APPROVAL SHEET

TITLE : SALT Software Standard

DOCUMENT NUMBER : 1000BS0010 ISSUE: B

SYNOPSIS : This document describes the software development process and specific software requirements for all SALT software.

KEYWORDS : Software, Operating System, Axis Control, real-time, computer.

PREPARED BY : Gerhard Swart

APPROVED : Kobus Meiring
SALT PROJECT MANAGER

February 2001

DATE : 
This issue is only valid when the above signatures are present.

Printed: 16/09/03 12:00
## ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATP</td>
<td>Acceptance Test Procedure</td>
</tr>
<tr>
<td>ATR</td>
<td>Acceptance Test Report</td>
</tr>
<tr>
<td>BITE</td>
<td>Built-in Test Equipment</td>
</tr>
<tr>
<td>BMS</td>
<td>Building Management System (formally called the Environmental Control Computer)</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-coupled Device (Camera)</td>
</tr>
<tr>
<td>CDR</td>
<td>Critical Design Review</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial off the shelf</td>
</tr>
<tr>
<td>HET</td>
<td>Hobby-Eberly Telescope</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output (Device)</td>
</tr>
<tr>
<td>ICD</td>
<td>Interface Control Dossier</td>
</tr>
<tr>
<td>MMI</td>
<td>Man-Machine Interface</td>
</tr>
<tr>
<td>MTBF</td>
<td>Mean Time Between Failures</td>
</tr>
<tr>
<td>MTTR</td>
<td>Mean Time to Repair</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PDR</td>
<td>Preliminary Design Review</td>
</tr>
<tr>
<td>PFIS</td>
<td>Prime Focus Imaging Spectrograph</td>
</tr>
<tr>
<td>PI</td>
<td>Principal Investigator (Astronomer)</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable-Logic Controller</td>
</tr>
<tr>
<td>RT</td>
<td>Real-time</td>
</tr>
<tr>
<td>SA</td>
<td>SALT Astronomer</td>
</tr>
<tr>
<td>SALT</td>
<td>Southern African Large Telescope</td>
</tr>
<tr>
<td>SDD</td>
<td>Software Design Document</td>
</tr>
<tr>
<td>SDP</td>
<td>Software Development Plan</td>
</tr>
<tr>
<td>SO</td>
<td>SALT Operator</td>
</tr>
<tr>
<td>SPCT</td>
<td>Single-point Communication Test</td>
</tr>
<tr>
<td>SRS</td>
<td>Software Requirement Specification</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>TBC</td>
<td>To Be Confirmed</td>
</tr>
<tr>
<td>TBD</td>
<td>To Be Determined</td>
</tr>
<tr>
<td>TCS</td>
<td>Telescope Control System</td>
</tr>
<tr>
<td>VI</td>
<td>Virtual Instrument (Labview function)</td>
</tr>
</tbody>
</table>
DEFINITIONS

Supplier  The organisation developing a SW Item.

Developer  The technical person(s) working for the supplier organisation responsible for the development of the SW item. This term is used interchangeably with “Supplier”.

Client  SALT (Pty) Ltd, as represented by the appropriate subsystem manager, contracting the development of the SW Item to the supplier.
# TABLE OF CONTENTS

1 Scope .................................................................................................................. 7  
2 Referenced Documents ....................................................................................... 8  
3 Tailoring of Requirements .................................................................................. 8  
4 Software Development Process Requirements .................................................. 12  
4.1 Software Development Planning ................................................................. 12  
4.2 Requirements Analysis ............................................................................... 13  
4.3 Software Specification Review ..................................................................... 13  
4.4 Software Design ............................................................................................ 14  
4.5 Software Critical Design Review ................................................................. 14  
4.6 Software Coding and Debug ......................................................................... 14  
4.7 Software Code Reviews ............................................................................... 15  
4.8 Module Testing .............................................................................................. 15  
4.9 Software Testing ............................................................................................ 15  
4.10 Integrated HW and SW Testing ................................................................. 15  
4.11 Subsystem Commissioning and Integration ............................................... 15  
4.12 Software handover ....................................................................................... 16  
5 Software Safety .................................................................................................. 17  
5.1 Safety certificate ............................................................................................ 17  
5.2 Communication Integrity ............................................................................ 17  
5.3 Initialisation .................................................................................................... 17  
5.4 Start-up and Shut Down Procedure .............................................................. 17  
6 Generic Software Requirements .......................................................................... 18  
6.1 Naming and Tagging Conventions ............................................................... 18  
6.2 Remote Initialisation ..................................................................................... 18  
6.3 Critical Dataset Table .................................................................................. 18  
6.4 Software cyclic execution ............................................................................ 18  
6.5 Data time stamping ....................................................................................... 18  
6.6 Modular Design ............................................................................................. 18  
6.7 Measuring Units ............................................................................................ 19  
6.8 Data resolution ............................................................................................... 19  
6.9 I/O Validation ................................................................................................ 19  
6.10 Synchronisation ............................................................................................ 19  
6.11 Redundant Code ........................................................................................... 19  
6.12 Software Comments .................................................................................... 19  
6.13 Self-changing code ....................................................................................... 20  
6.14 Manual Operation ......................................................................................... 20  
7 Specific PLC Software Requirements .................................................................. 21  
7.1 Scan Cycle ...................................................................................................... 21  
7.2 Assignment of Memory Areas ....................................................................... 21  
7.3 Retained Data ................................................................................................ 21  
7.4 Cycle of Inputs and Outputs Scanning .......................................................... 21  
7.5 Memory and Cycle Time Optimisation ......................................................... 21  
7.6 Programming Segments ............................................................................... 21  
7.7 I/O Manipulation ............................................................................................ 21  
7.8 Device Control ............................................................................................... 21  
7.9 Interlocks ........................................................................................................ 21
8 Specific Axis-controller Software Requirements ...................................... 22
  8.1 Axis Controller functions .................................................................... 22
  8.2 Synchronisation .................................................................................... 22
9 Specific PC Software Requirements ......................................................... 23
  9.1 Operating Systems ................................................................................ 23
  9.2 Development Software ........................................................................ 23
  9.3 Application Software ............................................................................ 23
  9.4 Man-Machine Interfaces ....................................................................... 23
10 Deliverables ............................................................................................ 24
11 Configuration Control .............................................................................. 25
1 Scope

This document specifies the requirements for all software that forms part of the Southern African Large Telescope (SALT). It indicates the required development process and specific coding and documentation practices that are applicable.

All SALT Software shall be subject to this standard except where specific exemption has been granted by the SALT Systems Engineer. Section 3 indicates which requirements are applicable to the various subsystems.

The purpose of most of the SALT software is to control part of the telescope to perform a defined set of functions. The term “system” refers to the integrated mechanical, electrical, computer hardware and software required to perform such a set of functions. This may be the whole or part of a SALT Subsystem (e.g. Tracker, Primary Mirror System). The term “software item” is used for a whole program, operating system, bios, interpreter or other group of separately compiled computer code.

The purpose of this standard is to ensure that the software used on SALT is fit for purpose. This means that it must perform the desired functions reliably under expected and unexpected conditions, must be modular and easily modified/upgraded and must be efficient in its use of computing resources. Adequate software maturity must also be achieved prior to the integration of sub-systems with each other to minimise integration time and costs.

Software development requirements to achieve the above objectives are well researched and documented in commercial and military software standards (the most notable are listed in section 2). This standard serves to summarise the most crucial requirements that are applicable to SALT and highlights implementation-specific requirements where this is considered essential.

Where suppliers have existing software development practices that are well documented and/or comply with recognised industry standards (e.g. ISO9001, MIL-STD-498 and IEEE/EIA 12207.0), these may substitute major portions of this document, but the compliance to the minimum requirements defined herein must be shown.

If any requirements of this document are considered excessive, inappropriate or difficult to meet, developers are invited to seek guidance from the authors or to request a concession of such requirements during the software planning phase.
2 Referenced Documents

The following documents are referenced in this standard.

<table>
<thead>
<tr>
<th>Document ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALT-1000AS0007</td>
<td>SALT System Specification</td>
</tr>
<tr>
<td>SALT-1000AA0030</td>
<td>SALT Safety Analysis</td>
</tr>
<tr>
<td>SALT-1000BS0021</td>
<td>SALT Requirements for Built-in Testing</td>
</tr>
<tr>
<td>SABS ISO 9001</td>
<td>Model for quality assurance in design/development, production, installation and servicing</td>
</tr>
<tr>
<td>SABS ISO 9000-3</td>
<td>Guidelines for the application of SABS ISO 9001 to the development, supply and maintenance of software</td>
</tr>
<tr>
<td>ISO/IEC 12207</td>
<td>Standard for Information Technology – Software life cycle processes</td>
</tr>
<tr>
<td></td>
<td>Internet Applications in Labview, Jeffrey Travis, 2000 Prentice Hall</td>
</tr>
<tr>
<td></td>
<td>National Instruments Labview Development Guidelines, July 2000, N</td>
</tr>
<tr>
<td></td>
<td>part number 3211393C-01</td>
</tr>
</tbody>
</table>

3 Tailoring of Requirements

The first step in using this standard is to identify the sections of this standard that are applicable to each specific item of software. Table 1 below provides a guideline in this regard, but final applicability must be agreed between the SALT Systems Engineer and the Software Developer (organisation) during the Software Development Planning (see section 4.1 for details). Paragraphs not referred to in the table are applicable to all software.

The paragraphs that are applicable for any software inadvertently not appearing in the table, will be negotiated on a case-by-case basis.
<table>
<thead>
<tr>
<th>Item</th>
<th>SW Items</th>
<th>Section 4: Development Process</th>
<th>Section 5: Software Safety</th>
<th>Section 7: PLC Rqrmnts</th>
<th>Section 8: Axis-control Rqrmnts</th>
<th>Section 9: PC SW Rqrmnts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dome Computer*</td>
<td>Operating System</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structure Computer*</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCS server, SA and SO workstations, Error logger, Firewall and Data Processor</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network switches and hubs</td>
<td>Bios (embedded SW)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration information</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Primary Mirror Computers (edge sensors, actuator control, align sensor)</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Item</td>
<td>SW Items</td>
<td>Section 4: Development Process</td>
<td>Section 5: Software Safety</td>
<td>Section 7: PLC Rqmnts</td>
<td>Section 8: Axis-control Rqmnts</td>
<td>Section 9: PC SW Rqmnts</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>----------------------</td>
<td>-----------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Primary mirror actuator control electronics, alignment sensor axis controllers</td>
<td>Compiler or interpreter</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Programming SW</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracker and Payload computers</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Tracker/Payload axis controllers and camera controllers</td>
<td>Compiler or interpreter</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Programming SW</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Instrument computers</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Instrument axis controllers, CCD controllers</td>
<td>Compiler or interpreter</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Programming SW</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Building Management System (BMS)</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bought-out application software</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
## Applicable Requirements

<table>
<thead>
<tr>
<th>Item</th>
<th>SW Items</th>
<th>Section 4: Development Process</th>
<th>Section 5: Software Safety</th>
<th>Section 7: PLC Rqrmnts</th>
<th>Section 8: Axis-control Rqrmnts</th>
<th>Section 9: PC SW Rqrmnts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software development environment</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Air conditioning, cooling, air and ventilation controllers, Fire detection, weather station</td>
<td>Compiler or interpreter</td>
<td>X</td>
<td>X or</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Programming SW</td>
<td>X</td>
<td>X or</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom Application code</td>
<td>X</td>
<td>X or</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SW Configuration data</td>
<td>X</td>
<td>X or</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard Application software</td>
<td>X</td>
<td>X or</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mirror Coating plant</td>
<td>Operating System</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Development environment</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bought-out software</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Custom application code</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration data</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

### Note:

"X" denotes applicability of the requirement to the SW item.

* The Dome and Structure controllers/computers should preferably be combined into one item.

Table 1: Tailoring guidelines
4 Software Development Process Requirements

Figure 2 shows the SALT software development process, based on the well-known “waterfall” model. As software quality is determined primarily by the development process followed, it is mandatory to follow this process for all software developed or customised specifically for SALT. This software process is applicable per “Item” listed in table 1, although some specific SW items (e.g. bought-out items such as Operating Systems) will not follow the whole process.

4.1 Software Development Planning

The first step is to plan the whole software development process (steps 4.2 to 4.12) and to document this formally in a Software Development Plan (SDP). Any deviations from this standard shall be agreed with the client at this stage and documented in the SDP.

During the planning process the developers shall determine the strategy with which the software will be developed, identifying for each step in Figure 2, what activities will be performed, what resources will be required (personnel, computers, system hardware) and how the requirements of this document will be met. A schedule showing the planned sequence of these events shall be included.

In particular, the software review and testing process shall be clearly defined, with client attendance planned for such activities.

The client shall be given the opportunity of reviewing the Software Development Plan, prior to the start of software design.
4.2 Requirements Analysis

Determining the true requirements of the software is an important activity that must not be underestimated.

During this process the particular functions of the system that have to be controlled by software are identified, analysed and specified to sufficient detail to allow the software to be designed. The detailed requirements for computer hardware/firmware and other components of the system (e.g. mechanical and electrical parts) will also be identified in parallel with this process. The following specific requirements shall be identified and documented in a Software Requirement Specification (SRS):

a. The **computer architecture** showing peripherals and all items containing software, identifying each software item by a unique name (unless the software is identical) and identifying which software items will be developed or modified (e.g. application software). The data links between each computer shall also be shown. The computers/ peripherals typically include axis controllers, computers, remote I/O devices, actuators and sensors.

b. The **software architecture** showing the major software modules (building blocks) per developed software item and identifying the major data flowing between modules.

c. A **mode/state transition diagram** is required for all developed software, showing software mode upon initialisation and the conditions for transition to other modes. The distinct requirements of each mode shall be identified.

d. A **resource allocation** in which the timing, memory allocation, processing power and overall loop iteration frequency have been defined and the required spare capacity reserved.

e. A diagram or table showing the required **software timing** including aspects such as data latency, real-time requirements and synchronisation with the real-world.

f. An **input and output allocation**, identifying the major inputs and outputs between the software and hardware.

g. The detailed **software requirements definition** of each software item, according to the complexity and type of the software item.
   
o. For operating systems, interpreters and other bought-out software, the specific software brand name, version and configuration requirements (e.g. priority of interrupts, error handler set-ups) need to be defined.

 o For developed software items, the required execution rate, input, output and processing (e.g. algorithms) must be defined per software module. Any other critical requirements pertaining to a particular module must also be identified (e.g. timing, data format, security)
   
o. User-configurable features need to be defined in concept

 o Calibration/adjustment/set-up procedures methods need to be defined in concept

 o Where software has direct interaction with the hardware (e.g. I/O), interfacing details need to be specified (e.g. drivers)

 o The allocation and use of interrupts need to be defined and their use motivated. The interaction between the operating system, synchronous, cyclic events and interrupt events need to be defined.

h. **Verification methods** for the measurement of software compliance to the requirements of the Software Requirements Specification shall be defined.

i. **Software coding constraints** that will enhance software maintainability, reliability and testability shall be defined (e.g. particular data structures, non-use of certain language features).

j. The operational **Man-Machine-Interface** (if applicable) shall be defined in a detailed fashion.

4.3 Software Specification Review (PDR)
Prior to starting the software design, the Software Requirement Specification shall be reviewed by the supplier development team and the client. The purpose of the review is to verify that the software requirements have been correctly derived from the Subsystem Specification, check that adequate analysis has been performed in defining the software requirements and to co-ordinate the software requirements with those of other components of the Subsystem. The Software Requirement Specification is subject to client approval prior to proceeding with Software Design.

4.4 Software Design

Prior to coding the software, it is essential to structure and design the software to meet not only the functional requirements of the SRS, but also the maintainability, reliability and testability requirements. The output of the Software Design process will be in various forms, but the major design aspects shall documented in one or more Software Design Documents (SDD). At least the following shall be addressed:

a. A high-level design description, describing the overall integration and interaction of the modules how this relates to the software states and modes.

b. An updated copy of the software architecture diagram defined in para. 4.2

c. A detailed functional-flow and data-flow diagram, showing all the software modules and the precise data flowing between them. The implementation of specific timing, synchronisation and interrupts requirements shall be illustrated.

d. The software design of each module must be provided. This shall indicate the specific data inputs, outputs, processing and timing requirements for that module and shall give specific formula’s and algorithms that are to be executed. Details of global and local data variables, interrupt operation, timing implementation and shall be defined. The design shall be documented in pseudo-code, flow diagrams or English narrative.

4.5 Software Critical Design Review (CDR)

Prior to full-scale software coding, the software design shall be reviewed by the supplier development team and the client. The purpose of the review is to verify that the requirements of the SRS and other implicit requirements have been adequately and efficiently addressed in the design. It is an opportunity for the development team to co-ordinate the hardware, software and equipment designs and to ensure that non-functional requirements such as maintainability, testability and reliability are adequate.

The CDR shall address the overall software design (architecture, data flow, timing) and detailed design of each module.

4.6 Software Coding and Debug

During this process the software code for each module is generated according to the design defined in the SDD. Specific coding standards, metrics and conventions are applied (as defined elsewhere in this document) and software comments inserted.

In parallel with the software coding process, a Software Acceptance Test Procedure (ATP) is defined and documented by the developer. Tests shall be defined to verify that the software complies with each requirement of the SRS. This document shall subject to approval by the client.
4.7 Software Code Reviews

The source code of each completed module is reviewed by the development team to check the appropriateness of software style, efficiency and to co-ordinate interfacing modules. The appropriate method of testing each module shall be agreed. The client may at his discretion attend such reviews. A record shall be kept of each review and the comments recorded. The implementation of such comments shall be verified during module testing.

4.8 Module Testing

Software modules shall be individually tested prior to integration with the other modules. Testing a module shall use either a simple stub simulating interfaces to other modules or another module (or group of modules) that has already been tested. The results of each module test shall be recorded.

4.9 Software Testing

Tested modules are incrementally integrated together and progressively checked. When all the software has been integrated, the tests defined in the Software ATP shall be executed where possible without the final hardware. This is considered a “first run” of the Software ATP and the results shall be recorded in an Acceptance Test Report (ATR) that may be the same document as the ATP, with results written in at each test.

This testing need not be with the final Hardware configuration and where interfacing hardware is not yet available, simulating stubs will be allowed.

The precise hardware and software configuration tested shall be defined. From this point forward, all software changes shall be logged.

The test shall be witnessed and ratified by a person not directly involved in the software development (e.g. developer QA or the client). At this point the software shall be fully under Configuration Control (see section 11) and all software changes managed.

4.10 Integrated HW and SW Testing

When the software has been integrated with the correct hardware the Software ATP is performed again, but this time it will include functions that interface to the hardware. The tests results are recorded in an Acceptance Test Report (ATR) that may be the same document as the ATP, with results written in at each test.

A TCS Server simulator will be developed by the client and issued to the developers of all non-TCS computer items. The purpose of the simulator will be to verify the TCS interface to each computer prior to delivery of that computer and its software by the developer. A communication test using the TCS simulator shall be part of each computer item’s ATP.

At this point, we have a stand-alone item of equipment containing hardware and software and controlling specific parts of a subsystem. This may be a computer, axis controller or other “Item” as indicated in Table 1.

4.11 Subsystem Commissioning and Integration

The next step of the process is to progressively integrate the equipment item containing the software with other items of equipment of the same subsystem. This Commissioning is complete when the Subsystem ATP, which verifies the performance of the Subsystem against its specification, has been passed.
The final step of the process, during which the final aspects of the software items performance is verified, is the System Integration, when all the subsystems are integrated to form an operating Telescope. Only when the SALT System ATP has been successfully completed, can each SW item be said to be complete.

Supplier support may however still be required after that date to resolve hidden problems that had not been previously identified.

4.12 Software handover

During step 4.11, the responsibility for maintenance of the software is transferred from the original developer to the client. At this point, a formal “Software Handover” shall occur, when a “snap-shot” of the software configuration is taken and a package compiled that contains the “delivered software”. See section 10. This delivered software package shall contain a full definition of the latest software configuration, including the following:

a. A Version Definition – a table indicating the current revision numbers of each of the software modules of each software item
b. The Software Configuration Definition – an electronic copy of all configuration data for operating systems, firmware, set-up data, calibration constants, user-defined parameters etc.
c. The Software Source Code of the present software version
d. Original legal copies of the operating systems, compilers, tools, utilities that are required to maintain the software
e. Final copies of Operating, Maintenance and Calibration procedures where applicable
f. A final version of the Safety Certificate

The supplier shall keep a configured backup copy of all software.
5 **Software Safety**

5.1 **Safety certificate**

A Safety Certificate shall be issued for each item of equipment containing software, whether such software has been procured as is, modified or newly developed. The certificate shall identify all the software items that form part of that equipment. A pro-forma of the software certificate can be found in the SALT Safety Analysis listed in section 2.

5.2 **Communication Integrity**

Communication integrity between subsystems and all equipment items shall be monitored by all items receiving data. Failure to receive correct data or failure to receive any data from a particular device shall be reported to a higher-level computer.

Detection of communication failure shall be facilitated by a “Single Point Communication Test” (SPCT) performed in the following way:

- a. Computer A publishes a varying SPCT number with the other data being published. This number is updated and changed at least twice a second and could typically be an internal software counter.
- b. Any computer using the data from Computer A also reads the SPCT and checks that it has changed in the last second. If not, communication with computer A has been interrupted and failure routines are activated.
- c. If a change in the SPCT is detected again, full communication must be re-established without operator interaction, except where this is required for operation reasons.

Each subsystem shall fail in a safe fashion if it does not receive the required data. Gradual degradation of system performance should be allowed where possible.

5.3 **Initialisation**

All Outputs shall be in a safe state when un-initialised or switched off. Similarly, initialised inputs (e.g. A/D converters or incremental encoders) shall not cause incorrect responses from the software.

The following initialisation sequence shall be followed by all software:

- a. Switch all outputs to a safe state (e.g. motors, OFF)
- b. Indicate “Initialisation State” to the outside world
- c. Check the integrity of the processing hardware and memory using simple arithmetic checks
- d. Check communication with and correctness of peripheral devices (a loop-back test on I/O)
- e. Verify the correctness of configuration data and then initialise variables accordingly
- f. Check communication with interfacing computers (SPCT)
- g. If all operations are successful, report “System Okay” to the higher level computer and enter into a “ready” state, where after the state will be determined by switches, commands, data etc. If operations a. to e. are not successful, report “System Start Failure” and indicate the type of failure encountered. If communications with another computer cannot be established, this should be reported to a higher-level system.

5.4 **Start-up and Shut Down Procedure**

During Start-up and Shut Down, preventative measures shall be taken to handle process conditions as well as Inputs and Outputs in a safe manner.
6 Generic Software Requirements

6.1 Naming and Tagging Conventions

Each SW Item shall be uniquely identified with the appropriate SALT number (see section 11) and a sensible name.

All variables, memory and block naming shall be clear, logical and understandable. A uniform convention shall be used throughout an item of equipment, using whole English words separated by underscores where possible (e.g. TCS_mode_command, Tracker_X_command, Tracker_X_position). Where compilers/interpreters do not support long variable names (e.g. some axis controllers), a consistent abbreviation may be used, with a clear definition in the appropriate documentation.

Naming conventions will be agreed during the Software PDR.

6.2 Remote Initialisation

It shall be possible to trigger the initialisation sequence described in 5.3 remotely via the normal communication to an item. (e.g. The TCS must be able to send a "reset" command across the Ethernet to any computer to trigger initialisation).

6.3 Critical Dataset Table

A set of Critical Data, over and above data required for functional operation, shall be agreed with the client for each computer item. This data set shall be updated at a rate of at least 5Hz between interacting computers and would typically include:

- Item Mode
- Item Health Status
- Loss of communication warning (identification of comms lost with another computer)
- Current Process Step or Mode
- Fault list

This Data Set will be finalised during the Critical Design Review.

6.4 Software cyclic execution

After the completion of initialisation, the code of a SW item shall execute in a cyclic fashion, at a constant rate, commensurate with the control bandwidth/frequency/latency required.

6.5 Data time stamping

Time-critical data will be agreed with each supplier and identified as such in the ICD. All such data shall be time-stamped in an agreed fashion to facilitate synchronisation of subsystems.

6.6 Modular Design

Software shall be designed in a scalable and modular fashion. All software modules (i.e. Labview VI’s, procedure and functions) shall be designed to minimise their data interfaces and to group functions that belong together, keeping in mind future growth and hardware upgrades. Compliance to these requirements shall be demonstrated at the PDR, CDR and code reviews. In particular, the following types of functions shall be in independent modules:

- Input/Output hardware communication drivers
- Input/Output scaling from hardware units (e.g. 1024bits) to/from engineering units.
- Initialisation sequences
• User configuration sequences
• Equipment mode/state control
• Mathematical/control algorithms
• Data storage and retrieval
• Data communication
• Fault monitoring and reporting

Identical software functions shall not be repeated in different areas but rather grouped together as a shared function or procedure.

6.7 Measuring Units

The SI metric system shall be used for all processing except where otherwise approved. The units of information displayed on MMI displays will be agreed during the CDR.

6.8 Data resolution

The selection of data types and resolution shall be commensurate with the data accuracy required to perform the desired functions.

6.9 I/O Validation

All Analogues Inputs shall be validated to be within the defined limits. Where possible, the health status of controlled devices shall be monitored and reported to a higher-level system.

6.10 Synchronisation

Two methods of synchronisation are allowed, the selection of which shall be commensurate with the time accuracy requirements and shall be subject to approval during the PDR.

a. Network synchronisation: An NTP server will provide accurate GPS time to all subsystems requesting this via Ethernet. The accuracy of this time should be better than 150ms and would be suitable for most applications.

b. Hardware synchronisation: A precision hardware time signal (e.g. 1 pulse-per-second and 1 kHz) will be made available to all items requiring very accurate time (e.g. TCS server, Tracker Computer, Payload Computer and Instrument Computers). A computer input reads this signal and synchronises SW functions accordingly. To utilise this accuracy will require an appropriate selection of operating system, e.g. RT Linux. Accuracy of better than 0.1ms should be achievable.

6.11 Redundant Code

All redundant code and variables shall be removed.

6.12 Software Comments

Over and above the software development documentation described in section 4, the following documentation shall form and integral part of the software code in the form of comments or function help:

• Each software module shall have a header or associated “help” definition describing the following:
  o The name and purpose of the module
  o The inputs and outputs of the module and their types
  o A detailed description of the functions performed by the module. (This may be English narrative or pseudo-code).
• A definition/description of local and global variables used in the module
• English description of the actions performed by instructions. As a guideline give one line of comments per two lines of code for text-based software. For Labview software each VI shall have help information defined, as indicated in the first bullet.

6.13 Self-changing code

Self-changing code shall not be allowed without the consent of the SALT Systems Engineer.

6.14 Manual Operation

Each of the following major items shall have the capability of operating in either an “Automatic” or “Manual” fashion. In the automatic mode the commands from a higher-level system (i.e. the TCS) shall have priority while in manual mode, the item’s own MMI shall have priority.

• Tracker Computer
• Payload Computer
• Dome Controller
• Structure Controller
• Primary Mirror segment controller
• Primary Mirror edge-sensor computer
• Primary Mirror CCAS computer
• Building Management System
7 Specific PLC Software Requirements

7.1 Scan Cycle
Scan cycles shall be as short as possible.

7.2 Assignment of Memory Areas
PLC memory shall be mapped into logical areas as per identified function. Any scratch-pad-memory usage shall be avoided.

7.3 Retained Data
Any critical data shall be retained upon program exit.

7.4 Cycle of Inputs and Outputs Scanning
The program structure shall ensure timely servicing of all Digital and Analogues Inputs and Outputs.

7.5 Memory and Cycle Time Optimisation
Techniques shall be implemented to optimise memory and the program execution time.

7.6 Programming Segments
Programming shall be grouped into segments that are commented and functional.

7.7 I/O Manipulation
Inputs and Outputs shall be addressed with the maximum bit resolution possible.

7.8 Device Control
Devices shall be controlled safely under normal operational conditions as well as during maintenance activities.

7.9 Interlocks
Interlock scanning shall be cyclical. No conditional scanning shall be allowed.
8 Specific Axis-controller Software Requirements

8.1 Axis Controller functions

The distribution of functions between the host computer and an axis controller are to be presented and approved during the PDR. Control complexity, control bandwidth and timing requirements will be considered.

8.2 Synchronisation

Where the equipment controlled by axis controller performs time-critical functions, hardware synchronisation with the host PC shall be used unless otherwise agreed.
9 **Specific PC Software Requirements**

9.1 **Operating Systems**

The following Operating systems will be allowed but will require approval on a case-by-case basis:

- For applications requiring hard real-time control or with strict time synchronisation (e.g. Tracker Computer, Payload Computer): RT Linux (the specific version of RT Linux is TBD1)
- For applications not requiring hard real-time control or strict time synchronisation (e.g. MMI workstations, BMS): Redhat Linux or Windows NT (the specific versions are TBD2). Factors to be considered in deciding which of these two are appropriate include availability of existing software, data security, robustness requirements and ease of maintenance.

9.2 **Development Software**

Labview version 6i is the preferred programming language for SALT and shall be used for all software on all PC's except where agreed otherwise. If version 6i for Linux is not available by the time software development on Linux platforms are due to start, Labview version 5 will be accepted.

As Labview is somewhat different to traditional text-based languages, it is highly recommended that software developers receive adequate training prior the software design. As with all software, poor structure and design will lead to unreliable and un-maintainable software, despite Labview’s apparent ease-of-use. The “National Instruments Labview Development Guidelines” (part of the Labview 6i documentation set) shall be adhered to in this regard.

9.3 **Application Software**

The Labview Style Guide in the “National Instruments Labview Development Guidelines” shall be followed. Specific detailed requirements such as font size, default VI properties and menu layouts will be agreed at the PDR. The Software Engineering Primer in “Labview Power Programming” and “Internet Applications in Labview” are recommended reading.

9.4 **Man-Machine Interfaces**

All operational and maintenance MMI’s shall be designed to run from a separate machine across an Ethernet network, using TCP/IP protocol. Labview’s Datasocket provides this functionality and are the preferred interfacing methods (TBC). The precise operation of this mechanism will be defined by the client in co-operation with the developer.

The content, communication interface and appearance of subsystem operational MMI’s (most notably the Tracker Acquisition and Guidance displays) shall be developed in co-operation with the SALT Systems Engineer and the TCS MMI developers to ensure optimal operational integration between these software items.
10 Deliverables

Unless agreed otherwise, at least the following items shall be delivered for each Item listed in Table 1.

a. Original documentation and electronic media of all the bought-out software installed on the item (including operating systems and device drivers).

b. Original documentation and electronic media of the software development environment in which the software was developed (e.g. compilers, version control tools etc.), unless agreed otherwise.

c. Development environment and operating system configuration data (e.g. memory map set-ups, compiler directives, copy of Linux configuration files, Windows INI files, registry files) on CD-ROM.

d. Documentation as described in section 4 (if applicable according to Table 1), including the following:
   - Software Development Plan
   - Software Requirement Specification
   - Software Design Document
   - Software Code Review report/minutes
   - Software Acceptance Test Procedure
   - Software Acceptance Test Report

e. All application software source code, compiled software, installation software and configuration information on CD-ROM.

f. Calibration, Maintenance and operating procedures if applicable

g. A Version Definition (See section 11)

h. A final version of the Safety Certificate
11 Configuration Control

Each SW Item shall be uniquely identified by a SALT configuration number and name obtainable from the appropriate SALT subsystem manager. Each software module shall also be uniquely identified.

The software supplier shall at all times during the software coding, testing and integration process, maintain a system whereby each software module has a revision number reflecting changes made. Changes after initial SW integration shall be controlled and documented. Every update to that module shall result in a change in the revision number of that module with a comment describing the nature of the update. Modules of previous revisions shall not be overwritten or destroyed but kept for recovery purposes.

The integrated software comprising many modules (i.e. the SW Item), shall also have a unique version number at critical stages in the process (e.g. at Software Testing, delivery etc.). A document (the Version Definition) shall be maintained which records the included modules and their revision numbers making up each version of a SW Item.

The following numbering scheme is preferred for revisions and versions:

Example: Tracker Payload SW Version 2.32 comprises the following modules
- Module 1_1: Initialisation Revision 1.5
- Module 1_2: Hardware set-up Revision 3.1
- Module 2_1: Mode control Revision 2.9
- ...............  

Where the number before the dot is incremented with major changes to the software or to mark a significant event (e.g. software delivery) and the number after the digit changes with minor modifications.