**ECSS-E-ST-70-01C**
16 April 2010

# Space engineering

## Spacecraft on-board control procedures

**ECSS Secretariat**
**ESA-ESTEC**
**Requirements & Standards Division**
**Noordwijk, The Netherlands**

## Foreword

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering and product assurance in space projects and applications. ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards. Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

This Standard has been prepared by the ECSS-E-ST-70-01 Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority.

## Disclaimer

| | |
|---|---|
| Published by: | ESA Requirements and Standards Division<br>ESTEC, P.O. Box 299,<br>2200 AG Noordwijk<br>The Netherlands |
| Copyright: | 2010 © by the European Space Agency for the members of ECSS |

# Change log

| | |
|---|---|
| ECSS-E-ST-70-01A | Never issued |
| ECSS-E-ST-70-01B | Never issued |
| ECSS-E-ST-70-01C<br>16 April 2010 | First issue |

# Table of contents

## Figures

# Introduction

On-board control procedures (OBCPs) have been implemented on an ad-hoc basis on several European missions over the last 25 years, so the validity and utility of the concept has been amply demonstrated.

The purpose of the present Standard is to define an OBCP concept that can be applied for any mission and which:

- fulfils the needs of all categories of user (system engineers, on-board software engineers, AIT engineers, operations engineers);
- ensures that OBCPs have a development lifecycle that is independent of the remainder of the on-board software (OBSW);.
- conforms with, and extends, existing ECSS monitoring and control standards, namely ECSS-E-70-41 and ECSS-E-ST-70-31.

# 1 Scope

This Standard defines the concept for an OBCP system, identifying the on-board functionality for OBCP execution and the ground functionality for OBCP preparation and subsequent control.

This Standard also defines the development lifecycle for OBCPs and identifies the relationships of this lifecycle with the overall space system, and in particular with the other elements of the on-board software.

This Standard assumes that missions implementing OBCPs are also compliant with ECSS-E-70-41, since a number of services contained therein are invoked in support of the operation of OBCPs and their interaction with the ground.

This Standard may be tailored for the specific characteristic and constraints of a space project in conformance with ECSS-S-ST-00.

# 2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Standard. For dated references, subsequent amendments to, or revision of any of these publications do not apply. However, parties to agreements based on this ECSS Standard are encouraged to investigate the possibility of applying the more recent editions of the normative documents indicated below. For undated references, the latest edition of the publication referred to applies.

| | |
|---|---|
| ECSS-S-ST-00-01 | ECSS system - Glossary of terms |
| ECSS-E-ST-40 | Space engineering - Software |
| ECSS-E-ST-70 | Space engineering - Ground systems and operations |
| ECSS-E-ST-70-31 | Space engineering - Ground systems and operations - Monitoring and control data definition |
| ECSS-E-70-41 | Space engineering - Ground systems and operations - Telemetry and telecommand packet utilization |

# 3
# Terms, definitions and abbreviated terms

## 3.1 Terms from other standards

For the purpose of this Standard, the terms and definitions from ECSS-ST-00-01, ECSS-E-ST-70, ECSS-E-ST-70-31 and ECSS-E-70-41 apply, in particular for the following terms:

**activity**

**event**

**event reporting service**

**ground system**

**on-board parameter**

**operations procedure**

**space project**

**spacecraft**

## 3.2 Terms specific to the present standard

### 3.2.1 automation

replacement of manual operations by computerized mechanisms

### 3.2.2 on-board control procedure

software program designed to be executed by an OBCP engine, which can easily be loaded, executed, and also replaced, on-board the spacecraft

NOTE Depending on the context, OBCP can refer to an OBCP in program source code form, or in OBCP code.

### 3.2.3 OBCP code

complete representation of an OBCP, in a form that can be loaded on-board for subsequent execution

NOTE 1 In previous missions, such code is typically referred to as token code, executable code or bytecode depending on the implementation of the relevant OBCP engine.

NOTE 2 In service 18 of ECSS-E-70-41A, OBCP code is referred to as procedure code.

### 3.2.4 OBCP engine

application of the on-board software handling the execution of OBCPs

NOTE OBCP operations are initiated by means of ECSS-E-70-41 Service 18 telecommands.

### 3.2.5 OBCP language

programming language in which OBCP source code is expressed by human programmers

### 3.2.6 OBCP system

the entire machinery for the creation (in the ground system), uplinking, and on-board handling of OBCPs

### 3.2.7 OBCP step

sequence of OBCP source code statements constituting the smallest operational unit within an OBCP

### 3.2.8 on-board software

software hosted and executed by any programmable on-board computer or processor

### 3.2.9 scheduling

controlling the allocation of OBSW processor (CPU) time for execution of the various OBSW functions, according to a predefined algorithm

NOTE OBSW functions include the OBCP engine and execution of OBCPs.

### 3.2.10 survival mode

configuration of a spacecraft in which it can remain safely without ground segment intervention for a specified period

NOTE Survival mode is also commonly known as safe mode. In survival mode, typically all non-essential on-board units or subsystems are powered off, either to conserve power or to avoid interference with other subsystems, and the spacecraft can be (automatically) oriented to a particular attitude with respect to the sun.

## 3.3 Abbreviated terms

The following abbreviations are defined and used within this standard:

| **Abbreviation** | **Meaning** |
|---|---|

| | |
|---|---|
| **AIT** | assembly, integration and test |
| **AOCS** | attitude and orbit control subsystem |
| **AR** | acceptance review |
| **CDR** | critical design review |
| **CPDU** | command pulse distribution unit |
| **CPU** | central processor unit |
| **DDR** | detailed design review |
| **EBNF** | extended Backus-Naur form |
| **EEPROM** | electrically erasable programmable read-only memory |
| **EGSE** | electrical ground support equipment |
| **FDIR** | failure detection, isolation and recovery |
| **I/O** | input/output |
| **MCS** | mission control system |
| **OBAP** | on-board application procedure |
| **OBCP** | on-board control procedure |
| **OBOP** | on-board operations procedure |
| **OBSW** | on-board software |
| **PDR** | preliminary design review |
| **QR** | qualification review |
| **RAM** | random access memory |
| **SDE** | software development environment |
| **SRR** | system requirements review |
| **TRR** | test results review |

# 4 The OBCP concept

## 4.1 Introduction

The OBCP concept is that of a procedure to be executed on-board, which can easily be loaded, executed, and also replaced, on-board the spacecraft without modifying the remainder of the on-board software.

## 4.2 Stakeholders and application areas for OBCPs

### 4.2.1 Stakeholders

Several categories of OBCP stakeholder are identified, each of whom has a distinct role in a space project, with corresponding responsibilities:

- System engineers (spacecraft and payload);
- On-board software engineers;
- AIT engineers;
- Operations engineers.

There is continuous interaction and cooperation between these stakeholders throughout the development and operation lifecycle of a space system, for example:

- during the spacecraft design phase, operations engineers are involved to ensure the operability of the overall space system;
- during in-orbit operations, system and software engineers support commissioning and troubleshooting activities. The system or software responsibility may be transferred from the satellite prime contractor to the operations organization at some predetermined time after launch (e.g. in the case of long-duration missions).

The potential uses for OBCPs are therefore categorized in clause 4.2.2 according to the domain of application rather than stakeholder category.

### 4.2.2 Domains of OBCP application

#### 4.2.2.1 System design

Typical scenarios where it may be decided to implement on-board functionality as OBCPs rather than as OBSW include:

- **Platform functions**
  - To isolate mission-specific platform functions from the remainder of the OBSW, e.g. thermal control or battery control.
  - To implement one-shot applications that are deleted after use, e.g. deployment of appendices, firing of pyrotechnics.
  - To accommodate the late specification of the detailed FDIR strategy and to facilitate the tuning of this strategy during system testing.

    NOTE It is not the intention to encourage the late definition of the FDIR strategy, but rather to accommodate the reality that the detailed strategy is often late. The decision about whether or not to use OBCPs for FDIR purposes is part of the trade-off addressed in detail in clause 6.
  - To accommodate the late delivery of, or the subsequent removal, addition or replacement of equipment.
  - To facilitate the tuning of on-board configuration sequences for complex equipment or subsystems following system testing, i.e. these sequences are modified directly avoiding the delays inherent in OBSW modification.
- **Payload functions**
  - To accommodate the late definition and tuning of:
    - complex payload configuration or control sequences;
    - monitoring algorithms and recovery actions.

#### 4.2.2.2 On-board software design and development

The benefits of implementing traditional OBSW functions as OBCPs include:

- the relative ease of development and validation of OBCPs vs. OBSW;
- the core OBSW can be made more generic and is hence potentially reusable across many missions, if mission-specific functions are implemented as OBCPs;
- simplification of the OBSW maintenance task, i.e. changes to OBCPs can be easily and safely performed without changing the core OBSW.

OBCPs can also be written by OBSW engineers for their own needs, such as:

- automation of tests;
- investigation and debugging purposes;

- the implementation of short-term workaround solutions for system or software problems without the need to wait for a new software delivery.

#### 4.2.2.3 AIT

OBCPs are useful tools for AIT engineers for testing and validation purposes during the AIT programme on-ground (i.e. not for in-flight use), for example:

- to perform fault injection for robustness testing, especially scenarios that are not feasible with traditional means (e.g. telemetry and telecommand, observation and stimulus);
- to implement long and complex configuration sequences for well-defined and repetitive scenarios;
- to develop temporary functions of the OBSW for testing purposes.

#### 4.2.2.4 Operations and operability

The availability of OBCPs enables operations procedures (both for routine functions and contingency operations) to be executed on-board as an alternative to on the ground (either under manual control or automated in the mission control system). This can streamline the operations (reduction of bandwidth, potential reduction in operations manpower, reduction in the loop delay inherent in ground control, simplification of ground procedures) as well as increasing their overall reliability. The use of OBCPs also enhances the on-board autonomy capabilities and increases the robustness to ground station outages. This applies both where there is continuous visibility from ground (e.g. geostationary missions) and where ground station coverage is discontinuous (e.g. LEO and deep-space missions).

In addition, the use of OBCPs facilitates the adaptation to prevailing mission conditions, e.g.:

- the implementation of on-board solutions to counteract unforeseen failures occurring in orbit;
- unpredictable environment, which can be encountered in deep-space missions;
- the implementation of end-of-life operations (e.g. deorbiting).

## 4.3 Types of OBCP

From the potential application of OBCPs described in clause 4.2.2, two types of OBCP can be distinguished:

1. "**On-board Application Procedures (OBAP)**" which implement part of the basic functionality of the spacecraft, i.e. which are an integral part of the spacecraft design. The overall qualification of the spacecraft requires the integration of the complete set of OBAPs. Historically, this type of OBCP has been called "Application Program" or "Flight Application Program".

   NOTE However, although the set of OBAPs is qualified as part of the spacecraft design, this does not imply that they are not maintained (i.e. possibly updated) after launch.

2. "**On-Board Operations Procedures (OBOP)**" that are used to "operate" the spacecraft, but which are not involved in the qualification of the spacecraft.

Whilst both types of OBCP may use essentially the same on-board capabilities and may even be similar in complexity:

1. There are major differences in how the two types are accommodated on-board in terms of resource allocation, scheduling and accessible services (see clause 5.4.4.5, for example).

2. The very nature of OBAPs requires that they undergo the same engineering processes as any other integral part of the spacecraft design.

   In particular, the lifecycle of an OBAP is intimately tied to that of the spacecraft system (as well as any specific platform subsystem or payload to which it relates), in terms of engineering processes and reviews. The lifecycle of an OBOP, on the other hand, is more akin to that of a ground operations procedure.

   The engineering processes for OBAPs and OBOPs and the corresponding requirements are elaborated in clause 6 of this Standard.

## 4.4 The OBCP system

The **OBCP system** that supports the stakeholder activities consists of an **OBCP preparation environment** located on the ground and an **OBCP execution environment** that is located partly on ground and partly on-board (see Figure 4-1).

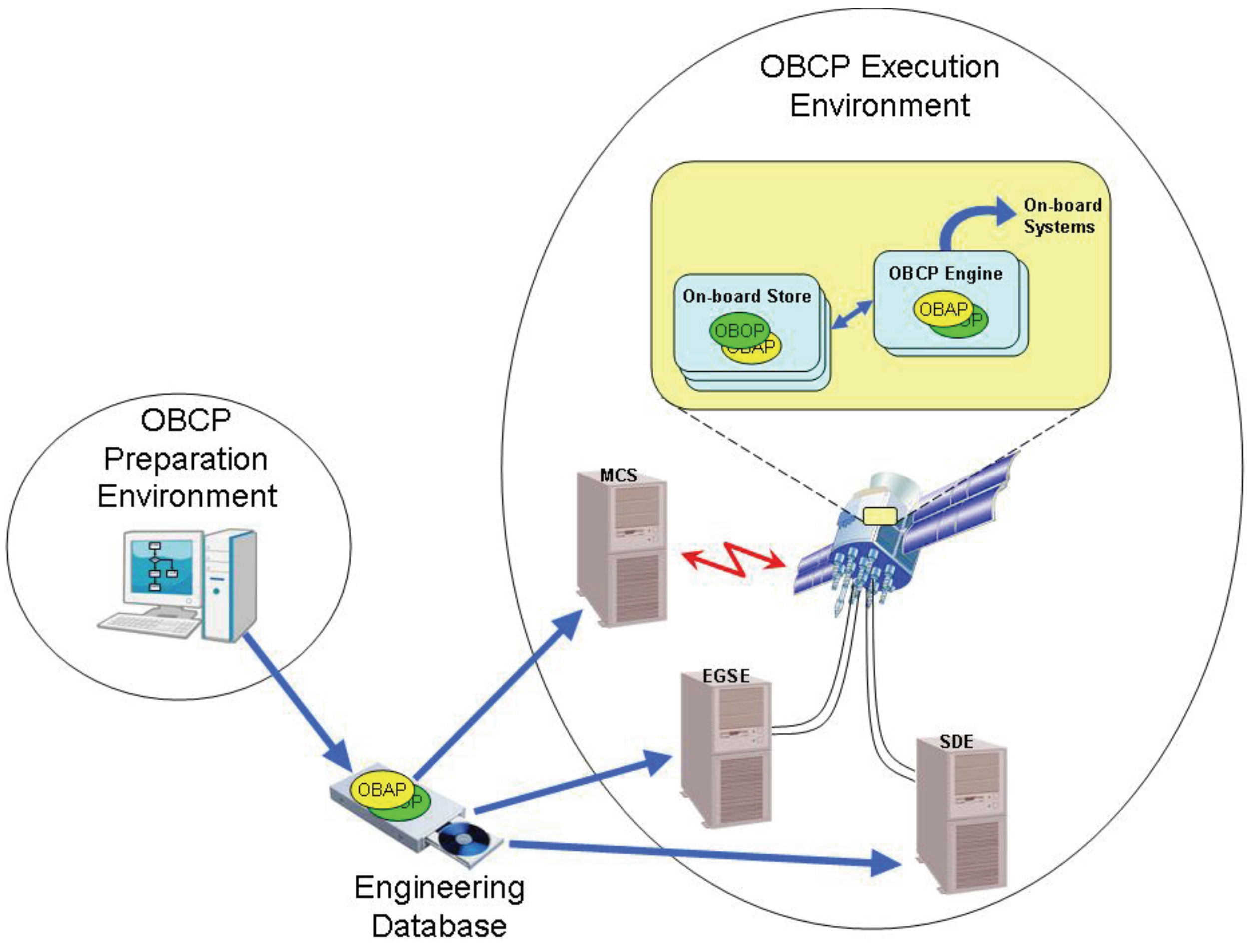

**Figure 4-1 The OBCP system**

The **OBCP preparation environment** is part of the overall test and operations preparation environment and includes:

- editors to support the scripting of OBCPs;
- execution constraint checking functions (e.g. resource utilization);
- consistency checking functions (e.g. compliance with telemetry and telecommand database definitions);
- OBCP configuration management;
- OBCP code production;

  NOTE OBCPs exist in two forms: the OBCP script used to define the OBCP and the OBCP code for on-board execution. No assumption is made in this Standard about whether these two forms are the same or whether a transformation (e.g. compilation, pseudo-code generation) is applied to the script to generate the code.

- OBCP validation tools e.g. a simulation environment.

The requirements for the OBCP preparation environment are contained in clause 5.3 of this Standard.

In accordance with ECSS-E-ST-10, all space system data is maintained within a repository called the **engineering data repository**. As defined in ECSS-E-ST-70-31, the monitoring and control component of the engineering data repository includes the definitions of system elements (corresponding to the hierarchical decomposition of the space system), activities (e.g. telecommands, ground procedures and OBCPs), reporting data (e.g. telemetry parameters) and events (occurrences of predefined conditions that can be used to trigger monitoring and control functions). The engineering data repository is therefore the repository for OBCPs that are developed and validated in the OBCP preparation environment and is also the repository from which they are accessed later by the execution environment. The configuration management of OBCPs is handled by the generic configuration management function of the engineering data repository and is not addressed further within this Standard.

The **OBCP execution environment** is part of the overall test and operations monitoring and control environment and includes:

- A ground part with:
  - — dedicated services for uplinking OBCPs, monitoring and controlling their execution;
  - — the means to visualize the on-board execution environment (including the status of the on-board OBCPs);
  - — constraint and consistency checking functions.
- An on-board part with:
  - — one or more OBCP engines which:
    - o provide monitoring and control services for interfacing with the ground or onboard services;
    - o provide standard services for interaction with other on-board systems (on-board software, platform subsystems and payloads);
    - o manage the execution of OBCPs.
  - — (optionally) one or more OBCP stores for the intermediate storage of OBCPs prior to loading to an OBCP engine for execution. The mechanisms for transferring the OBCP from ground to an on-board store and the management of OBCPs within on-board stores are outside the scope of this Standard.

The OBCP engine and the OBCP preparation environment are designed and developed as an integrated system and the partitioning of capabilities between them may vary from project to project.

This Standard assumes that, if there are multiple OBCP engines, they are independent of each other.

The requirements for the OBCP execution environment are contained in clause 5.4 of this Standard.

# 5 OBCP system capabilities

## 5.1 OBCP structure

a. Each OBCP shall have a unique “OBCP Identifier”, which is assigned by the ground system.

b. It shall be possible to define “arguments” for an OBCP i.e. parameters whose values are input to the OBCP at initiation time.

NOTE The definition of arguments, including data type, default value, maximum/minimum range of value, is covered by ECSS-E-ST-70-31, clause 6.7.1.2.

c. An OBCP shall consist of the following elements:

1. an optional declaration body which declares and initialises variables and declares local functions that are used internally within the OBCP;
2. an optional preconditions body which ensures that the OBCP main body is executed only if (or when) predefined initial conditions are satisfied;
3. a mandatory main body which fulfils the goal of the OBCP;
4. an optional confirmation body which verifies the conditions that determine whether the goal of the OBCP is met;
5. an optional contingency handling body that manages anomalies detected during the execution of the OBCP.

d. The nominal execution sequence for an OBCP shall be:

1. declaration body;
2. preconditions body;
3. main body;
4. confirmation body.

e. A step construct shall be provided to identify a block of functionally self-contained statements.

f. Steps shall be uniquely identified within the context of an OBCP.

g. The result generated by the confirmation body shall be used by the OBCP engine to determine the confirmation status of the OBCP.

h. In the case that there is no confirmation body, an implicit confirmation status shall be set by the OBCP engine.

NOTE For example, if all activities initiated by the OBCP were successful and no exception has been detected, then the OBCP confirmation status could be set to "success".

i. A "local function" construct shall be provided to encapsulate a self-contained sequence of statements which accepts input parameters and returns output parameters.

j. It shall be possible to initiate the execution of a local function from anywhere within an OBCP.

## 5.2 OBCP language capabilities

### 5.2.1 Introduction

The OBCP language enables the end-user to define the script of the OBCP, making reference as appropriate to system elements, activities, reporting data and events that are fully-defined within the engineering data repository. In the remainder of this clause, the capabilities of the language are organised as follows:

- data types;
- OBCP-internal declarations;
- assignments;
- expressions;
- flow controls;
- waits;
- external interactions;
- contingency handling.

### 5.2.2 General

a. The syntax of the OBCP language shall be formally specified.

NOTE For example, using the ISO extended Backus-Naur form (EBNF), see ISO/IEC 14977.

b. It shall be possible to include comments in every line of source code.

### 5.2.3 Data types

a. The OBCP language shall support the simple data types specified in clause 5.5 of ECSS-E-ST-70-31.

b. The OBCP language shall support the definition of structured data types i.e. arrays and records.

c. The OBCP language shall support explicit type-casting constructs.

NOTE This does not preclude that some implicit type-conversions are also supported.

### 5.2.4 Declarations

a. It shall be possible to declare variables of any data type, including their initialisation values.

b. It shall be possible to declare constants of any data type including their values.

c. It shall be possible to declare and define local functions.

d. A local function shall be uniquely identified within the context of an OBCP.

### 5.2.5 Assignments

a. It shall be possible to assign a value to a variable.

NOTE A variable, defined within the OBCP declaration body, is visible anywhere within the OBCP, including within a local function.

### 5.2.6 Expressions

a. Mathematical, time, string and bitwise operations and functions shall be supported.

b. It shall be possible to manipulate arrays, including operations on the elements of arrays.

c. It shall be possible to construct expressions that operate on constants, on-board parameters, activity arguments and variables.

d. It shall be possible to refer to constants together with their engineering units.

e. It shall be possible to mix compatible engineering units freely within an expression.

f. The automatic conversion between different, but compatible, engineering units shall be supported.

g. It shall be possible to refer to on-board parameters by their names as defined in the engineering data repository.

h. It shall be possible to express on-board parameter values in engineering units.

i. It shall be possible to express on-board parameter values in the raw form that is downlinked to the ground system.

j. It shall be possible to refer to events, and their associated parameters, by their names as defined in the engineering data repository.

### 5.2.7 Flow controls

a. It shall be possible to include the following execution controls within an OBCP:

1. simple conditional branching (i.e. if … then … else …);
2. multiple conditional branching where the path taken is dependent on the value of a specified parameter (or variable);
3. repeated execution of a statement (or a group of statements) with the possibility of repeating the execution a specified number of times, repeating the execution indefinitely whilst a given condition holds true or repeating the execution until a given condition becomes true.

b. It shall be possible to nest execution control constructs.

### 5.2.8 Waits

a. It shall be possible to specify the following types of wait statement within the main body of an OBCP:

1. wait until a given on-board time;
2. wait for a given interval of time to elapse;
3. wait until a given condition becomes true;
4. wait until a given event occurs;
5. wait until any event from a specified list of events occurs.

b. It shall be possible to specify a combination of conditions within a wait statement.

c. The conditions specified for an OBCP precondition or confirmation shall be any combination of the following:

1. wait until a given absolute time;
2. wait for a given interval of time to elapse;
3. wait until a given condition becomes true;
4. wait until a given event occurs;
5. wait until any event from a specified list of events occurs;
6. test whether a given condition is true.

d. It shall be possible to define timeout conditions for wait statements and the behaviour if the timeout is exceeded.

### 5.2.9 External interactions

#### 5.2.9.1 Acquiring and setting the values of on-board parameters

a. It shall be possible to acquire the value of an on-board parameter including the on-board time.

b. It shall be possible to set the value of specific on-board parameters.

NOTE The following are examples:

- Housekeeping parameters that are subsequently reported to ground.
- Parameters used to retain information between successive executions of a given OBCP.
- Parameters to be subsequently accessed by an on-board monitoring function.

#### 5.2.9.2 Initiating activities

a. It shall be possible to refer to activities, and their associated arguments, by their names as defined in the engineering data repository.

b. It shall be possible to request the execution of any on-board activity accessible from ground, unless explicitly excluded.

NOTE For example, command pulse distribution unit (CPDU) telecommands can be excluded if they can only be sent from ground.

c. It shall be possible to indicate explicitly, for each activity, which stages of execution verification to report.

d. It shall be possible to acquire the execution status of an initiated activity.

e. It shall be possible to acquire the confirmation status of an initiated activity.

NOTE The confirmation status is acquired at the completion of execution and reflects the success or failure of the activity.

f. It shall be possible to acquire the initiation time, start time, termination time or completion time of the last execution of an activity.

g. It shall be possible to request the execution of an activity and proceed immediately with the execution of the next statement of the procedure.

NOTE This implies that more than one initiated activity can be executing in parallel.

h. It shall be possible to request the execution of an activity and wait for a given execution status or confirmation status to be tested in order to determine the subsequent course of action.

i. It shall be possible to request the execution of an activity and proceed without waiting for its completion.

j. If verification of the execution of activities has been requested, the OBCP shall not complete execution until verification reporting is complete.

k. Where an OBCP waits for activity verification, it shall be possible to define a timeout.

l. Within an OBAP, it shall be possible to request the execution of specific on-board activities not accessible from ground.

NOTE For example, low-level commands such as on-board bus access.

#### 5.2.9.3 Raising and accessing events

a. It shall be possible to raise events with associated parameters.

NOTE Only an existing "system event" can be raised in this way i.e. it is not possible for an OBCP to declare and raise a "new" event.

b. It shall be possible to access events raised on-board together with their associated parameters.

### 5.2.10 Contingency handling

a. It shall be possible to define one or more contingency handling routines that are triggered by the occurrence of one or more conditions or events.

NOTE For example, resource being exceeded, parameter out-of-limits.

## 5.3 The OBCP preparation environment

### 5.3.1 OBCP script preparation

a. An editor (read and write) shall be provided for preparing OBCPs.

b. A viewer (read only) shall be provided for visualising OBCPs.

c. It shall be possible to collapse and expand the view of an OBCP, either globally or locally at the level of a step.

d. The script preparation environment shall be fully integrated with the engineering data repository, such that an OBCP can be populated with database objects that are selected from context-sensitive menus.

NOTE For example, only showing system elements, activities, reporting data and events belonging to the parent system element of the OBCP.

e. It shall be possible to override any default values contained in the engineering data repository.

NOTE For example, to specify a different activity verification profile from the default profile contained in the engineering data repository.

### 5.3.2 Syntax analysis, consistency, dependency and constraint checking

a. A syntax analyzer, consistency, dependency and constraint checker shall be provided that verifies and reports on:

1. the compliance of the OBCP script with the language grammar;
2. the consistency of the OBCP script with the current content of the engineering data repository;
3. inter-OBCP dependencies such as execution calls;
4. compliance with the constraints defined for the given OBCP.

NOTE 1 For example, constraints can include: the available OBCP engine services, on-board resources such as memory usage, maximum size of the OBCP code, CPU usage, maximum frequency of on-board parameter sampling requests or activity invocations.

NOTE 2 Different constraints may apply to all OBCPs, to OBOPs or OBAPs only or to defined sets of OBOPs or OBAPs.

NOTE 3 Different constraints may apply to different OBCP engines.

### 5.3.3 Report generation

a. It shall be possible to generate a report for an OBCP, containing at least:

1. a summary of referenced engineering data repository objects i.e. activities (including (OBCPs), parameters and events;
2. consumption of on-board CPU and memory;
3. prerequisites for execution;
4. script quality measures.

NOTE For example, nesting level, number of loops and other measures as defined in ECSS-Q-ST-80, clause 7.1.

b. It shall be possible to generate a report containing the activity call tree of all OBCPs, i.e. a tree showing all calling and called activities.

### 5.3.4 Verification and validation

a. A debugging facility should be provided that supports:

1. execution of OBCPs on a step-by-step basis;
2. execution of OBCPs on a statement-by-statement basis;
3. “step over”, “step into” and “step out of” statements;
4. forcing of a given execution path within an OBCP;

5. setting and visualisation of values for internal and external data;

6. stimulation of events;

7. simulation of the execution of an activity invoked by an OBCP, including the simulation of execution responses;

8. simulation of any interaction with the execution environment that can be experienced during the real execution of the OBCP;

   NOTE For example, control operations such as suspend and resume).

9. creation of breakpoints from which the subsequent execution of an OBCP can be resumed.

   NOTE Storing of a breakpoint includes storing of all relevant context data.

b. A flight representative verification and validation environment shall be provided to verify and validate OBCPs prior to their operational use.

   NOTE The required level of fidelity can be higher for OBAPs than for OBOPs. For example, OBAPs can require the use of on-board hardware in the loop.

c. Test coverage analysis tools shall be provided covering code, branch and decision coverage, including the identification of missing coverage in accordance with the selected OBCP engineering process (see clause 6.3).

## 5.3.5 OBCP characterisation

### 5.3.5.1 Memory predictability

a. Dynamic and static memory requirements shall be predictable at procedure preparation time.

   NOTE 1 Static memory requirements can be known at generation stage and monitored on-board, commonly split between mass memory (on-board store) and local RAM.

   NOTE 2 Dynamic memory allocation includes stack allocation and stack saving (and persistence between executions).

### 5.3.5.2 Time predictability

a. It shall be possible to predict the upper boundary of the execution duration of an OBCP.

### 5.3.5.3 Observability

a. The OBCP engine shall provide observability of OBCP-related run time events and OBCP data.

   NOTE For example, branching events.

b. It shall be possible to define levels of observability for OBCPs.

c. It shall be possible to select the level of observability for a given OBCP.

## 5.4 The OBCP execution environment

### 5.4.1 Ground capabilities

a. Prior to initiating OBCP operations, the ground system shall ensure that the following conditions are respected:

1. Compliance with the allowed state transitions for an OBCP.

   NOTE For example, an "Activate OBCP" request is only allowed if the OBCP is already loaded in the OBCP engine and not currently "Active".

2. Interdependencies between OBCPs.

   NOTE For example, when a given OBCP is to be activated, ensure that all OBCPs callable from that OBCP are already loaded or scheduled for loading before their calling time.

3. Compliance with resources separately allocated to OBAPs and OBOPs.

   NOTE For example, the maximum number of OBCPs concurrently loaded or concurrently active, CPU usage, memory usage.

b. The ground system shall uplink OBCPs to the spacecraft by telecommand in the form of OBCP code suitable for loading to the relevant OBCP engine.

c. Telecommand protocols for ground control of the management and operation of OBCPs on-board shall be based on the ECSS-E-70-41 on-board operations procedure service.

   NOTE The ECSS-E-70-41 memory management service is only used to support OBSW or OBCP anomaly investigations, and in exceptional cases when the use of other ECSS-E-70-41 services is no longer possible due to on-board failures.

### 5.4.2 OBCP monitoring and control

#### 5.4.2.1 Introduction

Figure 5-1 shows the different states through which an OBCP can pass and the transitions that are allowed between these states.

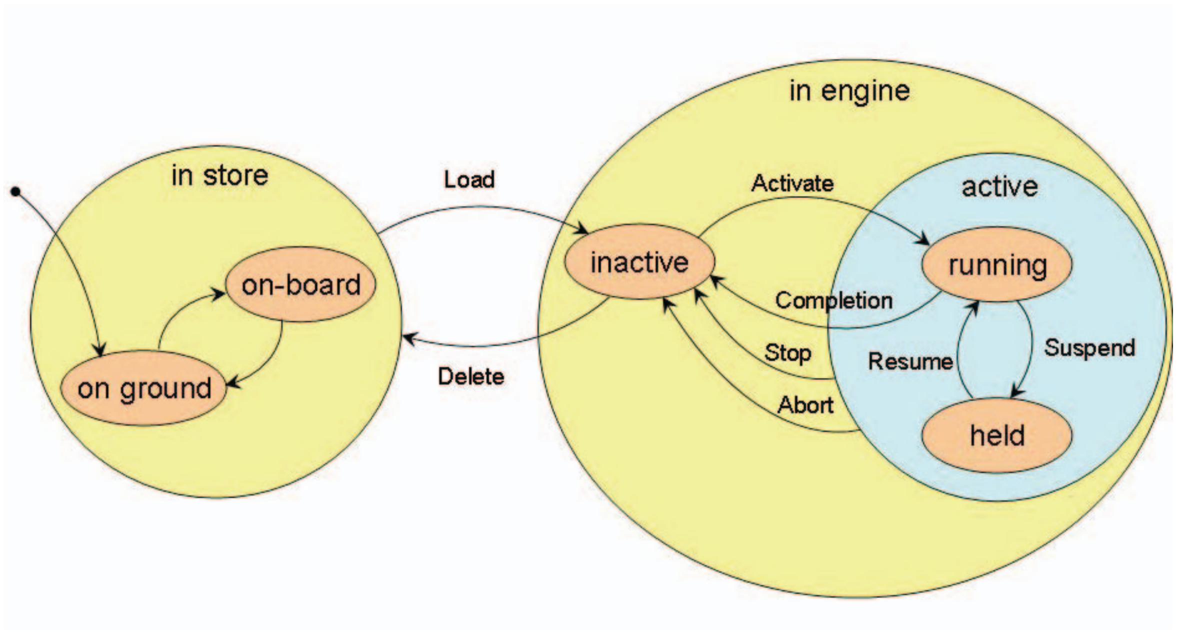

**Figure 5-1: OBCP state diagram**

NOTE Although the concept of on-board store and associated transitions are introduced in Figure 5-1, this Standard does not prescribe the implementation of on-board stores.

#### 5.4.2.2 OBCP execution states

a. The following OBCP execution states shall be supported:

1. **in store** i.e. the OBCP is in the executable form, stored either on-board or on-ground.

2. **in engine** i.e. the OBCP is in an on-board memory accessible by the OBCP engine:

   (a) **inactive** i.e. the OBCP is idle;

   (b) **active** i.e. the OBCP is under control of the OBCP engine:
   (1) **running** i.e. the OBCP is executing;
   (2) **held** i.e. the OBCP execution has been placed on hold.

#### 5.4.2.3 OBCP operations and reporting

a. The OBCP engine shall be able to perform the following operations on OBCPs, in response to requests by telecommand from the ground system or from on-board processes:

1. **Load** i.e. request the OBCP engine to load the OBCP into the OBCP engine (transition from "in store" to "in engine: inactive").

   NOTE This is ECSS-E-70-41 service 18, subtype 1.

2. **Activate** i.e. request the OBCP engine to initiate the execution of the OBCP (transition from "inactive" to "active: running").

   NOTE This is ECSS-E-70-41 service 18, subtype 3.

3. **Communicate Parameters** i.e. request the OBCP engine to communicate parameters to an active OBCP.

   NOTE This is ECSS-E-70-41 service 18, subtype 7.

4. **Suspend** i.e. request the OBCP engine to suspend the execution of a running OBCP at the end of the current statement or at the end of a specified step (transition from "active: running" to "active: held").

   NOTE This is ECSS-E-70-41 service 18, subtype 5.

5. **Resume** i.e. request the OBCP engine to resume a suspended OBCP from the point where it was suspended (transition from "active: held" to "active: running").

   NOTE This is ECSS-E-70-41 service 18, subtype 6.

6. **Stop** i.e. request the OBCP engine to stop execution of a running OBCP at the end of the current statement or at the end of a specified step, or a held OBCP (transition from "active" to "inactive").

   NOTE 1 This is ECSS-E-70-41 service 18, subtype 4.

   NOTE 2 Execution of the OBCP cannot be resumed subsequently from that point.

7. **Abort** i.e. request the OBCP engine to stop immediately the execution of an active OBCP (transition from "active" to "inactive").

   NOTE This is ECSS-E-70-41 service 18, subtype 12.

8. **Delete** i.e. request the OBCP engine to remove an inactive OBCP from the OBCP engine (transition from "inactive" to "in store").

   NOTE This is ECSS-E-70-41 service 18, subtype 2.

b. If the OBCP engine receives a request that is not consistent with the current state of an OBCP, it shall fail the request and generate a telecommand verification failure.

   NOTE For example, an Activate request for an already active OBCP.

c. At the end of its execution, the OBCP shall return autonomously to the "inactive" state (Transition: "**Completion**").

d. The Activate request shall include any necessary parameters to be passed to the OBCP.

e. Once loaded to the engine, each OBCP shall remain available for subsequent reactivation until it is deleted from the OBCP engine.

f. The execution of OBCP requests (Load, Delete, Activate, Communicate Parameters to OBCP, Stop, Suspend, Resume, Abort) shall be reported to the source of the telecommand by means of telecommand verification reports.

g. The execution of OBCP requests commanded from on-board shall be reported to the ground system by means of event reports, containing the following information:

   1. the identity of the source that requested the OBCP operation;

      NOTE For example, an OBSW application or another OBCP.

   2. the OBCP Identifier (object of the request);

   3. the execution status of the requested operation.

h. The OBCP engine shall report runtime errors detected during the execution of an OBCP by means of event reports.

### 5.4.3 OBCP integrity

a. A checksum associated with the OBCP code shall be loaded along with the OBCP, suitable for the verification of the code independently of any data security check mechanisms provided by the uplink protocol.

b. The OBCP code checksum shall be generated on the ground by the translation process which originally generates the code from the source language.

c. The OBCP code checksum shall be verified when the OBCP is loaded to the OBCP engine.

d. If the checksum verification fails, the OBCP engine shall fail the Load request and generate a telecommand verification failure.

e. The OBCP code checksum shall be verified whenever a checksum check is requested by ground.

### 5.4.4 On-board capabilities

#### 5.4.4.1 General

a. A given on-board application process shall implement at most one OBCP engine whose function is the execution and associated handling of OBCPs.

b. The following aspects of the OBSW and the OBCP engine functionality shall be defined:

   1. scheduling policy;

   2. static and dynamic memory allocation;

   3. garbage collection policy;

   4. observability and controllability of these functions by the ground system.

#### 5.4.4.2 OBCP engine services

a. The OBCP engine shall provide all the services necessary to implement the OBCP language capabilities.

NOTE There can be different services (or levels of service) applicable to OBOPs and OBAPs, e.g. OBOPs are only authorized to initiate activities accessible to ground.

b. The OBCP engine shall provide a "global function service" to OBCPs i.e. a set of standard functions that can be called from any OBCP.

NOTE Management of the global functions of an OBCP engine is implementation-dependent e.g. software patch, provision of additional ECSS-E-70-41 service 18 subtypes.

c. The behaviour of the OBCP engine in the event of a reset of, or discontinuity in, the reference clock on which the wait services of the OBCP engine depend, shall be defined.

#### 5.4.4.3 OBCP engine monitoring

a. The following housekeeping information shall be made available for each OBCP present in the OBCP engine:

1. the OBCP execution status;
2. when the OBCP is running, the identification of the step being executed;
3. when the OBCP is suspended, the identification of the next step to be executed.

NOTE Configuration of the reporting of housekeeping information is performed using ECSS-E-70-41 service 3.

b. OBCP engine housekeeping information shall be made available.

NOTE Examples are:

- Available free memory.
- CPU processing time used by the OBCP engine.
- I/O slots used.

c. All events of operational significance shall be reported to the ground system using the ECSS-E-70-41 event reporting service.

d. To support the investigation of anomalies, detailed information shall be made available on the basic functioning of the OBCP engine and loaded OBCPs.

NOTE 1 Examples are:

- The management of memory used for the accommodation of OBCPs.

- The content of the call stack at the time of the anomaly.

NOTE 2 Requirements for such reporting depend on the design and implementation of the individual OBCP engines and the OBSW.

#### 5.4.4.4 OBCP loading policy

a. The OBCP loading policy to be implemented by the OBCP engine shall be defined.

NOTE 1 A loading policy is only applicable when OBCPs can be held on-board in more than one OBCP store. The policy defines precedence rules which steer the automatic selection of the OBCP store from which the OBCP engine obtains the OBCP code when responding to a Load request.

NOTE 2 OBCP stores can be implemented in devices of different nature, such as RAM, mass memory unit or EEPROM. A typical loading policy could be: local RAM, platform mass memory (volatile or non-volatile on loss of power), EEPROM (non-volatile).

b. The following characteristics of the loading policy shall be defined:

1. whether the loading policy is reprogrammable;
2. how the loading policy is affected by reset or reboot of the OBSW, switch-over to a redundant processor or loss of power to on-board memories.

c. The loading policy shall be definable on a per-OBCP basis.

#### 5.4.4.5 Process scheduling (OBSW and OBCP engine)

a. The OBSW scheduling policy for the OBCP engine shall be defined.

b. A specified minimum allocation of processor time measured across specified regular time intervals shall be guaranteed to the OBCP engine.

NOTE There are a number of possible ways to meet these requirements, for example:

- schedule a fixed CPU time-slot for the OBCP engine at fixed time intervals (simple solution);
- schedule the OBCP engine in a prioritized scheduling scheme, with pre-emption by higher-criticality OBSW tasks;
- dedication of a co-processor to the OBCP engine.

c. The OBCP engine shall be able to execute several OBCPs in parallel.

d. The allocation of on-board resources for running OBOPs shall be independent from the allocation of on-board resources for running OBAPs.

NOTE For example, CPU time, memory.

e. The maximum number of OBOPs (active or inactive) that can be loaded within an OBCP engine at any given time shall be specified.

f. The maximum number of OBOPs active in an OBCP engine at any given time shall be specified.

g. The on-board resources allocated to a given OBOP shall be independent of the number of running OBOPs.

NOTE As a consequence, the OBOP designer is free to write an OBOP solely to serve its intended monitoring and control purpose without needing to consider the resources utilized by the OBOP.

h. The rules for the allocation and utilisation of available resources to concurrently executing OBAPs shall be defined.

NOTE This covers in particular the definition of a priority scheme for OBAP scheduling.

#### 5.4.4.6 Isolation of OBCPs

a. The OBCP engine shall ensure that internal faults and errors do not propagate to the OBSW.

NOTE This concerns faults that affect the CPU and memory resources provided by the OBCP engine to the OBCP and faults that affect the OBCP engine function itself. For example, an incorrect value written to an onboard parameter is excluded.

b. The OBCP engine shall ensure that it does not exceed its maximum allocated resources.

c. The operations of loading, activation and control of an OBCP shall not impact the operations of already active OBCPs.

d. The OBCP engine shall isolate OBCPs from each other in terms of fault and error propagation.

e. The OBCP engine shall prevent OBCPs from accessing unauthorised memory space.

#### 5.4.4.7 Exception handling

a. Any OBCP-internal error situation arising during execution shall be detected either by an OBCP-internal exception handler or by the OBCP engine.

NOTE Examples are:

- Overflow conditions and divide-by-zero when performing arithmetic operations
- Operations accessing arrays outside their declared boundaries.

b. The OBCP engine shall abort the execution of an OBCP following detection of an OBCP-internal error that is not handled by the OBCP exception handler.

c. An anomaly event for reporting to the ground system shall be generated on the occurrence of any OBCP-internal error.

d. A run-time error during execution of any OBCP exception handler shall result in immediate termination of the OBCP.

e. Where the conditions required for the execution of running OBCPs can no longer be provided by the OBCP engine, the actions to be taken shall be defined.

NOTE An example set of actions could be:

- the OBCP engine raises an event that can be caught by specific contingency handlers within running OBCPs. Such contingency handlers could establish a default state for all devices being controlled by the OBCPs.
- all running OBCPs are suspended;
- a report is issued to ground.

#### 5.4.4.8 Continuity of service

a. The concept for ensuring continuity of service shall be defined.

NOTE Providing for continuity of service can imply a need to define requirements for preservation of OBCP state information in non-volatile on-board memory.

b. It shall be possible to specify the list of OBCPs to be automatically loaded and activated within the OBCP engine at OBCP engine initialisation.

NOTE Engine initialization is defined as the actions to be performed by the OBCP engine following engine start-up and finishing when the engine is ready to accept an external command.

# 6 OBCP engineering processes

## 6.1 Introduction

The OBCP concept enables a level of decoupling between the OBCP development and spacecraft lifecycles, as illustrated in Figure 6-1:

- The OBOP lifecycle is fully decoupled from the spacecraft system and consequently, from the OBSW, e.g. OBOPs can be defined after launch without the need for system delta-qualification.
- The OBAP lifecycle is partly decoupled from the OBSW but still tightly linked to the system development process, e.g. OBAPs are qualified as part of the system.

The engineering process required for any OBCP is determined by the level of isolation provided by the OBCP system i.e.:

- The extent to which the capabilities of the system are protected against propagation of functional failures induced by the OBCP.
- The extent to which the capabilities of the system are protected against propagation of software failures induced by the OBCP.

Finally, the development process is also influenced by quality considerations and development risk assessment. Special attention should be paid to "critical" OBCPs (OBAPs or OBOPs) in terms of margins and potential impact on the spacecraft and the mission.

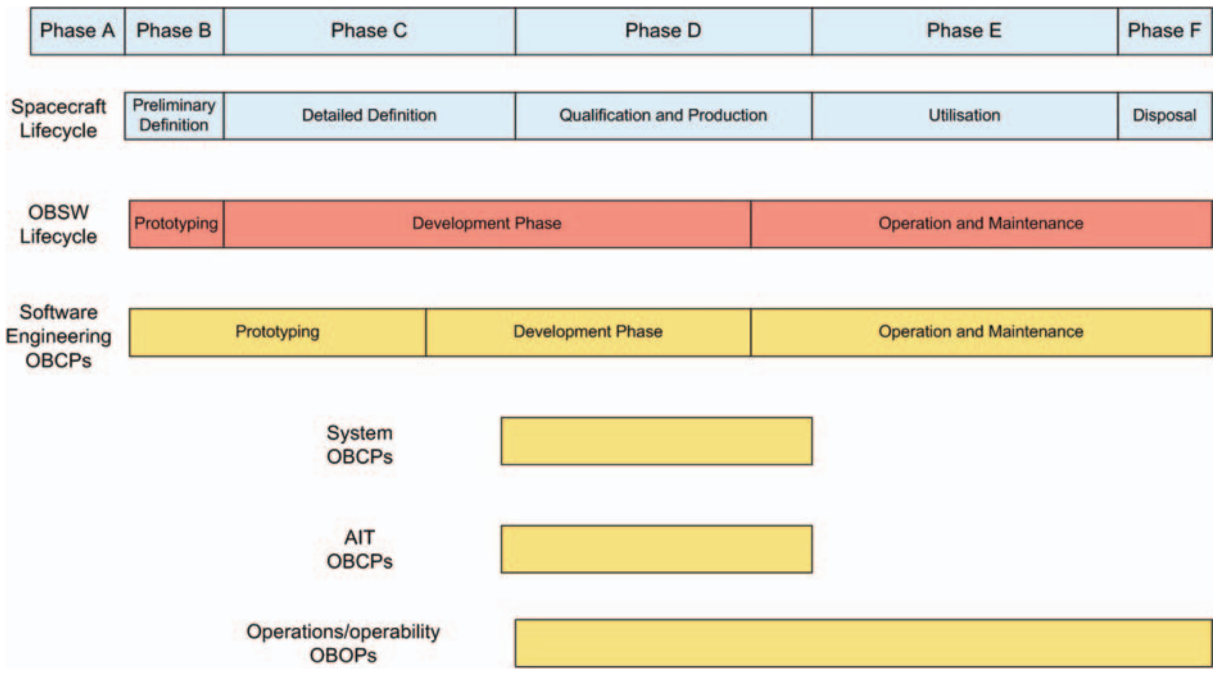

**Figure 6-1: Lifecycles of OBCPs originating from the different domains**

## 6.2 Overall management process of the OBCP system

### 6.2.1 Management process

a. The OBCP system management process (see Figure 6-2) shall be defined and documented.

b. The OBCP system management process shall include the following activities:

  1. collection of needs from the various stakeholders (see clause 4.2);

  2. trade-off analysis between implementing the needs as OBSW or as OBAP (see clause 6.2.2);

  3. analysis of the use case inputs for future OBOP needs (see clause 6.2.3) and expected benefits in terms of engineering effort (see clause 6.2.4) and assignment of requirements to the on-board system and to the ground system.

c. The output of the OBCP system management process shall be the consolidation of the required capabilities of the OBCP engine, the preparation environment, the ground execution environment and engineering processes.

d. All the requirements shall be traceable either to the OBCP engine, the preparation environment, the ground execution environment or engineering processes.

e. The output of the OBCP system management process shall be reviewed during the system requirements review (SRR), see Figure 6-3.

NOTE The list of criteria and the weights of individual criteria which drive the trade-off analysis for OBCPs (see clauses 6.2.2 and 6.2.3) may vary according to the project phase. Before the SRR, flexibility is much higher than during system development or during the in-orbit life.

f. The capabilities of the OBCP engine and the preparation environment shall be defined and frozen at the software PDR, see Figure 6-3.

NOTE The trade-off between implementation of one single OBCP engine or several OBCP engines is performed as part of the OBSW engineering process.

g. The resource allocation, monitoring and consolidation of budgets shall be managed at system level and reviewed during system and OBSW reviews.

NOTE Examples of budgets are: number of OBCPs executing in parallel, number of activities invoked by an OBCP.

h. OBAP validation and verification shall be completed prior to system qualification, see Figure 6-3.

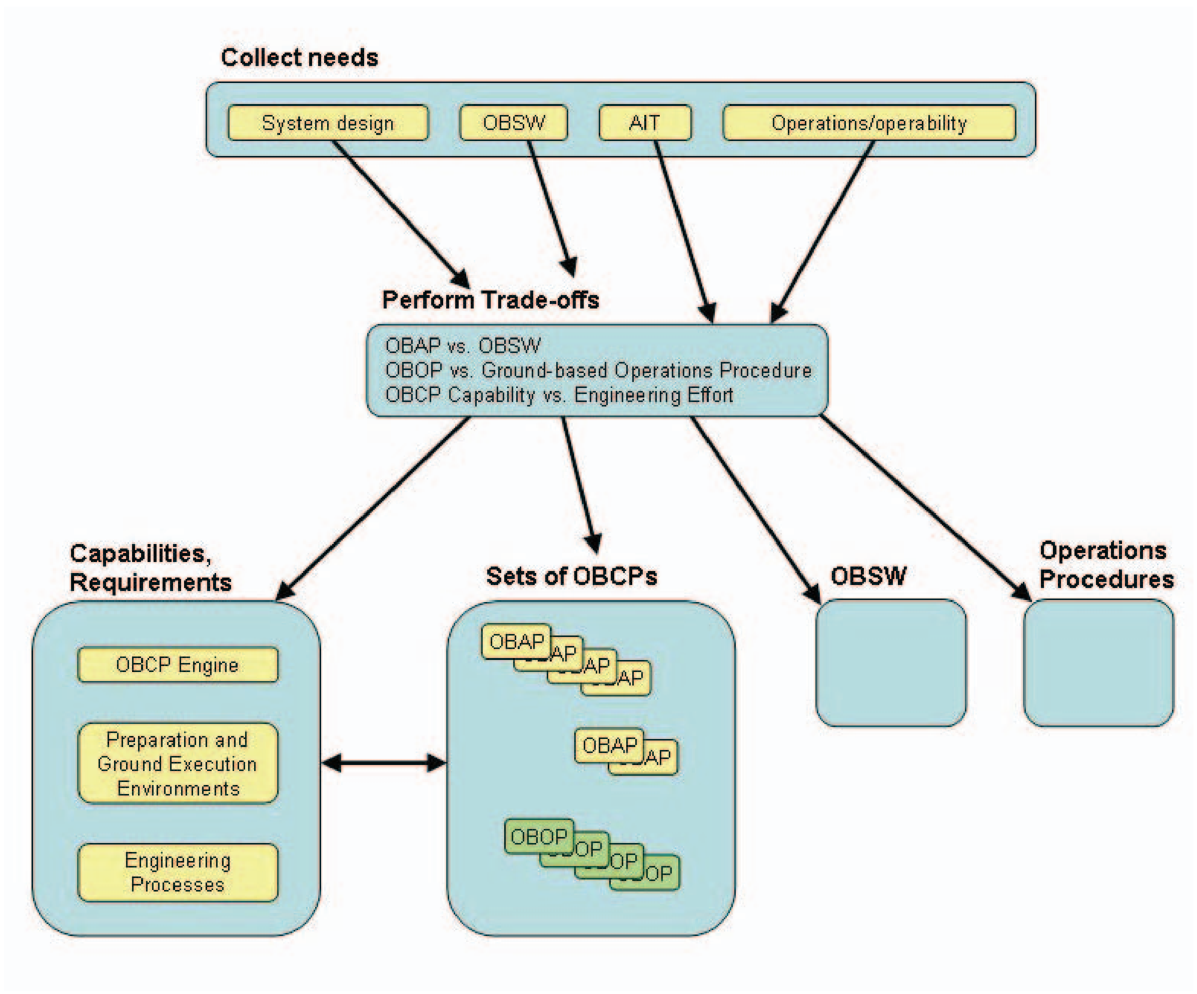

Figure 6-2: OBCP management overview

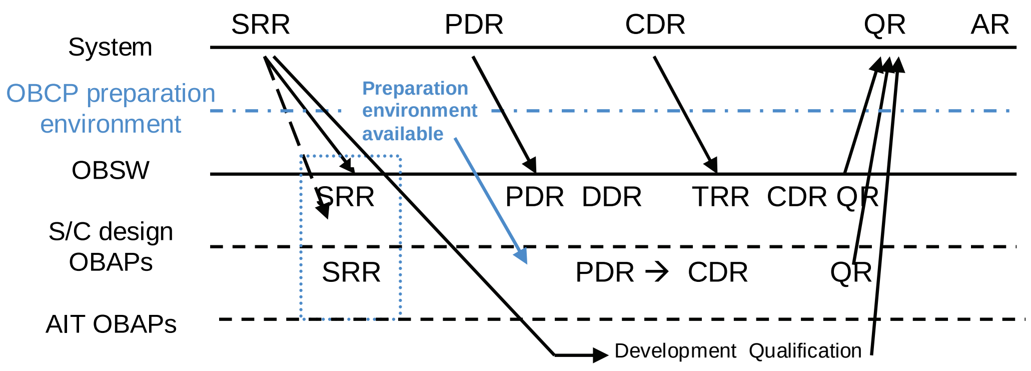
SRR
PDR
CDR
QR
AR
System
OBCP preparation environment
Preparation environment available
OBSW
SRR
PDR DDR
TRR CDR QR
S/C design OBAPs
SRR
PDR → CDR
QR
AIT OBAPs
Development Qualification

Figure 6-3: Synchronisation of OBAP lifecycles with system and OBSW lifecycles

### 6.2.2 OBAP vs. OBSW: criteria and trade-off analysis

a. The OBAP vs. OBSW trade-off shall be conducted according to a predefined set of criteria.

b. The justification for the trade-off decisions shall be documented.

NOTE The following criteria are typically used in making the trade-off between OBSW and OBAP:

- Major capabilities and advantages afforded by the use of OBAPs:
- Variability and flexibility, for example, due to changing mission requirements or system ageing.
- Programmatic, e.g. late definition, reuse. However late selection of OBAPs to develop OBSW functions has the disadvantage of concentrating the validation effort into a short and critical period.
- Major capabilities and advantages afforded by the use of OBSW:
- Real-time capabilities.
- Execution time (but this depends on the technology).
- Suitability for implementation of complex functions (i.e. suitability of the software engineering process and techniques).
- Suitability for implementing the core system management functions that have stable requirements and a close relationship with subsystem engineering (e.g. AOCS).
- Suitability for implementing those functions required to sustain survival modes of the spacecraft in view of the need to be fully (extensively) validated by the end of the development phase.
- Suitability for implementing the generic part of a system or of a family of systems (software product lines, reuse of components such as those supporting ECSS-E-70-41 standard services).
- Reduction of schedule risks by requiring that on-board functionality is defined early in the spacecraft development programme.

### 6.2.3 OBOP vs. ground-based operations

a. When analysing the use cases for potential OBOP development, a trade-off between implementing the identified use cases on-board or by means of ground-based operations procedures shall be performed.

NOTE The decision process can be supported by the following criteria:

- Major capabilities and advantages afforded by the OBOP concept:
- Enable spacecraft operations during phases of non-visibility and/or with long signal propagation delays.
- Improve security in the operations in case of loss of ground control.
- Reduce operator errors.
- Synchronise operations with asynchronous elements (e.g. on-board events).
- "Coded and up-linked once, used many times".
- Combine "atomic" operations within a single operation (e.g. define critical activities to be performed in one block).
- Decrease the need for round-the-clock human availability during the routine phase.
- Major capabilities and advantages afforded by using ground-based procedures (potentially supported by automated ground tools):
- Human response to unforeseen scenarios.
- Decrease the capabilities required from the on-board system.
- Decrease validation of the required on-board capabilities.
- Engineering effort to develop and validate a ground-based procedure is less than required for an OBOP.
- Less effort to update a procedure.
- Less complex configuration management system.
- Increased overall visibility of operations.

### 6.2.4 Trade-off between OBCP engine capability and engineering effort

a. When analysing the required OBCP engine capabilities (robustness w.r.t. failure propagation), a trade-off between implementing the required capability on-board or not, thereby increasing the engineering effort (validation), shall be performed.

NOTE 1 For OBAPs, isolation may be relaxed in order to simplify the OBCP engine. This is acceptable because of the increased validation effort.

NOTE 2 For OBOPs, a higher level of isolation is mandatory.

### 6.2.5 Overall organization and management

a. All the stakeholders shall be identified together with appropriate organisation and responsibilities and associated interactions between various teams (see clause 4.2).

b. Each category of stakeholder (see clause 4.2) shall be responsible for the overall development and validation of their own set of OBCPs.

NOTE For OBAPs, close collaboration between all interested parties is needed throughout the development lifecycle.

c. OBOP execution decisions shall be approved by one single authority in order to ensure compatibility with system constraints and consistency between OBOPs.

NOTE 1 Examples are:

- Check that the maximum number of running OBOPs is not violated.
- Check that no two OBOPs are accessing concurrently the same hardware resource.

NOTE 2 This does not preclude splitting OBOPs into different sets, with their development assigned to different teams, working to different schedules.

d. Policies and procedures for the allocation of resources between stakeholders shall be established.

NOTE 1 For OBOPs this could be implemented by assignment of a maximum number of OBOPs per stakeholder category (including margin).

NOTE 2 For OBAPs this could be implemented by assignment of CPU and memory.

e. A process shall be defined to ensure the enforcement of the resource allocation policies throughout the project lifetime.

## 6.3 OBCP engineering

a. The engineering process for OBOPs and for OBAPs shall be defined and documented.

b. For OBAPs, the usual software lifecycle process with requirements, design and coding phases shall be applied in accordance with ECSS-E-ST-40.

c. When tailoring ECSS-E-ST-40,, the following shall be ensured:

1. An adequate level of documentation and reviews.
2. Strict configuration control of the OBAPs.
3. An adequate level of validation effort and formalism.

d. For OBOPs, the usual operations procedure engineering process shall be applied in accordance with ECSS-E-ST-70.

e. When tailoring ECSS-E-ST-70, the following shall be ensured:

1. Timely documented feedback from operation engineers to the OBOP developers.
2. Concurrent maintenance of OBOP documentation.
3. Strict configuration control of the OBOPs both by developers and by test engineers.

# Bibliography

| | |
|---|---|
| ECSS-S-ST-00 | ECSS system – Description, implementation and general requirements |
| ECSS-E-ST-10 | Space engineering – System engineering general requirements |
| ECSS-Q-ST-80 | Space product assurance – Software product assurance |
| ISO/IEC 14977:1996 | Information Technology - Syntactic Metalanguage - Extended BNF - First Edition |