) Log

Contains a File object giving a detailed list of the messages produced during RTA/QLA. It is used by interactive analysis to monitor the behaviour of instruments. Its attributes include: start and stop times of the telemetry packets analysed.

### Instrument Station (IS) Report

Contains a File object giving a report which is provided by RTA/QLA to interactive analysis and the pipeline. The report contains the results of the initial inspection of the telemetry associated with a Downlink object.

**Memory Image**

Contains a Software object holding an onboard software element for an instrument. It is provided by an instrument team and uploaded by the flight control system, or used on the ground for comparison purposes.

**Observation**

An Observation is an Uplink object resulting from a Proposal by an astronomer. An Observation contains a set of AOTs, one for each instrument, in which some parameters have been entered by the astronomers and others calculated by Proposal Generation, e.g. target time. An Observation will usually result in the generation of a Standard Product. Its attributes include: scheduled date, executed flag. Its operations include: list AOT types.

**Orbit File**

Contains a File object giving the record of orbital position for planning period.

**PCS List**

Contains a File object giving a list of the permanent instrument command sequences which is provided by an instrument team. A PCS may be uploaded at the MOC if required. It is ready instantiated, with all parameters set in advance. Attributes include a list of the PCS names included in the File.

**Plan**

A collection of Uplink objects for a single planning period, defining the start and stop times for each Uplink. A Plan has an operation which yields a File object in a form acceptable to the scheduling system. Its attributes include: reject flag.

**Planning Template**

Contains a File object holding the observing slots available in a single planning period. The Planning Template is provided by the flight control system and is used for planning purposes. Its attributes include: planning period ID.

**Product**

The result of the detailed processing of the telemetry from an Uplink. Processing of Observations results in Standard Products. Processing of Calibrations and Tests produce Special Products. Its attributes include: pipeline software version, auxiliary data used, calibration data used, AOT type, quality flags.

**Proposal**

A Proposal consists of a set of Uplinks together with a text justification. Proposals are entered by users in order to obtain or justify the use of instrument

time using a tool called proposal generation. The OTAC and FSC staff, and certain internal users such as calibration and test scientists, use a tool called proposal handling to obtain privileged access to the proposal database. Its attributes include: originator list, call number, grade, allocated time.

A Test Procedure is a special type of Proposal entered by a calibration or test scientist, and usually contains Calibrations and Tests.

#### Registered Document

A subclass of the Document class, a Registered Document holds information on documents that have been registered but are not held on the database. It will contain as attributes the registration details, location and borrower information of any hard copy document, and the URL (or equivalent) to enable a soft copy to be obtained.

#### Report

Contains a File object giving text information associated with a Product. It may contain the final results of a Test, additional comments on the processing of a Standard Product, any interactive input used in the generation of a Special Product, and any other relevant material relating to the Product.

#### RTA/QLA Product

An overview product provided by RTA/QLA from initial analysis of telemetry. Its attributes include: RTA/QLA version, auxiliary data used.

#### Software

A generic class to hold all ground segment software associated with the mission. A Software object has attributes recording compiler version, operating system, etc., and gives access to source code, compiled code, makefiles, and all other data required to regenerate or modify a given release of software.

#### Special Product

A Product that has the capacity to be used to generate Calibration Data. Calibration Products and Test Products are both Special Products (see also Report).

#### Standard Product

A Product generated as a result of an astronomical observation (TBC).

#### Strategy Weights

Contains a File object produced by the mission planner at the FSC to guide the usage pattern of instrument resources during the course of the mission. The

Strategy Weights are an input to the planning system. Its attributes include: applicable period.

**Test**

A privileged type of Uplink object used by an instrument test. A Test contains a File object which holds the instructions for the test, defined using the instrument scripting language. Its attributes include: time of required execution.

**Test Procedure**

See Proposal.

**Test Product**

See Special Product.

**Time Slew Predictor**

The Time Slew Predictor contains a Software object holding software produced by the flight dynamics system and used by the planning system. It contains an attribute giving the flight dynamics version identifier.

**TM Packet**

A class to hold a single telemetry packet. Each Telemetry Packet will bear a spacecraft subsystem identifier and a spacecraft generated timestamp, as well as a unique packet number. The TM Packet may be implemented as a lightweight object that does not inherit from FINDAS Object.

**Trend Analysis**

Contains a File object holding an analysis of telemetry across Uplink boundaries and may be required to characterise the behaviour of certain instrument subsystems. Its attributes include: applicable period, parameter analysed.

**Uplink**

The basic unit of scientific usage. It provides the vehicle by which instrument exposure requests are entered into the system, and provides relations to all associated objects in the uplink segment. Through its relation to a Downlink object it gives access to all associated objects in the downlink segment. An Uplink may correspond to an Observation, a Calibration or Test. Its attributes include: primary instrument(s), schedulable, duration, scheduled time.