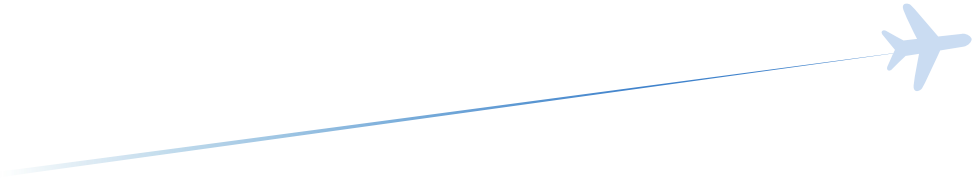


# Facts, Rules and Queries Capturing En-Route ATC Operations

| | |
|---|---|
| **Deliverable ID:** | **D4.4** |
| **Dissemination Level:** | **PU** |
| **Project Acronym:** | **AISA** |
| **Grant:** | **892618** |
| **Call:** | **H2020-SESAR-2019-2** |
| **Topic:** | **SESAR-ER4-01-2019** |
| **Consortium Coordinator:** | **FTTS** |
| **Edition date:** | **10 November 2021** |
| **Edition:** | **00.01.00** |
| **Template Edition:** | **02.00.02** |

Founding Members

EUROPEAN UNION EUROCONTROL

SESAR JOINT UNDERTAKING

## Authoring & Approval

### Authors of the document

| Name/Beneficiary | Position/Title | Date |
|---|---|---|
| Tomislav Radišić/FTTS | Assistant Professor | 19 March 2021 |
| Dorea Antolović/FTTS | Project Associate | 22 March 2021 |
| Mia Bazina/FTTS | Project Associate | 23 March 2021 |
| Ivan Tukarić/FTTS | Project Associate | 13 September 2021 |

### Reviewers internal to the project

| Name/Beneficiary | Position/Title | Date |
|---|---|---|
| Tomislav Radišić/FTTS | Assistant Professor | 28 September 2021 |
| Michael Schrefl/JKU | University Professor | 25 October 2021 |
| Bernd Neumayr/JKU | University Assistant | 3 November 2021 |

### Approved for submission to the SJU By - Representatives of beneficiaries involved in the project

| Name/Beneficiary | Position/Title | Date |
|---|---|---|
| Tomislav Radišić/FTTS | Assistant Professor | 10 November 2021 |

### Rejected By - Representatives of beneficiaries involved in the project

| Name/Beneficiary | Position/Title | Date |
|---|---|---|
| | | |
| | | |

## Document History

| Edition | Date | Status | Author | Justification |
|---|---|---|---|---|
| 00.00.01 | 22/03/2021 | Initial draft | Dorea Antolović | New document |
| 00.00.02 | 14/06/2021 | Draft | Mia Bazina | New sections |
| 00.00.03 | 28/09/2021 | Final draft | Tomislav Radišić | First final draft |
| 00.00.04 | 05/11/2021 | Comments integrated | Tomislav Radišić | Comments integrated to final draft |
| 00.01.00 | 10/11/2021 | First issue | Tomislav Radišić | First issue |







# AISA

AI SITUATIONAL AWARENESS FOUNDATION FOR ADVANCING AUTOMATION

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 892618 under European Union's Horizon 2020 research and innovation programme.

## Abstract

This deliverable gives an insight into the knowledge engineering needed for ensuring all relevant rules and facts about en-route ATC operations have been captured. The relevant facts and rules have mostly been gathered via interviews with licenced ATCOs and by closely examining FIXM and AIXM. The KG-Prolog Mapper is used for converting RDF facts from the KG into Prolog facts. Lastly, SPARQL queries that will be used to monitor the traffic situation and ATCOs have been developed and explained in this deliverable that can be considered a continuation of D4.3 Populated knowledge graph.

# Table of Contents



# List of Figures

## Executive Summary

This document shows all relevant rules and facts about en-route ATC operations and the knowledge engineering behind it. All chosen ATCO tasks are broken down in corresponding flowcharts as well as affiliated inputs, outputs and SPARQL queries, as implemented in Java. The most important parts of KG-Prolog Mapper development are also explained.

## Intended Audience

This deliverable is intended for those employed within the SESAR Joint Undertaking and by ATM community experts, as well as other professionals working on research and development in the fields of data and knowledge engineering and artificial intelligence. It may also concern those employed in EUROCONTROL and the ANSPs as they might take inspiration from the proposed implementation. The components described in this deliverable are just some out of many components of the AI Situational Awareness System developed in this project and they provide a basis for further developments before and during the simulator stages of the AISA project. This deliverable will also be useful to all the partners involved in the project going into Work Package 5 of the AISA project.

# 1 Introduction

This deliverable describes activities related to task 4.4 Knowledge engineering for en-route Air Traffic Control (ATC) operations. Knowledge engineering for en-route ATC operations encompassed the encodement of all relevant facts and rules about the ATC operations gathered by interviews with licenced ATCOs (Air Traffic Control Officers). SPARQL queries used to monitor the traffic situation and ATCO have also been developed and are written out for each task.

## 1.1 Definitions

Given below are definitions for some of the most important topics handled in this deliverable.

**Knowledge Graph (KG).** A knowledge graph is a persistent RDF (Resource Description Framework) dataset comprising data and metadata. The schema of the KG contents is specified via the RDF Schema (RDFS) and the Shapes Constraint Language (SHACL) [1].

**KG System.** Various definitions of a KG system exist, but in the scope of this project we will limit them to a few. "KG system" can be used to refer to: the KG itself; application-independent software components for storing, processing and querying the KG; a set of application-specific engines responsible for loading, querying, inserting, processing, importing and exporting data and metadata in the KG; and a control component used for invoking different engines [2].

**Rules.** As opposed to rule-based knowledge (used in Prolog) and knowledge-based rules, this deliverable uses the term "rules" to refer to rules of ATC operations, as per the Grant Agreement [3]. Any time the deliverable refers to another type of rules, they are explicitly noted as such – e.g. "Prolog rules".

**Prolog.** Prolog is a logic programming language that is used for applications of symbolic computation in relational databases, abstract problem solving, many areas of artificial intelligence etc. [4].

**SHACL (Shapes Constraint Language).** A language for describing and validating RDF graphs. SHACL uses the RDF and RDFS vocabularies, however full RDFS inferencing is not required [1].

**SPARQL (SPARQL Protocol and RDF Query Language).** As the name suggests, SPARQL is a query language for RDF. It can be used to express queries across diverse data sources and it contains capabilities for querying required and optional graph patterns. The results of SPARQL queries can be results sets or RDF graphs [5].

Later on in the deliverable, more thorough explanations of these terms are given.

## 1.2 Purpose of the document

The purpose of this deliverable is to describe the work needed to complete task 4.4 of knowledge engineering for en-route ATC operations. The task's purpose was to capture and encode all relevant facts and rules about en-route ATC operations, were gathered via interviews with licenced ATCOs. The other important part of this task and subsequently the deliverable were SPARQL queries that were developed in order to monitor the traffic situation and ATCO status. The queries are an integral part of Java classes which represent ATCO task implementations in the Java programming language.

## 1.3 Structure and methodology

This document describes the knowledge engineering for en-route ATC operations performed as part of task 4.4. This first chapter is a brief introduction to this deliverable and the corresponding task. In chapter 2 a brief overview is given of the term knowledge engineering. Chapter 3 gives an insight on what exactly are Prolog, SHACL and SPARQL and how they are incorporated in this project, as well as the role of the KG-Prolog Mapper. The fourth chapter is there to further explain why the reasoning was implemented in Java, how it can be implemented in Prolog and our argumentation for the discrepancy. The main focus of this deliverable is on ATC facts and rules which are provided in the final chapter. The tasks are grouped into eleven types of tasks and each of the tasks has analogous inputs, outputs, flowcharts that describe the task, and SPARQL queries used to extract relevant data.

Appendix A provides a glossary of acronyms used throughout the deliverable.

## 1.4 Relations to other documents

A full list of references is given at the end of this deliverable. This deliverable is linked to the following project deliverables:

- AISA D2.1: Concept of Operations for AI Situational Awareness System
- AISA D4.2: KG-Prolog Mapper

# 2 Knowledge engineering

In order to try and imitate the awareness of an ATCO, a thorough interview was conducted with licensed ATCOs about their day-to-day activities. They gave the necessary insight needed to set the knowledge engineering in motion. These interviews were conducted early in the project (July 2020), but their importance stretches throughout the work packages. Another important input was the information regarding facts and rules found in the Flight Information Exchange Model (FIXM) and the Aeronautical Information Exchange Model (AIXM) [6].

These models offer very specific and accurate constraints regarding all sorts of relevant features, such as minimum and maximum rate of climb (ROC) or rate of descent (ROD), airport/heliport IATA (International Air Transport Association) codes and their limitations (three letters from A to Z) etc. An example of constraints is given in Figure 1, where AIXM dictates the format of IATA codes.

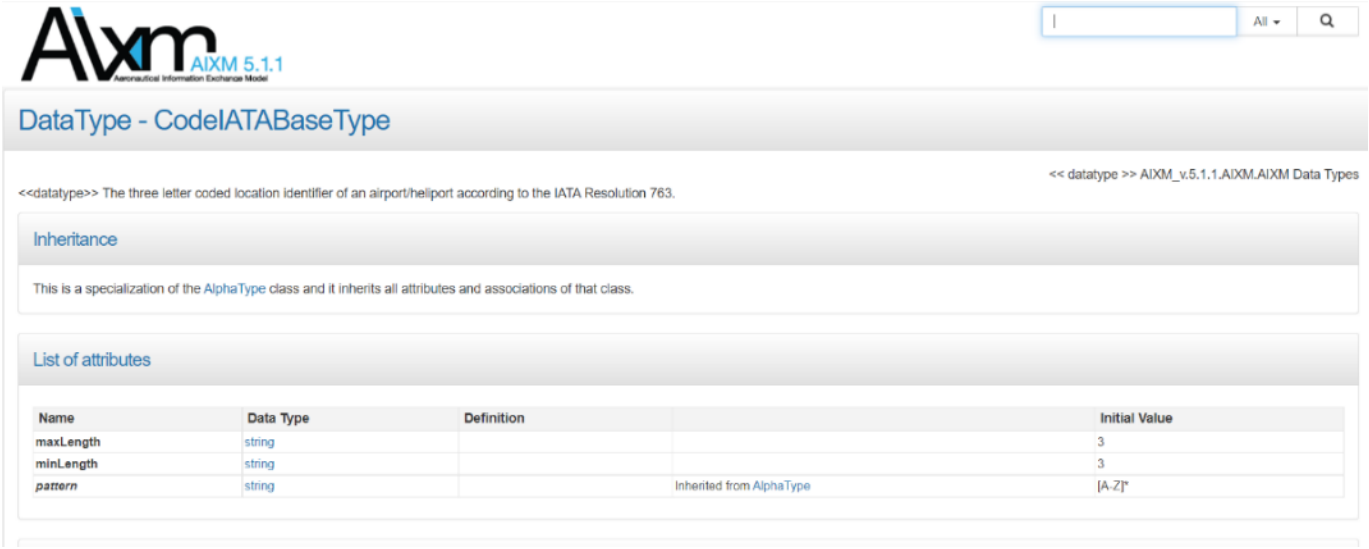



**Figure 1. Example of constraints regarding IATA codes [6]**

# 3 Prolog, SHACL, SPARQL and Mapper

SHACL, Prolog and SPARQL are three languages that play a big part in the AISA project. Knowledge graphs (KG) serve as large collections of facts which can be queried by SPARQL and the KG-Prolog mapper. The mapper was developed by a JKU (Johannes Kepler Universität Linz) team to automatically translate SHACL shapes defined in RDFS to Prolog predicates, associated SPARQL queries and Prolog code. Details about the KG-Prolog mapper are discussed in detail in D4.2 KG-Prolog mapper [2]. This chapter only gives a broad look into these three languages and the mapper itself.

## 3.1 Prolog

Prolog stands for "**pro**gramming in **log**ic" and is a programming language which centers on basic mechanisms such as pattern matching, tree-base data structuring, and automatic backtracking. Prolog can reason about spatial relations and their consistency considering the general rule. All of this makes Prolog a powerful language for use in artificial intelligence (AI) applications [7].

Prolog is able to determine whether or not a given statement follows logically from other statements. For example, for statements "All aircraft can fly" and "Boeing 747 is an aircraft", Prolog will manage to infer the conclusion and answer affirmatively to the query "Can Boeing 747 fly?". It's important to note that Prolog assumes that its database – which stores Prolog facts and rules – is a *closed world*, meaning that statements are either true or false. This also means that a negative answer will be returned for queries with a negative answer (e.g. "Can all cars fly?"), but also for those queries which cannot be proved as true because of a lack of information.

## 3.2 SHACL

SHACL is a language for describing and validating RDF graphs. It specifies a vocabulary (using triples) to describe the shape that data should have. This shape contains information about nodes or a set of nodes to which they can be applied. The SHACL syntax is defined in terms of RDF - one of RDF serialization formats, turtle (.ttl), was used in this project.

There are two inputs for a SHACL processor: a data graph containing the RDF data that needs to be validated, and a shapes graph that contains the shapes. It is also possible to use a single graph that will contain both the data and shapes. Two main types of shapes exist: node shapes and property shapes. Node shapes declare constraints directly on a node, while property shapes declare constraints on the values that are associated with a node through a path. A property sh:path declares the path that goes from the focus node to the described value and the most frequent paths are predicate paths formed by a single IRI (Internationalized Resource Identifier). A node shape usually has several property shapes declared through the sh:property predicate [8].

## 3.3 SPARQL

SPARQL is a standard query language and protocol for RDF databases. It queries a great variety of data and can efficiently extract information hidden in non-uniform data which is stored in different formats and sources. A SPARQL query can be executed on any database that can be viewed as RDF via middleware. SPARQL queries can access multiple data stores because SPARQL is also an HTTP-based

(Hypertext Transfer Protocol) transport protocol, where any endpoint can be accessed with a standardized transport layer [9].

For the purposes of this project SPARQL queries have been used to extract the relevant data from the KGs to be able to use this data for calculations and reaching conclusions based on the data. The fifth chapter provides used SPARQL queries for each given task based on the required information that is needed for the particular task.

## 3.4 KG-Prolog Mapper

As mentioned earlier in this deliverable, the KG-Prolog mapper was developed to automatically translate SHACL shapes to Prolog predicates, associated SPARQL queries and Prolog code. First of all, the concept of the KG system needs to be explained. It consists of the KG and the application-independent software components (for storing, processing, and querying the KG), as well as a set of application-specific engines responsible for loading, querying, inserting, processing, importing, and exporting data and metadata in the KG. Lastly, the fourth component is a control component made for invoking different engines.

The AISA KG is an RDF dataset with all the static and dynamic data and metadata relevant for AI situational awareness. It is stored on a KG server and queried and updated via SPARQL. The schema used is specified in RDF Schema and SHACL, while the data and metadata are added dynamically to the KG and processed and queried via application-specific engines which are mainly implemented in Java. Advanced reasoning tasks are going to be realized based on rule-based knowledge represented in Prolog [2].

There are two approaches for the KG-Prolog mapping, both with their advantages and flaws. The schema-oblivious approach represents every RDF quadruple as a Prolog fact. This approach is suited for writing results from Prolog programs to the KG and provides a flexible approach for querying the KG from Prolog. It is, however, unhandy for Prolog programmers when it comes to KG data with a complex structure since knowledge about one object is distributed over multiple facts. The other approach is the schema-aware approach that overcomes the shortcomings of the schema-oblivious approach when it comes to reading structured data from the KG. All facts about a single object in one named graph are collected in a single fact, in accordance with the KG schema. This approach is more convenient for Prolog programmers as knowledge about one object is already collected in schema-conforming facts [2].

Three different realization variants and one subvariant have been developed. Variant A takes the KG data as input and generates schema-specific SPARQL queries in Java. Variant B does the same, but the queries are embedded into Prolog rules. In the original variant the input facts are virtual and SPARQL queries are executed as part of the Prolog program, while in the subvariant (B2) the queries are executed separately and declare the input facts in the Prolog database. Variant C replicates the RDF quadruples from the KG in Prolog's RDF database and generates schema-specific mapping rules in Prolog. Preliminary performance studies have been conducted on all the variants and the subvariant and the conclusion was that variants A, C and subvariant B2 have similar performance characteristics, and only the variant B has a significantly poorer performance. The C variant was then chosen for integration into the KG system, for two reasons. The first reason was the availability of the full KG in the form of RDF quadruples that allows for flexibility in choosing between the schema-aware and schema-oblivious approach. The other reason is the simplicity of adding new data to the KG which makes it available to Prolog [2].

In summary, Prolog programs are integrated in the KG module and a Prolog-based KG module is represented in Java and in Prolog. Prolog modules execute queries over the RDF database, perform reasoning and write results into the current graph in the RDF database which is then replicated to the KG. Based on all conducted analyses and considerations, a combination of schema-aware and schema-oblivious approaches has been selected. It combines the flexibility of the schema-oblivious approach and the convenience of the schema-aware approach [2].

# 4 Logic implementation

The implementation of "reasoning" in task 4.4 was done in the Java programming language by creating Java classes that correspond to each ATCO task. The classes use SPARQL queries, described in this deliverable, to access data stored in the KG. Data is stored in variables and sometimes converted, and then used for additional variable calculations or directly in logical statements. Logical statements consist of conditional statements and logical operators, connecting the input with one of the pre-determined outputs.

During the process of task formulation, it had been determined that task logical trees are simple enough to implement all tasks in Java. With the development of the KG2PrologMapper, some tasks and reasoning were implemented in Prolog for familiarization and testing. Since Java and Prolog were both confirmed to operate satisfactorily, the decision was made to keep Java implementations of tasks but consider Prolog as a valid alternative and option for more complex reasoning cases.

The main arguments for the decision were time constraints and system homogeneity. The conversion of all task implementations to Prolog would have been time consuming, while the result would remain effectively identical. On the other hand, the conversion of *some* tasks into Prolog would have led to mixing of different reasoning methods.

The difference between Java and Prolog is best demonstrated by an example. We used task 1.2 Check that aircraft is at cleared FL (Flight Level) to show, in parallel, the implementation of reasoning in Java and Prolog. Since Java was ultimately chosen for logic implementation, in the entirety of the next chapter only Java task implementations are shown.

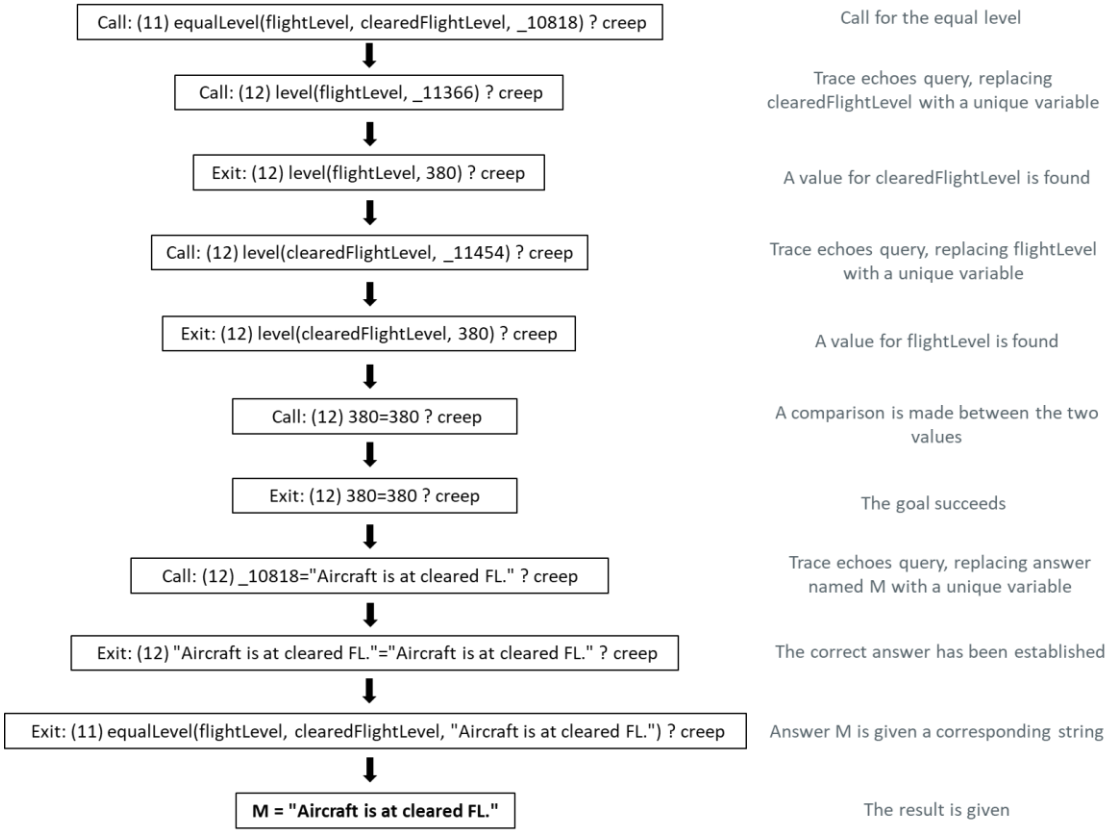



**Figure 2. Prolog reasoning for task 1.2**

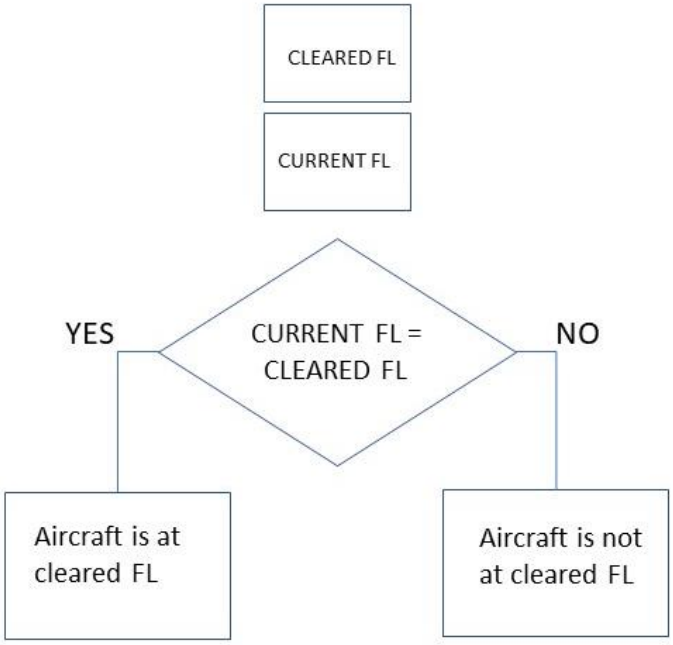



**Figure 3. Java reasoning for task 1.2**

The input for both programmes were *clearedFlightLevel* and *flightLevel* and their respective values. The simplified excerpt from Prolog for easier understanding looks like this:

```
level('flightLevel',380).
level('clearedFlightLevel',380).
equalLevel(A,B,M) :-
level(A,X), level(B,Y), X = Y, M = "Aircraft is at cleared FL.";
level(A,X), level(B,Y), X \= Y, M = "Aircraft is not at cleared FL.".
```

The excerpt consists of two facts about flight level designations (and their values) and one rule that defines what constitutes as equal levels. The query then uses function arguments, which are flight level designations, to find and compare values while also giving each result an appropriate output string.

The Java function is as follows:

```
String checkIfAtCFL(float fl,float cfl){
                        if(fl==cfl) {
                                return "Aircraft is at cleared FL.";
                        } else {
                                return "Aircraft is not at cleared FL.";
                        }
```

The “checkIfAtCFL” function has two inputs, flight level *fl* and cleared flight level *cfl*, which are compared using the if function. As in Prolog, there are two different possible outputs.

The earlier graphical comparison highlights the overall difference between Prolog and traditional conditional logic. The language used is also different – Prolog logic is expressed in terms of relations (facts, rules), and a computation is the process of executing a query over those relations. In Java, as previously described, conditional logic uses *logical operators* and *conditional statements*, which run when the main method is called.

# 5 Facts and rules

All the collected facts and rules can be seen in this chapter as they are represented graphically via flowcharts. Various AISA tasks have been established prior to this deliverable and discussed in D2.2 Requirements for automation of monitoring tasks via AI SA [10]. Each task has input data (needed for calculations and logical processes) and its source, as well as the expected output. Flowcharts explain the process of reaching conclusions based on the input and the calculations, while SPARQL queries show how to extract the input data from the KG.

Some flowcharts contain blue and/or orange steps. The former represent calculations done in ML modules, while latter represent variables needed for Java task calculations. Some SPARQL queries are querying differing graphs, g1 stands for a current time step graph while g0 represents a previous time step graph. There is also a default graph that contains all the static data and it is queried separately from the dynamic data.

## 5.1 Conformance management

The following set of tasks relates to conformance management. In all those tasks the system uses existing classes and properties in order to infer new knowledge and returns all newly inferred knowledge back into the KG as new property values.

### 5.1.1 Check that aircraft is climbing/descending towards cleared FL

- Input data:
  - Cleared FL (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel
  - Current FL (AC_ROUTE_INFO.ttl)
    - fixm:FlightLevel --> fixm:level
  - Previous FL (AC_ROUTE_INFO.ttl)
    - fixm:FlightLevel --> fixm:level
  - Current ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data:
  - Aircraft is climbing towards cleared FL
  - Aircraft is descending towards cleared FL
  - Aircraft is not climbing towards cleared FL
  - Aircraft is not descending towards cleared FL

- Flowchart:

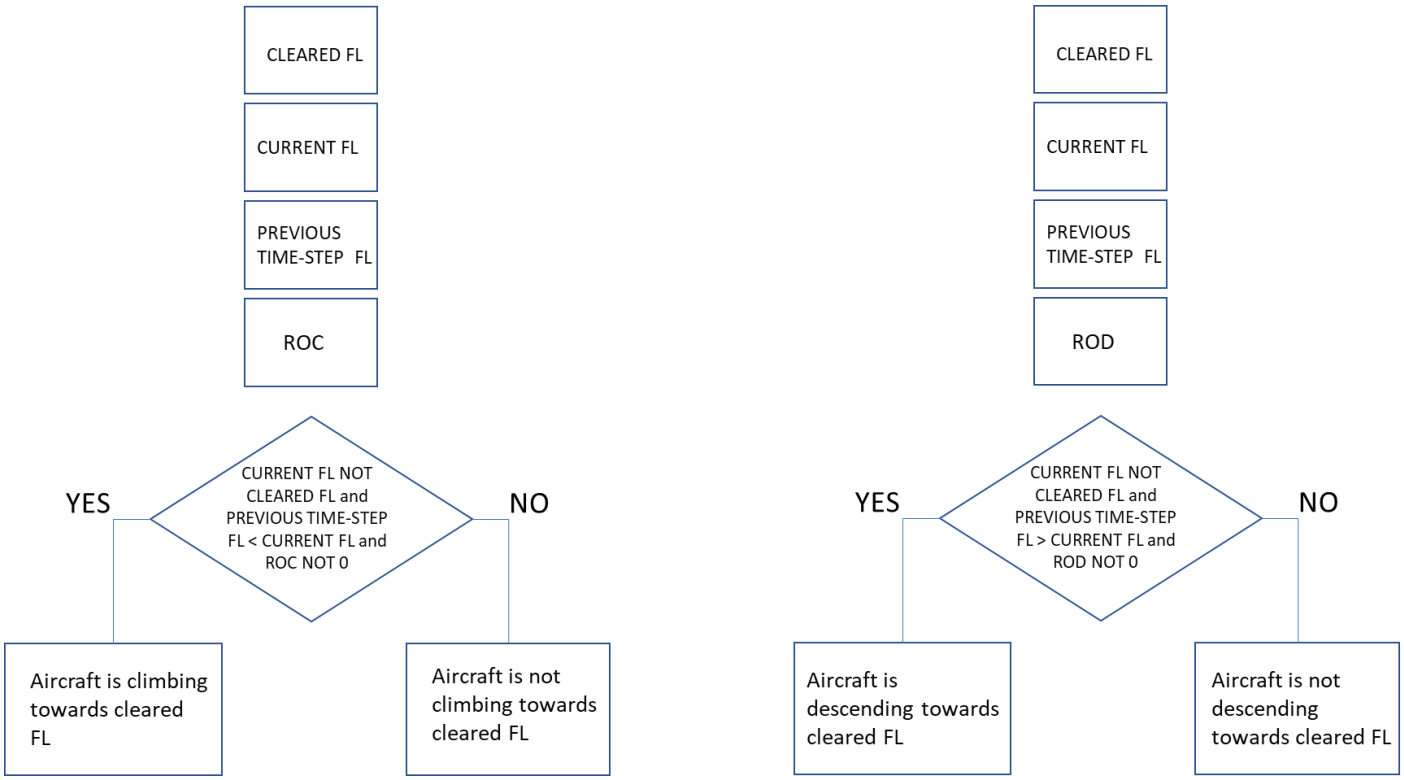



**Figure 4. Task 1.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?verticalRate ?currentFL ?clearedFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?verticalRate ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?currentFL ;
plain:flightIdentification/plain:cleared/plain:clearedFlightLevel/rdf:value ?clearedFL .}}

SELECT ?callsign ?previousFL
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?previousFL .}}
```

### 5.1.2 Check that aircraft is at cleared FL

- Input data:
  - Cleared FL (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel

- Current FL (AC_ROUTE_INFO.ttl)
  - fixm:FlightLevel --> fixm:level

- Output data:
  - Aircraft is at cleared FL
  - Aircraft is not at cleared FL

- Flowchart:

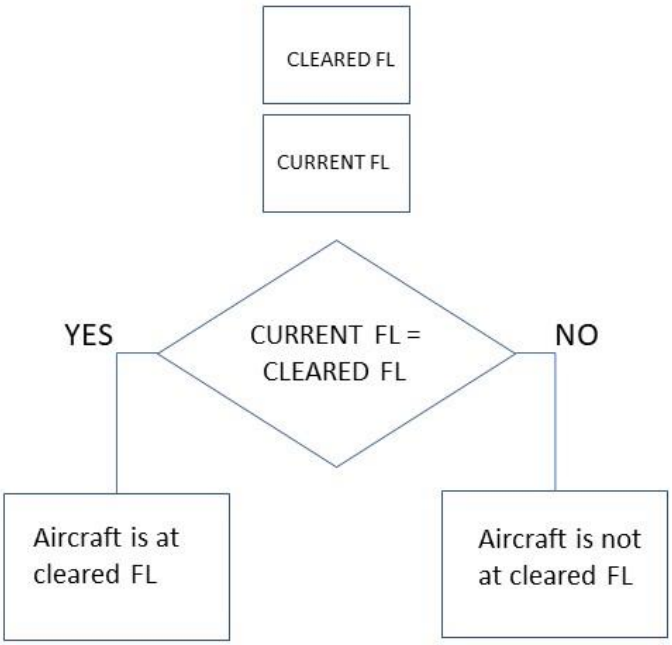



**Figure 5. Task 1.2 flowchart**

- SPARQL query

```
SELECT ?callsign ?currentFL ?clearedFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?currentFL ;
plain:flightIdentification/plain:cleared/plain:clearedFlightLevel/rdf:value ?clearedFL .}}
```

### 5.1.3 Check that aircraft is maintaining FL

- Input data:
  - Current FL (AC_ROUTE_INFO.ttl)
    - fixm:FlightLevel --> fixm:level
  - Previous time-step FL (AC_ROUTE_INFO.ttl)

        - fixm:FlightLevel --> fixm:level
    - Current ROC/ROD (AC_ROUTE_INFO.ttl)
        - fixm:VerticalRate

- Output data
    - Aircraft is maintaining FL
    - Aircraft is not maintaining FL

- Flowchart

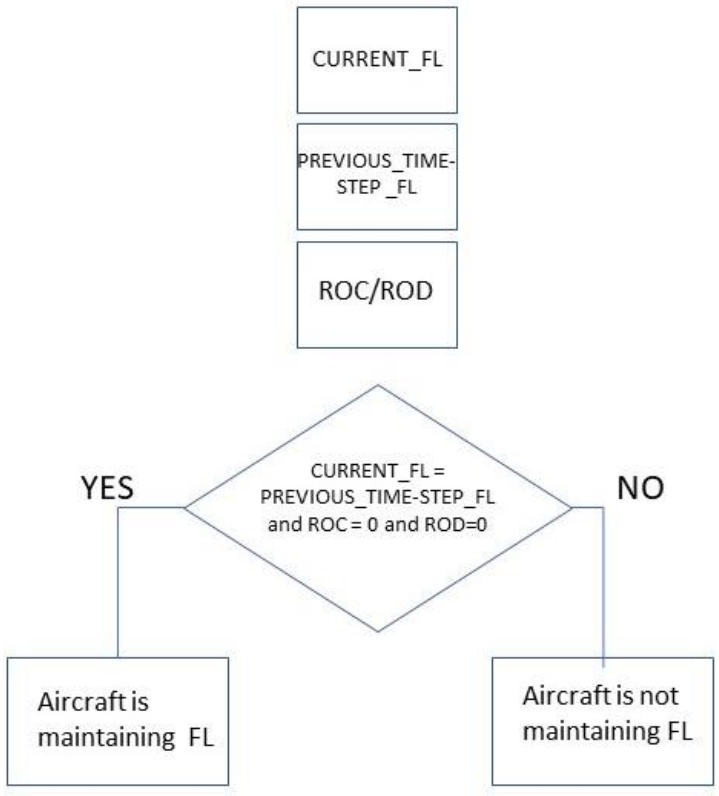



**Figure 6. Task 1.3 flowchart**

- SPARQL queries

```
SELECT ?callsign ?verticalRate ?currentFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?verticalRate ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?currentFL .}}

SELECT ?callsign ?previousFL
WHERE { GRAPH ?g0
```



```
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?previousFL .}}
```

### 5.1.4 Check that aircraft is turning towards/opposite of cleared heading

- Input data
  - Current heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Previous time-step heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Cleared heading (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:track
- Output data
  - Aircraft is turning towards cleared heading
  - Aircraft is not turning towards cleared heading
  - Aircraft is turning opposite of cleared heading
- Flowchart

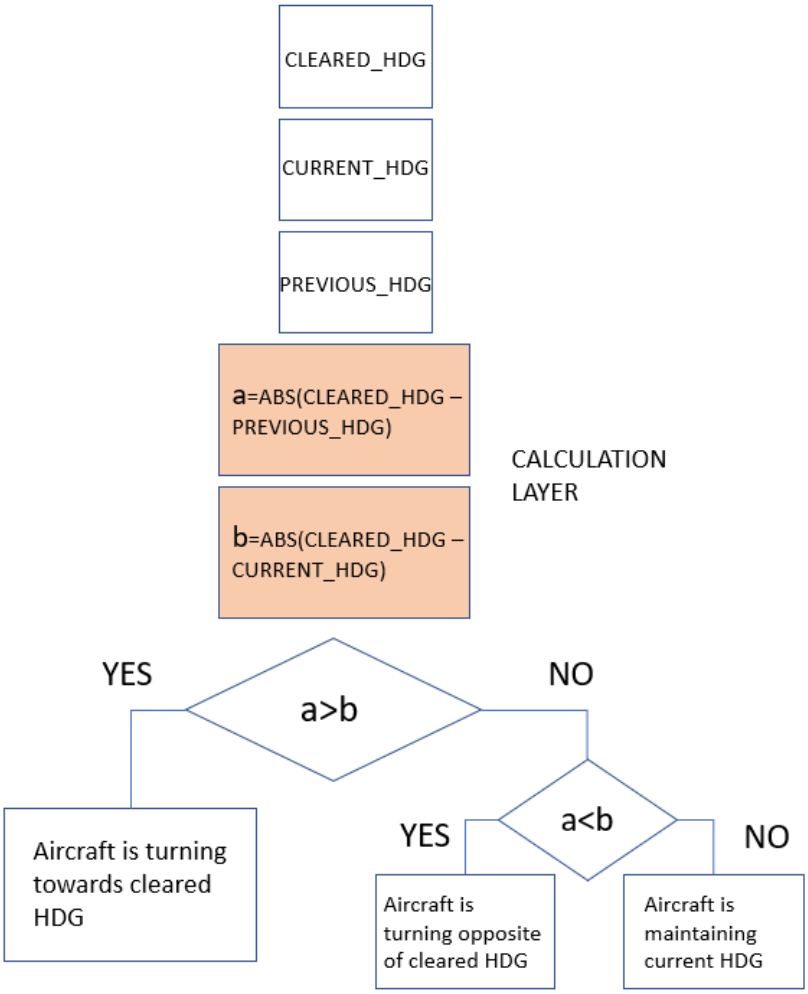



**Figure 7. Task 1.4 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentHeading ?clearedHeading
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedHeading .}}

SELECT ?callsign ?previousHeading
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?previousHeading .}}
```

### 5.1.5 Check that aircraft is at cleared heading

- Input data

- Cleared heading (CLEARED_FLIGHT_INFORMATION.ttl)
  - fixm:ClearedFlightInformation --> fixm:track
- Current heading (AC_ROUTE_INFO.ttl)
  - fixm:AircraftPosition --> fixm:track

- Output data
  - Aircraft is at cleared heading
  - Aircraft is not at cleared heading
- Flowchart

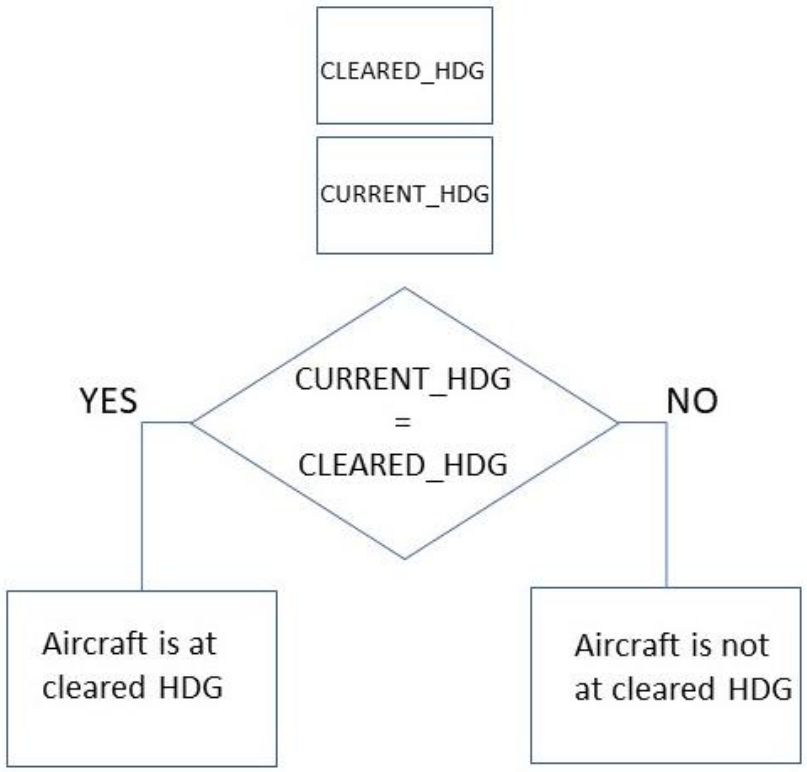



**Figure 8. Task 1.5 flowchart**

- SPARQL query

```
SELECT ?callsign ?currentHeading ?clearedHeading
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedHeading .}}
```

### 5.1.6 Check that aircraft is maintaining current heading (different than cleared heading)

- Input data
  - Current heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Previous time-step heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Cleared heading (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:track
- Output data
  - Aircraft is maintaining current heading (different than cleared heading)
  - Aircraft is not maintaining current heading (different than cleared heading)
  - Aircraft is maintaining cleared heading
- Flowchart

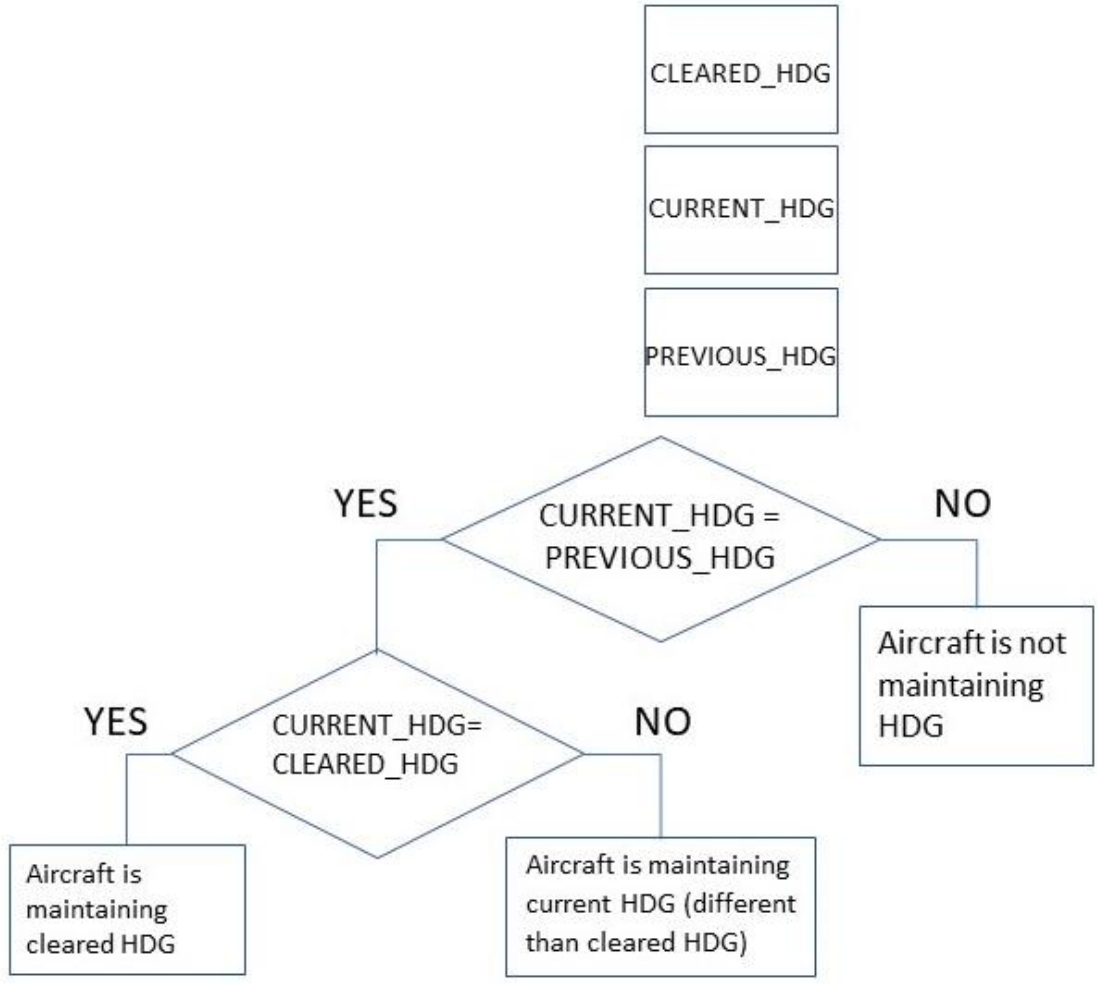



**Figure 9. Task 1.6 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentHeading ?clearedHeading
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedHeading .}}

SELECT ?callsign ?previousHeading
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?previousHeading .}}
```

### 5.1.7 Check that aircraft is accelerating/decelerating towards cleared speed

- Input data
  - Cleared speed (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedSpeed --> fixm:AirspeedChoice and fixm:AirspeedRange
  - Current speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
  - Previous time-step speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
- Output data
  - Aircraft is accelerating towards cleared speed
  - Aircraft is not accelerating towards cleared speed
  - Aircraft is decelerating towards cleared speed
  - Aircraft is not decelerating towards cleared speed
- Flowchart

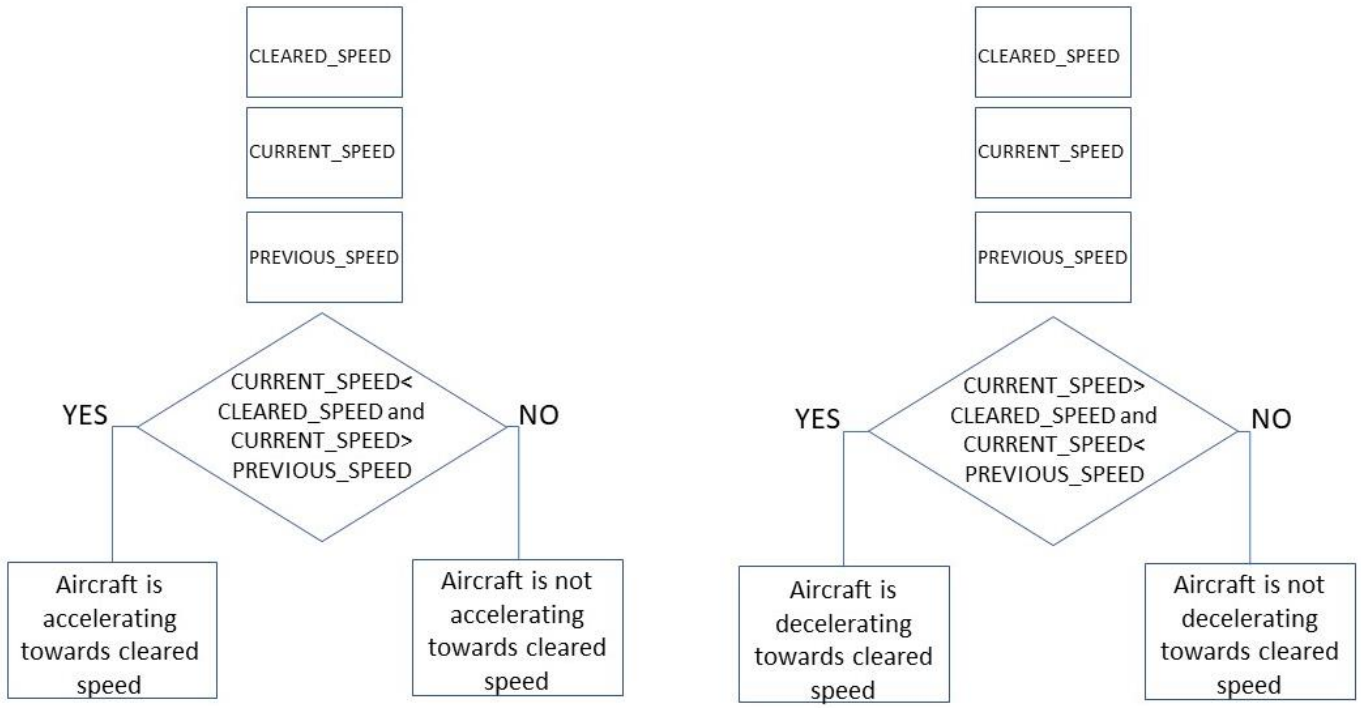



**Figure 10. Task 1.7 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentSpeed ?clearedSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?currentSpeed ;
```

plain:flightIdentification/plain:cleared/plain:clearedSpeed/rdf:value ?clearedSpeed .}}

SELECT ?callsign ?previousSpeed
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?previousSpeed .}}

### 5.1.8 Check that aircraft is flying at cleared speed

- Input data
  - Cleared speed (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedSpeed --> fixm:AirspeedChoice and fixm:AirspeedRange
  - Current speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
- Output data
  - Aircraft is flying at cleared speed
  - Aircraft is not flying at cleared speed
- Flowchart

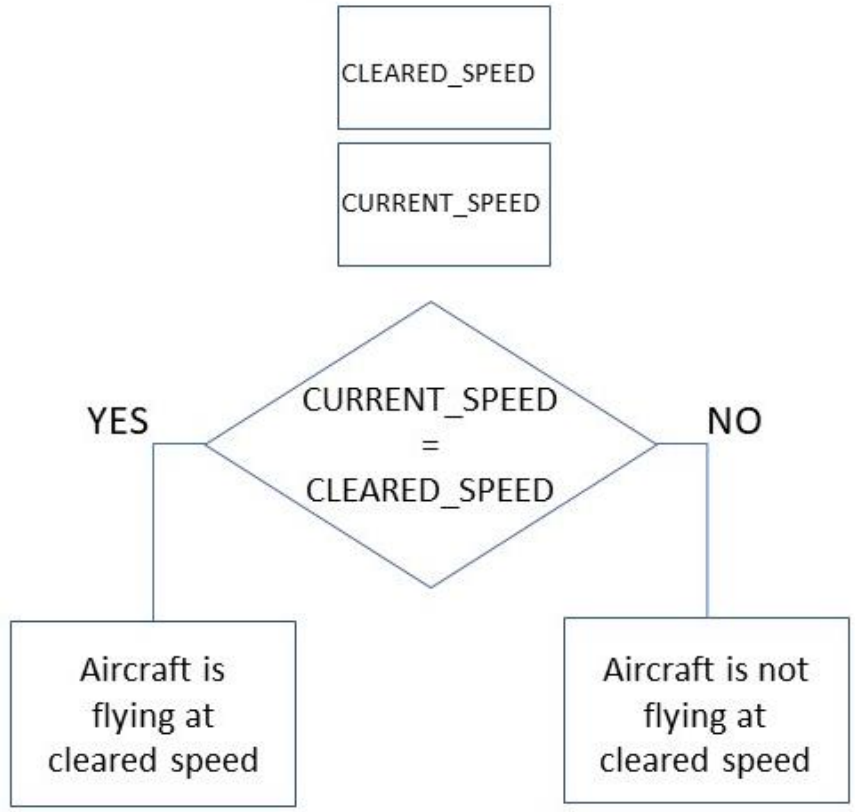



**Figure 11. Task 1.8 flowchart**

- SPARQL query

```
SELECT ?callsign ?currentSpeed ?clearedSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?currentSpeed ;
plain:flightIdentification/plain:cleared/plain:clearedSpeed/rdf:value ?clearedSpeed .}}
```

### 5.1.9 Check that aircraft is maintaining current speed (different than cleared speed)

- Input data
  - Cleared speed (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedSpeed --> fixm:AirspeedChoice and fixm:AirspeedRange
  - Current speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed

- Previous time-step speed (AC_ROUTE_INFO.ttl)
  - fixm:AircraftPosition --> fixm:actualSpeed

- Output data
  - Aircraft is maintaining current speed (different than cleared speed)
  - Aircraft is not maintaining speed
  - Aircraft is maintaining cleared speed

- Flowchart

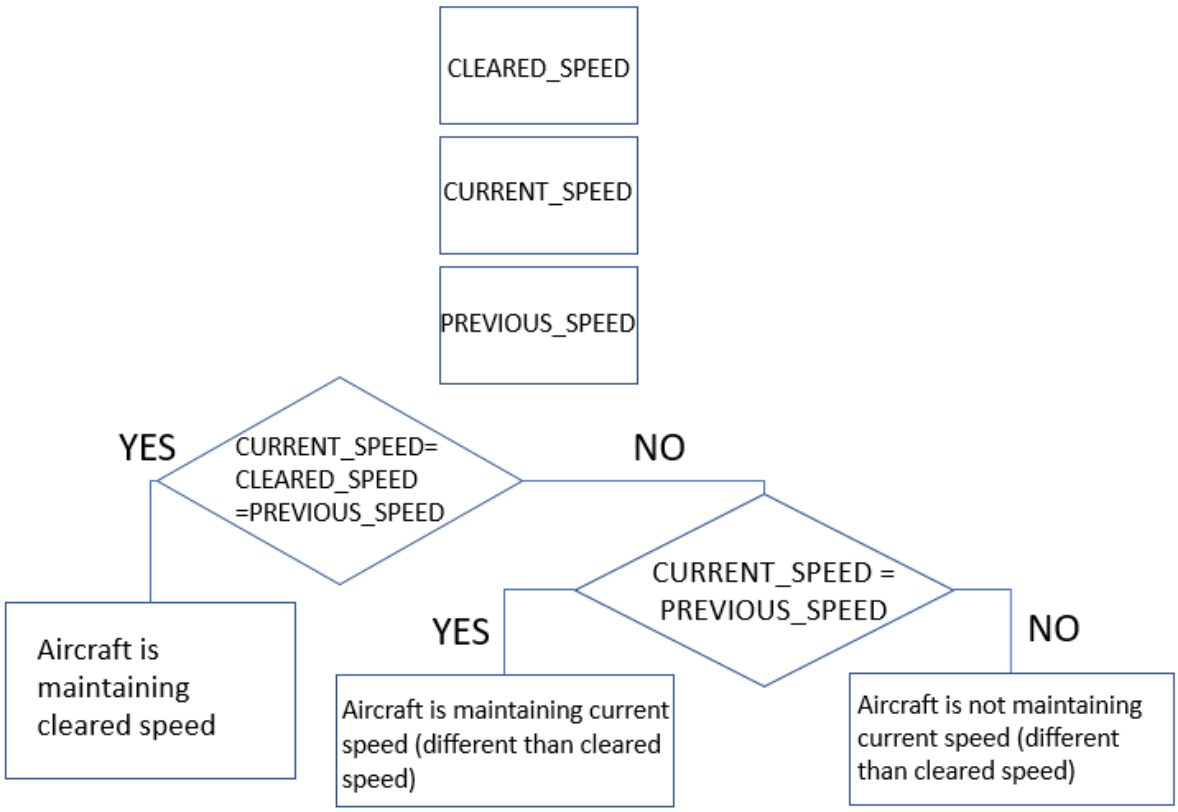



**Figure 12. Task 1.9 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentSpeed ?clearedSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?currentSpeed ;
plain:flightIdentification/plain:cleared/plain:clearedSpeed/rdf:value ?clearedSpeed .}}

SELECT ?callsign ?previousSpeed
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?previousSpeed .}}
```

### 5.1.10 Check that aircraft is flying towards cleared point

- Input data
  - Current a/c (aircraft) position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
  - Previous time-step a/c position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
  - Current a/c heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Cleared point position (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:EnRoute --> fixm:ClearedInformation --> fixm:DirectRouting
  - Position tolerance (±2.5NM)
- Output data
  - Aircraft is flying towards cleared point
  - Aircraft is not flying towards cleared point
- Flowchart

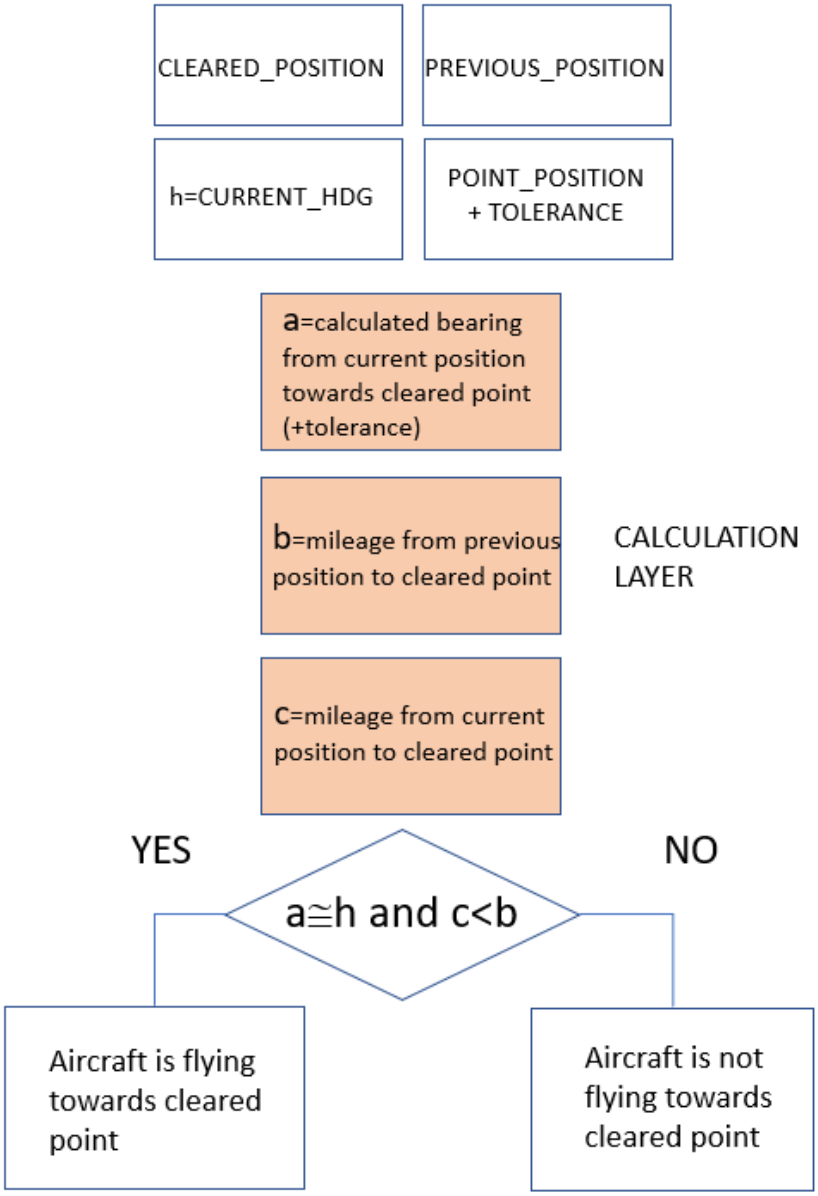



**Figure 13. Task 1.10 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentPosition ?currentHeading ?clearedPoint
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?currentPosition ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:cleared/plain:directRouting/plain:to/rdf:value ?clearedPoint .}}

SELECT ?callsign ?previousPosition
WHERE { GRAPH ?g0
```

{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?previousPosition .}}

### 5.1.11 Check that aircraft is at cleared point

- Input data
  - Current a/c position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
  - Cleared point position (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:EnRoute --> fixm:ClearedInformation --> fixm:DirectRouting
  - Position tolerance (±2.5NM)
- Output data
  - Aircraft is at cleared point
  - Aircraft is not at cleared point
- Flowchart

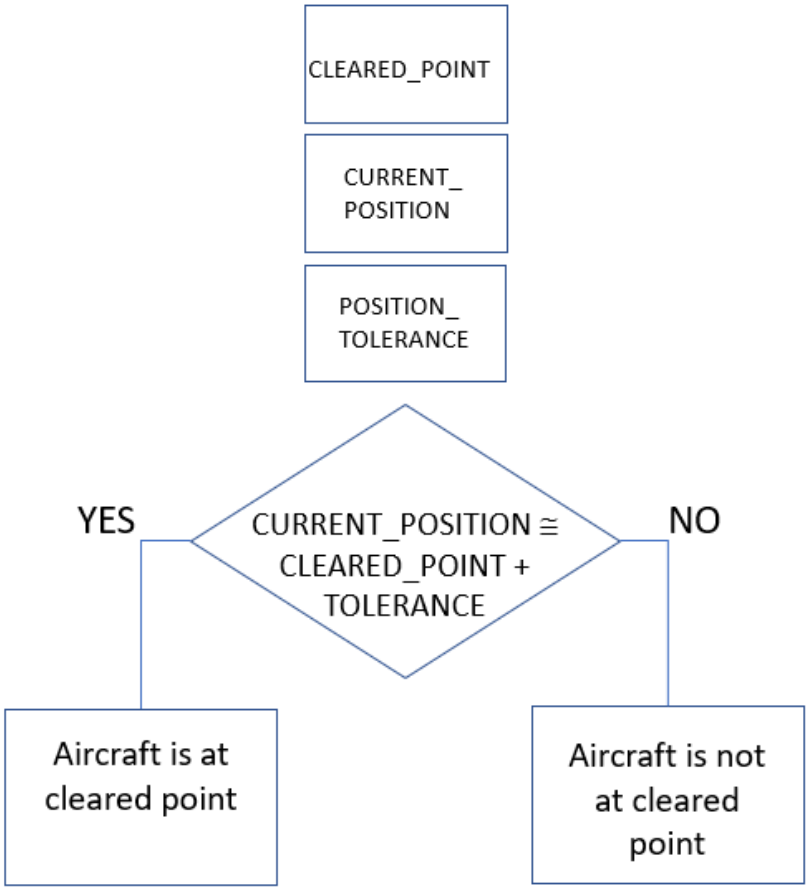



**Figure 14. Task 1.11 flowchart**

- SPARQL query

```
SELECT ?callsign ?currentPosition ?clearedPoint
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?currentPosition ;
plain:flightIdentification/plain:cleared/plain:directRouting/plain:to/rdf:value ?clearedPoint .}}
```

### 5.1.12 Check that aircraft's current ROC/ROD is lower/higher than cleared

- Input data
  - Cleared ROC/ROD (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm: ClearedFlightInformation --> fixm:rateOfClimbDescent --> fixm:VerticalRate
  - Current ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data
  - Aircraft's current ROC is lower than cleared
  - Aircraft's current ROC is higher than cleared
  - Aircraft's current ROD is lower than cleared
  - Aircraft's current ROD is higher than cleared
- Flowchart

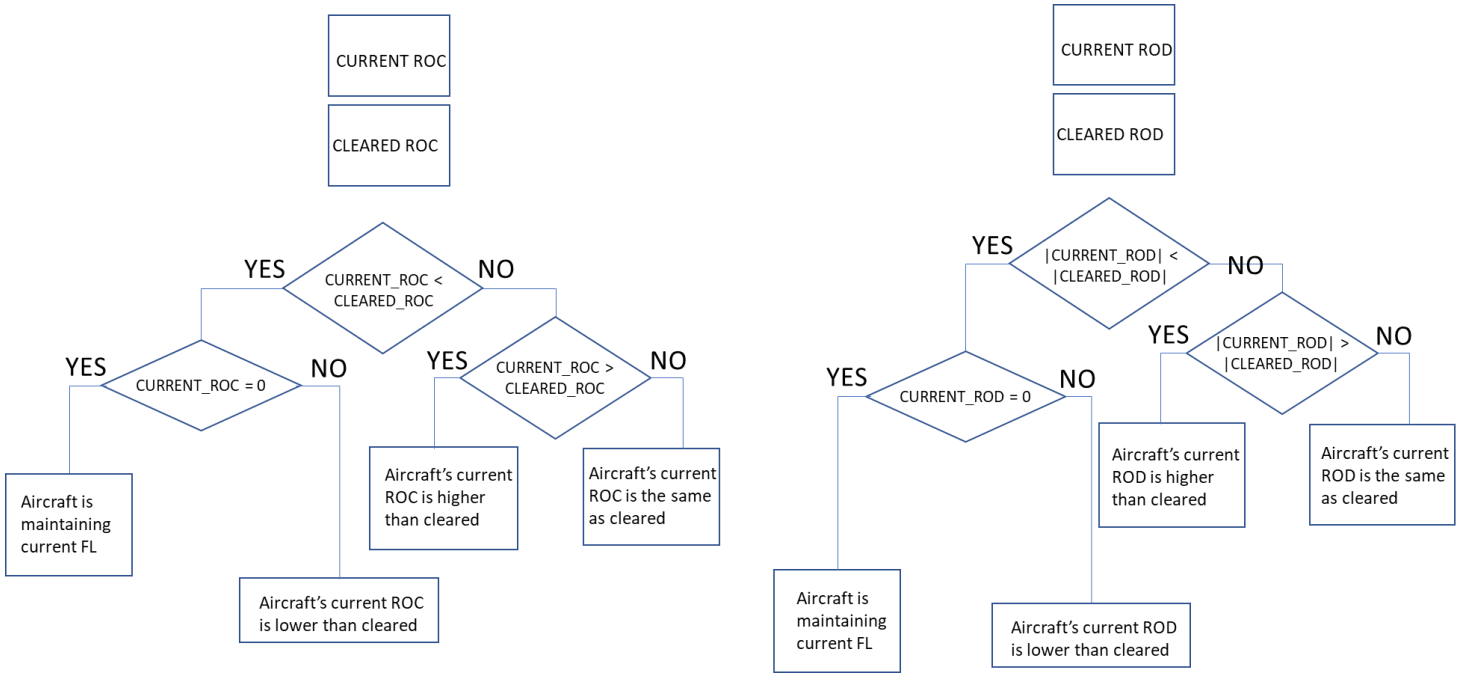



**Figure 15. Task 1.12 flowchart**

- SPARQL query

```
SELECT ?callsign ?currentVerticalRate ?clearedVerticalRate
```

```
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedVerticalRate .}}
```

### 5.1.13 Check that aircraft is maintaining cleared ROC/ROD

- Input data
  - Cleared ROC/ROD (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:rateOfClimbDescent --> fixm:VerticalRate
  - Current ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
  - Previous time-step ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data
  - Aircraft is maintaining cleared ROC
  - Aircraft is not maintaining ROC
  - Aircraft is maintaining current ROC (different than cleared ROC)
  - Aircraft is maintaining cleared ROD
  - Aircraft is not maintaining cleared ROD
  - Aircraft is maintaining current ROD (different than cleared ROD)
- Flowchart

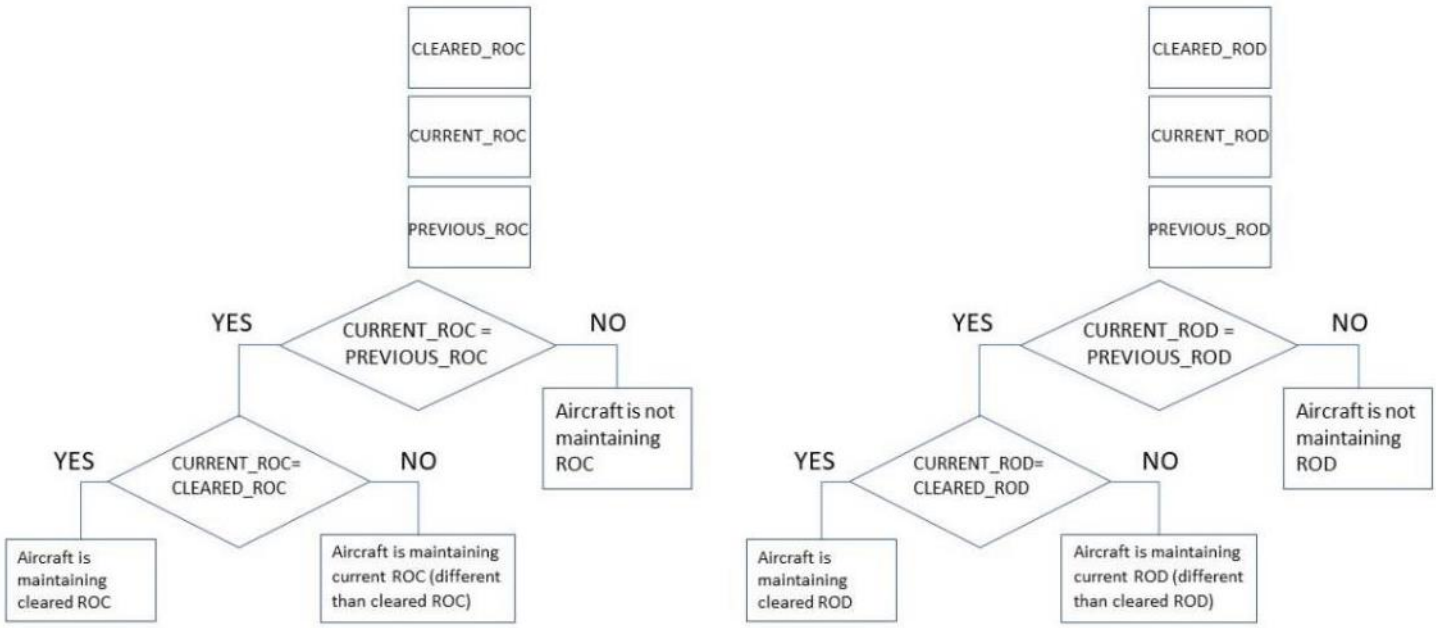



**Figure 16. Task 1.13 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentVerticalRate ?clearedVerticalRate
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedVerticalRate .}}

SELECT ?callsign ?previousVerticalRate
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?previousVerticalRate .}}
```

### 5.1.14 Check that aircraft is increasing/decreasing towards cleared ROC/ROD

- Input data
  - Cleared ROC/ROD (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm: ClearedFlightInformation --> fixm:rateOfClimbDescent --> fixm:VerticalRate
  - Current ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
  - Previous time-step ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data
  - Aircraft is increasing towards cleared ROC/ROD
  - Aircraft is not increasing towards cleared ROC/ROD
  - Aircraft is decreasing towards cleared ROC/ROD
  - Aircraft is not decreasing towards cleared ROC/ROD

- Flowchart

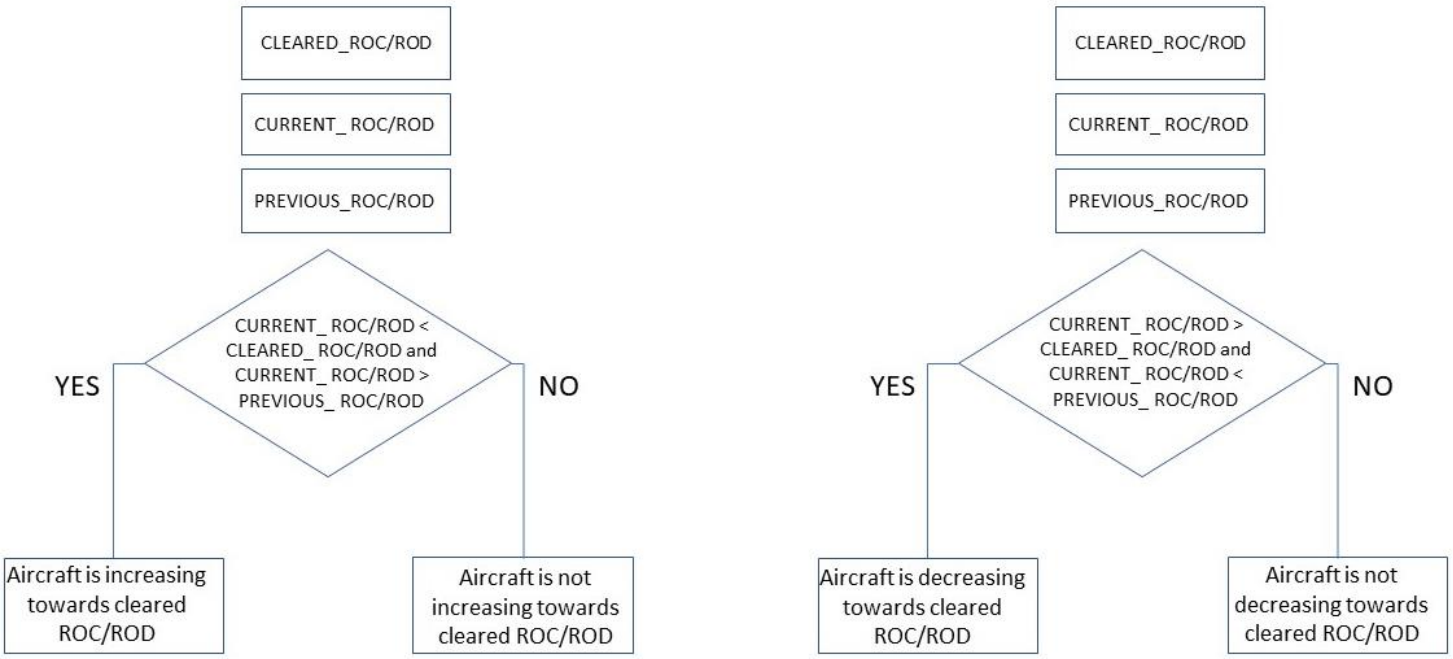



**Figure 17. Task 1.14 flowchart**

- SPARQL queries

```
SELECT ?callsign ?currentVerticalRate ?clearedVerticalRate
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedVerticalRate .}}

SELECT ?callsign ?previousVerticalRate
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?previousVerticalRate .}}
```

## 5.1.15 Check that aircraft is following the 3D trajectory

- Input data
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Planned trajectory (“flight plans”) (AC_ROUTE_INFO.ttl)

- fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
- fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:flightLevel

- Output data
  - A/C is following the 3D trajectory
  - A/C is not following the 3D trajectory

- Flowchart

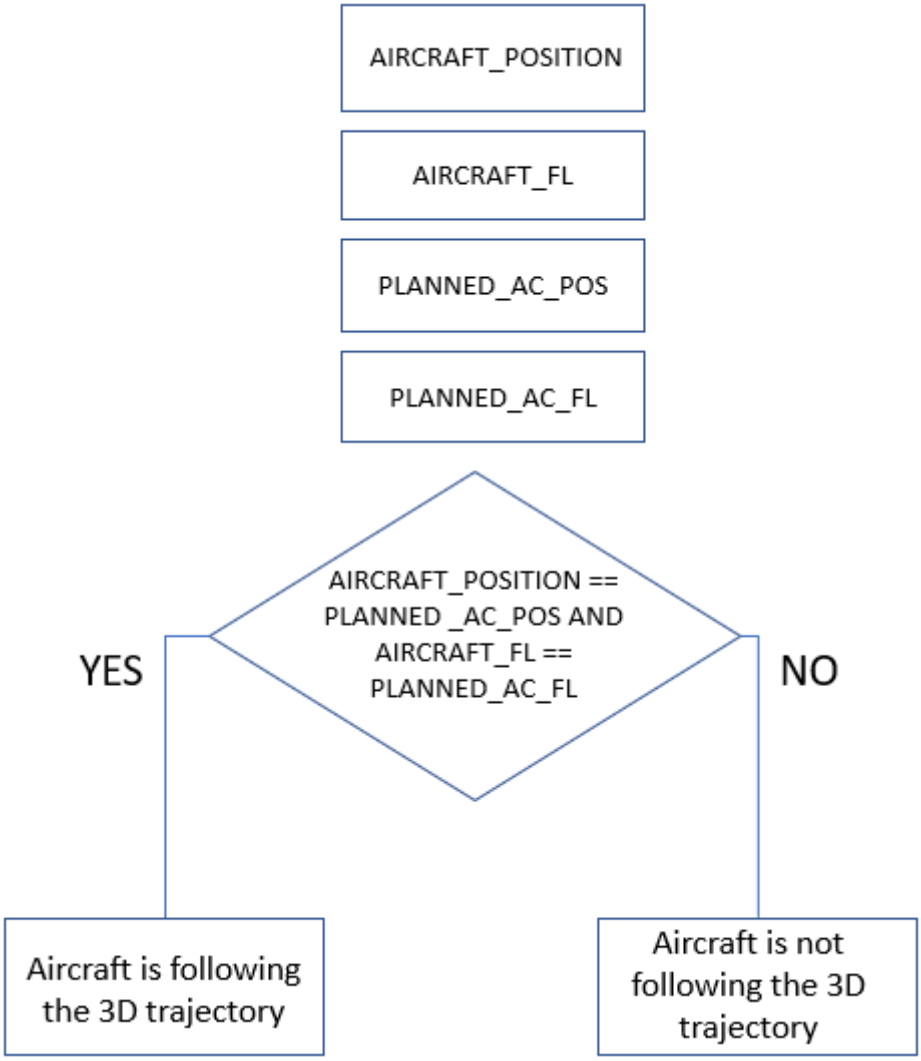



**Figure 18. Task 1.15 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
```

plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}

### 5.1.16 Check if the deviation from 3D trajectory is within tolerance

- Input data
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Planned trajectory points (“flight plans”) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:flightLevel
  - Position tolerance (±2.5NM)
  - FL tolerance (±100ft)
- Output data
  - Deviation from 3D trajectory is within tolerance
  - Deviation from 3D trajectory is not within tolerance
- Flowchart

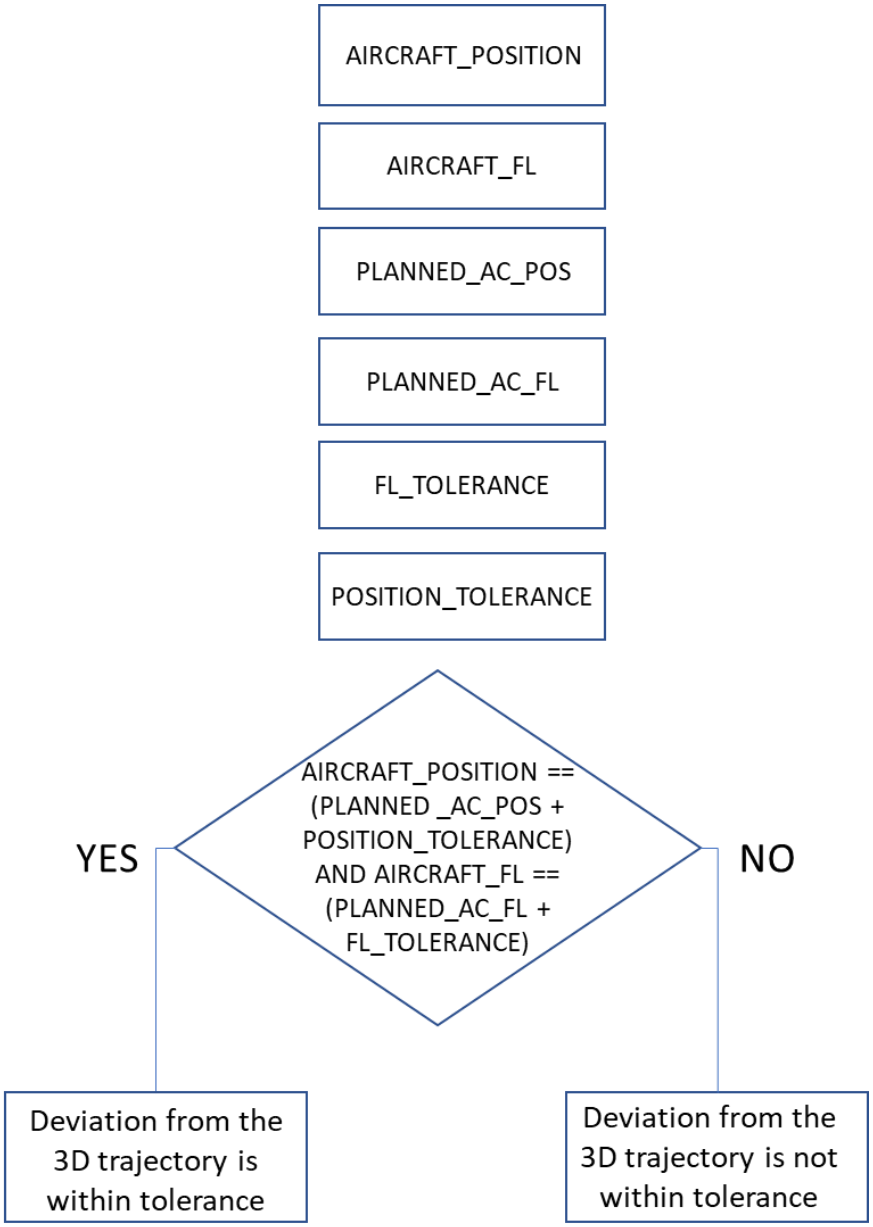



**Figure 19. Task 1.16 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}
```

## 5.1.17 Check that aircraft is following the 4D trajectory

- Input data
  - Task 1.15. Check that aircraft is following the 3D trajectory
  - Aircraft position time (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:positionTime
  - Planned trajectory points (“flight plans”) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:time
- Output data
  - Aircraft is following the 4D trajectory
  - Aircraft is not following the 4D trajectory
- Flowchart

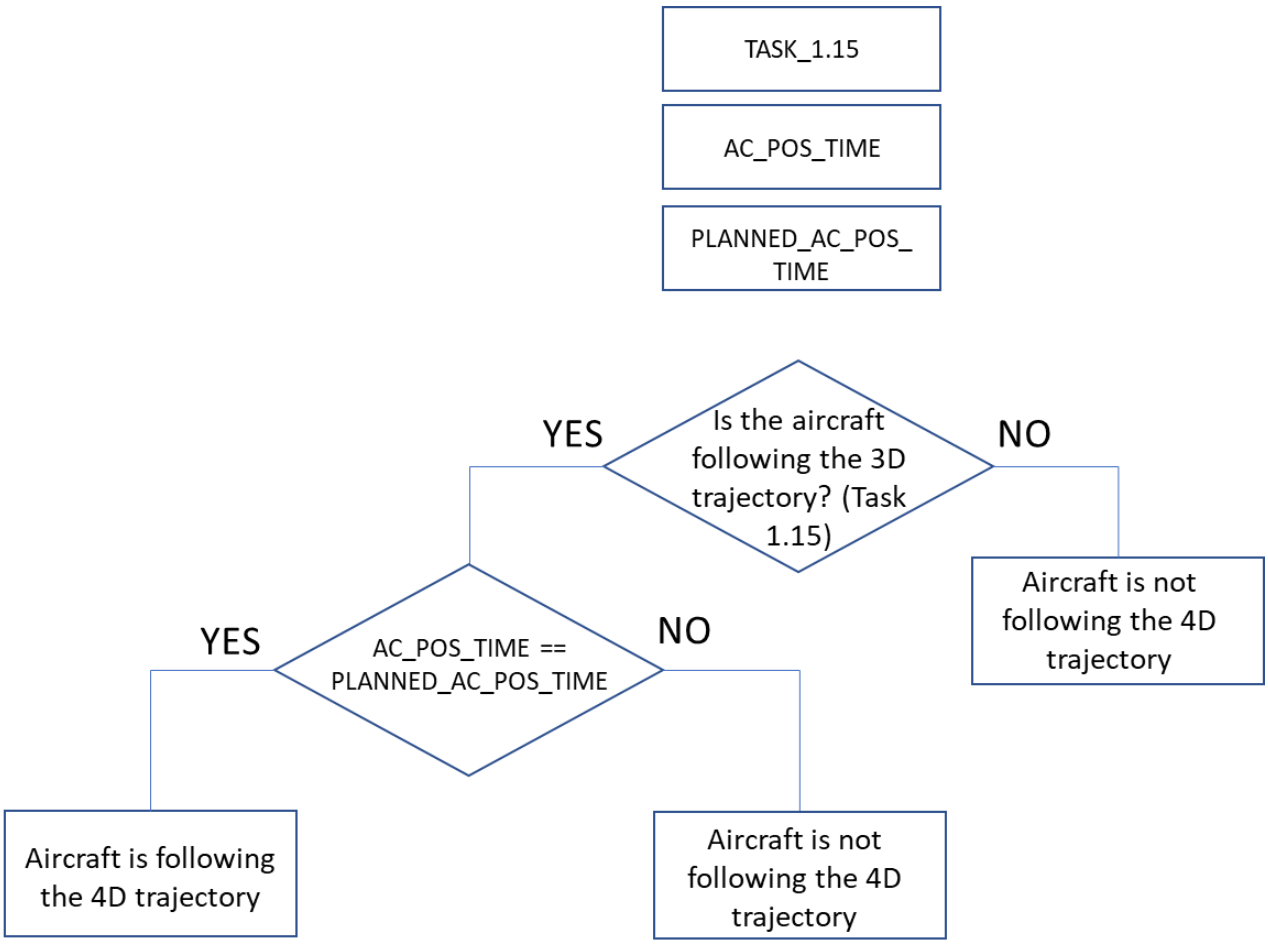



**Figure 20. Task 1.17 flowchart**

- SPARQL query

```
SELECT ?callsign ?positionTime ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
```

plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}

### 5.1.18 Check if the deviation from 4D trajectory is within tolerance

- Input data
  - Task 1.16 Check if the deviation from 3D trajectory is within tolerance
  - Aircraft position time (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:positionTime
  - Planned trajectory points (“flight plans”) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time
  - Time tolerance (±30sec)
- Output data
  - Deviation from 4D trajectory is within tolerance
  - Deviation from 4D trajectory is not within tolerance
- Flowchart

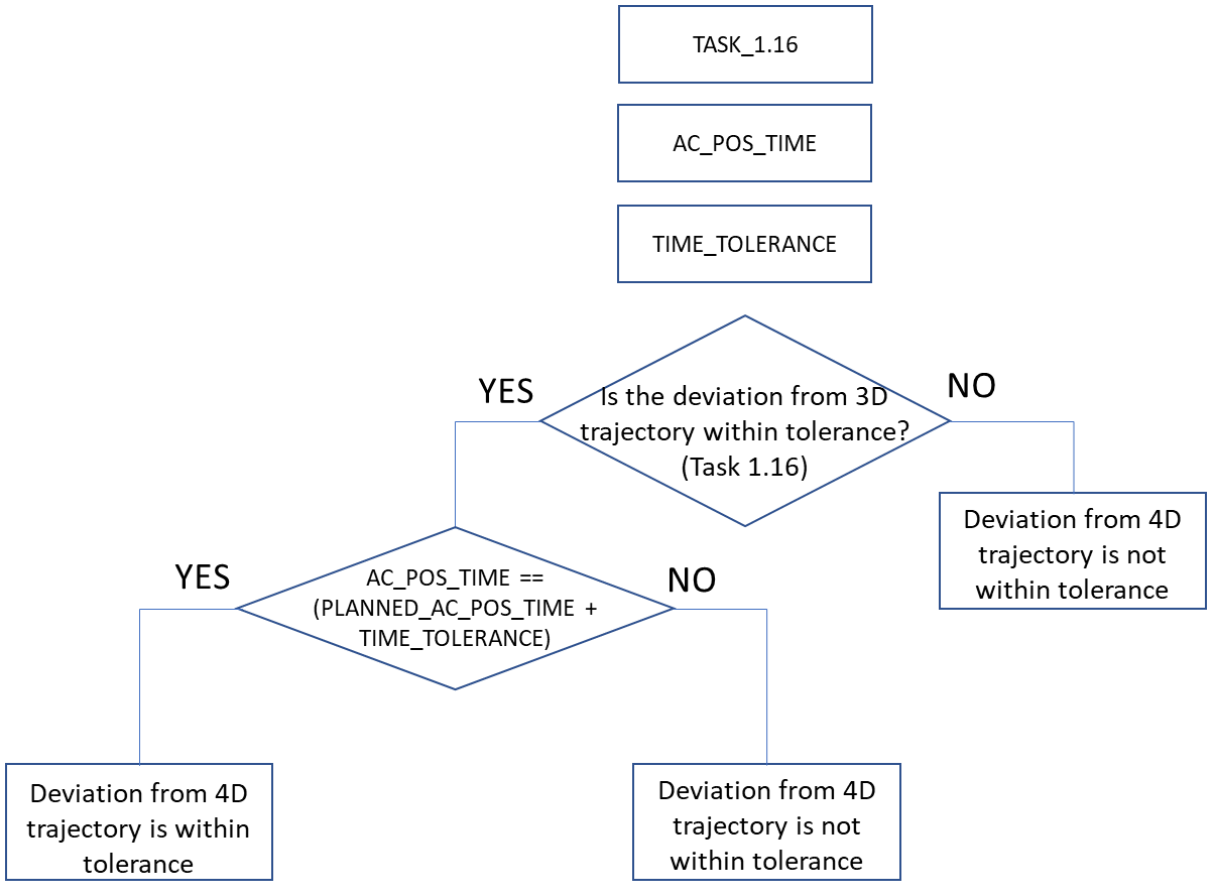



**Figure 21. Task 1.18 flowchart**

- SPARQL query

```
SELECT ?callsign ?positionTime ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}
```

## 5.2 Detect Incoming Planned Flights

Tasks regarding detection of incoming planned flights deal with the possibility of inferring whether or not an aircraft is going to enter the sector whether that be by determining if it is close to the sector boundary or that its altitude matches the sector altitude.

### 5.2.1 Check that aircraft is close to Sector boundary

- Input data
  - Task 3.2 Check that aircraft is planned
  - Sector boundary – horizontal (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Sector boundary - vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - X - Control variable – time to boundary (5 minutes)
  - Y – Control variable – vertical distance from sector boundary (2000ft)
  - Calculation layer
    - a=calculated time to reach sector boundary from current position with current speed
    - b=calculated altitude difference between a/c FL and sector upper/lower limit
- Output data
  - Aircraft is close to sector boundary
  - Aircraft is not close to aircraft boundary
- Flowchart

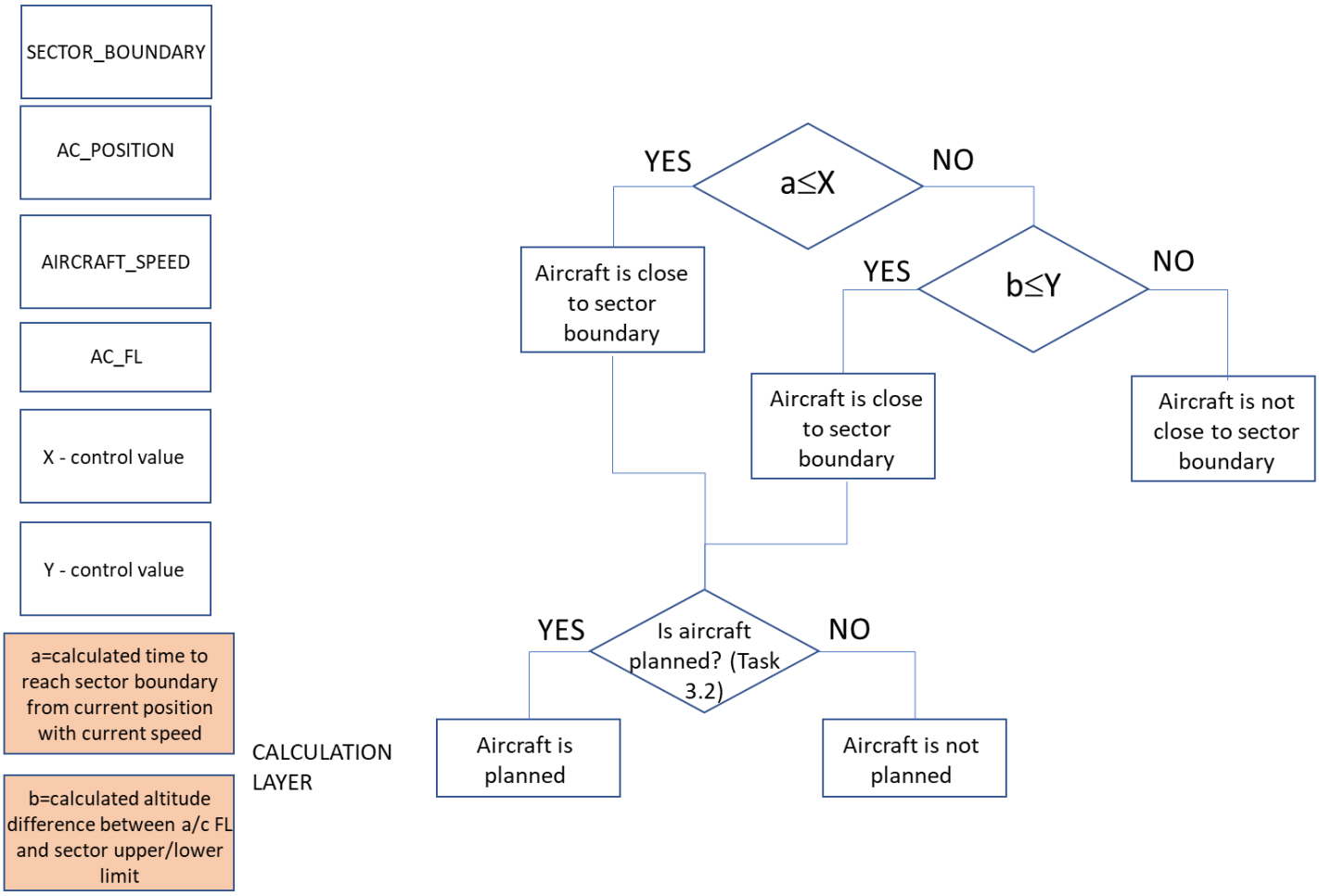



**Figure 22. Task 2.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?aircraftSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
```

```
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

### 5.2.2 Check that aircraft is approaching Sector boundary

- Input data
  - Task 3.2 Check that aircraft is planned
  - Sector boundary – horizontal (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Sector boundary - vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Previous time-step aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Previous time-step aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Aircraft speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
- Output data
  - Aircraft is approaching sector boundary
  - Aircraft is not approaching sector boundary
- Flowchart

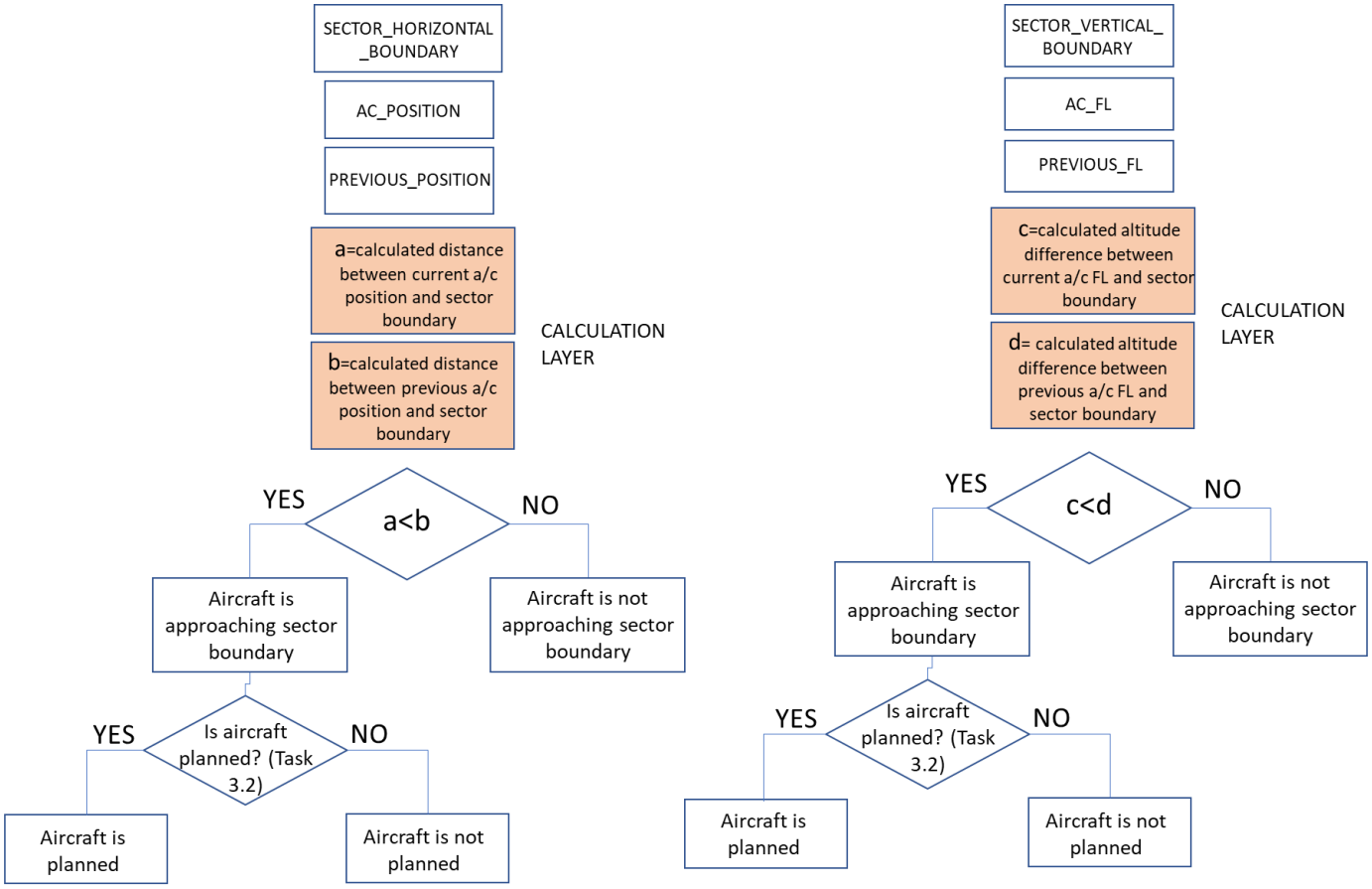



**Figure 23. Task 2.2 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?aircraftSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed .}}

SELECT ?callsign ?previousFL ?previousPosition
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?previousFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?previousPosition .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
```

```
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/gml:
patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value ?horizontalProjection .}}
```

### 5.2.3 Check that aircraft's altitude is within the altitude band of the Sector

- Input data
  - Task 3.2 Check that aircraft is planned
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Sector lower limit (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit
  - Sector upper limit (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:upperLimit
- Output data
  - Aircraft's altitude is within the altitude band of the sector
  - Aircraft's altitude is not within the altitude band of the sector
- Flowchart

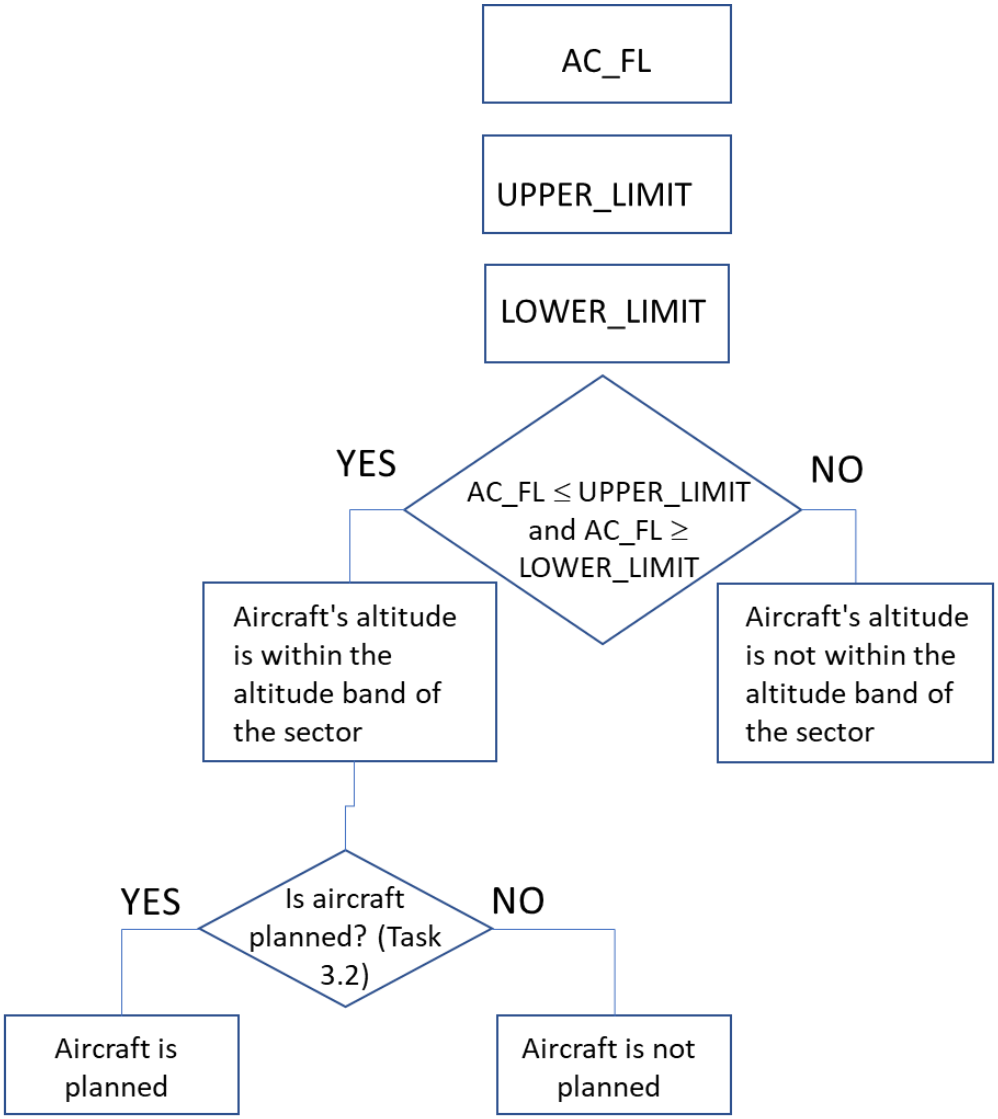



**Figure 24. Task 2.3 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
```

```
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}
```

### 5.2.4 Check that aircraft's altitude is approaching the Sector altitude

- Input data:
  - Task 3.2 Check that aircraft is planned
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Sector boundary – vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
  - Previous time-step flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Current ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data
  - Aircraft's altitude is approaching the Sector altitude
  - Aircraft's altitude is not approaching the Sector altitude
- Flowchart

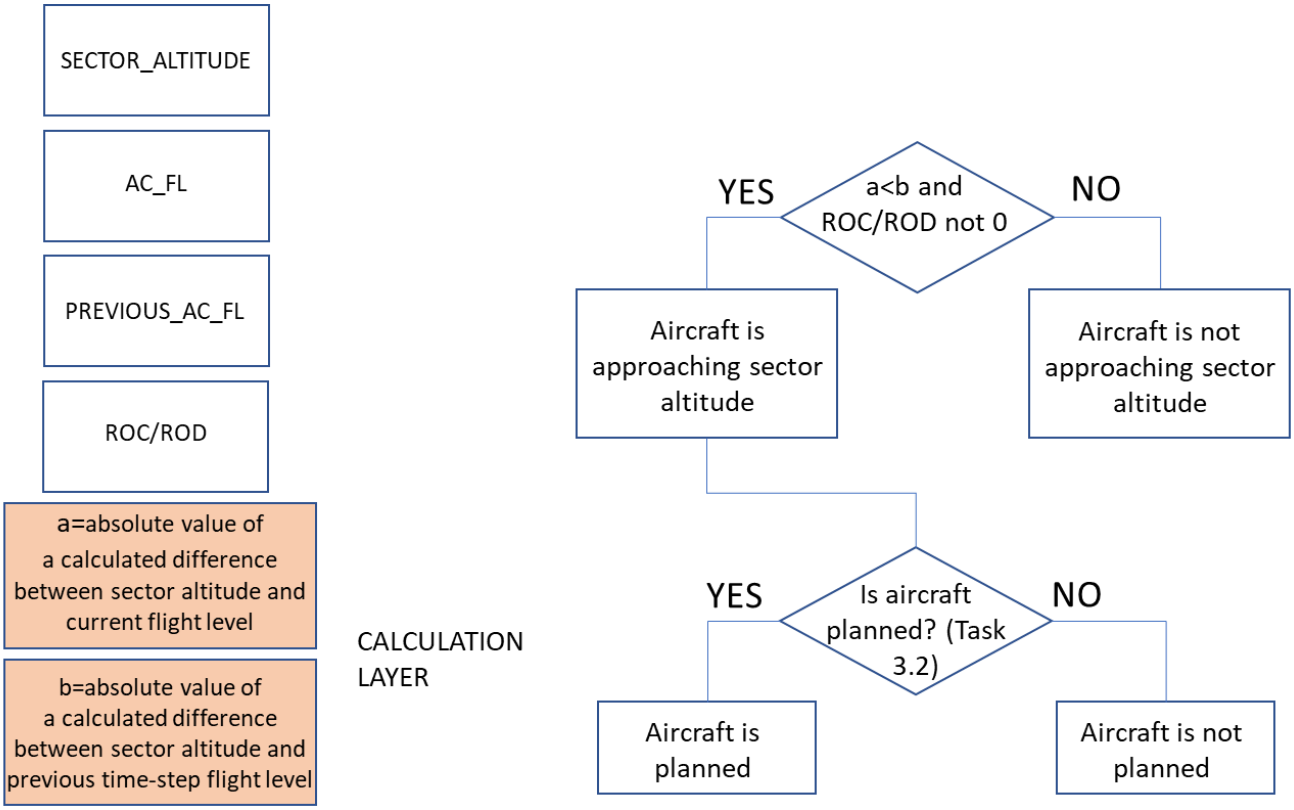



**Figure 25. Task 2.4 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?currentVerticalRate
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate .}}

SELECT ?callsign ?previousFL
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?previousFL .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}
```

## 5.3 Assume, Identify, and Confirm Flight

These tasks try to make sure that the system will be able to identify a flight, confirm it is indeed a planned flight that can be assumed and eventually assume it.

### 5.3.1 Check that aircraft is incoming

- Input data
  - Sector boundary horizontal (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Previous time-step position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Sector vertical boundaries (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Previous aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
- Output data
  - Aircraft is incoming
  - Aircraft is not incoming
    - Aircraft will not enter the sector
    - Aircraft is inside the sector
- Flowchart

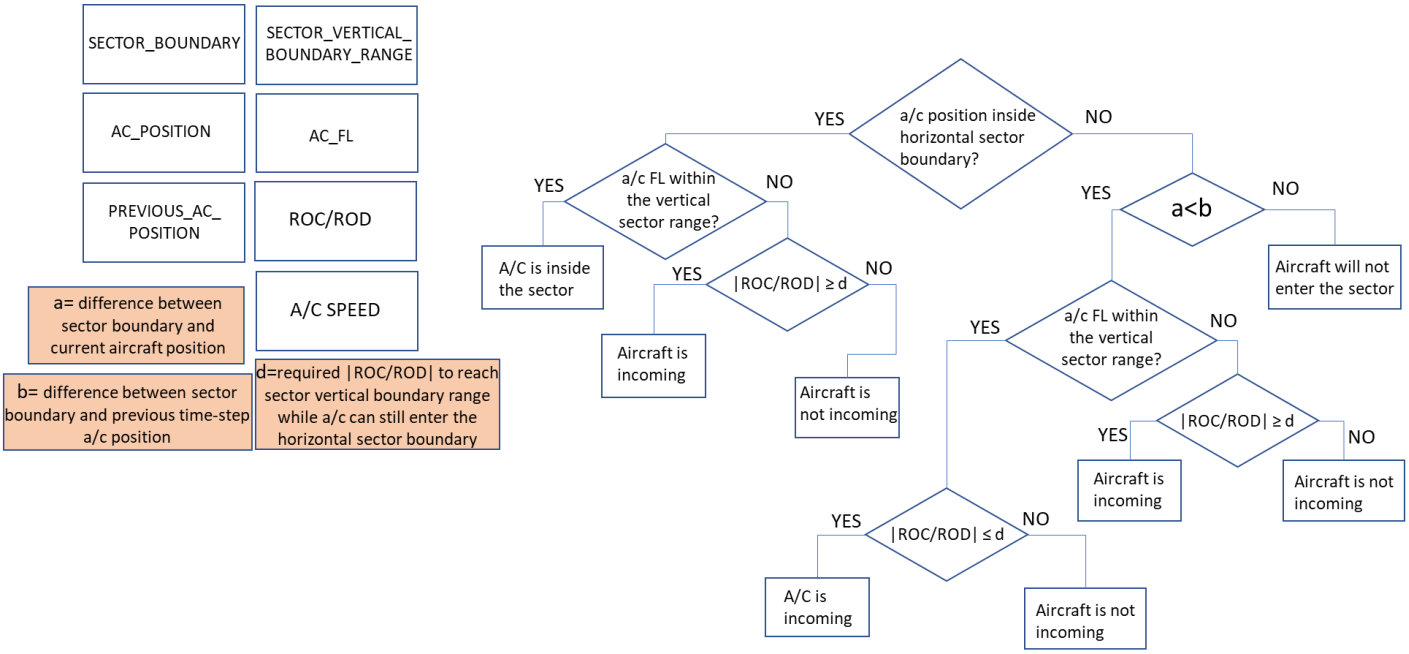



**Figure 26. Task 3.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?currentVerticalRate
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate .}}

SELECT ?callsign ?previousFL ?previousPosition
WHERE { GRAPH ?g0
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?previousFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?previousPosition .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}
```

```
SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

### 5.3.2 Check that aircraft is planned

- Input data
  - Planned route (AC_ROUTE_INFO.ttl)
    - fixm:EfplRoute --> fixm:routeText
  - Flight plan (planned trajectory through the sector) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint
  - Sector boundaries (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
- Output data
  - Aircraft is planned
  - Aircraft is not planned
- Flowchart

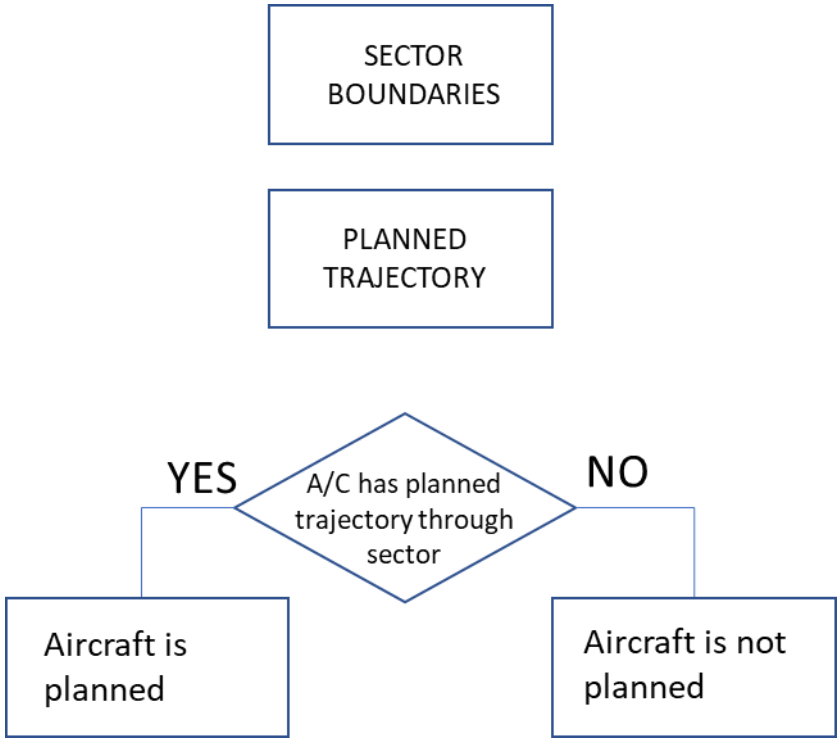



**Figure 27. Task 3.2 flowchart**

- SPARQL queries

```
SELECT ?callsign ?plannedTrajectory ?routeText
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory ;
plain:efplTrajectory/plain:efplRoute/plain:routeText/rdf:value ?routeText .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

### 5.3.3 Check that aircraft has sent the initial call (via datalink)

- Input data
  - Datalink (PLAIN_DATA.ttl)
    - plain:Datalink--> plain:initialCall
- Output data
  - Aircraft has sent the initial call
  - Aircraft didn't send the initial call
- Flowchart

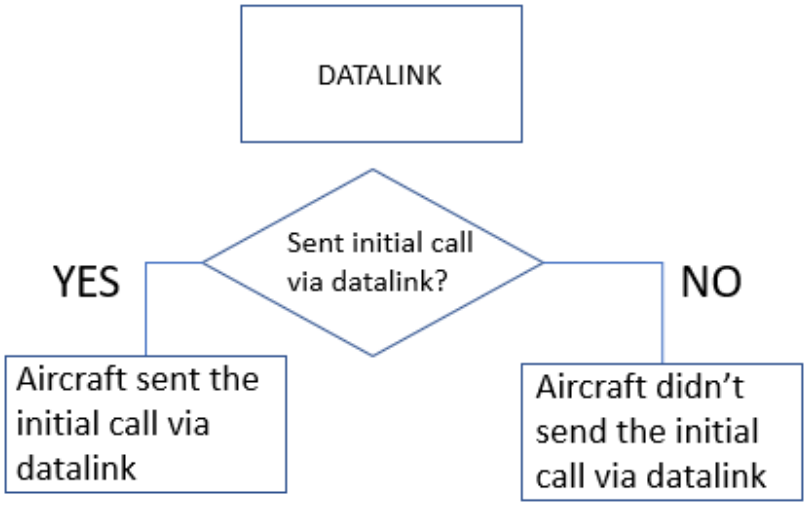



**Figure 28. Task 3.3 flowchart**

- SPARQL query

```
SELECT ?callsign ?initialCall
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:datalink/plain:initialCall/rdf:value ?initialCall .}}
```

### 5.3.4 Confirm that aircraft can be assumed

- Input data
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Sector horizontal boundary (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Sector vertical boundaries (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
  - ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
  - Aircraft speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
  - X - Control variable – time to horizontal boundary determined for transfer (2 min)
  - Y – Control variable – altitude difference from vertical boundary determined for transfer (500 ft)

- Calculation layer:
  - a=time from aircraft position to sector horizontal boundary considering speed
  - b=altitude (FL) difference between current a/c FL and the sector vertical boundary
  - c=the calculated time to climb to reach sector lower limit/descend to reach sector upper limit (considering altitude to climb/descend and ROC/ROD)
  - d=required ROC/ROD to reach sector vertical boundary range while a/c can still enter the horizontal sector boundary

- Output data
  - Aircraft can not be assumed
    - Notify the controller when aircraft is already in the sector
  - Aircraft can be assumed
    - Special case is when aircraft will exit the horizontal boundary by the time it climbs or descends to meet the vertical boundary criteria

- Flowchart

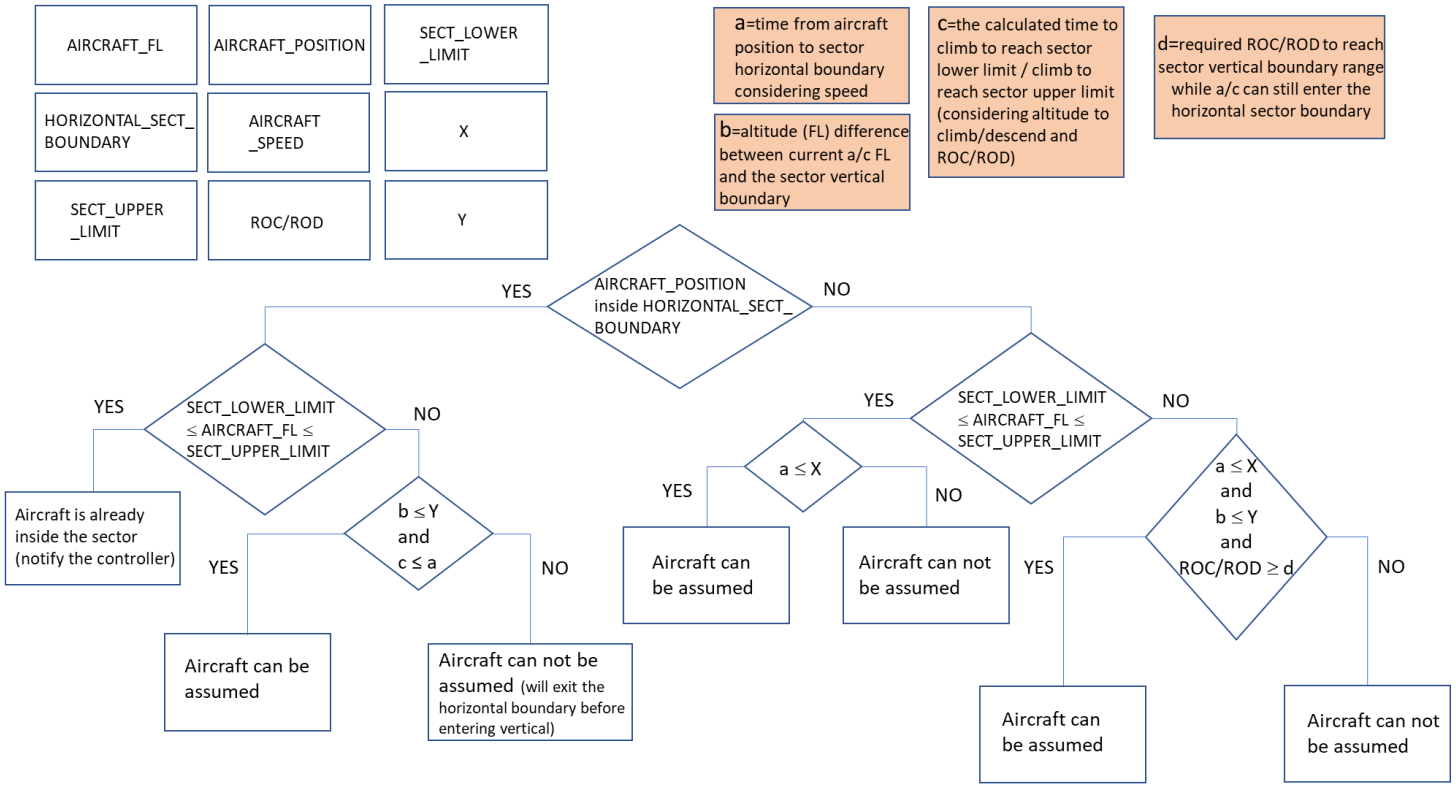



**Figure 29. Task 3.4 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?currentVerticalRate ?aircraftSpeed
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
```

```
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

## 5.4 Assess if Exit Conditions are Met

When trying to determine whether an aircraft meets all the exit conditions different criteria need to be met, such as expected time on the exit point, as well as the required FL.

### 5.4.1 Check that aircraft is flying towards the exit point

- Input data
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Heading (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:track
  - Exit point position (ENTRY_EXIT_POINTS.ttl)
    - aixm:Point --> gml:pos
    - aixm:Point --> aixm:type
  - Position tolerance (±2.5NM)
- Output data
  - Aircraft is flying towards the exit point
  - Aircraft isn't flying towards the exit point
- Flowchart

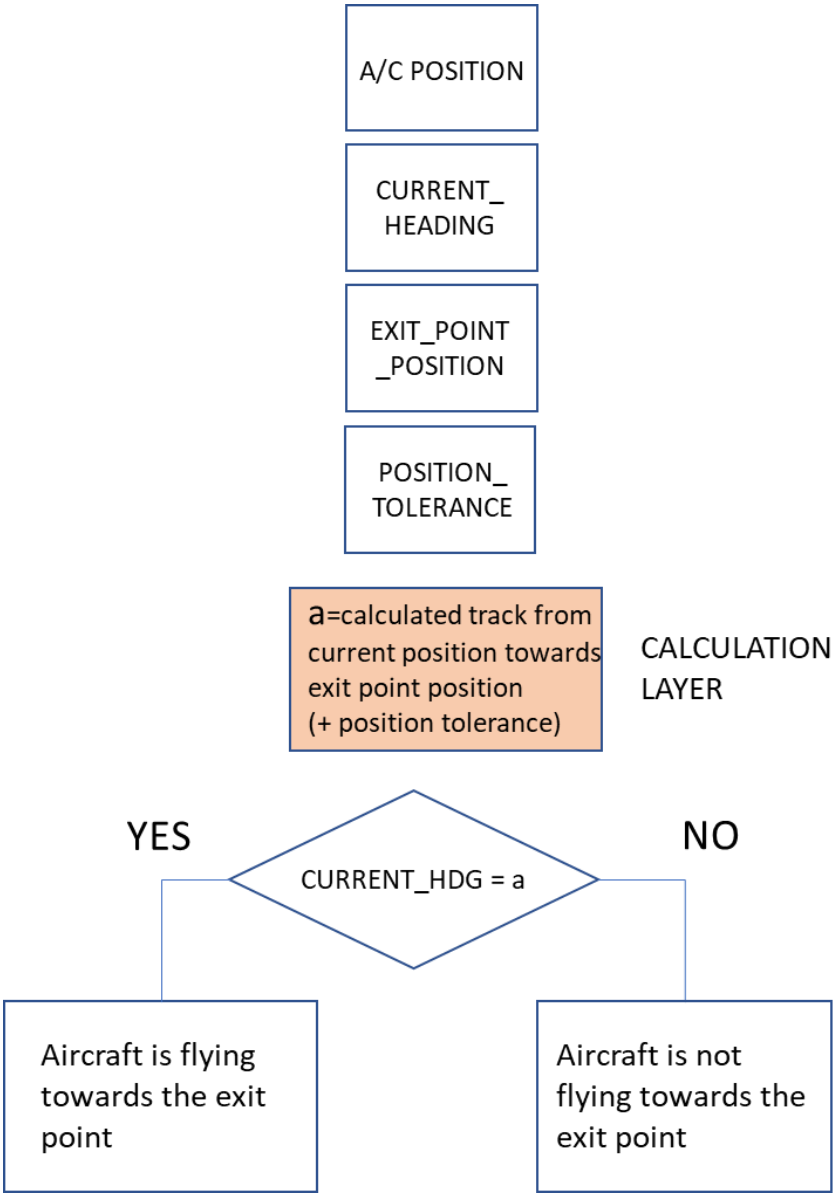



**Figure 30. Task 4.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?currentHeading
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading .}}

SELECT ?name ?pointType ?pointPosition
WHERE { GRAPH ?default
{?ExitPoint a plain:SignificantPointInAirspace ;
plain:type/rdf:value ?pointType ;
```

```
plain:location/gml:pos/rdf:value ?pointPosition
BIND(STRAFTER(STR(?ExitPoint),"https://www.plain.aero/releases/SESAR_Ext-1.0/RDF.xml#") AS
?name) }}
```

### 5.4.2 Check that aircraft will reach the exit point on the required FL

- Input data
  - FLAS (Flight Level Allocation Scheme) - altitude for exit coordination point (PLAIN_DATA.ttl)
    - plain:Altitude --> plain:flightLevel
  - Cleared FL (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel
  - Exit point position (ENTRY_EXIT_POINTS.ttl)
    - aixm:Point --> gml:pos
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:VerticalRate
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft speed (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:actualSpeed
  - Calculation layer:
    - a=distance to climb/descend (difference between current and exit FL)
    - b=time to climb (using the difference between current position and exit point position and a/c speed)
- Output data
  - Aircraft will reach the exit point on the required FL
  - Aircraft will not reach the exit point on the required FL
- Flowchart

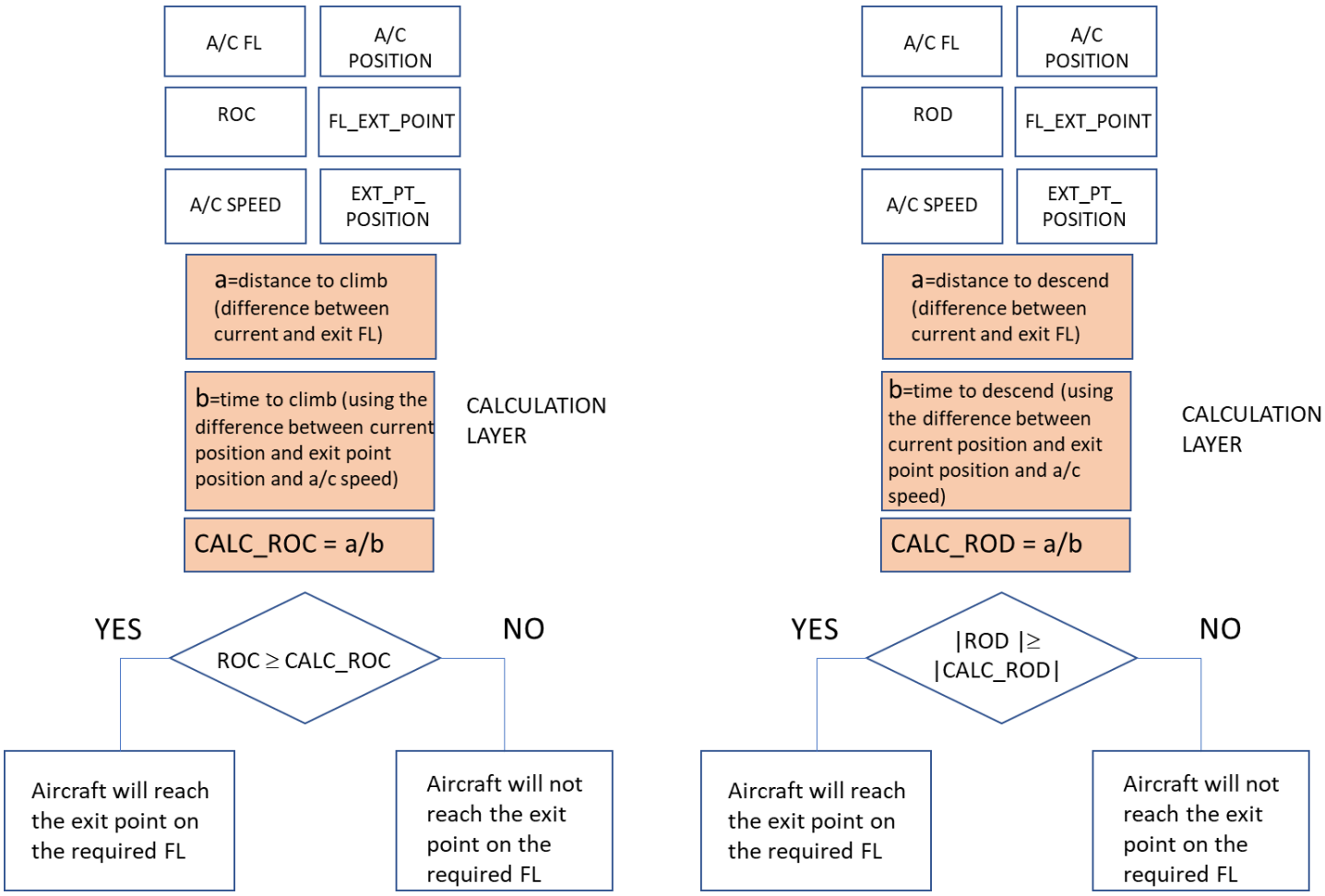



**Figure 31. Task 4.2 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftFL ?aircraftPosition ?currentVerticalRate ?aircraftSpeed ?clearedFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:cleared/plain:clearedFlightLevel/rdf:value ?clearedFL .}}

SELECT ?exitPoint ?exitPointFL
WHERE { GRAPH ?default
{?FlightIdentification a plain:FlightIdentification ;
plain:route/plain:coordinationPoint/rdf:value ?exitPoint ;
plain:route/plain:altitude/plain:flightLevel/rdf:value ?exitPointFL .}}
```

### 5.4.3 Check that aircraft will reach the exit point at the expected time

- Input data
  - A/C routes (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time
  - Aircraft current position and time at position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
    - plain:AircraftPosition --> plain:positionTime
  - Exit point position (ENTRY_EXIT_POINTS.ttl)
    - aixm:Point --> gml:pos
    - aixm:Point --> aixm:type
  - Task 1.18 Check if deviation from 4D trajectory is within tolerance
  - Position time tolerance (±30 sec)
- Output data
  - Aircraft will reach the exit point at the expected time
  - Aircraft will not reach the exit point at the expected time
  - ML (Machine Learning) trajectory prediction module was successful
  - ML trajectory prediction module was not successful
- Flowchart

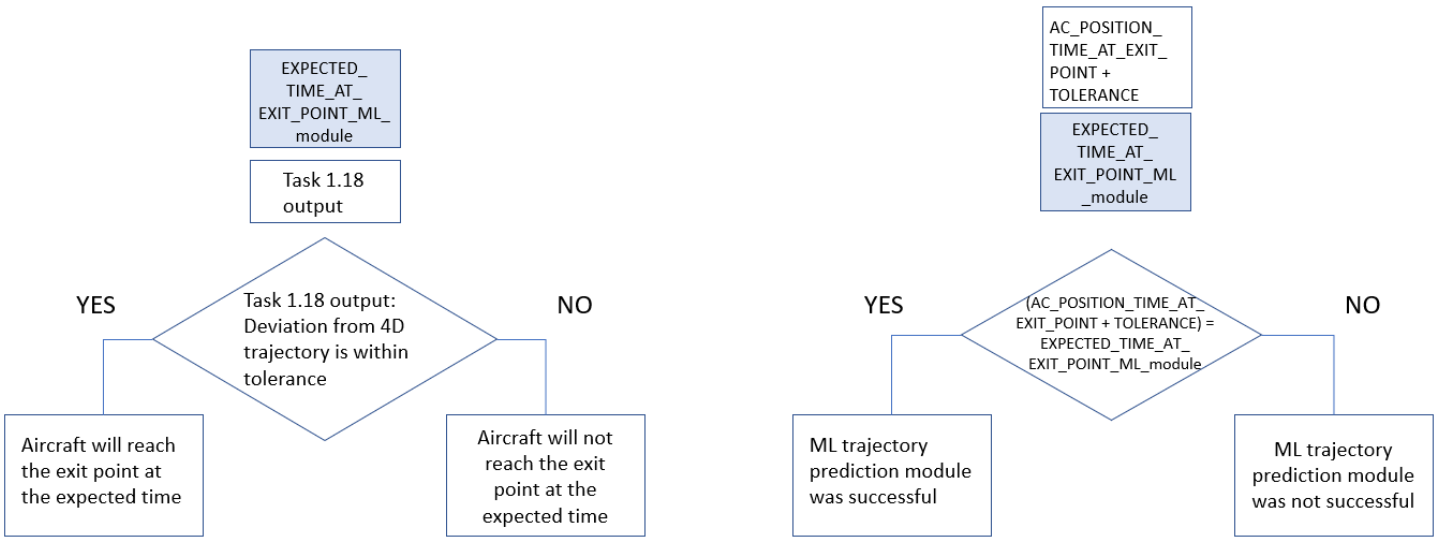



**Figure 32. Task 4.3 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?aircraftSpeed ?positionTime ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
```

```
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}

SELECT ?name ?pointType ?pointPosition
WHERE { GRAPH ?default
{?ExitPoint a plain:SignificantPointInAirspace ;
plain:type/rdf:value ?pointType ;
plain:location/gml:pos/rdf:value ?pointPosition
BIND(STRAFTER(STR(?ExitPoint),"https://www.plain.aero/releases/SESAR_Ext-1.0/RDF.xml#") AS
?name)
FILTER (?pointType = "EXIT" || ?pointType = "BOTH") }}
```

## 5.5 Conflict Management

Conflict management tasks rely on the conflict detection module created by Universidad Politécnica de Madrid (UPM) that is able to detect potential conflict using the concept of Situation of Interest (SI). One SI is when an aircraft pair is expected to intersect with a horizontal separation lower than a pre-defined separation and infringe the vertical separation minima. The horizontal value used is 10 NM, while the vertical separation is 1000 ft.

### 5.5.1 Check all aircraft pairs for conflict (ML module)

- Input data
  - Aircraft data - current a/c position, heading, altitude, speed and ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
    - fixm:AircraftPosition --> fixm:track
    - fixm:AircraftPosition --> fixm:flightLevel
    - fixm:AircraftPosition --> fixm:actualSpeed and fixm:VerticalRate
  - Trajectory (4D) – horizontal (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint
  - Trajectory (4D) – vertical (AC_ROUTE_INFO.ttl)
    - fixm:EfplPoint4D --> fixm:flightLevel
  - Conflict detection for a/c pairs (calculation layer)
  - X - Control variable – distance from other aircraft - 10 NM horizontally
  - Y - Control variable – distance from other aircraft - 1000 ft vertically
- Output data
  - All aircraft pairs are checked for conflict
- Flowchart

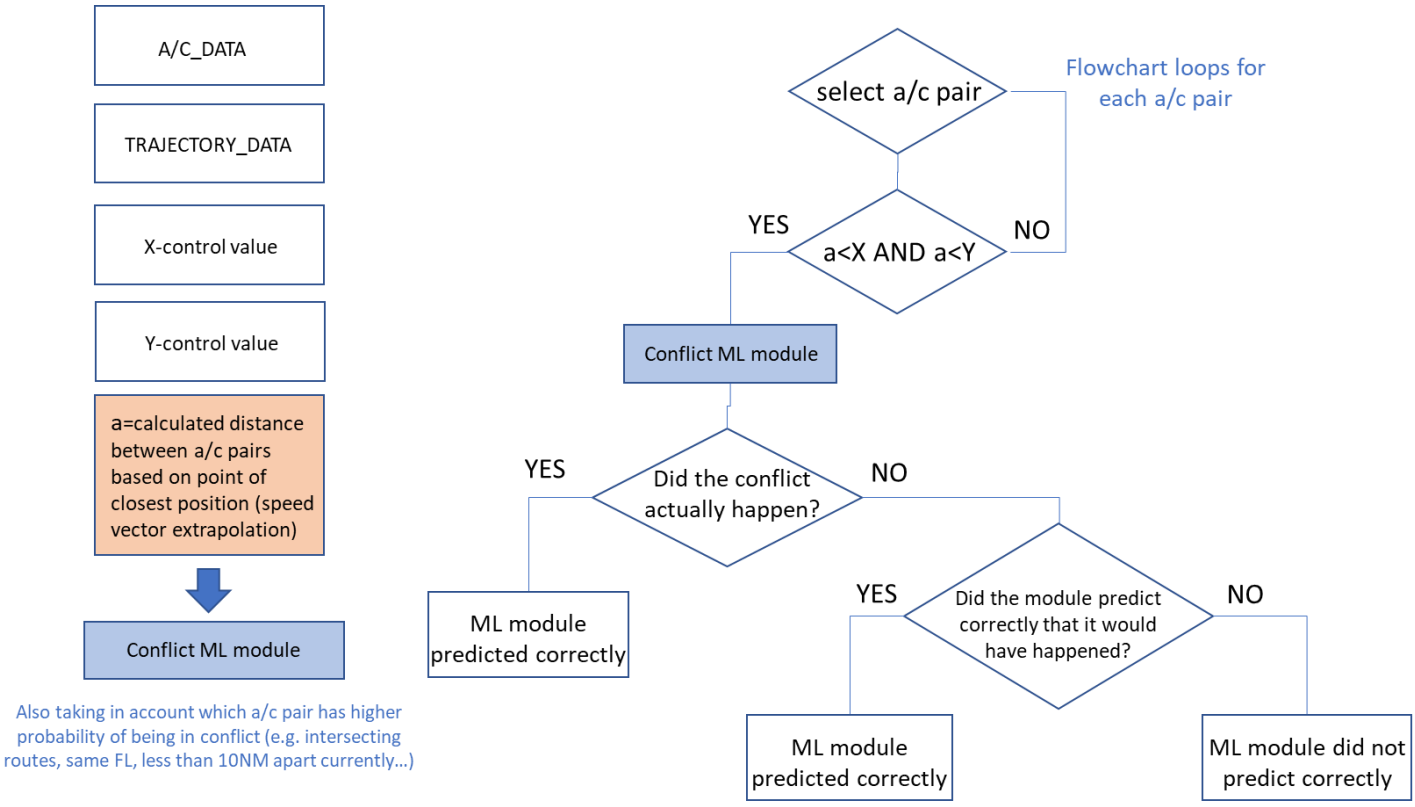



**Figure 33. Task 5.1 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircraftSpeed ?currentHeading ?currentVerticalRate
?positionTime ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}
```

### 5.5.2 Check plausibility of the predicted conflicts

- Input data
  - Aircraft data - current a/c position, heading, flight level, speed and ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
    - fixm:AircraftPosition --> fixm:track

        - fixm:AircraftPosition - fixm:flightLevel
        - fixm:AircraftPosition --> fixm:actualSpeed and fixm:VerticalRate

    - Cleared flight information (CLEARED_FLIGHT_INFORMATION.ttl)
        - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel
        - plain:clearedSpeed
        - plain:heading
        - plain:offtrackClearance
        - plain:rateOfClimbDescent
        - plain:directRouting

    - Aircraft performance data (PLAIN_DATA.ttl)
        - plain:Performance --> plain:performanceAltitude
        - plain:trueAirSpeed
        - plain:machNumber
        - plain:rateOfClimb
        - plain:rateOfDescent

    - Trajectory (4D) – horizontal, vertical and temporal (AC_ROUTE_INFO.ttl)
        - fixm:Trajectory --> fixm:trajectoryPoint
        - fixm:EfplPoint4D --> fixm:flightLevel
        - fixm:EfplPoint4D --> fixm:time

    - Conflict detection for a/c pairs (calculation layer)

    - X - Control variable – distance from other aircraft - 10 NM horizontally

    - Y - Control variable – distance from other aircraft - 1000 ft vertically

- Output data
    - Calculated plausibility of predicted conflicts

- Flowchart

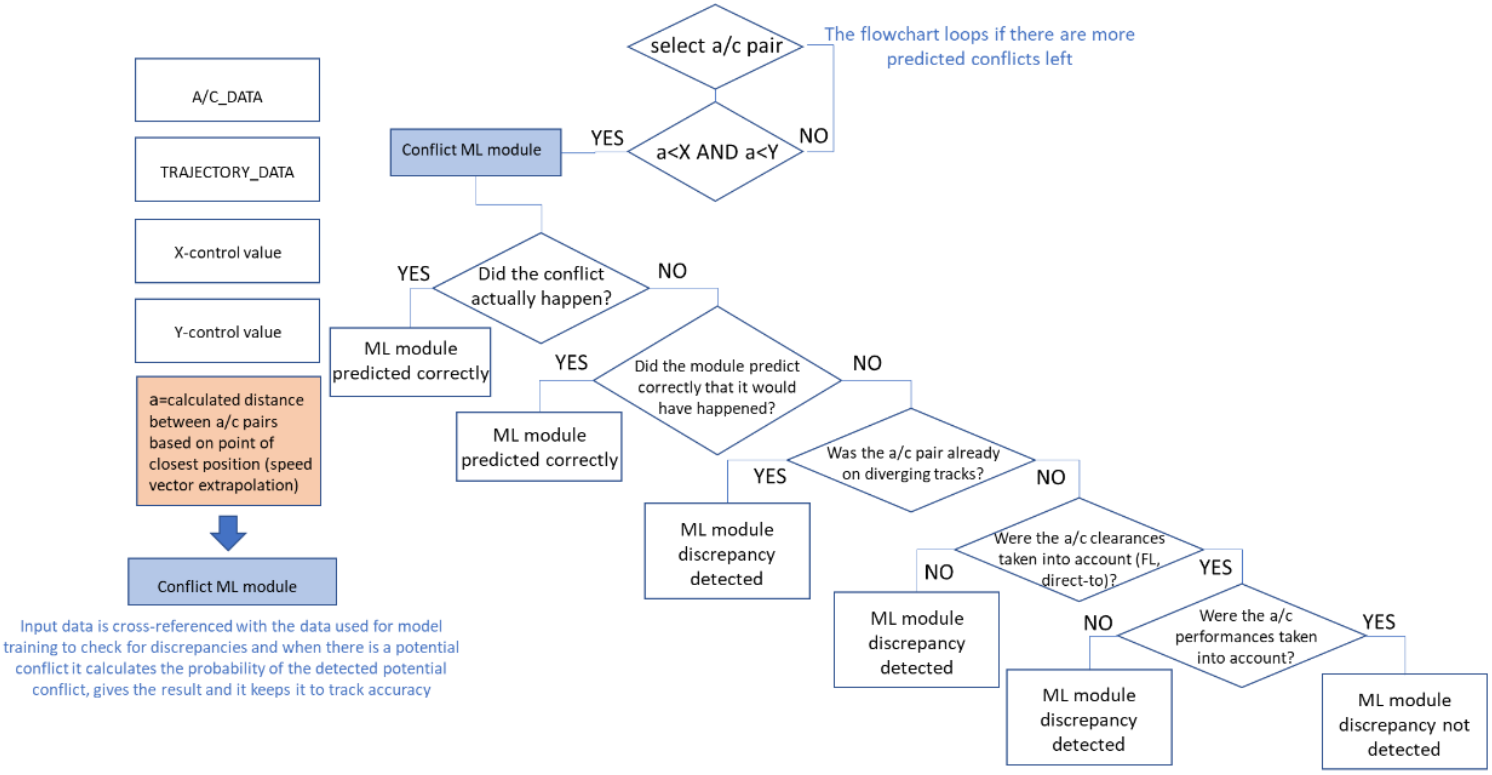



**Figure 34. Task 5.2 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircraftSpeed ?currentHeading ?currentVerticalRate
?positionTime ?plannedTrajectory ?clearedFL ?clearedSpeed ?clearedHeading
?offtrackClearance ?clearedVerticalRate ?directFrom ?directTo ?performanceAltitude
?trueAirSpeed ?machNumber ?rateOfClimb ?rateOfDescent
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory ;
plain:flightIdentification/plain:cleared/plain:clearedFlightLevel/rdf:value ?clearedFL ;
plain:flightIdentification/plain:cleared/plain:clearedSpeed/rdf:value ?clearedSpeed ;
plain:flightIdentification/plain:cleared/plain:heading/rdf:value ?clearedHeading ;
plain:flightIdentification/plain:cleared/plain:offtrackClearance/rdf:value ?offtrackClearance ;
plain:flightIdentification/plain:cleared/plain:rateOfClimbDescent/rdf:value ?clearedVerticalRate ;
plain:flightIdentification/plain:cleared/plain:directRouting/plain:from/rdf:value ?directFrom ;
```

plain:flightIdentification/plain:cleared/plain:directRouting/plain:to/rdf:value ?directTo .
?FlightIdentification a plain:FlightIdentification ;
plain:performance/plain:performanceAltitude/rdf:value ?performanceAltitude ;
plain:performance/plain:trueAirSpeed/rdf:value ?trueAirSpeed ;
plain:performance/plain:machNumber/rdf:value ?machNumber ;
plain:performance/plain:rateOfClimb/rdf:value ?rateOfClimb ;
plain:performance/plain:rateOfDescent/rdf:value ?rateOfDescent .}}

### 5.5.3 Check which conflicts are to occur within the sector

- Input data
  - Aircraft data - current a/c position, heading, flight level, speed and ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
    - fixm:AircraftPosition --> fixm:track
    - fixm:AircraftPosition --> fixm:flightLevel
    - fixm:AircraftPosition --> fixm:actualSpeed and fixm:VerticalRate
  - Aircraft performance data (PLAIN_DATA.ttl)
    - plain:Performance --> plain:performanceAltitude
    - plain:trueAirSpeed
    - plain:machNumber
    - plain:rateOfClimb
    - plain:rateOfDescent
  - Sector boundary – horizontal and vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Trajectory (4D) – horizontal, vertical and temporal (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint
    - fixm:EfplPoint4D --> fixm:flightLevel
    - fixm:EfplPoint4D --> fixm:time
  - Conflict detection for a/c pairs (calculation layer)
  - X - Control variable – distance from other aircraft - 10 NM horizontally
  - Y - Control variable – distance from other aircraft - 1000 ft vertically
- Output data
  - Conflicts that will occur within the sector are checked
- Flowchart

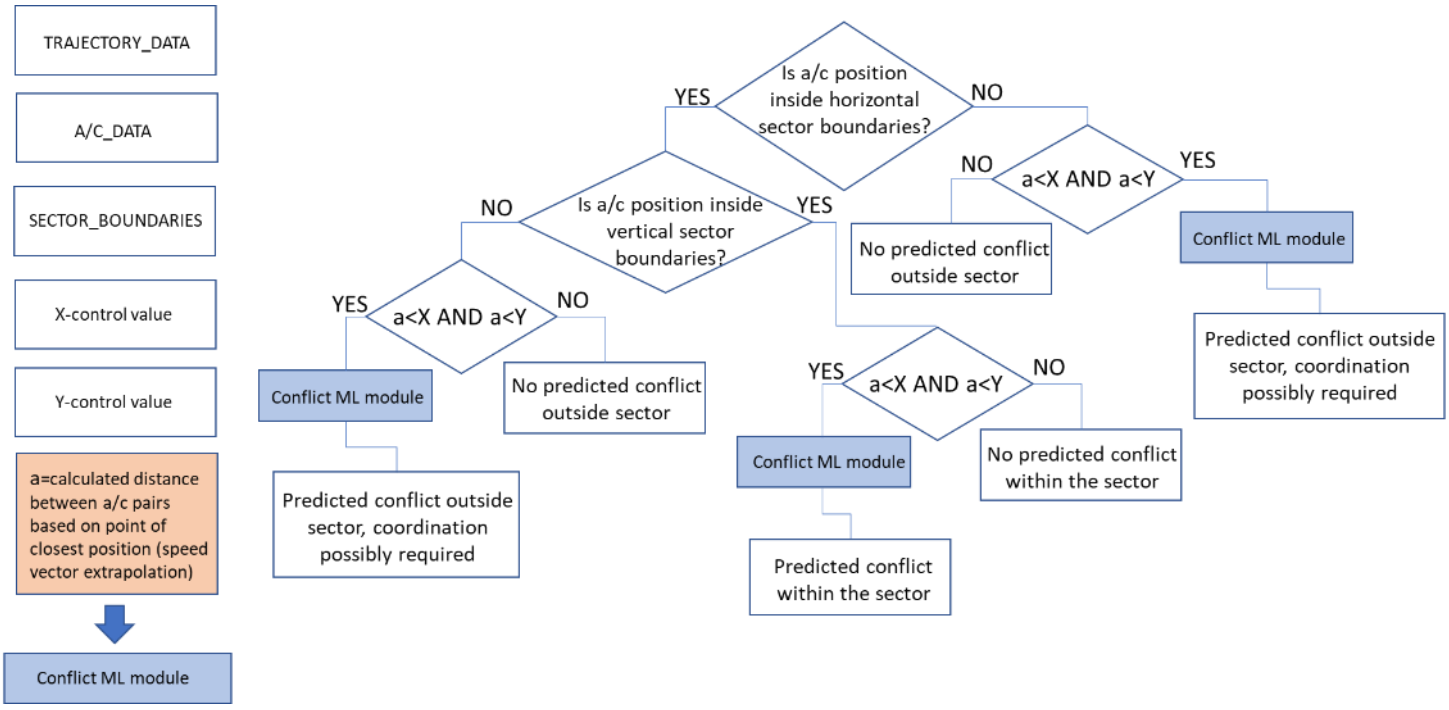



**Figure 35. Task 5.3 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircaftSpeed ?currentHeading ?currentVerticalRate
?positionTime ?plannedTrajectory ?performanceAltitude ?trueAirSpeed ?machNumber
?rateOfClimb ?rateOfDescent
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .
?FlightIdentification a plain:FlightIdentification ;
plain:performance/plain:performanceAltitude/rdf:value ?performanceAltitude ;
plain:performance/plain:trueAirSpeed/rdf:value ?trueAirSpeed ;
plain:performance/plain:machNumber/rdf:value ?machNumber ;
plain:performance/plain:rateOfClimb/rdf:value ?rateOfClimb ;
plain:performance/plain:rateOfDescent/rdf:value ?rateOfDescent .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
```

```
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

### 5.5.4 Rank conflicts based on urgency

- Input data
  - Aircraft data - current a/c position, heading, flight level, speed and ROC/ROD (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
    - fixm:AircraftPosition --> fixm:track
    - fixm:AircraftPosition --> fixm:flightLevel
    - fixm:AircraftPosition --> fixm:actualSpeed and fixm:VerticalRate
  - Trajectory (4D) – horizontal, vertical and temporal (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint
    - fixm:EfplPoint4D --> fixm:flightLevel
    - fixm:EfplPoint4D --> fixm:time
  - X - Control variable – distance from other aircraft - 10 NM horizontally
  - Y - Control variable – distance from other aircraft - 1000 ft vertically
- Output data
  - Urgency ranking of conflicts

- Flowchart

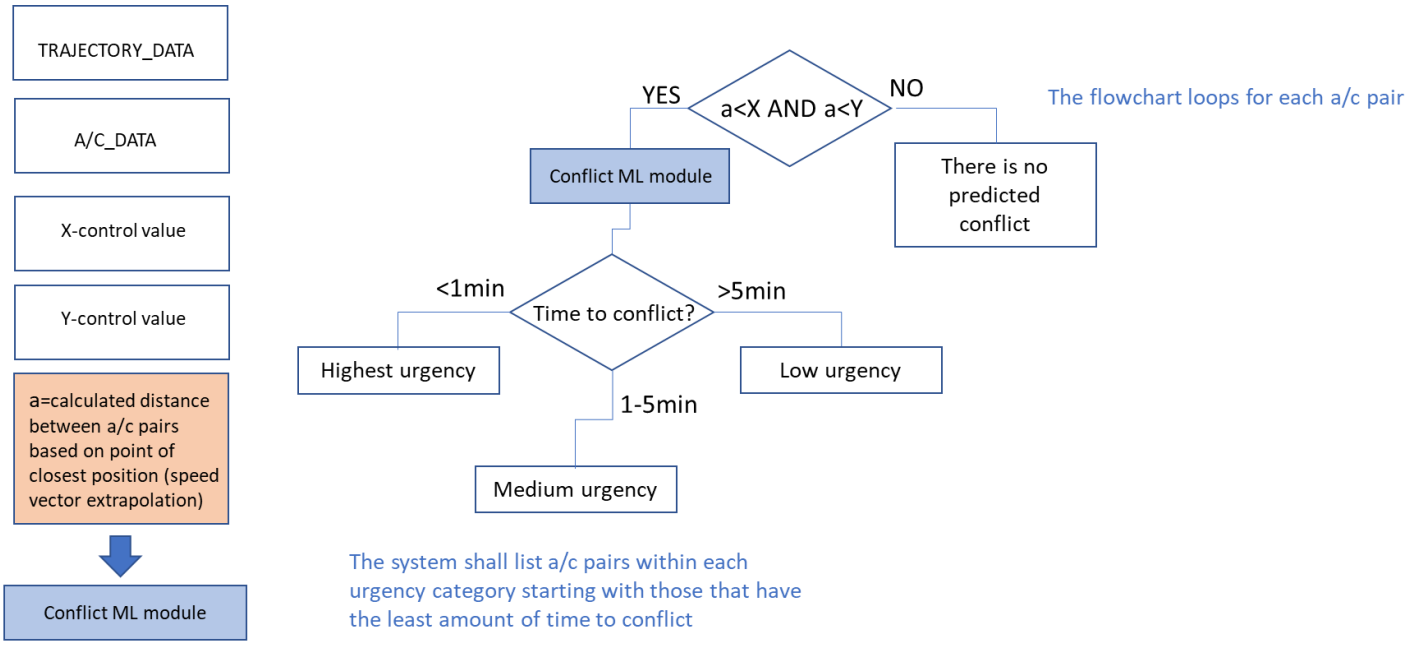



**Figure 36. Task 5.4 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircraftSpeed ?currentHeading ?currentVerticalRate
?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}
```

## 5.6 Execute Aircraft's Plan

It is important to detect when an aircraft needs to climb/descend or if certain corrections to its route need to be made in order to avoid restricted areas.

### 5.6.1 Detect aircraft that have to climb/descend to requested FL

- Input data
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
    - fixm:Route --> fixm:requestedAltitude

- Output data
  - Aircraft has to climb to requested FL
  - Aircraft has to descend to requested FL
  - Aircraft is at the requested FL

- Flowchart

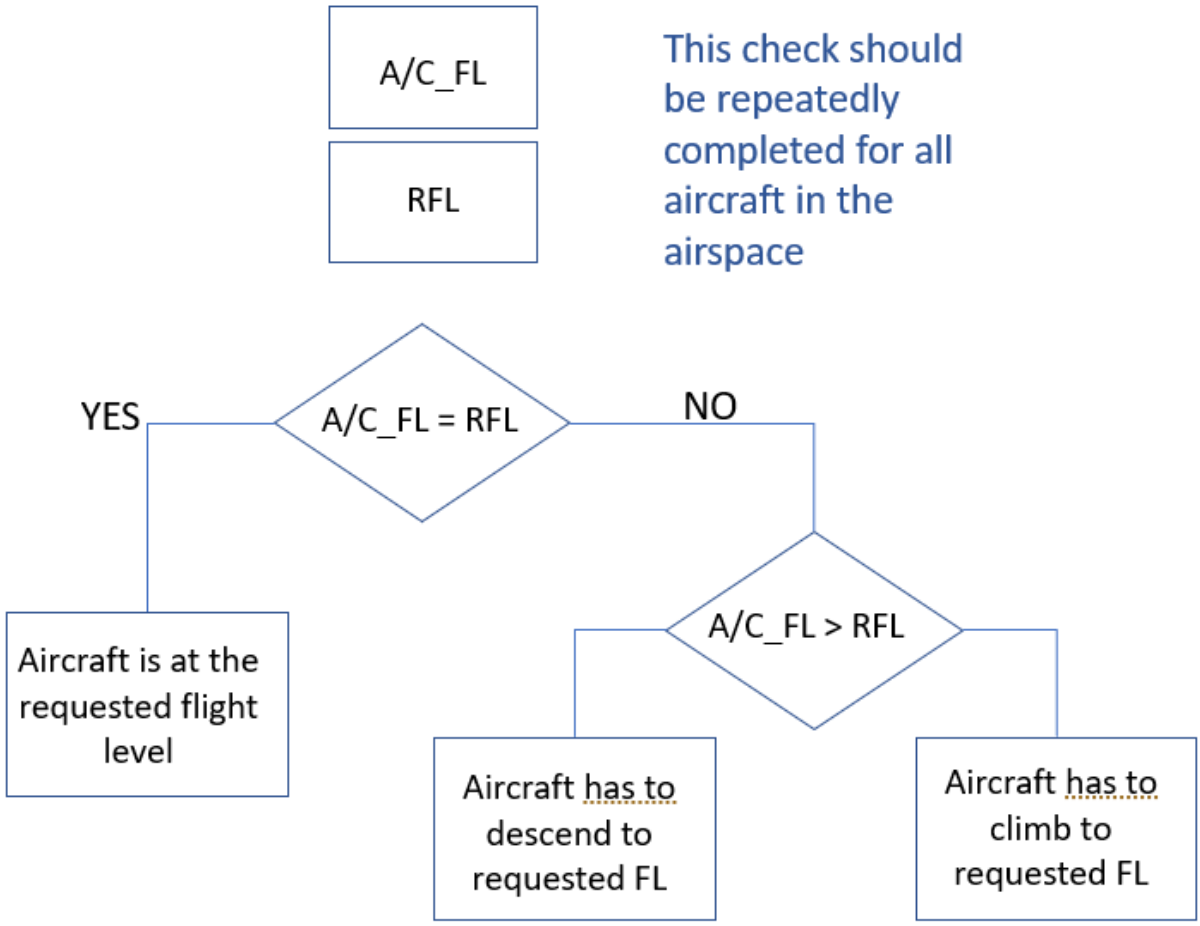



**Figure 37. Task 6.1 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftFL ?requestedFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:route/plain:requestedAltitude/rdf:value ?requestedFL .}}
```

### 5.6.2 Detect aircraft that have to climb/descend to exit FL

- Input data
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel

- FLAS/LOAS (Letters of Agreement) - to know which is the exit FL (PLAIN_DATA.ttl)
  - plain:Altitude --> plain:flightLevel
- Cleared FL – if there is no FLAS (CLEARED_FLIGHT_INFORMATION.ttl)
  - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel

- Output data
  - Aircraft has to climb to exit FL
  - Aircraft has to descend to exit FL
  - Aircraft is at the exit FL

- Flowchart

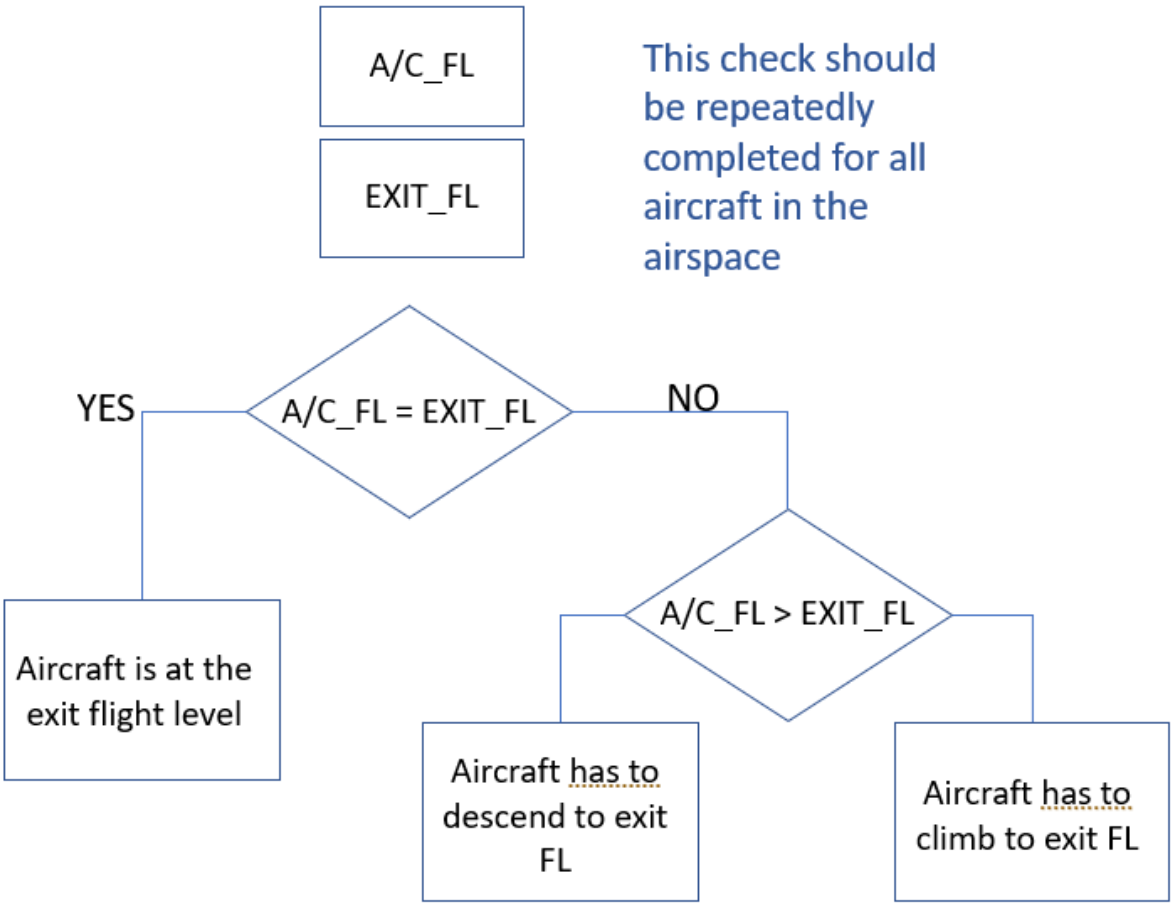



**Figure 38. Task 6.2 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftFL ?clearedFL
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:cleared/plain:clearedFlightLevel/rdf:value ?clearedFL .}}
```

```
SELECT ?exitPoint ?exitFL
WHERE { GRAPH ?default
{?FlightIdentification a plain:FlightIdentification ;
plain:route/plain:coordinationPoint/rdf:value ?exitPoint ;
plain:route/plain:altitude/plain:flightLevel/rdf:value ?exitFL .}}
```

### 5.6.3 Detect aircraft that will reach top of descent within the Sector (ML module)

- Input data
  - 4DT (4-Dimensional Trajectory) ML module
- Output data
  - Aircraft will reach top of descent within the sector
  - Aircraft will not reach top of descent within the sector
- Flowchart

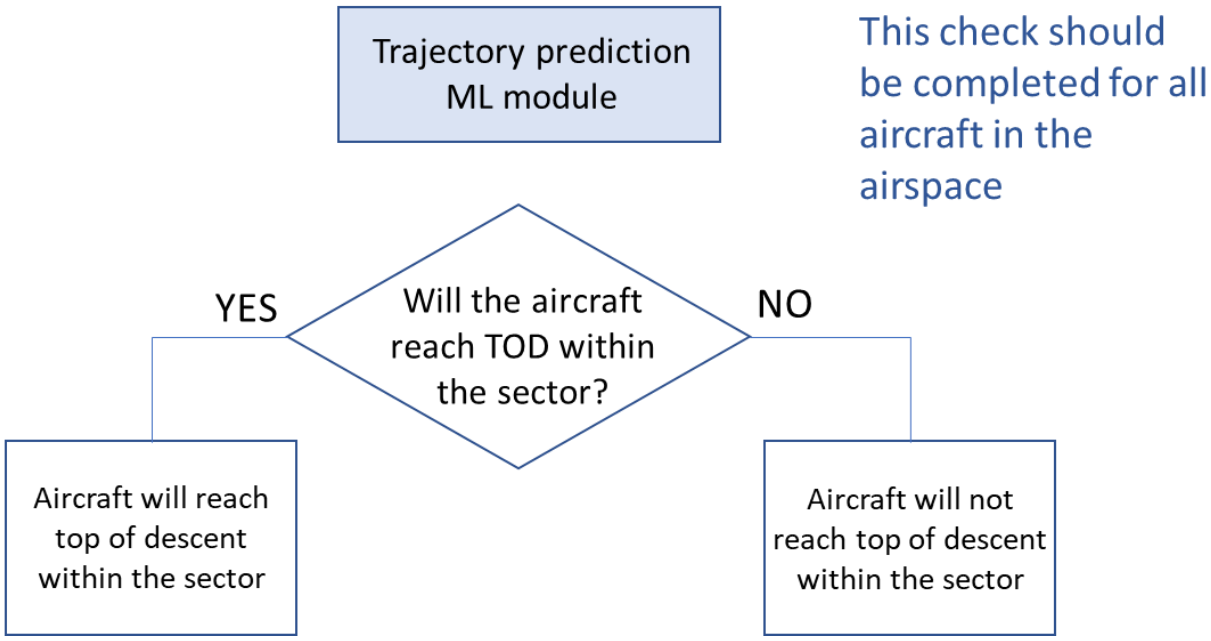



**Figure 39. Task 6.3 flowchart**

- SPARQL query
  - a query is not necessary as all the data comes from the ML module

### 5.6.4 Detect if planned trajectory passes through restricted airspace

- Input data
  - Restricted airspace (horizontal) (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceVolume --> aixm:horizontalProjection --> aixm:Surface



  - Restricted airspace (vertical) (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceVolume --> aixm:lowerLimit
    - aixm:AirspaceVolume --> aixm:upperLimit
  - Restricted airspace activation (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceActivation --> aixm:timeInterval
    - aixm:startTime
    - aixm:endTime
    - aixm:AirspaceActivation --> aixm:status
  - Flight plan (NEST - Network strategic tool) – horizontal (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint
  - Flight plan (NEST) – vertical (AC_ROUTE_INFO.ttl)
    - fixm:EfplPoint4D --> fixm:flightLevel
  - Flight plan (NEST) – temporal (AC_ROUTE_INFO.ttl)
    - fixm:EfplPoint4D --> fixm:time

- Output data
  - Planned trajectory passes through restricted airspace
  - Planned trajectory doesn’t pass through restricted airspace

- Flowchart

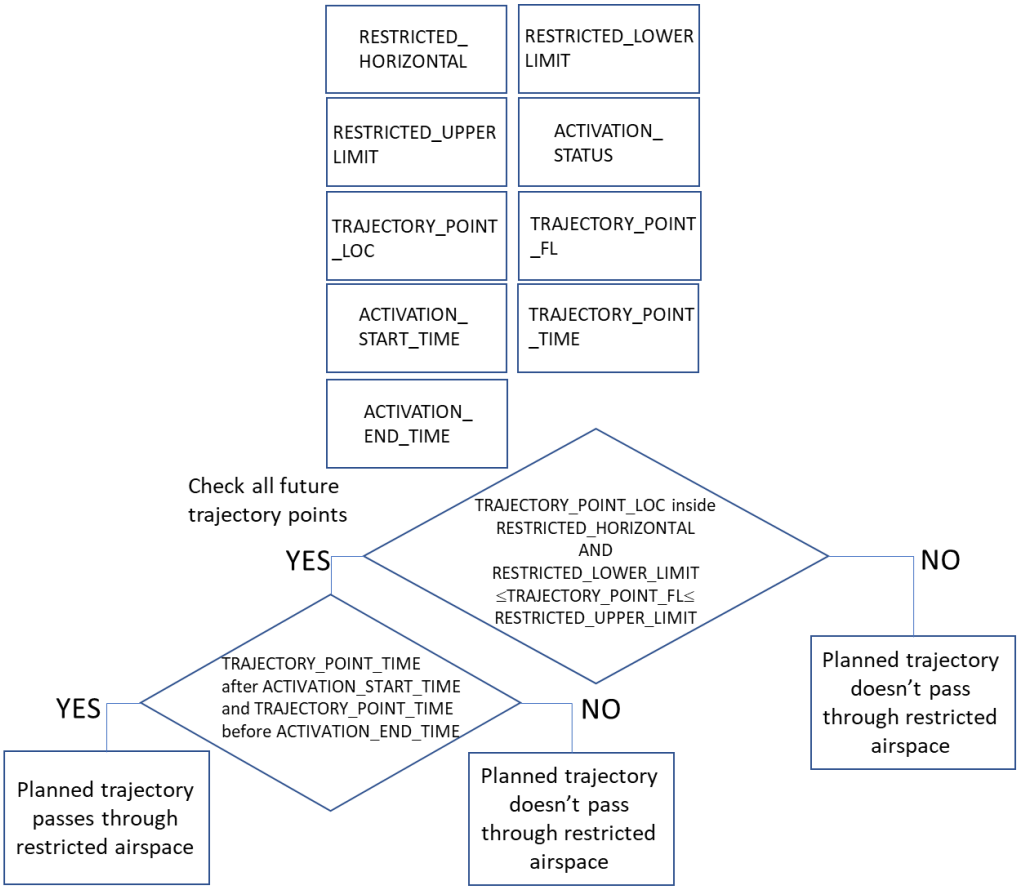



**Figure 40. Task 6.4 flowchart**

- SPARQL queries

```
SELECT ?callsign ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}

SELECT ?name ?upperLimit ?lowerLimit ?status ?startTime ?endTime
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit ;
plain:timeSlice/plain:activation/plain:status/rdf:value ?status ;
plain:timeSlice/plain:activation/plain:timeInterval/plain:startTime/rdf:value ?startTime ;
plain:timeSlice/plain:activation/plain:timeInterval/plain:endTime/rdf:value ?endTime .
FILTER (!REGEX(?name,"LSAZM567")) .}}
```

```
SELECT ?name ?horizontalProjectionEast
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
plain:patch/gml:exterior/plain:segment/gml:segments/gml:posList/rdf:value
?horizontalProjectionEast .
FILTER REGEX(?name,"LS-T/TRA_EAST") }}

SELECT ?name ?horizontalProjectionWest
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
plain:patch/gml:exterior/plain:segment/gml:segments/gml:posList/rdf:value
?horizontalProjectionWest .
FILTER REGEX(?name,"LS-T/TRA_WEST") }}
```

## 5.7 Transfer Aircraft

When exit conditions have been met, aircraft needs to be transferred. They need to be given a change of frequency and their status needs to be set as transferred.

### 5.7.1 Check which aircraft need to be transferred

- Input data
  - Current a/c position, altitude, speed, track and ROC/ROD for time calculations (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
    - fixm:AircraftPosition --> fixm:flightLevel
    - fixm:AircraftPosition --> fixm:track
    - fixm:AircraftPosition --> fixm:actualSpeed and fixm:VerticalRate
  - Sector boundary – horizontal and vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Trajectory points (AC_ROUTE_INFO.ttl)
    - fixm:EfplRoute --> fixm:routeText
  - Task 6.2 Detect aircraft that have to climb/descend to exit FL
  - X - Control variable – time from boundary determined for transfer (2 minutes)
  - Y – Control variable – time from vertical boundary determined for transfer (1 minute)
- Output data
  - Aircraft needs to be transferred
  - Aircraft does not need to be transferred

- Flowchart

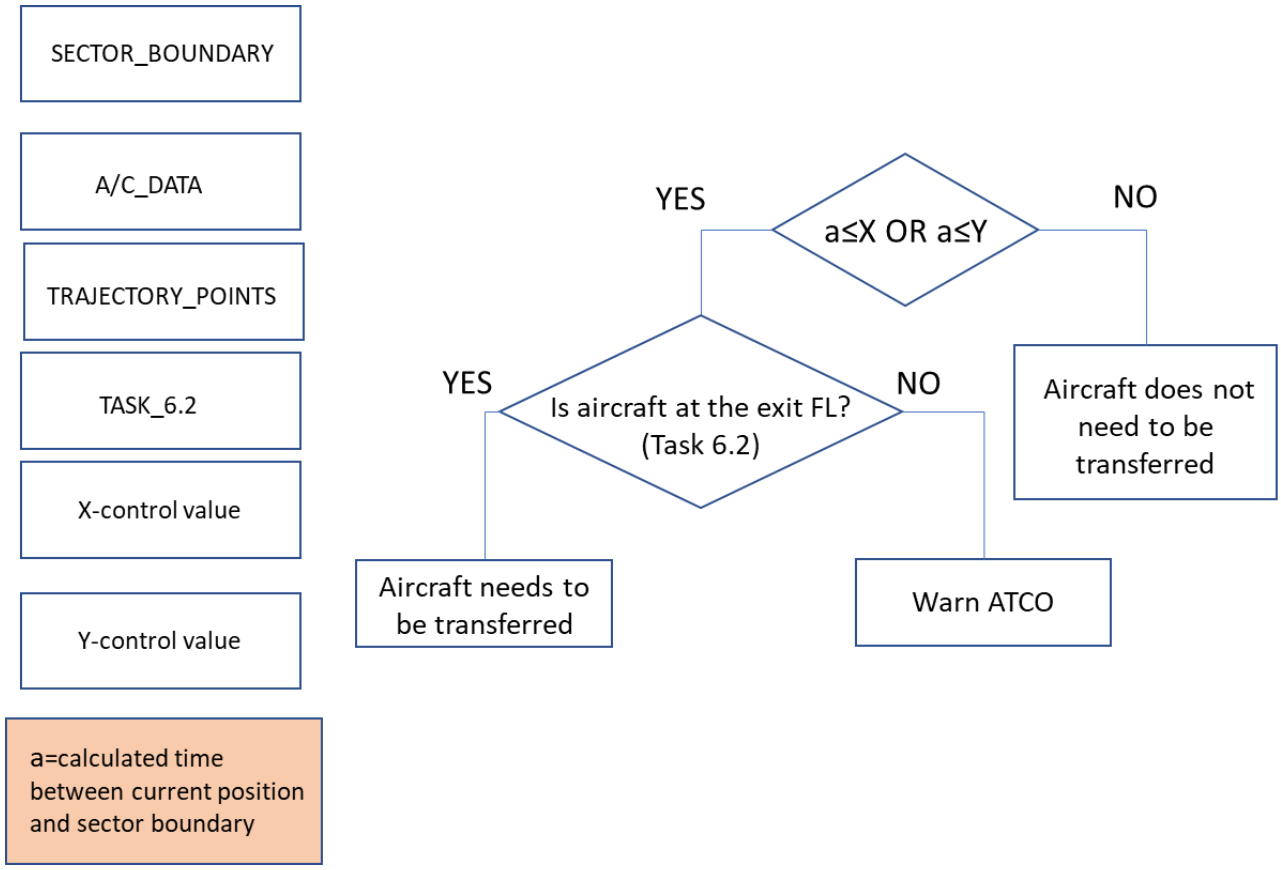



**Figure 41. Task 7.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircraftSpeed ?currentHeading
?currentVerticalRate ?routeText
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:track/rdf:value ?currentHeading ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:efplTrajectory/plain:efplRoute/plain:routeText/rdf:value ?routeText .}}

SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
```

```
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .
FILTER REGEX(?name,"LSAZM567") }}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}
```

### 5.7.2 Check if change of frequency is issued to A/C (via datalink)

- Input data
  - Datalink (change of frequency) (PLAIN_DATA.ttl)
    - plain:Datalink --> plain:frequencyChange
- Output data
  - Change of frequency was issued
  - Change of frequency was not issued
- Flowchart

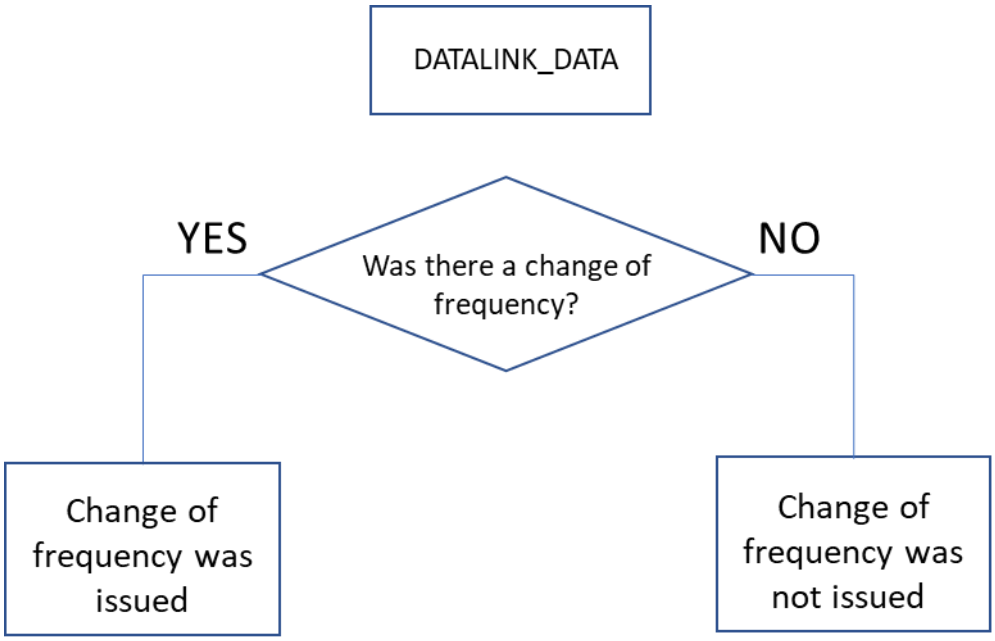



**Figure 42. Task 7.2 flowchart**

- SPARQL query

```
SELECT ?callsign ?frequencyChange
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
```

plain:flightIdentification/plain:datalink/plain:frequencyChange/rdf:value ?frequencyChange .}}

### 5.7.3 Change aircraft status to transferred

- Input data
  - Coordination status (AC_ROUTE_INFO.ttl)
    - fixm:CoordinationStatus --> fixm:coordinationStatus
- Output data
  - Aircraft's status can be changed to transferred
  - Aircraft status can't be changed to transferred/is already changed
- Flowchart

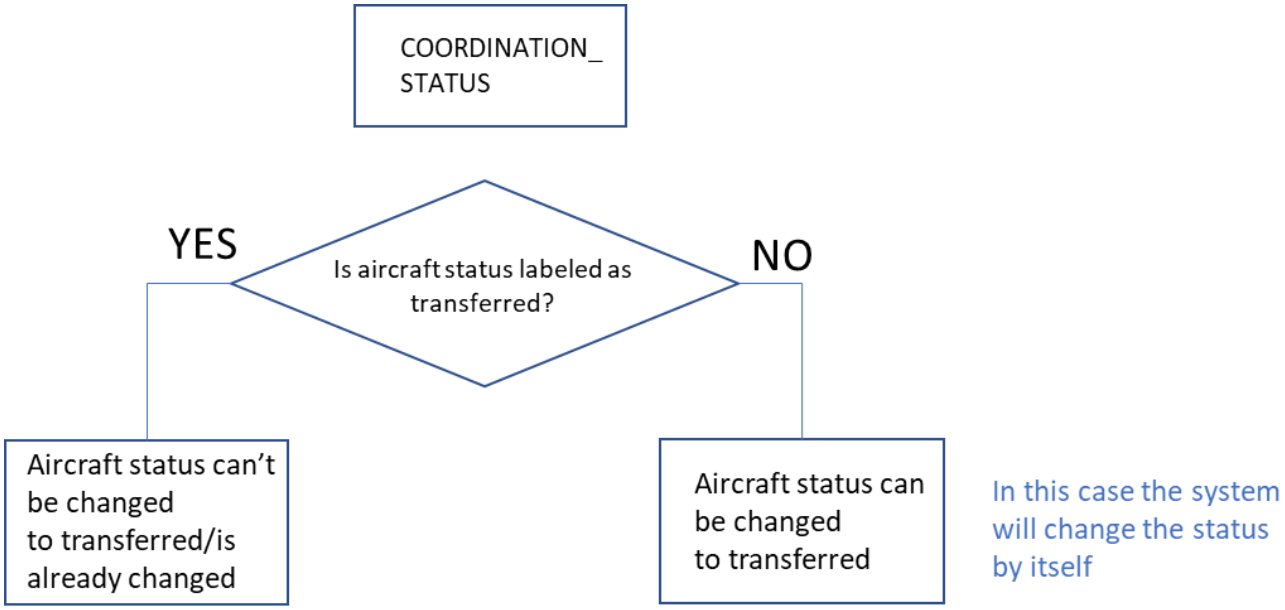



**Figure 43. Task 7.3 flowchart**

- SPARQL query

```
SELECT ?callsign ?coordinationStatus
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:coordination/plain:coordinationStatus/rdf:value
?coordinationStatus .}}
```

## 5.8 Maximise Quality of Service

There are different ways in which ATCOs are able to provide a better quality of service. After all the necessary safety precautions have been taken, it is possible to maximise the quality of service by issuing direct-to orders or simply optimising the aircraft route and approving requested alternatives.

### 5.8.1 Detect direct-to candidates

- Input data
  - Sector boundary – horizontal and vertical (LSAZM567.ttl)
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:lowerLimit/aixm:upperLimit
    - aixm:Airspace --> aixm:AirspaceVolume --> aixm:horizontalProjection
  - Planned trajectory points (“flight plans”) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:flightLevel
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Military area (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceVolume --> aixm:horizontalProjection --> aixm:Surface
    - aixm:AirspaceVolume --> aixm:lowerLimit
    - aixm:AirspaceVolume --> aixm:upperLimit
  - Task 8.2 Determine military airspace availability
- Output data
  - Aircraft is a candidate for direct-to
  - Aircraft is not a candidate for direct-to
- Flowchart

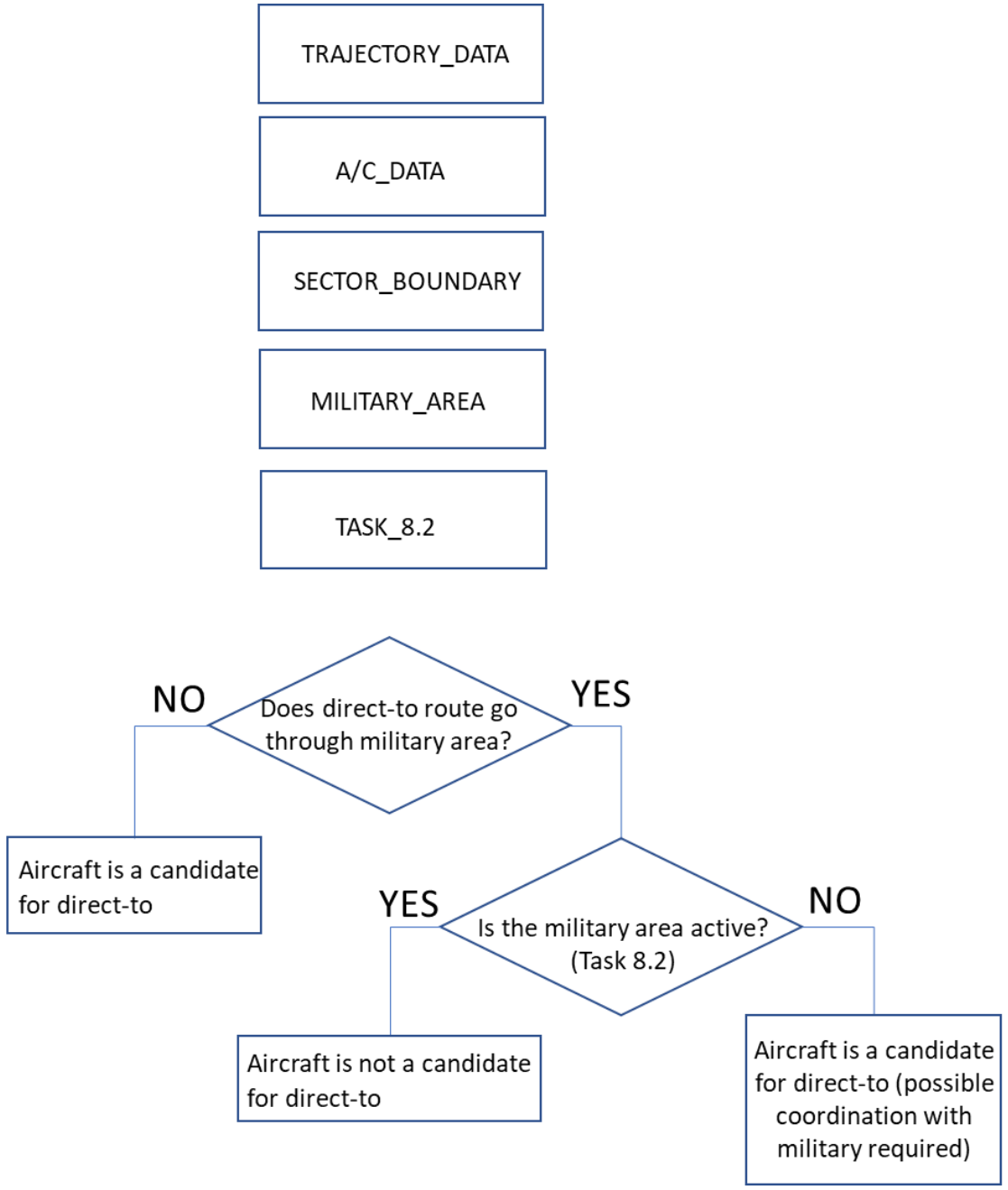



**Figure 44. Task 8.1 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory .}}
```

```
SELECT ?name ?upperLimit ?lowerLimit
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?upperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?lowerLimit .}}

SELECT ?horizontalProjection
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
gml:patches/gml:exterior/plain:segment/gml:segment/gml:posList/rdf:value
?horizontalProjection .}}

SELECT ?name ?horizontalProjectionEast
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
plain:patch/gml:exterior/plain:segment/gml:segments/gml:posList/rdf:value
?horizontalProjectionEast .
FILTER REGEX(?name,"LS-T/TRA_EAST") }}

SELECT ?name ?horizontalProjectionWest
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
plain:patch/gml:exterior/plain:segment/gml:segments/gml:posList/rdf:value
?horizontalProjectionWest .
FILTER REGEX(?name,"LS-T/TRA_WEST") }}
```

### 5.8.2 Determine military airspace availability

- Input data
  - Military area (MILITARY_AIRSPACE.ttl)
    - aixm:Airspace --> aixm:activation --> aixm:AirspaceActivation --> aixm:status
- Output data
  - Aircraft can now use previously reserved military airspace
  - Aircraft can't use previously reserved military airspace
- Flowchart

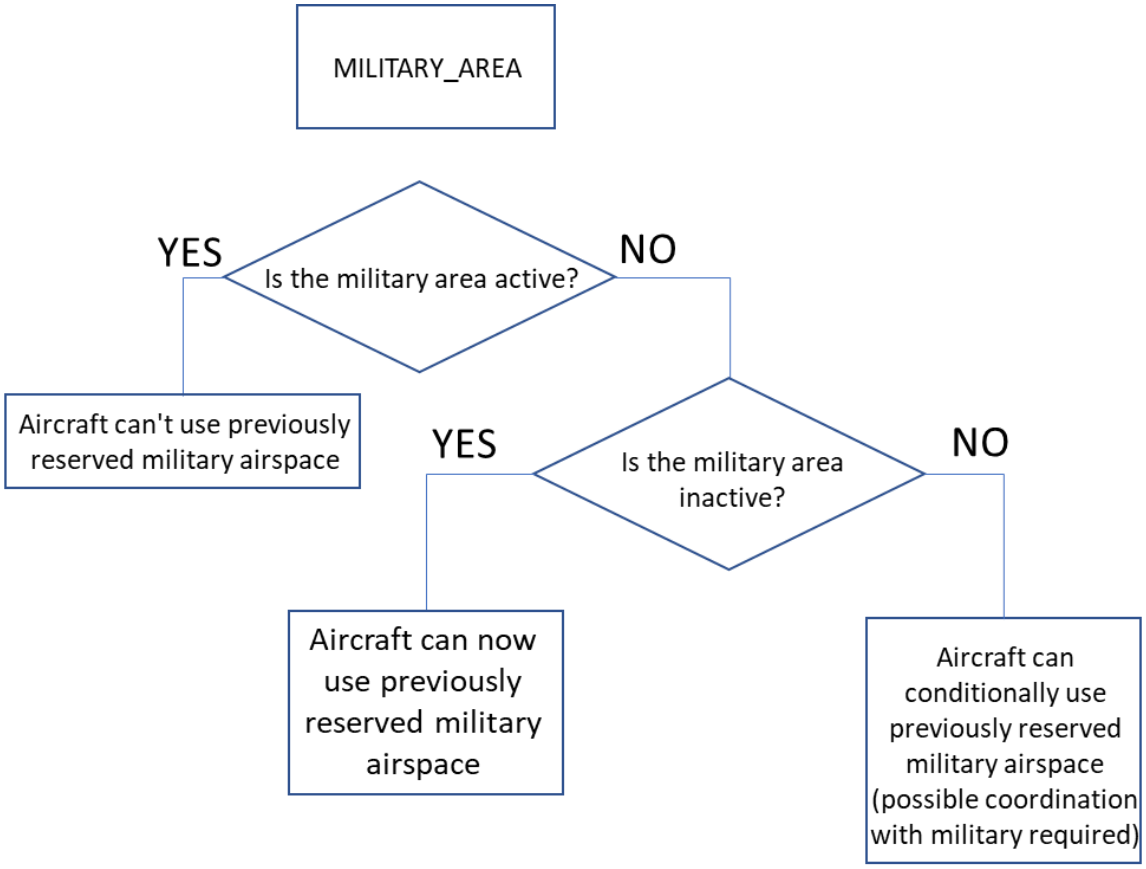



**Figure 45. Task 8.2 flowchart**

- SPARQL query

```
SELECT ?name ?status
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:activation/plain:status/rdf:value ?status ;
FILTER (!REGEX(?name,"LSAZM567")) .}}
```

### 5.8.3 Check suggestion for shortened RBT

- Input data
  - Aircraft route (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:flightLevel
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:time
    - fixm:EfplRoute --> fixm:routeText
  - Military area (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceVolume --> aixm:horizontalProjection --> aixm:Surface
    - aixm:AirspaceVolume --> aixm:lowerLimit
    - aixm:AirspaceVolume --> aixm:upperLimit

- Aircraft position and FL (AC_ROUTE_INFO.ttl)
  - fixm:AircraftPosition --> fixm:position --> fixm:SignificantPoint
  - fixm:AircraftPosition --> fixm:flightLevel
- Task 8.2 Determine military airspace availability
- Conflict detection on the direct route (calculation layer)
- X - Control variable – distance from other aircraft - 10 NM horizontally
- Y - Control variable – distance from other aircraft - 1000 ft vertically

- Output data
  - The suggested shortened RBT (Reference Business Trajectory) causes conflicts
  - The suggested shortened RBT does not cause conflicts

- Flowchart

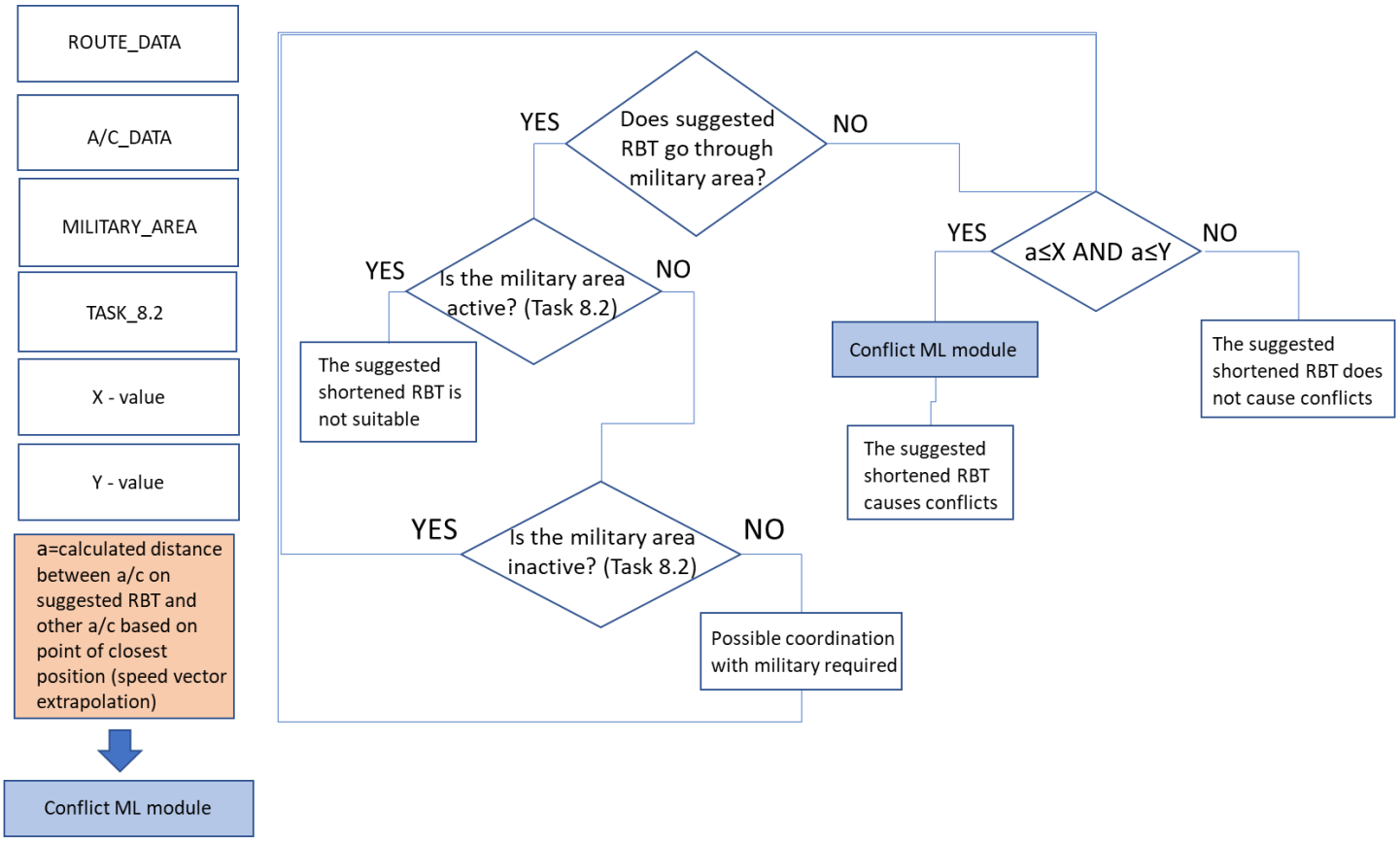



**Figure 46. Task 8.3 flowchart**

- SPARQL queries

```
SELECT ?callsign ?aircraftPosition ?aircraftFL ?routeText ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight a plain:EfplFlight ;
```

```
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:efplTrajectory/plain:efplRoute/plain:routeText/rdf:value ?routeText ;
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory. }}

SELECT ?name ?militaryLowerLimit ?militaryUpperLimit ?horizontalProjection
WHERE { GRAPH ?default
{?airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:lowerLimit/rdf:value
?militaryLowerLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:upperLimit/rdf:value
?militaryUpperLimit ;
plain:timeSlice/plain:geometryComponent/plain:theAirspaceVolume/plain:horizontalProjection/
plain:patch/gml:exterior/plain:segment/gml:segments/gml:posList/rdf:value
?horizontalProjection .}}
```

## 5.9 Workload Monitoring

In order to work efficiently, it is important to keep track of the workload and air traffic complexity so that certain actions can be taken if the workload gets too high.

### 5.9.1 Track current number of assumed aircraft

- Input data
  - Coordination status (AC_ROUTE_INFO.ttl)
    - fixm:CoordinationStatus --> fixm:coordinationStatus
  - List of assumed a/c (PLAIN_DATA.ttl)
    - plain:List --> plain:assumed
- Output data
  - Number of assumed aircraft

- Flowchart

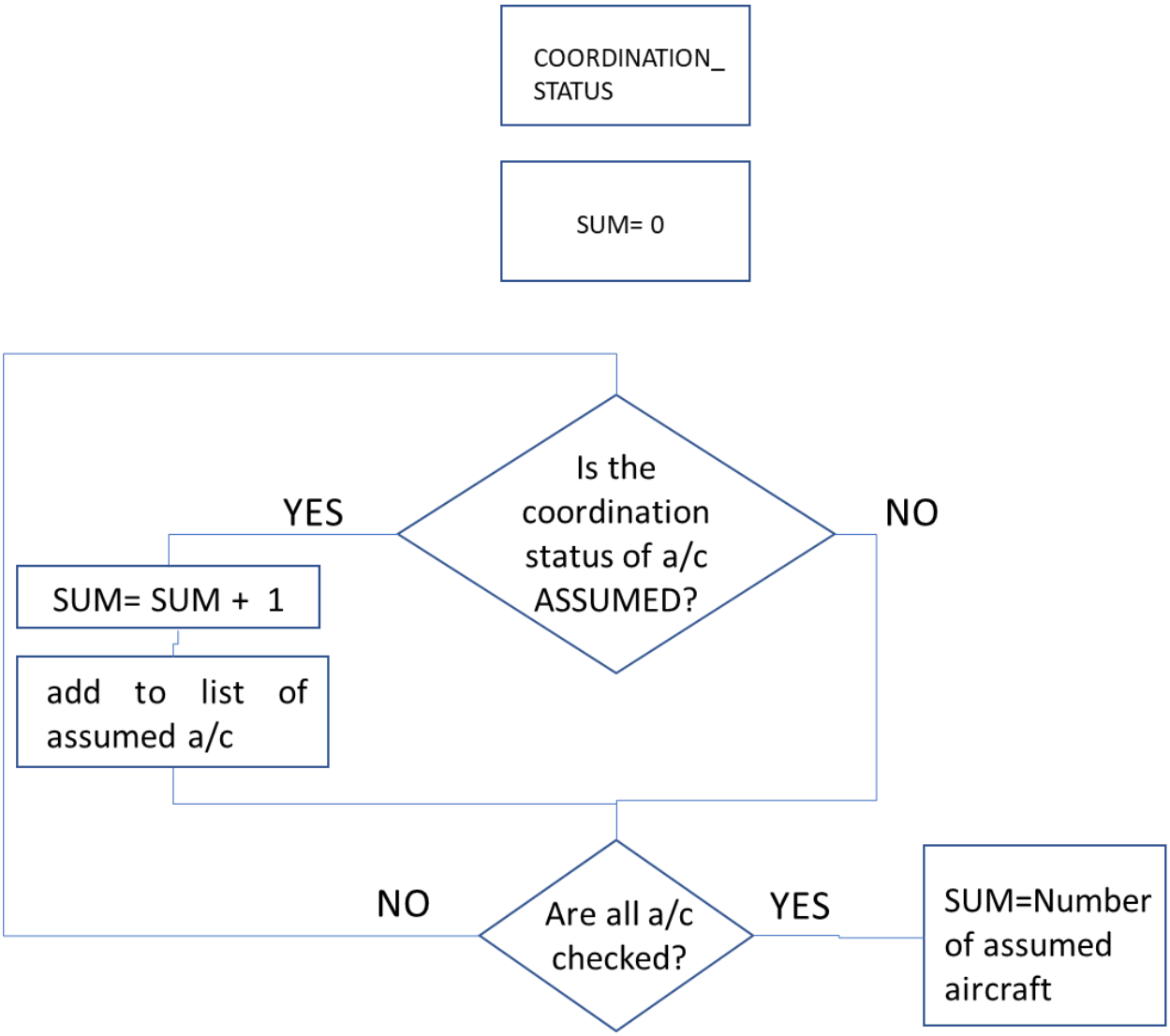



**Figure 47. Task 9.1 flowchart**

- SPARQL query

```
SELECT ?callsign ?coordinationStatus ?list
WHERE { GRAPH ?default
{?efplFlight a plain:EfplFlight ;
plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:coordination/plain:coordinationStatus/rdf:value
?coordinationStatus ;
plain:flightIdentification/plain:list/plain:assumed/rdf:value ?list .}}
```

### 5.9.2 Track number of conflicts and potential conflicts

- Input data
  - Planned trajectory points (“flight plans”) (AC_ROUTE_INFO.ttl)

- fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
- fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:flightLevel
- fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time

- Aircraft data - position, speed, altitude, ROC/ROD (AC_ROUTE_INFO.ttl)
  - fixm:AircraftPosition --> fixm:position --> fixm:SsignificantPoint
  - fixm:AircraftPosition --> fixm:actualSpeed
  - fixm:AircraftPosition --> fixm:flightLevel
  - fixm:VerticalRate and fixm:Route --> fixm:requestedAltitude

- FLAS (PLAIN_DATA.ttl)
  - plain:Altitude --> plain:flightLevel

- X - Control variable – distance from other aircraft - 10 NM horizontally

- Y - Control variable – distance from other aircraft - 1000 ft vertically

- Conflict detection on routes (calculation layer)

- Output data
  - Number of conflicts and potential conflicts

- Flowchart

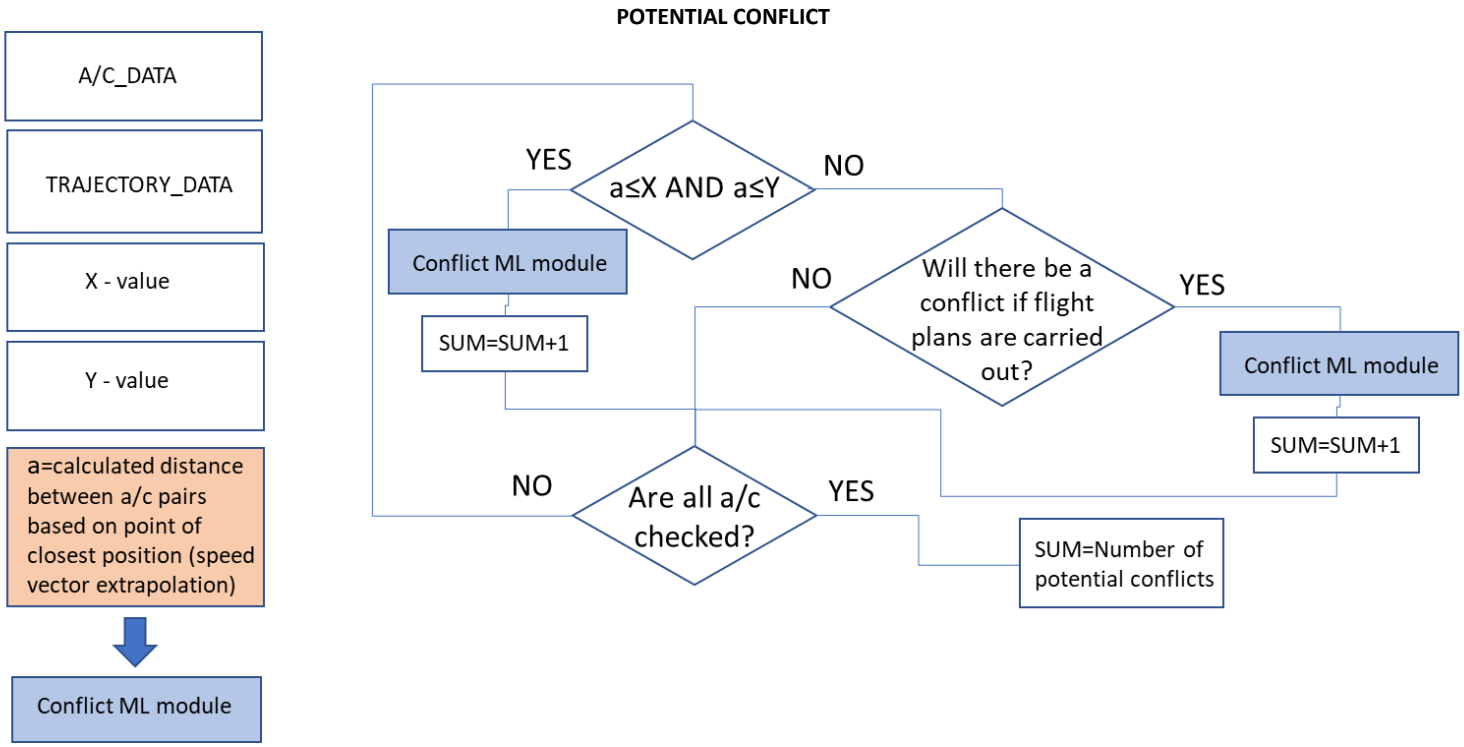



**Figure 48. Task 9.2 potential conflict flowchart**

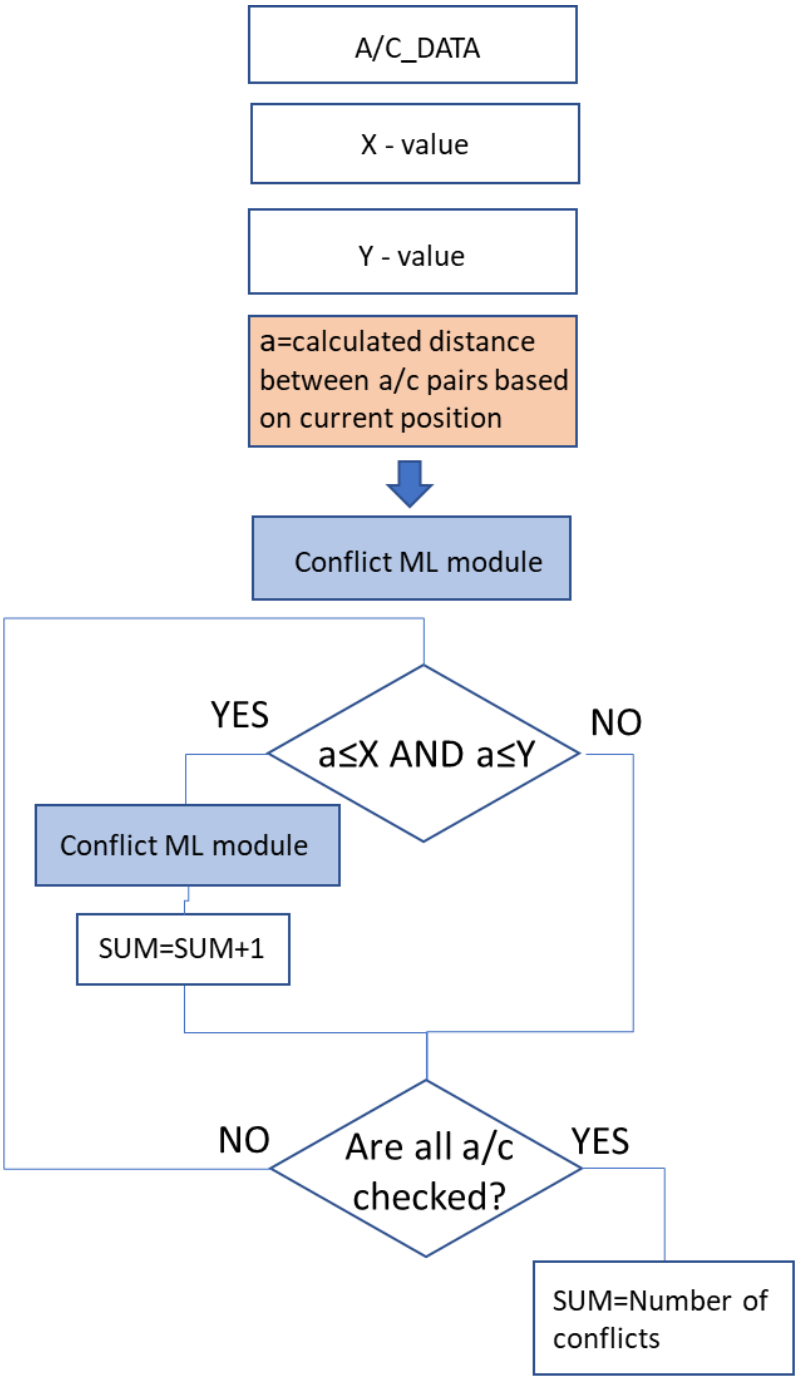



**Figure 49. Task 9.2 conflict flowchart**

- SPARQL query

  SELECT ?callsign ?aircraftPosition ?aircraftFL ?aircraftSpeed ?currentVerticalRate ?requestedFL ?plannedTrajectory ?exitFL
  WHERE { GRAPH ?g1



```
{?efplFlight plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/ plain:positionPoint/rdf:value ?aircraftPosition ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:actualSpeed/rdf:value ?aircraftSpeed ;
plain:flightIdentification/plain:position/plain:rate/rdf:value ?currentVerticalRate ;
plain:flightIdentification/plain:route/plain:requestedAltitude/rdf:value ?requestedFL .
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory ;
file:alt370 plain:flightLevel/rdf:value ?exitFL .}}
```

### 5.9.3 Determine future number of sector entries

- Input data
  - Task 3.1. List of incoming aircraft
  - Task 9.1. List of assumed aircraft
  - Planned entries (point position and time) (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time
  - Planned exits (AC_ROUTE_INFO.ttl)
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:pos
    - fixm:Trajectory --> fixm:trajectoryPoint --> fixm:EfplPoint4D --> fixm:time
  - Observing period (every five minutes for the next hour)
- Output data
  - Number of future sector entries within the hour, divided into 5-minute periods
- Flowchart

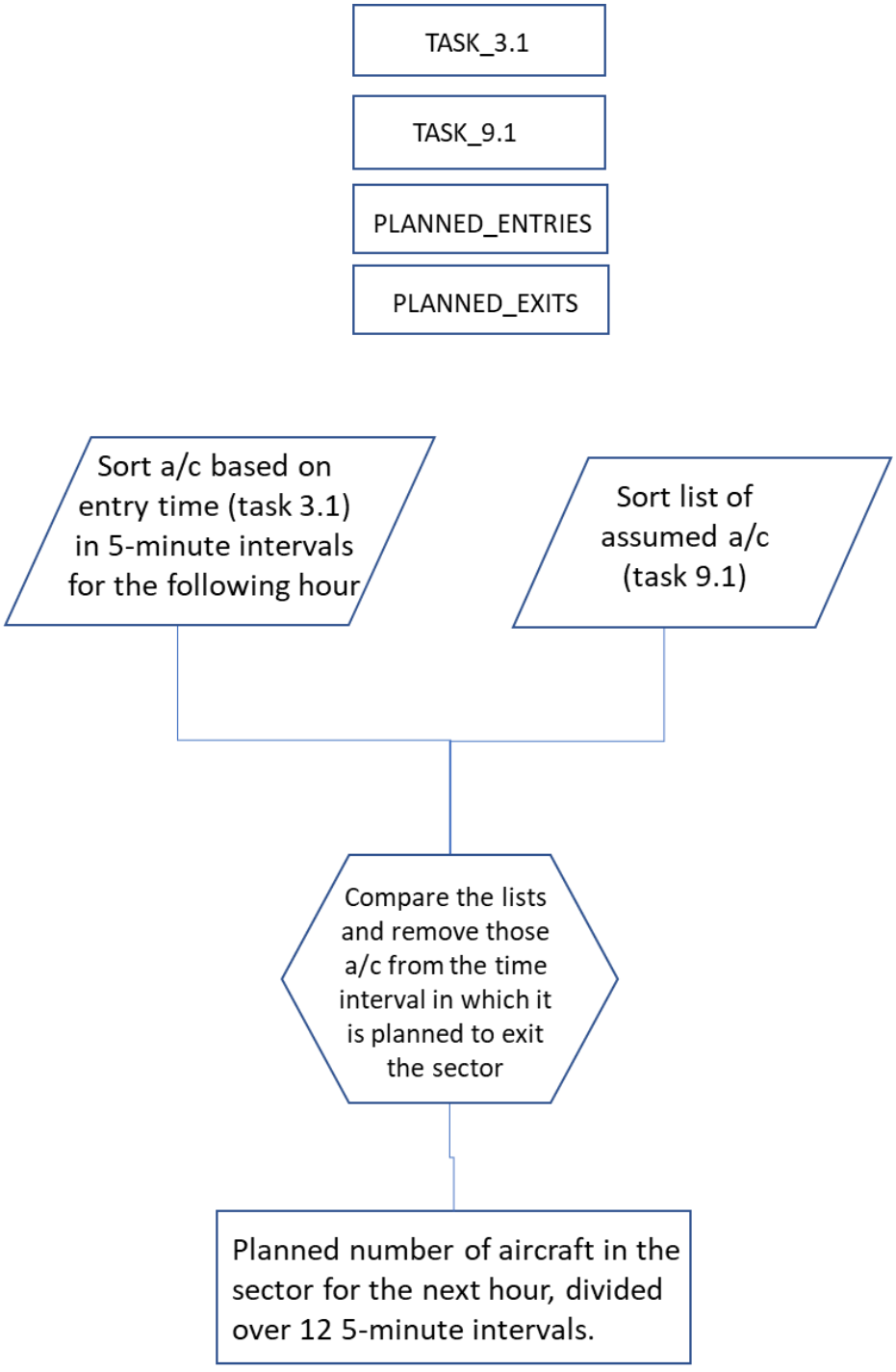



**Figure 50. Task 9.3 flowchart**

- SPARQL query

```
SELECT ?callsign ?plannedTrajectory
WHERE { GRAPH ?g1
{?efplFlight plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign .
plain:efplTrajectory/plain:trajectory/plain:trajectoryList/rdf:value ?plannedTrajectory.}}
```

### 5.9.4 Determine sector air traffic complexity (ML module)

- Input data
  - Airspace complexity ML module
- Output data
  - Sector air traffic complexity
- Flowchart

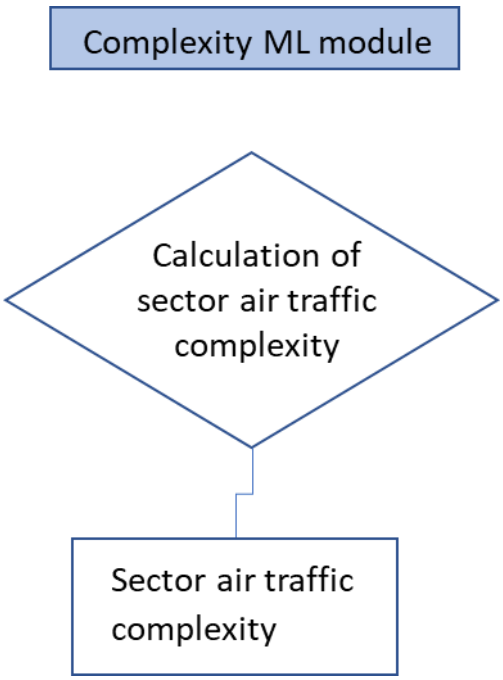



**Figure 51. Task 9.4 flowchart**

- SPARQL query
  - a query is not necessary as all the data comes from the ML module

### 5.9.5 Determine plausibility of traffic complexity assessment

- Input data
  - Task 9.1 Track current number of assumed aircraft
  - Conflict detection ML module (number of conflicts and previous time-step number of conflicts)
  - Airspace complexity ML module (complexity and previous time-step complexity) (C)
  - Complexity tolerance (±0.5)
  - Calculation layer
    - N = number of assumed aircraft + number of conflicts

- $^{+}$ = current C/current N > previous C/previous N
- $^{-}$ = current C/current N < previous C/previous N
- $^{=}$ = current C/current N = previous C/previous N

- Output data
  - Plausibility of traffic complexity assessment
- Flowchart

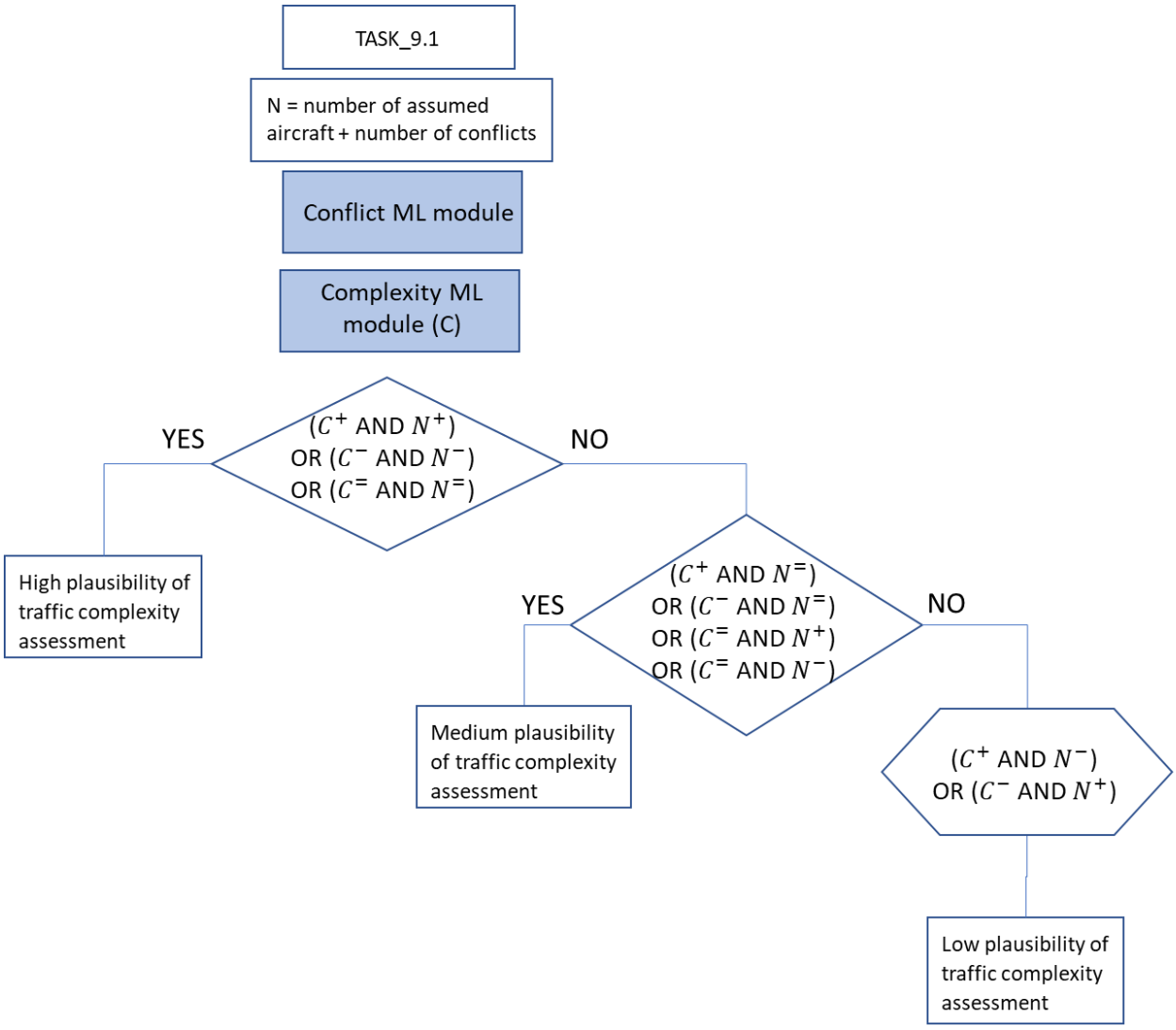



**Figure 52. Task 9.5 flowchart**

- SPARQL query
  - a query is not necessary as all the data comes from ML modules and previous task

## 5.10 Identify Missing Information

These tasks are important as they review existing data and based on it (or lack thereof) report back to the ATCO in order to either warn them or notify them about certain changes in data or missing data.

### 5.10.1 Identify aircraft with possible equipment degradation

- Input data
  - Datalink (PLAIN_DATA.ttl)
    - plain:Datalink --> plain:equipmentDegradation
  - Beacon code (ENROUTE_CLEARED.ttl)
    - fixm:EnRoute --> fixm:beaconCodeAssignment --> fixm:currentBeaconCode
  - Aircraft characteristic (entire AIRCRAFT_CHARACTERISTIC.ttl code)
  - Aircraft performance (PLAIN_DATA.ttl)
    - plain:Performance --> plain:performanceAltitude
    - plain:trueAirSpeed
    - plain:machNumber
    - plain:rateOfClimb
    - plain:rateOfDescent
- Output data
  - Aircraft has possible equipment degradation
  - Aircraft doesn't have equipment degradation
- Flowchart

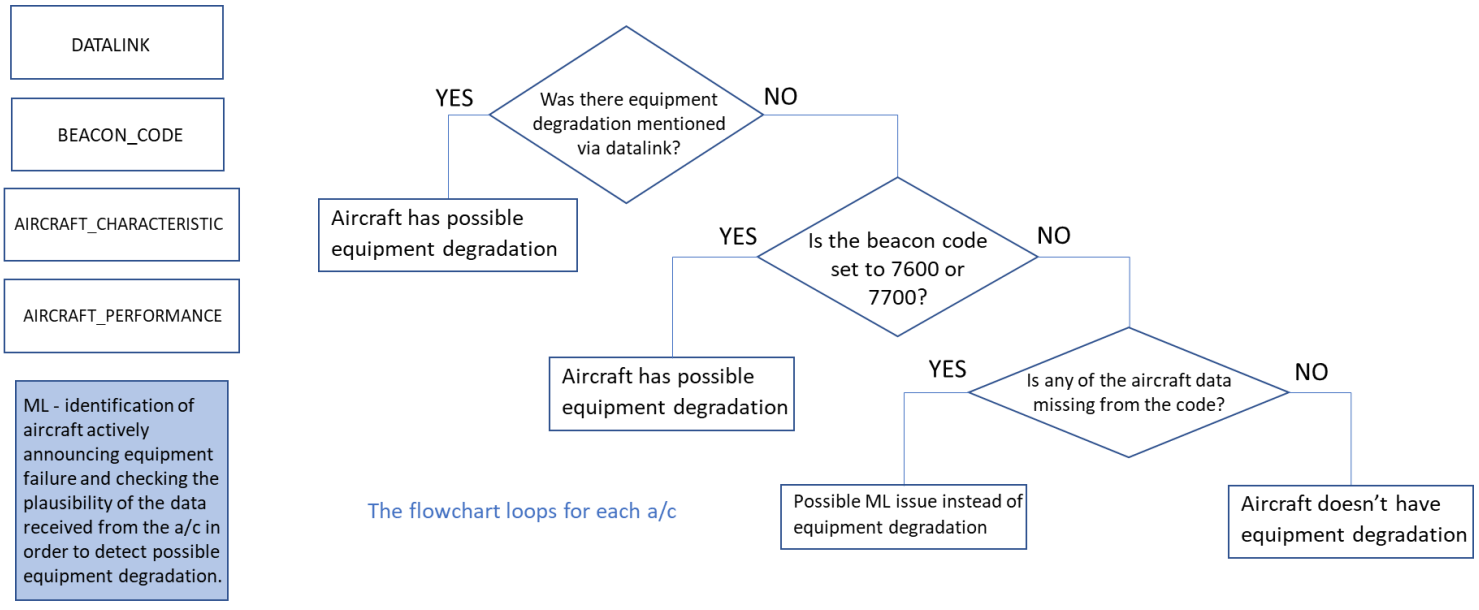



**Figure 53. Task 10.1 flowchart**

- SPARQL query

```
SELECT ?beaconCode ?equipmentDegradation ?performanceAltitude ?trueAirSpeed
?machNumber ?rateOfClimb ?rateOfDescent ?aircraftLandingCategory ?speed ?typeAircraftICAO
?verticalSeparationCapability ?wakeTurbulence ?weight
WHERE { GRAPH ?g1
{?beaconCodeAssignment plain:currentBeaconCode/rdf:value ?beaconCode .
?datalink plain:equipmentDegradation/rdf:value ?equipmentDegradation .
?performance a plain:Performance ;
plain:performanceAltitude/rdf:value ?performanceAltitude ;
plain:trueAirSpeed/rdf:value ?trueAirSpeed ;
plain:machNumber/rdf:value ?machNumber ;
plain:rateOfClimb/rdf:value ?rateOfClimb ;
plain:rateOfDescent/rdf:value ?rateOfDescent .
?ACCharacteristic a plain:AircraftCharacteristic ;
plain:aircraftLandingCategory/rdf:value ?aircraftLandingCategory ;
plain:speed/rdf:value ?speed ;
plain:typeAircraftICAO/rdf:value ?typeAircraftICAO ;
plain:verticalSeparationCapability/rdf:value ?verticalSeparationCapability ;
plain:wakeTurbulence/rdf:value ?wakeTurbulence ;
plain:weight/rdf:value ?weight .}}
```

### 5.10.2 Check situation at destination airport

- Input data
  - Airport data - operational status and warnings (AIRPORT.ttl)
    - aixm:AirportHeliportAvailability --> aixm:operationalStatus
    - aixm:AirportHeliportAvailability --> aixm:warning
- Output data
  - Situation at destination airport checked
- Flowchart

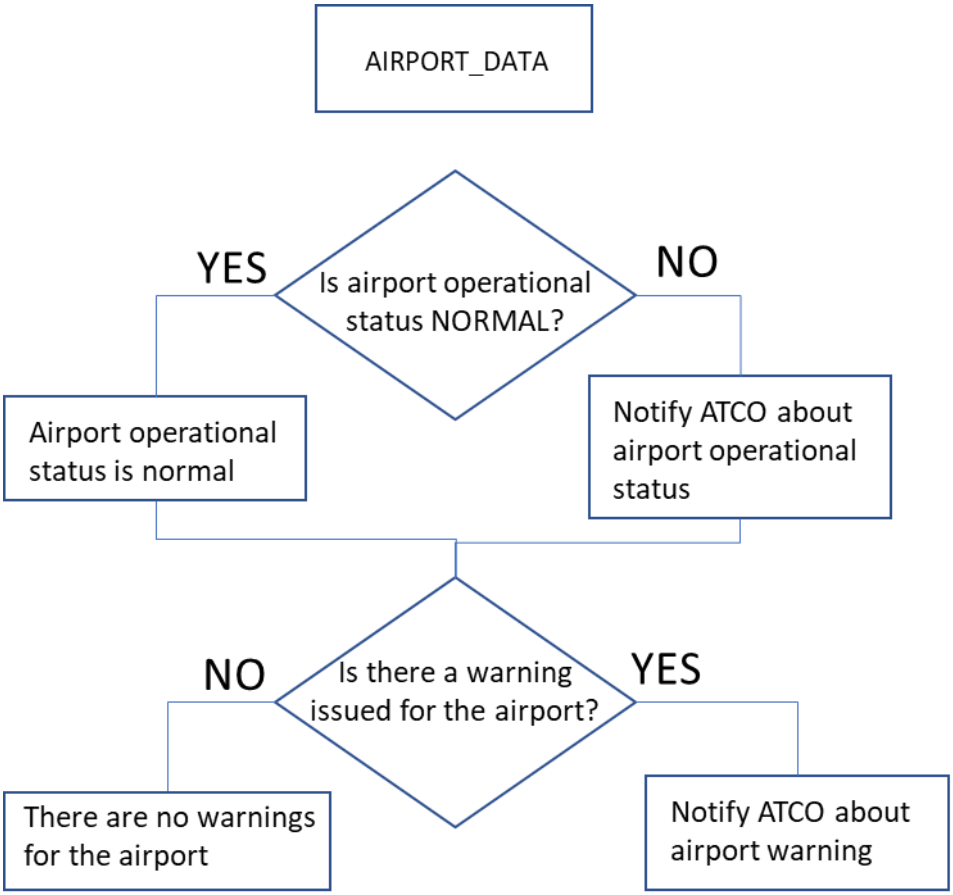



**Figure 54. Task 10.2 flowchart**

- SPARQL query

```
SELECT ?airportName ?operationalStatus ?warning
WHERE { GRAPH ?default
{?airportHeliport a plain:AirportHeliportTimeSlice ;
plain:name/rdf:value ?airportName .
plain:availability/plain:operationalStatus/rdf:value ?operationalStatus .
plain:availability/plain:warning/rdf:value ?warning .}}
```

### 5.10.3 Check situation at alternate airports

- Input data
  - Airport data - operational status and warnings (AIRPORT.ttl)
    - aixm:AirportHeliportAvailability --> aixm:operationalStatus
    - aixm:AirportHeliportAvailability --> aixm:warning
- Output data
  - Situation at alternate airport checked
- Flowchart

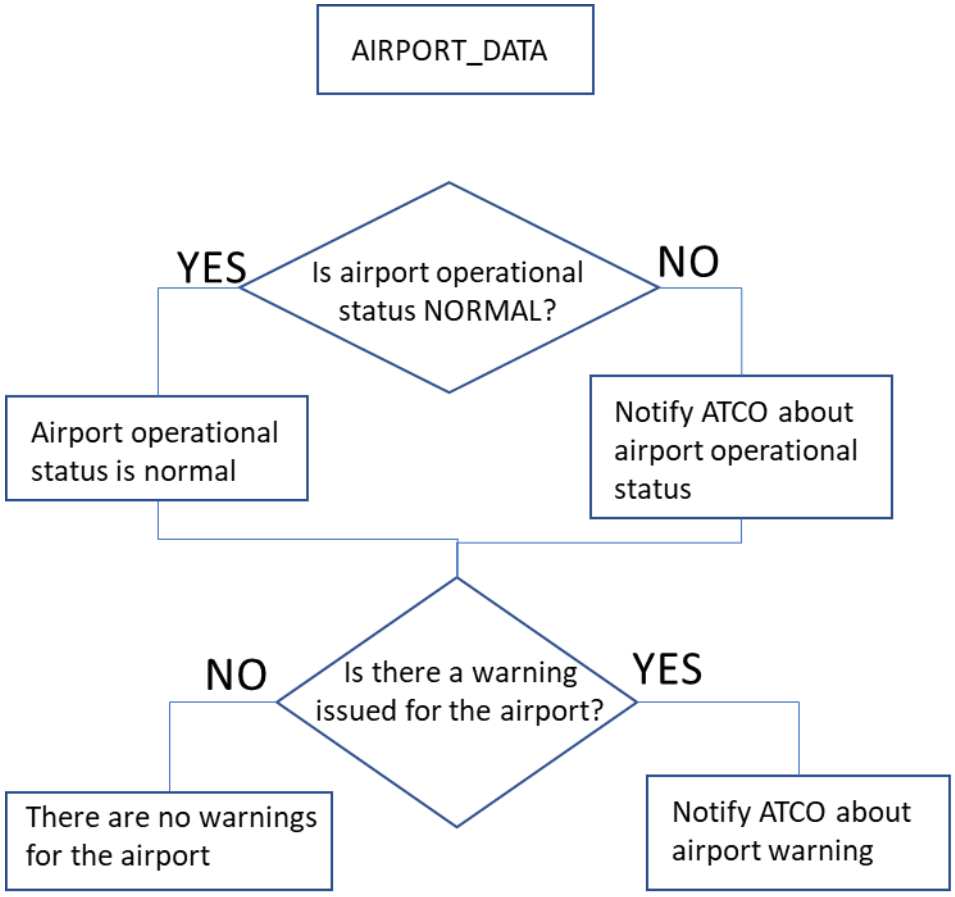



**Figure 55. Task 10.3 flowchart**

- SPARQL query

```
SELECT ?airportName ?operationalStatus ?warning
WHERE { GRAPH ?default
{?airportHeliport a plain:AirportHeliportTimeSlice ;
plain:name/rdf:value ?airportName ;
plain:availability/plain:operationalStatus/rdf:value ?operationalStatus ;
plain:availability/plain:warning/rdf:value ?warning .}}
```

### 5.10.4 Monitor adverse weather areas

- Input data
  - Weather data (PLAIN_DATA.ttl)
    - plain:Weather --> plain:temperature
    - plain:windSpeed
    - plain:thunder
    - plain:icing
    - plain:precipitation
    - plain:cloudCoverage
  - Previous time-steps weather data (PLAIN_DATA.ttl)

- plain:Weather --> plain:temperature
- plain:windSpeed
- plain:thunder
- plain:icing
- plain:precipitation
- plain:cloudCoverage

- Output data
  - No changes in the observed weather zone
  - There is a change in weather conditions that could affect decision-making

- Flowchart

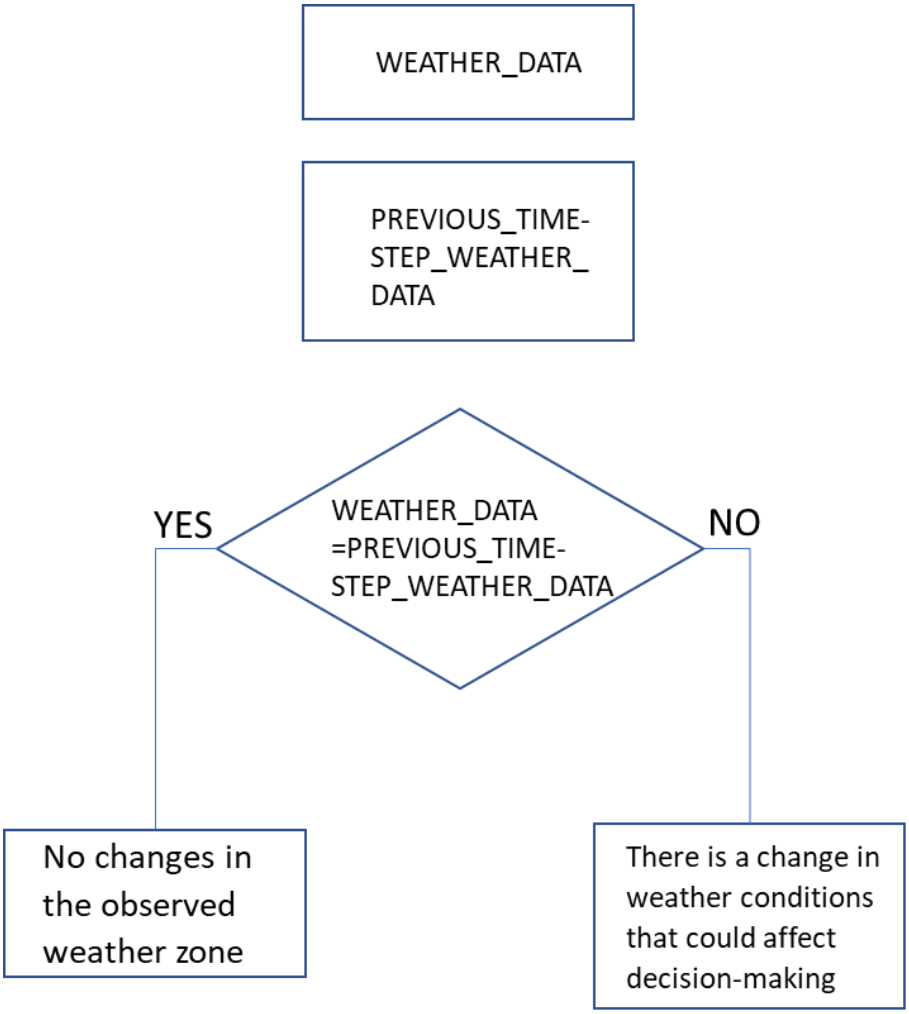



**Figure 56. Task 10.4 flowchart**

- SPARQL query

```
SELECT ?temperature ?windSpeed ?windDirection ?thunder ?icing ?precipitation
?cloudCoverage
WHERE { GRAPH ?g1
{?weather a plain:Weather;
```

```
plain:temperature/rdf:value ?temperature .
plain:windSpeed/rdf:value ?windSpeed .
plain:windDirection/rdf:value ?windDirection .
plain:thunder/rdf:value ?thunder .
plain:icing/rdf:value ?icing .
plain:precipitation/rdf:value ?precipitation .
plain:cloudCoverage/rdf:value ?cloudCoverage .}}

SELECT ?previousTemperature ?previousWindSpeed ?previousWindDirection
?previousThunder ?previousIcing ?previousPrecipitation ?previousCloudCoverage
WHERE { GRAPH ?g0
{?weather a plain:Weather;
plain:temperature/rdf:value ?temperature .
plain:windSpeed/rdf:value ?windSpeed .
plain:windDirection/rdf:value ?windDirection .
plain:thunder/rdf:value ?thunder .
plain:icing/rdf:value ?icing .
plain:precipitation/rdf:value ?precipitation .
plain:cloudCoverage/rdf:value ?cloudCoverage .}}
```

### 5.10.5 Monitor restricted airspace

- Input data
  - Restricted airspace status (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceActivation --> aixm:status
  - Previous time-step restricted airspace status (MILITARY_AIRSPACE.ttl)
    - aixm:AirspaceActivation --> aixm:status
- Output data
  - No new actions can be taken
  - New actions could be taken
- Flowchart

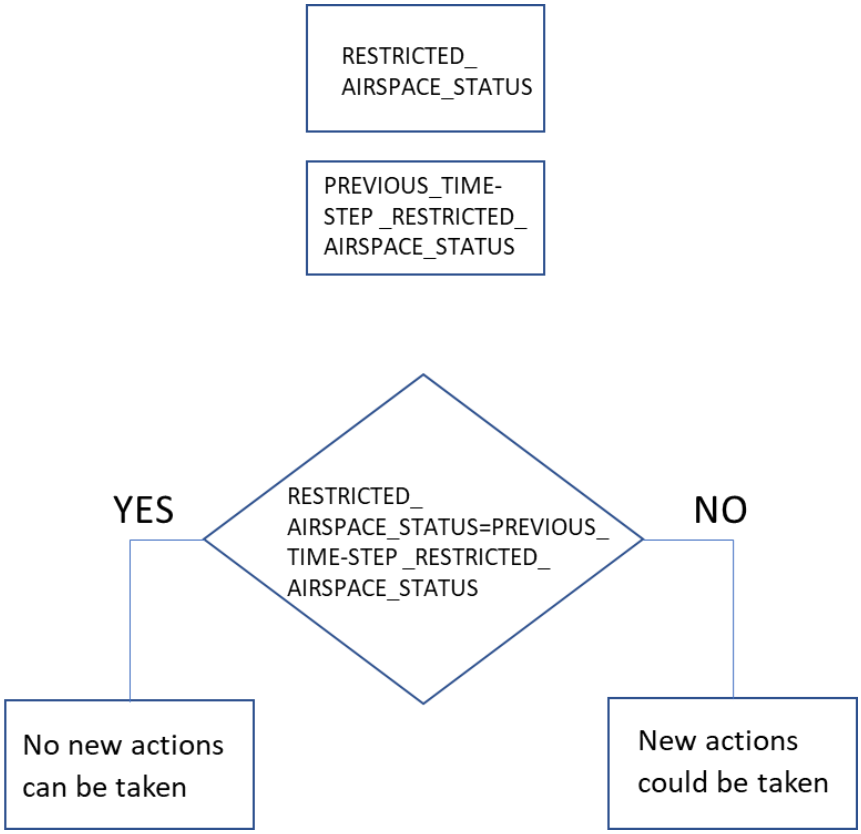



**Figure 57. Task 10.5 flowchart**

- SPARQL query

```
SELECT ?name ?status
WHERE { GRAPH ?default
{?Airspace a plain:Airspace ;
plain:timeSlice/plain:name/rdf:value ?name ;
plain:timeSlice/plain:activation/plain:status/rdf:value ?status ;
FILTER (!REGEX(?name,"LSAZM567")) .}}
```

## 5.10.6 Infer missing information

- Input data
  - All AIXM/FIXM/PLAIN data
- Output data
  - There is missing information
  - There is no missing information
  - Information is outside standard constraints
  - Information is within standard constraints

- Flowchart

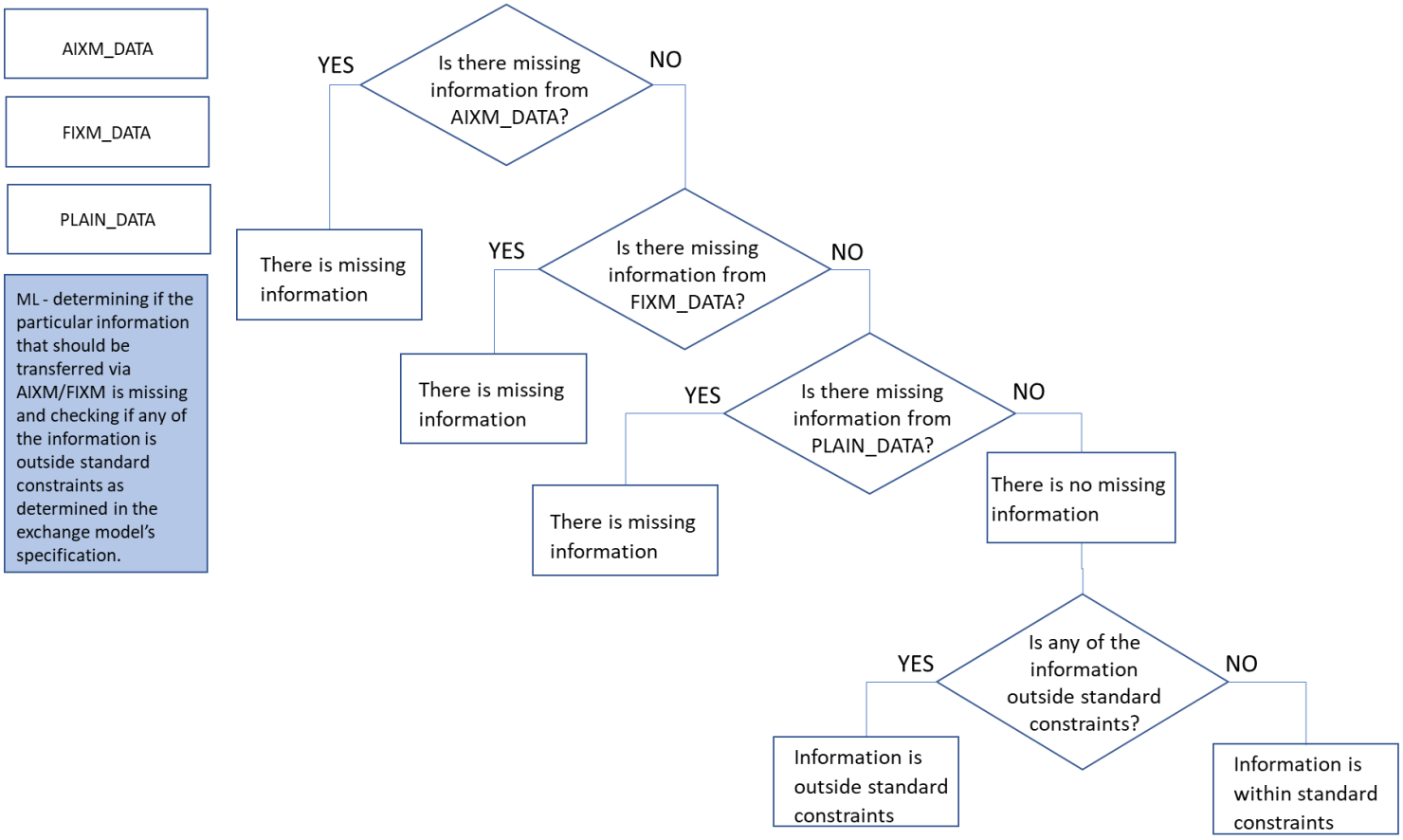



**Figure 58. Task 10.6 flowchart**

- SPARQL query
  - the query will encompass all relevant data

## 5.11 Monitor Status of ATC Sub-systems

As these modules have been recently developed, constant monitoring is necessary to detect possible discrepancies.

### 5.11.1 Monitor performance of ATC conflict detection module

- Input data
  - Cleared flight information and previous time-step cleared flight information (CLEARED_FLIGHT_INFORMATION.ttl)
    - fixm:ClearedFlightInformation --> fixm:clearedFlightLevel
    - plain:clearedSpeed
    - plain:heading
    - plain:offtrackClearance
    - plain:rateOfClimbDescent
    - plain:directRouting

- Conflict detection ML module (prediction and actual time of predicted conflict)

- Output data
  - The module's performance is satisfactory
  - The module's performance is not satisfactory

- Flowchart

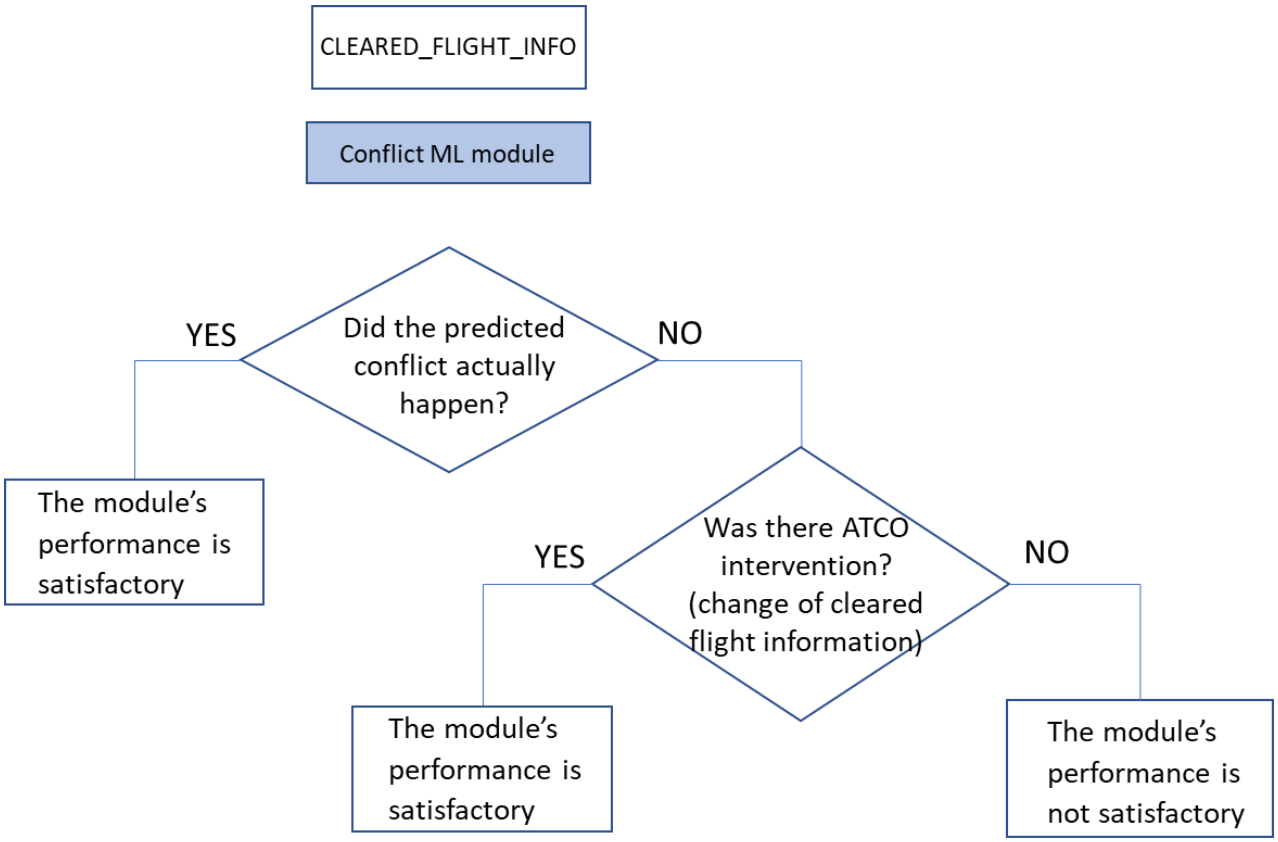



**Figure 59. Task 11.1 flowchart**

- SPARQL query

```
SELECT ?clearedFL ?clearedSpeed ?clearedHeading ?offtrackClearance ?verticalRate
?directFrom ?directTo
WHERE { GRAPH ?g1
{?clearedFlightInformation plain:clearedFlightLevel/rdf:value ?clearedFL ;
plain:clearedSpeed/rdf:value ?clearedSpeed ;
plain:heading/rdf:value ?clearedHeading ;
plain:offtrackClearance/rdf:value ?offtrackClearance ;
plain:rateOfClimbDescent/rdf:value ?verticalRate ;
plain:directRouting/plain:from/rdf:value ?directFrom ;
plain:directRouting/plain:to/rdf:value ?directTo . }}
```

## 5.11.2 Monitor performance of complexity assessment module

- Input data
  - Task 9.5 Determine plausibility of traffic complexity assessment
- Output data
  - The module's performance is satisfactory
  - The module's performance is not satisfactory
- Flowchart

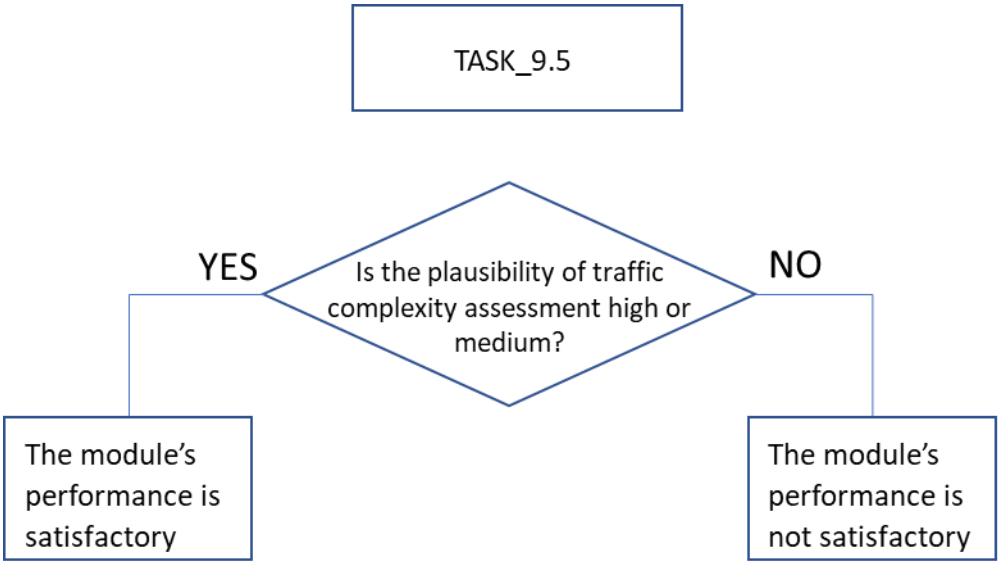



**Figure 60. Task 11.2 flowchart**

- SPARQL query
  - a query is not necessary as all the data comes from one of the previous tasks

## 5.11.3 Monitor performance of trajectory prediction module

- Input data
  - Aircraft position (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:position
  - Aircraft flight level (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:flightLevel
  - Aircraft position time (AC_ROUTE_INFO.ttl)
    - fixm:AircraftPosition --> fixm:positionTime
  - Position tolerance (±2.5NM)
  - Time tolerance (±30sec)
  - FL tolerance (±100ft)
  - 4D trajectory prediction ML module

- Output data
  - The module's performance is satisfactory
  - The module's performance is not satisfactory

- Flowchart

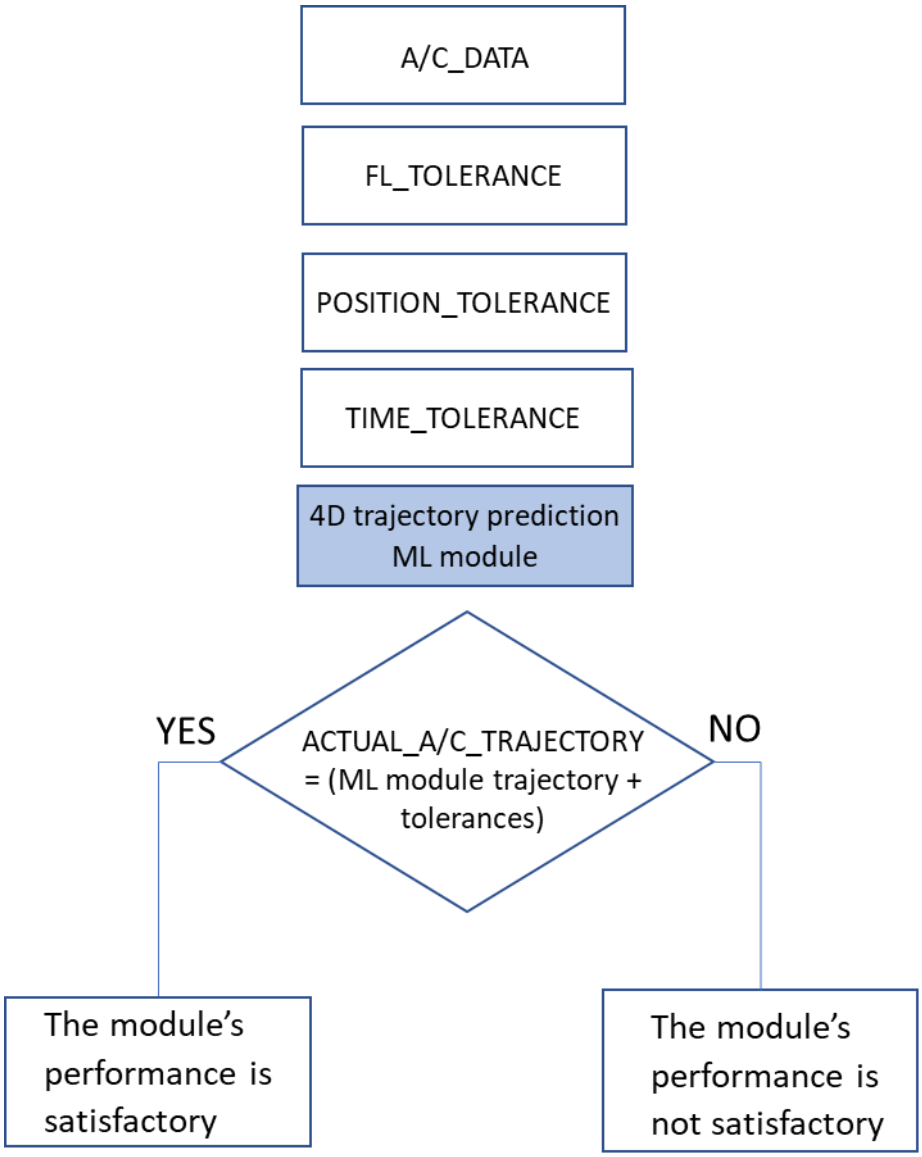



**Figure 61. Task 11.3 flowchart**

- SPARQL query

```
SELECT ?callsign ?aircraftFL ?positionTime ?aircraftPosition
WHERE { GRAPH ?g1
{?efplFlight plain:flightIdentification/plain:aircraftIdentification/rdf:value ?callsign ;
plain:flightIdentification/plain:position/plain:flightLevel/rdf:value ?aircraftFL ;
plain:flightIdentification/plain:position/plain:positionTime/rdf:value ?positionTime .
plain:flightIdentification/plain:position/plain:positionPoint/rdf:value ?aircraftPosition .}}
```

# 6 References


[1] “Shapes Constraint Language (SHACL).” Accessed: Oct. 19, 2021. [Online]. Available: https://www.w3.org/TR/shacl/

[2] Bernd Neumayr and Marlene Hartmann, “KG-Prolog mapper.” Sep. 2021.

[3] SESARJU, “AISA Grant Agreement No 892618.” 2020.

[4] William F. Clocksin and Christopher S. Mellish, *Programming in Prolog*. 2003.

[5] “SPARQL Query Language for RDF.” Accessed: Oct. 19, 2021. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query

[6] “AIXM.” https://www.aixm.aero/ (accessed Aug. 17, 2021).

[7] Ivan Bratko, *Prolog programming for artificial intelligence*. 2012. Accessed: Sep. 22, 2021. [Online]. Available: https://silp.iiita.ac.in/wp-content/uploads/PROLOG.pdf

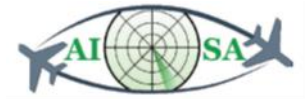
[8] J. E. L. Gayo, E. Prud’hommeaux, I. Boneva, and D. Kontokostas, *Validating RDF Data*, vol. 7. 2017. Accessed: Sep. 22, 2021. [Online]. Available: http://www.morganclaypool.com/doi/10.2200/S00786ED1V01Y201707WBE016

[9] “What is SPARQL?,” *Ontotext*. https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/ (accessed Sep. 23, 2021).

[10] A. Kocsis *et al.*, “Requirements for automation of monitoring tasks via AI SA.” Mar. 2021.

# Appendix A Glossary

| Abbreviation | Term |
|---|---|
| 4DT | 4-Dimensional Trajectory |
| AI | Artificial Intelligence |
| AIXM | Aeronautical Information Exchange Model |
| ANSP | Air Navigation Service Provider |
| ATC | Air Traffic Control |
| ATCO | Air Traffic Control Officer |
| EFPL | Extended Flight Plan |
| FIXM | Flight Information Exchange Model |
| FL | Flight Level |
| FLAS | Flight Level Allocation Scheme |
| FTTS | Sveučilište u Zagrebu Fakultet prometnih znanosti |
| GML | Geography Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IATA | International Air Transport Association |
| ICAO | International Civil Aviation Organization |
| IRI | Internationalized Resource Identifier |
| JKU | Johannes Kepler Universität Linz |
| KG | Knowledge Graph |
| LoA | Letters of Agreement |
| ML | Machine Learning |
| NEST | Network strategic tool |
| RBT | Reference Business Trajectory |
| RDF | Resource Description Framework |
| RDFS | Resource Description Framework Schema |
| ROC | Rate of Climb |
| ROD | Rate of Descent |
| SHACL | Shapes Constraint Language |
| SI | Situation of Interest |
| SPARQL | SPARQL Protocol and RDF Query Language |
| TTL | Turtle |
| UPM | Universidad Politécnica de Madrid |


AIRSA


SESAR
JOINT UNDERTAKING


University of Zagreb


Technische
Universität
Braunschweig


POLITÉCNICA
UNIVERSIDAD
POLITÉCNICA
DE MADRID


skyguide


zh
aw
School of
Engineering
ZAV Center for Aviation


JKU
JOHANNES KEPLER
UNIVERSITÄT LINZ


onsulting
Slot