



CHEOPS ground segment: Systems and automation for mission and science operations[☆]

Alexis Heitzmann ^a,^{*} María J. González Bonilla ^b, Anja Bekkelien ^a, Babatunde Akinsanmi ^a, Mathias O.W. Beck ^c, Nicolas Billot ^a, Christopher Broeg ^{d,e}, Adrien Deline ^a, David Ehrenreich ^{a,f}, Andrea Fortier ^{d,e}, Marcus G.F. Kirsch ^g, Monika Lendl ^a, Nuria Alfaro Llorente ^b, Naiara Fernández de Bobadilla Vallano ^b, María Fuentes Tabas ^b, Anthony G. Maldonado ^b, Eva M. Vega Carrasco ^b, David Modrego Contreras ^h

^a Department of Astronomy, University of Geneva, Chemin Pegasi 51, Versoix, 1290, Switzerland

^b INTA - Instituto Nacional de Técnica Aeroespacial, CEIT - Centro Espacial INTA Torrejón, Carretera de Ajalvir km. 4, Torrejón de Ardoz, 28864, Spain

^c ETH Zurich, Department of Physics, Wolfgang-Pauli-Strasse 2, Zurich, CH-8093, Switzerland

^d Center for Space and Habitability, University of Bern, Gesellschaftsstrasse 6, Bern, 3012, Switzerland

^e Weltraumforschung und Planetologie, Physikalisches Institut, University of Bern, Gesellschaftsstrasse 6, Bern, 3012, Switzerland

^f Centre Vie dans l'Univers, Faculté des sciences, Université de Genève, Quai Ernest-Ansermet 30, Genève 4, 1211, Switzerland

^g ESA - European Space Agency, European Space Operations Centre - ESOC, Robert-Bosch-Str. 5, Darmstadt, 64293, Hesse, Germany

^h ISDEFE - Ingeniería de Sistemas para la Defensa de España (External Consultant at INTA), Calle de Beatriz de Bobadilla 3, Madrid, 28040, Spain

ARTICLE INFO

Keywords:

CHEOPS

Mission Operations

Science Operations

Software architecture

ABSTRACT

The CHAracterising ExOPlanet Satellite (CHEOPS) is the first European Space Agency (ESA) small-class mission. It has been performing photometric astronomical observations with a particular emphasis on exoplanetary science for the past five years. A distinctive feature of CHEOPS is that the responsibility for all operational aspects of the mission lies with the CHEOPS consortium rather than ESA. As a result, all subsystems, their architecture, and operational processes have been independently developed and tailored specifically to CHEOPS. This paper offers an overview of the CHEOPS operational subsystems, the design, and the automation framework that compose the two main components of the CHEOPS ground segment: the Mission Operations Center (MOC) and the Science Operations Center (SOC). This comprehensive description of the CHEOPS workflow aims to serve as a reference and potential source of inspiration for future small and/or independent space missions.

1. Introduction

The CHAracterising ExOPlanet Satellite (CHEOPS, Benz et al. 2021) is the first small-class (S-class) mission of the ESA Cosmic Vision 2015–2025 programme. CHEOPS was born from a partnership between ESA and the CHEOPS Consortium led by the University of Bern, Switzerland. The mission's main scientific goal is the characterization of exoplanets using the transit method (Winn, 2010). Mission requirements were driven by the scientific needs to reach a photometric precision of 20 ppm (parts per million) in 6 h of integration time for bright stars (corresponding to the transit of an Earth-size planet around a G5 dwarf star brighter than 9th magnitude in the V band), and 85 ppm in 3 h integrations for stars of 12th magnitude in the V band (corresponding to a Neptune-size planet transiting a K-type

star). Fortier et al. (2024) provides an in-depth analysis of CHEOPS performance throughout the first 3.5 years of operations (2019–2023).

CHEOPS was launched on December 18th, 2019 from Kourou, French Guyana on Soyuz flight VS23. The satellite platform is an AS-250 from Airbus Defence & Space and the payload, developed under the responsibility of the University of Bern, is a Ritchey-Chrétien telescope with a primary mirror of 30 cm effective diameter. Incoming light is fed to its single instrument, which is a high-performance photometer with a frame-transfer back-side illuminated charge-coupled device (CCD) in its focal plane. CHEOPS is nadir-locked in a Sun-synchronous Low-Earth orbit at an altitude of 690 km, and a local time of the ascending node of 6 a.m. This means that the orbital period of CHEOPS is 98.7 min, closely following the day/night terminator, maximizing the

[☆] This article is part of a Special issue entitled: 'Observatories Software' published in Astronomy and Computing.

^{*} Corresponding author.

E-mail addresses: alexis.heizmann@unige.ch (A. Heitzmann), glezbnj@inta.es (M.J. González Bonilla), cheops-pso-members@listes.unige.ch (A. Deline).

Sun exposition of the solar panels and minimizing Earth stray-light contamination. As a result of the orbit design and the **Ground Station (G/S)** used to communicate with the satellite, the spacecraft is visible by the **G/S** approximately 4–6 times a day, 2–3 times per morning and evening.

CHEOPS' ground segment is classically divided into two main entities. The **MOC**, located in the **Centro Espacial INTA Torrejón (CEIT)**, at the **Instituto Nacional de Técnica Aeroespacial (INTA)**, Spain, handles telecommunication with the spacecraft, the satellite status monitoring, the orbit determination and control, and the attitude monitoring. The **SOC**, hosted by the astronomy department of the **Université de Genève**, Switzerland, is the bridge between the **Project Science Office (PSO)** and the **MOC**. The **SOC** is responsible for handling observer requests, planning the sequence of scientific activities to be executed on board, as well as the data processing, distribution and archiving. Unlike other ESA missions, all aspects of mission operations for the **CHEOPS** mission fall under the responsibility of the Consortium. **CHEOPS'** operational subsystems have been developed either from scratch following **European Cooperation for Space Standardization (ECSS)** standards, or through the reuse, customization, or extension of existing systems. The development of the mission was led by the University of Bern, with important contributions from members of the **CHEOPS** Consortium located in Austria, Belgium, France, Germany, Hungary, Italy, Portugal, Spain, Sweden, and the United Kingdom. The consortium contribution matched the **ESA** contribution, for a total of 105 million €, covering design, fabrication, launch and operations for the nominal mission. **CHEOPS'** nominal mission concluded in 2023. The consortium was granted a first extension, running until the end of 2026. A second extension is still under consideration and, if approved in 2026, would extend the mission through 2029.

This paper presents an overview of the architecture of the **CHEOPS** ground segment through the mission operations cycle, describing the key subsystems and software involved in each stage and their interactions, with an emphasis on the key role automation plays in ensuring the efficient and reliable operation of the satellite. The success of the mission, achieved through a tailored development approach outside the standard ESA framework, could serve as an inspiration for future mission development and operations.

Section 2 outlines the generation of the required inputs for mission planning: the observing time requests (**Observation Request (OR)**) from the scientific community, and the orbit prediction as well as the satellite contacts with the **G/S** from **MOC**. Then, Section 3 details the conversion of these inputs by the **SOC** into a sequence of activities, known as the **Activity Plan (AP)**, and how the **MOC** transforms the **AP** into **Telecommand (TC)**s that will be executed onboard **CHEOPS**. Section 4 covers the operations at **MOC** during **G/S** contacts, when the **TC**s are uplinked to the satellite and both the housekeeping and science data are downlinked, ingested in the **MOC** (housekeeping data only), and forwarded to **SOC**. This raw data are then processed, their quality assessed, and sent to the archive and the mirror archive, as outlined in Section 5. Finally, Section 6 presents some additional tools and software developed and used to complement the main **CHEOPS** operations. Fig. 1 provides a high-level overview of the operational sites, main operational procedures, and information flow.

2. Inputs generation for mission planning

The **CHEOPS** mission is driven by **OR**s submitted by the Science Team, Guest Observers, or the Instrument Team. These **OR**s must be translated into an **AP** that includes an optimized sequence of activities to be performed by the spacecraft. In addition to these **OR**s from the scientific community, the predicted orbit and the planned **G/S** contacts are needed for the **AP** generation. The creation of these inputs, which requires the participation and coordination of both **MOC** and **SOC**, is detailed in this section.

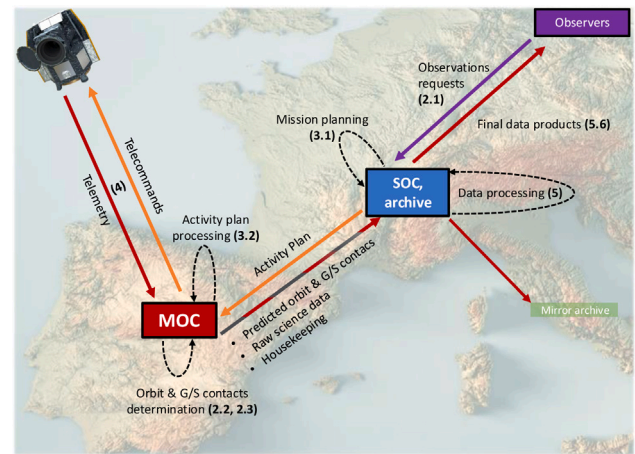


Fig. 1. Overview of the **CHEOPS** operational chain. The two main ground segment components, **MOC** and **SOC**, are represented by coloured boxes. The purple arrow represents the **OR**s from the scientific community. The orange arrows describe the products generated from the weekly planning operations from the reception of the observations requests to the uplink of the **AP TC**s. The red arrows trace the data flow, from raw, to stored processed data accessible to the community. The grey arrow (hashed with red) are for the **MOC** generated files and used at **SOC** for the planning (orbit files and pass time determinations). Finally, the internal operational procedures are displayed as dashed black arrows. The steps outlined in this paper are marked with a bold number corresponding to their respective section. All following Figures in the paper describe sub-systems of the 2 main blocs: **MOC** and **SOC**, following the colour coding proposed here.

Fig. 2 aims to guide the reader through the mission operations flow (excluding any science or housekeeping data aspects) described in Sections 2 and 3, from the generation of the inputs required during the mission planning to create the **AP**, to the execution of the activities onboard the satellite.

2.1. Observing requests from users

The interface between the users (scientists) and the mission operations team is the **Proposal Handling Tool phase 2 (PHT2)**,¹ a web interface accessible to users who have been allocated observing time on **CHEOPS**. The software system is built with a MariaDB² database server as the back-end, an Apache Web server running on CentOS with OpenSSL³ and PHP,⁴ and a front-end developed using WordPress.⁵ A screenshot of the web interface is shown on the bottom of Fig. 9. The PHP application connects to the MariaDB database through the MySQL Native Driver (mysqlnd, developed and maintained by the PHP project), which acts as the database client library.

New observation requests can be submitted using the User Interface or by submitting an XML file, containing all relevant information about an observation (details are outside the scope of this paper, but the interested reader is referred to the **Proposal Handling Tool phase 2 (PHT2)** guidelines⁶). All relative information for the users, targets, programmes and **OR**s are stored in the **PHT2** database. The **PHT2** also handles all status modifications from users or following observations (feedback from the **Mission Planning System (MPS)**, subject of Section 3.1). Every Monday, as part of the first step of the weekly planning phase, the

¹ <https://cheops.unige.ch/pht2/>

² <https://mariadb.com/>

³ <https://openssl-library.org/>

⁴ <https://www.php.net/>

⁵ <https://wordpress.org/>

⁶ [PHT2guidelines](#)

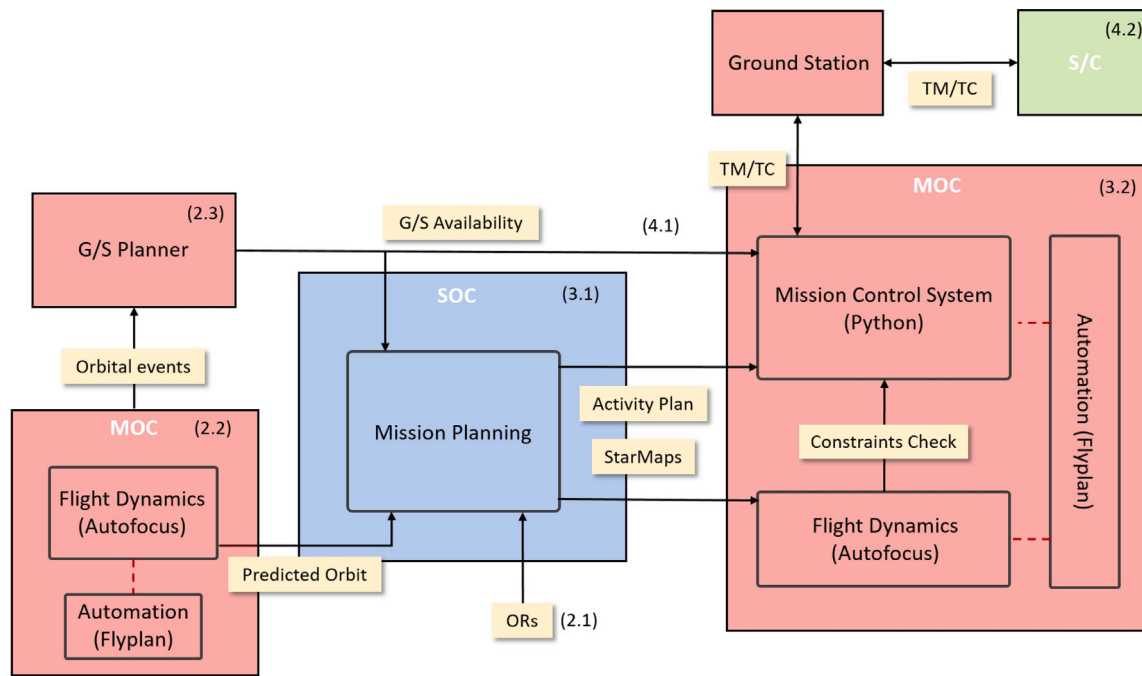


Fig. 2. Overview of the CHEOPS planning cycle. Red boxes refer to the MOC, blue boxes to the SOC and green box to CHEOPS spacecraft. Black arrows indicate the flow of information, with the main inputs and outputs transferred between each component shown as yellow boxes. The numbers in parentheses indicate the subsection describing the sub-systems of interest. This figure summarizes the weekly planning cycle described extensively in Sections 2 and 3.

SOC reviews all new, modified or stopped ORs. This review process validates all new ORs to make sure these are technically sound (e.g., target location or instrument settings) and that the observing strategy is properly implemented. XML files are generated for each new/modified OR and serve as input for the MPS.

2.2. Predicted orbit and orbital events

The CHEOPS predicted orbit file is the next input required to generate the AP at SOC. At this stage, the visibilities between CHEOPS and the G/S are also computed. This information is needed by the G/S scheduling office to generate another key input for the mission planning generation: the G/S contacts assigned to CHEOPS. Fig. 3 aims to illustrate the input generation process within the MOC, as described in this section and in Section 2.3.

The main subsystems involved in the generation of the predicted orbit file and the G/S visibilities are the Flight Dynamics System (FDS) and the Automation system, both located at MOC, and are described in detail below.

The FDS is responsible for computing the satellite's orbit, derived products, orbital manoeuvres, attitude and pointing. The CHEOPS FDS is built on the Focus⁷ software suite, a collection of flight dynamics modules developed by GMV.⁸ The software is written in Fortran with a Java-based Human-machine interface (HMI). Each module reads input files in ASCII format that can be read and edited in the HMI.

The various Focus modules within FDS can be executed either manually, in isolation by users, or automatically in sequences. In the automated mode, the outputs of one module serve as inputs for subsequent modules, and the configuration values must be coordinated. This functionality is handled by an additional software component, Autofocus. Autofocus enables the automated execution of FDS workflows that involve the sequential operation of multiple modules. The workflows are encoded in the Spacecraft Operations Language (native

to Autofocus), which facilitates configuration of the modules, initialization of their execution and retrieval of output information. This capability enhances operational efficiency and ensures consistency in the execution of complex flight dynamics processes.

The Automation system is based on the Flyplan software (provided by GMV). This tool enables automatic planning and execution of MOC routine activities, reducing operator workload, minimizing the need for their presence outside regular working hours, and preventing human errors. Flyplan offers a comprehensive Gantt-style visualization that provides clear insight into the schedule and reflects the execution status of all tasks. Its customizability allows the operator to create scheduling routines in Jython, the Python interpreter designed to run on the Java platform, that can schedule the execution of automated activities at predefined times. Among these activities are the FDS Autofocus scripts.

The following paragraphs detail the generation of the products needed for mission planning using the aforementioned tools: the predicted orbit and the G/S visibilities. The generation of these products is based on the operational orbit, which contains the orbit from the entire mission. The operational orbit is updated after each orbit determination.

The key Focus Software (SW) modules involved in the orbit determination are detailed below:

PREPRO – It preprocesses tracking data files received from G/S, which contain Doppler measurements recorded during CHEOPS passes. PREPRO converts two-way Doppler measurements (frequency shifts between the signal transmitted to and received from CHEOPS, in Hz) into radial velocity (km/s) and stores the observations in the NAPEOS (ESA's navigation tool for Earth orbiting satellites) Tracking Data Format to be used for orbit determination in the BAHN module.

BAHN – BAHN performs orbit determination using the Doppler measurements pre-processed with PREPRO. BAHN provides an estimation of the satellite's position and velocity in the J2000 reference frame at a user-defined reference epoch via a batch least-squares filter. If convergence is achieved, BAHN generates an orbit file including the determined period and a propagation six weeks in the future. This orbit is used to reconstruct the operational orbit.

⁷ <https://www.gmv.com/es-es/productos/espacio/focussuite>

⁸ <https://gmw.com/>

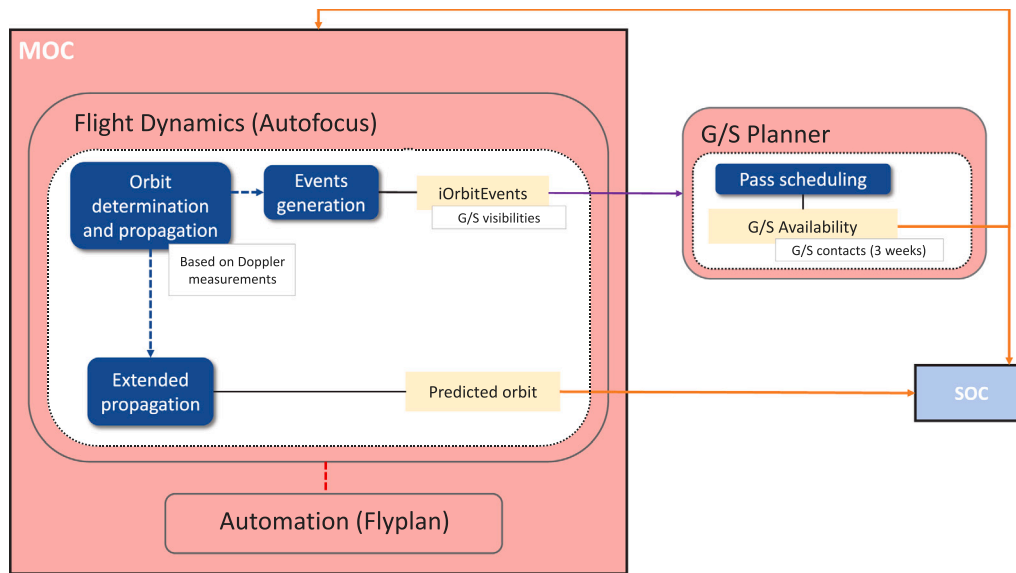


Fig. 3. Input generation at **MOC** for mission planning at **SOC**. Red boxes represent the **MOC** and include the various subsystems involved in the product generation process. Within these, dark blue boxes indicate the specific steps performed, and the yellow boxes the resulting output products. Data flow is colour-coded: purple for internal transfers, and orange for transfers to **SOC** (represented by the light blue box). This figure illustrates the generation of the predicted orbit and **G/S** contacts at **MOC**, which are required for the **AP** creation at **SOC**. The red dashed line indicates that this process is triggered by the automation system.

The sequential execution of these **SW** modules is automated in an Autofocus sequence scheduled in Flyplan twice a day after the morning passes and the evening passes.

Once the operational orbit has been updated with the latest orbit determination, it serves as input for the following **SW** modules intended to generate the derived products:

EVENTS – This component generates the **G/S** events with the visibilities between **CHEOPS** and the **G/S** available for the mission. The resulting event file includes potential passes for the subsequent 5 weeks.

PROPAG – Propagation of the orbit is done by means of a numerical integration method that calculates successive state vectors over a user-specified propagation interval. **PROPAG** makes use of solar and geomagnetic activity, Earth orientation parameters and perturbation models (geopotential, atmospheric, tropospheric and ionospheric effects). From an initial state vector included in the operational orbit, it propagates the orbit 4 months in the future.

The last step to generate the final **FDS** products consists of launching the **PRODGEN** module in **Focus**:

PRODGEN – From the outputs generated by the **FDS** modules previously described, **PRODGEN** generates the final products required by other subsystems, including:

- **iOrbitEvents**: derived from the **EVENTS** file, this product contains visibility information needed at the **CEIT G/S Planner** for contact scheduling at the **INTA G/S** in coordination with other supported missions.
- **Predicted Orbit**: generated from the **PROPAG** orbit, it is sent to **SOC** and serves as input for the **AP** creation and occasional long-term planning.

The sequential execution of the **SW** modules required to generate the derived products is automated through an Autofocus sequence, scheduled in Flyplan once a week at a predefined time.

2.3. Planned **G/S** contacts

The information of future **G/S** contacts planned for **CHEOPS** is the last input required for the **AP** generation at **SOC**.

The **CEIT G/S Planner SW** tool is used for **CHEOPS** pass scheduling. This tool is written in C# and fully developed by **INTA**. It understands the pass requests in different formats, graphically shows the current **G/S** schedule and the requests, detects and shows any conflict, modifies both the current schedule and the requests, and saves and propagates any solution. The tool not only receives pass requests from **CHEOPS**, but also from other missions and generates a conflict-free solution. It also configures the control system of each **G/S** with the passes of all the missions including the start and end times of the support, as well as configurations or the times of mode changes.

The **G/S Planner SW** tool receives the **iOrbitEvents** file with the visibilities between **CHEOPS** and the **G/S** used for the mission (see Section 2.2), and produces the **GSAvailability**, an XML file containing the planned **G/S** passes for **CHEOPS** for the following three weeks, specifically including the assigned **G/S**, start time and duration.

The **GSAvailability** file is sent to **SOC** and serves as an input so the pass related activities can be included in the planning. This file is also processed internally at **MOC** as described in Section 4.1.

3. Mission planning generation and processing

Once the input files have been created and available at **SOC**, an optimized target observation plan is generated, which is then converted into the **AP** and forwarded to **MOC** (Section 3.1). At **MOC**, the Activity Plan is translated into telecommands for subsequent uplink to the satellite (Section 3.2).

3.1. Activity Plan creation

The main step of mission planning is to create an observation plan for **CHEOPS** and translating it into a sequence of procedures, the **AP**, to be executed by the spacecraft. This process is carried out weekly at the **SOC** using the **MPS**, the software responsible for generating the observation schedule for the upcoming cycle, which typically spans one week beginning on Saturday.

The **MPS** software was specifically created for **CHEOPS** and is maintained by Deimos Engenharia, S.A.⁹ The **MPS** reads all input (e.g.,

⁹ <https://deimos-space.com/>

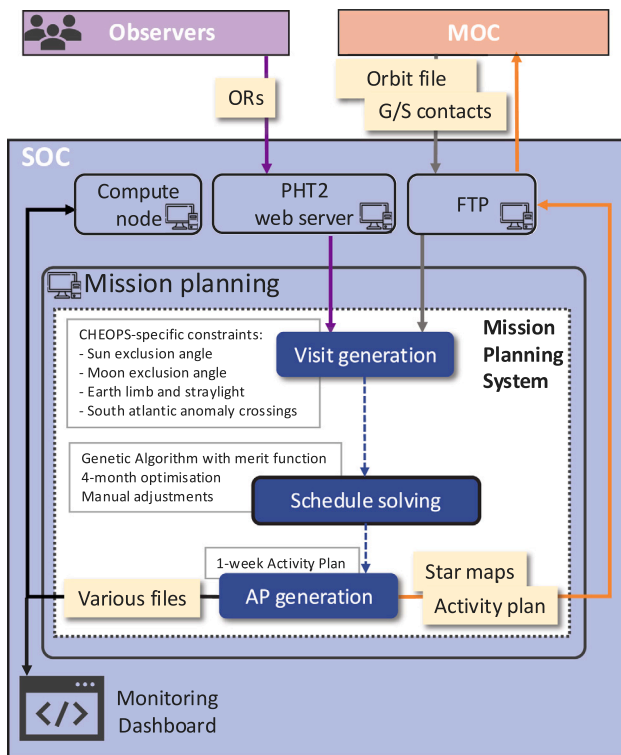


Fig. 4. Detailed view of SOC's mission planning. The large blue box represent the SOC entity. Each sub box with a computer icon show the virtual machines that play a role in the mission planning. For a general view of all the SOC hardware architecture, see Fig. 11. Dark blue boxes show the steps of the mission planning with added description in the grey outlined rectangles. External entities are shown in purple and red respectively for the scientific community (observers) and the MOC. Data flow is colour coded, with (i) purple for the ORs from the scientists, (ii) grey for files provided by MOC and used for the mission planning, (iii) orange for files generated by the mission planning and (iv) black for other files internally generated and exchanged between machine.

predicted orbit files, planned G/S contacts, configuration files, etc.) and output (AP) files from an intray and an outray folders. The exchange of files between MOC and SOC is made via an FTP server and all files transferred through this FTP server are managed by the file dispatcher (described in Section 5.2). The SOC operator creates the planning through the User Interface of the MPS.

The source code of the MPS is written in Java and follows a client-server architecture. The client is based on a Deimos Engenharia, S.A. extended version of the Jaret Timebars library,¹⁰ combined with Eclipse Rich Client Platform,¹¹ and comprises a science functions submodule, that includes, e.g. CHEOPS orbit propagations, that uses ESA's Earth Observation CFI software.¹² Communications between the client and the server use RESTful Web Services. The server has three submodules; the web services for the Application Programming Interface (API) endpoints,¹³ a persistence module handling communication with the MPS database,¹⁴ and the Input/Output module,¹⁵ handling both XML and FITS (Pence et al., 2010) files. All the necessary data required by the MPS are stored in the MPS database, which uses PostgreSQL.¹⁶

As stated previously, the main role of the MPS is to generate the AP. The process, broadly outlined on Fig. 4, begins with the MPS generating all possible observations in a given time frame, accounting for all orbit related constraints. The goal is then to create an optimal sequence of observations, maximizing the time-critical AP fill factor. This means optimizing a merit function, composed of: OR completion rate (currently unused), the ratio of Guaranteed Time to Guest Observers number of observations, the scientific priority of ORs, and the filling factor (minimizing CHEOPS' idle time). The optimization is performed by a genetic algorithm called the Schedule Solver. This piece of software was originally developed by Deimos Engenharia, S.A. and later improved by the SOC and is written in C. It is run outside of the MPS software and takes as input a text file containing all the possible visits generated by the MPS. The Schedule Solver starts by creating multiple sequences of randomly chosen visits (i.e. observations). During each iteration, the algorithm selects, combines, and mutates (applies random changes to) these visits sequences. These alterations are governed by parameters such as crossover and mutation probability. The cycle repeats for a predefined number of iterations, ultimately selecting the AP with the highest overall ranking. The output from the schedule solver is a TXT file listing the optimal solution (a suite of observations) that can be ingested back in the MPS. After human vetting of the solution by SOC and a validation by the PSO of the observation plan (and of any proposed changes), the MPS converts the sequence of observations to the AP. The AP is an XML file listing all activities for CHEOPS to execute. These include slews between targets, target acquisitions, inhibition of data taking when crossing the South Atlantic Anomaly (region of high density of cosmic ray hits degrading the data quality), etc. These activities also specify the configuration of the instrument. Before sending the AP to the MOC, additional files are generated with custom Python utility scripts. Among these are binary files called 'StarMaps', used by the instrument to locate the correct star when pointing to the target location, and developed by the University of Vienna. All these files are pushed to the file dispatcher (described in Section 5.2) which takes care of sending the files to MOC through a FTP server.

3.2. Activity Plan processing

Upon reception of the AP and the StarMaps at MOC, they are processed and translated into TCs for uplink to the satellite during a scheduled pass. This processing task involves the coordination of multiple subsystems and software tools within the MOC, including the Automation system, FDS (Section 2.2) and Mission Control System (MCS), each of them playing a specific role in the workflow. Fig. 5 presents these systems and the interface between them. For a general overview of the MOC hardware architecture, the reader is referred to Appendix A.1.

The complete processing is orchestrated by the Automation system, requiring no operator intervention. For this, the scheduling routine `monitorCHEOPS` in Flyplan is essential: while some operations are triggered at predefined times (time-driven), this script enables the automation system to react to the availability of data (data-driven) whenever needed. By continuously monitoring a configurable list of pathnames for new inputs, it allows FDS and MCS tasks to be triggered dynamically as data become available, supporting flexible and autonomous operations.

CHEOPS MCS is based on ESA's SCOS-2000 (version 5.4.2). SCOS-2000 (Peccia, 2003) is a spacecraft control system framework developed using an object-oriented analysis and design approach. Written in C++, it provides the basic processing expected of a spacecraft control system, which includes but is not limited to Telemetry (TM) processing and telecommanding.

SCOS-2000 allows missions to configure the system for their requirements using system configuration options such as environment variables and the database component, the Mission Information Base

¹⁰ <https://jaret.de/timebars/>

¹¹ <https://www.eclipse.org/>

¹² <https://eop-cfi.esa.int/>

¹³ Build with ApacheCXF

¹⁴ Build with HibernateORM

¹⁵ Build with the Java Nom-tam-fits library

¹⁶ <https://www.postgresql.org/>

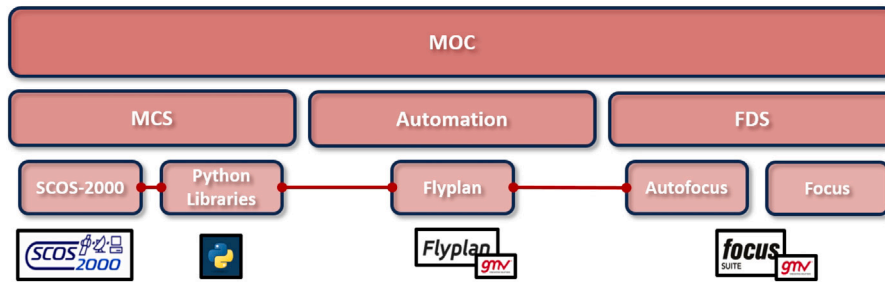


Fig. 5. Overview of the main **MOC** subsystems, their associated **SW**, and the connections between them. The automation system can trigger Python scripts that interact with SCOS-2000 and Autofocus procedures.

(**MIB**). The latter involves defining mission-specific information, such as detailing every **TC** and **TM** packet.

Of particular relevance is the **CommandBuilder**, an **MCS** tool that converts certain input files with activities into **TCs** expressed in an **MCS** legible format. The result is stored in an **SSF** file, an **ASCII** file that can be read by any standard **SCOS-2000** using the **MIB**.

The **MCS** automation capability is provided through a library of Python modules. This scripting layer grants access to all **MCS** tasks, enabling interaction with **SCOS-2000** and the execution of routine operations, such as processing the **AP**.

At this stage, the monitorCHEOPS component triggers the **AP** processing in the different subsystems: first, an **FDS** sequence is launched in Autofocus to ensure the **AP** complies with operational constraints, as detailed in Section 3.2.1; second, a Python script in the **MCS** is initialized to convert the **AP** and the StarMaps into **TCs**; and, finally, one last Python script is triggered to assign the **TCs** generated to the next available pass for its automatic uplink to the satellite. The last 2 steps are the focus of Section 3.2.2. Fig. 6 provides a guide to this sequence of operations.

3.2.1. Operational constraints check

The Autofocus sequence begins with the simulation of the satellite’s attitude and the operational constraints compliance check. The following modules in the **FDS** support these tasks:

ATTSIM - It simulates the spacecraft attitude based on the activities included in the **AP** and generates a predicted attitude file required for the subsequent module.

CONCHECK - Due to the physical and operational constraints placed on the satellite, the line of sight of the telescope must never be less than a defined Sun Exclusion Angle (initially set to 120 degrees and relaxed to 117 as of February 2025 to allow greater sky coverage) with respect to the Sun vector. This value is configurable by the user in the **CONCHECK** module, which is in charge of reading the **AP** and checking the compliance of the constraint for every slew (transition between two **CHEOPS** pointings) detailed in the **AP**. A constraint checker report indicating a Pass status is generated and disseminated to the **MCS** exclusively upon successful completion of this verification process.

3.2.2. TC conversion and pass assignment

At this point, the **AP** and the StarMaps are processed by the **MCS**. A Python script is triggered by Flyplan to verify the status of the constraint checker report and then invoke the **CommandBuilder**, where each activity is mapped to one or several **TCs** using configuration rules. Moreover, the **CommandBuilder** estimates the **AP** uplink duration and generates a Conversion Report file containing the conversion result which is then provided to **SOC**.

The last activity consists of an additional Python script in charge of assigning the **AP** **SSF** file to a pass for its uplink. Using the estimated uplink duration as input, this script selects the next pass in the current **GSA** availability file with sufficient duration for the uplink and assigns the converted **SSF** file to it.

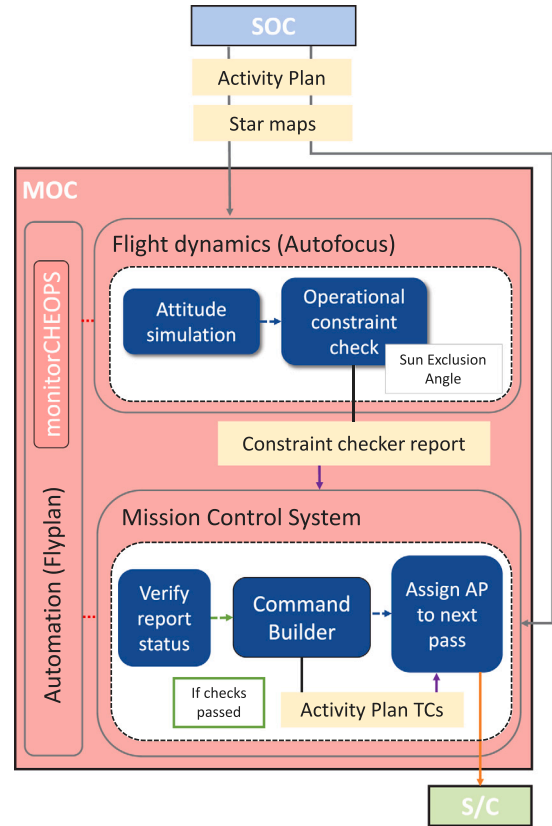


Fig. 6. Illustration of the **AP** processing sequence. Files are detected by the monitorCHEOPS component of the automation system, which triggers the **AP** processing. A sanity check of the **AP** is performed by the **FDS** and if successful, the sequence of activities in the **AP** are converted to **TCs** by the **MCS** before being uplinked to the spacecraft.

4. CHEOPS passes: TC uplink and TM downlink

As previously mentioned, there are four to six contacts between **CHEOPS** and the **G/S** per day. Before the pass, the **TCs** that will be automatically uplinked during the pass must be made available in the corresponding pass folder. Then, during the pass, the **TCs** are uplinked to the satellite (including the **TCs** from the **AP**), the real-time **TM** (housekeeping data) is received by the **G/S** and directly transferred to the **MCS** so the operators can monitor the satellite high-level status in real time if desired, and the recorded **TM** is received in the **G/S** and recorded in a binary file. Finally, after the pass, the **TM** file from the **G/S** is processed in the **MCS** to generate separate **TM** files, so the science and housekeeping data files can be transferred to **SOC**.

The **TM** received from the satellite is converted into **TM** packets and stored in the **MCS Packet Archive (PARC)**. The **PARC** is used

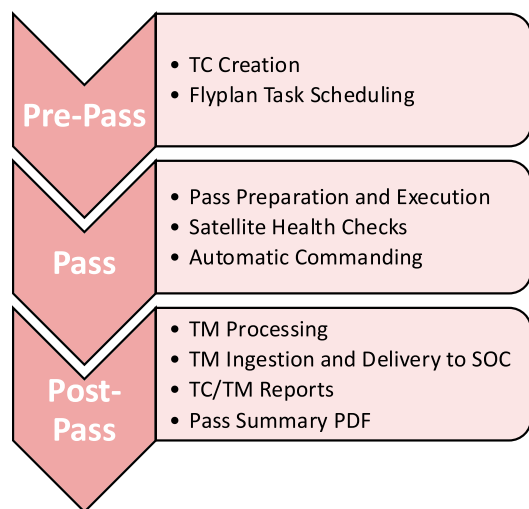


Fig. 7. Pass-related activities overview.

for archiving **TM**, **TC**, and event data. It stores the data in a MySQL database, allowing users to retrieve data.

Several MOC subsystems and software tools are used for the pass related activities: the **MCS** along with the Python scripts layer and the Automation system.

The following sections describe the main pass related activities (shown in Fig. 7) and the subsystems involved in more detail.

4.1. Pre-pass activities

Once the **GSAvailability** file is generated with the **CEIT G/S Planner** tool (Section 2.3), it is detected by the **monitorCHEOPS** activity in **Flyplan** and processed. The **GSAvailability** file is processed through the **SCOS-2000 CommandBuilder** to create the routine **TCs** associated to the pass. For every pass found in the file, **TCs** stacks are generated to begin and end **MCS-G/S** communication, and to start and stop the downlink of recorded **TM** in the satellite. A pass folder with a specific identifier is also created for each pass, and the **TCs** that have already been generated are assigned to the corresponding pass folder.

Once all **TCs** are created and assigned to their pass folder, the **Task-Injector** program is responsible for scheduling pass-related activities in **Flyplan** with a defined configuration and are scheduled with respect to the pass start time. These pass and post-pass activities are described in detail hereafter.

4.2. Pass activities

Four main activities are performed during the pass using **Flyplan**-managed Python scripts that interact with **SCOS-2000**, which are part of the **MCS** automation (Section 3.2.2).

4.2.1. Pass preparation and execution

This **CHEOPS** activity starts a few minutes before the pass, and is responsible for establishing the **MCS-G/S** connection, uplinking the **TCs** assigned to the pass, and downlinking the **TM**.

First, the Python script monitors and restarts different **SCOS-2000** tasks to ensure **MCS** tools needed for the pass are up and running. Furthermore, it identifies the current pass folder, which contains the **TC** stacks to be uplinked during the pass.

The **Auto Stack** is an **MCS** application that allows the automation system to load and send pregenerated **TCs**. For **CHEOPS**, two **Auto Stack** instances have been configured and reserved for automated operations, enabling simultaneous tasks to be performed by the Automation system.

At the beginning of the pass, the Python script loads and sends the **TCs** found in the current pass folder to open the **MCS-G/S** connection from the **Auto Stack**.

The **SMON** is an **MCS** process that allows **TM** parameter provision from the **PARC**. The Python script uses this process to retrieve the engineering value of specific **TM** from the **PARC** in different data formats, along with the time-stamp if desired. It is possible to retrieve the most recent value, any value in the past, or wait for the next sample to be received.

Thus, once the **MCS** starts to receive real-time **TM** from the satellite, the Python script waits for and retrieves the on-board time **TM** value from the **PARC**. This **TM** is used to calculate the accuracy of the Time Correlation coefficients.

Modern spacecraft often carry a GPS receiver that provides an accurate synchronization between the spacecraft time and the ground time (usually synchronized to **Coordinated Universal Time (UTC)**) needed for time-tagging both **TMs** and **TCs**. Nevertheless, **CHEOPS'** time is based on a local oscillator inside the On Board Computer. Therefore, an additional **SCOS-2000** feature was configured for **CHEOPS** to accurately reconstitute the **On-Board Time (OBT)** following the **ECSS** time management system. This feature supports a linear approximation for a free-running clock, like the one used in **CHEOPS**, updating the transformation coefficients every time the monitored accuracy is lower than required.

The Python scripts also have the capability to get or modify **SCOS-2000 MISC** variable values. This feature allows the automation system to verify the ground commanding capability is available before sending the **TCs** loaded in the **Auto Stacks**, change the transmission mode, and check if the **Auto Stack** has completed sending the **TCs** so the following can be loaded.

The **TCs** loaded in the **Auto Stacks** are first sent to the **G/S** and then to **CHEOPS**. The **AD** transmission mode is used for the communication, since it allows the detection of missing **TCs** and their retransmission during the communication. Several steps of acknowledgement are enabled in order to trace both the reception and transmission of the **TCs** by the **G/S**, and the reception and execution (if it is a **TC** to be executed immediately) of the **TCs** on board. Using this transmission mode, **SCOS-2000** verifies the correct and complete uplink of the **TCs** (see **Consultative Committee for Space Data Systems (CCSDS)**, 2010).

Once the ground commanding capability is available, the remaining **TCs** in the current pass folder are loaded one by one in the **Auto Stacks** and sent to **CHEOPS**. The **AP** (in case it is assigned in the current pass folder, see Section 3.2.2) and the **TCs** to start and end the downlink of the recorded **TM** are sent at this stage.

At the end of the pass, the **TCs** to close the **MCS-G/S** connection are loaded in the **Auto Stack** and sent.

4.2.2. Satellite status check and satellite systems health-checks

These two activities are initiated at the beginning of the pass.

The satellite status check enables the detection of anomalies upon the initial reception of real-time **TM**. As soon as **TM** is received, this Python script automatically notifies the operators via e-mail, providing a summary of the **Spacecraft (S/C)** high-level status and operational modes. This functionality facilitates the prompt and straightforward identification of any unexpected conditions on board. Furthermore, in cases where no **TM** is received, the system alerts the operators, thereby supporting the detection of potential pass-related issues, particularly when operators are not present during the passes.

The satellite systems health-check is intended to check the high-level status of different platform subsystems based on **Flight Operational Procedures** provided by the manufacturer. These **Flight Operational Procedures** are written in Python and invoked as external commands.

These Python scripts use the capability to wait for and read **TM** values from **SCOS-2000** (see Section 4.2.1 for **TM** parameter provision).

4.2.3. Automatic commanding

This activity is also executed at the beginning of the pass, and offers two additional features that make use of the **SCOS-2000 External Interface**. The first is the real-time injection and transmission of **TCs** to **CHEOPS** in response to specific **TM** values, and the second is the detection of defined **TM** packets received on ground so they can be retrieved from the **PARC** and decoded.

Among its uses is the recovery of **TM** from previous passes in which gaps were detected. Given that only two **TM** downlinks can be executed simultaneously, the automatic commanding is responsible for checking that the previous **TM** downlink has ended before commanding the next one.

4.3. Post-pass activities

The pass processing activity is responsible for processing the **TM** recorded on board between consecutive **G/S** contacts that is downlinked during the pass. It enables the provision of science and house-keeping **TM** to **SOC**, and the necessary **TM** to monitor the satellite status to **MOC**. Furthermore, it allows **MOC** operators to verify the execution status of the different automatic activities performed during the pass.

This activity is orchestrated by Flyplan a few minutes after the pass, and consists of a series of Python scripts that perform the activities outlined below. These scripts are part of the **MCS** automation (Section 3.2.2).

First, the start time and the **G/S** data for the current and previous pass are searched in the current **GSA** availability file. This enables the generation of **TM** reports covering the period from the beginning of the previous pass to the beginning of the current pass.

Secondly, the binary **TM** file is fetched from the **G/S** via FTP connection to be processed in the **MCS** and transferred to **SOC**:

- The **SCOS-2000 TM Processor** application is used to preprocess the **G/S** files in order to generate separate **TM** files, one per Virtual Channel (real-time housekeeping data, recorded housekeeping data, and science data).
- The **TM** files with housekeeping and science data are transferred to **SOC**.
- The **SCOS-2000 TM Replayer** application is used to ingest the recorded housekeeping **TM** into the **PARC**.
- The **TM** files are checked to identify **TM** gaps by detecting a jump in the Virtual Channel frame counter. If any gap is discovered, the gap information is included in a pending **TM** gaps file.

Later, two additional products are generated and sent to **SOC**:

- The **TCReport** file to inform the **MPS** of the execution status of the **TCs** associated to the activities programmed by the **MPS**. This file is generated using the **SCOS-2000 RGTgenerator** application, that processes the Conversion Report file generated by the **CommandBuilder** during the **AP** processing, and retrieves the execution status of the **TCs** from the **PARC**. The result is an XML file in Earth Explorer format readable by the **MPS**.
- The **OBT-UTC** report file, which is an XML file that includes the Time Correlation coefficients for the conversion of the **TM** timestamp from **UTC** to **OBT** by **SOC**.

Additionally, some products are created to facilitate the monitoring of the satellite status by **MOC** operators, and to feed the **FDS** subsystem:

- **TC** history, on-board events, and out-of-limits reports are created using **SCOS-2000** executable programs that print data shown in the **MCS** displays over a specified time period in an ASCII file.
- **TM** reports in ASCII or XML format. The **SCOS-2000 HKTMreporter** program, developed for **CHEOPS**, generates these reports using **PARC TM** packets. This provides the **FDS** with the **TM** required for some computations,

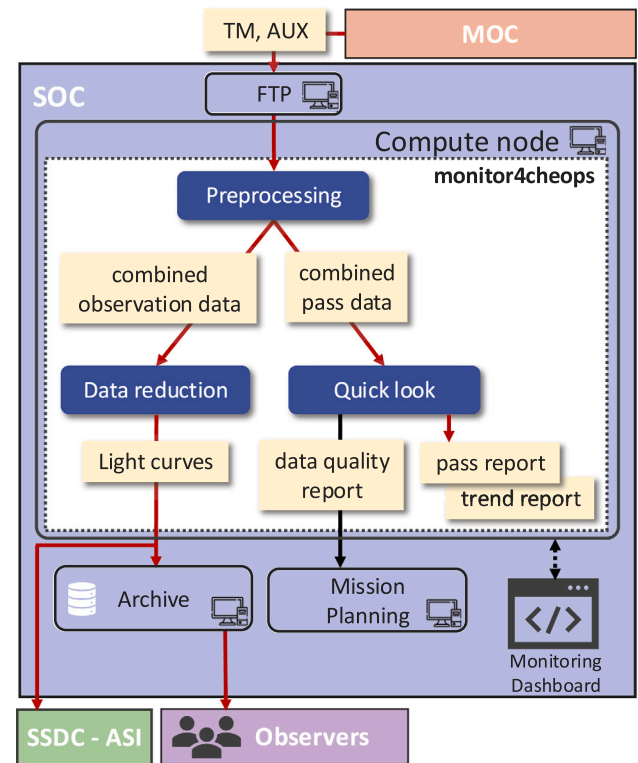


Fig. 8. Detailed view of **SOC**'s data reduction, subject of Section 5. The large blue box represent the **SOC** entity. Each sub box with a computer icon show the virtual machines that play a role in the data reduction. For a general view of all the **SOC** hardware architecture, the reader is referred to **Fig. 11**. Dark blue boxes show the high-level steps of the data reduction. External entities are shown in purple, red and green respectively for the scientific community (observers), the **MOC** and the mirror archive, hosted at the **SSDC-ISA**. Data flow is colour coded, with red for all data related products (raw and processed, science and housekeeping data) and black for other files internally generated and exchanged between machines.

for instance, the comparison of the real and simulated attitude in order to detect any deviations. Moreover, it enables the generation of trend plots for specific **TM** defined by the platform manufacturer.

Finally, a PDF detailing the results of the different pass-related activities is generated using the Python library **PyFPDF**. The different Python scripts read the logs and reports generated during the pass and post-pass activities. A colour-coding system is used to display the result of the different activities, along with a few warning messages in case of failure. This PDF, as well as the most relevant logs and reports from the pass, are forwarded to the **MOC** operators using the Python **smtp**lib and email libraries.

5. Data processing

When the housekeeping **TM** and science data arrive at the **SOC**, the files are automatically detected by the system and ingested into the **SOC** data processing pipeline, which is shown in **Fig. 8** and described in more detail in the following sections. The pipeline converts all data to FITS format and reduces the science data to photometric light curves. It also produces data quality reports for the **SOC** operator per-downlink. The end data products are ingested into the **CHEOPS** Mission Archive (see Section 5.6) that is hosted by the **SOC** and, for backup purposes and redundancy, replicated in a mirror archive hosted by the **SSDC-ISA**.

The data products are distributed to the end users via the archive's web interface, the Archive Browser.¹⁷

Importantly, the entirety of the **SOC** processing pipeline (i.e., the content of Fig. 8) is, except for the ingestion of the quality reports in the **MPS**, fully automated and does not require any human intervention.

5.1. Design decisions

Due to the very restricted development time and cost imposed by the small-class mission type, the **SOC** was designed to be as simple and light-weight as possible. To achieve this, the **SOC** relied heavily on having collaborators with prior experience in designing and implementing **SOC** software for space missions.

To keep physical hardware to a minimum, the system was fully virtualized. This allowed it to run on existing department-level hardware infrastructure that is shared with other projects while benefitting from infrastructure maintenance and upgrades in a transparent manner. **SOC** would also reuse existing project management and software development tools maintained by the department, including a centralized Subversion version control system and a Jenkins server¹⁸ for automatic regression testing.

Due to the aforementioned restrictions, the priority was first placed on implementing the functionalities of the data processing and archival system while in a second phase focusing on streamlining the user interfaces and graphical representations of the system. The system would also reuse established and proven software libraries developed for other missions when this was possible.

Lastly, the data processing and data archival system was designed to run in a fully automated and data-driven fashion in order to minimize the operational workload.

5.2. Internal file transfer

The **SOC** system consists of multiple hosts, described in Appendix A.2. The file transfer between the hosts is ensured by a Python script, the **file_dispatcher**. The script uses a configuration file that defines the source and destination hosts and directories, the file types to be transferred to a given destination as well as optional pre- and post-transfer procedures that are generally used to back up transferred files or to delete unused files. The **file_dispatcher** is executed at regular intervals by a cron job to continuously monitor directories for new files and trigger file transfers.

5.3. Core libraries

common_sw provides interfaces, algorithms and other resources that are common to all software modules in the pipeline. Most notably, **common_sw** provides the FITS data model which defines the structure of the data products that are produced by the pipeline and distributed to the end users.¹⁹ The definitions are in XML format and specify the structure of the header and data unit of all FITS extensions. C++ classes are generated from the XML definitions and contain accessor methods for modifying the content of the header and data unit, as well as functionality for reading and writing to disk, implemented using the **National Aeronautics and Space Administration (NASA) CFITSIO** library (Pence, 1999). Furthermore, a Python interface to the C++ classes is generated using Boost Python,²⁰ an open source library for exposing C++ classes and functions to Python and vice versa. These classes ensure that all data products created by the pipeline adhere to the definitions across all subsystems. **common_sw** also provides infrastructure for auto-generating documentation, both the data product

documentation for the end users, which uses the XSLT language to transform the XML definitions to HTML, as well as code documentation for the developers using the documentation generator Doxygen.²¹

science_tm is a C++ library that provides low-level routines for compressing and decompressing the data of the science **TM** received from **MOC** and is used by the Preprocessing chain when converting **TM** to FITS images.

xml_schema is a collection of XML schemas that define file interfaces between **MOC** and **SOC**, as well as **SOC**-internal interfaces. The pipeline software uses CodeSynthesis XSD,²² an XML data binding compiler, to generate C++ bindings for the schemas that are used at run-time to read XML files.

5.4. Data processing pipeline

Monitor4cheops is the component responsible for orchestrating the execution of the data processing pipeline, implemented in C++. It runs as a background service and automatically triggers the processing when data files from **MOC** are ingested into the system. Monitor4cheops has a generic plug-in architecture where the individual steps in the pipeline, which are implemented as stand-alone executables, are plugged into the system via configuration files. Thus, monitor4cheops is separate from the processing pipeline itself, making for a flexible and scalable system. The plug-ins adhere to a common interface; as input, they receive an XML file that defines the values of the input parameters and the location of the input files, and as part of their output, they are required to create a special *trigger* file that is used by monitor4cheops to chain the processing steps.

Monitor4cheops maintains an internal file repository of all ingested files, including the output files from the processing steps. File metadata is stored in RAM, and also duplicated to XML files on disk, and is used for selecting input files for the processing steps. The metadata includes the observation ID, pass ID, and file version. The processing steps themselves are defined in configuration files that specify their order in the pipeline, their input file types, and input parameters.

The processing steps are logically grouped into the Preprocessing, Quick Look and Data Reduction processing chains, which are described in more detail in the following sections and shown in Fig. 8.

5.4.1. Preprocessing chain

Preprocessing is the first chain to be executed and it converts the raw **TM**, in the form of Consultative Committee for Space Data Systems²³ packets, as well as auxiliary data in XML format, to FITS data products. Preprocessing consists of two components:

The **Conversion** component decodes and reformats the **TM** according to the spacecraft's **MIB** and performs consistency checks to detect and report missing packets. The data is time-tagged using **OBT** to **UTC** correlation. Conversion calls the **science_tm** library to decompress science images and finally converts the packets to FITS format.

The **Combination** component is responsible for grouping FITS data by observation, which is done in two steps. First, by grouping the converted pass data and then, once all data of a given observation has been downlinked, which may take multiple passes (possibly days), by grouping the per-pass data. The former is input to the Quick Look chain, while the latter is input to the Data Reduction chain.

Preprocessing is implemented in C++ and uses the **NASA SPICE** library (Acton, 1996) for attitude computation.

¹⁷ https://cheops.unige.ch/archive_browser

¹⁸ <https://www.jenkins.io/>

¹⁹ CHEOPS Data Products Definition Document

²⁰ Building Hybrid Systems with Boost.Python

²¹ Doxygen

²² CodeSynthesis XSD

²³ CCSDS space packet protocol

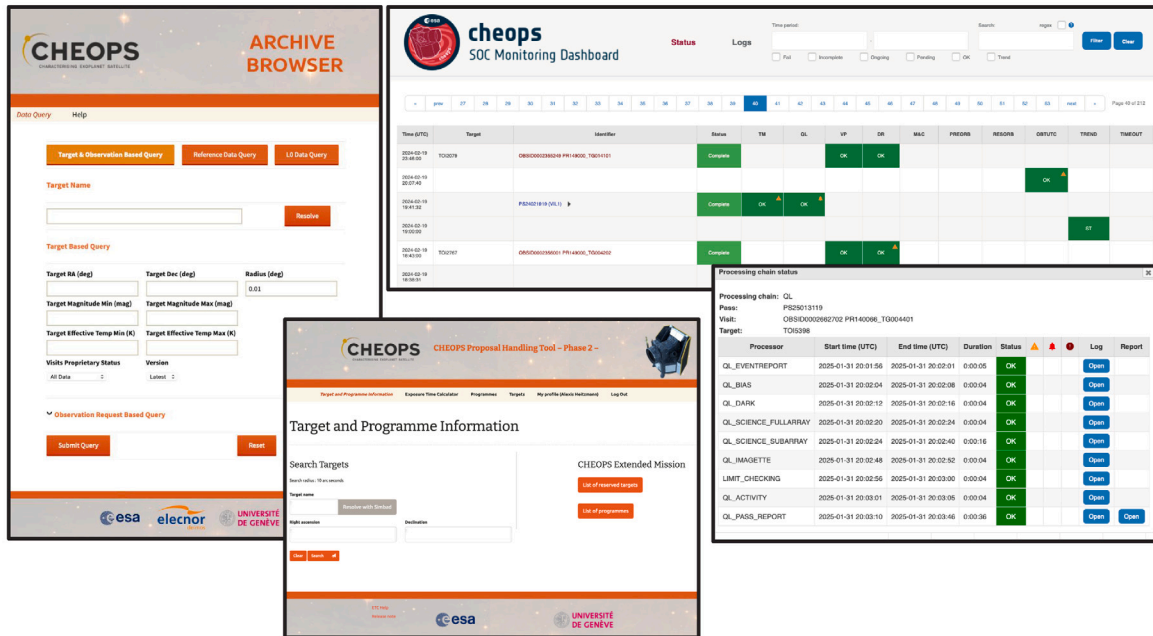


Fig. 9. Mosaic of some of the SOC tools. **Top left:** Archive browser (Section 5.6), allowing users to retrieve their data. **Top right:** Monitoring dashboard (Section 5.5), tracking the progress and logs of all the data processing pipeline. This is used daily by SOC to assess data quality, catch errors in data processing/reduction or identify missing data. The snippet on the right is a detailed view of all the ‘processors’ in a specific module of the pipeline, here the Quick Look (Section 5.4.2). **Bottom centre:** Screenshot of the PHT2 tool (Section 2.1), where all the users ORs are entered and checked before being sent to the Mission Planning System.

5.4.2. Quick look chain

The Quick Look chain, developed by the University of Cambridge, produces data quality reports that are used by the SOC operator and Instrument Team to monitor the status of observations and the instrument. It derives quality control parameters from images, image metadata, housekeeping, and auxiliary data and produces two report types: (i) pass reports giving an immediate overview of the quality of the observational data in the pass, including technical quality, spacecraft pointing, source detection, and visualization of selected science images, and (ii) trend reports generated at regular intervals for monitoring the long-term evolution of housekeeping parameters. The trend report generation is a two-step procedure. The housekeeping parameters received since the last report are first aggregated by binning and averaging the values and writing the aggregated points to disk. Then, the report is generated by selecting aggregated points for a given time interval.

In addition, Quick Look produces technical quality parameters of observations that are fed back to the MPS to assess the efficiency of the observation.

Quick Look was designed to be flexible with regard to the selection of reported parameters to accommodate changes needed as a result of the evolution of instrument characteristics over the lifetime of the mission. The parameters configuration was therefore externalized to reference files.

5.4.3. Data reduction chain

The Data Reduction is the last chain in the pipeline (see Fig. 8), responsible for creating the photometric light curves. The Data Reduction is extensively described in Hoyer et al. (2020). It is implemented entirely in Python and consists of the following modules:

Calibration, which transforms the raw images output by Preprocessing into photo-electrons calibrated images.

Correction, which corrects the calibrated images for environmental effects including smearing, bad pixels, and background and stray light pollution.

Photometry, which performs aperture photometry on the corrected images and creates the final light curves.

5.5. Monitoring dashboard

The Monitoring Dashboard is a web application that displays a graphical view of the processing pipeline and provides access to reports and log files created by the processing steps. It is the primary tool of the SOC operator for monitoring data processing and troubleshooting anomalies. The main view is an interactive timeline that lists the passes and observations and shows the status of the various processing chains. Clicking on a processing chain opens a window with detailed information about the execution of its processing steps. Both these interfaces can be seen on the right of Fig. 9.

The application is implemented in Python using the open source Django web framework.²⁴ The information displayed in it is taken from an SQL database that is populated by monitor4cheops.

5.6. Mission archive

When an observation has been fully processed, the resulting data products are ingested into the Mission Archive and made available for download via the archive’s web interface. The data owner is automatically notified by email of the availability of their data products.

The archive consists of two components: the archive server and the web interface, the Archive Browser, of which a screenshot is shown in Fig. 9. The archive server is based on a pre-existing generic data archival solution for space missions developed by Deimos Engenharia, S.A. A thin layer of customization was built on top of the generic solution to adapt it to the CHEOPS data model. The Archive Browser is a WordPress based web site that was developed specifically for CHEOPS as an interface between the archive server and the end users. The users can query for data using attributes related to the target and/or the observation. Proprietary data can only be downloaded by the data owner, but once the proprietary period has expired (generally one year after the observation), the data become available to all users.

²⁴ <https://www.djangoproject.com/>

After the end of the mission, all the data will be migrated to the **ESA** archive, guaranteeing long term availability of all the **CHEOPS** products.

5.7. Outlook

The **SOC** system has been running on CentOS 7 since before the launch of **CHEOPS** in 2019. A global system upgrade was planned in 2024, the year when this CentOS version reached end-of-life. As Red Hat had at that point discontinued the development of CentOS in favour of CentOS Stream, AlmaLinux 9, which is binary-compatible with Red Hat Enterprise Linux, was selected as its replacement. The migration entailed upgrades of important third party dependencies, such as Python, C++, Boost and Postgresql, which in turn necessitated updates of the **SOC** software.

At the time of writing in mid-2025, the migration of the compute node and the mission planning system server has been completed and the migration of the Mission Archive is soon to be completed. The archive software is tightly coupled with the OS and it was deemed too costly to migrate the software itself to AlmaLinux. Instead, the **SOC** has opted to containerize the archive (using podman²⁵) to allow the original software to run on the upgraded hosts.

6. Additional tools

This section provides short descriptions of additional software tools used to assist in **CHEOPS** Operations.

6.1. Parameter archive and plotting tool

ESA's Mission Utility and Support Tools (MUST) and its web-based interface, **WebMUST**, are frequently utilized for **TM** visualization and trend analysis as alternatives to SCOS-2000, which is limited in certain functionalities. Originally implemented at the European Space Operations Center, the tool was later transferred to **INTA** and Universität Bern due to maintenance considerations. Currently, its operations and maintenance are managed collaboratively by the operations and instrument teams with the support of Solenix.²⁶ The **MUST** database is updated with RAPID files generated by SCOS-2000. For periodic trend analyses, the Python-based **MUSTLink API** is employed. This **API** facilitates authenticated server access and enables efficient data retrieval by utilizing a paginated approach, which divides large requests into smaller subsets to prevent server overload and ensure robust performance.

6.2. Generic File Transfer System (GFTS)

GFTS is a file transfer system used to automatically deliver files between hosts with or without a **GFTS** instance installed. This tool guarantees the orderly transfer of files among **MOC** components and external entities such as **SOC**, while also verifying the integrity of the received files. **GFTS** establishes the interface with **SOC** via FTP and initiates action procedures to move the files to the appropriate folders.

6.3. CHEOPS Spacecraft Simulator

The spacecraft simulator, based on **ESA's** Simulus 5.4 framework and the SIMSAT platform, was specifically tailored for the **CHEOPS** mission. It provides an accurate representation of the spacecraft's on-board software. Although it lacks certain payload modelling capabilities, the simulator has been instrumental in validating both existing and new operational procedures. It played a critical role during pre-launch simulation campaigns and continues to support training activities and

contingency operations. The simulator features generic models that encompass **TM** and **TC** encoding and decoding, thermal network simulation, and communication with ground stations via the Space Link Extension protocol. Configuration loading and autonomous simulation of specific operations, such as **G/S** operator tasks, are facilitated through scripting in Java.

6.4. CHEOPSim

The **CHEOPS** simulator, described in **Futyán et al. (2020)**, is a standalone **SOC** subsystem that creates accurate simulations of data from the payload, in the same format as the output from the Preprocessing chain (see Section 5.4.1). This tool was primarily used prior to routine operations and its main purposes were to create input data for testing and validating the **SOC** processing chains and to aid in the preparation of observations. Users interact with **CHEOPSim** through a web interface that launches simulations on a computing cluster.

Although beyond the scope of this paper, it is of note that **CHEOPSim** was vital for the assessment of **CHEOPS** scientific and instrumental performance.

6.5. CHEOPS visibility tool

Because of the combination of **CHEOPS** Low Earth Orbit and specific pointing restrictions (Sun and Moon Exclusion Angles), observations are subject to interruptions, and specific lines of sight are only visible seasonally. To help users know when their target is visible with **CHEOPS**, the **SOC** developed, in 2024, the **CHEOPS** Visibility Tool.²⁷ This Python code takes as input visibility and efficiency (fraction of a visit that actually contains images, result of interruptions) tables pre-calculated using the **MPS** at different time of the year and for different line of sights. These tables are interpolated in space and time to provide the user with a sky map showing the cone of visibility of **CHEOPS** for any given day of the year, as well as the window of observability and the efficiency for specific targets of interest.

7. Conclusions

CHEOPS is the first **ESA** S-class mission, dedicated to characterizing exoplanets orbiting around nearby bright stars with exquisite precision, paving the way for flagship **ESA** and **NASA** missions, such as **PLATO** (**Rauer et al., 2014**) or the James Webb Space Telescope (**Gardner et al., 2006**).

The **CHEOPS** ground segment is divided into two main entities, the **MOC** (near Madrid, Spain) and the **SOC** (near Geneva, Switzerland). These entities host the key subsystems and software that enable a seamless and efficient execution of the mission operations workflow. This workflow includes translating scientist observation requests into a plan of activities to be executed by the satellite, and the downlink, processing, and archiving the data to make it available to the scientists.

CHEOPS operations automation was contemplated from the very beginning. The current automation enables a very efficient operation of the ground segment by the **CHEOPS** operations team. With the high level of control over the various ground segment subsystems provided by this automation, routine operations can be performed smoothly without the need for staff members to be present outside regular working hours. This not only reduces workforce demands, but also ensures consistency in the execution of complex processes, and lowers the likelihood of human error. As a result, the teams can focus on performing specialized non-routine manual tasks that require in-depth and specialized knowledge of the different subsystems, enhancing the automation system, and developing new features driven by the evolving

²⁵ <https://podman.io/>

²⁶ <https://www.solenix.ch/products/must>

²⁷ https://gitlab.unige.ch/cheops/CHEOPS_visibility_tool

needs of the scientific community, such as the relaxation of the Sun Exclusion Angle, or data taking inside the South Atlantic Anomaly.

After the initial four years of operations, the mission was granted a three-year extension, and after five years of successful operations, it has become apparent that automation enables the fulfilment of the mission's operational requirements without compromising any of its aspects. **CHEOPS** sets a benchmark for future missions, offering valuable inspiration from the design and tailoring of the ground segment with a strong emphasis on the automation of activities.

The CHEOPS consortium strongly encourages anyone interested in learning more about the inner workings of any aspect of the mission operations to directly contact the team at cheops-psy-members@listes.unige.ch.

List of acronyms

AP	Activity Plan
API	Application Programming Interface
CEIT	Centro Espacial INTA Torrejón
CHEOPS	CHaracterising ExOPlanet Satellite
ECSS	European Cooperation for Space Standardization
ESA	European Space Agency
FDS	Flight Dynamics System
G/S	Ground Station
GFTS	Generic File Transfer System
HMI	Human-machine interface
INTA	Instituto Nacional de Técnica Aeroespacial
MCS	Mission Control System
MIB	Mission Information Base
MOC	Mission Operations Center
MPS	Mission Planning System
MUST	Mission Utility and Support Tools
NASA	National Aeronautics and Space Administration
OBT	On-Board Time
OR	Observation Request
PARC	MCS Packet Archive
PHT2	Proposal Handling Tool phase 2
PSO	Project Science Office
S/C	Spacecraft
SOC	Science Operations Center
SSDC-ISA	Space Science Data Center facility of the Italian Space Agency
SW	Software
TC	Telecommand
TM	Telemetry
UTC	Coordinated Universal Time

CRedit authorship contribution statement

Alexis Heitzmann: Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. **María J. González Bonilla:** Writing – review & editing, Visualization, Project administration. **Anja Bekkelien:** Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. **Babatunde Akinsanmi:** Writing – review & editing. **Mathias O.W. Beck:** Writing – review & editing, Software, Project administration. **Nicolas Billot:** Writing – review & editing, Software, Project administration. **Christopher Broeg:** Writing – review & editing, Project administration. **Adrien Deline:** Writing – review & editing, Software, Project administration. **David Ehrenreich:** Project administration. **Andrea Fortier:** Project administration. **Marcus G.F. Kirsch:** Writing – review & editing, Project administration. **Monika Lendl:** Writing – review & editing, Project administration. **Nuria Alfaro Llorente:** Project administration. **Naiara Fernández de Bobadilla Vallano:** Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. **María Fuentes**

Tabas: Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. **Anthony G. Maldonado:** Writing – review & editing, Writing – original draft, Visualization, Software, Data curation. **Eva M. Vega Carrasco:** Project administration. **David Modrego Contreras:** Writing – review & editing, Writing – original draft, Visualization, Software, Project administration, Data curation.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, openAI's ChatGPT was used a few times by the authors to rephrase and clarify complex paragraphs. Each modification by ChatGPT was thoroughly reviewed to avoid any change in meaning and solely address clarity. The authors assume full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

CHEOPS is an ESA mission in partnership with Switzerland with important contributions to the payload and the ground segment from Austria, Belgium, France, Germany, Hungary, Italy, Portugal, Spain, Sweden, and the United Kingdom. The **CHEOPS** consortium would like to gratefully acknowledge the support received by all the agencies, offices, universities, and industries involved.

The Swiss ground segment development and operations were funded by the Swiss Space Office through the ANC and PRODEX programmes.

The Spanish **MOC** was funded by the Spanish Center for the Development of Technology and Innovation (CDTI) through the GSTP and PRODEX programmes.

After five years of operations, **CHEOPS** continues to run smoothly, an achievement made possible by the dedication and expertise of those who contributed to the development of its subsystems. The authors would like to express our heartfelt appreciation to the following individuals, whose flexibility, willingness to explore new approaches and invaluable efforts in the design, development, and management of the **SOC** have been instrumental to the success of **CHEOPS**: Mathias Beck, Reiner Rohlf, David Futyan, Mohamed Meharga, Didier Queloz. We also wish to acknowledge all contributions from: Florian Georges, Magali Deleuil, Sergio Hoyer, Pascal Gutterman, Olivier Demangeon, Jean-Charles Meunier, Nick Walton, Guy Rixon, Gabor Kovács, Alexis Brandeker, Philip Loschl, Roland Ottensamer, Francesco Verrechia, Kate Isaak, Carlos Corral Van Damme and Willy Benz.

The authors also warmly acknowledge the contribution of the whole team at Deimos Engenharia, S.A.: Antonio Gutiérrez, Carlos António, Ignacio Garcia and Inês Estrela for the design and development of the **MPS** and the Archive, and their continuing support. Deimos Engenharia, S.A. also wishes to acknowledge contributions from past Deimos collaborators to the development of the **MPS**: Alejandro Martinez-Herrera, Diogo Andrade, Henrique Sousa, Marcos Bento, Paula Guerreiro, Pedro Neves, Rita Castro, Sofia Freitas, and Tural Malikli.

The authors express their gratitude to Richard Thomas Southworth for his invaluable contributions to mission operations, leveraging his extensive expertise at the European Space Operations Center. Additionally, they sincerely acknowledge José Ramiro Peñataro Blanco from GMV for his continuous technical support to daily operations. Their guidance has proven essential to the success of the operations throughout the mission. Finally and equally deserving of recognition, are the CEIT G/S and IT Network teams, whose indispensable contributions serve as the foundation of the MOC.

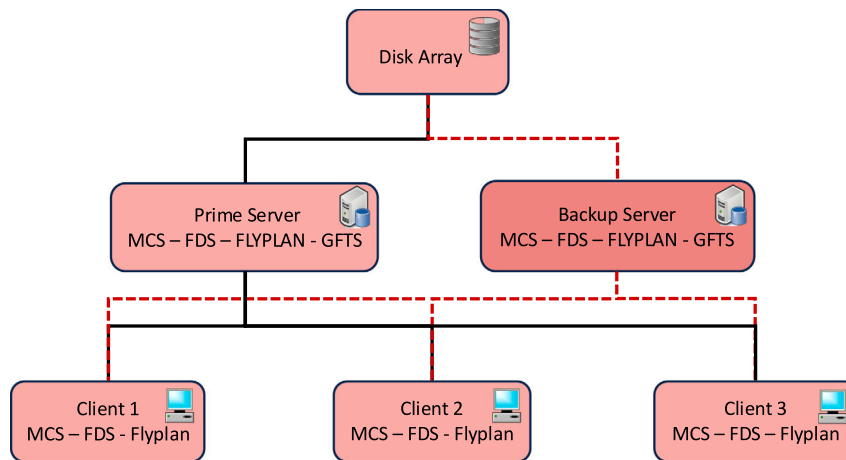


Fig. 10. MOC Operational platform architecture. The solid lines indicate the nominal configuration, whereas the dashed lines illustrate the redundant configuration.

Appendix. Hardware architecture

In this appendix, the hardware infrastructure hosting the CHEOPS subsystems performing Operations related tasks is briefly described. As the Mission and the Science Operations Centers are two physically distinct entities (near Madrid and Geneva respectively), they each have their own infrastructure, that are linked through an FTP server.

A.1. MOC

MOC is composed of two platforms. The operational platform includes all the hardware and software used to actually monitor and control the spacecraft. Its architecture, shown in Fig. 10, is composed of two servers, three clients and a disk array where all the data are stored in RAID-6 configuration. Only the nominal server is able to read/write data on the array, but a change to the redundant configuration can be executed by switching off all tools, unmounting the disk from one server, mounting it in the other one and switching on the tools on the new server. The operational software is composed of the MCS, FDS, automation system and file transfer tool. MCS, FDS and the automation system are based on a server/client structure, with their server sides installed in all the servers and their client sides installed in all the clients. The file transfer tool is only installed in the servers.

The second platform is referred to as the reference platform and is used for training and validation. Originally based on a physical platform, it was migrated to a virtual environment in 2022 when hardware maintenance became problematic. This platform is composed of two servers and one client. One of the servers and the client are configured in the same way as the ones in the operational platform (with the MCS, FDS and automation servers) and the other server is used for the satellite simulator. The reference platform can be configured to access the G/S so it can be used as back up of the operational platform.

A.2. SOC

The SOC system is based entirely on virtual machines and uses hardware infrastructure shared with other department-level projects for all of its processing and data storage needs. The system, which is shown in Fig. 11, consists of a web server for the PHT2, a server for the MPS, an FTP server for file transfer to and from MOC, a compute node running the data processing pipeline, with access to a computing cluster for resource intensive processing steps, a mission archive server and its accompanying web server hosting the archive web interface. The SSDC-ISA hosts mirrors of the archive and archive web servers.

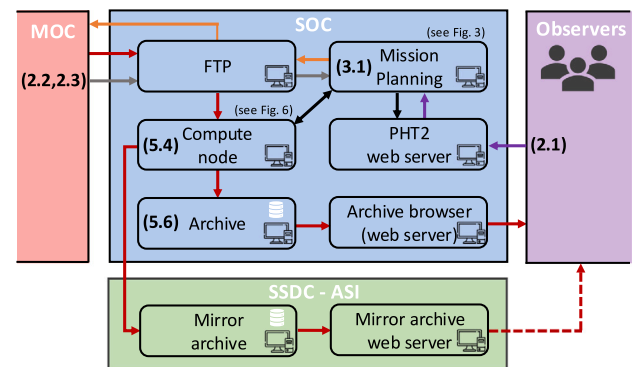


Fig. 11. Overview of the SOC's hardware infrastructure. Black outlined boxes accompanied by a computer logo show the different virtual machines. The interaction between each machine, both internally, and with external entities such as MOC (red box), CHEOPS observers (purple box), and the mirror archive (SSDC-ISA, green box) are shown with coloured arrows: (i) purple for the ORs from the observers, (ii) grey for files provided by MOC and used for the mission planning, (iii) orange for files generated by the mission planning, (iv) red for all data related products (raw and processed, science and housekeeping data) and (v) black for other files internally generated and exchanged between machines. The numbers in bracket indicate the subsection describing the sub-processes of interest.

Data availability

No data was used for the research described in the article.

References

- Acton, C.H., 1996. Ancillary data services of NASA's navigation and ancillary information facility. *Planet. Space Sci.* 44 (1), 65–70. doi:10.1016/0032-0633(95)00107-7, Planetary data system, URL: <https://www.sciencedirect.com/science/article/pii/0032063395001077>.
- Benz, W., Broeg, C., Fortier, A., et al., 2021. The CHEOPS mission. *Exp. Astron.* 51 (1), 109–151. doi:10.1007/s10686-020-09679-4, arXiv:2009.11633.
- Consultative Committee for Space Data Systems (CCSDS), 2010. Communications Operation Procedure-1 (COP-1). Technical Report CCSDS 232.1-B-2, Consultative Committee for Space Data Systems (CCSDS), Blue Book, Issue 2, URL: <https://ccsds.org/Pubs/232x1b2e2c1.pdf>.
- Fortier, A., Simon, A.E., Broeg, C., et al., 2024. CHEOPS in-flight performance. A comprehensive look at the first 3.5 yr of operations. *Astron. Astrophys.* 687, A302. doi:10.1051/0004-6361/202348576, arXiv:2406.01716.

- Futyan, D., Fortier, A., Beck, M., et al., 2020. Expected performances of the Characterising Exoplanet Satellite (CHEOPS). II. The CHEOPS simulator. *Astron. Astrophys.* 635, A23. doi:10.1051/0004-6361/201936616, arXiv:2001.05587.
- Gardner, J.P., Mather, J.C., Clampin, M., et al., 2006. The James Webb Space Telescope. *Space Sci. Rev.* 123 (4), 485–606. doi:10.1007/s11214-006-8315-7, URL: <https://link.springer.com/10.1007/s11214-006-8315-7>.
- Hoyer, S., Guterman, P., Demangeon, O., et al., 2020. Expected performances of the characterising exoplanet satellite (CHEOPS) - III. Data reduction pipeline: architecture and simulated performances. *Astron. Astrophys.* 635, A24. doi:10.1051/0004-6361/201936325.
- Peccia, N., 2003. SCOS-2000 esa's spacecraft control for the 21st century. In: 2003 Ground System Architectures Workshop.
- Pence, W., 1999. CFITSIO, v2.0: A New Full-Featured Data Interface. In: Mehringer, D.M., Plante, R.L., Roberts, D.A. (Eds.), *Astronomical Data Analysis Software and Systems VIII*. In: *Astronomical Society of the Pacific Conference Series*, vol. 172, p. 487.
- Pence, W.D., Chiappetti, L., Page, C.G., et al., 2010. Definition of the flexible image transport system (FITS), version 3.0. *Astron. Astrophys.* 524, A42. doi:10.1051/0004-6361/201015362.
- Rauer, H., Catala, C., Aerts, C., et al., 2014. The PLATO 2.0 mission. *Exp. Astron.* 38 (1–2), 249–330. doi:10.1007/s10686-014-9383-4, URL: <http://link.springer.com/10.1007/s10686-014-9383-4>.
- Winn, J.N., 2010. Exoplanet Transits and Occultations. In: Seager, S. (Ed.), *Exoplanets*. pp. 55–77. doi:10.48550/arXiv.1001.2010.