**Automated Service Builder for Semantic Service Oriented Architectures**

**ASB**

**Executive Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | : | ASB Executive Summary | | |
| Abstract | : | The ASB project has developed a dynamic, scalable multi-mission (automated) processing framework. Scalability is achieved by using a Cloud environment for running the processes and by dynamically managing the available Cloud resources depending on the processing demands. Based on the original work and a CCN to address the rapid evolution of user needs during the past 4 years since the start of the project ASB now provides a framework to create systematic, automated processing facilities such as those needed by IPFs and PDGS systems. ASB provides a development layer for building product processors which can be executed by users for on-demand processing or via an API for integration into GIS systems. | | |
|  | |
|  | |
| ESA Contract N° | : | ESRIN/4000112543/14/I-NC | | |
| Technical Officer | : | Philippe Mougnaud ESA/ESRIN | | |
|  |  |  | | |
|  | ESA CONTRACT REPORT  The work described in the report was done under ESA contract.  Responsibility for the contents resides with the author or organisation that prepared it. | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| DOCUMENT APPROVAL SHEET | | | |
| Responsibility | Name & Position | Company | Signature and Date |
| Prepared by: | Bernard Valentin  Project Manager | Space Applications Services |  |
| Leslie Gale  M&BD Manager | Space Applications Services |  |
| PA Checked by: | PA&S Engineer | Space Applications Services |  |
| Approved by: | Project Manager | Space Applications Services |  |

|  |  |  |
| --- | --- | --- |
| Space Applications Services NV/SA |  | Tel: +32-(0)2-721.54.84 |
| Leuvensesteenweg 325 |  | Fax: +32-(0)2-721.54.44 |
| 1932 Zaventem, Belgium |  | Web: www.spaceapplications.com |

This document is the property of

Space Applications Services NV/SA.

All rights reserved.

Table of Contents

[1 INTRODUCTION 4](#_Toc528350630)

[1.1 Background 4](#_Toc528350631)

[1.2 Motivation 4](#_Toc528350632)

[1.3 Key Objectives 5](#_Toc528350633)

[1.4 ASB Evolution 6](#_Toc528350634)

[2 TECHNICAL CHALLENGES 7](#_Toc528350635)

[2.1 Introduction 7](#_Toc528350636)

[2.2 Design Influences and Trade-offs 7](#_Toc528350637)

[2.3 Platform and Application Agnostic Framework 7](#_Toc528350638)

[2.4 OGC WPS‑Based Interface to Processes 9](#_Toc528350639)

[3 ASB FRAMEWORK 10](#_Toc528350640)

[3.1 ASB Components 10](#_Toc528350641)

[3.2 Generic Modelling Scheme and Associated Tools 12](#_Toc528350642)

[3.3 Built-in-Operations 12](#_Toc528350643)

[4 INSTANTIATING A PROCESSING FACILITY USING ASB 13](#_Toc528350644)

[5 DEMONSTRATIONS AND USER EVALUATION 16](#_Toc528350645)

[6 DISSEMINATION 18](#_Toc528350646)

[7 CONCLUSIONS 19](#_Toc528350647)

[8 FUTURE POTENTIAL 20](#_Toc528350648)

**List of Figures**

[Figure 1 – ASB Execution Modes 8](#_Toc528350649)

[Figure 2 – Virtual Machines VS Docker Containers 8](#_Toc528350650)

[Figure 3 – ASB framework Logical Architecture 10](#_Toc528350651)

[Figure 4 - Setup Steps: Algorithm Import and Workflow Configuration 13](#_Toc528350652)

[Figure 5 – ASB Processor Workflow Editor 14](#_Toc528350653)

[Figure 6 – PROBA‑V n‑Daily Compositor Parameterization Form 14](#_Toc528350654)

[Figure 7 – Product Order Execution Report 15](#_Toc528350655)

[Figure 8 – ASB as the technology/services basis for platform and services for Exploitation Platforms 20](#_Toc528350656)

[Figure 9 - EOPEN application of the ASB Framework 20](#_Toc528350657)

List of Tables

[Table 1 - ASB Components 11](#_Toc528350658)

[Table 2 – Coverage of Common Architecture Objectives 21](#_Toc528350659)

# INTRODUCTION

This document is the Executive Summary for the project (ASB) under ESA contract number 4000112543/14/I-NC. The ASB project was led by Space Applications Service (SpaceApps) with, in the main ASB project completed July 2017, subcontractors Spacebel and VITO providing the following contributions:

* Spacebel – designing, implementing, and testing the Credit Service component.
* VITO – analysing the SMOS and PROBA‑V processors and supporting the implementation of the PROBA‑V Demonstrator

A CCN was performed by SpaceApps focusing on developments to support new needs from algorithm developers and users (ASB Evolution). The CCN started in January 2018 and was completed in September 2018.

## Background

The ASB project was initiated in 2014 to investigate and demonstrate new approaches for developing the Processing Facility (PF) of a PDGS. In 2014 it was still common to have dedicated server based processing facilities. Software teams had the task of translating the scientifically developed and tested algorithms into an operational Instrument Processing Facility (IPF) taking into account processing and storage capacity limits imposed by the mission’s server/workstation-based processing facility (PF).

This is reflected in the SoW for ASB:

*Physically the PF is supposed to consist of several workstations. Each PF workstation hosts one or more IPFs, and a set of configuration files. Such deployment strategy on the one hand requires dedicated hardware for setting up the final PF, and on the other hand limits its intrinsic flexibility and scalability.*

The main objective of the ASB project as formulated in the SoW was to analyse new approaches for the deployment and management of the PF. In particular, the project shall address the re-organization of tasks (e.g. IPF modules) to increase the modularity and improve the performances, e.g. maximising the level of parallelization.

## Motivation

The framework used for the delivery of EO Services has evolved from Server-based to Grid-based processing environments (e.g. G‑POD), and onward to Cloud Computing. ESA at ESRIN within the context of several projects has been investigating and applying OGC standards, and more specifically the Web Processing Service (WPS) Interface Standard that provides rules for standardizing inputs and output for geospatial processing services. These standards are adopted in various domains such as Industry, Government and Academic. In 2014 they were underused in EO Services.

IPF’s transform instrument raw data into higher-level products by means of processors forming end-to-end Processing Chains. Each processor applies algorithms to correct, transform, merge data products or extract specific features from them. In the last years, the enthusiasm of the IT sector for Big Data fostered the development of Cluster-based computing platforms such as Hadoop and Spark, and Cluster Management frameworks such as Mesos. Additionally, Cloud technology led major EO actors such as NASA JPL to experiment the distribution of image processing into the Cloud.

Many operational IPFs and similar implementations of Processing Chains used at prototyping level were designed and deployed prior to the Cloud era that do not take advantage of the distribution of the processing power on a Cluster. They are typically deployed into static architectures, where updates are tedious, and scalability is hardly possible.

To investigate new alternative approaches it was proposed that ASB should adopt the latest technologies promoting the work of ESRIN to overcome some of the persistent problems with existing IPFs by proposing a scalable and dynamically re-configurable platform for the execution of IPF’s, or of any resources demanding processing facilities. Also, and for the intended users of ASB of great importance, ASB would focus on automating as many steps as possible to take information detailed in the Algorithm Theoretical Basis Document (ATBD) and the Detailed Process Model (DPM),both usually prepared by the principal investigators, to create workflows describing the product processor. The workflow would make it possible to deploy the processors on a cloud processing platforms transparent to users, but under their control.

## Key Objectives

To address the issues identified with the IPF development (anno 2014) a set of key objectives were defined to provide a dynamic, scalable multi-mission (automated) processing environment.

|  |  |  |
| --- | --- | --- |
| Key Objective | Rationale | Requirement |
| **Scalability** | EO products are huge and processing them requires great amounts of computing power, network and storage. | The infrastructure shall be able to scale and adapt to the users' needs in term of products generation and get rid of the classical infrastructure limitations. |
| **Automated Generation of Workflows** | Efforts to standardize the interfaces among the processors and the processes (ESA Generic IPF Interface Specifications) have been made but disparities remain between processor interfaces of different missions. | A generic interface for all the processors is needed to provide modularity and make it possible to share processes between different processing chains and missions. |
| **Generic Orchestration** | Current approaches are closely linked to the mission and the computing platforms used for the IPFs | A generic mechanism, adaptable to the specific orchestration baselines, is essential to run workflows from different missions into the same platform. |
| **IPF Evolution** | Traditional IPF’s are static and upgrading the workflow definitions or the processes are not easy tasks. | A simplified means to deploy new (versions of) processes and edit the workflows accordingly is therefore essential. |
| **Monitoring and Control** | IPF’s do not provide fine-grained control over the workflow execution | The solution provides advanced monitoring and control capabilities to overcome this limitation. |

## ASB Evolution

During the execution of the project the landscape for exploitation facilities rapidly evolved. A CCN was established enhance ASB and demonstrate how the ASB concept can accommodate new user demands.

Although targeting initially the ground segment facility development community, in particular those interested in the evolution of systematic data processing, SpaceApps through discussions with researchers and scientists became aware of the needs to support the development and validation of algorithm software including re-development post the start of mission operations.

Effectively, the algorithm developers (principal investigators) were looking for a bridge between their research/development process in a prototyping phase to the phase where their results are migrated to an operational status that would remove the need to port (re-implement) their software. The approach taken was to offer the principal investigators the means to import their algorithms (software) into ASB framework so that their algorithms were usable as processes in workflows for defining processors. This allows the principal investigators to finalise their verification and validation work using the same core services that would be used for the operational processing. This avoids the discontinuous approach where the work of the principal investigators is taken by IT specialists and ported to the operational environment. Something that was seen as undesirable by many principal investigators. Extending ASB to the principal investigators creates a seamless process from prototyping to execution on the operational processing platform.

This dual application of ASB is possible due to modular design and careful and controlled use of opens source solutions and OGC standards to create a platform and application agnostic solution. To provide this dual use required an extension to the ASB user interface offering a Development Interface for users to ingest and make available their own algorithms in the workflows.

Having established the means to allow developers to create new processors a further demand arose from the user community to execute on demand pre-defined processes. An additional requirement is that no knowledge of the application should be required other than being able to provide the correct inputs to the processor. This introduced a third user group, the Processor User.

Lastly, additional influences resulting from the continued development of exploitation platforms (Thematic, DIAS, Exploitation Platform Common Architecture) resulted in the introduction of BaaS (backend as a Service) to the ASB Framework making it possible to switch platforms or use multiple platforms to host processing of user algorithms.

By responding to the new user demands ASB has evolved to be a more mature framework able to offer a full coverage of processing capabilities based on a single framework core for systematic, unsupervised, automated processing and on-demand processing.

# TECHNICAL CHALLENGES

## Introduction

The ASB project set itself three key technical challenges:

* Take advantage and consider design influences from ESA activities and third party software;
* Create a Processing Platform and Application Agnostic Framework;
* Use the OGC Web Processing Services.

## Design Influences and Trade-offs

Analysis of the Generic IPF Interface Specifications, Reference: MMFI-GSEG-EOPG-TN-07-0003, Version 1.8, dated 3-Aug-2009, ATBDs and products such as the SMOS L1OP IPF and the PROBA-V n-Daily Compositor processor resulted in the specification of the needs for processing chains and data flow definitions, verification techniques and derived functional requirements that must be considered to develop a generic IPF platform.

Third-party software and services were assessed comparing the capabilities and offerings against the relevant ASB technical specifications. The results are documented in the ASB Third-Party Products Trade-Off Analysis Report. For ASB the products *Airflow* for orchestrating processor workflows, *Apache Mesos*, *Marathon* and *Zookeeper* for distributing and running *Docker* container images in a cloud‑based environment on‑the‑fly, and the *Logstash*, *Elasticsearch*, *Kibana* product stack for collecting, storing and visualizing the log traces generated by the various components were selected.

## Platform and Application Agnostic Framework

To be platform (the hosting platform for execution of the processing) and application (need to provide heterogeneous capabilities) agnostic the ASB Framework has been developed in such a manner that the location and nature of the actual execution environment(s) are hidden from all but two of the core components: the Task Manager and the Resource Manager. All the other components are unaware of where and how the processes are executed.

This makes it possible to use ASB in three modes.

1. "stand‑alone" mode where all the components as well as the processing chains are deployed and run on a single host. This setup is in particular convenient for developing and testing algorithms on small amounts of data.
2. "cluster" mode where the available processing resources are considered as fixed, with no possibility to scale up or down the cluster.
3. "cloud" mode, where the process execution environment is a Private or Public Cloud, where resources are scalable. The available processing resources, in the form of Virtual Machines, are automatically detected and used to deploy processes.

Figure 1, below, presents the different modes and the possible deployments configurations.



Figure 1 – ASB Execution Modes

The containerization technology, as implemented with the Docker software tools was selected because it provides a much more flexible and efficient means to dynamically deploy/un‑deploy software services, and to run these with good performances.

In particular, containers have the big advantage to run directly on top of the hosting system, sharing the kernel, low‑level libraries, as well as hardware resources such as CPUs, memory and disk. On the contrary, Virtual Machines run on top of a hypervisor, require a full Operating System to be installed in the virtual environment, and have reserved hardware resources.

Containerization is used both for developing the ASB Core Components (each component running in its own container), and for packaging the processes of the processing chains.

Figure 2 below presents the difference between Virtual Machines managed by a hypervisor, and Docker Containers, directly managed by the Host OS via the Docker Engine.

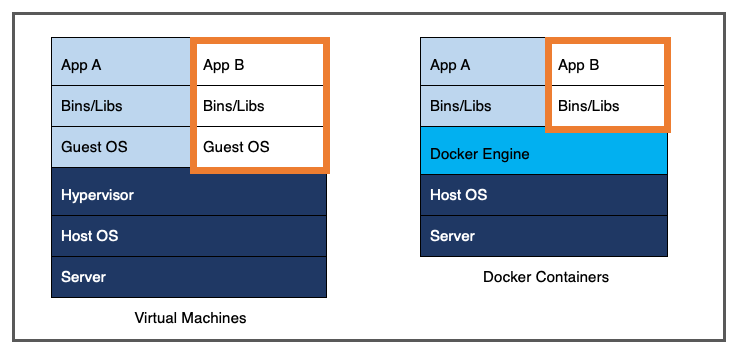


Figure 2 – Virtual Machines VS Docker Containers[[1]](#footnote-2)

Figure 3 in section 0 shows the different components that exist in the platform. Each of these components is packaged within a Docker container. A virtual network is configured in such a manner that the components behave as if they were connected on the same subnet, even if they are deployed on different hosts.

The Core Components must be installed in a static manner. The "Process" box located in "Virtual Machine(s)", however, represents process‑specific containers that are deployed and run dynamically, as it is explained in section 0, below.

## OGC WPS‑Based Interface to Processes

The OGC Web Processing Service (WPS) standard offers a simple method for finding, accessing, and using all kinds of calculation models. Being an OGC standard, it is mostly used in geospatial infrastructures however nothing in the specification prevents using it in other contexts.

The WPS interface standardizes the way processes and their inputs/outputs are described, how a client can request the execution of a process, and how the outputs form a process are handled. WPS uses standard HTTP and XML as a mechanism for describing processes and the data to be exchanged.

The main advantage of WPS is interoperability of network-enabled data processing. It hides the underlying processes specificities by exposing a common standard interface.

The ASB software components as well as the processes available for composing processing chains are packaged within Docker containers. In each of the process containers a WPS‑compliant service is deployed that acts as an adaptation layer for the actual executable process ("inner process"). The inner process typically applies a transformation or extraction algorithm on input data.

In the course of the project, a demonstrator has been prepared to execute the PROBA‑V n‑Daily Compositor processing chain. In this demonstrator, however, WPS‑compliant processing services have been configured to behave as proxies to processing services hosted and run in the PROBA‑V MEP environment. All the ASB framework components, including the Task Manager, are unaware of the fact that the actual processing is performed in a remote environment instead of within the process container.

# ASB FRAMEWORK

The ASB project (including CCN2) has realized a framework which is made of core components. Each component exposes its functions through a remotely callable API. This makes it possible to distribute the components on several (physical or virtual) hosts, should it be necessary. The ASB framework is built according to the microservices architecture (MSA) principles. In order to facilitate both the installation and the distribution of the components, these are deployed individually in Docker containers.

Three user-roles are identified:

* 1. Administrator, who has access to all components and who is able to manage the resources used by an instance of the framework.
  2. Developer, who can define, edit and delete workflows and import user provided algorithms and scripts
  3. Processor user, who is able to select and to run predefined processors.

Figure 3 depicts the different software components that belong to the platform. One can distinguish the following categories:

* + Web interfaces, the components that provide access to the ASB services
  + Controller Node with the "Core Components". They provide the different services of the platform.
  + Cloud Platform(s) where the resources are deployed in the cloud for specific processors: process images, process instances, and generated products.

## ASB Components



Figure 3 – ASB framework Logical Architecture

Table 1 - ASB Components

|  |
| --- |
| **Product Catalogue,** is the place where the metadata of all the products available in the system is stored. The products themselves are stored in the Cloud-based Persistent Storage.  **Knowledge Base,** the information necessary to generate the products. In particular, it stores the definition of knowledge elements such as data types, input and output parameters, processes and processors.  **Service Builder,** transforms product requests into Workflow Execution Orders by fetching the necessary definitions from the Knowledge Base, resolving the dependencies between processes and selecting the appropriate inputs.  **Orchestrator,** manages the execution of the processor workflows.  **Task Manager,** abstracts the execution of each task launched by the Orchestrator, by hiding the Cloud Environment where the task are executed. The Task Manager balances the load of the tasks stemming from the Orchestrator according to the resources deployed in the Cloud Environment.  **Resource Manager,** has the responsibility to deploy and un‑deploy resources in the Cloud. The component maintains a registry of all the processes that can be deployed.  **Process Images Registry,** is used to store the image of the containers in which the processing services are pre‑installed. Images are available for download, instantiation and execution. Each of these images may be instantiated and started multiple times within the same VM or within several VMs.  **Quotation Service,** returns the estimation of IT resource for a given process and specific inputs.  **Credit Service,** holds the credit values for each end-user of the system. These values are used by the Service Builder in order to control access to the product generation. The Credit Service has the ability to convert any IT resource value or estimation into credits.  **Log Manager,** has the responsibility to collect and store the log traces issued by the other ASB components. It also gives the administrator the possibility to browse and query these data in a highly configurable Web‑based interface.  **Processing Service,** is the execution unit of the atomic tasks of a Workflow Execution Order. It wraps a pre-existing processor behind an OGC WPS interface, so that it can process given inputs into outputs in a standardized way. A *Processing Service* can also be used to get a resources consumption estimation of a specific process from the Quotation Service.  **Import Service,** provides the means to define a user component (user defined algorithms, software, scripts). New data types may need to be defined as well as input and output parameters. Software and library dependencies are defined. The Import Service generated wrapper template is edited to include the calls to the user component(s) and is then ingested and dockerized. The Import Service populates the Knowledge Base. The processes are then registered in the Process Imager Registry. An imported user component is visible and usable as a process in the workflow editor.  **Administration Interface,** exposes a "master" Web page that presents a structured table of content. The table of content contains hyperlinks that direct the user to the various administration interfaces implemented in the ASB Core Components.  **Developer Interface,** provides access to the Import Service, Workflow Editor, Knowledge Base, the Product Catalogue and the ability to run the (newly) created processors to generate products. This interface also allows following the progress of the products being built and to access information about the products that have been generated in the past.  **Processing Service Interface,** provides access to pre-existing processor so that user can provide the process inputs and execute a processor to create products.  **Backend Service**, installed as a BaaS, the BaaS is used by the Resource Manager to create the Process Imagery Registry on the host platform (stand-alone, server or cloud) |

## Generic Modelling Scheme and Associated Tools

The Knowledge Base component introduced above is responsible for storing and guaranteeing the validity of the definition of the elements present in a typical processing modelling scheme. No mission or process‑specific characteristics have been included in the schema. It has been designed to be generic and extendable.

In particular, it registers the definition of the following elements:

* (Semantic) **Data Types** are implemented as ontology concepts. They are used to assign a sense to any input and output parameter.
* **Input Parameters** and **Output Parameters** are instances of data types that may be attached to specific processes.
* **Processes** are atomic units of treatment on data that generate one or many outputs from one or many inputs. Process properties include a name, a version, and a description. Each process is linked to its input and output parameters, and to the related mission (see below). The execution of individual processes cannot be requested in ASB. Instead, processes must be combined within processor workflows (see below).
* **Processors** combine one or more processes to form a workflow. As such, they allow the production of a certain level of product from raw data or lesser level products, or the extraction of features from input data. In addition to the workflow definition, processor properties include a name, a version, and a description. Processors are also be related to a specific mission. The execution of processors may be requested, either through the end‑user interface or a WPS‑based front‑end interface.
* **Missions** are used to give processes and processors a context. In particular, it facilitates the grouping and filtering of the knowledge elements on the user interfaces.

## Built-in-Operations

To assist developers some additional features were developed:

* Simple notification by email can be inserted in the workflow
* Splitter Function which executes a segment of the workflow from the split statement to the join statement. The intervening workflow is parallelised creating as many jobs as there are items in the input list. Typically useful for executing multiple scenes such as would be needed for reprocessing a time series or processing large areas creating multiple products.

# INSTANTIATING A PROCESSING FACILITY USING ASB

Once the mode (stand-alone, server/cluster or cloud) has been selected the components of the framework can be installed. For a number of cloud providers the BaaS is already available. This step requires IT support but the following steps require little IT knowledge to establish the facility.

The algorithms to be used in the workflows of the processing facility need to be imported and prepared using the Import Service. Following normal practice algorithm developers prepare concepts offline, implement and test their algorithms. The first step of the Import Service is the Algorithm Packaging and Import as shown in Figure 4 resulting in a docker image containing the algorithm.

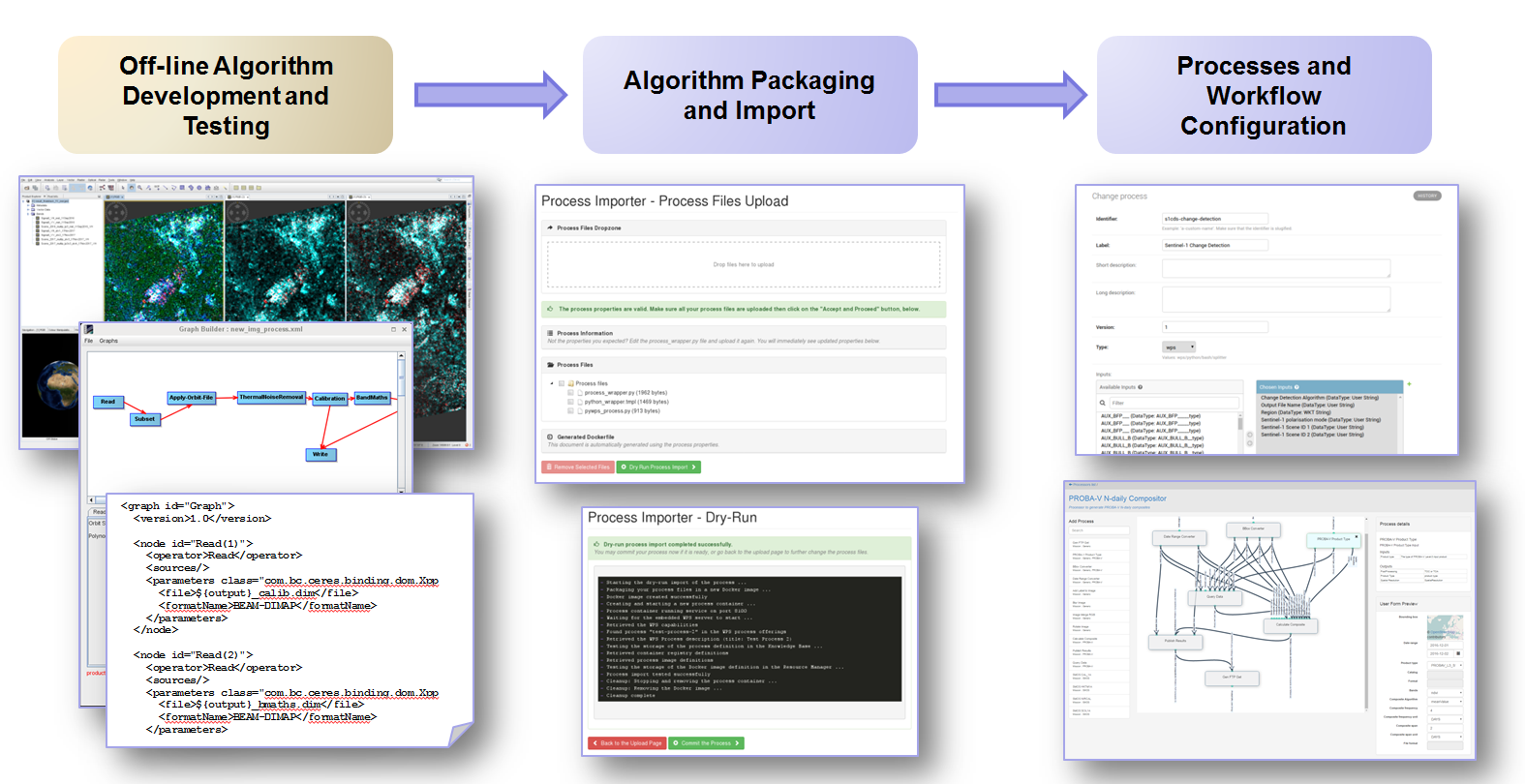


Figure 4 - Setup Steps: Algorithm Import and Workflow Configuration

The Import Service is repeated for as many modules the user wishes to import. In the example shown in Figure 5 seven import steps have been completed to make seven processes available in the ASB Processor Editor.

The Processor Workflow Editor allows, by connecting data flows, dependencies to be defined which are used by the ASB Workflow Engine for triggering the execution of the tasks configured in the workflow execution orders depending on their inter‑dependencies and the availability of their inputs..

Once the workflow is completed the resulting processor can be selected to be run. This invokes the Service Builder component that uses the information stored in the Knowledge Base to:

* Build a workflow execution order and transfer this to the Orchestrator component.
* Collect and store the workflow execution results.
* Generate execution reports and provide access to the generated data files.

Note, no further programming is required to run a processor. Forms are automatically generated based on the workflow definitions whereby the unconnected input parameters are presented in a form and the user requested to supply values. The Web components used to implement the input fields depend on the actual input data types (e.g. bounding box, date range, enumeration, and numeric range).

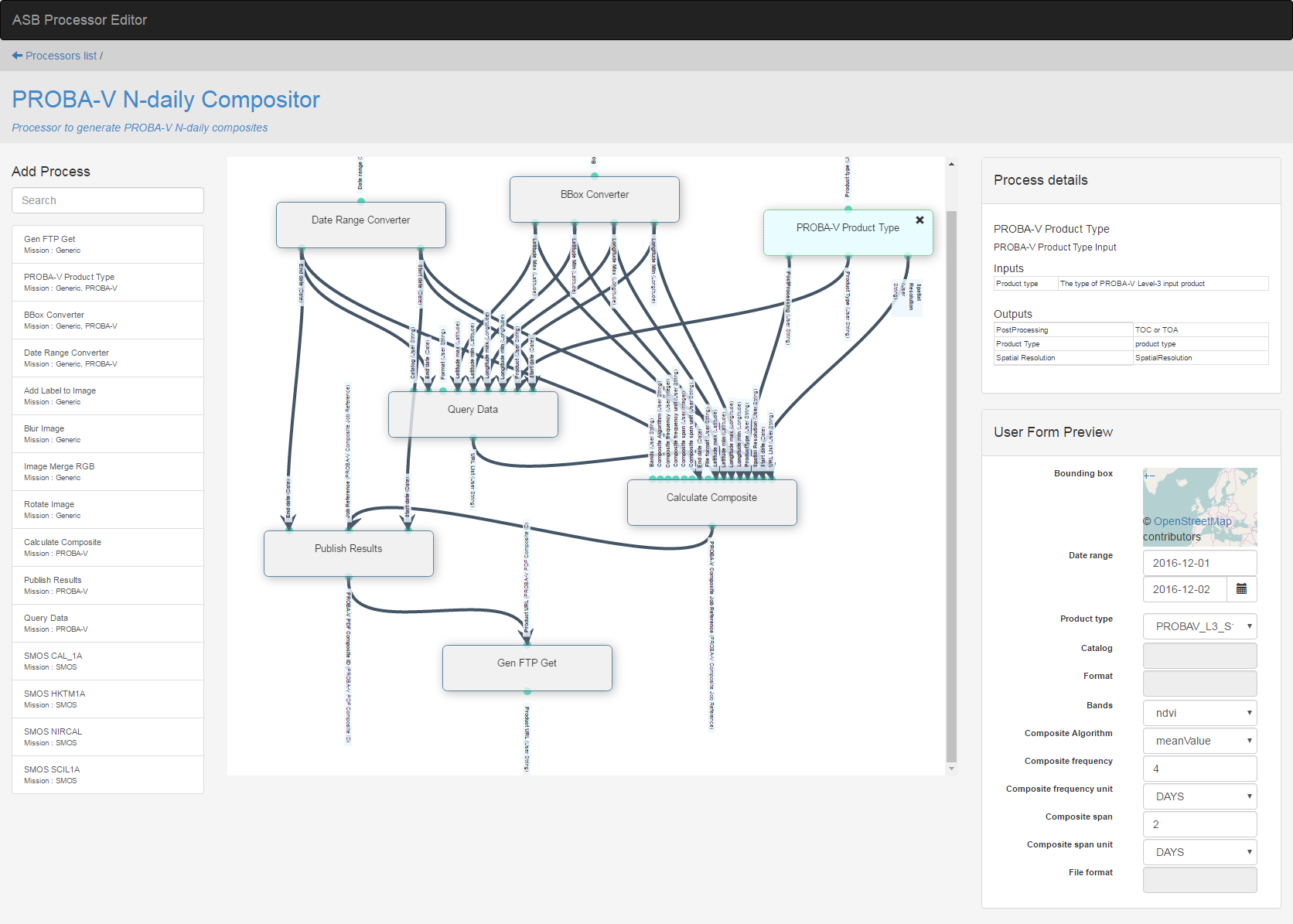


Figure 5 – ASB Processor Workflow Editor

Figure 6 is an example of a automatically generated form for the PROBA‑V n‑Daily Compositor processor. It lists all the input parameters that are configured on the different processes of the processor and that do not receive a value via the output parameter of a preceding process. Figure 7 shows the resulting products report.

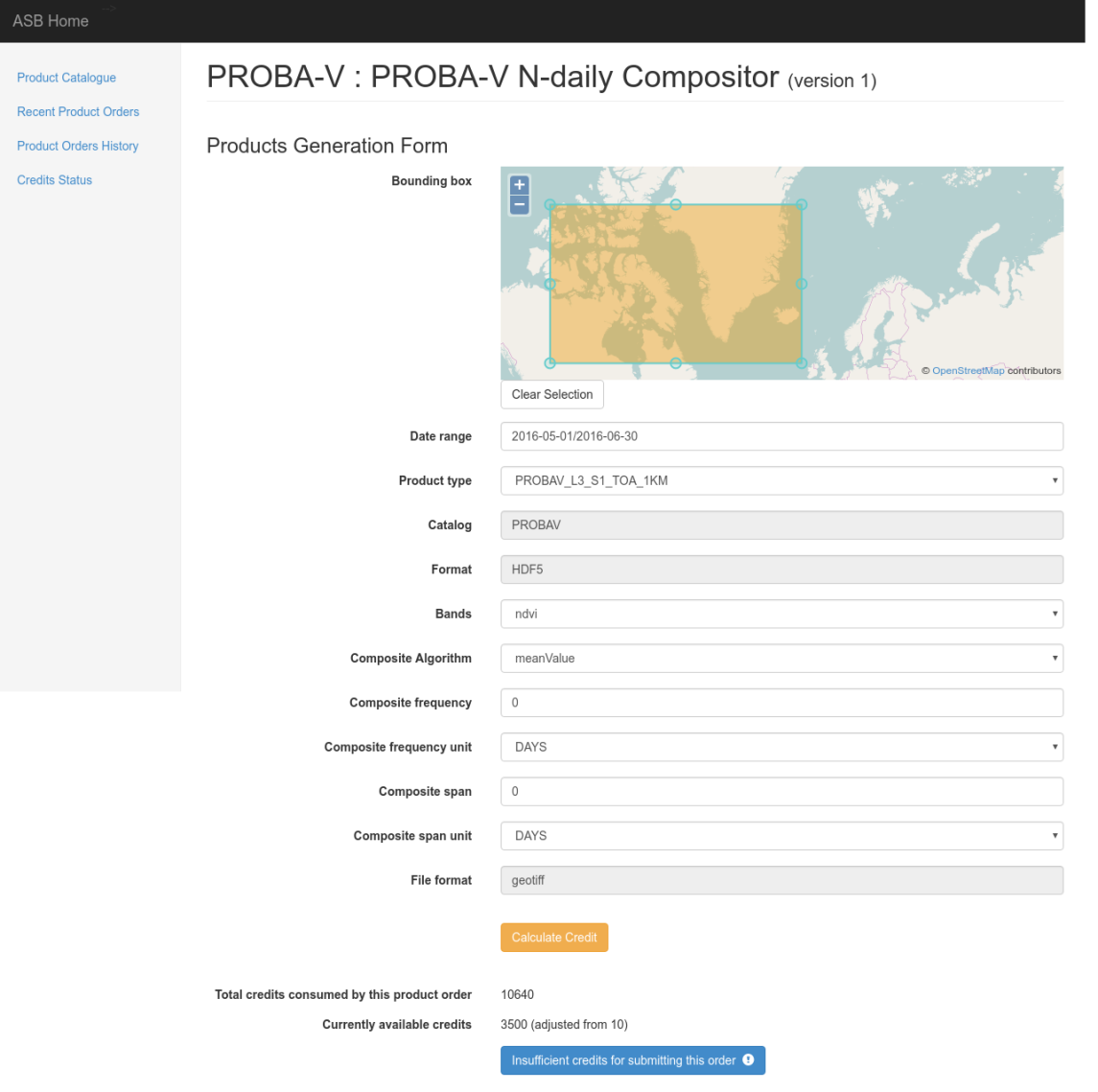


Figure 6 – PROBA‑V n‑Daily Compositor Parameterization Form

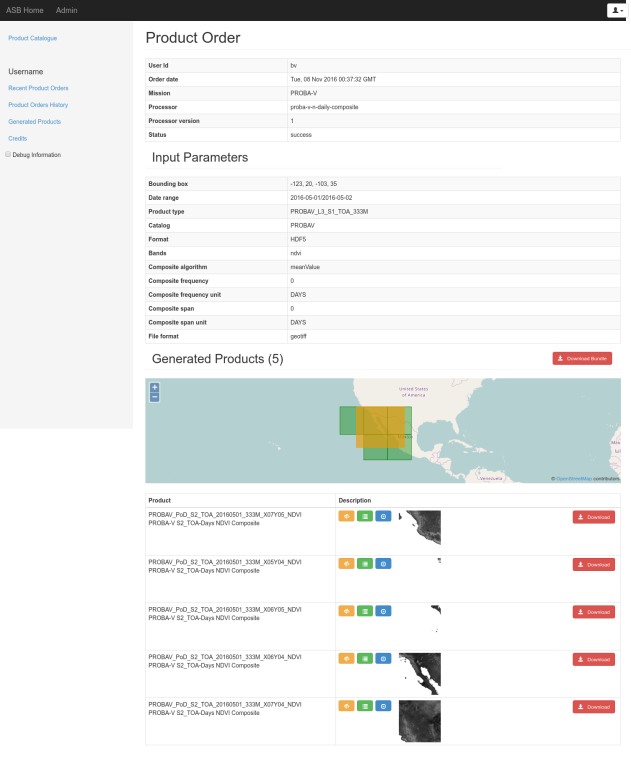


Figure 7 – Product Order Execution Report

# DEMONSTRATIONS AND USER EVALUATION

Activities have been performed to demonstrate the capabilities of the ASB framework and to collect feedback.

In the PROBA‑V Mission Exploitation Platform (MEP) Third‑Party Services (TPS) project the ASB framework was extended to provide an Orchestrator Service. The research partners have been involved to create operational services and provide an evaluation of the ASB Framework.

A number of presentations and demonstrations have been organized:

* With researchers at Université catholique de Louvain (UCL), developers and maintainers of a use case in the PROBA‑V MEP TPS project.
* With researchers at the Wageningen University and Research Center (WUR), developers and maintainers of use cases in the PROBA‑V MEP TPS project.
* With SMOS PDGS/IPF stakeholders at ESA/ESRIN, as part of the ASB project Final Presentation.
* PROBA-V Symposium Ostend

In addition to the above face‑to‑face meetings, the MEP‑TPS project partners have also contributed by filling-in an online questionnaire.

As part of CCN2 two additional stakeholder evaluations were held:

* With the developers at LIST of the Proba-V Toolbox from the MEP-TPS project;
* With the EOPEN project (<https://eopen-project.eu>), users of the ASB Framework in the H2020.

The overall conclusion of potential users of ASB were very positive, although effort was needed to dispel some misconceptions arising from experiences with other platforms.

**Misconception 1: The application teams thought they would have to do a lot of programming work to use the platform. They are not IT oriented and were considering very basic approaches for processing their data applying AI solutions.**

Feedback-1: To import and run my algorithm seems very simple. I am not aware of what I have to code yet (insert in the wrapper to call the user application) but it looks to be very little effort. This is a clearly a big advantage as we get a fully operational algorithm with low effort from our side. We will not have to develop a processing service environment.

Feedback-2: Building a workflow with the editor using our imported algorithms is user friendly and looks easy. Email notifications and splitter task are useful.

Feedback-3: This simplifies our task considerably and provides a better solution than we anticipated.

**Misconception 2: These systems (ASB) seem to be good. The splitter task is OK but can I use solutions such as Hadoop?**

Feedback: We were not aware that the building of a Docker solution means we can basically use any programming language and also execute using a Hadoop environment if we want to.

**Library dependency feedback: Can we have other libraries?**

Answer from SpaceApps: Yes, but they would need to be included into the platform by SpaceApps.

**The live demonstration of the scenario where a developer user imports and executes as part of a workflow a Python script which in turn runs an R script has shown how flexible the ASB framework is for embedding algorithms implemented in various languages.**

Feedback: The Web-based user interface seems user friendly.

Feedback: We use Jupyter notebooks but have to use different notebooks for Python and R. Very interesting is the ability to chain processes implemented in different languages (e.g. Python, R) and using different libraries and toolboxes (e.g. LIST TSA, SNAP). We would need a more complete demonstrator where processes implemented in different languages and using different libraries/toolboxes are used in the same workflow.

**Test code implemented in R applies on raster data located in the PROBA-V MEP environment. Because of this, the code must be deployed and executed within the MEP. It is difficult to test the code outside the MEP. ASB provides a means to configure a processor locally and automate the deployment and the execution of the code in the MEP.**

Feedback: ASB could prove to be very useful to build operational systems.

# DISSEMINATION

The results of the ASB project have been disseminated as indicated in the following table.

|  |  |
| --- | --- |
| Action/Event | Description |
| ASB Project Description on ESA's RSS Join & Share Web Portal | A project description page has been published on ESA's RSS Join & Share Web portal. This summarizes the context of the project, the objectives and the expected results. Wiki pages restricted to the project members include the project team structure, the meetings material, the deliverables (CIDL) and the progress reports:  <https://wiki.services.eoportal.org/tiki-index.php?page=ASB> |
| ASB Flyer for the BiDS'14 | Flyer has been distributed in particular at the first Conference on Big Data from Space (BiDS'14) organized on November 12th to 14th, 2014, at ESRIN:  <http://esaconferencebureau.com/2014-events/BigDatafromSpace/introduction> |
| ASB Poster and Paper for the BiDS'16 | A poster presenting the ASB project and platform that has been displayed at the Big Data from Space Conference in March 2016 (BiDS'16, <http://congrexprojects.com/2016-events/16m05/introduction>).  The accompanying paper may be found in the conference proceedings freely available at the following address: <http://publications.jrc.ec.europa.eu/repository/handle/JRC100655>. |
| ASB Paper for the BiDS'17 | A paper was presented on the achievements on the ASB project at the Big Data from Space Conference in Toulouse in November 2017, BiDS'17, <http://www.bigdatafromspace2017.org/>  *ASB –A Platform and Application Agnostic Solution for Implementing Complex Processing Chains Over Globally Distributed Processing And Data Resources*. Bernard Valentin 1, Matthieu Melcot 1, Leslie Gale 1,Philippe Mougnaud 2, Michele Iapaolo 2,1 Space Applications Services NV/SA, Zaventem, Belgium,2 European Space Agency / ESRIN,Frascati, Italy |

# CONCLUSIONS

The ASB project has met and exceeded the goals set out in the original SoW. Responding to the demands from users and other projects ASB has evolved rapidly from its original concept to just serve Instrument Processing Facilities without compromising the concepts established at the start of the ASB project where the IPF was the primary goal.

ASB has the capability to perform the tasks of a processing facility (systematic, unsupervised, automated processing) as well as being able to provide on-demand (user control) processing.

ASB supports the integration and testing of user defined modules requiring no IT support or the intervention of the ASB Framework support team through the use of the ASB Import Service. The IMPORT Service is an automated tool that removes all programming effort from the user who wants to include their own algorithms in the framework for use in defining processes in workflows defining processors for EO products.

ASB has coined the phrase *“Processing anywhere made easy”.* This has been achieved by the introduction of a BaaS (Backend as a Service) that implements all cloud platform dependencies. ASB has been demonstrated using CloudSigma, OTC from T-Systems, Hetzner, Proba-V MEP and the private cloud of SpaceApps. Swapping from one service provider to another is simply a question of selecting the desired platform backend. In fact some users were surprised to realise that for ASB a DIAS is just another platform. The ASB Framework places no limitations of the platform choice for hosting the processing.

ASB could be used as a proxy to all DIAS platforms offering a simple solution for users to compare DIAS performance and select the DIAS the best fits their needs.

ASB covers a development lifecycle that includes the algorithm development making it suitable for implementing a service to support Third Party Services exploiting existing data and processing capabilities.

The ASB project has delivered a mature framework that significantly reduces the effort of developers to create an operational processing service lowering barriers for use (low code implementation and integration effort) and reducing the usually needed high investment to establish an operational processing facility.

The ASB Framework has been used by ESA in the MEP-TPS project and is already in use in commercial services for automated processing for change detection based on Sentinel 1 data.

The ASB Framework is ready for use to realise complex Processing Services.

# FUTURE POTENTIAL

The ASB Framework can provide the technology basis for implementing the services of the interface layers (Information, EO and Platform) between users and services.

In the case of the Sentinels and Copernicus, ASB could prove to be a valuable means to build vertical stacks through the EO Ecosystem layers as illustrated in Figure 8,



Figure 8 – ASB as the technology/services basis for  
platform and services for Exploitation Platforms

This approach is being used to support change detection services (a small scale TEP) where the application platform is a GIS system. It will also be used in the EC H2020 EOPEN project (see figure 9) where the ASB Framework will be used to implement the processing platform used to create and provide processing services for Flood Detection and Response, Food Security and Climate Change impact on land use in northern Finland.



Figure 9 - EOPEN application of the ASB Framework

Knowledge gained in developing ASB can also provide insights into ongoing ESA related activities and be used for rapid prototyping and demonstrations.

An initial review has been made of the Exploitation Platform Open Architecture Common Architecture Objectives and they seem to be aligned with the implemented capabilities of ASB.

Table 2 – Coverage of Common Architecture Objectives

|  |  |
| --- | --- |
| Common Architecture Objective | Addressed by ASB |
| Architecture based on standard interfaces (OGC based) | Yes |
| Middleware for: |  |
| Federated identity management | No |
| Processing & chaining (Scheduling/Orchestration, Workflow) | Yes |
| Data provisioning, management & accounting | Partially |
| Operations of a standard interoperable service for Sentinels and ESA missions as 3rd party service on top of DIAS. | Yes |
| Support to mission providers to integrate their mission hosted processing/data analytics services into the network of platforms and qualification of their services. | Yes |

Future developments to be considered are:

* Including popular Toolboxes as libraries for use in workflows;
* Implementing interfaces to platform APIs (DIAS, Planet etc);
* Include data discovery and data harvesting capabilities.

ASB can be used in any programme where a need exists to create and distribute processing services:

* Ground Segments,
* Instrument Processing facilities and re-processing,
* Exploitation Platforms (Mission, Thematic, Regional..)
* IoT – Cloud-Edge processing,
* Dynamic re-configurations of processing services;
* Global orchestration of processing;
* Benchmarking and performance comparisons;
* Multiple platforms operations;
* Business workflows;
* GIS systems and services.

1. Source: <https://www.docker.com/whatisdocker/> [↑](#footnote-ref-2)