Grant Agreement No: 687591

14/7/17

Big Data Analytics for Time Critical Mobility Forecasting

datAcron

D1.6 Software design (interim)

Deliverable Form	
Project Reference No.	H2020-ICT-2015 687591
Deliverable No.	1.6
Relevant Work Package:	WP 1
Nature:	R
Dissemination Level:	CO
Document version:	1.0
Due Date:	30/06/2017
Date of latest revision:	13/07/2017
Completion Date:	13/07/2017
Lead partner:	UPRC
Authors:	Christos Doulkeridis (UPRC), Apostolos Glenis (UPRC), Panagiotis Nikitopoulos (UPRC), Giorgos Santipantakis (UPRC), Yannis Theodoridis (UPRC), Akrivi Vlachou (UPRC), George Vouros (UPRC), Michael Mock (FRHF), Nikos Pelekis (UPRC), Harris Georgiou (UPRC), Kostas Patroumpas (UPRC), Elias Alevizos (NCSR'D), Georg Fuchs (FRHF)
Reviewers:	Yannis Theodoridis (UPRC), Michael Mock (FHRF)
Document description:	This deliverable specifies the software design in datAcron.
Document location:	WP1/Deliverables/D1.6/Final

HISTORY OF CHANGES

Version	Date	Changes	Author	Remarks
0.1	15/5/2017	First version of table of contents	C. Doulkeridis	
0.2	12/6/2017	Added design of integrated prototype	C. Doulkeridis	
0.3	16/6/2017	Updated description of integrated prototype	Y. Theodoridis	
0.5	19/6/2017	First version for internal review	C. Doulkeridis	
0.6	26/6/2017	Revised version based on internal review comments	C. Doulkeridis	
0.9	12/7/2017	Near-final version	C.Doulkeridis, A.Glenis, P.Nikitopoulos, G.Santipantakis, A.Vlachou, G.Vouros	Version shared with all partners
1.0	14/7/2017	Final version	C.Doulkeridis, G.Vouros	

EXECUTIVE SUMMARY

This report comprises the sixth deliverable (D1.6) of datAcron work package 1 “System architecture and data management” with main objective to describe the software design in datAcron, in accordance with the requirements specified in deliverable D1.1 and the architecture specified in deliverable D1.2. The task has two iterations. This is the result of the first iteration, which focuses on fast prototyping and testing of the software architecture, aiming at providing valuable experiences as well as software components towards the second iteration. The second design iteration focuses on the integration with WP2-WP4.

TABLE OF CONTENTS

TERMS & ABBREVIATIONS

CSV	Comma-separated values
ADS-B	Automatic Dependent Surveillance-Broadcast
AIS	Automatic Identification System
RDF	Resource Description Framework
TTL	Terse RDF Triple Language

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This document is the deliverable D1.6 “Software design (interim)” of Task T1.3 of work package 1 ”System Architecture and Data Management” of the datAcron project. It defines the datAcron software design with respect to the datAcron architecture and its software modules, focusing on early prototyping and testing. This is the result of the first iteration of software design, aiming at providing valuable experiences as well as software modules towards the second iteration. The second design iteration focuses on the integration with WP2–WP4.

However, even at this first iteration, we provide an integrated prototype that implements basic functionality and contains modules from work packages WP2–WP4, focusing mainly on stream processing in real-time and online analytics, and leaving the part of offline analytics for the period after M18.

1.1 Purpose and Scope

The “Software design (interim)” aims at developing a fast prototype which can demonstrate the basic, integrated functionality of datAcron. In particular, the prototype that is developed on M18 provides integrated functionality, by implementing interconnection between modules developed as results of different work packages. In more detail, the prototype offers the following functionality on M18:

- Real-time consumption of raw surveillance data streams, on which the following operations are performed: in-situ processing (including low-level event detection (WP1) and synopsis generation(WP2)), semantic integration with weather and contextual data sources (WP1), future location prediction (WP2), complex event detection and forecasting (WP3), and real-time visualization (WP4).
- Querying integrated RDF data with a limited set of spatio-temporal queries, demonstrating the feasibility of query processing (WP1).

Deliverable D1.6 is submitted on month M18 of the project, in order to show the feasibility of realizing the datAcron scientific and technical objectives, by means of a fast prototype. Thus, the deliverable reports on the modules that have been developed, as well as the Big Data technologies adopted for processing (Apache Flink and Apache Spark) and communication (Kafka). Most importantly, this deliverable presents progress towards an *integrated* prototype, thereby demonstrating that we have achieved the interconnection of the different individual modules in a coherent integrated platform. Extending, fine-tuning, evaluating and further enhancing the capabilities of this prototype in order to achieve the scientific and technical objectives of the project is work planned for the remainder of the project.

1.2 Approach for the Work package and Relation to other Deliverables

D1.6 is based on D1.2 “Architecture Specification” delivered in M12, as it builds on and refines D1.2 into a more concrete software architecture, taking also into account the features offered by the technological solutions adopted. Specifically, D1.6 refines the datAcron architecture in a more concrete software architecture and delivers an integrated software prototype as proof-of-concept, which realizes critical flows of information in datAcron on M18. Furthermore, D1.6 presents the two main software layers: the *real-time processing layer* and the *batch processing layer*.

It should be pointed out that D1.6 is prepared during the same time that deliverables D1.3 “Cross-streaming data processing (interim)”, D1.4 “Data Integration, Management (interim)”, D2.1 “’Cross-streaming, real-time detection of moving object trajectories (interim)’, and D1.5 “Data storage and querying (interim)” are prepared. Deliverables D1.3, D1.4 and D2.1 provide detailed description on real-time layer of the datAcron architecture. Also, D1.5 describes the design of the batch processing solution developed in the context of datAcron. As such, they connect to this deliverable, and intermediate versions of D1.3, D2.1 and D1.5 have been considered during the preparation of the current deliverable.

1.3 Methodology and Structure of the Deliverable

In terms of work methodology, this deliverable takes as input D1.2 “Architecture Specification” and refines it in order to become a more concrete software architecture design, which is able to combine the individual modules developed in datAcron. During the last six months, we have been in close collaboration with partners in WP2, WP3, and WP4, that develop their solutions and prototype implementations, in order to provide specific guidelines on the technology used for inter-communication of modules, and for coordinating and orchestrating the development activities. As a result, our adopted methodology resembles agile software development, since our main focus was on early prototyping (for the first iteration) and continuous improvement and response to changes (for the second iteration). Consequently, we are able to present the design of an integrated prototype, and we refer to D1.7 “Integrated prototype (interim)” for this.

The remaining of this report is structured as follows:

- Section 2 overviews the datAcron architecture, as reported in D1.2, in terms of modules, their interactions, as well as the general guidelines for developing a fast prototype. Also, it explains how it connects to the Big Data Analysis pipeline, and presents the two main layers: the real-time processing layer and the batch processing layer.

- Section 3 presents the software design of the integrated prototype of datAcron, focusing on flows of information and the architecture that encompasses all envisioned flows and modules. Moreover, it reports on the implementation status of the integrated prototype and provides details on indicative, yet critical, implemented information flows.

- Section 4 summarizes the work reported in this deliverable, and provides the plan for the second iteration.

Finally, in the Appendix A, details are provided with respect to the datAcron cluster, the platform where we deploy the datAcron prototype for testing and experimentation.

2 The datAcron Integrated System Architecture

In this section, we briefly recall the system architecture specified in deliverable D1.2 of the project on M12. First, we describe the major steps in Big Data Analysis (Section 2.1), and then we present the overall datAcron architecture, its inputs, outputs, and modules (Section 2.2); moreover, we connect the individual modules to steps of Big Data Analysis, thus positioning them in this pipeline and illustrating their exact role. Finally, we describe the two constituent layers of the datAcron architecture: the *Batch Layer*, which is responsible for distributed storage and efficient querying of integrated spatio-temporal RDF data (Section 2.3), and the *Real-time Layer*, which involves all streaming operations and online analytics performed in real-time in the stream of information available in datAcron (Section 2.4).

2.1 Major Steps in Big Data Analysis

The project datAcron aims at recognizing and forecasting complex events and trajectories from a wealth of input data, both data-at-rest and data-in-motion, by applying appropriate techniques for Big Data analysis. The technical challenges associated with Big Data analysis are manifold, and perhaps better illustrated in [2, 3], where the *Big Data Analysis Pipeline* is presented.

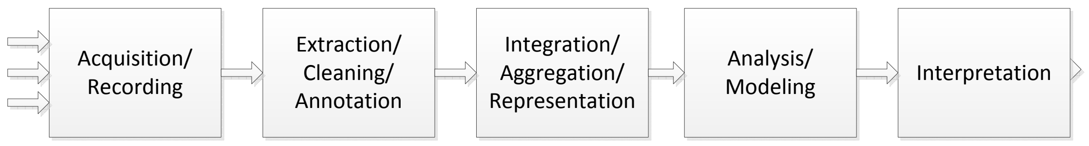
Acquisition/ Recording
Extraction/ Cleaning/ Annotation
Integration/ Aggregation/ Representation
Analysis/ Modeling
Interpretation

Figure 1: Major steps in analysis of Big Data.

As depicted in Figure 1, five major phases (or steps) are identified in the processing pipeline. Below, we explain the datAcron related activities and research challenges related to each of these phases, thus clearly showing how datAcron connects to the phases of the Big Data Analysis pipeline:

1. **Data Acquisition and Recording**: Large volumes of data are created in a streaming fashion, including surveillance data, weather forecasts, and other contextual data, and need to be consumed in datAcron. One major challenge is to perform online filtering of this data, in order to keep only the necessary data that contain the useful information. To this end, we apply data summarization techniques on surveillance data, thus keeping only the "critical points" of a moving object's trajectory, which signify changes in the mobility of the moving object. This compression technique achieves data reduction rate above 90%, without compromising the quality of the compressed trajectories. In addition, in datAcron, we have to deal with archival data sources (data-at-rest), which also need specialized data connectors depending on the provided input format.

   Another challenge in the data acquisition phase is to push computation to the edges of the Big Data management system. We perform online data summarization of surveillance data on the input stream directly, as soon as it enters the system. Moreover, we employ in-situ processing techniques, near to the streaming data sources, in order to identify low-level events, such as monitoring the entrance/leave of moving objects in specific areas of interest (such as protected marine areas).

2. **Information Extraction and Cleaning**: In datAcron, miscellaneous data in various formats are provided to the system for processing and analysis. A basic prerequisite for the subsequent analysis tasks is to extract the useful data and transform it in a form that is suitable for processing. As a concrete example, weather forecasts are provided as large binary files (GRIB format), which cannot be effectively analyzed. Therefore, we extract the useful meteorological variables from these files together with their spatio-temporal information, so that they can be later associated with mobility data.

   In addition, surveillance data are typically noisy, contain errors, and are associated with uncertainty. Data cleaning techniques are applied in the streams of surveillance data, in order to reconstruct trajectories with minimum errors that will lead to more accurate analysis results with higher probability. Indicative examples of challenges addressed in this respect include handling delayed surveillance data, dealing with intentional erroneous data (spoofing) or hardware/equipment errors. Also, with respect to data-at-rest, the involved challenges also include cleaning and extracting useful information using an RDFization approach.

3. **Data Integration, Aggregation, and Representation**: After having addressed data cleaning, the next challenge is how to integrate the heterogeneous data coming from various data sources, in order to provide a unified and combined view. Our approach is to transform and represent all input data in RDF, following a common schema (ontology) that is designed purposefully to accommodate the different data sources. However, data transformation does not suffice by itself. To achieve data integration, we apply online link discovery techniques in order to interlink streaming data from different sources, a task of major significance in datAcron.

   By means of link discovery, we derive enriched data representations across different data sources, thereby providing richer information to the higher level analysis tasks in datAcron. We refer to Deliverable D1.4 "Data Integration, Management (interim)" for details on the various types of link discovery considered in datAcron.

4. **Query Processing, Data Modeling, and Analysis**: Another Big Data challenge addressed in datAcron relates to scalable processing of vast-sized RDF graphs that encompass spatio-temporal information. Towards this goal, we design and develop a parallel spatio-temporal RDF processing engine on top of Apache Spark. Individual challenges that need to be solved in this context include RDF graph partitioning, implementing parallel query operators that shall be used by the processing engine, and exploiting the capabilities of Spark in the context of trajectory data. We refer to Deliverable D1.5 "Data storage and querying (interim)" for more details on this.

   Complex event detection is also performed in datAcron, where the objective is to detect events related to the movement of objects in real-time. Last, but not least, particular attention is set towards predictive analytics, namely trajectory prediction and event forecasting. Both short-term and long-term predictions are useful depending on the domain, and in particular for maritime, a hard problem is to perform long-term prediction. We distinguish between location prediction (where a moving object will be after X time units) and trajectory prediction (what path will a moving object follow in order to reach position X).

5. **Interpretation**: To assist the task of human-based interpretation of analysis results, as well as the detection of patterns that may further guide the detection of interesting events – tasks that are fundamental for any Big Data analysis platform – datAcron relies on visualizations and visual analytics. By means of those tools, it is possible to perform visual and interactive exploration of moving objects and their trajectories, visualize aggregates or data summaries, and ultimately identify trends or validate analysis results that would be hard to find automatically.

Thus, the connection between the datAcron architecture and the Big Data Analysis pipeline, as well as the activities related to the different phases of Big Data analysis should be clear, based on the above discussion.

2.2 Overview of the datAcron Architecture

2.2.1 Modules

At a high-level, the datAcron architecture is composed of the following six main modules (as reflected in the description provided in deliverable D1.2):

- ***In-situ Processing***: This module is responsible for executing processing tasks, such as detection of low-level events, on the premise of the actual streams.
- ***Synopses Generator***: Its main role is to provide the algorithms for trajectory compression, by eliminating many positions of moving objects that do not significantly affect the quality of the representation.
- ***Data Manager***: The data management module stores integrated data, produced by integrating data-at-rest with data-in-motion, as well as analysis results from other modules, and provides querying functionality on top of a unified view of data, due to the data integration. Persistent storage and querying of integrated data is provided by means of a *distributed RDF store*, which is a module maintained by the *Data Manager*.
- ***Trajectory Detection and Prediction***: This module performs trajectory prediction, both in real-time and offline, as well as advanced data analytics related to moving objects.
- ***Event Recognition and Forecasting***: This module is responsible for detection and forecasting of complex events related to the mobility of objects.
- ***Visual Analytics***: The exploratory data analytics module provides visualization facilities as well as the opportunity to explore different values for the parameters of the algorithms and provide better models for the event detection and trajectory prediction modules.

Table 1 shows the mapping of datAcron modules to the different steps of Big Data Analysis pipeline.

2.2.2 Inputs

Inputs to the datAcron architecture consist of data-at-rest (archival data) and data-in-motion (streaming data). Archival data are loaded in the *Data Manager*, and during this process they are transformed, integrated, and stored as will be described in detail in the following paragraphs.

datAcron module	**Step in Big Data Analysis**
In-situ Processing	Data Acquisition and Recording, Information Extraction and Cleaning
Synopses Generator	Data Acquisition and Recording, Information Extraction and Cleaning
Data Manager	Data Integration, Aggregation, and Representation, Query processing
Trajectory Detection and Prediction	Data Analysis
Event Recognition and Forecasting	Data Analysis
Visual Analytics	Data Analysis, Interpretation

Table 1: The modules in the datAcron integrated prototype.

On the other hand, a distinction is made for streaming data, namely whether they are positional data describing the spatio-temporal movement of objects (trajectories) or not. Trajectories are treated as "first-class" citizens in datAcron, thus trajectory data is summarized (at *Synopses Generator*) and associated with low-level events (during *In-situ Processing*). Also, they are integrated in the *Data Manager* module, before storing, with existing (static) data, such as ports, airports, information about the moving object (vessel/aircraft type, model, etc.). Other streaming data, such as weather forecasts, flight plans, regulations, etc., are directly fetched by the *Data Manager*, in order to be integrated with the other available data.

2.2.3 Flows of Information

In the datAcron architecture, we have identified different *flows of information*, which will be explained in detail in Section 3. However, to make this section self-contained, we briefly explain how the input data is processed and made available to the different datAcron modules.

The basic idea is that raw surveillance data is accessed as an incoming data stream in datAcron. This stream is processed by different datAcron modules that enrich it with extra information, including low-level events, detection of "critical points" with annotations of kinematic nature, weather-related information, as well as information and links to other data sources (e.g., contextual). In more detail, this stream contains cleansed data, after removing noisy raw data. The remaining raw data positions are either tagged or not by datAcron modules. A tagged position means that it is one (or both) of the following: it is a critical point or it is associated with a low-level event (e.g., enter an area of interest). Non-tagged positions are positions that do not contribute any important information, therefore, they can be omitted from further processing (at least, they do not deserve to be stored in the datAcron store). This single stream is available to all data analytics modules in real-time, and it is also directed to the *Data Manager* for storage and future batch processing.
The afore-described stream is available to *all modules* that perform data analytics in real-time, namely *Trajectory Detection and Prediction*, *Event Recognition and Forecasting* and *Visual Analytics*, in order to provide the input necessary for the respective data analysis tasks. However, the intermediate output stream of each module can be accessed by any other module too[1]. In essence, this results in a *loosely-coupled architecture*, where higher level modules that perform

[1]In D1.2, we have defined these separate output streams of each module, and made clear that they are available to all other modules. In this refined version of the software architecture described in the present deliverable, we provide a single stream that contains all information and can be accessed by any module. This is achieved by having each module taking as input the Kafka topic which is output by the previous module, following a "chained" approach. This approach solves several technical issues and allows fast prototyping, which is the objective of this deliverable.

data analytics can consume the output of other modules that perform data extraction or integration, in order to optimize their operation in real-time. Also, data analytics modules may also interact with each other; for instance, the *Visual Analytics* module visualizes the events detected or predicted by the *Event Recognition and Forecasting* module in order to perform visual analytics, and the *Event Recognition and Forecasting* module takes as input the trajectories detected or predicted by *Trajectory Detection and Prediction* to identify complex events related to trajectories.

2.2.4 Outputs

When considering outputs of the datAcron architecture to the end-user, these consist of detected and forecast events, data analytics results (detected and predicted trajectories and events, exploratory visual analytics, etc.) initiated by a user that performs a specific task, and results to queries over the integrated data provided by the *Data Manager*.

2.2.5 Key Issues and Benefits

The key issues for the datAcron architecture are as follows:

- The data synopses computed near to the sources aim to largely reduce at a high compression rate the streaming data that the data management and analytics layers have to manage. However, access to the raw streaming data is still an option for the analytics modules, in case a module requires this explicitly.
- The data synopses computed from multiple streams can already be integrated at the lower processing modules (near to the sources). Data synopses and archival data are transformed into a common form according to the dataAcron RDFS schema, are integrated (where necessary) and are pipelined to the rest of the analytics modules directly, in real-time. This alleviates the need for analytics modules to access the datAcron store frequently.
- "Raw" streaming data are not stored as they enter the system: Persistent storage concerns data synopses, semantically annotated and integrated to archival data, trajectories and events detected. The datAcron store will provide advanced query answering services for other system modules and human or software clients to access these data, according to their requirements on integrated data views.

The above architecture has certain benefits:

- All data from streaming and archival data sources, as well as trajectories and events computed by analytics modules can be semantically integrated by discovering links between respective instances, providing semantically-rich coherent views of data. Doing so, datAcron seamlessly annotates trajectories and events with semantic information, and it links these among themselves as well as with the rest of archival and cross-streaming data.
- All analytics modules can take full benefit of the computations of others, also taking advantage of interlinking between their results. Thus, the trajectory detection and forecasting methods can benefit from events detected or forecast and vise-versa. Similarly for the visual analytics methods.
- Users can interact and explore data via integrated data views, being supported for decision-making.

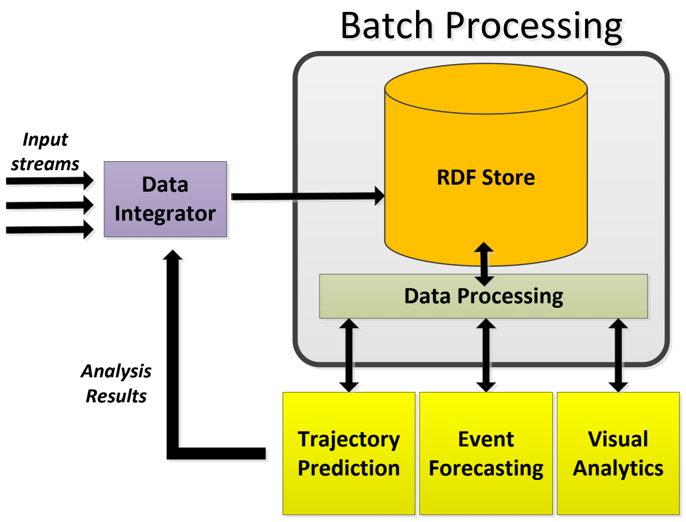

Figure 2: Batch processing.

2.3 Design of Batch Layer

Batch processing (Figure 2) refers to the *Data Manager* module, which provides access to integrated data, both historical and fresh data. However, there is an unavoidable gap between real-time data and fresh data, due to sheer volume of data handled by the *Data Manager* and the induced latency for integrating the streaming data with the historical data. As such, it is expected that *Data Manager* provides access to data up to a certain time point in the near past, whereas the most recent data is handled in a streaming fashion. Eventually, this streaming data will update the data stored in *Data Manager*.

On top of the *Data Manager* module, other datAcron modules operate and perform long-running analysis tasks. For example: the *Trajectory Detection and Prediction* module performs processing-intensive analytics tasks offline, such as clustering, classification, or others; the *Event Recognition and Forecasting* module analyzes historical data to identify patterns of events; the *Visual Analytics* module retrieves data in order to facilitate interactive analysis, data exploration and visualization tasks. All afore-described analysis tasks translate to batch processing over subsets of the historical data available in datAcron, and do not necessarily require the most recent data that arrived in the system.

At the time of this writing, the state-of-the-art solution for batch processing is Apache Spark [10, 11]. This framework is adopted in datAcron in order to develop the batch processing functionality. In the following, it is described how Spark is extended, in order to accommodate the needs and peculiarities associated with processing and analyzing trajectory data represented in RDF. For more details, we refer to Deliverable D1.5 “Data storage and querying (interim)” which describes the batch layer in detail, and is also submitted at the same time with this deliverable (on M18).

2.4 Design of Real-time Layer

In this section, we report the general guidelines with respect to the design of a datAcron module operating in real-time (stream processing) in order to be easily plugged in the software architecture.

In terms of software design, the main directive followed in the datAcron real-time layer is that every real-time module must interact with other real-time modules in the architecture using Kafka. This approach has some advantages:

- it allows modular design of the individual modules;
- it does not demand a common implementation framework or programming language;
- it makes integration easy, since the architecture is loosely-coupled, in contrast to tight coupling that would impose the use of specific APIs;
- it enables the use of different serialization techniques by different modules when producing output (for some tasks binary formats may be more suitable, e.g., Avro, Parquet, while in other cases text in the form of JSON might be a better option).

As described in Deliverable D1.2 "Architecture Specification" and analyzed also below, the real-time layer in datAcron (also illustrated in Figure 3 with respect to architectural modules) requires two main functionalities: (a) stream processing, and (b) stream-based communication.

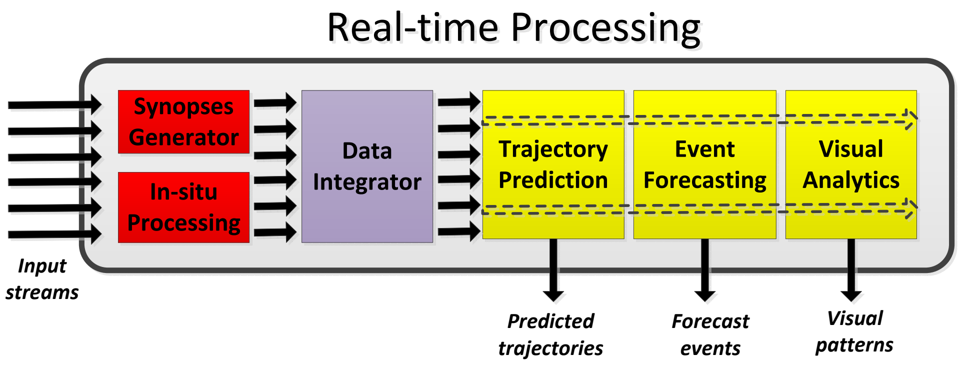

Figure 3: Real-time processing.

2.4.1 Stream Processing

Multiple operations in datAcron are performed on streaming data: the generation of trajectory synopses, data integration, event detection (low-level events and complex events), trajectory prediction, event forecasting, and interactive visual analytics. The majority of these operations must be performed in near real-time, respecting the requirements for *operational latency*.

Taking into account the requirements for stream processing of the datAcron modules as well as initial experimental tests, Apache Flink is going to be used primarily for implementing stream processing functionality. We refer to Deliverable D1.3 "Cross-streaming data processing (interim)", which reports experimental results from a comparison between stream processing frameworks that support the choice of Flink. However, this does not exclude the usage of other stream processing frameworks (e.g., Kafka streams) for specific modules of datAcron, if such a need arises.

2.4.2 Stream-based Communication

To support different stream-based functionality, modules in datAcron need an interconnection infrastructure, since typically the operation of one module may rely on the output of another module. In order to support flexible interconnection and communication between modules without imposing a rigid architecture, we opt for a *loosely-coupled architecture* for datAcron, which has the additional advantage that different modules can be developed using different technologies.

In particular, all outputs of modules are streamed out, thus allowing any other module to connect to any output stream(s) and have access to its content. In technical terms, datAcron adopts the use of Apache Kafka as messaging system to implement the interconnection infrastructure necessary to support our loosely-coupled architecture.

3 The datAcron Integrated Prototype

The purpose of this section is to refine the integrated software architecture of datAcron and clarify the details of each module / flow as of M18. Therefore, in Section 3.1, we present details of participating modules, and, in Section 3.2, we discuss the respective flows materialized by these modules. In Section 3.3, we present the implementation status as of M18, essentially reporting of the fast prototype to be delivered and demonstrated. Section 3.4 summarizes the results presented regarding the datAcron prototype.

3.1 datAcron Modules

The envisaged datAcron architecture consists of the following modules listed in Table 2.

Module Nr.	Module acronym	Module Title	Task in Charge	Work package
1)	Maritime	Maritime raw data feeder	T5.2	WP5
2)	Aviation	Aviation raw data feeders	T6.2	WP6
3)	LED	In-situ processing 1 – Low-level event detector	T1.3.1	WP1
4)	SG	In-situ processing 2 – Synopses generator	T2.1	WP2
5)	SI	Semantic integrator	T1.3.2	WP1
6)	DM	Data manager	T1.3.3	WP1
7)	T/FLP	Trajectory / Future location predictor	T2.2	WP2
8)	TDA	Trajectory data analytics	T2.3	WP2
9)	CER/F	Complex event recognition / forecasting (CER/F)	T3.1-2-3	WP3
10)	IVA	Interactive visual analytics	T4.3	WP4
11)	Viz	Real-time visualization	T4.4	WP4

Table 2: The modules in the datAcron integrated prototype.

A short description as well as the implementation status (as of M18) of each module follows.

- **Maritime raw data feeder (Maritime)**: The data feed is a decimated stream that comes from a range of terrestrial AIS receivers and 18 satellites in a low earth orbit. The maritime AIS data stream is collected, tested for veracity using a streaming analytics module and then filtered to provide the data required for the datAcron project. The AIS data stream is then converted, in real time, from an IEC 61162-1 data stream to a JSON format data stream to allow it to be ingested into the remainder of the datAcron system.
- **Aviation raw data feeder (Aviation)**: This module comprises of a set of 6 European Surveillance data feeds. Namely:

1. FlightAware real time surveillance feed: This module sends a stream of real time surveillance from flightaware global live feed data in plain text to the stdout so it can be piped to any consumer. Only one connection is allowed for datAcron project.
2. FlightAware replay surveillance feed, online mode: This module sends a stream of data for a given period in the past. It requires internet connection and uses flight aware services. The data streamed starts at the beginning of the period and last till all messages are delivered to the end of the period (a few days are maximum period span allowed).
3. FlightAware replay surveillance feed, offline mode: This module sends a stream of data reading a local json file previously stored from the real time surveillance feed. It does NOT require internet.
4. ADSBExhcange real time surveillance feed: This module sends a stream of real time surveillance from ADSBExchange global live feed data in plain text to the stdout so it can be piped to any consumer.
5. ADSBExhcange replay surveillance feed, offline mode: This module sends a stream of data reading a local file previously stored from the real time ADSBExchange surveillance feed. It does NOT require internet.
6. ADSBHub replay surveillance feed, offline mode: This module sends a stream of data reading a local file previously stored from the real time ADSBHub surveillance feed. It does NOT require internet.

- **In-situ processing 1 – Low-level event detector (LED)**: In-situ processing in general refers to the ability to process data streams in-situ – as close to the source where the data originates. This is in particular challenging, when the stream processing of the data requires additional input from other sources, either other instances of the stream or from global system settings or user interaction. As a proof of achieving the architectural integration of in-situ processing, datAcron will apply the event forecasting techniques in-situ on single streams, providing an enriched event stream for visualization in real-time. Based on this, event forecasting will be extended towards distributed online learning, making it possible to learn forecasting models cross-stream from other moving objects.

- **In-situ processing 2 - Synopses Generator (SG)**: The Synopses Generator consumes streaming positions of raw surveillance data and eliminates any inherent noise such as delayed or duplicate messages. Moreover, it identifies critical points along each trajectory, such as stop, turn, or speed change, in order to provide an approximate, lightweight synopsis per moving object.

- **Semantic integrator (SI)**: Its functionality is to (a) transform data from all sources to RDF and (b) discover links between different sources, output this as a stream and also send for storage.

- **Data manager (DM)**: Its functionality is to store information into a distributed spatio-temporal RDF store and provide query answering facility.

- **Trajectory / Future location predictor (T/FLP)**: FLP calculates motion functions by harvesting the cleansed KAFKA stream (from the Synopses Generator module) consisting of the most recent locations from a moving object to predict its short-term future location in real time by taking into consideration the tendency of the movement. Each predicted point will be streamed out in real time to other modules. Regarding TP, it will present a similar functionality targeting at predicting the future trajectory of a moving object as far in time horizon an possible.

- **Trajectory data analytics (TDA)**: The goal of this module is twofold: on the one hand it provides advanced analytics that are going to serve specialized requirements in the datAcron architecture (e.g. data-driven discovery of the networks/routes upon which the movement of the vessels/aircrafts take place), while on the other hand it provides global patterns that represent meta models devised from the local patterns (e.g. clusters and sequential patterns of semantic trajectories).
- **Complex event recognition/forecasting (CER/F)**: CER is about real-time detection of complex events, whereas CEF is about real-time forecasting of complex events. Both modules are working on the synopsis of the moving object generated by the two in-situ processing modules (LED & SG) and, in addition, can take additional information achieved via the enrichment and linking performed by SI. The output of CER is a real-time stream with detected events. On the other hand, CEF enriches the input stream with a forecast about the probability of each monitored pattern. As event forecasting requires learning of CER probabilities, it is more restrictive with respect to the potentially supported patterns than the pure event detection.
- **Interactive visual analytics (IVA)**: The IVA module builds on top of the real-time visualization module to provide limited analytical capacity on streaming data. The primary use is to allow analysts, and possibly advanced operators, to fine-tune and observe impact of parameter adjustments to the T/FLP and CER/F modules compared to actual data in (near) real-time. It therefore complements the situation monitoring capabilities of the real-time visualization used by ordinary operators on the one hand (by providing parameter settings to the detection modules), and the full-fledged VA suite used for in-depth exploration and analysis in offline (strategic latency) settings.
- **Real-time visualization (Viz)**: This module provides a map-based visualization of the stream of enriched spatio-temporal events generated by the T/FLP and CER/F modules. It is able to display different event types (e.g., critical points) simultaneously with individual visual encoding for each type. In addition events associated with the same moving object identifier are automatically integrated into trajectory representations so operators can observe movement patterns. The overall design follows the "overview-first, zoom-and-filter, details-on-demand" approach, meaning that operators can define filters on the input stream to drill down on areas and event types of interest.

3.2 datAcron Flows

Having the modules presented above integrated, a number of information flows of three different types are envisaged. In particular:

- *Information management* flows are about the reconstruction of trajectories and their enrichment with useful annotation, which is to be performed online (operational latency), and their storage for querying purposes, which is to be performed offline (tactical latency).
- *Online analytics* flows are about consuming the available streaming information, which is to be performed online (operational latency); and

- *Offline analytics* flows are about consuming the available stored information, which is to be performed offline (strategic latency).

Note that there exist three main consumers (namely, T/FLP, CER/F, and IVA), therefore, 3+3 flows are envisaged, for the online and offline analytics, respectively. Table 3 presents the list of flows (along with the respective latency type and partners in charge of coordinating their implementation).

Flow Nr.	**Flow Title**	**Latency**
Information management flows		
1)	Trajectory reconstruction and semantic enrichment	Operational
2)	RDF storage	Tactical
Online analytics flows		
3)	Trajectory/FL prediction online	Operational
4)	Complex event recognition / forecasting online	Operational
5)	Visual Analytics online	Tactical
Offline analytics flows		
6)	Trajectory data analytics offline (*)	Strategic
7)	Complex event recognition / forecasting offline (*)	Strategic
8)	Visual Analytics offline	Strategic

Table 3: The datAcron flows of information (note: flows marked with * are not planned to be implemented until M18).

The functionality of each flow is discussed in the following sections. In accordance with the flows, Figure 4 illustrates the datAcron architecture, which is a refined version of the architecture specified in Deliverable D1.2 "Architecture Specification".

3.2.1 Flow #1: Trajectory reconstruction & semantic enrichment (operational latency)

- Short description: Maritime / Aviation raw data stream is (1a) cleansed, enriched with derived information (e.g. speed) as well as low-level events (e.g. intersection with zones of interest), synopsized by tagging "critical points" (change of heading or altitude, etc.), and (1b) further enriched with info from other external data sources / streams (weather info, etc.); the final output (1c) is streamed out to be consumed by other modules, including its visualization (1d).
- Modules involved: Maritime / Aviation; LED; SG; SI; Viz.

3.2.2 Flow #2: RDF storage (tactical latency)

- Short description: A subset of the enhanced surveillance data stream, i.e. the annotated surveillance data, as well as selected output streamed out by other modules is (2a) processed and (2b) stored in the RDF store.
- Modules involved: DM.

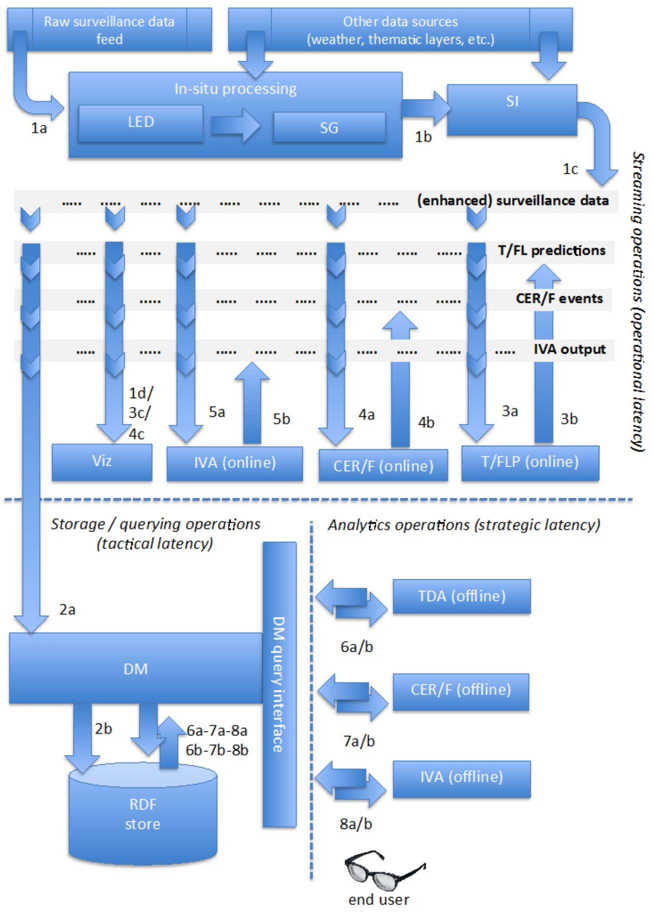

Figure 4: The refined datAcron architecture.

3.2.3 Flow #3: Trajectory/FL prediction online (operational latency)

- Short description: T/FLP (3a) consumes the enhanced surveillance data stream (as well as other streams, if needed) for the purposes of online trajectory / future location prediction and (3b) streams out its output to be consumed by other modules, including its visualization (3c).
- Modules involved: T/FLP; Viz.

3.2.4 Flow #4: Complex event recognition / forecasting online (operational latency)

- Short description: CER/F (4a) consumes the enhanced surveillance data stream (as well as other streams, if needed) for the purposes of online event recognition / forecasting and (4b) streams out its output to be consumed by other modules, including its visualization (4c).

- Modules involved: CER/F; Viz.

3.2.5 Flow #5: Interactive visual analytics online (operational latency)

- Short description: IVA consumes the enhanced surveillance data streams (3c, 4c), streamed meta data on the T/FLP and CER/F modules (current parameter settings, 5a), and, if needed, base data for comparison (1d) for the purposes of online VA; and (5b) streams out its output (updated parameter settings, areas-of-interest) in KVP format to be consumed by other modules.
- Modules involved: IVA; T/FLP; CER/F.

3.2.6 Flow #6: Trajectory data analytics offline (strategic latency)

- Short description: TDA (6a) queries the RDF store in order for complex patterns to be discovered and (6b) stores selected results back to the RDF store for future use.
- Modules involved: TDA; DM.

3.2.7 Flow #7: Complex event recognition / forecasting offline (strategic latency)

- Short description: CER/F (7a) queries the RDF store in order for complex events to be detected/forecasted and (7b) stores selected results back to the RDF store for future use.
- Modules involved: CER/F; DM.

3.2.8 Flow #8: Interactive visual analytics offline (strategic latency)

- Short description: IVA (8a) queries the RDF store in order for large batches of raw data for complex offline analysis and (8b) stores selected results (derived attributes, spatio-temporal patterns, clustering results, parameter settings) back to the RDF store for future use.
- Modules involved: IVA; DM.

3.3 Implementation Status

At the time this deliverable is submitted, we are finalizing the integration of the first five flows (1–5). In this way, we are able to demonstrate the fast prototyping, by providing an integrated prototype where all technical workpackages interact and provide a common functionality.

Figure 5 shows in more technical detail the information flows 1 and 2, which is are two of the most critical information flows in datAcron, as they provide enriched surveillance data for further online analytics tasks (flow 1), and generate integrated data for storage and offline analysis (flow 2).

3.3.1 Flow #1

In this flow, the prototype consumes as input raw surveillance data, both for maritime and aviation surveillance sources. This input is provided as a Kafka stream. In-situ processing modules operate on this stream.

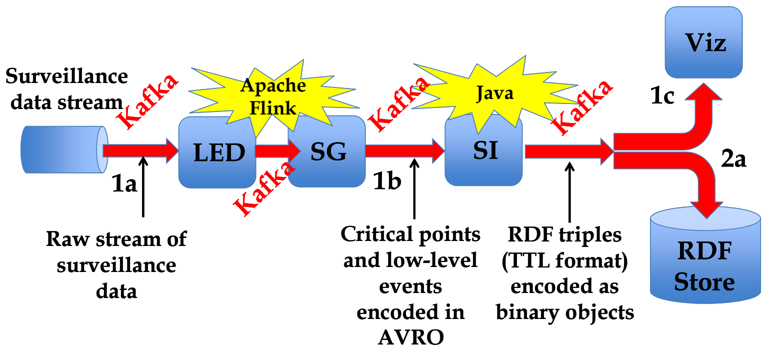
Surveillance data stream
Kafka
Apache Flink
LED
SG
Kafka
Kafka
Java
SI
Kafka
Viz
1c
2a
RDF Store
1a
Raw stream of surveillance data
1b
Critical points and low-level events encoded in AVRO
RDF triples (TTL format) encoded as binary objects

Figure 5: Flows #1 and #2 in the implemented prototype.

- LED aims to provide a Flink component that processes a stream of raw messages (i.e., AIS dynamic messages) and enrich it with derived attributes such as min/max, average and variance of original fields. As such, LED accesses the raw stream containing positions of moving objects (vessels or aircrafts) and identifies low-level events, such as entry/exit to areas of interest. In addition, a stream simulator for the raw messages is developed in the context of this module, which provides a functionality to replay the original stream of raw messages by generating a simulated new Kafka Stream and taking into account the time delay between two consecutive messages of a trajectory, furthermore, this delay can be scaled in/out by a configuration parameter. In more detail, LED computes the following attributes on per trajectory basis:
  - Min/Max, average and median of speed
  - Min/Max, average and median of acceleration
  - Min/Max, median of time difference between successive points
  - Min/Max duration, dates and coordinates

  The output is provided as a Kafka stream that contains the original information as well as the afore-described fields. For the maritime use-case, the output line header is:

  ```
  id,status,turn,speed,course,heading,longitude,latitude,timestamp,
  AverageDiffTime,NumberOfPoints,LastDiffTime,MinSpeed,MinDiffTime,
  MaxSpeed,MaxDiffTime, MinLong,MaxLong,MinLat,MaxLat,LastDifftime,
  AverageSpeed,VarianceSpeed
  ```

  Table 4 explains the output description; the same order of attributes is used as in AIS messages of NARI source with additional attributes computed by this module. The detailed description of the operation of LED is provided as a separate deliverable D1.3 “Cross-streaming data processing (interim)” on M18.

- SG accesses the output of LED and performs two major operations. First, it performs data cleansing, thus eliminating noisy data. Second, it identifies “critical points” on per trajectory basis. Essentially, SG tags the most significant positions that contain information that can accurately described the trajectory with information describing each critical points (e.g., “turn”, “gap_start”, “gap_end”, etc.). The output is provided in Avro[2] format

[2]https://avro.apache.org/

Attribute	Data type	Description
id	integer	A globally unique identifier for the moving object (usually, the MMSI of vessels).
status	integer	Navigational status
turn	double	Rate of turn, right or left, 0 to 720 degrees per minute
speed	double	Speed over ground in knotsint (allowed values: 0-102.2 knots)
course	double	Course over ground (allowed values: 0-359.9 degrees)
heading	integer	True heading in degrees (0-359), relative to true north
longitude	double	Longitude (georeference: WGS 1984)
latitude	double	Latitude (georeference: WGS 1984)
timestamp	long	timestamp in UNIX epochs (i.e., milliseconds elapsed since 1970-01-01 00:00:00.000)
AverageDiffTime	long	The average of difference time between the positions message of a trajectory
NumberOfPoints	int	The accumulated number of the received points
LastDiffTime	double	The time difference of the current message and the last previous received message
MinSpeed	double	The minimum value of speed until current message
MinDiffTime	long	The minimum value of time difference until current message
MaxSpeed	double	The maximum value of speed until current message
MaxDiffTime	double	The maximum value of time difference until current message
MinLong	double	The minimum value of longitude until current message
MaxLong	double	The maximum value of longitude until current message
MinLat	double	The minimum value of latitude until current message
MaxLat	double	The maximum value of latitude until current message
AverageSpeed	double	The average of the speed
VarianceSpeed	double	The variance of speed

Table 4: LED output description.

as a Kafka stream. The detailed description of the operation of SG is provided as a separate deliverable D2.1 “Cross-streaming, real-time detection of moving object trajectories (interim)” on M18.

- SI receives the Kafka stream produced by SG and performs transformation to RDF as well as data integration, by enriching positions with information about weather as well as other contextual information. The output is provided in RDF, encoded in Terse RDF Triple Language (TTL)[3] and serialized in binary format as Java objects, also provided as a Kafka stream. The detailed description of the operation of SI is provided as a separate deliverable D1.4 “Data Integration, Management (interim)” on M18.
- Viz receives this enriched stream of information and provides real-time visualizations that can be used by operational users for improved situational monitoring and awareness.

3.3.2 Flow #2

This flow is a continuation of Flow #1, namely it receives the RDF data provided in a Kafka stream and stores it in the distributed RDF store. For this purpose, the RDF data is encoded using a spatio-temporal RDF encoding scheme introduced in datAcron that enables efficient storage and retrieval of spatio-temporal RDF data. In addition, we provide querying facilities on top of the distributed RDF store. We have developed batch processing functionality using Apache Spark and extending it towards querying spatio-temporal RDF data. In this way, we can retrieve portions of the integrated data based on different filtering criteria, and the produced data sets can be used for more advanced analysis tasks (e.g., trajectory clustering, pattern discovery, building forecasting models, etc.).

In M18, we have implemented a small set of queries on maritime data to demonstrate this functionality, and we shall extend this set in the subsequent months to provide richer functionality. The detailed description of the operation of the DM module which contains the distributed

[3] https://www.w3.org/TR/turtle/

RDF store is provided as a separate deliverable D1.5 “Data storage and querying (interim)” on M18.

3.3.3 Flow #3

This flow addresses the trajectory and future location prediction in an online fashion.

Sub-flow #3a The sub-flow #3a in the datAcron architectural diagram is actually the input for module T/FLP. Normally, this involves data from full-resolution data, synopses and semantics integrated into the corresponding sub-flows #1x and then published in the “enhanced surveillance data stream” (shared).

The current implementation of algorithms in module T/FLP rely on streaming simulation mode, typically in various packets of customized CSV files. This is a shortcut, until the full integration is completed and the corresponding modules are available online as input feeders (see Flow #1).

In full integration mode, this combined information may also be available by combining the intermediate outputs of the modules in Flow #1, e.g. by module SI (typically a Kafka stream with composite RDF records).

There is also extended logging in module SG, which may be useful for some future location prediction algorithms in module T/FLP, especially the exact processing and labeling of critical points (e.g. turns / “change of heading”). Some of these are already been used now in designing and optimizing the current implementations in T/FLP. However, no specific requirement or specification is proposed for now, regarding possible inclusion in sub-flow #3a, until these algorithms are finalized and in fully operational mode.

As of M18, T/FLP operates on the following set of attributes, in order to perform its task:

- id
- ts (timestamp)
- longitude
- latitude
- altitude
- annotation
- speed
- heading

Sub-flows #3b/#3c The sub-flows #3b/#3c in the datAcron architectural diagram are actually the outputs from module T/FLP. Normally, this includes forecasts of variables from T/FL predictors and they are published in the “enhanced surveillance data stream” (shared).

In the current implementation, the prediction results from T/FLP are stored as JSON Arrays in HDFS, in order to evaluate the proposed algorithm, enrich synopses results (fill communication gaps) and give extended input in clustering or statistical procedures. In next steps we are going to store the prediction results in byte array format (Serialize Arvo Schema with Avro Tools and produce byte array) for more efficient storage and retrieval, in order to be used by algorithms in the TDA module (see: Sub-flow #6b).

According to the current implementations in module T/FLP, there is a proposed schema that can be used as template for the final integration with modules VA, SI and the shared data stream. The stream specification is Kafka in JSON format and its current design is as follows (to be revised):

```
Sub-flows #3b,#3c: Kafka/ Avro record (provisional)

"name": "PredictionArray",
"type":"record",
"fields":[
   {
       "name" : "point" , "type" : {
       "type": "array",
       "items":{
        "name":"RMFPoint",
       "type":"record",
             "fields":[
              {"name":"id", "type":"long"},
              {"name":"timestamp", "type":"long"},
                {"name":"longitude", "type":"double"},
                {"name":"latitude", "type":"double"},
                {"name":"altitude", "type":"double"},
                {"name":"speed", "type":"double"},
                {"name":"heading", "type":"double"}
                ]
            }
           }
   },
   {"name": "prediction", "type": "boolean"}
]
```

3.3.4 Flow #4

As of M18, the modules Complex Event Recognition (CER) and Complex Event Forecasting (CEF) are deployed and operate on the maritime AIS data from the Brest area. The Complex Event Recognition has also been tested and operates on aviation data. Both modules consume the data provided by the SG module and either read in the data from file or from a Kafka topic. Outputs are produced as streams on Kafka.

For simulating real-time data as being received from the external world, a "Synopses Stream Simulator" has been developed that does the following:

- ingest the synopses CSV data from Kafka stream or by CSV file source reader,
- process the synopses to reconstruct the trajectories and simulate the original stream by delaying the propagation the synopses based on the time difference between the synopses of a trajectory (time delay can be scaled in/out by a given parameter),
- then, the synopses are published to Kafka Stream in JSON format as shown in the following example:

```
{
```

```
"timestamp": 1451606409000,
"id": "227006770",
"longitude": 0.133466666666667,
"latitude": 49.4754,
"annotation": {
"stop_start": false,
"stop_end": false,
"change_in_speed_start": false,
"change_in_speed_end": false,
"slow_motion_start": false,
"slow_motion_end": false,
"gap_start": false,
"gap_end": true,
"change_in_heading": false,
"noise": false
},
"distance": 0,
"speed": 0,
"heading": 0,
"time_elapsed": 0,
"msg_error_flag": ""
}
```

For testing integration with the real-time visualization (Viz), Viz is reading in this Kafka stream and plotting the trajectory in simulated real-time. In addition, Viz is receiving events streams with forecasts.

Complex Event Forecasting This module currently processes the synopses stream of vessels and attaches predications of predefined patterns (i.e., defined by regex) as illustrated in the following JSON output line of a synopsis after adding the *predictionsMap* by this module, where is the predictions list for each pattern is expressed as *[current relative timestamp, start time of completion interval, end time of completion interval, probability of the pattern]*:

```
{
   "timestamp":1451606409000,
   "id":"244710897",
   "longitude":4.42071333333333,
   "latitude":51.8845133333333,
   "annotation":{
      "stop_start":false,
      "stop_end":false,
      "change_in_speed_start":false,
      "change_in_speed_end":false,
      "slow_motion_start":false,
      "slow_motion_end":false,
      "gap_start":false,
      "gap_end":true,
      "change_in_heading":false,
      "noise":false
   },
   "distance":0.0,
   "speed":0.0,
   "heading":0.0,
   "time_elapsed":0,
   "msg_error_flag":"",
   "predictionsMap":{
      "change_in_heading.gap_start.gap_end.change_in_heading":[
         2.0,
         13.0,
         17.0,
         0.65
      ],
      "change_in_heading.gap_start.(gap_end|change_in_heading)":[
         2.0,
         9.0,
         14.0,
         0.70
      ]
   }
}
```

The forecasting module is implemented in Flink with Java 8, reading from Kafka, sending to Kafka and is currently tested to forecast a patterns.

Complex Event Recognition Regarding event recognition, several patterns have been implemented and tested. For the maritime use case:

- MSI#08: Speed not compatible with area
- MSI#19: Under way (using engine or sailing)
- MSI#26: Loitering
- MSI#28: Rendez-vous

For the flight planning use case:

- top-Of-Climb
- top-Of-Descent
- deviation from flight plan

3.3.5 Flow #5

This flow is about interactive visual analytics online. Although SG and TFL/P are not explicitly associated with Flow #5, they are included here for completeness, as they are actually what feeds into a "stream combinator", which fuses streams #1d, #3c and #4c into the single enhanced surveillance data stream that (for the current implementation) serves as the only input to the visualization module Viz.

Viz digests this enriched stream of spatial events that are further and automatically integrated into trajectory objects, displayed jointly as points and lines, respectively, on a (2D) map display.

It should be noted that from the perspective of the visualization, actual position reports (ground truth/historic data, stream #1d), predicted events (#4c), and trajectory synopsis (#3c) are all comprised of (a set of) spatial events. The only difference is what additional event attributes are available for visual mapping – e.g., for the semantic type label for synopsis/critical points, or a flag indicating whether this is an actual, observed datum or a predicted position. The principal format of the input stream, including the thematic event attributes and flags currently supported, has been described in detail in Flow 4 above.

The IVA module builds on top of Viz to provide limited analytical capacity on streaming data. Therefore, the principal input to the IVA is identical to that of the Viz – streams #1d, #3c, #4c. The primary use is to allow analysts, and possibly advanced operators, to fine-tune and observe impact of parameter adjustments. This adds flows #5a and #5b to the picture, which represent the input to and output from, respectively, the IVA module to the T/FLP and SG modules.

Sub-flow #5a The role of sub-flow #5a in the datAcron architectural diagram will be to communicate current parameter settings from the computational modules T/FLP and SG to the IVA module.

We have preliminarily identified the following information that could be meaningfully communicated between them, in addition to any semantic information already encoded and communicated through event streams #3c and #4c:

1. Current values of named free parameters of specific detection/prediction algorithms, e.g., the minimum CPA distance threshold for MSI#28 (rendez-vous), for UI display purposes
2. Locations (points) or areas (polygons) of interest, such as protected areas (MSI#02), reference locations (MSI#01, MSI#03), aeronautical waypoints, or ATC sectors, to display this context information on the map display
3. Sets (arrays) of historic attribute values (e.g., vessel speeds) with a defined "window length" (start and end time stamps relative to current real time), to populate "detail on demand" information overlays for selected entities during real-time analysis.
4. Other control commands affecting the computational modules, such as a "reset" command for the in-situ processing module to trigger a corresponding reset of statistical aggregates collected over the stream up to the given moment.

In terms of actual online integration, the networked connection between modules Viz, SG, and T/FLP would utilize the same Kafka with JSON-encoded payload as has already been tested successfully.

```
Sub-flow #5a: Kafka/Avro record (provisional)

"name": "HistoricDataArray",
"type":"record",
"fields":[
{
       "name" : "point" , "type" : {
       "type": "array",
       "items":{
        "name":"RMFPoint",
       "type":"record",
        "fields":[
              {"name":"id", "type":"long"},
              {"name":"timestamp", "type":"long"},
                   {"name":"longitude", "type":"double"},
                   {"name":"latitude", "type":"double"},
                   {"name":"altitude", "type":"double"},
                   {"name":"speed", "type":"double"},
                   {"name":"heading", "type":"double"}
                 ]
            }
           }
     },
{"name": "windowstart", "type": "long"},
{"name": "windowend", "type": "long"},
   {"name": "historic", "type": "boolean"}
]
```

Note the proposed format replays all information that uniquely identifies any spatial event (id, timestamp, geographic coordinates), in addition to any requested attributes. This allows to register detail overlay with recent data that is retained on the map display.

Sub-flow #5b The role of sub-flow #5b in the datAcron architectural diagram is to communicate updated parameter settings from the IVA module to the computational modules T/FLP and SG. Data format and technical realization would be equivalent to those described above for sub-flow #5a, with the exception that only stream content (a) and (b) – i.e., user-adjusted free parameter settings, user-defined points or areas of interest – are meaningful.

Implementation Details of Flow #5 The implementation of the real-time visualization/interactive VA module follows the client-server architecture (see D4.4.1, Section 3). On the server side the Kafka streams are consumed. All the applications on the server side are written in Java. The client side is browser-based and written in JavaScript.

In the current version of the implementation, neither stream #5a nor stream #5b are supported with the exception of a proof-of-concept implementation of a "reset" command that restarts the aggregate collected over the full event stream as observed by the in-situ module; further extensions of the real-time visualization module into the IVA module, including UI additions to display and set, respectively, the contents of streams #5a and #5b, is scheduled for beyond M18.

3.4 Summary

This section presented datAcron modules and the design of respective flows involving these modules. In essence, this defines the software design of the integrated prototype. In addition, we presented the status of the "fast prototype" that was developed in M18, and showed that all datAcron technical work packages provide software modules in an integrated prototype that demonstrates basic and critical functionality for the project and its objectives.

4 Conclusions & Outlook

In this deliverable, we report the software design on M18 of the project, which is subject to revisions and optimizations in the second part of the project. The main point is that an early integrated prototype has been developed, thus achieving inter-communication between the different solutions implemented by the technical work packages.

This prototype demonstrates its functionality and operation in real-time processing and online scenarios, most notably with low-level event detection, generation of trajectory synopses, semantic enrichment of positional data with contextual and weather data, as well as with higher data analysis tasks, including future location prediction, complex event recognition and forecasting and real-time visualizations. Also, the prototype is able to demonstrate batch processing of a limited set of queries over integrated RDF data, using Spark as a Big Data platform for development.

In the second iteration, the plan is to improve this prototype and extend its functionality in different ways:

- extend the functionality of individual modules, in order to optimize their internal operation,
- evaluate and if necessary refine the implementation to achieve the required latency constraints in real-time processing,
- provide a richer suite of queries over the distributed RDF store, thereby enabling retrieval of enriched data sets using a wide variety of filtering criteria, which can be used for data analysis tasks,
- develop and integrate with WP2–WP4 the offline analytics, described as Flows #6–#8.

Overall, the software design of the integrated datAcron prototype proceeds according to plan, and in this report we already presented integrated functionality, which is planned to be demonstrated in the first review of datAcron.

References

[1] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Katherine J. Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 411–422, 2007.

[2] Divyakant Agrawal, Philip Bernstein, Elisa Bertino, Susan Davidson, Umeshwar Dayal, Michael Franklin, Johannes Gehrke, Laura Haas, Alon Halevy, Jiawei Han, H.V. Jagadish, Alexandros Labrinidis, Sam Madden, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, Kenneth Ross, Cyrus Shahabi, Dan Suciu, Shiv Vaithyanathan, and Jennifer Widom. Challenges and opportunities with big data – a community white paper developed by leading researchers across the united states. Technical report, March 2012.

[3] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, 2014.

[4] Thomas Neumann and Gerhard Weikum. The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113, 2010.

[5] Alexander Schätzle, Martin Przyjaciel-Zablocki, Simon Skilevic, and Georg Lausen. S2RDF: RDF querying with SPARQL on Spark. *PVLDB*, 9(10):804–815, 2016.

[6] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds. Efficient RDF storage and retrieval in Jena2. In *Proceedings of the First International Workshop on Semantic Web and Databases (SWDB)*, pages 131–150, 2003.

[7] Simin You, Jianting Zhang, and Le Gruenwald. Spatial join query processing in cloud: Analyzing design choices and performance comparisons. In *Proceedings of the 44th International Conference on Parallel Processing Workshops (ICPPW)*, pages 90–97, 2015.

[8] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. GeoSpark: a cluster computing framework for processing large-scale spatial data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 70:1–70:4, 2015.

[9] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. A demonstration of GeoSpark: A cluster computing framework for processing big spatial data. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 1410–1413, 2016.

[10] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing*, 2010.

[11] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark: a unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.

A The datAcron Cluster

The datAcron infrastructure consists of a cluster of 10 physical nodes with the following specifications:

- CPU Intel Xeon E5, 6 cores, 1.6GHz
- 128GB DDR4 RAM
- 6TB HDD + 200GB SSD
- 1Gbit connection to the outside world

Node 1 is configured as NameNode and Resource Manager, while Nodes 2 – 10 correspond to DataNodes.

In terms of software versions, we have installed in all machines:

- Ubuntu: 16.04.2 (kernel 4.4.0) x64
- Java: 1.8.0_121
- Hadoop, HDFS, YARN: 2.7.2
- Spark: 2.1.1 (with Scala 2.11.8)
- Redis: 3.2
- Scala: 2.11.7
- Flink: 1.2
- Confluent: 3.1.1
- Vagrant: 1.9.1
- Python: 2.7.12
- Ansible: 2.3.0

We also provide more details with respect to HDFS and YARN (a container is either a map or a reduce task):

- HDFS
  - Total available space: 48.35 TB (5.37TB on each node)
  - Replication factor: 3
  - Total effective space: 16.12 TB
- YARN
  - Total available vCores: 90 (10 on each node)
  - Total available ram: 360GB (60GB on each node)
  - Minimum container ram size: 1GB
  - Maximum container ram size: 10GB

The above specifications describe the capabilities of our infrastructure in terms of resources, more specifically storage, memory, networking, and processing.