Architecture Description of SHIP Conditions Database

Version 3.0

Authors:

Beza G. Tassew (Architect)

Daniel t. Fratte (Architect)

Collaborators:

Ani Megerdoumian (Project manager)

Dimas Satria (Designer, Test Engineer)

Giovanni De Almeida Calheiros (Quality Manager, Developer)

Konstantinos Karmas (Configuration Manager, Developer)

Pranav Bhatnagar (Team Leader, SCRUM Master)

August, 2018

Beza G. Tassew

Contents

1. Introduction

This document describes the architecture of SHiP Conditions Database that is designed and developed by the CERN team of PDEng Software Technology Trainees at Eindhoven University of Technology.

Physicists at CERN aim to discover the basic elements that make up the universe and how they came into being. For this purpose, they built the Large Hadron Collider (LHC, the largest particle accelerator in the world) for conducting their experiments. Particle accelerators, typically, guide proton beams against one another so that the particles collide. SHiP (Search for Hidden Particles) is a new experiment facility to be built on the smaller ring of the LHC (called Super Proton Synchrotron, SPS). Its purpose is to discover the weakly interacting particles which may otherwise not be discovered by the larger and more general purpose experimental facilities.

Apart from data regarding proton collisions (event data), data regarding the state of the machines conducting the experiment (detectors) is also essential for making accurate interpretations of the event data. The challenge of storing the status of detectors, also called conditions, arises from the nature of the data in addition to the non-standardization of conditions database design and technologies. The main goal of this project was to explore new ways of storing conditions data specifically using NoSQL technologies.

The solution provided includes a database design to serve this purpose and a proof of concept to demonstrate how this can be realized. This will aid the SHiP project management in selecting the technology that best suits their needs in the future. Furthermore, if they decide to use NoSQL technology it will serve as a sound basis for a full implementation.

1.1. Identifying information

The SHiP conditions database architecture describes the design of the conditions database model as well as the interfaces of communication.

The system of interest described here is a NoSQL approach to the SHiP Conditions database, which includes the modelling of the conditions data according to the SHiP experiment requirements and the interfaces required to store and retrieve conditions data.

2. Stakeholders and concerns

2.1. Stakeholders

The Key Stakeholders of this system are:

- SHiP Management: This stakeholder represents the owners of the SHiP conditions Database, who are responsible for selecting, outsourcing and/or approving a solution.
- Developer at CERN: represents the developers who will be responsible for the maintenance and development of the conditions database system and integrate the solution into the already existing experiment framework.
- SHiP Sub Detector Experts: These experts are the people in charge of each sub detector that collectively make up the SHiP detector.
- Physicist Doing Analysis: This represents the main users of the system. Physicist who will perform event reconstruction and analysis at CERN as well as the various locations outside of CERN.
- Developers of the System: This represents the OOTI-2017 CERN team who will be in charge of developing the conditions database system.

2.2. Stakeholders and Concerns

The main purpose of a conditions database is to supply conditions for event reconstruction and analysis by recording conditions of the sub detectors at certain intervals of time. The main use case of the conditions database therefore includes reconstruction of events for a good quality of analysis.

The concerns-stakeholders relationship matrix maps every key stakeholder of the system to their concerns to the system. Different stakeholders may have same concerns with the other stakeholders. Figure 1 shows the concerns-stakeholders relationship matrix of the system.

Source \ Target	Developer at SHiP-CERN	Developers of the System	Physicists Doing Analysis	SHiP Management	SHiP Sub detector Experts
Evaluate NoSQL Technology for use in Conditions Database for SHiP		↑		↑	
Extend Fairship to use Conditions Database	↑				
Have a better conditions storage based on other LHC experiments		↑		↑	
Have a quality event analysis			↑	↑	
Maintain the System	↑				
Retrieve Subdetector Conditions					↑
Store Calibration and Alignment conditions					↑
Support Storage of Each Subdetector Conditions					↑
Understand Conditions Data Domain		↑			
Understand Conditions Domain		↑			

Figure 1. Concerns-Stakeholders relationship matrix

3. Architectural Views

3.1 Conditions Domain Model View

3.1.1 Overview

This view describes the conditions domain model that includes the devices and processes that create conditions data and the data that is required from the conditions database.

3.1.2 Concerns and stakeholders

This view is aimed at creating a common understanding of conditions and the relevant experiment domain knowledge. This view is associated with the stakeholders and their concerns as described in table 1.

Table 1. Concerns and stakeholders for conditions domain model view

Concern	**Stakeholder**
Understand the Condition Domain	Developers of the system
Select a better technology based on other LHC Experiments	SHiP management

Figure 2 shows the conditions domain model.

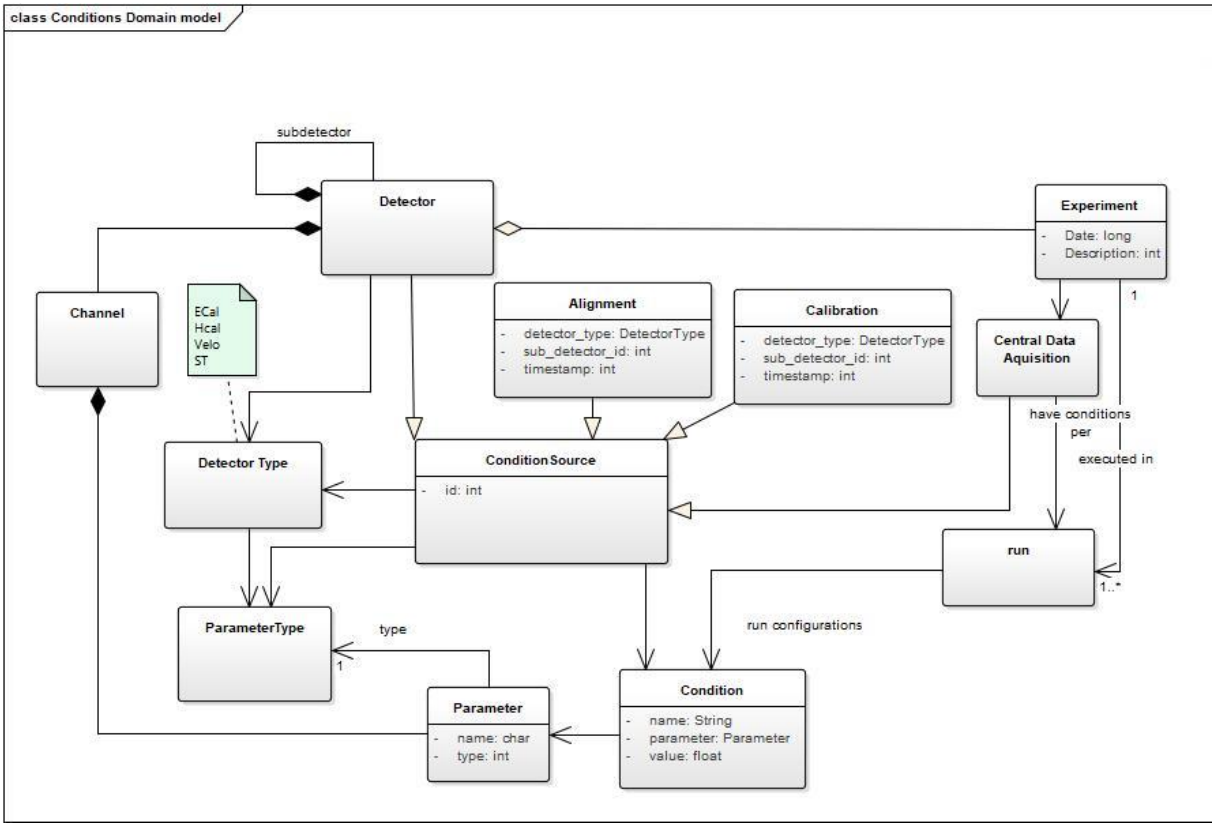

Figure 2. Conditions domain model view

The conditions domain model view in Figure 2 can be represented in a set of statements. Table 2 shows the statements about the conditions domain model of the system.

Table 2. The statements about the conditions domain model

S-1	Conditions can come from ● The sub detector measurements like voltage, temp… (conditions during a collision) ● Alignment ● Calibration ● Central Data Acquisition (online subsystem)
S-2	There are one or more detectors under the different sub detectors ● e.g. Under Muon sub detector there might be a list of detectors that have conditions to be recorded
S-3	A sub detector has multiple channels
S-4	Parameters are recorded per each channel
S-5	A condition can be described by multiple parameters
S-6	A parameter may have a standard unit of measurement
S-7	Alignment can be done for a specific sub detector (e.g. Alignment for Muon-detector-1)
S-8	Alignment can be done for a set of sub detectors (e.g. Muon Alignment)
S-9	Calibration can be done for a specific sub detector
S-10	Calibration can be done for a set of sub detectors
S-11	Calibration and Alignment lead to versioning of the data
S-12	Conditions database should provide a snapshot of the conditions data valid at a given point in time. This snapshot of the data is identified by a Tag
S-13	A tag can be a sub detector name, run id, calibration run identifier, alignment run identifier

3.1. Conditions Data Domain Model View

3.1.1. Overview

This view describes the entities that exist in conditions data. It helps to extract the condition data domain from the conditions domain excluding domain entities that are related to conditions but should not be represented as conditions data.

3.1.2. Concerns and stakeholders

This concerns addressed by this view and the stakeholders associated are shown in Table 3.

Table 3. Concerns and stakeholders for conditions data domain model view

Concern	**Stakeholders**
Understand conditions data domain	Developers of the system
Maintain the system	Developer at SHiP-CERN
Support storage of each sub detector conditions	Sub detector experts

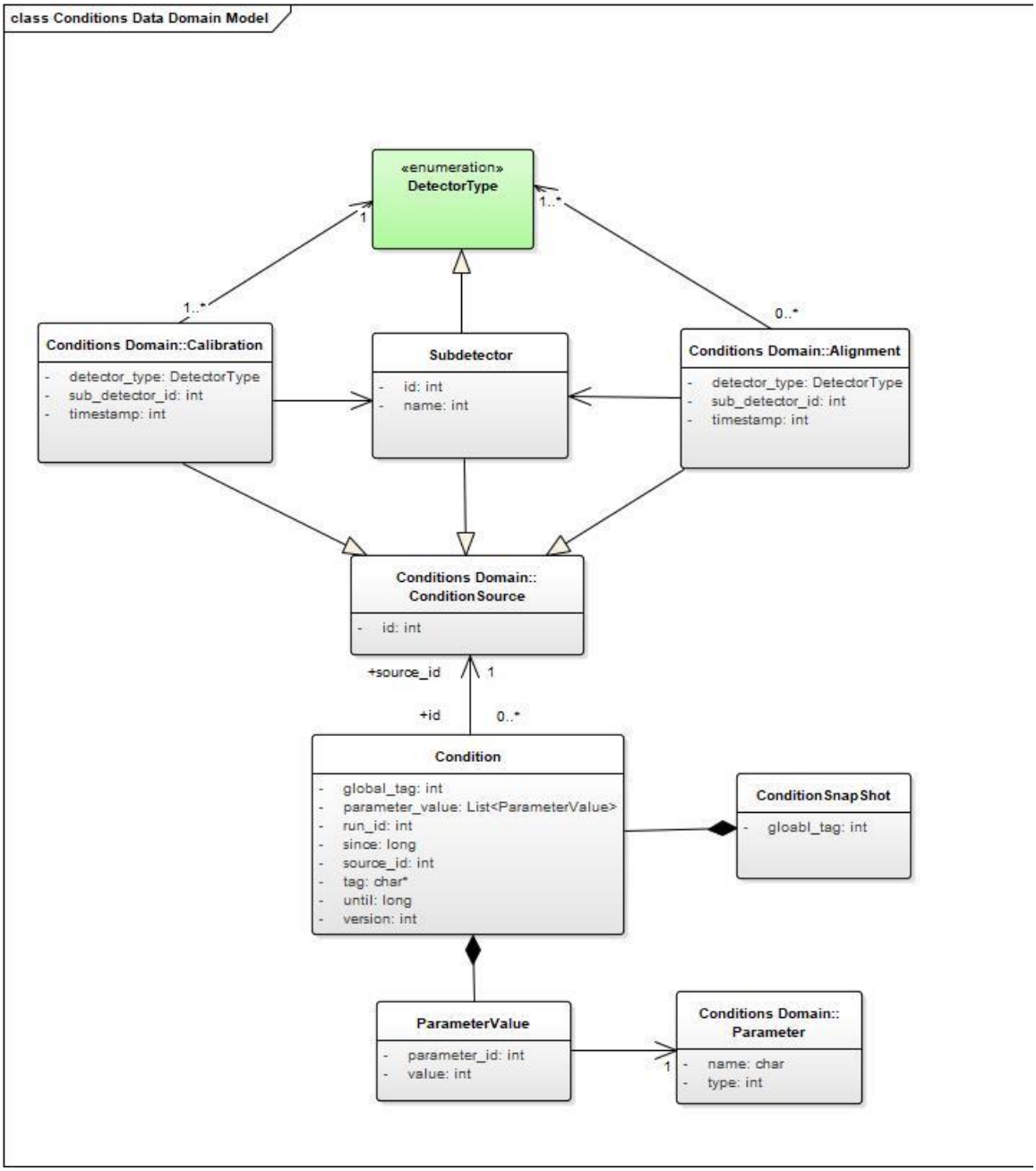

Figure 3. Conditions Data Domain Model

3.2. Conditions Database Requirement Traceability View

3.2.1. Overview

This view describes the basic functionalities required by the client and to be developed by the developers. Each Functional Requirement can be traced back to at least to one Stakeholder concern

3.2.2. Concerns and stakeholders

The concerns addressed by this view and the stakeholders they are associated are shown in Table 4.

Table 4. Concerns and stakeholders for conditions database requirement traceability view

Concern	**Stakeholders**
Understand conditions domain	Developers of the system
Have a better conditions storage based on other LHC experiments	SHiP management
Support storage of each sub detector conditions	Sub detector experts
Retrieve sub detector conditions	Sub detector experts

This view shows how the requirements of the system link to the concerns of the different stakeholder concerns. It is used for reasoning with respect to requirement prioritization and tradeoffs.

	Evaluate NoSQL Technology	Extend Fairship to use Conditions Database	Have a better conditions storage based on oth	Have a quality event analysis	Maintain the System	Retrieve Subdetector Conditions	Store Calibration and Alignment conditions	Support Storage of Each Subdetector Conditio	Understand Conditions Data Domain	Understand Conditions Domain	UserRequirement1
Comparative Study of Conditions Database	↑		↑						↑	↑	
Deliver Design & Implementation Documentation					↑						
Design Conditions Database Model									↑		
Muon Subdetector Requirements								↑			
Read Conditions byTag						↑					
Read Conditions Data				↑		↑					
Read Conditions Data from FairSHiP		↑									
Tag Condition set				↑		↑					
Version Conditions						↑					
Write Conditions Data							↑	↑			
Write Conditions Data from FairSHiP		↑									

Figure 4. The requirement - view relationship matrix

3.3. Use Cases View

3.3.1. Overview

This view describes the main use cases of the system. It is used to determine what functionalities the system should have in order to fulfill a certain requirement.

3.3.2. Concerns and stakeholders

The main stakeholders associated with this system are the developers of the system.

3.3.3. Use case View

The use case view describes the main use cases of the system and the relationships to processes or stakeholders that use them.

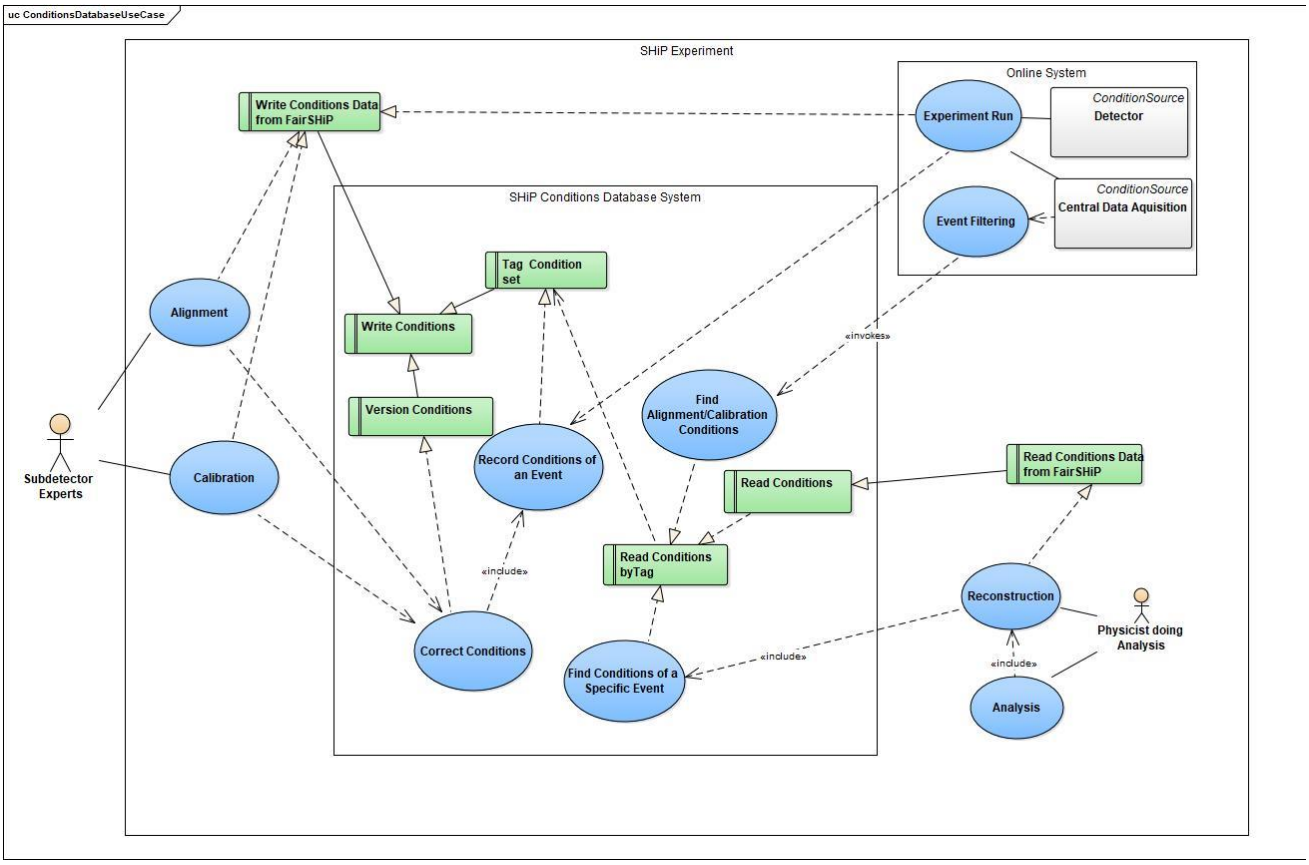

Figure 5. Conditions database use cases

3.4. Component View

This view describes the deployment setup of the solution. It shows the parts of the conditions database system that should be implemented in order to be able to achieve the functionalities required.

This view aims to define the sub modules of the system as well as interfaces and communication between those. This also helps the future developers of the system to be able to understand the different parts of the system.

3.4.1. Concerns and Stakeholders

The concerns addressed by this view and the associated stakeholders are shown below in Table 5.

Table 5. Concerns and stakeholders for component view

Concern	**Stakeholders**
Deliver design & implementation documentation	Developers of the system
Maintain the system	Developer at CERN

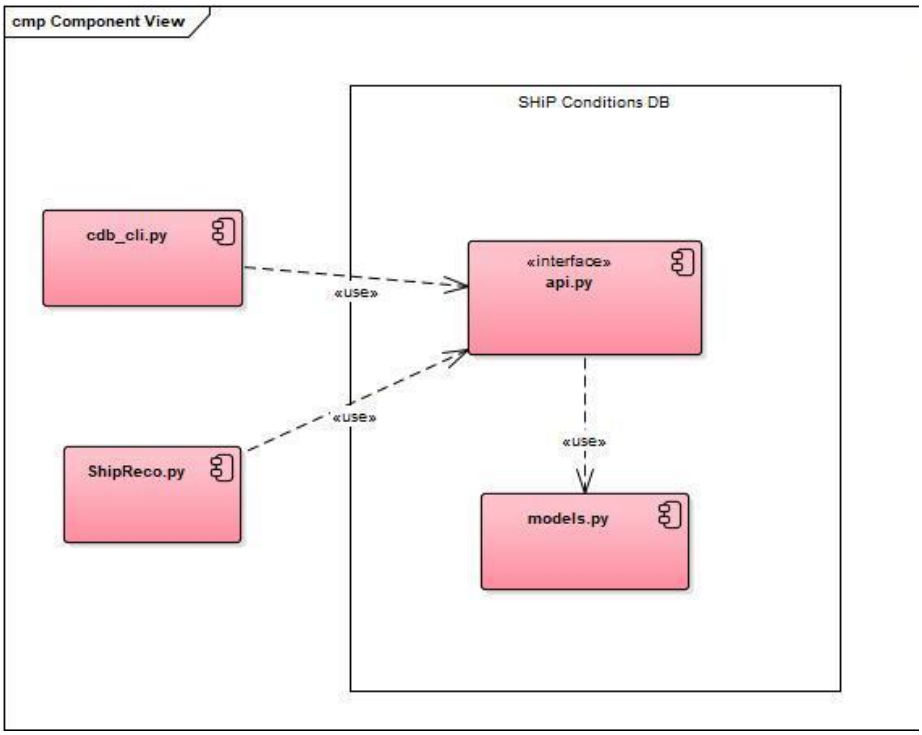

Figure 7. Component view

3.5. Deployment View

3.5.1. Overview

This view describes how the conditions database system will be used . This is aimed towards future maintainers of the system in order to be able to deploy the software and use in their company setup.

3.5.2. Concerns and Stakeholders

The concerns addressed by this viewpoint and the associated stakeholders are shown in Table 6.

Table 6. Concerns and stakeholders for deployment view

Concern	**Stakeholder**
Deliver Design & Implementation Documentation	Developers of the system
Maintain the System	Developer at CERN

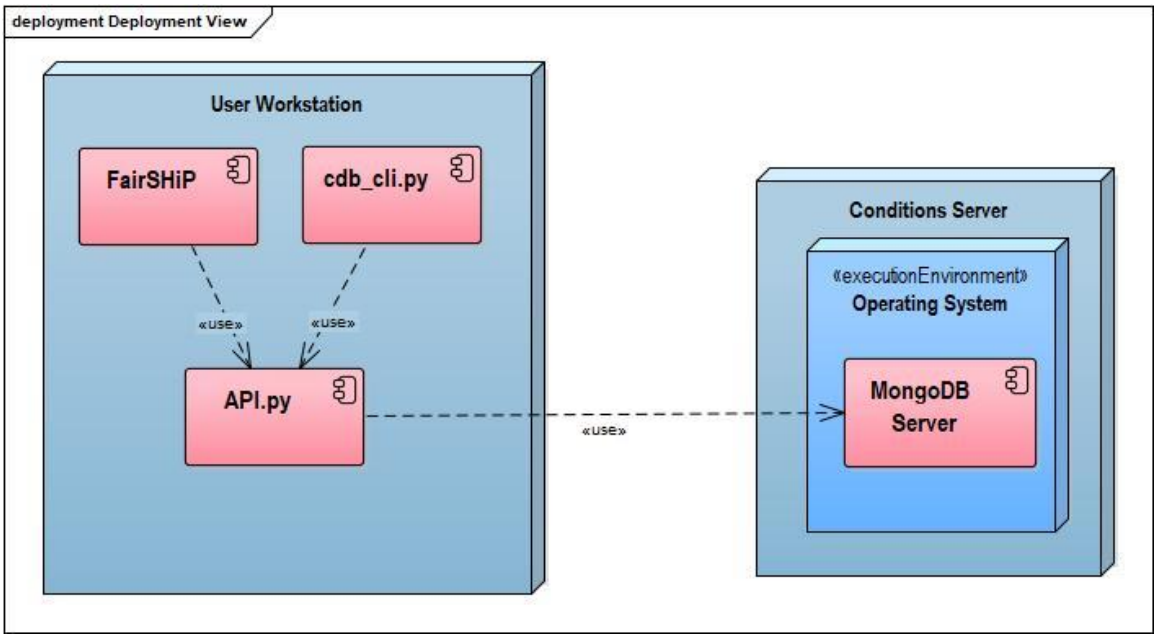

Figure 8. Deployment view

A Architecture decisions and rationale

A.1 Decisions

- Database Technology Selection: MongoDB
    - Rationale: easy to implement, good learning curve for development and maintenance, good performance with complex data sets and complex queries, already shown to have high performance in other CERN experiments(CMS), supports indexing
    - Decision Owner: The team
    - Alternatives: Cassandra, Riak, MySQL

- Programing language Selection: Python 2.7
    - Rationale: easy to implement, low learning curve for development and maintenance, compatibility with other customer software, fast and easy integration to the chosen database technology.
    - Decision Owner: The team
    - Alternatives: C++

- General design
    - The solution was approached by thinking in terms of a NoSQL database. The database can be accessed by the API that MongoDB provides, however, a more abstract layer was designed in order to provide a cleaner interface to the data. Therefore, a python API layer was implemented that interacts with the database service. Moreover, with the aim of handling the data we created a command line interface(CLI) that consumes the services that the SHiP API provides. This intends to be a simple example of this service consumption with the ultimate goal of triggering data queries from within a different client e.g. FairSHiP.

    - Rationale: it is a classic client-server approach, easy to design, implement and extend with the given resources for the project.

    - Decision Owner: The architects (Beza and Daniel)
    - Alternatives: Monolithic script that performs the roles of data access layer, user interface, and domain logic.

- Specific Performance Requirements (for the SHiP Project) are not covered by the architecture. These include
    - write performance
        - Rationale: The client did not want to consider write performances as these can be improved by adding more capacity to their infrastructure and also due to the fact that in the coming years the speed and capacity of discs was only going to increase.

        - Decision Owner: SHiP Management

- IOV implementation – since and until fields in conditions. These two are properties of a Condition, they have accuracy up to nanoseconds and they are used in order to retrieve all the conditions at some specific time point i.e. get a snapshot of the system.

- Data Domain Model to NoSQL Document Model Transformations
    - Many to many relations – Separate Documents
    - One to many relations – Child embedded in parent