TITLE:

Deploy

BepiColombo - Modelling Approach

DOCUMENT STATUS SHEET

Issue	Date	Modified Items / Reason for Change
1	29.08.2008	First issue

TABLE OF CONTENTS

1. Introduction

1.1 Purpose of the document

This document is describing the approach for modelling a limited subset of BepiColombo services in EventB. The document identifies a subset of BepiColombo requirements and places them within the modelling framework. However, this setting is only temporary and more requirements may be added in future.

The document is not self-contained and it is meant to be read together with the documents listed in Section 1.3.

1.2 Overview of the document

The document is structured as follows:

Section 1 is the introduction. It provides the purpose of the document and the list of reference documents.

Section 2 describes the BepiColombo software in general.

Section 3 identifies a subset of BepiColombo functionality that will be considered in modeling using EventB.

Section 4 describes modeling decisions and outlines the modeling approach.

Section 5 is an appendix containing the detailed list of identified requirements to be modeled.

1.3 Referenced documents

[RD1] BepiColombo SIXS/MIXS OBSW, Software Requirements Specification, BC-SIX-RS-03105, Issue 1.1

[RD2] SIXS/MIXS Onboard Software Architectural Design Document, BC-SIX-DS-03101, Issue 0.2 (draft)

[RD3] SIXS Experiment Interface Document Part-B , BC-EST-RS-02518, Issue 3

[RD4] MIXS Experiment Interface Document (EID.B), BC-EST-RS-2517, Issue 1.1

[RD5] BEPICOLOMBO SIXS FDIR Definition Document, BC-SIX-TN-00008, Issue 1

2. General Description of BepiColombo Software

The main goal of BepiColombo mission is the exploration of the planet Mercury. The mission comprises various scientific studies, e.g., investigation on the geological evolution of the planet, analysis of its internal structure and a surface, determination of the global surface temperature etc.

For the purpose of fulfilling its scientific goals, BepiColombo mission will send two orbiters. One of these is Mercury Planetary Orbiter (MPO) responsible for carrying remote sensing and radioscience instrumentation. An important part of this element is Data Processing Unit (DPU).

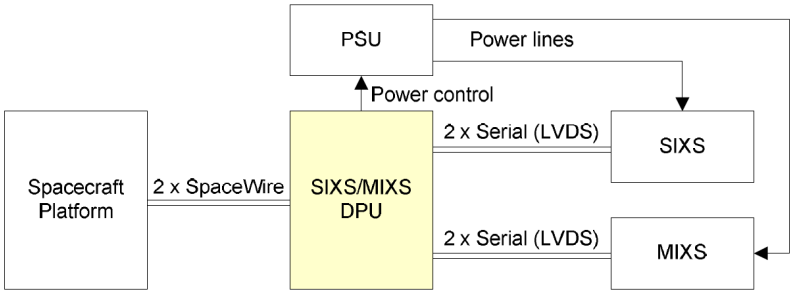


Figure 1 System architecture

SIXS/MIXS DPU is used to control the power of a specific instrument (SIXS/MIXS) and its operating states, monitor instrument operation, handle telecommand (TC) and telemetry (TM) communication.

Solar **I**ntensity **X**-ray and particle **S**pectrometer (SIXS) records the radiation from the Sun at the position of the spacecraft, while **M**ercury **I**maging **X**-ray **S**pectrometer (MIXS) records the fluorescent X-rays from the planet surface. To achieve this, both instruments contain two sensor units:

- SIXS-X (X-ray spectrometer),
- SIXS-P (particle spectrometer),
- MIXS-T (telescope) and
- MIXS-C (collimator).

The SIXS/MIXS DPU is connected to the BepiColombo spacecraft via SpaceWire interfaces which are used to receive telecommands from the spacecraft and transmit science and housekeeping telemetry data to the spacecraft. The science telemetry data is then transmitted to ground during spacecraft visibility.

Both SIXS and MIXS instruments are connected to the SIXS/MIXS DPU via serial interfaces which are used to command instrument mode change and transmit science and housekeeping data.

The power of SIXS and MIXS instruments is provided by Power Supply Unit (PSU) and controlled by the SIXS/MIXS DPU.

The BepiColombo On-Board software (OBSW) (see Figure 2) consists of three main different software components:

- The Core Software (CSW).
- The SIXS instrument Application Software (SIXS ASW), handles interfaces towards the corresponding SIXS instrument.
  - SIXS-P ASW and SIXS-X ASW designate the software component controlling P and X sensor units (SU).
- The MIXS instrument Application Software (MIXS ASW), handles interfaces towards the corresponding MIXS instrument.
  - MIXS-T ASW and MIXS-C ASW designate the software component controlling T and C focal plain assembly (FPA).

CSW is common interface software for the MIXS ASW and SIXS ASW. It works as a TC/TM interface with the BepiColombo platform, and on the other end, as an interface for the MIXS/SIXS DPU and MIXS/SIXS Sensor Units.

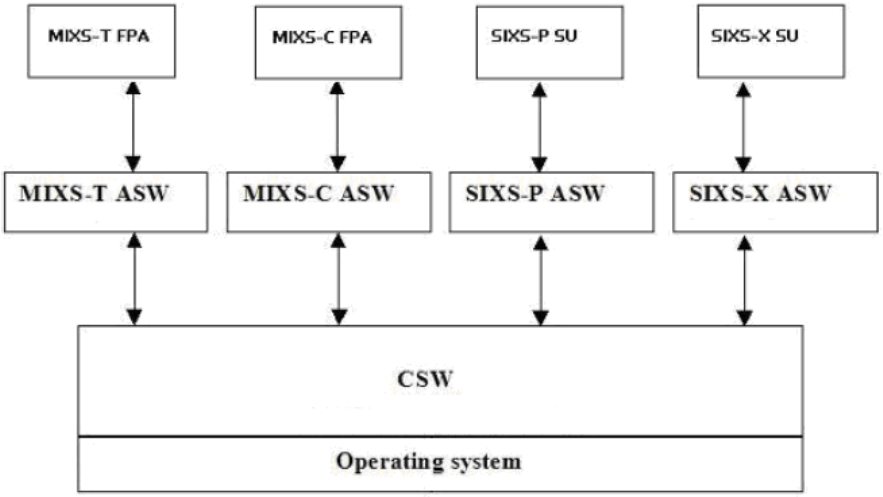


Figure 2 SW architecture

Additional software components provide services of the underlying operating system (RTOS) and services to hide the actual hardware interface calls.

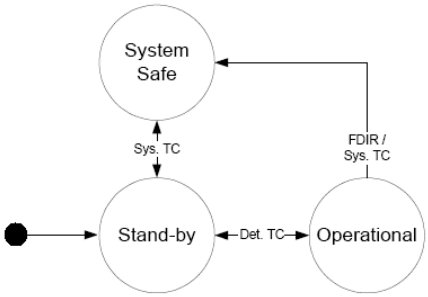
3. BepiColombo Functionality within Deploy

The following section contains part of the BepiColombo requirements that are going to be used as the basis for modelling. Since these requirements are focused around specific state diagrams, only the diagrams are presented and the remaining requirements describing concrete state constraints and functionality are only identified by their requirement number and can be traced to [RD1].

Chosen requirements are grouped based on the related software component. Hence, we first list the relevant requirements for the Core SW, and then for the SIXS and MIXS ASW.

3.1 Requirements for the Core SW

Core SW should implement system level modes and mode changes presented in the following figure:



Sys. TC – TC that is sent to the Core SW
Det. TC – any mode-changing TC sent to any sensor

Figure 3 System modes and mode changes

The system level mode can be seen as the main mode and the sensor modes as sub-modes of the entire system. System level mode is the main mode of the entire system. System mode can be changed/commanded from the spacecraft, however, the Core SW then has to take care that the sensors are in their corresponding submodes.

There are three different system level modes:

1. ***Stand-by*** – all the instruments are in their respective Stand-by, Off or Safe modes (they need not to be in the same mode).

2. ***Operational*** – some instruments are in a mode different from Stand-by, Off or Safe.

3. ***Safe*** – instruments are in their respective Off mode.

Detailed correspondence between system level modes and concrete instrument modes is given in Section 3.4.

Requirements:

Specific requirements for the Core SW are listed in Appendix A.1.

Allowed system mode transitions are given in the following mode transition table.

To\From	Stand-by	Operational	System Safe
Stand-by	-	TC	TC
Operational	TC	-	FDIR/TC
System Safe	TC	N/A	-

Table 1 Allowed system mode transitions

3.2 Requirements for the SIXS ASW

SIXS-(P/X) ASW should implement (sub)modes and (sub)mode changes for both sensor units of SIXS instrument: SIXS-P and SIXS-X.

3.2.1 SIXS-P ASW

SIXS-P ASW is responsible for handling the operating modes and mode changes of the SIXS-P sensor unit, as described in the following figure.

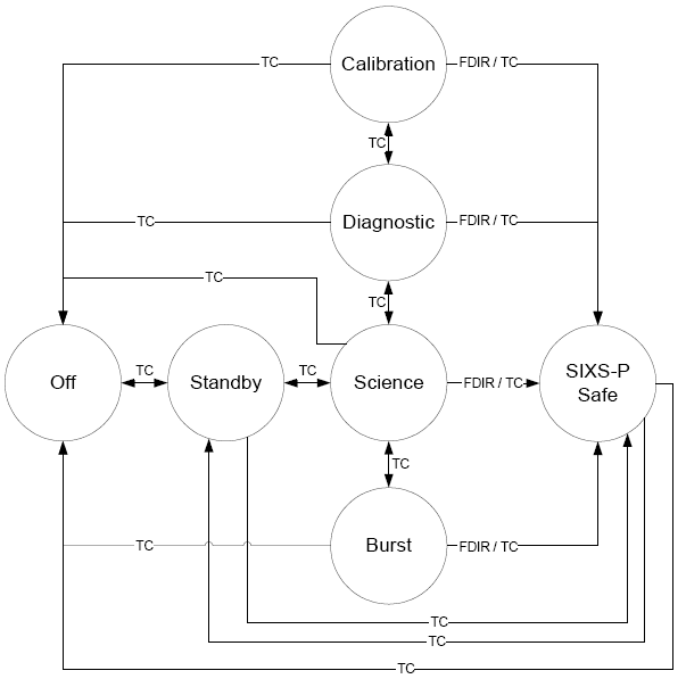


Figure 4 SIXS-P modes and mode changes

SIXS-P has seven different operating modes:

1. ***Off*** – SIXS-P sensor unit is not powered.
2. ***Stand-by*** – sensor unit is powered on but a detector firmware is not running. Only housekeeping functionality is enabled.
3. ***Science*** – a detector firmware is running. Science data and housekeeping data are normally transmitted. This is a normal operating mode of the SIXS-P SU.
4. ***Burst*** – a detector firmware is running. Science data is transmitted at an increased rate while housekeeping data is transmitted normally.
5. ***Diagnostic*** – a detector firmware is running. Raw event samples are transmitted instead of normal science data. Housekeeping data is transmitted normally.
6. ***Calibration*** – the same as diagnostic mode but calibrator is on.
7. ***Safe*** – sensor unit is in safe state that is similar to stand-by. Only a restricted set of TCs (e.g. go to Off and Stand-by mode) are allowed.

Requirements:

Specific requirements for the SIXS-P ASW are listed in Appendix A.2.

Allowed mode changes are described in the mode transition table below.

From\To	Off	Stand-by	Science	Burst	Diagnostic	Calibration	SIXS-P Safe
Off	-	TC	N/A	N/A	N/A	N/A	N/A
Stand-by	TC	-	TC	N/A	N/A	N/A	TC
Science	TC	TC	-	TC	TC	N/A	FDIR/TC
Burst	TC	N/A	TC	-	N/A	N/A	FDIR/TC
Diagnostic	TC	N/A	TC	N/A	-	TC	FDIR/TC
Calibration	TC	N/A	N/A	N/A	TC	-	FDIR/TC
SIXS-P Safe	TC	TC	N/A	N/A	N/A	N/A	-

Table 2 Allowed mode transitions for SIXS-P sensor unit

3.2.2 SIXS-X ASW

SIXS-X ASW is responsible for handling the operating modes and mode changes of the SIXS-X sensor unit, as described in the following figure.

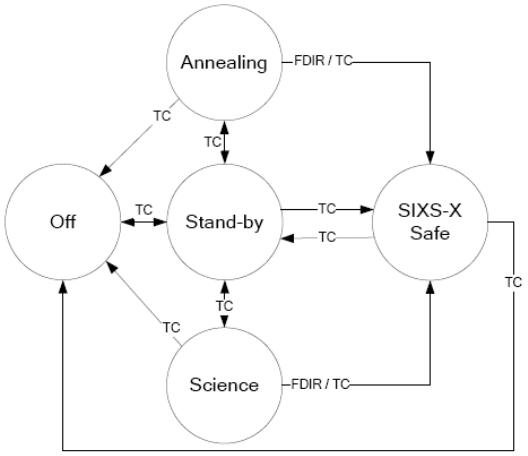


Figure 5 SIXS-X modes and mode changes

SIXS-X has five different operating modes:

1. ***Off*** – SIXS-X sensor unit is not powered.

2. ***Stand-by*** – sensor unit is powered on but a detector firmware is not running. Only housekeeping functionality is enabled.

3. ***Annealing*** – the same as stand-by mode but annealing voltage is on.

4. ***Science*** – a detector firmware is running. Science data and housekeeping data are normally transmitted. This is a normal operating mode of the SIXS-X SU.

5. ***Safe*** – sensor unit is in safe state that is similar to stand-by. Only a restricted set of TCs (e.g. go to Off and Stand-by mode) are allowed.

Requirements:

Specific requirements for the SIXS-X ASW are listed in Appendix A.3.

Allowed mode changes are described in the mode transition table below.

From\To	Off	Stand-by	Science	Annealing	SIXS-X Safe
Off	-	TC	N/A	N/A	N/A
Stand-by	TC	-	TC	TC	TC
Science	TC	TC	-	N/A	FDIR/TC
Annealing	TC	TC	N/A	-	FDIR/TC
SIXS-X Safe	TC	TC	N/A	N/A	-

Table 3 Allowed mode transitions for SIXS-X sensor unit

For both SIXS-P and SIXS-X all states have a transition to *Off* state and all mode transitions are triggered by a telecommand, except for transitions to *Safe* mode.

3.3 Requirements for the MIXS ASW

MIXS-(T/C) ASW should implement (sub)modes and (sub)mode changes for both sensor units of MIXS instrument: MIXS-T and MIXS-C.

3.3.1 MIXS-T ASW

MIXS-T ASW is responsible for handling the operating modes and mode changes of the MIXS-T sensor unit, as described in the following figure.

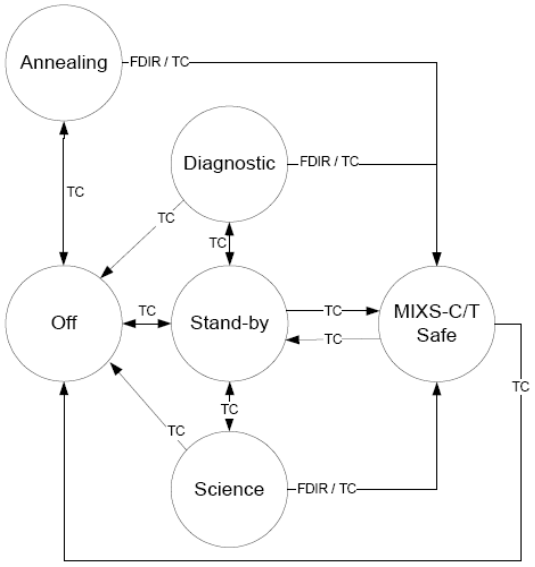


Figure 6 MIXS-T modes and mode changes

MIXS-T has six different operating modes:

1. ***Off*** – MIXS-T FPA is not powered.
2. ***Stand-by*** – FPA is powered on and waiting for configuration. No data is produced.
3. ***Diagnostic*** – FPA is switched on and configured to operate in test mode. Raw image data are put into the telemetry in place of normal science data.
4. ***Science*** – Science data is normally produced. This is a normal operating mode of the MIXS-T.
5. ***Annealing*** – FPA is powered off and annealing voltage is on. No data is produced.
6. ***Safe*** – FPA is in safe state. Only a restricted set of TCs (e.g. go to Off and Stand-by mode) are allowed.

Requirements:

Specific requirements for the MIXS-T ASW are listed in Appendix A.4.

Allowed mode changes are described in the mode transition table below.

From\To	Off	Stand-by	Science	Diagnostic	Annealing	MIXS-T Safe
Off	-	TC	N/A	N/A	TC	N/A
Stand-by	TC	-	TC	TC	N/A	TC
Science	TC	TC	-	N/A	N/A	TC/FDIR
Diagnostic	TC	TC	N/A	-	N/A	TC/FDIR
Annealing	TC	N/A	N/A	N/A	-	TC/FDIR
MIXS-T Safe	TC	TC	N/A	N/A	N/A	-

Table 4 Allowed mode transitions for MIXS-T sensor unit

3.3.2 MIXS-C ASW

MIXS-C ASW is responsible for handling the operating modes and mode changes of the MIXS-C sensor unit, as described in Figure 6.

Requirements:

Specific requirements for the MIXS-C ASW are listed in Appendix A.5.

Allowed mode changes are described in the mode transition table (see Table 4).

For both MIXS-T and MIXS-C holds that all states have a transition to *Off* state. Moreover, all mode transitions are triggered by a telecommand, except for transitions to *Safe* mode.

MIXS instrument can be switched off from any mode in case of emergency. However, it is preferable to command MIXS-T and MIXS-C to *Stand-by* mode first and then switch off.

3.4 Correspondence between System-Level Modes and Sensor Submodes

The following table outlines dependencies between specific sensor submodes and system level modes.

	SENSOR submode							
SYSTEM mode	**Off**	**Stand-by**	**Diagnostic**	**Science**	**Annealing**	**Burst**	**Calibration**	**Safe**
Stand-by	X	X						X
Operational	X	X	X	X	X	X	X	X
Safe	X							

Note: If the spacecraft switches off the power from the DPU box, all the instruments will also be powered off.

3.5 Telecommands and telemetries

In addition to telecommands handled by CSW, the SIXS and MIXS ASW should handle SIXS and MIXS-related telecommands. The ASW should also collect and package science and housekeeping telemetry data from the instruments and send them to the spacecraft.

In this section we identify the telecommand and telemetry packets handled by the CSW and MIXS and SIXS ASW as specified in [RD2].

Service Type	Sub-type	Service Request (TC)	Sub-type	Service Report (TM)
1			1	TC acceptance report – success
1			2	TC acceptance report – failure
1			7	TC execution report – success
1			8	TC execution report – failure
3	5	Enable HK report generation	25	HK report
3	6	Disable HK report generation		

Service Type	Sub-type	Service Request (TC)	Sub-type	Service Report (TM)
3	129	Modify HK report generation interval		
5			1	Event packet – nominal
5			2	Event packet – low severity error/anomaly
5			3	Event packet – medium severity error/anomaly
5			4	Event packet – high severity error/anomaly
6	2	Load memory using absolute addresses		
6	5	Dump memory using absolute addresses	6	Memory dump using absolute addresses
6	9	Check memory using absolute addresses	10	Memory check using absolute addresses
9	129	Accept time update		
17	1	Perform connection test	2	Link connection report
17	128	Test telecommand of maximum length		
21	1	Enable science transfer	3	Science report
21	2	Disable science transfer		
21	128	Reset output buffer		
21	129	Change science data rate		

In addition to these, there are also more specific telecommands and telemetries (private services) for commanding the SIXS and MIXS instruments and they are as defined in [RD3, RD4].

Requirements:

Specific requirements for the PUS Services are listed in Appendix A.6.

4. Modelling Approach

In the previous section we identified BepiColombo functionality which is the subject of our modelling. In this section, we describe the approach to modelling the BepiColombo system (with reduced functionality).

We focus on modelling the overall behaviour of the system, i.e., modelling the system level operating modes and mode transitions triggered either by dedicated telecommands or some failure condition (more details in [RD5]) in case of transition to a *Safe* mode. In several development steps (see Figure 7), we refine the initial, more general model of system behaviour to include the behaviour of different subsystems. Each subsystem (ASW of MIXS-T, MIXS-C, SIXS-P, SIXS-X) has its own operating modes, which are becoming submodes at the system level. While refining the models, we check that the correspondence properties (defining the dependencies between the system and subsystem operating modes defined in the [RD1]) are preserved. These properties are expressed as invariants over the corresponding models.

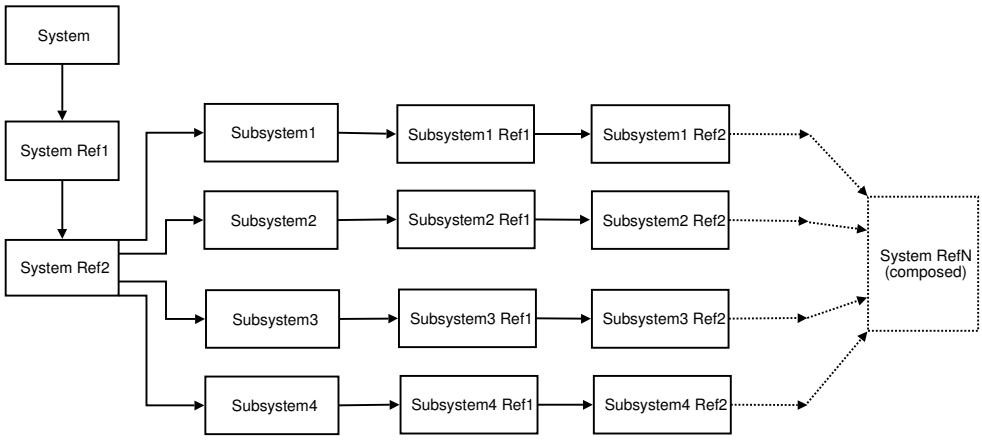


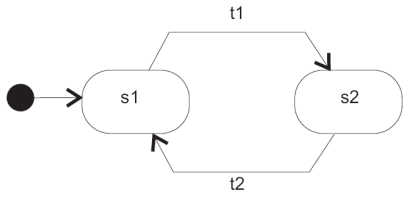
Figure 7 BepiColombo modelling stages

4.1 Some modelling guidelines

In order to handle the complexity of models and increase the efficiency in modelling we adopt a set of rules to be used internally within the modelling team. First, we suggest a pattern for translating a state diagram into an EventB model. Secondly, we define the naming convention.

4.1.1 Pattern for state diagram translation into EventB

Consider the following state diagram:



When modelling in EventB, we first create a *context* for each model describing the structural parts of the system. For instance, to describe some state space S with its elements s1 and s2 in EventB we introduce an enumerated set S = {s1, s2} into the context of the system.

To model the state changes in EventB we introduce a variable state for each associated state diagram. The enumerated set S then defines the type of this variable, i.e., state ∈ S. The initial value of the variable state (initial value corresponds to the initial state s1) is assigned in the INITIALIZATION event of the EventB model.

The transitions (see t1 and t2) between states define events that update the variable state. An event is enabled for execution only if the value of the variable state is equal to the state name from which there is a corresponding transition. For instance, the event which would correspond to the transition t1 occurs only if the variable state has the value s1. Furthermore, transitions may have additional guard conditions constraining the situations in which transition may occur. These guards correspond to the guards of the appropriate EventB events.

4.1.2 Naming convention

General rule is to use common names for all EventB machines, contexts, variables, constants and events if possible but to prefix each common name with the name of the (sub)system where it is used. For example:

- The state variable for each of the (sub)systems should have the common root name state, but when defined for the CSW it should be prefixed as csw_state; similarly for MIXS-T ASW it should be named mixst_state etc.
- The set which introduces the state space for MIXS-T ASW should be defined within the corresponding MIXS-T context as MIXST_STATES, whereas in SIXS-X context it should be defined as SIXSX_STATES etc.
- The events for SIXS-X should have prefixes sixsx; similarly events for MIXS-C should be prefixed with mixsc etc.

4.2 Specifying the system

In detail, modelling BepiColombo system starts by specifying the system level behaviour as presented in Figure 3. Only system level modes are introduced and the way they are changed, suppressing the actual triggers of mode transitions. Namely, details of telecommands and various failure conditions are left undefined. The specification also introduces a simplistic model of TC/TM protocol.

4.3 Refining the system

The following modelling step refines the specification by incorporating more knowledge of TC/TM handling, i.e., introducing different types of telecommands and telemetries as shown in Section 3.5. We specify "placeholders" for modelling TC verification and TC processing and compiling corresponding TMs. We focus on specifying mode management capabilities of the Core SW and constraints regarding allowed telecommands in specific modes.

The required level of detail regarding the functionality behind system level modes is not yet fully determined. Hence, this and possibly following refinement steps aim at incorporating them into the system models. Once it is determined that the system specification (specification of a Core SW) is detailed enough or that it covers everything what was originally planned to be modelled, we continue by modelling the subsystems.

4.4 Modelling the subsystems

Each subsystem is modelled independently as a refinement of the model with the highest achieved level of detail (see System Ref2 in Figure 7). Hence, we have four parallel, very similar developments. This is possible since the instruments we are modelling are independent as well and introducing any interdependency among them would be artificial.

Introducing the details of instruments software behaviour is achieved using *superposition refinement*. It allows us to describe a new behavior but preserve the old one. However, we should guarantee that the new behavior does not create a situation in which the control is consumed indefinitely.

4.4.1 SIXS-P

The first refined model specifies SIXS-P ASW. We specify the behaviour of the SIXS-P as shown in Figure 4 in a similar way as when specifying the system level behaviour. However, an important novelty is introducing invariants as properties defining the correspondence between the system level modes and SIXS-P submodes.

Refining SIXS-P. In a similar way as we introduced the TC/TM handling into the system specification, we introduce it into the specification of SIXS-P subsystem by refining the initial subsystem specification. However, we have to model the way TCs are forwarded from the system level, as well as the mode management on a subsystem level.

Mentioned refinement steps may be changed if needed. Also, refinement process can be continued if encountered that the level of detail in certain requirements is too big to be implemented at once.

4.4.2 SIXS-X

The second independent refinement-based development continues from the specification of the refined system-level behaviour and adds in details of SIXS-X behaviour as shown in Figure 5. We also specify dependencies between the system level modes and SIXS-X submodes in forms of different invariants.

Refining SIXS-X. As for SIXS-P, we implement the details of TC/TM handling and further refine specific, i.e., private services.

4.4.3 MIXS-T and MIXS-C

MIXS-T and MIXS-C ASW share the same behavioural description regarding the modes they operate in and mode transitions. In this sense, they share common mode management logic. However, they have a different functionality and hence only the initial specification of these two subcomponents can be largely reused. Namely, both specification of MIXS-T and specification of MIXS-C start from the refined system-level behaviour and operationally are identical. The only difference is in the names of the variables used to describe their behaviour. Then, further refinements of each of them follow a different requirements selection.

4.5 Composing the system

After each of the instruments software has been specified with enough details, we compose a complete system (see System RefN in Figure 7) continuing from the last refined system level specification (System Ref2 in Figure 7). The final expected system specification is obtained by simulating the subsystem composition – we simply create a system refinement and then copy all the syntactic elements from the refinements of the instruments software. We copy new variables, invariants, events, and modify refined system events as needed. In such a way, we create a system refinement with all the details of the instrument software (all MIXS-T/C and SIXS-P/X at once).

The resulting approach takes into account the non-existence of a feature in EventB tool which would support module-oriented development and composition instead of developing rather monolithic specifications. The justification for this approach is in reducing the complexity while modelling somewhat similar instruments mode management and moreover, different functionality of the independent instruments software. Composed system specification is needed in order to implement certain required dependencies between the system level modes and instrument software submodes. For instance, there is a requirement stating that in the system level *Operational* mode some of the instrument software modes shall be in a mode different than *Stand-by*, *Off* or *Safe*. Hence, we need knowledge of all four instrument software modes to be able to address the above mentioned requirement.

4.6 Extension of the model

Currently planned model of the BepiColombo OBSW will not be complete with respect to all defined system requirements. Here we list some identified problems which still need to be tackled in the future model extensions.

- Our currently planned development does not completely implement system functionality. This is largely due to insufficiently modelled system environment. Obtaining a concrete environment model is seen as a goal of future model extensions. This would include modelling of interfacing hardware like FPGAs which are used for pre-processing of scientific data, calibrator, FPAs etc.
- The model of the OBSW misses an important feature – scheduling. Although more detailed requirements of scheduling are still undefined, we already foresee a need for modelling a concrete scheduling policy for some on-board processes.
- Initial OBSW model considers only very generic PUS services without any details of their implementation. Hence, a telecommand verification function is very simplistic. It can be extended in future models so that it comprises a list of various checks performed on different fields within the telecommand.
- We do not explicitly model any FDIR functionality as it is not yet fully defined. Instead, we only introduce a very abstract representation of a failure condition. This allows us to model the system behaviour even when some errors (e.g., hardware malfunctioning) are present in the system. However, with more concrete environment model we will be able to implement FDIR functionality as well.

4.7 Traceability

This document only describes how the requirements will be traced through modelling.

Each requirement should be uniquely identified by its number. At each modelling phase we introduce a matrix which records the effect of the requirement implementation on the current EventB model – a matrix shows which model elements are affected by requirement implementation. We trace each implemented requirement down to the level of EventB elements (i.e., events, variables, invariants, constants, axioms, theorems, event actions, event guards, variants). Such level of detail should allow us to manage requirements changes more efficiently.

APPENDIX A. Detailed List Of Requirements

A.1 Core Software Requirements

5.1.6:
SW-SR-COR-600, SW-SR-COR-800, SW-SR-COR-900, SW-SR-COR-1000, SW-SR-COR-1100

A.2 SIXS-P ASW Requirements

5.1.7.1:
SW-SR-PSW-200, SW-SR-PSW-300, SW-SR-PSW-400, SW-SR-PSW-500, SW-SR-PSW-600, SW-SR-PSW-700, SW-SR-PSW-800, SW-SR-PSW-900, SW-SR-PSW-1000, SW-SR-PSW-1100, SW-SR-PSW-1600

5.1.7.2:
SW-SR-PSW-1600

5.1.7.3:
SW-SR-PSW-1700, SW-SR-PSW-2200

5.1.7.5:
SW-SR-PSW-2700, SW-SR-PSW-2800

A.3 SIXS-X ASW Requirements

5.1.8.1:
SW-SR-XSW-200, SW-SR-XSW-300, SW-SR-XSW-400, SW-SR-XSW-500, SW-SR-XSW-600, SW-SR-XSW-700, SW-SR-XSW-800, SW-SR-XSW-900, SW-SR-XSW-1000, SW-SR-XSW-1100, SW-SR-XSW-1200

5.1.8.2:
SW-SR-XSW-1200

5.1.8.3:
SW-SR-XSW-1300, SW-SR-XSW-1800

5.1.8.5:
SW-SR-XSW-2300, SW-SR-XSW-2400

A.4 MIXS-T ASW Requirements

5.1.9.1:
SW-SR-TSW-1000, SW-SR-TSW-10100, SW-SR-TSW-10200, SW-SR-TSW-10300, SW-SR-TSW-10400, SW-SR-TSW-10500, SW-SR-TSW-10600, SW-SR-TSW-10700, SW-SR-TSW-10800, SW-SR-TSW-10900, SW-SR-TSW-11000, SW-SR-TSW-11200, SW-SR-TSW-11300

5.1.9.7:
SW-SR-TSW-4000, SW-SR-TSW-4100

5.1.9.9:
SW-SR-TSW-4600, SW-SR-TSW-5100

5.1.9.11:
SW-SR-TSW-5500, SW-SR-TSW-5600, SW-SR-TSW-5700, SW-SR-TSW-6200

A.5 MIXS-C ASW Requirements

5.1.10.1:
SW-SR-CSW-10000, SW-SR-CSW-10100, SW-SR-CSW-10200, SW-SR-CSW-10300, SW-SR-CSW-10400, SW-SR-CSW-10500, SW-SR-CSW-10600, SW-SR-CSW-10700, SW-SR-CSW-10800, SW-SR-CSW-10900, SW-SR-CSW-11000, SW-SR-CSW-11200, SW-SR-CSW-11300

5.1.10.7:
SW-SR-CSW-4100, SW-SR-CSW-4200

5.1.10.9:
SW-SR-CSW-4700, SW-SR-CSW-5200

5.1.10.11:
SW-SR-CSW-5600, SW-SR-CSW-5700, SW-SR-CSW-5800, SW-SR-CSW-6300

A.6 PUS Service Requirements

5.1.5.1:
SW-SR-TCTM-100, SW-SR-TCTM-200, SW-SR-TCTM-300, SW-SR-TCTM-400

5.1.5.2:
SW-SR-TCTM-1000, SW-SR-TCTM-1100, SW-SR-TCTM-1300

5.1.5.8:
SW-SR-TCTM-8000