33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

Graph-Based Knowledge Representation and Algorithms for Air and Maintenance Operations

Ella Olsson[1], Olov Candell[1], Peter Funk[2], Rickard Sohlberg[2], Miguel Castaño[3], Mats A. Gustafsson[1], Peter Bladh[1]

[1]Saab Aeronautics, Linköping, Sweden
[2]Mälardalen University, Västerås, Sweden
[3]Luleå University of Technology, Div. of Operation & Maintenance Engineering, Luleå, Sweden

Abstract

This work presents an approach for information exchange between adjacent air operations domains by means of graph technologies. The approach has the ability to leverage interoperability and collaboration between air- and ground-based systems and stakeholders in respective domains. In its foundation, it provides a means for relevant actors to access and assess relevant data, information and knowledge, and thus provide input in terms of viable action alternatives in a complex and dynamic operational context. As a proof-of-concept, we have utilized a full-stack application framework to implement a decision support demonstrator for operational aircraft maintenance. Our solution facilitates a lightweight and dynamic representation of relevant domain knowledge, readily available for exploitation by graph algorithms, adapted to our domain. We have based our implementation on the full-stack application framework Grand-Stack, which is an architecture designed to exploit the power of graphs throughout its stack.

Keywords: Graph Database, Graph Algorithms, Interoperability, Aircraft Maintenance, Grand-Stack

1. Introduction

The increasing dynamics envisioned in future military air operations raises new needs for more agile and highly mobile air base logistic and maintenance capabilities, and a faster, more flexible response to changes in operational needs and challenges [1]. Publications on U.S. Air Force Future Operating Concepts speaks in terms of operational agility, superior decision speed and dynamic command & control (C2), when discussing a “unifying principle that guides how the Air Force conducts its core missions in the future” [2]. This relates well to the Swedish air force approach, with an overarching need for an airbase system that provides the air units a greater freedom of dynamically dispersed operations across large geographical areas, with good endurance and efficient resource utilization [3]. Interest and system needs for air operations is also shifting towards system-of-systems (SoS) and interoperability [4][5]. I.e. solutions where systems, to a greater extent, are able to exchange information and services with other systems, regardless of operational domain type, vendor, units or coalition partners. Hence, maintenance and support systems for air operations will need better capabilities for collaboration with other systems, spanning both ground-based and air-borne systems, as well as with related system for command and control (C2) and defense logistics.

This paper presents an approach for information exchange between adjacent air operations domains by means of graph technologies. The approach has the ability to leverage interoperability and collaboration between air- and ground-based systems, as well as coalition partners. In its foundation, it provides a means for relevant actors to access and assess relevant data, information and knowledge, and thus provide input in terms of viable action alternatives in a complex and dynamic operational context. A graph model forms the basis for information representation, and its design has been influenced by the Swedish air operations doctrine and policies in order to facilitate integration of tactical/operational level of logistic C2 as well as with envisioned air operations C2 structures and needs.

The graph model is envisioned to be feed with data from various types of information sources including both ground-based and airborne components. These components produce a vast amount of technical data from sources such as integrated health management systems (IVHM) and maintenance and logistics support systems respectively, as well as tactical data from tactical ground support systems and C2.

The graph model is implemented in a Neo4j database, which acts as data layer a proof-of-concept (PoC) demonstrator application. The PoC itself is based on the Grand Stack architecture, which is an architecture, designed, “using a consistent data model in order to leverage graphs throughout the stack” [6]. Further, “The combination of GraphQL, React, Apollo, and Neo4j Database, aka “the GRANDstack,” provides an easily adoptable end-to-end solution perfect for building fullstack GraphQL applications.“ [6]

2. Background and Purpose

2.1 Background and Initial Problem Statement

The rationale for the study has its based in general, future air operations requirements and needs, as deducted from the public Swedish and U.S. air operations related doctrine and policies [1][2][3][4][5]. From these needs, we have used a top-down approach to link C2 rationales and principles to the underlying of organizational structure and applied processes. The result has provided us with detailed information of how tactical needs (e.g. Air Tasking Order (ATO)) and maintenance capabilities are interconnected and how information is exchanged from higher to lower organizational units [7]. Further, this information exchange has provided additional insights on how tactical needs (in terms of orders) are transformed to more specific mission production requirements and plans, and how to we can match these mission requirements to actual maintenance capabilities available as lower organizational units and services within the air base system. Within the air base system, maintenance and logistics planning also requires a high level of logistics functional services and situational awareness [8], on macro as well as on micro levels. Furthermore, the resulting actions often need to follow complex hierarchical pathways through a decision hierarchy for the execution to commence according to plan. The addition of time- and/or resources constrained environments further stresses the planning task and the risk for errors increase [9].

2.2 Technology-Driven Solutions

From a technological perspective, the last decade has spawned major breakthroughs in the area of web-based information technologies and their applications, driven by the advent of social networks and streaming media services. These technologies has disrupted the way we communicate with each other and the way we consume media. With a vastly growing user base, combined with high-speed networking and an increasing set of social use cases, these applications has in term adopted smarter and more effective information and storage solutions [11], tailored to the specific requirements of e.g. social network applications. Such applications live in a networked environment and handles networked data in terms of interconnected and networked entities. Technologies that have made their way also into defense applications [9].

These requirements has given rise to a new breed of graph-based database managements systems, that differs from the commonly used Relational Database Management System (RDBMS)[12]. Whereas the RDBMS is based on relational algebra and it is good at representing stable data with fixed pre-set, e.g. one-to-one, one-to-many etc. relations [13], the graph model is based on a network model with a close similarity between its conceptualization of the world and its physical implementation. A graph model usually consists of nodes, relationships, properties, and labels. Where nodes can be seen as information slots/documents that stores information in strings similar to documents and relationships is used to connect nodes together in a graph structure. A relation can also have properties that can extend its sematic meaning, or be used as metadata for the graph algorithms. In addition, a graph database usually has a performance increase when handling connected data relative to RDBMS and this performance remains relative constant even as the data set grows [14].

Given the node topology of modern air operations where air-, and ground-based systems, as well as coalition partners can be seen as interconnected nodes in a complex and dynamic SoS architecture - we believe that the concept of a graph model naturally lends itself well as a basis for a real-time, on-line, information sharing in modern air and maintenance operations

3. A Graph-Based Application for Planning and Decision Support in Air- and Maintenance Operations

The objective of this research is three-fold:

1. To investigate the application of graph models in air-, and maintenance operations. Previous research findings has lead us to believe that this approach is viable solution for an effective multi-domain integration and information sharing framework, able to mirror the dynamic context of todays and future military air- and maintenance operations. This has lead us to develop a graph representation of air-, and maintenance operations, which is further described in section 3.2.

2. In addition, our objective is also to raise the Technology Readiness Level (TRL) [15] from 2-3 to 4-5 for the proposed technologies within the operation, maintenance and logistics support application domain. This has been done by leveraging our previous results from a conceptual level to a demonstrator level, which will enable us to validate our concept and components using a graph model implementation, which also will form the basis for additional functionality. The demonstrator application is further described in section 3.3.

3. In our previous research, we have developed an enterprise architecture model that models the integration of air and maintenance operations [7]. The model goes from high-level operational views that represent the interaction and relations of air and maintenance operations at the operational level. Further, we have drilled down along the operational view into service and system views that models the inherent functionality, systems and organizations of air and maintenance operations. At the system level, care has been taken to model specific maintenance functions such as flight line servicing. Combined, the architecture views can be seen as a rudimentary command and control interface for air and maintenance operations. This interface aims to provide services that helps relevant actors to assess relevant data and information and to provide input in terms of viable action alternatives, actors may include, maintenance ground personnel, commanders and other stakeholders. The interface is further described in section 3.1 along with its implementation in section 3.4.

3.1 Command and Control Interface

We have elicited a set of high-level requirements based on the capability taxonomy as defined in [7], which models a set of high-level capabilities of air and maintenance operations, including planning, control and execution of military air operations. The capability taxonomy contains three main capabilities as listed below [3][7]:

1. **Support Operational Air units -** Maintaining military air operations is the main capability of the aircraft maintenance domain. This capability includes the ability to set up, maintain air bases, and serve flight crews. It also includes the ability to plan, execute and follow-up of maintenance operations.

2. **Planning, Control and Coordination of Military Air Operations -** The basis for planning and coordination of military air operations is to have the ability to maintain an event-driven readiness and an ability to concentrate resources in time and space in order to produce required air missions.

3. **Conduct Air Operations with operational air units -** This capability includes the ability to plan, execute, evaluate and report military air operations. We will focus on the capability to

perform air operations with the JAS Gripen combat air system. The types of missions that the JAS Gripen fighter system is capable of is depicted in the figure below, shortly explained thereafter. All imposing a ~~unique~~ set of type specific requirements and restrictions on the utilization of the related maintenance resources.

The above-identified capabilities may be supported by a set of standardized services, which can be related to the NATO C3 Technical Services Taxonomy [8]. In this taxonomy, we identify a sub-set of services, namely the Logistics Functional Services. These services provide unique computing and information services for logistics support by supporting a set of (military) activities relating to planning, execution, sustainment, and maintenance of forces, through:

- **Recognized Logistic Picture Services (RLP) -** "provide the means to create, manage and disseminate the Recognized Logistics Picture. These services will generate a de-conflicted and agreed picture of the logistics environment through the collection, aggregation, correlation and fusion of information from multiple sources" [8].

- **Logistics Planning Services -** "deliver functionality to automatically access, process and disseminate information related to threat environment; identified available logistic nodes; available infrastructure and its suitability for logistic operations; host-nation support capabilities and capacity; military interoperability and cooperation agreements; environmental protection; climate; and terrain." [8].

- **Movement Services -** "deliver functionality to automatically access, process and disseminate information to coordinate and control movements to/from and within the theater of operations." [8]

- **Asset Tracking Services -** "deliver functionality to automatically access, process and disseminate asset information pertaining to location, status, and condition of products, vehicles, and other assets." [8].

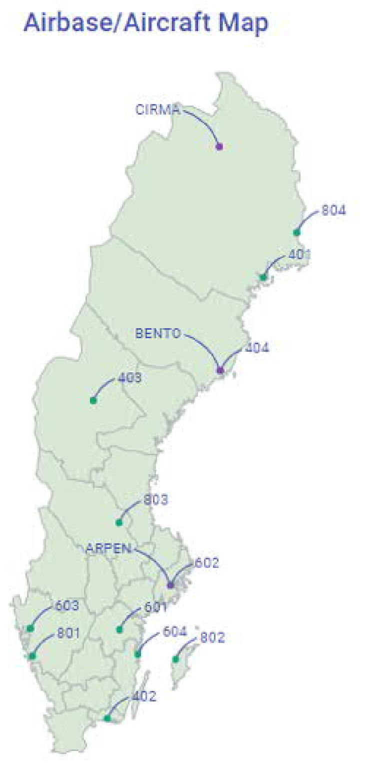


Figure 1. A birds-eye view of the **Logistic situational awareness** (LSA) demonstrator use case presenting spatial aspects of ingoing monitored entities.

3.2 A Graph-Based Information Model of Air- and Maintenance Operations

In this research, we have created our graph models from a domain-centric view based on our background knowledge, combined with recent finding of information entities and relationships in the air and maintenance operational domains. The model has been created using the process of graph data modeling. We base the process upon our previous studies of the domain of interest [7] and previous work in which we have identified key entities in the air and maintenance operational domains, as well as the C2 domain. These entities has then been modeled as a connected graph of nodes and relationships with properties and labels using the concept of a property graph model where the nodes store information about entities data records, and directional relationships are used to connect nodes. Further, nodes stores data in key-value pairs in nodes and relationships and labels are used to group nodes and relationships. The domain model is depicted in the figure below.

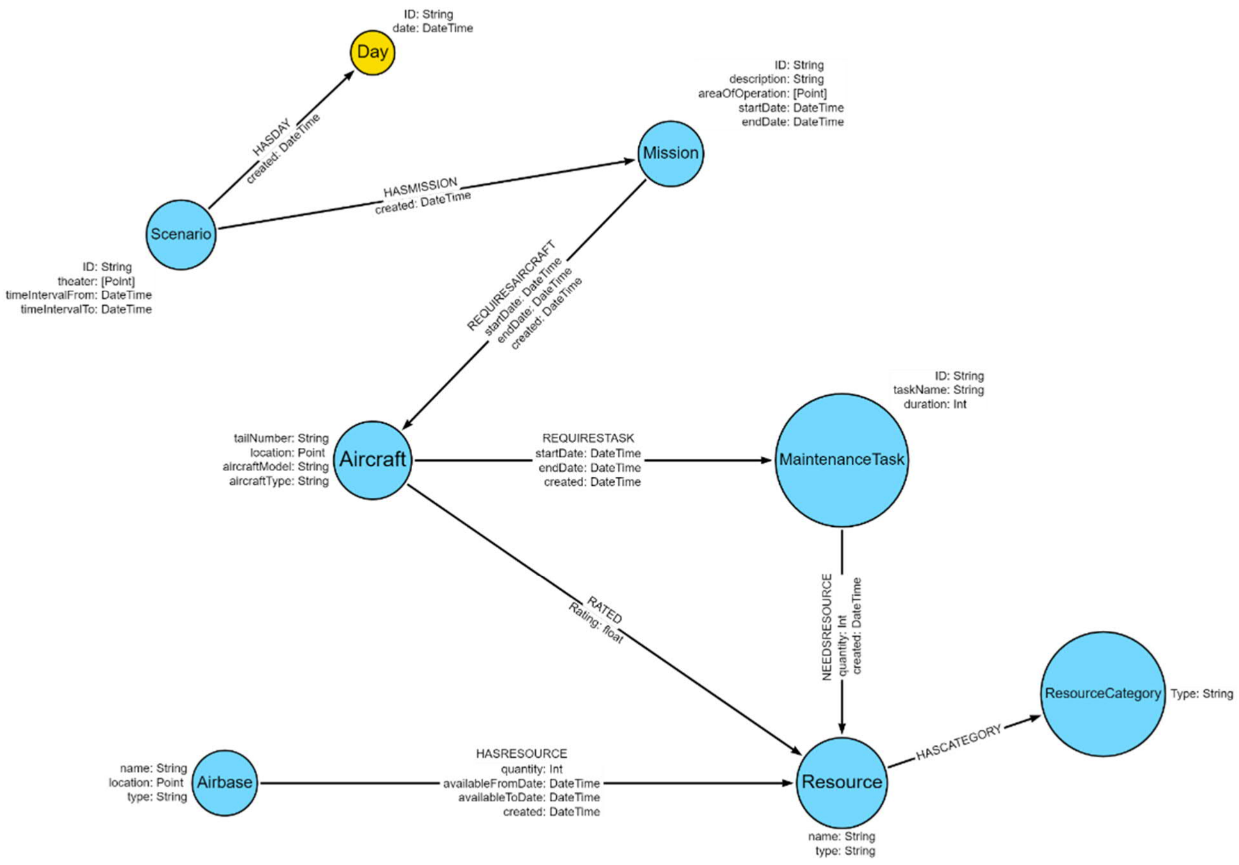


Figure 2. The complete domain model [7] of c2 and maintenance operations.

3.3 Demonstrator Architecture

This section presents the system-wide design decisions of the demonstrator application. We will describe the behavioral design and other design decisions affecting the selection and design of the software components that make up the matching system. The architecture is based on the GRANDstack framework [16] and the grand-stack-starter application, which can be found here [17]. In addition, we employ additional server and planner components in the back-end as further detailed later in this section.

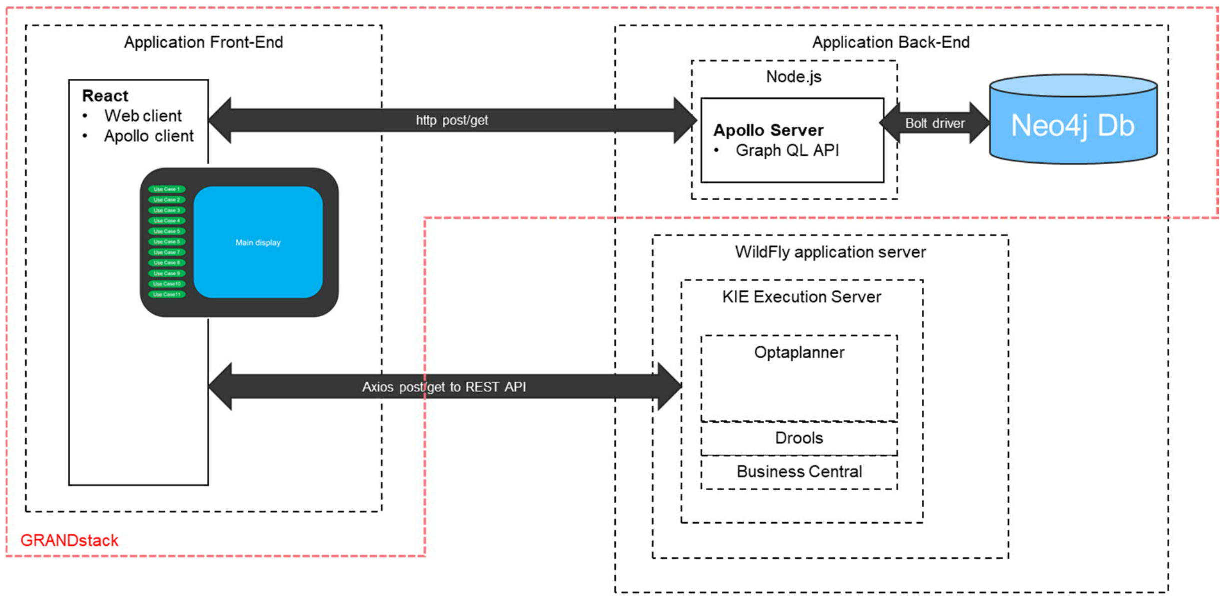


Figure 3. The system-wide architecture of the demonstrator application.

GRANDstack - The **GRANDstack** framework is centered on the GraphQL query language with the aim of "… adapting your application's design and data store to leverage graphs throughout the stack" and to "…decrease friction by using a consistent data model improving developer productivity, performance and maintainability."[16]. The GRANDstack framework is an assembly of widely used and adopted web-based components that combined leverages a full application stack. These components are:

React is a component based library for building user interfaces using JavaScript[16]. We use the React library for the implementation of our use cases, which in term has been derived from our requirements. React is used to fetch and process data as well as to update data in the database and to render the data on in a web browser. We also use react library components to render the GUI of our application. GUI implementation in React involves the design of views that map application data to React handles which in turn has the capability to provide updates efficiently in response do data or user updates. We use a specific React view for each use case and each view is then implemented as a tab in the pattern of the grand-stack-starter application. This enables to easily compose our use cases together into a demonstrator application.

Apollo - We use the **Apollo Client** and the **Apollo server** for building our GraphQL API. Apollo server resides on the database side, or the back-end of our application and it is used to abstract Neo4j cypher queries into the GraphQL language. We use the Apollo client on the client side, or the front-end of our application, for querying the GraphQL APIs. The Apollo client can be used in conjunction with many frontend frameworks, including the React framework, which is used in the demonstrator application [16].

Neo4j - We use the **Neo4j** native graph database for persistent data storage. Neo4j is a well-established, open source graph database that supports many programming languages and it runs on most commonly used operating systems. It is well suited for use in applications such as artificial intelligence and real-time recommendations, among other. Neo4j uses a native graph model for storage and it enables querying its data as a graph. In our application, we use the graph data model in all layers from storage, to queries and GUI. For storage, Neo4j uses a Property graph data model which is a directed graph represented by nodes and relations between nodes. Both nodes and relations can have attributes and nodes can also have labels which is a set oriented concept [18] that aids in e.g. grouping of nodes, creation of sub-graphs and querying etc.

GraphQL is a data layer agnostic specification for building APIs [16]. The data used in the API specification is strictly typed and the type definitions makes up the GraphQl API. A GraphQL client uses these types when it makes its API calls. Types are defined with data fields and connections to other types. Relations are also explicitly defined as types. We use the GraphQL api to abstract the Noe4j database. We describe node labels as type definitions and attributes as fields. We also explicitly describe node relations as types and define how they connect types. We call the API using GraphQL queries, which traverse the graph data mapping it to type definitions. A GraphQL query hierarchically traverses through subsets of the graph database guided by the structure of the query and the relations in the database.

Node.js - The backend of our demonstrator application is a **Node.js** application that serves a GraphQL endpoint over HTTP. The Node.js application uses an Apollo server that connect to the Neo4j database via a Bolt protocol [19].

Apart from GRANDstack the following additional components have been added to our architecture:

The **Kie Server** is a modular, standalone Java-based server component that can be used to instantiate and execute rules (Drools) and processes. The KIE Server exposes this functionality via REST, JMS or and Java interfaces to client application. Created as a web deployable WAR file, this engine can be deployed on any web container that can handle WAR files.

WildFly - In our demonstrator setup, we use the lightweight open-source application server **WildFly** as the container. WildFly was configured to enable Cross-origin HTTP requests (CORS) since the requests is sent from different origins.

OptaPlanner is an AI constraint solver [20] able to optimize planning and scheduling problems. We employ the OptaPlanner with **Drools** Planner, which uses the Drools Expert (rule engine) for score calculation of our planning solutions. This enables us to write scalable constraints in a declarative manner. Drools optimize a planning problem based on as set of facts, which represent the input data that is to be processed by the rules. The rules matches the facts with actions which will fire on relevant facts. In our case, we rank the facts using rule actions in order to find a viable plan. Rules are written in Drools Rule Language and stored in .drl files[21]. Both Drools Planner and the OptaPlanner are open source developed to help solve planning problems.

3.3.1 Application GUI

The main application GUI is structured similarly as the grand-stack-starter application [17] with a main dashboard and tab list where the tabs contains separate functionality. We have adopted this structure and implemented each use case as a separate tab in the tab list. The main structure of the application GUI is depicted in the figure below.

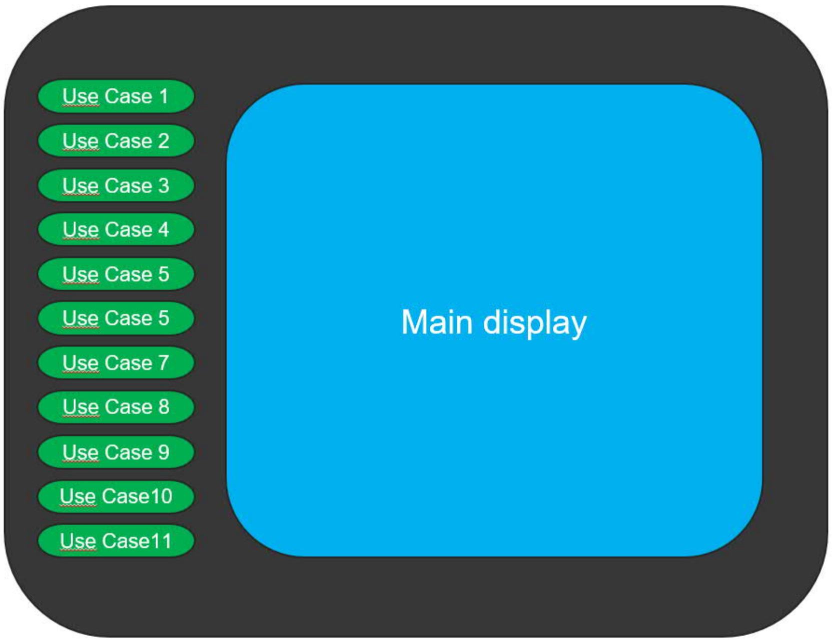


Figure 4. The general structure of the application GUI.

3.3.2 Front-End

GUI implementation in React involves the design of views that map application data to React handles which in turn has the capability to updates efficiently in response do data or user updates. We use a specific React view for each use case and each view is then implemented as a tab in the pattern of the grand-stack-starter application. This enables us to easily compose our use cases together into a demonstrator application.

The front end is comprised of a set of React JavaScript components. There is one React components for each implemented Use Case. In general, each React component implements a set of GraphQL queries and a main render() method that process results from the query and returns a formatted display which is output to the main application display in a web browser. In some cases it also updates the data in the database by mutation type queries.

Each React component that fetches and/or updates data in the database uses an instance of the Apollo Client for querying the GraphQL APIs.

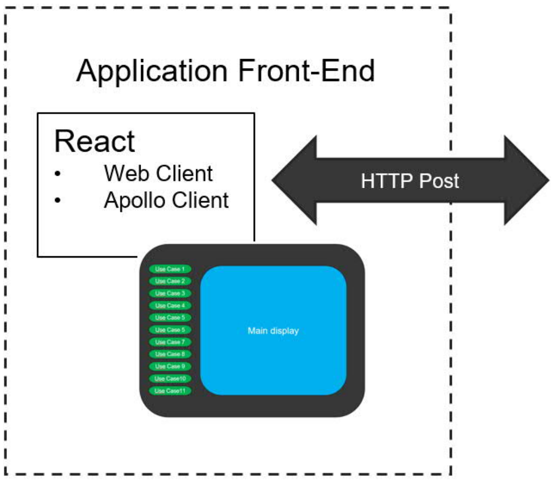


Figure 5. The demonstrator application front-end.

3.3.3 Back-End

The backend of our demonstrator application is a Node.js application that serves a GraphQL endpoint over HTTP. The Node.js application uses an Apollo server that connect to the Neo4j database via a bolt protocol [19]. The GraphQL data layer abstracts the Neo4j property graph model into a set of storage agnostic and strictly typed data types used by the GraphQL client when it makes its API calls. A GraphQL query, hierarchically traverses through subsets of the graph database guided by the structure of the query and the relations in the database. Finally we employ a Neo4j native graph database for persistent data storage where we use the Neo4j native graph model for storage and GraphQL queries.

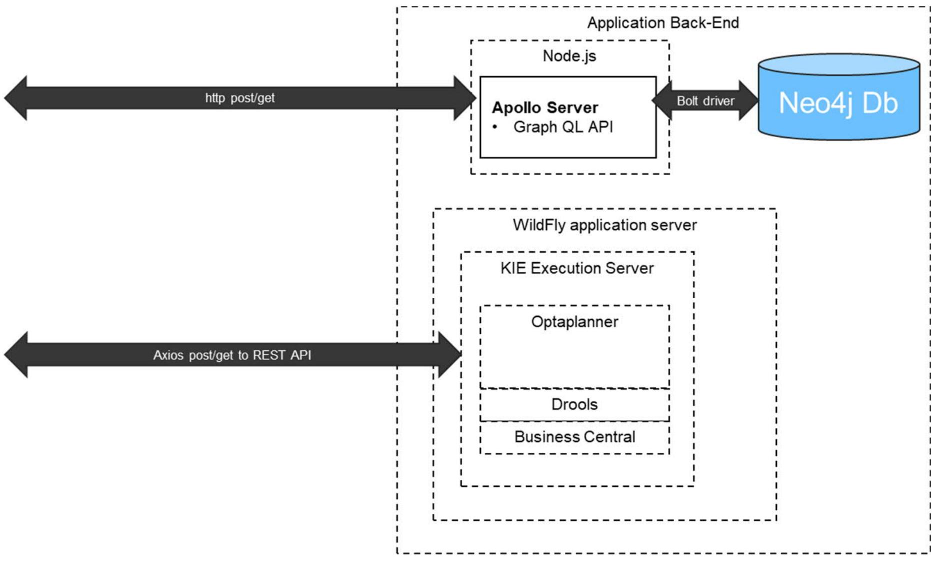


Figure 6. The demonstrator application back-end.

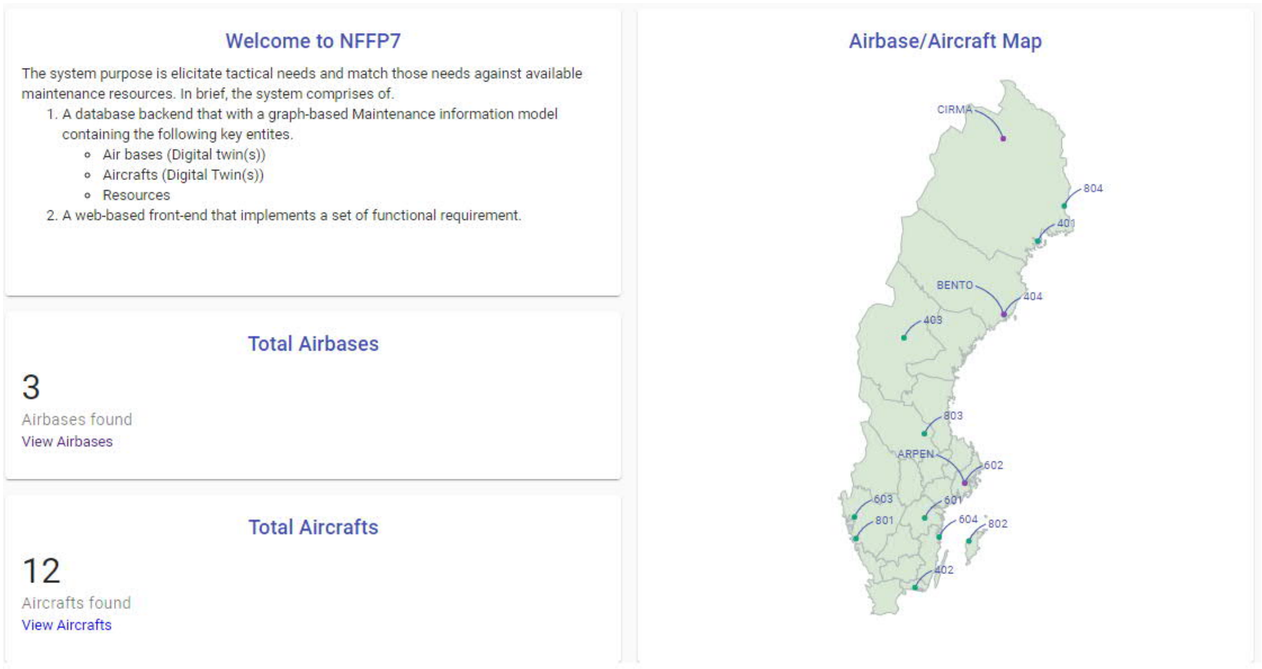
3.4 Implemented Use-Cases

On basis of the high level requirements and the identified, standardized set of services (Recognized Logistic Picture Services, the Logistics Planning Services and the Asset Tracking Services), as detailed in section 3.1, we have derived a set of use case requirements for our demonstrator implementation. The follow subsection will present each use case and its implementation in more detail.

3.4.1 Logistics Situational Awareness

The logistics Situational awareness services aims to improve the awareness of what is happening in an area of interest with focus on logistics locations. In our specific case we focus on the key drivers of logistics requirement in terms of the locations of the aircrafts and the key suppliers of logistics capabilities: the air base system.

Locate aircrafts and air bases on map - The goal of this use case is to present the location of each available aircraft and air base on a map. The use case requires that there are recent updates of the aircraft and air base entities, including their locations stored in the database. This will result in a presentation of each available aircraft and air base on a map in the main application view. The use case trigger is whenever the specific use case tab is selected in the application. The figure below depicts the map view of the use case.



Welcome to NFFP7

The system purpose is elicitate tactical needs and match those needs against available maintenance resources. In brief, the system comprises of.

1. A database backend that with a graph-based Maintenance information model containing the following key entites.
   - Air bases (Digital twin(s))
   - Aircrafts (Digital Twin(s))
   - Resources
2. A web-based front-end that implements a set of functional requirement.

Total Airbases

3

Airbases found

View Airbases

Total Aircrafts

12

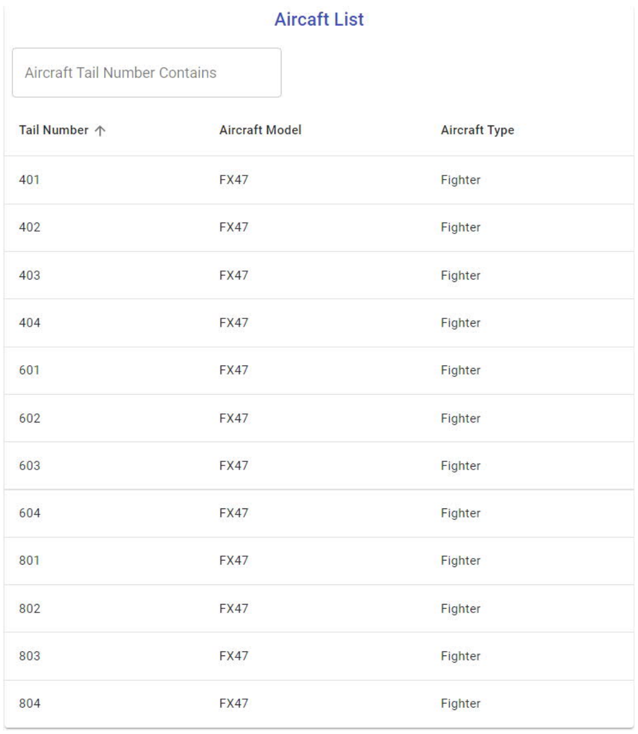
Aircrafts found

View Aircrafts

Figure 7. A birds-eye view of our LST use case.

3.4.2 Asset Tracking

See Available aircrafts and/or air bases - The goal of this use case is to list available aircraft, or available air bases in the main application view. The use case requires that there are recent updates of available aircraft and air base entities, stored in the database. This will result in either a list of available aircrafts, or available air bases, presented in the main application view. The use case trigger is whenever the specific use case tab for available aircrafts or available air bases is selected in the application. The figure below depicts the aircraft and the air base view of this use case respectively.

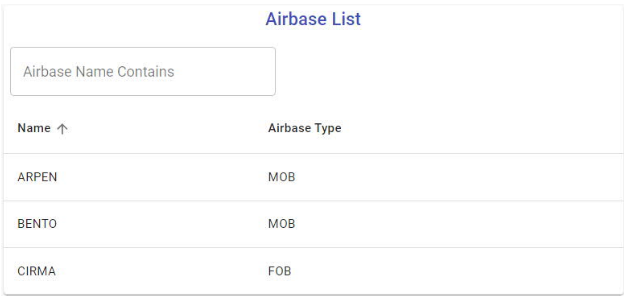


Aircaft List

Aircraft Tail Number Contains

Tail Number ↑	Aircraft Model	Aircraft Type
401	FX47	Fighter
402	FX47	Fighter
403	FX47	Fighter
404	FX47	Fighter
601	FX47	Fighter
602	FX47	Fighter
603	FX47	Fighter
604	FX47	Fighter
801	FX47	Fighter
802	FX47	Fighter
803	FX47	Fighter
804	FX47	Fighter

Figure 8. Demonstrator view of Aircraft asset tracking.



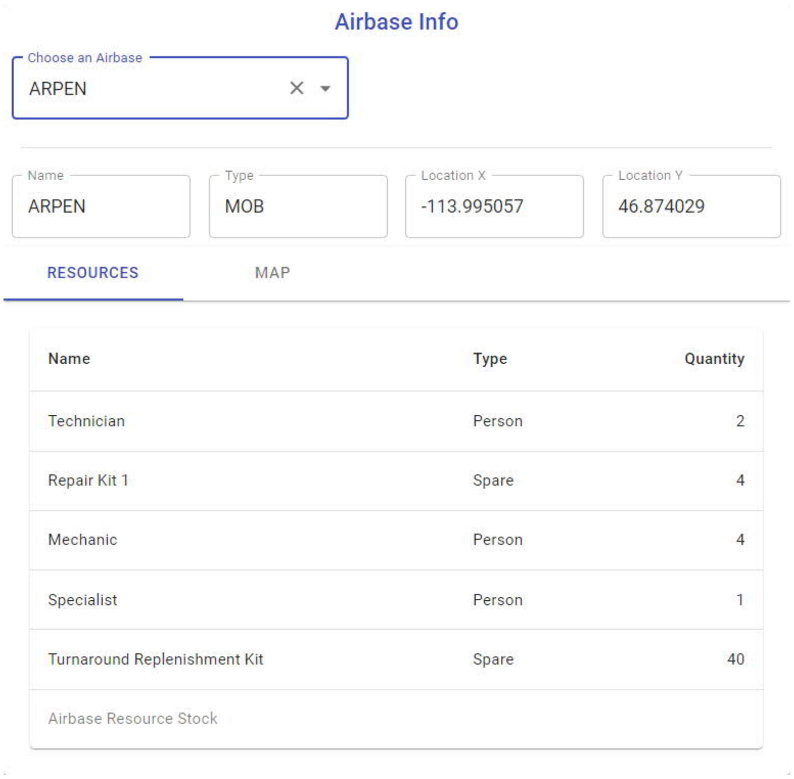
Airbase List

Airbase Name Contains

Name ↑	Airbase Type
ARPEN	MOB
BENTO	MOB
CIRMA	FOB

Figure 9. Demonstrator view of Air base asset tracking.

See available resources at air bases - The goal of this use case is to list available resources located at a specific air base and present details about each resource in the main application view. The use case requires that there are recent updates of available resources for each air base, stored in the database. This will result in a list, presenting details about each available resource for the selected air base in the main application view. The use case is triggered whenever the specific use case tab is selected in the application and a specific air base is selected in the GUI. The figure below depicts the use case main view.



Airbase Info

Choose an Airbase: ARPEN

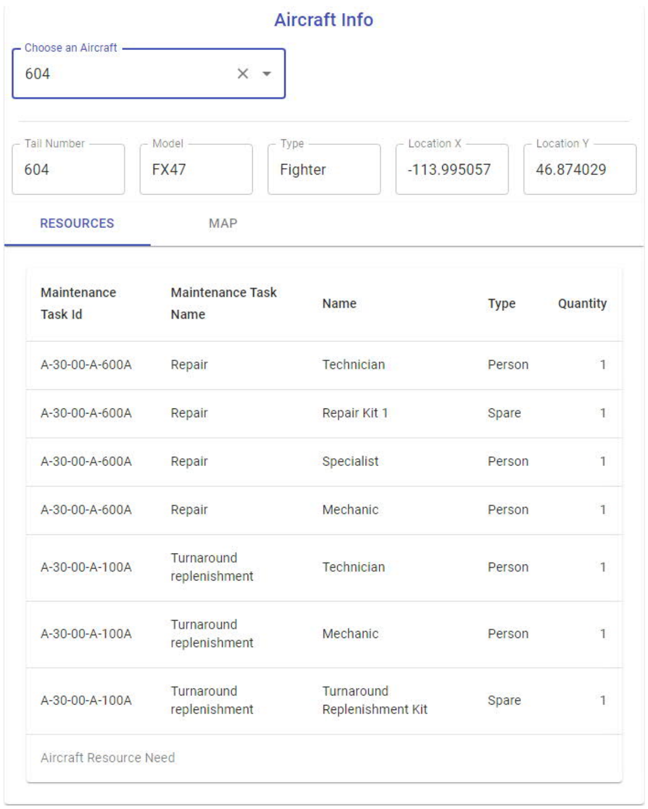
Name: ARPEN
Type: MOB
Location X: -113.995057
Location Y: 46.874029

Name	Type	Quantity
Technician	Person	2
Repair Kit 1	Spare	4
Mechanic	Person	4
Specialist	Person	1
Turnaround Replenishment Kit	Spare	40

Airbase Resource Stock

Figure 10. Demonstrator view of available resources located at a specific air base.

See resource requirements for individual aircrafts - The goal of this use case is to list resource requirements for a selected aircraft and present details about each resource requirement in the main application view. The use case requires that there are recent updates of the resource requirements of the selected aircraft, stored in the database. This will result in a list, presenting details about each resource requirement for the selected aircraft in the main application view. The use case is triggered whenever the specific use case tab is selected in the application and a specific air base is selected in the GUI. The figure below depicts the use case main view.



Choose an Aircraft: 604

Tail Number	Model	Type	Location X	Location Y
604	FX47	Fighter	-113.995057	46.874029

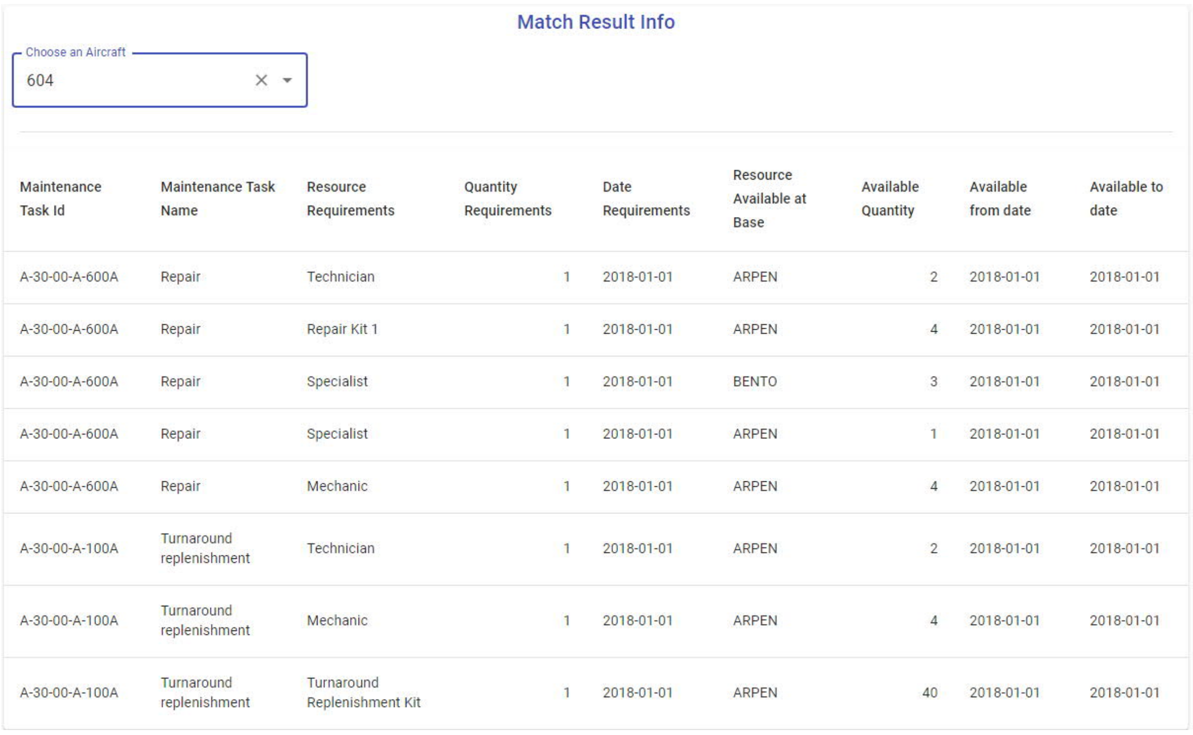
Maintenance Task Id	Maintenance Task Name	Name	Type	Quantity
A-30-00-A-600A	Repair	Technician	Person	1
A-30-00-A-600A	Repair	Repair Kit 1	Spare	1
A-30-00-A-600A	Repair	Specialist	Person	1
A-30-00-A-600A	Repair	Mechanic	Person	1
A-30-00-A-100A	Turnaround replenishment	Technician	Person	1
A-30-00-A-100A	Turnaround replenishment	Mechanic	Person	1
A-30-00-A-100A	Turnaround replenishment	Turnaround Replenishment Kit	Spare	1

Aircraft Resource Need

Figure 11. Demonstrator view of to list resource requirements for a selected aircraft.

3.4.3 Planning

Cypher-Based Matching - The goal of this use case is to match resource requirements for a selected aircraft and present details about the match result as a list in the main application view. The use case requires that there are current information of the specific resource requirements of the selected aircraft as well as current information about available resources on available air bases, stored in the database. The match result will be presented in a list, presenting details about each resource requirement and the matching resource capability and on which air base the specific resource is available at. The results is presented in the main application view. The use case is triggered whenever the specific use case tab is selected in the application, and a specific aircraft tail number is entered in the GUI. The figure below depicts the use case main view.



Match Result Info

Choose an Aircraft: 604

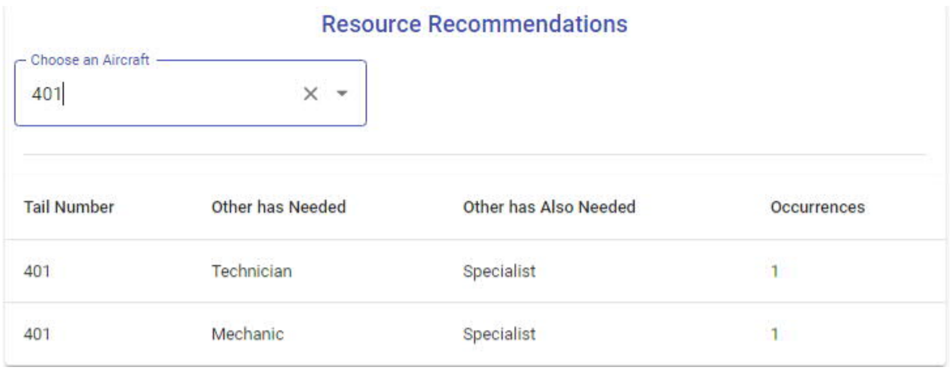
Maintenance Task Id	Maintenance Task Name	Resource Requirements	Quantity Requirements	Date Requirements	Resource Available at Base	Available Quantity	Available from date	Available to date
A-30-00-A-600A	Repair	Technician	1	2018-01-01	ARPEN	2	2018-01-01	2018-01-01
A-30-00-A-600A	Repair	Repair Kit 1	1	2018-01-01	ARPEN	4	2018-01-01	2018-01-01
A-30-00-A-600A	Repair	Specialist	1	2018-01-01	BENTO	3	2018-01-01	2018-01-01
A-30-00-A-600A	Repair	Specialist	1	2018-01-01	ARPEN	1	2018-01-01	2018-01-01
A-30-00-A-600A	Repair	Mechanic	1	2018-01-01	ARPEN	4	2018-01-01	2018-01-01
A-30-00-A-100A	Turnaround replenishment	Technician	1	2018-01-01	ARPEN	2	2018-01-01	2018-01-01
A-30-00-A-100A	Turnaround replenishment	Mechanic	1	2018-01-01	ARPEN	4	2018-01-01	2018-01-01
A-30-00-A-100A	Turnaround replenishment	Turnaround Replenishment Kit	1	2018-01-01	ARPEN	40	2018-01-01	2018-01-01

Figure 12. Demonstrator view of the match list for resource requirements for a selected aircraft.

Data driven recommendations ('shopping' analogy) - The goal of this use case is to predict future resource requirements for a specific aircraft based on other aircraft's resource requirements. This is done on basis of the similarity of other aircrafts resource requirements compared with the aircraft in question. Similarity is calculated based on resource categories. In our information model, we have three resource categories: Person, GSE and Spare. The use case requires that there are current information of the specific resource requirements of the selected aircraft as well as current information about resource requirements of all other aircrafts, available in the database.

We aim to use collected data as a basis to infer probable future maintenance needs for an aircraft. We solve this by adapting a commonly used graph-based recommendations algorithm [22][22][24][25] to into our maintenance domain. The algorithm we apply is a type of content-based filtering algorithm where we infer potential future maintenance resource needs for a specific aircraft based on similarities of the aircrafts previous resource needs with other aircrafts resource needs. The content-based filtering algorithm is able to identify similarities between entities using attributes of the items. We apply this algorithm to our maintenance domain data in order to find similarities between maintenance needs of different aircrafts in order to infer future maintenance resource needs for a specific aircraft.

The recommendation result will present a prediction of what resource needs the aircraft in question might have, based on resource needs in the same categories that other aircrafts with similar resource need profiles has required, but has yet not been required by the aircraft in question. The resource prediction recommendation is presented as a list where the resources needed by the aircraft in question is matched up with future recommendations of resources that has been needed by other aircrafts with similar resource need profiles. The list is presented in the main application view. The use case is triggered whenever the specific use case tab is selected in the application, and a specific aircraft tail number is entered in the GUI. The figure below depicts the use case main view.



Resource Recommendations

Choose an Aircraft: 401

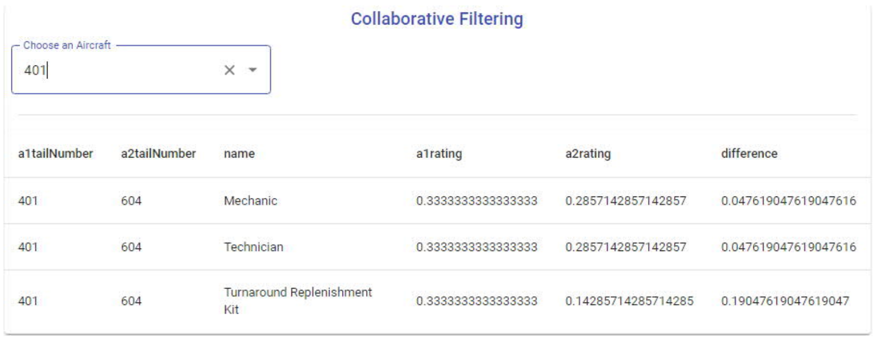
Tail Number	Other has Needed	Other has Also Needed	Occurrences
401	Technician	Specialist	1
401	Mechanic	Specialist	1

Figure 13. Demonstrator view of resource requirements prediction for a specific aircraft based on other aircraft's resource requirements.

Data driven recommendations based on similarity (collaborative filtering) - The goal of this use case is compare an aircraft with other aircrafts on basis of a similarity measure, which is based on a relative resource need for each aircraft. The result is a ranked list of similar aircrafts; compared to the one in question, based on the resource need measure. Ranking is done by calculating the relative use of a specific resource for each aircraft and compare this measure with all other aircrafts. In our information model, we can calculate the relative use of a resource for a specific aircraft by dividing it is usage with the sum of all resources, used by the aircraft. We can find all resources using the relation [RequiresTask], as depicted in Figure 2 between the aircraft and its maintenance tasks and to further follow the relation from each maintenance task to its resource requirements.

Note: In this case, we use a collaborative filtering algorithm [22][22][24][25] for assessing the similarity of aircrafts maintenance resource needs. The collaborative filtering algorithm uses the notion of a [Rates] relation in order to find relevant resource recommendations based on aircrafts with a similar history of resource usage. In our case the algorithms assumes that aircrafts are similar if they hves a history of similar resource usage. We use the resource usage as analoge to having similar maintenance resource preferences. E.g., what are the resources that those similar aircrafts consumes?

The use case requires that there are current information of the specific resource requirements of the selected aircraft as well as current information about resource requirements of all other aircrafts, available in the database. The similarity result will present a list ranked by the similarity measure for each resource used by the aircraft in question, combined with the other aircrafts, which has the most similar resource usage. The list is presented in the main application view. The use case is triggered whenever the specific use case tab is selected in the application, and a specific aircraft tail number is entered in the GUI. The figure below depicts the use case main view.



Collaborative Filtering

Choose an Aircraft: 401

a1tailNumber	a2tailNumber	name	a1rating	a2rating	difference
401	604	Mechanic	0.3333333333333333	0.2857142857142857	0.047619047619047616
401	604	Technician	0.3333333333333333	0.2857142857142857	0.047619047619047616
401	604	Turnaround Replenishment Kit	0.3333333333333333	0.14285714285714285	0.19047619047619047

Figure 14. Demonstrator view of a ranked list of similar aircrafts, compared to the one in question, based on the resource need measurements.

Maintenance Planning Optimization - For maintenance planning optimization, we employ the OptaPlanner constraint solver [20], which is able to optimize planning, and scheduling problems. We define our planning problem as described below:

Our air base system has a number air bases and our operational air units has a number of aircrafts. In our problem definition, we need to maintain these aircrafts using the resources located at our air bases. Each aircraft produces its own maintenance requirements in terms of maintenance task needs and each maintenance task need contains a set of resource requirements. See Figure 15 for the conceptual data model that we use to define this problem. The problem now becomes an allocation problem where we need to assign available resources at our air bases (maintenance capabilities) to our resource requirements (tactical need) produces by the aircrafts. For this problem the following constraints has to be fulfilled:

- A specific maintenance capability must be able to handle a specific set of hard constraints formulated by the tactical need of an aircraft:

- Maintenance capability quantity: The quantity of a specific maintenance capability must be at least be the sum of the resources (tactical needs) assigned to that specific capability.

- Maintenance capability name: The name of a specific maintenance capability must be the same as the resource name (tactical needs) assigned to that specific capability.

- Maintenance capability availability: The resources (tactical needs) production date/time must lie within the window of availability (start to end date/time) of a specific maintenance capability.

- This problem is a form of bin packing. The following is a simplified example, in which we assign three tactical needs (resource requirements) to available maintenance capabilities with two constraints; quantity and name:

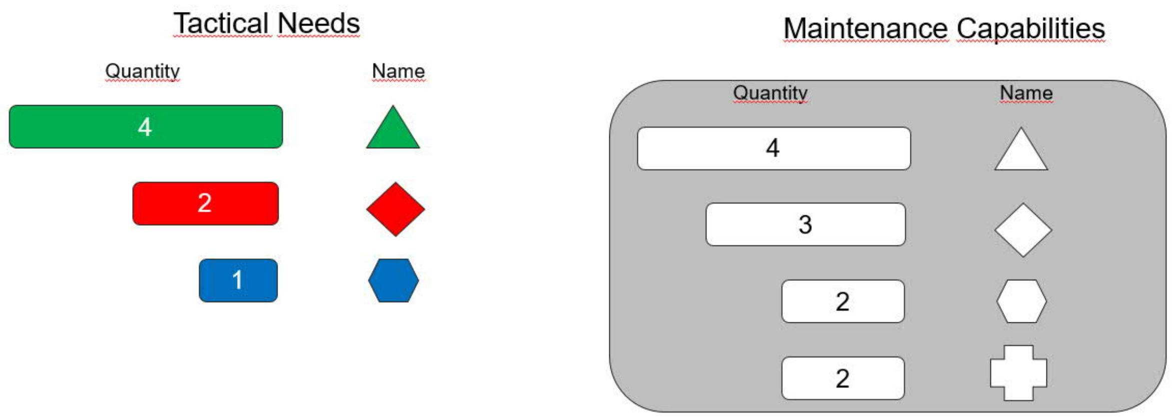


Figure 15. A conceptual view of the matching problem as implement in the OptaPlanner.

OptaPlanner uses a score calculation for finding matches between tactical needs and maintenance capabilities. Every solution that OptaPlanner produces a matching solution that has a score where the score is used to compare the quality between solutions where a higher score equals a better solution. OptaPlanner uses a Drools rule engine to assign and compare score between solutions. The resulting best solution is the solution with the highest Score, which in our case is the solution that has the highest scoring match considering the set of hard constraints as presented above. The figure below shows a matching solution using the bin packing analogy of a best planning solution.

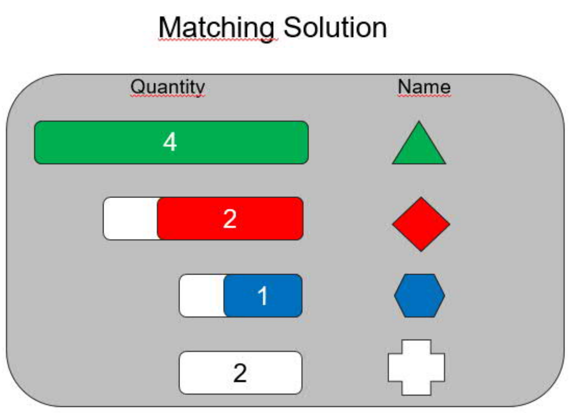


Figure 16 A conceptual view of the matching solution as implemented in the OptaPlanner.

4. Conclusions

We have shown how a decision support in operational aircraft maintenance can be implemented using as a graph-based solution. Our approach facilitates a lightweight and dynamic representation of relevant domain knowledge readily available for exploitation by powerful graph-based algorithms adapted to our domain. Our decision support solution has been implemented in a demonstrator using a state-of-the art development environment consisting of mainly free and open-source solutions for development environment and the distributed version control system. Further, we have based our implementation on the web-centric software stack grand-stack, using a consistent data model in order to leverage graphs throughout the stack. Our solution implements a set of services related to the C3 taxonomy. The resulting decision support solution can be seen as a "social network" for air and maintenance operations where relational bindings between entities ties the network together and enables the application of the same type of graph algorithms that are used in e.g. Facebook, Amazon, Netflix etc. The graph database offers a non-intrusive cross-domain information fusion service, able to leverage data- and relation-driven insights that may be unattainable using now available stove-piped systems. It also has the capability to grow incrementally as knowledge about the domain increases. It means that it is possible to add new entities, relations and even sub-graphs to the database as knowledge about the domain increases, without disturbing existing queries and application functionality. A graph database is agile in this manner as the graph data model can involve in pace with the rest of the application.

5. Future Work

5.1 Future improvements of the Graph model

A possible expansion of the graph model may look like the picture below.

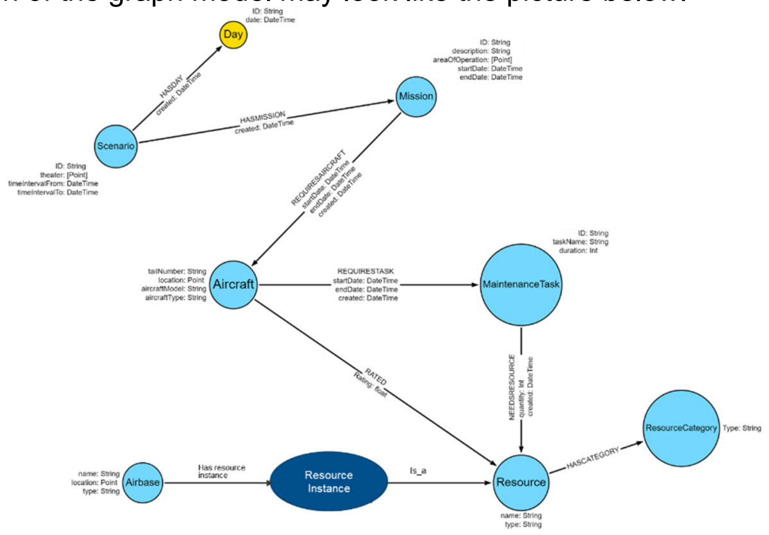


Figure 17. Suggested extension of the graph model to facilitate individual resource instances.

A new graph node "ResourceInstance" is introduced and is a Resource but acts as a specific instance of a Resource. This can be convenient when you need to plan for individual resources in a time schedule, e.g. a specific person/employee as a resource or a support equipment with limited time capability.

From a maintenance perspective, all replaceable items in an aircraft that are classified as a life-controlled item would be a candidate for instantiation as a Resource Instance. A further evolvement of the graph model would be representation of the "journal" of a specific life controlled item with its current state, maintenance, repair and overhaul log etc. Such data would typically be sourced from a Fleet/Maintenance Management System.

The current representation of an aircraft in the NFFP7 graph model consists of one node Aircraft per aircraft tail number. This is the high-level representation of an aircraft as a tactical resource as well as a maintenance object itself. Another possible expansion of the graph model would be to include nodes representing the "As maintained" structure of the aircraft with its installed equipment's (which also are considered life controlled items) and further detail the maintenance plan for each node in the "As maintained" structure. This relates to findings in NFFP5 regarding as built structures and maintenance needs information modelling.

5.2 Graph-Based Publish/Subscribe

For context integration, we have looked at publish/subscribe graph database concept [26] aimed for large-scale collaboration spaces that are both highly distributed and dynamic. This complies well with the domain of operational aircraft maintenance that contains distributed information stored in various propriety databases.

The concept relies on local databases, on-aircraft, at-aircraft, on bases and other places that advertise changes to data that they are willing to notify to clients, and publish events to the graph database through the publish/subscribe system when such changes occur.

Clients can subscribe to one or more of these change events and in consequence, are informed in a timely manner when particular state changes occur in some database, e.g. "notify me of the specific types of newly allocated maintenance resources at a specific wing". The notification data can stem from a wide variety of distributed databases [26]. An example of this concept that relies on an active database tier that is built upon a passive database system. In this example, the active database uses a model to define its reactive behavior where database events triggers rules that evaluates the events and triggers actions that formulates the task to execute after the rule has been triggered and its condition validated.

5.3 Digital Twins

Within the presented study, a work package has been performed studying capabilities required for maintenance-oriented Digital Twins (DTs) within the context of military aviation, and the challenges related to such capabilities [27]. Bosch has adopted a similar digital twin approach to this concept [28]. A digital twin can in its basic form be seen as a virtual representation of real-world asset. This is the definition made by Bosch. The virtual representation may be further extended to implement functionality and additional information. This extension is referred to as adopting a more holistic view of the real world asses where its capabilities is more closely reflected by the digital twin. Bosch has its own cloud service implementing a holistic digital twin concept. In this framework, a digital twin is a 'thing' that helps orchestrating all aspects of a physical device. The 'thing' is composed of features representing states, attributes etc. of the real world asset. Features can also be links to functions and capabilities of the physical device. The framework maps physical devices via a micro service to a virtual representation of the device. Further the virtual representation (thing) can be accessed by tailored user applications. Adopting this concept into the aircraft maintenance domain is exemplified in the section below. In Bosch's concept, the digital twin is composed of three feature types:

1. Features that represents states properties. E.g. a specific engine status variable etc.

2. Functionality such as operations, events and triggers are represented as messages. In this case the things service routes the messages to/from the physical device via a diver connectivity layer.
3. Features that represent functionality is implemented by integrating separate micro services/components. They are responsible for listening to signals/notification coming from the things micro service process and optionally send responses back.

6. Acknowledgment

This paper is a dissemination of results produced in a project within the Swedish National Aeronautical Research Program 7 (NFFP7), funded by the Swedish Innovation Agency, Vinnova. The project has recently finished, and the final report was approved by the funding agency in January 2022.

7. References

[1] U.S. Air Force. *Doctrine Publication 4-0 - Combat Support*. 2020

[2] USAF. Air Force. *Future Operating Concepts, AFFOC – a view of the Airforce in 2035.*

[3] Försvarsmakten. *Reglemente Taktik för Luftoperationer 2017* (TR LuftOp 2017) M7739-353126. Stockholm, Sweden, 2017.

[4] Försvarsmakten. *Doktrin Gemensamma Operationer 2020*. M7739-354030. Stockholm, Sweden, 2020.

[5] U.S. Air Force, U.S. Space Force. *The Department of the Air Force role in Joint All-Domain Operations*. Air Force Doctrine Publication 3-99, Space Doctrine Publication 3-99. 2021.

[6] Full Stack GraphQL Applications With React, Node.js, and Neo4j. William Lyon MEAP began October 2019 Publication in Summer 2022 (estimated) ISBN 9781617297038 300 pages (estimated) filed under Development.

[7] Olsson, E., Candell, O., Funk, P., Sohlberg, R. *Enterprise Modeling for Dynamic Matching of Tactical Needs and Aircraft Maintenance Capabilities*. In: Karim, R., Ahmadi, A., Soleimanmeigouni, I., Kour, R., Rao, R. (eds) International Congress and Workshop on Industrial AI 2021. IAI 2021. Lecture Notes in Mechanical Engineering. Springer, Cham. pp 370–383, 2022

[8] *NATO C3 Taxonomy Baseline 3.1*. AC/322-D(2019)0034 (INV). 2019.

[9] U.S. Chairman of the Joint Chiefs of Staff. *Joint Air Operations, Joint Publication 3-30*. (2019) 2021.

[10] Everstine, Brian W. *Moving from Situational Awareness to C2*. U.S. Air Force Magazine. Oct. 1, 2020. https://www.airforcemag.com/article/moving-from-situational-awareness-to-c2/

[11] *Database engine trending* https://db-engines.com/en/ranking_categories

[12] Hurlburt, George F., Thiruvathukal, George K., and Lee, Maria R. *The graph database: jack of all trades or just not SQL?.* IT Professional 19.6 (2017): 21-25. 2017.

[13] Needham, M., and Hodler , Amy E. *Graph Algorithms. Practical Examples in Apache Spark & Neo4j*. Pp. 16. 2019

[14] Graph Databases, *New Opportunities for Connected Data*, Ian Robinson and Jim Webber & Emil Eifrem. Pp 8. 2nd ed. O'Reilly Media.

[15] Technology Readiness Assessment Guide, DOE

[16] *Full Stack GraphQL Applications With React, Node.js, and Neo4j*. William Lyon MEAP began October 19 Publication in November 2021 (estimated), ISBN 9781617297038

[17] https://grandstack.io/

[18] Bruggen, R., and Mohanta , P. *Learning Neo4j Run blazingly fast queries on complex graph datasets with the power of the Neo4j graph database*. 2014.

[19] https://en.wikipedia.org/wiki/Bolt_(network_protocol)

[20] https://www.optaplanner.org/

[21] Weppenaar, D.V.I.; Vermaak, H.J. *Solving Planning Problems with Drools Planner – A Tutorial*. Interim. 10. pp91-109. 2011.

[22] Miller, Justin J. *Graph database applications and concepts with Neo4j*. Proceedings of the southern association for information systems conference, Atlanta, GA, USA. Vol. 2324. No. 36. 2013.

[23] Webber, J. and Robinson, I. *The Top 5 Use Cases of Graph Databases - Unlocking New Possibilities with Connected Data*. Neo Technology. White Paper. 2017.

https://go.neo4j.com/rs/710-RRC-335/images/Neo4j_Top5_UseCases_Graph%20Databases.pdf
[24] Meteren, Robin Van., and Someren, Maarten Van. *Using content-based filtering for recommendation*. Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop. Vol. 30. 2000.
[25] Webber, J. *Neo4j. White Paper Powering Real-Time Recommendations with Graph Database Technology*. https://neo4j.com/whitepapers/recommendations-graph-database-business/
[26] Moody, K., Vargas, L., and Bacon, J. *Integrating Databases with Publish/Subscribe.* 25th IEEE International Conference on Distributed Computing Systems Workshops. June 6-10 2005. Columbus, Ohio, USA. 2005.
[27] Castaño et al. *A Review of Digital Twin Capabilities and Challenges in the Context of IVHM in Aviation (Enablement of Digital Twin Capabilities in the context of IVHM in Aviation - Issues and Challenges)* Accepted for presentation and publication. ICAS, Stockholm, Sweden 4-9 September 2022. 2022.
[28] Bosch IOT Suite. https://docs.bosch-iot-suite.com/device-management/Devices-and-their-digital-twins.html