

Supporting the Community Approach by hosting a Collaborative Development Platform for the SST Core Software

Hélène Ma⁽¹⁾, Elmar Brendel⁽²⁾, Johannes Klug⁽²⁾, Rui Gomes Gondar⁽³⁾, Tim Flohrer⁽¹⁾, Christian Unfried⁽⁴⁾, SST User Forum⁽⁵⁾

(1) OPS-SD, ESA/ESOC, Robert-Bosch-Strasse 5, DE-64293 Darmstadt, Germany
Email: helene.ma@ext.esa.int, tim.flohrer@esa.int

(2) OPS-GAE, ESA/ESOC, Robert-Bosch-Strasse 5, DE-64293 Darmstadt, Germany
Email: elmar.brendel@ext.esa.int, johannes.klug@esa.int

(3) OPS-SC, ESA/ESOC, Robert-Bosch-Strasse 5, DE-64293 Darmstadt, Germany
Email: rui.gomes@ext.esa.int

(4) Space Analyses GmbH, Marxergasse 24/2, A-1030 Wien, Austria
Email: christian.unfried@spaceanalyses.at

(5) SST UF
Email: SST-UF@esa.int, Germany

ABSTRACT

The implementation of a community approach for the Space Surveillance and Tracking (SST) Core Software has been a key objective of the Space Situational Awareness (SSA) and Space Safety programmes (S2P). ESA has put forward software suite upgrades for the processing of observation data, including functions for scheduling, orbit determination and correlation as well as SST service output generation. Three ESA activities are being conducted in parallel to maintain and modernize the system and to introduce new features and add-ons.

Within the P3-SST-XXVI contract “Core Software Development Baselines and Architecture Modernisation under Community Approach”, the software repository structure has been modularized to allow better maintainability of the source code. A major effort has been taken to introduce automatic test cases as part of the CI/CD pipeline system. This improves the robustness of the software in corrective and evolutive maintenance by detecting failures early enough due to the re-execution of the automated tests. Another main new feature is the parallelization of tracklet processing within the Data Processing Chain subsystem.

The P3-SST-XXVII contract “Customised SST software elements in the SST Core Software and Expert Centre” has performed an exemplary tailoring of the software to the needs of the Polish community. Especially the alignment of the interface with the Polish sensors was required. The SST Expert Centre software has been adapted to link between the Polish sensors on one side and the Core Software on the other. An end-to-end validation campaign involving measurement data from an observation campaign was performed to demonstrate

the integration of Expert Centre and Core Software and to assess the build-up and maintenance of an object catalogue.

The S1-SC-08 contract “Supporting the community for the Core Software development, maintenance, and standardisation” is operating the Collaborative Development Platform (CDP) outside the ESA environment to support the Core Software Community. Within this activity, a dataset of reference scenarios and a set of tutorials is made available to the users to familiarize with the software and to test individual use cases. In addition, a new method for track-to-track correlation was implemented in the software to improve the exploitation of tracking data for more efficient cataloguing of objects.

The SST User Forum (SST-UF), with the background of the multitude of SSA systems under development and operational implementation in the participating states, has collected and consolidated the user needs into a dedicated high-level roadmap. A licensing approach and collaboration schema for the future community work has been developed, and tools to coordinate the community in further evolving the approach and the related baseline have been deployed in the CDP. The user community has demonstrated its readiness to control and lead future releases of the software, while tailoring their national deployments. This forms the basis for further discussion of the software needs and prioritisation of developments for the future evolution of the Core Software.

This paper will summarise the outcomes of the three ESA activities and focus on the progress the community approach has made with the active involvement of the SST-UF in the Core Software developments.

Keywords: Community Approach; SST User Forum; Collaborative Development Platform; SST Core Software; ESA's Space Safety Programme.

1 INTRODUCTION

In support of the European partners ESA introduced the concept of a community approach towards an Space Surveillance and Tracking (SST) Core Software in 2017 [1][2] to provide interoperability, compatibility and harmonisation in system outputs and services provision of the various developments in Europe, while allowing closed-source national tailoring according to specific needs.

The SST Core Software is a toolset designed as a set of SST core services and applications. Its implementation and evolution have been a key objective of ESA's Space Situational Awareness (SSA) programme, and now of the Space Debris and Clean Space segment of ESA's Space Safety Programme (S2P), aiming for the goal of European autonomy in accessing and using space in a safe and secure environment.

This paper briefly describes the SST Core Software and then presents the ongoing workplan activities focusing on the advancements in implementing, extending and deepening the community approach which includes a User Forum and a Collaboration and Software Development Framework.

2 SST CORE SOFTWARE

The SST Core Software is a system of SST services and

applications which, starting from SST sensor data and third-party orbital data, cover the detection, identification, cataloguing, and monitoring of artificial objects orbiting the Earth, the prediction and analysis of risk events, and the observation planning and sensor tasking (Figure 1).

As described in [3], this system is a composite of components and tools that provide a comprehensive aggregation of functionalities to process measurements from different sensor data types (radar, passive optical, and laser) into data and services required by the SST user community.

The system is designed to be modular, i.e. to facilitate future additions as well as tuning of the software to specific user needs. All the subsystems apply the existing standard data formats used in SST activities such as, but not limited to, CCSDS (Consultative Committee for Space Data Systems) and CEN-CENELEC (the European Committee for Standardisation together with the European Committee for Electrotechnical Standardisation) data formats. Therefore, external entities like a Sensor Simulator, a directly external sensor, or an Expert Centre [4], can be integrated seamlessly with the SST Core Software. These entities are expected to work autonomously and to develop capabilities to interact with the Data Processing loop of the system.

The SST Core Software has been containerised using Docker during previous ESA development activities. Gitlab Continuous Integration and Continuous Development (CI/CD) pipelines have been in place to automatically build, test and deploy the application. The latest iteration, additions and improvements of this approach is detailed in the next section.

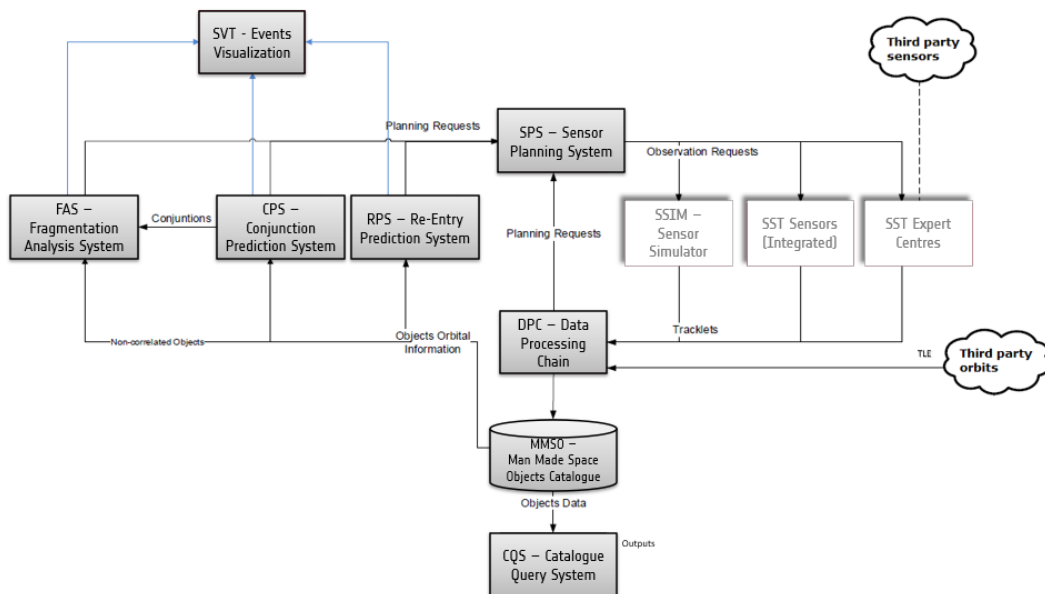


Figure 1: Architecture of the SST Core Software at system level.

3 OUTCOMES OF THE DEVELOPMENT ACTIVITIES DESCRIBING THE MODERN ARCHITECTURE AND NEW FUNCTIONALITIES

Over time and during the course of various SST Core Software activities, several new features, issues, gaps and problems were identified and tracked in a ticket database that has been populated with it. The criticality and urgency for those tracked tickets has been iteratively checked to make a selection and proceed with the implementation phase. Each time a set of improvements was selected and implemented.

The current state of the SST Core Software offers an integrated application consisting of modular tools and services [5]. It allows end-to-end workflows within one single system. Yet, it can also interface with other external tools and data providers beyond its own boundaries.

The overall design is inspired by the service-based architecture concept. Individual modules offer services via Simple Object Access Protocol (SOAP) interface endpoints. These are utilised internally, e.g. when an operator triggers an action via a GUI, or when a specific module notifies another subcomponent about an event. This concept is implemented based on the WildFly application server framework which each Core Software subcomponent runs to expose and offer its own services to others.

In a first step, the software has been containerised to remove direct dependencies on host environments and to transition towards a more micro-service inspired approach. This enables to adopt more modern hosting paradigms leaning towards cloud environments. All SST Core Software services and modules have been split into docker containers.

To efficiently build the container images (and the

comprised applications within), a CI/CD approach based on Gitlab pipelines was chosen. Recently this has been further refined and now allows modular source code repositories per micro-service. Individual changes on a specific module will only trigger the sub pipeline of the corresponding micro-service. This avoids running the full build process for the entire application each time. However, the means of running the full build, and deploying the software as whole have been aligned and still maintained.

An entirely new Gitlab project has been added that allows the execution and orchestration of automated test cases via a CI/CD pipeline on the target host machine. This includes even GUI-based invocation of test steps which has been achieved with Eclipse SWTBot. For this automation concept, an addon test container is deployed next to the Core Software stack that contains the input data and test scripts (based on JBehave) responsible for the execution of the individual test cases (Figure 2). The outcomes can be reviewed in reports or as job artifacts in the performed CI/CD pipeline runs.

Another large update was introduced on the Data Processing Chain (DPC) module. The processing of sensor data has been parallelised. Multiple DPC docker container instances can be spawned to concurrently work on the retrieved data. For this purpose, an Apache Kafka message broker has been added that coordinates the distribution of processing jobs per instance. In a potential further evolution, such message broker or message queue system could help to streamline the system internal communication in the future when adopting it centrally for all the components.

Also, on DPC side, algorithmic improvements have been carried out. The so-called track-to-track correlation method was introduced [5]. It is more efficient in exploiting at best the received tracking data to handle the object catalogue build-up and updates based on identified

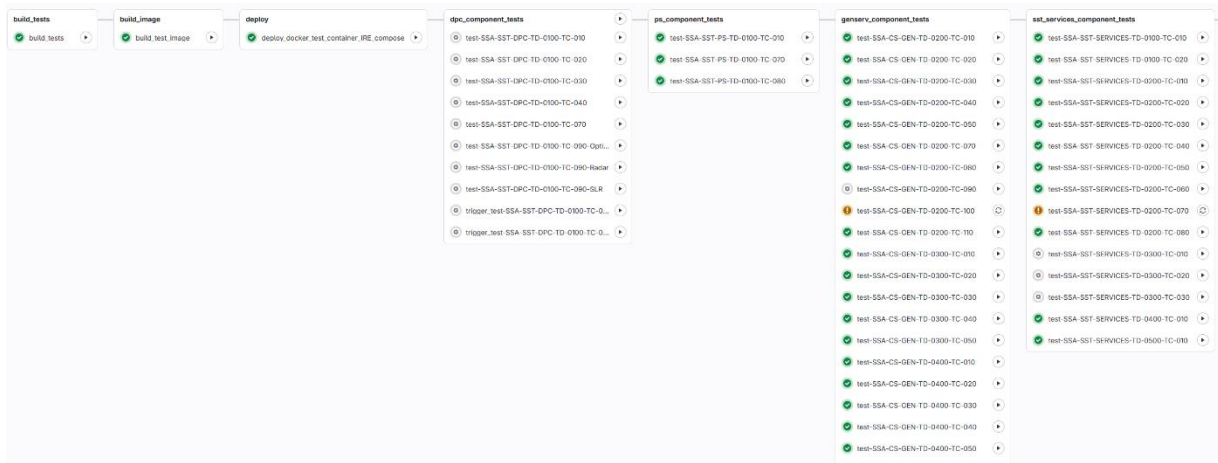


Figure 2: Overview of an automated test case pipeline run in Gitlab.

objects.

Further test campaigns have been carried out to process real sensor data in an integrated environment comprised of the SST Core Software with the SST Expert Centre, which is another standalone application. This setup can offer more elaborate workflows for tasking more specialised sensors.

Among the tickets tracked from the database, some are being tackled to ensure the modularisation of the software. While the SST Core Software already considers a prototype of the Fragmentation Data Message (which has the CCSDS format like but does not exist yet in CCSDS), the software can process TDM version 2.0 (CCSDS 503.0-B-2.0, Blue Book, June 2020) and it is planned on the ability to read CDM version 2.0 (CCSDS 508.0-P-1.1, Pink Book, March 2024).

4 APPLICATION OF THE COMMUNITY APPROACH VIA A USER FORUM

Since 2017, the SST User Forum (SST-UF) acts as the community’s technical advisory body for the SST Core Software. As of March 2025, the Forum is composed of the following Member States that are participating with their national representatives to support the community needs with ESA: Austria, Germany, Finland, Italy, The Netherlands, Norway, Poland, Portugal, Spain, Sweden, Switzerland, and the United Kingdom. Each Member State that subscribes to ESA’s Space Safety Programme is eligible for SST-UF membership, hence can propose to

join the SST-UF and designate one technical representative for their participation.

The SST-UF and ESA are meeting twice a year to share the status of the Programme activities related to the developments of the Core Software, and to brief on the national activities in the domain and on the national community needs. Demonstrations are also conducted to SST-UF members on the software status and capabilities, with workshops addressing specific topics areas.

Since 2020, the SST Core Software is released under the ESA Community license (strong copyleft, v 2.4) to the Member States participating in the SST-UF. Licenses can be granted to the SST-UF Point of contact, and sublicensing is possible.

The SST-UF has been playing a key role in providing guidance on all aspects of the software evolution, from the consolidation of requirements and their implementation to the governance and licensing approach and the standardisation efforts for SST data exchange formats.

4.1 Governance model and Collaborative Development Platform

The details of the SST-UF Governance Model are now matured with the identified stakeholders, their roles and the processes linking it all, see the general workflow in Figure 3. Note that the governance process excludes the case when proprietary developments are not shared to the community.

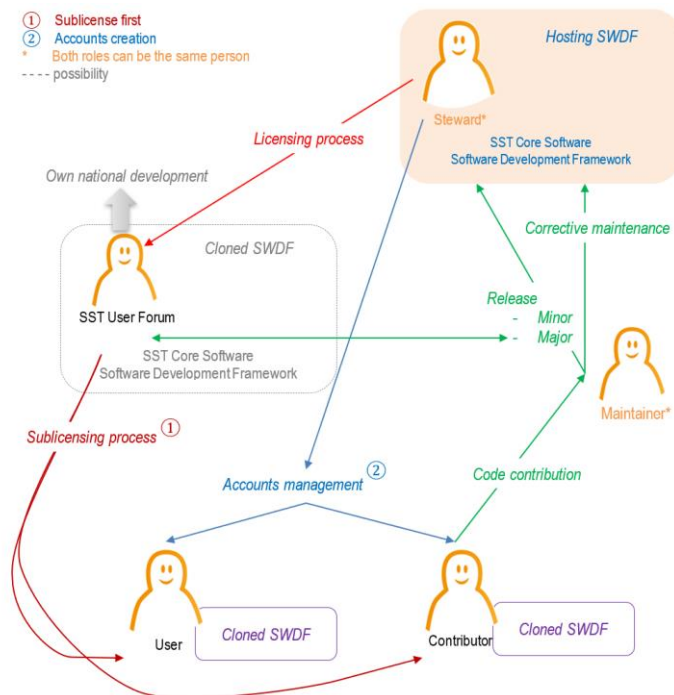


Figure 3: Workflow of the Core Software development within the collaboration framework described from the governance model.

- The SST-UF acts as the steering board deciding on the general direction of the Core Software, hence on the big technical evolutions. The SST-UF (or their nominated software engineering delegates) shall also have an overview of their planned and ongoing national contributions and agree to freeze periods if that were to be required for their contributions. They also handle their national sublicensing process and can run their own copy of the Collaboration Framework (SWDF) for their national developments.
- The Steward hosts the SWDF and handles the accounts management for Contributors and Users with the approval of the national representatives.
- The Maintainer will likely be carried out by the same entity as the Steward: this role performs corrective maintenance, issues minor software releases, and major software releases with SST User Forum agreement. The Maintainer supports documentation quality.
- The Contributors and Users first seek licensing via their national User Forum representative. They are the national stakeholders who get the permission to work with a copy of the SST Core Software. Both roles can overlap sometimes, but the Users will operate the software for their purposes, while the Contributors develop and submit code contributions to the software via the Maintainer. The Contributors create automated unit tests, extend the build pipeline to automatically build their code, provide good documentation, and resolve any merge conflicts before submitting the code to the maintainer. The Users are strongly encouraged to feedback their bugfixes and invited to share their developments to the Contributors with the community.

Note that this model shows a series of recommended approaches, as these strongly depend on the project and team in question. The model has already been put into practice with a licensing approach and a collaboration schema (SWDF) under the Collaborative Development Platform (Figure 4) established in the frame of P3-SST-XXVI and S1-SC-08 activities.

4.2 The Collaborative Development Platform

The Collaborative Development Platform (CDP, <https://cdp.sst-cs.org/>) provides four main tools to the community members. Their main common purpose is allowing to coordinate the further evolution of the community approach and running the software lifecycle management for the related application baseline.

Gitlab is the key component of the CDP. It enables to define the workflows, roles and processes needed to manage the software releases. It hosts the source code repositories, organised in several modularised subprojects. The version control and branching are controlled from here. Following the agreed community

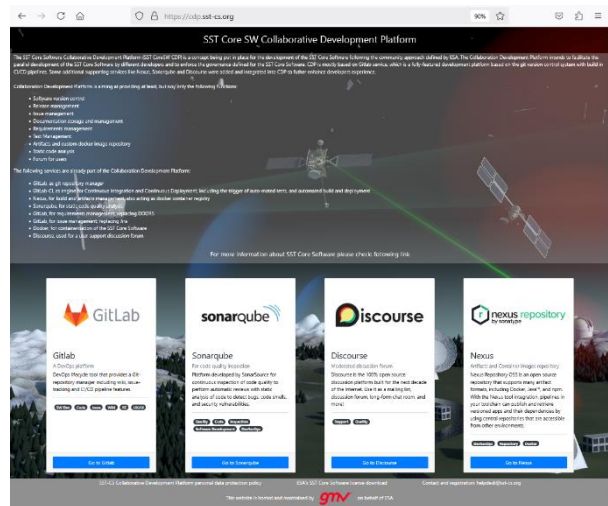


Figure 4: Landing page of the Collaborative Development Platform (CDP).

governance model and the included processes, the adoption of new software patches and features (submitted by “Contributors”) can be managed from Gitlab. This is typically handled by merge-requests which can include code review cycles and approvals steps executed by specifically defined user roles (“Maintainer”). Another important feature of Gitlab is the already described CI/CD pipelines which allow to build, test and deploy the software. Again, specific users can own privileged roles in order to trigger CI jobs, perform configuration changes or deploy the software stack to a certain runtime environment. Lastly, Gitlab provides the means to keep track and host auxiliary information. Software documentation and user tutorials can be directly stored in each individual project or at project group level. Generic ‘issue’ tickets can be labelled and tagged to maintain lists of items such as requirements, user stories or bug descriptions which are also linkable.

Nexus provides an archive of the generated outputs of the Gitlab CI/CD pipelines. It stores the final docker images representing the runtime software. Depending on the software versioning scheme defined by the governance model, it can host the history of previously generated images. Next to the docker images, also lower-level products can be stored such as the individual modules, libraries and binaries of the different SST Core Software subcomponents. These will be picked up by later stages of the CI/CD pipeline to assemble the final images. Not only the outputs are stored in Nexus, but also (previously agreed) dependencies which are needed as a starting point to build and compile the applications. These are typically bespoke binaries and libraries that are not part of common open-source standard repositories and unique to the SST community.

SonarQube allows to scan and review the source code from a software engineering point of view. Several default quality metrics are in place, which measure

different aspects like performance, code coherence and segregation, security risk as well as other characteristics. These metrics are definable per programming language. It is also possible to introduce custom quality profiles if more specific needs exist. The identified QA issues are ranked based on their severity to allow programmers to prioritize their review and code refactoring activities. Within the CDP, the SonarQube is embedded as CI/CD pipelines step. It can be configured to block the stage or entire run of the pipeline in case the SonarQube quality checks are (severely) violated.

Discourse is a message board platform to allow users to communicate and exchange information according to configurable topics. It is a central system for the community to request support or raise identified issues, post experiences and lessons learned as well as discussing further steps to evolve the SST Core Software. All interested parties of the SST UF member states can also use this platform to directly communicate with the other stakeholders (Section 4.1) and ESA for the same purposes as well as for any preparation required for the meetings.

Apart from the visible end-user services, there are infrastructure components for the purpose of hosting and maintaining the platform itself. A NGINX webserver is responsible for routing the traffic to each individual service and to provide the main landing page. There is a single-sign-on solution that allows to access each individual end-user service with the same personal credentials. Furthermore, anti-virus & -malware scanners are in place as well automatic data backup processes for disaster recovery. The responsibility for the platform is with admin-staff who monitors the platform, proactively review events and checks log messages as well as applies recent security patches or newer software versions of individual components. They are also (as “Stewards”) the

first point of contact for end-users with technical difficulties or support with moderating the discussions on Discourse.

4.3 Roadmap

One purpose of the SST-UF is to determine the overall direction of the community’s work on the software development and collaboration framework. Therefore, the forum members are regularly invited to translate their current needs of their national communities into a set of user requirements and stories listed by priority order. These needs were also collected and consolidated into a dedicated high-level roadmap as shown in Figure 5.

This roadmap gives a global vision of the progress towards a goal of the SST Core Software from the current parent baseline and split into different categories. It shows the main tasks achieved within the on-going activities, and a backlog of main gaps that are identified in these activities and those describing the needs expressed by the SST-UF. Within a single snapshot, the roadmap pictures the state of the SST Core Software and the community approach, hence supports the SST-UF to commonly agree on the development planning.

5 OUTLOOK

Within the on-going activities, the SST Core Software has evolved into a modern architecture as a first step. Additionally, the data processing has been parallelised and improved with a new track processing [5] and the integration of the Expert Centre to the SST Core Software has been demonstrated. Some SST-UF needs are being responded and implemented in the system, and the community approach has been successfully applied

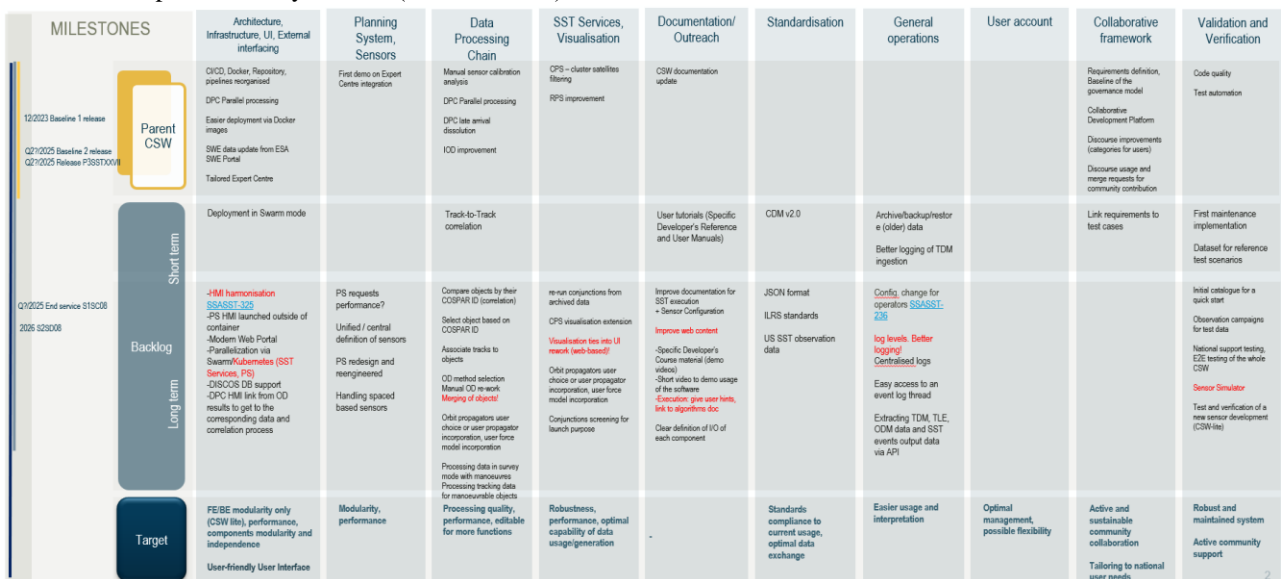


Figure 5: High-level roadmap.

within the Collaborative Development Platform (CDP).

Space Debris, 1-4 April 2025, Bonn, Germany.

With Period 2 of ESA's Space Safety Programme, a new activity will soon kick off with the demonstration of the full deployment and end-to-end testing of the Core Software in a national and operational environment.

While the CDP will continue to be hosted, the active support from the user community on the development and testing of the system will be demonstrated, including user stories definition and consolidation, and more effort on the new standardization formats compliancy to the current usage scenarios of the national entity as support for Validation and Verification process.

6 ACKNOWLEDGMENTS

The authors would like to thank the SST User Forum members for their contribution to this paper, and for their active contributions to the User Forum in general.

7 REFERENCES

1. Space Surveillance and Tracking in ESA's SSA Programme, T. Flohrer and H. Krag, In: *Proceedings of the Seventh European Conference on Space Debris*, 18-21 April 2017, ESA/ESOC, Darmstadt, Germany.
2. SST-CC: A community Approach for a European Space Surveillance and Tracking Core Software Development, Christian Unfried et al., *6th European Ground System Architecture Workshop*, 20-21 June 2017, ESA/ESOC, Darmstadt, Germany.
3. Establishing a community approach to SST Core Software through a User Forum, J. Klug, H. Ma, T. Flohrer, E. Brendel, E. Villanueva, J. Tirado, A. Mediavilla Garay, C. Unfried, M. Nylund, the SST UF, In: *Proceedings of the Eighth European Conference on Space Debris*, 20-23 April 2021, Darmstadt, Germany.
4. A Pan-European Expert Centre service and coordination facility in support of space surveillance, C. Paccolat, T. Schildknecht, M. Steindorfer, B. Jilete, T. Flohrer, In: *Proceedings of the Eighth European Conference on Space Debris*, 20-23 April 2021, Darmstadt, Germany.
5. Improvements to SST Core Software: Advanced Software Engineering Practices and Integration of Track-to-Track Algorithm, Diego. Ramírez Rodríguez, J. D. Mchugh, F. J. Simarro Mecinas, N. Mathews, A. Scalabrin, J. Beck, I. Holbrough, E. Brendel, H. Ma, J. Klug, R. M. Gondar, In: *Proceedings of the Ninth European Conference on*