Visible & Infrared Survey Telescope for Astronomy
Project Office

Tel: +44 (0)131 668 8411
Fax: +44 (0)131 668 8412
Email: vista@roe.ac.uk,
WWW: http://www.roe.ac.uk/atc/vista

Document Title: VISTA Software Management Plan

Document Number: VIS-PLA-ATC-00150-0006

Issue: 2

Date: 27 Sept. 2001

Document Prepared By:	**J M Stewart** **Software Engineer**	**Signature and Date:**	27 Sept. 2001
Document Approved By:	**S Craig** **Project Engineer**	**Signature and Date:**	01/10/01
Document Released By:	**A McPherson** **Project Manager**	**Signature and Date:**	3/10/01

The information contained in this document is strictly confidential and is intended for the addressee only. The unauthorised use, disclosure, copying, alteration or distribution of this document is strictly prohibited and may be unlawful.

Change Record

Issue	Date	Section(s) Affected	Description of Change/Change Request Reference/Remarks
1	24 June 2001		Submitted to Close Out Review July 2001
2	27 Sept. 2001	2.2, 3.1, 5.4	Packages and modules defined more consistently, scope of incremental delivery broadened, other clarifications.

Table of Contents

1. Introduction

1.1 Overview

This document describes how the software aspect of the VISTA Project will be managed in Phase B. It describes the management both of the telescope control software and instrument software, which have rather different characteristics due to the degree of software reuse from the VLT. Instrument software includes both control and dataflow software.

1.2 Scope

The scope of this document is the management of the development of all software to be used at the VISTA telescope situated at ESO's Cerro Paranal site. It excludes any software that may be used either at ESO HQ in Garching or in the UK, e.g. the final data reduction pipeline and catalogue production, though the input data to these processes depends on the software referred to in this document.

Software includes not only compilable code, but also VISTA specific configuration data, scripts, recipes etc. that may be used with software systems, responsibility for which resides elsewhere.

The document does not define a Work Breakdown Structure, but uses likely WBS items to illustrate the generate points.

1.3 Applicable Documents

The following documents shall be considered part of the current document. Where any conflicts occur, the current document shall be considered the superseding document.

AD01 *VISTA Project Tracking and Control Plan*, VIS-PLA-VPO-00001-0008, Issue 2, October. 2001.

AD02 *VISTA Phase B Project Procurement Plan*, VIS-PLA-VPO-00001-0006, Issue 2, 25 Sept. 2001.

AD03 *VISTA Documentation Management Plan*, VIS-PLA-VPO-00001-0001, Issue 3, 26 September 2001.

AD04 *VISTA Technical Specification*, VIS-SPE-ATC-00000-0003, Issue 2, 26 October 2001.

AD05 *VLT Software Management Plan*, VLT-PLA-ESO-00000-0006, Issue 2.0, 21 May 1992.

AD06 *VLT Software Programming Standards*, VLT-PRO-ESO-10000-0228, Issue 1.0, 10 March 1993.

AD07 *VISTA Management Plan,* VIS-PLA-VPO-00001-0002, Issue 3, October 2001.

AD08 *VLT Software Documentation Review Procedure*, VLT-PRO-ESO-10000-0201, Issue 1.0, 12 March 1992.

AD09 *Guidelines for the Development of VLT Application Software*, VLT-MAN-ESO-17210, 0667, V1.0, 3 December 1997.

AD10 *DICB - Data Interface Control Document*, GEN-SPE-ESO-19400-0794, V1.1, 25 November 1997.

AD11 *VLT Instrument Software Specification*, VLT-SPE-ESO-17212-0001, V2.0, 23 February 1995.

AD12 *Data Flow for VLT Instruments Requirements Specification*, VLT-SPE-ESO-19000-1618, Issue 1.0, 21 April 1999.

AD13 *VLT Instrumentation Software Specification*, VLT-SPE-ESO-17212-0001, Issue 2.0, 12 April 1995.

AD14 *INS Common Software Specification*, VLT-SPE-ESO-17240-0385, V2.1, 15 July 1996.

AD15 *Configuration Management Module User Manual*, VLT-MAN-ESO-17200-0780, Issue 1.2, 10 March 1997.

AD16 *Data Interface Control Document*, GEN-SPE-ESO-19400-794, Issue 1.1, 25 November 1997.

AD17 *VLT Software Configuration Control Plan*, VLT-PLA-ESO-00000-0004, Version 1.0, 20 December 2001.

1.4 Reference Documents

The following documents are relevant and are referenced in the text of the current document, but are not considered part of the current document.

RD01 *VISTA Science Requirements Document*, VIS-SPE-VSC-00000-0001, v2.0, 26 October 2000.

RD02 *VISTA Operational Concepts Document*, VIS-SPE-VSC-00000-0002, v1.0, 28 March 2001.

RD03 *VISTA Computer Hardware Architectural Design*, VIS-SPE-ATC-00150-0002, Issue 2, October 2001.

RD04 *VISTA Software Architectural Design*, VIS-SPE-ATC-00150-0002, Issue 2, October 2001.

RD05 P Ward and S Mellor, *Structured Development for Realtime Systems*, 1985, Yourdon Press.

RD06 P. Kruchten, *The Rational Unified Process, an Introduction*, 1999, Addison Wesley Longman, Inc.

1.5 Definitions

Contractor	The organisation made responsible for a work package, e.g. a commercial company or an academic institution
Module	A piece of software (code and documentation) able to perform functions and having an interface available to an external user to access the functions provided. It is the basic unit for planning, project control and configuration control (AD06 Section 2.2). Examples are M2 Control or IR Camera Instrument Control System.
Package	The highest level of software subdivision, e.g. the Infrared Camera Software package, forming a logical collection of modules.
Workpackage	A unit of work given to a conrtactor, possibly a complete package or perhaps just a module within a package.

1.6 Abbreviations and Acronyms

ATR	Acceptance Test Review
CCS	Central Control Software (VLT infrastructure software)
CMM	Configuration Management Module
FDR	Final Design Review
GUI	Graphical User Interface
ICD	Interface Control Document
ICS	Instrument Control System

IR	Infrared
LCC	LCU Common Software (VLT infrastructure software)
LCU	Local Control Unit (VME/VXworks based computer)
NA	Not applicable
OCDD	Operational Concepts Definition Document
OO	Object oriented
OS	Observation System
PDR	Preliminary Design Review
QA	Quality assurance
SA/D	Structured Analysis and Design
SRD	Science Requirements Document
SPR	Software Package Responsible
UML	Unified Modeling Language
VPO	VISTA Project Office
VTS	VISTA Technical Specification
WBS	Work breakdown structure
WFS	Wavefront sensing

2. General Considerations

2.1 Reuse of VLT Software

The VISTA Project is somewhat unusual in the large degree to which existing software will be reused. The factors that drive this are

- VISTA will be on the same site as the ESO VLT (Cerro Paranal)
- VISTA will use certain Paranal computing services
- VISTA software will be maintained by ESO staff
- VISTA will be operated by ESO staff

These factors cause major constraints to be placed on VISTA software. Although these constraints may have certain disadvantages, the net benefit is very positive, since much less

software development (and therefore cost) will be required. The constraints can be summarised as:

- wherever feasible VISTA will reuse existing VLT applications
- where an existing VLT application is not exactly usable on VISTA, it will be modified for VISTA if possible
- modifications to VLT software will be kept as low in the software hierarchy as possible
- VLT infrastructure software will be used for all new or modified VISTA applications
- VLT software services will be used for VISTA where possible.

2.2 Packages and Modules

It is convenient to envisage the software as comprising modules and packages. A module is a piece of software (code and documentation) able to perform functions and having an interface available to an external user to access the functions provided. It is the basic unit for planning, project control and configuration control. This definition is as used for VLT software (AD06 Section 2.2), though the interpretation in VISTA's context may be slightly different. An example of a telescope control software module is M2 Control System.

A package is a higher level of software subdivision. A package is a logical collection of modules, which must be integrated and can potentially be delivered from a single contract.
The following are possible packages and modules for VISTA:

Telescope software packages

- Telescope Control System
- M1 Control System
- M2 Control System
- Enclosure Control System
- Axis Control Systems
- Guide & Wavefront Sensing System

- Infrared Camera Software Package comprising modules:
  - Observation Software (OS) - top level control
  - Instrument Control Software (ICS) - mechanism control
  - Detector Control System (DCS) - data acquisition

- Visible Camera Software Package comprising modules:
  - Observation Software (OS) - top level control
  - Instrument Control Software (ICS) - mechanism control
  - Detector Control System (DCS) - data acquisition

- Data Reduction Pipeline - off-line, data driven

2.3 Differences between Telescope and Instrument Software

VISTA software includes both instrument software and telescope software. There are some important distinctions between these two sets of software that are taken into account in this management plan.

2.3.1 Telescope Software

The telescope control system is divided into packages at the level of, e.g., M1 Control and Guiding. Each such package will in general have a different vendor responsible for its implementation. The VPO, directly or indirectly, is responsible for integrating these into a single system.

Each VISTA telescope software package has an existing fully operational VLT counterpart. Interfaces between telescope software VISTA packages will differ only slightly, if at all, from their VLT counterparts. The similarity of code between VISTA and VLT systems will vary between complete and negligible. (However there will always be full use made of VLT infrastructure code.)

2.3.2 Instrument Software

Instrument software packages will exist at a much higher level i.e. the IR Camera Software Package and the Visible Camera Software Package. Each package is divided into modules, e.g. the OS and ICS (see Section 2.2). This allows the VPO to assign work at a higher level of modularity.

Each instrument package and each module therein have counterparts in the VLT system, but the packages and modules to be implemented for VISTA are new systems. The scope of the modules' functionality and the interfaces between them are the same as for the VLT systems, but the details and implementations will be different since the VLT does not have instruments with the same functionality as VISTA's. (Although VISTA instrument software packages and modules will be different to the VLT's, they will fully reuse the VLT's infrastructure software and follow the structure of applications, e.g. though the use of templates.)

3. Software Life Cycle

3.1 Incremental Delivery

During the design phase, incremental delivery may be used to deliver prototypes to test aspects of the design.

During the coding phase, the code and associated documentation shall be released incrementally. The initial release shall implement the most important interface for that particular module, e.g. the GUI for a high level module or the device interface for a device driver. Successive releases shall implement increasing amounts of functionality and interfaces to other systems.

Each incremental release of a module shall be integrated with other modules within the package, so that errors resulting from any phase (requirements, analysis, design or coding) can be caught early. The Rational Unified Process RD06 provides background on incremental delivery (or iterative development), especially in the context of object oriented development but applicable also to structured analysis and design.

The work performed between successive incremental deliveries will primarily be coding, but the experience gained may also cause changes to the design, e.g. if it becomes clear that a design change would allow the product to be delivered earlier or more cost effectively. Changes to earlier phases, e.g. analysis and requirements, may also be demonstrated to be beneficial and it is an important advanatage of iterative development that any such changes are identified earlier rather than later. However if changes to requirements or interfaces are proposed, they shall be considered as part of a formal change management process (section 6.5).

3.1.1 Requirements

3.1.2 Telescope Software Requirements

General requirements for VISTA (i.e. not limited to software) are defined in the Technical Specification (AD04), derived from the Science Requirements (RD01), which provides further background. Operational Concepts (RD02) are also important in determining the software requirements, since they describe in large part how the telescope and its two instruments will be driven from software systems.

It is not appropriate to generate an overall VISTA Software Requirements Document, because of the major constraint and benefit of using the VLT Control System. A more WBS oriented approach has been taken therefore. The VLT system architecture has been interpreted in the VISTA context and individual systems treated individually in one of the following ways.

- Evaluate Corresponding VLT Package against VISTA Requirements
- Generate Software Specification from VISTA Subsystem Specification
- Generate Software Requirements for VISTA Package

These approaches are described below (3.1.3).

In some cases, e.g. the TCS, the functionality of the VLT system is checked against VISTA's requirements as specified in the VTS and OCDD. In other cases, e.g. the Infrared Camera, requirements are determined in the normal manner.

3.1.3 Instrument Software Requirements

Package requirements are the responsibility of the VPO.

Inputs to this stage are

- Science Requirements (RD01)
- Operational Concepts (RD02)
- VISTA Technical Specification (AD04)

Outputs are

- Infrared Camera Software Requirements
- Visible Camera Software Requirements

Because of the close similarities in the two instruments these requirements may be written as a single document identifying common requirements and requirements specific to each instrument. (Alternatively they may be separate documents with extensive cross-referencing or cutting and pasting.)

3.2 System Design

Another relevant Phase A deliverable is the VISTA Software Architectural Design. This will conform with the VLT architecture (AD11), which defines the functionality of the different instrument software modules. Because of this, separate Module Requirements will not be formally generated, but Module Functional Specifications and interfaces will be (below).

The software system design results in a breakdown into subsystems based on the VLT software system design. The VISTA software system design identifies:

- software systems
- interfaces
- network architecture
- constraints

It does not specify the details of the functionality of each system nor the details of each interface. These will be specified on a system by system basis. The VISTA hardware and software system design is described in (RD03) and (RD04).

3.3 Analysis

3.3.1 Telescope Software Analysis

Based on the system design, workpackages will be specified for implementation by contractors, e.g. commercial companies or academic institutions. Before placing contracts, requirements and specifications will be defined. As mentioned above, different approaches will be taken for different packages, depending largely on the degree of reuse of VLT software.

3.3.1.1 Evaluate Corresponding VLT Package against VISTA Requirements

It is a constraint (and a major benefit) that certain VLT packages will be reused on VISTA. A important example is the Telescope Control Software. Rather than write requirements from scratch, the adopted approach is to check the functionality of the VLT system against VISTA system requirements as defined in the VTS and OCDD.

From such a check, the following conclusions may arise, each requiring a different solution:

1) Package can be used as is - no further development is necessary.
2) Package can be used with minor modifications - ESO will modify the package either in a general sense for all the Paranal telescopes or specifically for VISTA. In the former case there may be no cost to VISTA.
3) Package can be used with major modifications - a contract will be placed, perhaps with ESO or the contractor who implemented the package for the VLT.
4) Package must be rewritten - a contract will be placed with a contractor.

Cases 3) and 4) then become one of the two cases below.

3.3.1.2 Generate Software Requirements for VISTA Module

In cases where VISTA requires a new system, it is appropriate to start a package at the requirements stage. Instrument software most clearly comes into this category, but a few telescope systems may also, e.g. guiding and wavefront sensing since the techniques used to implement these functions may be very different to those used on the VLT.

3.3.1.3 Generate Module Software Specification from VISTA Subsystem Specification

In many cases, the specification of module software requirements and functionality will flow down from the functional and performance requirements of a multi-disciplinary system. For example, the requirements and specification of the guider software depend primarily on the requirements of the guider system, taking into account the constraints imposed by the chosen VISTA software architecture (i.e. the adoption of the VLT architecture).

For multidisciplinary packages, three basic approaches are possible:

(a) place a single contract for both hardware and software
(b) place a separate contract for the software
(c) place a single contract with a named (software) subcontractor

Approach (a) has the advantage that less specification, management and interface design is required of the VPO. It also lessens the risk of failures at the hardware/software boundary. It has the disadvantage that a single contractor, even with subcontractors, may not be able to

cover all areas adequately, especially when the proprietary nature of some existing VLT software is taken into account.

Approach (b) leaves open the possibility of placing a software contract with a company already familiar with VLT software (perhaps even ESO itself), whilst being able to choose the best hardware vendor. Approach (c) has similar advantages and, in addition, removes some detailed work from the VPO, though it some situations could increase the contract price.

There are potential problems incurred by contracting software out to companies or organisations unfamiliar with the infrastucture they will be constrained to use, particularly for case (a). Risks will be assessed and mitigated using the general project procedures (AD07).
For approach (a) the software specification will be part of the overall specification and will include

- the software interface definition
- the constraints of using VLT infrastructure software and complying with ESO standards

For approach (b), there will be a standalone software specification, which will include:

- software interface definition
- software specification
- hardware interface definition
- the constraints of using VLT infrastructure software and complying with ESO standards

Table 1 Ways of handling different types of modules.

	Potential examples		Provided by Project Office	Provided from Workpackage
	Software only	**Hardware & software**		
Reuse without modifications	- BOB		NA - using these is a constraint	
Minor modifications		- Axis Control	• Check functionality against VISTA requirements • Request changes by ESO	
	- TCS			
Rewrite (partial)		- M2 Control	• Requirements • Functional Specification • Interface Definitions • Test Plan • Existing application software • ESO infrastructure software	• Software Design • Implementation • Incremental Deliveries
Rewrite (complete)		- M1 Control	• Requirements • Functional Specification • Interface Definitions • Test Plan • ESO infrastructure software	• Software Design • Implementation • Incremental Deliveries
New package	- OS - On-line DR - Pipeline		• Software Requirements • Functional Specification • Interface Definitions	• Software Design • Implementation • Incremental Deliveries
		- ICS - Guider - WFS	• Package Requirements • Software Functional Specification • Interface Definitions	• Software Design • Implementation • Incremental Deliveries

3.3.2 Instrument Software Analysis

The analysis phase will define module functional specifications and interface definitions. It will be the responsibility of the organisation to whom the instrument software is assigned, which may or may not be the same organisation given responsibility for the instrument hardware. (There are clear advantages in making one organisation responsible for both, but pragmatic considerations may dictate otherwise.)

Inputs to this phase are

- Instrument Software Requirements
- Instrument hardware design
- VISTA Software Architecture

Outputs are (for each instrument package):

- OS Functional Specification
- ICS Functional Specification
- DCS Functional Specification or Assessment of Chosen DCS
- On-line Data Reduction Functional Specification
- Pipeline Functional Specification
- Interface definitions, where these are not covered by the Functional Specifications.

In the event that an existing DCS is chosen, e.g. those used by existing VLT instruments, an assessment of this DCS against VISTA's SRD, VTS and OCDD will replace the generation of a formal Functional Specification.

The outputs from this phase will be reviewed at the Instrument Software Package PDR. This may be part of the Instrument PDR, though there are advantages in holding a separate Software PDR after the Instrument PDR. (One disadvantage of this approach is the risk of software having to work around inappropriate hardware decisions. To mitigate this risk, software needs to be considered at all stages of the decision making process.)

3.3.3 Module Design

The design phase decides exactly how the Functional Specification will be met in software. The design will be described using either structured analysis and design methods or object oriented methods (see Section 5). The design should be detailed to the point that an assessment may be met as to its soundness, but should not be so detailed that it describes what is better described in code during the implementation phase. Some aspects of the design may best be described as code stubs, e.g. the GUI screens.

Inputs to this phase are:

- Module Functional Specification
- VISTA Software Architecture

- VLT standards

Outputs are:

- Software Design Document
- GUI mock-ups
- Data Dictionary
- Module Test Procedure
- Prototypes (if the need is clear and agreed)

The deliverable of this phase shall be reviewed at the Package FDR. It may be appropriate to review draft deliverables at the Package FDR and to hold separate FDRs for each module.

3.3.4 Module Coding

The coding phase implements the module design in executable code and generates user documentation, where users include telescope/instrument operators and other software developers. In the current document, this phase incorporates two phases described in AD05, viz. the coding phase and the unit test and module integration phase, since in practice there is considerable overlap between these activities.

The inputs to this phase are:

- Software Design
- Data Dictionary

The outputs are

- incremental code release(s) for integration and hardware tests
- final debugged code release
- user documentation

3.3.5 Module Acceptance Testing

This phase completes the work on each module by testing it against its Functional Specification and Interface Definitions.

The inputs are

- final code release
- user documentation
- Module Test Procedure

The outputs are

- accepted code and documentation release
- Module Acceptance Test Report

3.3.6 Package Integration

This phase integrates the discrete modules into a complete instrument software package. It is in fact the conclusion of a continuous process, since it is expected that incremental releases of modules will have been made to the Package Manager for integration testing.

Inputs to this phase are:

- final code release
- user documentation
- module acceptance test reports

Outputs are:

- final package code release
- user documentation

3.3.7 Package Acceptance Testing

This phase tests the acceptability of the package of modules both to the VISTA Project, in terms of meeting requirements, and ESO, in terms of operations and maintenance.

Inputs to this phase are

- final package code release
- user documentation
- Module Acceptance Test Reports
- Package Software Requirements

Outputs are

- Package Acceptance Test Report

3.3.8 Contract Monitoring

Contract tracking procedures are described in *VISTA Project Procurement Plan* (AD02).

4. Responsibilities and Rôles

4.1 VISTA Project Office

4.1.1 General Responsibilities

The Project Office is responsible for managing the project, but not for providing effort to work on specific workpackages. These will be contracted out, in some cases potentially to the same organisation that hosts the Project Office.

The following rôles are identified within the VISTA Project Office; each role will be performed by a named individual. Some rôles may be performed by the same person, e.g. on some packages the Project Software Engineer may perform the rôle of Software Package Responsible. In such cases other project personnel, e.g. the Project Manager or Systems Engineer may be involved in approvals processes. Execution of some duties may be delegated to others within the organisation, but responsibilities remain with the rôles defined below.

4.1.2 Project Manager

The responsibilities of the VISTA Project Manager are described in AD01. The VISTA Project Manager manages all top level aspects of the project. More detailed responsibilities in the computing area are delegated to the Project Software Engineer.

4.1.3 Project Systems Engineer

The Systems Engineer is responsible for the systems design, error budgets and interfaces between systems.

4.1.4 Project Software Engineer

The Project Software Engineer has responsibilities delegated by the Project Manager as follows:

- Production of Software Management Plan (this document)
- Contact point with ESO on software matters
- Top level software and hardware architecture
- Setting general requirements and constraints on workpackages
- Workpackage breakdown
- Software system engineering
- Software interface control
- Standards
- QA procedures
- Project planning and cost control
- Control of external software contractors, with delegation to the SPR
- Overall monitoring
- Overall configuration management

4.1.5 Work Package Manager

The Work Package Manager has responsibilities delegated by the Project Manager to manage a work package or contract (AD07).

4.1.6 Software Package Responsible

The Software Package Responsible (SPR) perform the following duties delegated by the Project Software Engineer:

- Analysis and design of the package producing the Software Functional Specification

- Liaison point with contractor on software issues
- Liaison with other SPRs if appropriate, e.g. if commercial confidentiality does not preclude it
- Monitor progress of contractor on software issues
- Approve package's analysis and design including organising document reviews
- Integrate the package with the overall system
- Performance of acceptance tests and tests of incremental deliveries
- Organise reviews of software
- Software configuration management
- Enforcement of standards and QA

The Software Package Manager works under the direction of the Work Package Manager. For purely software packages, the Software Package Manager would likely be the same person as the Work Package Manager.

4.2 Workpackage Developers

Workpackages will be assigned to, e.g., commercial contractors or academic support institutions. So far as the management of the project is concerned, it makes little difference what type of organisation is responsible for a workpackage.

Each workpackage will have a named manager, responsible for all aspects of the workpackage including

- liaison point with the VISTA Project Office
- project planning and cost control, including provision of a WBS
- submission of regular (e.g. monthly) progress reports
- submission of documents to the review procedure
- presentation of the project at reviews
- detailed design of the package producing the Software Design Description (note that AD05 assigns this responsibility to the SPR within the Project Office)
- implementation
- provision of incremental releases
- preparation of test plans
- executing acceptance tests

The following rôles are identified within the organisation responsible for the implementation of an instrumentation package:

Package Manager

- responsible for the entire package of modules and their integration

Module Manager

- responsible for one particular module

For a telescope package, e.g. M1 Control, there is only a Package Manager.

4.3 ESO

Responsibilities of ESO as applied directly to the development of VISTA software are described below. Responsibilities related to the impact of VISTA within ESO are not within the scope of this document.

- provision and support of ESO infrastructure software, including bug fixes
- provision of appropriate software modules reusable from the VLT, either in whole or in part
- support of modules reused from the VLT, unless modified
- provision of new versions of infrastructure software to VISTA and its contractors (contractors will not be obliged to use new versions)
- support to the contractor (as negotiated)

5. Tools, Techniques and Standards

5.1 Analysis and Design

The VLT's initial analysis and design method was Ward/Mellor's Structured Analysis and Design (SA/D) with Realtime Extensions (RD05). This is still widely used, but object oriented analysis and design has become widespread. VISTA will in general allow either approach to be used for any package or module, subject to approval at PDR.

For SA/D, no particular tool is made mandatory, although the VPO uses Select Yourdon from Select Software Tools. In practice, such tools are likely to be used largely as a diagramming aid with some degree of consistency checking, rather than as a means of generating a full design consistent with the code. SA/D diagrams should where relevant be inserted into documents, e.g. Software Design Documents. SA/D models should also be included with the final module and package releases, whether or not ESO and the VPO have access to the particular tool used.

Developers are encouraged to use object oriented analysis and design techniques. Where they do so, the Unified Modelling Language (UML) shall be used. The Preferred tool is Rational Rose, from Rational Inc., but alternatives may be proposed prior to contracts being placed. Where UML is used for analysis and design and C++ or Java is used for coding, the model and the code shall be supplied consistent with each other.

Table 2 Analysis and design tools and methods.

Phase	SA/D	OO
Requirements	Structured English or Use Cases	Use Cases
Analysis & Design	Context diagram Data flow diagram State transition diagram Data dictionary	Sequence diagram Collaboration diagram Class diagram
Coding	C (or C++ as a better C)	C++, Java
Method	Ward Mellor	UML
A&D Tools	Select Yourdon (suggested)	Rational Rose (suggested)
Configuration Management	CMM (mandatory)	CMM (mandatory)

5.2 Programming Standards

ESO's VLT programming environment shall be used throughout including the LCC on LCUs and CCS at the higher levels. This environment provides callable functions etc. to perform many of the tasks that would otherwise require development and a structure within which application code can be developed in an efficient and maintainable manner.

The only languages that may be used without explicit permission are C, C++ and Tcl. Coding shall comply with the VLT Programming Standards (AD06).

All software shall comply with the *Guidelines for the Development of VLT Applications* (AD09).

Instrument control software shall comply with the *VLT Instrument Software Specification (*AD11).

Data flow software shall comply with the *Data Flow for VLT Instruments Requirement Specification* (AD12).

These standards shall be fully followed, unless explicit permission is granted by the VPO, who will not grant such without ESO's agreement.

5.3 Data Definitions

All data written to the permanent archive shall be documented and approved as specified in the Data Interface Control Document (AD10).

5.4 Configuration Management

Software configuration shall be fully implemented as defined in the VLT Software Configuration Control Plan (AD17), using ESO's own Configuration Management Module (CMM) Tool (AD15). Within a module developers may choose to use another tool, but each release of module or package code, whether incremental or final, shall use CMM.

6. Review Procedures

6.1 Formal Reviews

Formal reviews, e.g. PDR, FDR and Acceptance, shall be organised as specified elsewhere for the entire VISTA Project (AD07).The Package Manager shall be responsible for presenting work from those phases which are the responsibility of the contractor. The SPR, under the direction of the Project Manager, shall be responsible for ensuring that this work is properly reviewed.

6.2 Phasing Hardware and Software Reviews

Phasing software reviews to occur later than the corresponding hardware review has some advantages:

- software requirements depend on overall requirements
- software details depend on hardware details
- software may be scrutinised more fully

but has the disadvantages:

- increased risk of inappropriate hardware decisions necessitating software workarounds at greater cost
- increased risk of software work starting too late to meet schedules

If a software review is phased later than a corresponding hardware review, there shall be adequate software representation at the hardware review to mitigate these risks. The decision whether or not to phase the reviews will be taken by the VISTA Project Manager.

6.3 Documentation Plan

Prior to formal reviews, it is efficient to review documents on an individual basis prior to their release. Procedures for reviewing documents are described in (AD08).

VISTA's overall documentation management is described in (AD03). Software documents to be produced during the different phases of the VISTA project are described in the preceding sections and summarised in Table 3.

Table 3 Software documents to be produced in the different phases of the project.

Phase (Review)	Review	Output Documents
System Design	Phase A	• Software Architecture Document
Package Req's	Phase A	• IR Camera Software Req's • Opt. Camera Software Req's • Evaluation of VLT TCS against VISTA Req's
Package Analysis	PDR	• Functional Specification • Interface definitions
Module Design	FDR	• Software Design Document • Data Dictionary • Test Procedure
Module Coding		• Module code release • Module User Manual
Module Acceptance Testing	Module ATR	• Acceptance Test Report
Package Integration		• Package code release • Package User Manual
Package Acceptance Testing	ATR	• Package Acceptance Test Report

6.4 Risk Management

Risk management shall be performed following VISTA's overall project procedures (AD07). Risks especially relevant to software and ways of mitigating them are listed in Table 4.

Table 4 Risk factors especially relevant to software and ways to mitigate them.

Risk	Mitigation
Changing requirements	• Impact assessment before approving change
Misunderstood requirements	• Thorough requirements reviews • Award workpackages to groups already familiar with the domain (e.g. telescopes and astronomical instruments)
Use of ESO infrastructure software	• Assess contractor's experience and personnel before awarding contract • Arrange for infrastructure software support, e.g. with ESO
Hardware/software incompatibility	• Single contract for both hardware and software • Thorough control of ICDs

Risk	Mitigation
Incompatibility with other software systems	• Thorough control of ICDs • Integration of incremental deliveries

6.5 Requirements Change Management

Change management is particularly important in software for the following reasons:

- software is easily changed late in the process (or is perceived to be)
- software can always be made "better"
- these is more scope for ambiguity in requirements
- some performance metrics are difficult to quantify, e.g. in the user interface area

A formal process is necessary to manage changes to requirements and interfaces. The general procedure shall be:

- create change request
- analyse impact on cost and schedule
- review the impact analysis, approve or otherwise and prioritise
- allocate for implementation

These are described more fully in the context of the overall project in (AD07).

oOo