Southern African Large Telescope

SALTICAM

Document Number 3390BP0001: Software Development Plan

Darragh O'Donoghue
Luis Balona
Dave Carter
Etienne Bauermeister
Geoff Evans
Willie Koorts
James O'Connor
Faranah Osman
Stan van der Merwe

Issue 1.2
22 October 2002

Issue History

Number And File Name	Person	Issue	Date	Change History
	DOD	1.0	10 Aug 2002	VI FDR version
3390BP0001 SALTICAM Software Development Plan Issue 1.1.doc		1.1	16 Sep 2002	Post VI FDR Development
3390BP0001 SALTICAM Software Development Plan Issue 1.2.doc		1.2	22 Oct 2002	Post software PDR update

TABLE OF CONTENTS

ACRONYMS AND ABBREVIATIONS

AC	Acquisition Camera (SALTICAM mode)
ACSI	Acquisition Camera and Science Imager (SALTICAM configuration)
ATP	Acceptance Test Procedure
ATR	Acceptance Test Report
BITE	Built-in Test Equipment
BMS	Building Management System (formally called the Environmental Control Computer)
CCD	Charge-coupled Device (Camera)
CDR	Critical Design Review
COTS	Commercial off the shelf
FDR	Final Design Review = CDR
HET	Hobby-Eberly Telescope
I/O	Input/Output (Device)
ICD	Interface Control Dossier
MMI	Man-Machine Interface
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
OEM	Original Equipment Manufacturer
PC	Personal Computer
PDR	Preliminary Design Review
PFIS	Prime Focus Imaging Spectrograph
PI	Principal Investigator (Astronomer)
PLC	Programmable-Logic Controller
RT	Real-time
SA	SALT Astronomer
SALT	Southern African Large Telescope
SDD	Software Design Document
SDP	Software Development Plan
SI	Scientific Imager (SALTICAM mode)
SO	SALT Operator
SPCT	Single-point Communication Test
SRS	Software Requirement Specification
SW	Software
TBC	To Be Confirmed
TBD	To Be Determined
TCS	Telescope Control System
VI	Virtual Instrument (Labview function) *OR* Verification Instrument (SALTICAM mode) (context will define which is meant)

1 Scope

This document specifies the development process for all software for SALTICAM, the Verification Instrument, Acquisition Camera and Scientific Imager of the Southern African Large Telescope (SALT). It indicates the applicable specifications, development process and specific coding and documentation practices that are applicable, based on the SALT Software Standard.

The purpose of SALTICAM is to interact with the SOMMI and/or SAMMI to enable them to use the SALTICAM hardware to obtain images of the focal plane of SALT. Such images should be displayed on the SALTICAM PC monitor near the SO or SA, used for centroiding and closed loop guiding, or stored on the science database.

The SALTICAM software comprises several different units running on various computers.

2 Referenced Documents

The following documents are referenced in this standard.

3300AS0001	SALTICAM Specification Issue 2
3390AS0001	SALTICAM Software Specification Issue 1.3
1000AA0030	SALT Safety Analysis
1000AB0044	SALT Labview Coding Standard
1000BS0010	SALT Software Standard
1000AS0040	SALT Operational Requirement

3 Description of Software To Be Developed

The requirements of the SALTICAM software are defined in the SALTICAM Specification and the SALTICAM Software Specification referenced in section 2. SALTICAM software comprises the following computers and units/applications. Only the software in **bold** is new application software that is covered by the development plan. The software in ***bold italics*** is assumed to be the responsibility of the SALT Project.

a. SALT Operator Workstation
   - ***SALT Operator SALTICAM MMI Software (designated SALTICAM SOMMI). This is the main operator interface to SALTICAM in Verification Instrument (VI) or Acquisition Camera (AC) modes.***
   - Labview Data Socket (part of the standard Labview Application)
   - Internet Explorer or Netscape Web Browsers
   - *This machine also contains applications that are developed by the suppliers of the various telescope subsystems. They are generically referred to as "Subsystem MMI's".*
   - Observation Planning tool (see next paragraph)

b. SALT Astronomer Workstation
   - ***SALT Astronomer SALTICAM MMI Software (designated SALTICAM SAMMI). This is the main astronomer interface to SALTICAM in Science Imager (SI) mode.***
   - Labview Data Socket (part of the standard Labview Application)
   - Internet Explorer or Netscape Web Browsers
   - *This machine also contains applications that are developed by the suppliers of the various telescope instruments. They are generically referred to as "Instrument MMI's"*
   - Observation Planning Tool (also used on the PI Computer). This software allows one to simulate certain telescope operations to determine the feasibility of an observation and to interact with the Observation Planning and Star Catalogue Databases (see Data Processor).

c. SALTICAM PC

- **SALTICAM Kernel Software (designated SALTICAM KER). This will interact with the SALTICAM SOMMI and SALTICAM SAMMI residing in the SOMMI and SAMMI machines. SALTICAM KER will also interact with SALTICAM MMI and SALTICAM CON as described below:**
- **SALTICAM MMI Software (designated SALTICAM MMI). This is the interface to SALTICAM for development and maintenance via the SALTICAM PC keyboard. It will be similar to SALTICAM SOMMI and SALTICAM SAMMI. It is possible that this may substitute for SALTICAM SAMMI.**
- **SALTICAM Control Software (designated SALTICAM CON). This is software that will receive instructions from SALTICAM KER, and control all the hardware.**
- Labview Data Socket (part of the standard Labview Application)(if available for Linux).
- **SALTICAM PCI Card Software (designated SALTICAM PCI). This is software that is supplied by Astronomical Research Cameras with their SDSU II CCD controllers. If Real Time Linux is used, its functionality within the Real Time Linux operating system environment will be emulated by SAAO developed software.**
- **SALTICAM SDSU II Control Software (designated SALTICAM SDSU), including the software in the subsytem controller. This is software that is initially supplied by Astronomical Research Cameras with their SDSU II CCD controllers. The supplied software will be used as a prototype for an SAAO developed equivalent, tailored for the SALTICAM application.**
- *This machine will contain no other applications.*

d. Data Reduction PC
   - This will be very similar to the PI computer, but will be located at SALT.
   - **SALTICAM Data reduction pipeline for SI mode (designated SALTICAM DRED).**

In addition, the SALTICAM software will interact with these machines forming part of the SALT TCS. Their main applications software units are also indicated:

e. TCS Server
   - SALT TCS Server application
   - Labview Data Socket (part of the standard Labview Application)

f. Data Processor
   - Observation Planning database and query engine. This database interacts with the Observation Planning Tool, SOMMI and SAMMI, to exchange information regarding planned observations.
   - Star catalogue query engine. This database provides information used by SOMMI and SAMMI to display fiducial overlays on the Acquisition and Guidance displays, to assist field identification by the users.
   - Web Server. This provides the Astronomer, Operator and PI with an interface to retrieve and organise science and telescope data. The observation planning tool interface to the databases may be routed through this server. All tools required by the PI's also need to be downloadable from here.
   - Science database. This is the organised storage and retrieval of all instrument configuration, calibration, science and telescope data pertaining to science observations made.
   - Data Parser and Unparser (also used on the PI Computer). This software compresses data to/from the PI in order to minimise internet bandwidth requirements.
   - Star catalogue information (from existing catalogues)

g. Event Logger
   - Event Logger application. This software is used to record, retrieve and display user-defined events, based on data flowing between the TCS components and the telescope subsystems. A second function is to display telescope status and failure information that is vital to both the Astronomer and Operator.
   - Weather display. This application receives telescope environmental information and displays this to the operator and astronomer, with alarms where appropriate.
   - Weather web server (standard Labview application)

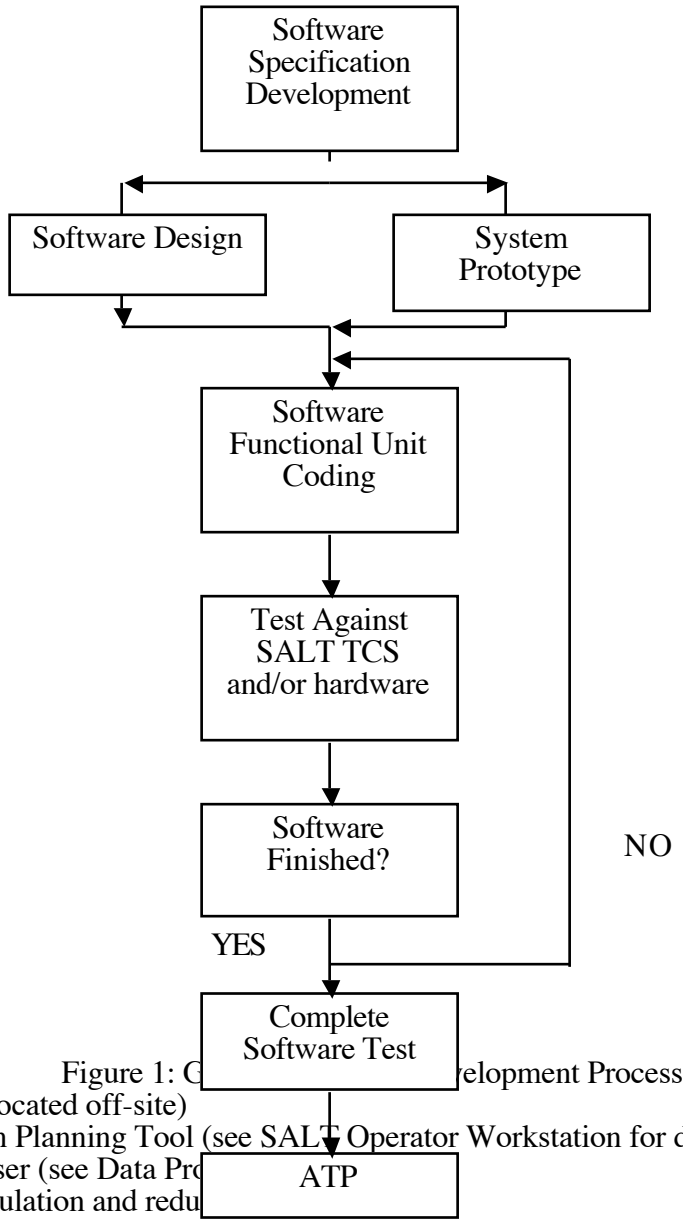

Figure 1: C velopment Process

h. PI Computer(s) (located off-site)
   - Observation Planning Tool (see SALT Operator Workstation for details)
   - Data unparser (see Data Pr
   - Data manipulation and redu

4 Development Sequence and Schedule

Figure 1 indicates the planned SALTICAM software development sequence. Table 1 shows the planned schedule for the work.

Table 1: SALTICAM Software Schedule

Milestone	Date
Development Plan	12 Aug 2002
Specification	15 Sep 2002
Design & Prototype	30 Sep 2002
Coding & Test of Modules Complete	31 Jan 2003
Integration Complete	28 Feb 2003

ATP	Mid-March 2003
Science data reduction pipeline	1 Oct 2003

4.1 Requirements Analysis

A distinct software specification (SRS) shall be written for each of the following software items, using the document number 3390AS0001:

Title	Designation
SALTICAM KER Software Specification	SALTICAM KER
SALTICAM MMI Software Specification	SALTICAM MMI
SALTICAM Control Software Specification	SALTICAM CON
SALTICAM PCI Card Software Specification	SALTICAM PCI
SALTICAM SDSU Control Software Specification	SALTICAM SDSU
SALTICAM Data Reduction Software Specification	SALTICAMS DRED

We assume there already exist requirements analysis for:

Title	Designation
SALTICAM SOMMI Software Specification	SALTICAM SOMMI
SALTICAM SAMMI Software Specification	SALTICAM SAMMI

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*:*

a. The **software architecture** showing the major software modules (building blocks) per developed software item and identifying the major data flowing between modules.
b. 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.
c. 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.
d. A diagram or table showing the required **software timing** including aspects such as data latency, real-time requirements and synchronisation with the real-world.
e. An **input and output allocation**, identifying the major inputs and outputs between the software and hardware.
f. The detailed **software requirements definition** of each software item, according to the complexity and type of the software item.
   - 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)
   - User-configurable features need to be defined in concept
   - Calibration/adjustment/set-up procedures methods need to be defined in concept
   - Where software has direct interaction with the hardware (e.g. I/O), interfacing details need to be specified (e.g. drivers)
   - 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.
g. **Verification and metrics methods** for the measurement of software compliance to the requirements of the Software Requirements Specification shall be defined.

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

4.2 Software Specification Review (PDR)

Prior to starting the software design, the Software Requirement Specification shall be reviewed by the development team and the SALT Team. The purpose of the review is to verify that the software requirements have been correctly derived from the SALTICAM Specification, check that adequate analysis has been performed in defining the software requirements and to co-ordinate the software requirements with other parts of the TCS.

4.3 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**.
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. For Labview software, the mechanism of data flow (i.e. wires or VI server calls) shall be identified.
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 variables, interrupt operation, timing implementation and shall be defined. The design shall be documented in pseudo-code, flow diagrams or English narrative.

4.4 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.

4.5 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.

4.6 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.

4.7 Module Testing

Software modules shall be individually tested prior to integration with the other modules, to an appropriate level. Testing a module may 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, albeit informally.

4.8 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. The precise hardware and software configuration tested shall be defined. From this point forward, all software changes shall be logged. A TCS Server simulator shall be used to verify the communications interface to each computer *prior to delivery of that computer and its software by the developer*. A communication test using the simulator shall be part of each computer item's ATP.

At this point the software shall be fully under Configuration Control (see section 9) and all software changes managed.

4.9 Integrated HW and SW Testing

Incorporated into 4.8.

4.10 Subsystem Commissioning and Integration

The next step of the process is to progressively integrate the software with other parts of the TCS. This Commissioning is complete when the SALTICAM Software ATP, which verifies the performance 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 instrument. Only when the SALTICAM ATP has been successfully completed, can each SW item be said to be complete.

4.11 Software handover

During step 4.10, the responsibility for maintenance of the software is transferred from the original developer to the maintenance crew. 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 8. 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*

5 Software Safety

5.1 Safety certificate

A Safety Certificate shall be issued for SALTICAM. The certificate shall identify all the software items that form part of the SALTICAM software suite.

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 the operator via the Event Logger.

Detection of communication failure shall be facilitated by using the "Validity Word" in the communicated Labview data, or a similar method for non-Labview Software.

Each software item 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

SALTICAM software shall be in a safe state when un-initialised or switched off. Similarly, un-initialised inputs (e.g. from other subsystems) 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 operator
c. Check the integrity of the processing hardware and memory using simple arithmetic checks
d. Check communication with and correctness of peripheral devices (if applicable)
e. Verify the correctness of configuration data and then initialise variables accordingly
f. Check communication with interfacing computers
g. If all operations are successful, report "System Okay" to the operator 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.

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 component shall be uniquely identified with a sensible name. File extensions native to the programming language used, shall be adhered to (i.e. Labview files *.vi, *.glb, *.ctl and *.rtm)

All variables, memory and block naming shall be clear, logical and understandable. A uniform convention

shall be used throughout an item, preferably using whole English words. Where compilers/interpreters do not support long variable names, 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 operator must be able to send a "reset" command across the Ethernet to any computer to trigger initialisation). This is not applicable to MMI applications.

6.3 Data

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 1Hz and be sent to the Event Logger:

- Item Mode
- Item Health Status
- Fault list

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

Where internal variables may assist diagnostics, their values should also be transmitted to the Event Logger.

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 – see SALT Labview Coding Standard) 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 for angles, which shall be in Radians. The units of information displayed on MMI displays shall be in "human-friendly" units and 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

N/A.

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 10MHz) will be made available to all items requiring very accurate time (e.g., Tracker Computer, Payload Computer and Instrument Computers). A computer input reads this signal and synchronises SW functions accordingly.

6.11 Unused Code

All unused code and variables shall be removed from the software.

6.12 Software Comments

Over and above the software development documentation, 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:
  - The name and purpose of the module
  - The inputs and outputs of the module and their types
  - 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 SW code. 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.

6.14 Manual Operation

N/A.

6.15 Communication methods

Communication between computers on the Ethernet network, shall be TCP/IP-based Data socket communication or http://, as agreed.

A central database (ICD) shall identify the source, destination, type and update frequency of each parameter. Communications software for Labview items will be provided by SALT, according to the required data types.

7 Specific SALTICAM Software Requirements

7.1 Operating Systems

The Real Time Linux Operating System is proposed for the SALTICAM PC. (It is possible that standard Linux may suffice).

7.2 Development Software

The following development environments are proposed:

- SALTICAM KER: Labview 6
- SALTICAM MMI: Labview 6
- SALTICAM CON: C
- SALTICAM PCI: C
- SALTICAM SDSU: DSP Assembler
- SALTICAM DRED: SQL, C

7.3 Application Software

The SALT Labview Coding Standard shall be applied to all Labview software. The Software Engineering Primer in "Labview Power Programming" and "Internet Applications in Labview" are recommended reading.

7.4 Man-Machine Interfaces

All MMI's shall be subject to approval by the Project Scientist.

8 Deliverables

Unless agreed otherwise, at least the following items shall be delivered for each software item.

a. Original documentation and electronic media of all the bought-out software installed on the computer (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 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 9)

The following will be available for the SALTICAM Software as a whole:
h. A final version of the Safety Certificate
i. A Software Development Plan (this document)
j. Acceptance Test Procedure and Report

9 Configuration Control

Each SW Item shall be uniquely identified by the SALT configuration number and name defined in section 4.1. Each software module shall also be uniquely identified.

During the software coding, testing and integration process, maintain the Labview version control tool shall be used, 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. 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 A: *Initialisation* Revision 5
- Module B: *Hardware set-up* Revision 31
- Module C: *Mode control* Revision 29
- ……

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.