Grant Agreement No: 687591

24/9/18

Big Data Analytics for Time Critical Mobility Forecasting

datAcron

D1.12 Integrated prototype (final)

Deliverable Form	
Project Reference No.	H2020-ICT-2015 687591
Deliverable No.	1.12
Relevant Work Package:	WP 1
Nature:	D
Dissemination Level:	PU
Document version:	1.0
Due Date:	30/09/2018
Date of latest revision:	29/09/2018
Completion Date:	29/09/2018
Lead partner:	UPRC
Authors:	Christos Doulkeridis (UPRC), Apostolos Glenis (UPRC), Panagiotis Nikitopoulos (UPRC), Giorgos Santipantakis (UPRC), Akrivi Vlachou (UPRC), George Vouros (UPRC), Nikos Pelekis (UPRC), Harris Georgiou (UPRC), Kostas Patroumpas (UPRC), Georg Fuchs (FRHF)
Reviewers:	Yannis Theodoridis (UPRC), Michael Mock (FRHF), Elias Alevizos (NCSR'D)
Document description:	This deliverable briefly describes the datAcron integrated prototype.
Document location:	WP1/Deliverables/D1.12/Final

HISTORY OF CHANGES

Version	Date	Changes	Author	Remarks
0.1	1/9/2018	First version of report	C. Doulkeridis	
0.2	17/9/2018	Restructured version	C. Doulkeridis	
0.3	20/9/2018	Added material from WP4	F. Patterson, C. Doulkeridis	
0.4	21/9/2018	Added material from WP2	P. Tampakis, N. Pelekis, C. Doulkeridis	
1.0	24/9/2018	Version for internal review	C.Doulkeridis, A.Glenis, P.Nikitopoulos, G.Santipantakis, A.Vlachou, G.Vouros	Version shared with all partners

EXECUTIVE SUMMARY

This report is deliverable D1.12 of datAcron work package 1 "System architecture and data management" with main objective to describe the datAcron integrated prototype system, in accordance with the requirements specified in deliverable D1.1, the architecture specified in deliverable D1.2, and the software design described in D1.11.

Essentially, D1.12 provides evidence on the functionality supported by the integrated system using examples from both domains. As D1.12 is intended as demonstration in the context of the project, this document is the accompanying write-up that aims to present the functionality of the integrated system, starting from the visualization components and then delving into the details of online components that feed the visualizations.

TABLE OF CONTENTS

TERMS & ABBREVIATIONS

ADS-B	Automatic Dependent Surveillance-Broadcast
AIS	Automatic Identification System
RDF	Resource Description Framework
LED	Low-level Event Detector
SG	Synopsis Generator
SI	Semantic Integrator
DM	Data Manager
T/FLP	Trajectory/Future Location Predictor
CER/F	Complex Event Recognition/Forecasting
TDA	Trajectory Data Analytics
IVA	Interactive Visual Analytics
Viz	Real-time Visualization

LIST OF FIGURES

LIST OF TABLES

1 Introduction

This document is the deliverable D1.12 “Integrated prototype (final)” of Task T1.4 of work package 1 ”System Architecture and Data Management” of the datAcron project. D1.12 is a demonstration deliverable submitted on M33 of the project, therefore this document serves as the accompanying report that aims at providing a brief overview of the datAcron integrated prototype.

Recall that the integrated prototype is the result of two iterations during the project: in the first iteration of software design, the aim was at providing an early prototype and integrating software modules towards the second iteration; in the second design iteration, the focus was on the integration with WP2–WP4.

1.1 Purpose and Scope

The “Integrated prototype (final)” aims at demonstrating the datAcron integrated prototype which encompasses the complete functionality provided in the context of the project.

In particular, the prototype provides integrated functionality, by implementing interconnection between all datAcron modules developed as results of different work packages. In more detail, the prototype offers the following functionality:

- Real-time consumption of raw surveillance data streams, on which the following operations are performed: in-situ processing (including low-level event detection and synopsis generation), semantic integration with weather and contextual data sources, future location prediction, complex event detection and forecasting, and real-time visualization.
- Querying integrated RDF data using SPARQL with spatio-temporal constraints, demonstrating the feasibility of scalable query processing.
- Data analytics on integrated data from different sources, aiming at discovering complex patterns that would not be feasible to detect by analyzing the individual data sources.

The integrated prototype is built based on a Big Data architecture (cf. Deliverable D1.2 “Architecture Specification”), which comprises online functionality on streaming data (data in motion), as well as offline functionality on archival data (data at rest).

The individual modules are designed and implemented using state-of-the-art frameworks for Big Data processing, most notably Apache Flink and Apache Spark. In this way, scalability and fault-tolerance is achieved at the level of each individual module. The intercommunication of the different online modules is achieved using Apache Kafka, the de-facto standard for building real-time data pipelines nowadays, offering scalability, persistence, replication, and fault-tolerance.

1.2 Approach for the Work package and Relation to other Deliverables

Similarly to D1.11, D1.12 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.

It should be pointed out that D1.12 is prepared after deliverables D1.8 “Cross-streaming data processing (final)”, D1.9 “Data Integration, Management (final)”, D1.10 “Data storage and querying (final)”, and D1.11 “Software design (final)” are prepared. Deliverables D1.8 and D1.9 provide detailed description on real-time layer of the datAcron architecture. Also, D1.10 describes the design and implementation of the batch processing solution developed in the context of datAcron. As such, they connect to this deliverable, and intermediate versions of D1.8, D1.9 and D1.10 have been considered during the preparation of the current deliverable.

Moreover, during the preparation of D1.12, intermediate and final versions of deliverables of the other work packages have been taken into account (as of M33), in order to provide a comprehensive result that reflects the work carried out in the context of the entire project.

1.3 Methodology and Structure of the Deliverable

In terms of work methodology, this deliverable takes as input D1.11 “Software design (final)” and presents the development of an integrated prototype system according to the specification of D1.11. To be self-contained (to the extent possible), the current deliverable repeats information from D1.11, namely related to its design and implementation.

The integrated prototype has been developed 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.

The remaining of this report is structured as follows:

- Section 2 presents the design of the integrated prototype of datAcron, focusing in flows of information and the architecture that encompasses all envisioned flows and modules.
- Section 3 describes implementation aspects on the integrated prototype, by explaining each flow of information in more detail.
- Section 4 presents online visualizations and visual representation and analysis of results produced by offline analytics, in order to illustrate the types of data, events, predictions, etc., that feed the integrated prototype, as well as the modules generating this data.
- Section 5 summarizes the work reported in this deliverable, and provides the plan for the second iteration.

2 Design of datAcron Integrated Prototype

The purpose of this section is to briefly present the datAcron integrated prototype, in terms of (a) modules that implement the functionality (Section 2.1), and (b) flows of information materialized by means of the modules (Section 2.2).

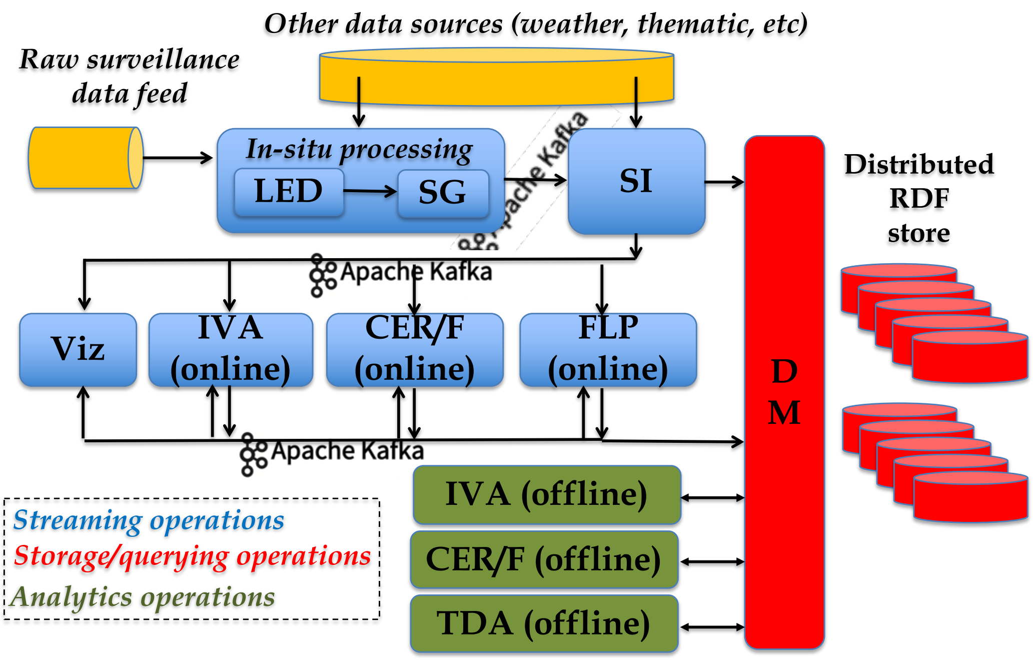


Figure 1: The datAcron Integrated Prototype.

Figure 1 illustrates the architecture presenting datAcron modules and their links (flows of information).

2.1 datAcron Modules

The envisaged datAcron architecture consists of the following modules listed in Table 1.

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 datAcron system.

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 1: The modules in the datAcron integrated prototype.

- **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 has implemented a distributed online learning framework on distributed streams as described in detail in D1.8, allowing for faster adaption of models to changes in real-time and providing an enriched event stream for visualization in real-time. Based on this, event forecasting has been extended towards distributed online learning as described in D3.5, 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 an enriched stream of positional information and also send this stream 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 presents a similar functionality targeting at predicting the future trajectory of a moving object as far in time horizon as possible.
- **Trajectory data analytics (TDA)**: The goal of this module is twofold: on the one hand it provides advanced analytics that 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 event 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.

2.2 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 2 presents the list of flows (along with the respective latency type and partners in charge of coordinating their implementation).

The functionality of each flow is discussed in the following sections. In accordance with the flows, Figure 1 illustrates the datAcron architecture, which is a refined version of the architecture specified in Deliverable D1.2 "Architecture Specification".

2.2.1 Flow #1: Trajectory reconstruction & semantic enrichment

- 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.

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

2.2.2 Flow #2: RDF storage

- Short description: A subset of the enhanced surveillance data stream, i.e. the annotated, synopsized 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.

2.2.3 Flow #3: Trajectory/FL prediction online

- 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). Stores prediction results in the form of Parquet files in HDFS for evaluation purpose.
- Modules involved: T/FLP; Viz.

2.2.4 Flow #4: Complex event recognition / forecasting online

- 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.

2.2.5 Flow #5: Interactive visual analytics online

- 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.

2.2.6 Flow #6: Trajectory data analytics offline

- 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.

2.2.7 Flow #7: Complex event recognition / forecasting offline

- Short description: CER/F (7a) queries the RDF store in order to fetch data 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.

2.2.8 Flow #8: Interactive visual analytics offline

- Short description: IVA (8a) queries the RDF store to get 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 Implementation of datAcron Integrated Prototype

In this section, we provide more technical details on the datAcron flows, thus presenting their implementation and the specific interactions between the individual modules.

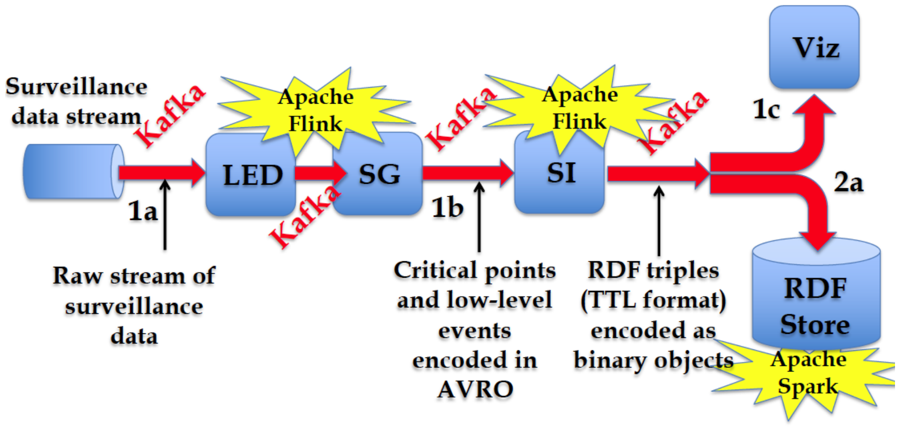

Surveillance data stream
Kafka
Apache Flink
Kafka
Apache Flink
Kafka
Viz
1c
LED
SG
SI
2a
1a
Kafka
1b
Raw stream of surveillance data
Critical points and low-level events encoded in AVRO
RDF triples (TTL format) encoded as binary objects
RDF Store
Apache Spark

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

Figure 2 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).

3.1 Trajectory Reconstruction & Semantic Enrichment (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.

- LED is a Flink component that processes a stream of raw messages (i.e., AIS dynamic messages) and enriches 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.

  Regarding surveillance data in the aviation domain, LED is integrating streaming data from different data sources, namely ADSB data provided by ADSBHub and by FligthAware and IFS RadarData. For AIS data in the maritime domain, several error flags are being generated. The detailed description of the operation of LED is provided as a separate deliverable D1.8 "Cross-streaming data processing (final)" on M30.

- 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[1] format as a Kafka stream. The detailed description of the operation of SG is provided in 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)[2] 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 D1.9 "Data Integration, Management (final)". The implementation of RDF data generators is in Java, whereas the link discovery framework – which is the most processing-intensive operation in SI – is implemented in Apache Flink.
- 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.

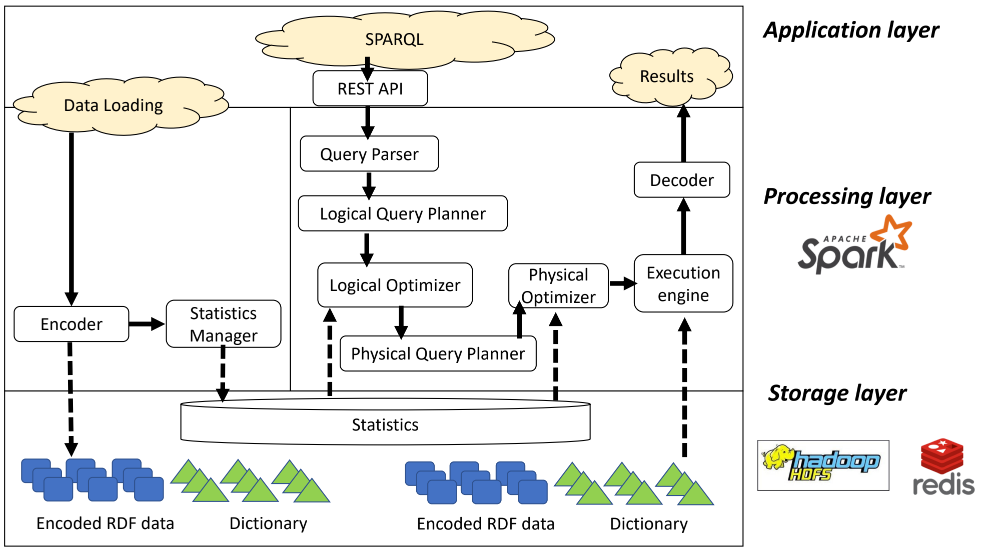


Figure 3: Overview of the *datAcron distributed RDF engine* (flow #2).

[1]https://avro.apache.org/
[2]https://www.w3.org/TR/turtle/

3.2 Distributed RDF Storage and Processing (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 querying purposes. The detailed description of the operation of the DM module which contains the distributed RDF store is provided as a separate deliverable D1.10 "Data storage and querying (final)".

Figure 3 illustrates the architecture of the *datAcron distributed RDF engine.* It is responsible for the *batch processing* part of the datAcron architecture (cf. deliverable D1.2), and has been designed and built from scratch during the course of the project. By design, the *datAcron distributed RDF engine* targets the following objectives:

- Scalable storage and processing for vast volumes of RDF data, in the order of Billions of RDF triples (targeting the *Volume* dimension of Big Data)
- Support for spatio-temporal RDF data, i.e., RDF data which are mostly associated with spatio-temporal information
- Efficient processing strategies that employ optimization techniques, in order to prune significant subsets of data and reduce execution time
- A prototype implementation that includes the functionality provided by a typical data warehouse system, from loading data to querying and optimization.

Description of Layers The *datAcron distributed RDF engine* encompasses two main layers: (a) a distributed storage layer, and (b) a parallel data processing layer. In addition, an application layer has been introduced to demonstrate the offered functionality and allow its use by external applications.

The **distributed storage layer** provides a scalable storage solution for the spatio-temporal RDF data that is generated by the online components described in Flow #1. HDFS is used for storage of the RDF data, whereas the actual files storing the RDF triples are Parquet files. Following best practices in storage of RDF data, we employ a *dictionary* approach which maps strings representing URIs or literals to integer values, thus enabling more effective storage due to compression and more efficient access. Consequently, besides the encoded RDF triples, we need to store the dictionary that maps strings to integers and vice-versa. Due to the access patterns on dictionary data which consist of random lookups, we select an in-memory, distributed key-value store (REDIS) for storing the dictionary. Last, but not least, the storage layer also maintains various data statistics, mainly in the form of histograms, in order to support query planning and optimization.

The **parallel processing layer** implements a parallel data processing engine for spatio-temporal RDF data. The main modules of the engine are implemented from scratch in Apache Spark, thus providing a efficient and scalable processing solution. It receives as input SPARQL queries along with spatio-temporal constraints and returns matching RDF data, after parallel/distributed query execution. The main modules of the query engine include:

1. Query parser: checks the correctness of syntax, ensures that the SPARQL query is specified correctly, and transforms the query into an internal representation that will be used by the remaining modules of the processing engine
2. Logical query planner: constructs the logical query plan, a tree representation of the different logical operators in the query

3. Logical optimizer: performs rule-based optimization to derive an optimized logical plan that will be passed further to the physical execution modules
4. Physical query planner: constructs candidate physical execution plans, taking into account various factors, such as storage organization, data location, etc.
5. Physical optimizer: performs cost-based optimization to derive the best physical execution plan
6. Execution engine: practically executes the Spark code that implements the selected physical execution plan
7. Decoder: performs lookups in the dictionary, in order to transform the resulting RDF data from encoded integer values back to their original string representation, so that they are meaningful to the user.

The **application layer** is not a core module of the implemented *datAcron distributed RDF engine*; rather, its role is for support the invocation of query processing from other modules of the datAcron integrated prototype. For this purpose, we use Apache Livy[3], an incubator Apache project that allows submission of Spark jobs from web and mobile applications. In practice, we have developed a web application that implements a REST API and allows submission of SPARQL queries to the *datAcron distributed RDF engine.* In addition, we have developed a primitive web page that also allows humans to submit SPARQL queries using the REST API.

Operations An application that uses the *datAcron distributed RDF engine* can use two main operations: data loading and querying.

The **data loading** operation considers as input an RDF data set that can be arbitrarily large. In the context of datAcron, this data set consists of spatio-temporal RDF data and it is the output of the online data transformation and link discovery components described in Deliverable D1.9. During data loading, the RDF data is encoded to integer values based on the Encoder (described in Deliverable D1.10). The Encoder produces as output encoded RDF data and a dictionary that allows decoding of integer values back to strings (URIs or literals). The encoded RDF data is stored in HDFS, whereas the dictionary is stored in-memory using REDIS, a scalable in-memory distributed key-value store. In addition, during the data loading operation, the Statistics Manager computes various statistics about the loaded data, which are also stored to be used during query execution, in order to derive an optimized query execution plan.

The **querying** operation takes as input a SPARQL query, which is directed to the processing layer of the engine for query evaluation, and provides the matched RDF triples as result. The processing layer is implemented in Apache Spark, a state-of-the-art solution for batch, in-memory processing of Big Data. First, the SPARQL query undergoes query parsing, which aims to ensure correct formulation and to represent it as a logical query plan that can be manipulated by the other components of the processing engine. The logical query plan is given as input to the *logical optimizer* whose role is to perform a set of standard optimizations (such as reordering join operations), and generate a set of potential physical execution plans. Then, the *physical optimizer* examines these physical plans and performs cost-based optimization, also exploiting the available statistics, to select the best physical execution plan. This is provided to the execution engine that performs query processing and provides the resulting RDF triples. Prior to result delivery to the application, a *decoder* performs translation of the encoded triples back to their string representation.

[3]https://livy.incubator.apache.org/

3.3 Online Future Location Prediction and Trajectory Prediction (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 enriched data integrated into the corresponding sub-flows #1x and then published in the "enhanced surveillance data stream".

T/FLP contains algorithms for trajectory and future location prediction. These algorithms operate in two different modes: (a) normal mode, in which they consume the output of Flow #1, or (b) streaming simulation mode, in which they consume CSV files in a streaming fashion, mainly for testing their functionality. T/FLP operates on the following set of attributes, in order to perform its task: ID, ts (timestamp), longitude, latitude, altitude (for the aviation domain).

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".

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. The prediction results can be stored 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 Trajectory Data Analytics module (see: Sub-flow #6b).

3.4 Complex Event Recognition / Forecasting Online (Flow #4)

The modules Complex Event Recognition (CER) and Complex Event Forecasting (CEF) are deployed and operate on both the maritime AIS data and the aviation surveillance 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.

Regarding the 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 processes the synopses stream of vessels and attaches predictions of predefined patterns (i.e., defined by regex). Each pattern is expressed as *current relative timestamp, start time of completion interval, end time of completion interval, probability of the pattern.* As an example, consider the following prediction of a pattern: "change_in_heading.gap_start.gap_end.change_in_heading":2.0, 13.0, 17.0, 0.65.

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

Complex Event Recognition Regarding event recognition, the CER module consumes serialized ST_RDF java objects and produces serialized ComplexEvent java objects that contain a JSON string with the recognised complex events. Several patterns have been implemented and tested. For the maritime use case, Table 3 provides an overview of the recognised complex events and their association to maritime situational indicators (MSIs).

MSI	Complex event
8	Not compatible with area
9	Not compatible with vessel type
19	Under way
20	At anchor or moored
21	Movement ability affected
22	Aground
23	Engaged in fishing
24	Tugging
25	In SAR operation
26	Loitering
27	Dead in water, drifting
28	Rendezvous

Table 3: List of implemented MSIs and complex events.

In turn, for the flight planning use case, the following complex events have been implemented and can be recognised: top-Of-Climb, top-Of-Descent, deviation from flight plan, hold entry, hold exit.

3.5 Interactive Visual Analytics Online (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 (Section 3.4).

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 is to communicate current parameter settings from the computational modules T/FLP and SG to the IVA module.

We have 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 been deployed successfully.

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.

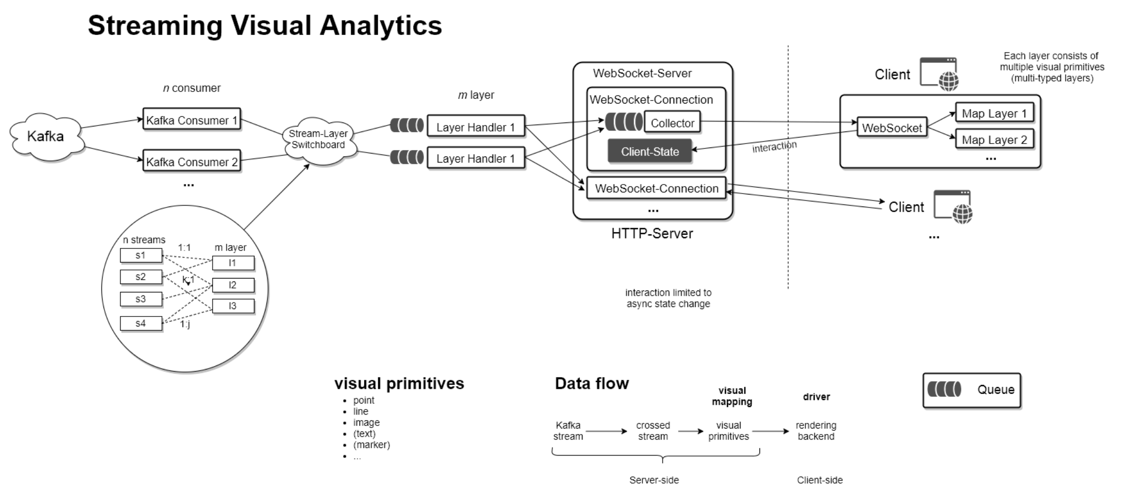


Figure 4: Architecture of the IVA component.

Implementation Details of Flow #5 The implementation of the real-time visualization VA module follows the client-server architecture . Figure 4 depicts the architecture of this component. While we refer to deliverable D4.8 (Section 3), for more details, a brief overview is provided here to make this deliverable self-contained.

On the server side, Kafka streams are consumed. All the applications on the server side are developed in Java. The client side is browser-based and written in JavaScript. The incoming messages from the Kafka streams are processed one by one and from every incoming Kafka message a JSON object is generated. Each JSON object is forwarded to the client using WebSockets.

The architecture depicted in Figure 4 is generic with respect to its inputs. In fact, any input source providing enriched positional information of moving objects can be consumed and fed to the visualization front end, as long as it is provided as a Kafka stream. Essentially, this facilitated the integration with the Kafka streams provided by the components T/FLP and CER/F, as well as with the enhanced surveillance stream provided by SI.

3.6 Trajectory Data Analytics Offline (Flow #6)

This flow refers to the offline Trajectory Data Analytics module and is broken down to sub-flow #6a and sub-flow #6b. More specifically, sub-flow #6a provides the input from the DM to the offline Trajectory Data Analytics module and sub-flow #6b writes back the output of this module to the DM.

Actually, in this module the analyst selects the desired component of the offline Trajectory Data Analytics module and poses a SPARQL query to the DM so as to retrieve its input. Subsequently, the RDF triples are fetched (Flow #6a) and parsed, in order to feed the needs of the desired component of the offline Trajectory Data Analytics module, and the output can be written back to the DM which can be accessible to the analyst again via the SPARQL Interface.

An example of the input of a component is provided for the Distributed sub-trajectory clustering analysis. More specifically, the analyst poses a SPARQL query to the DM and retrieves

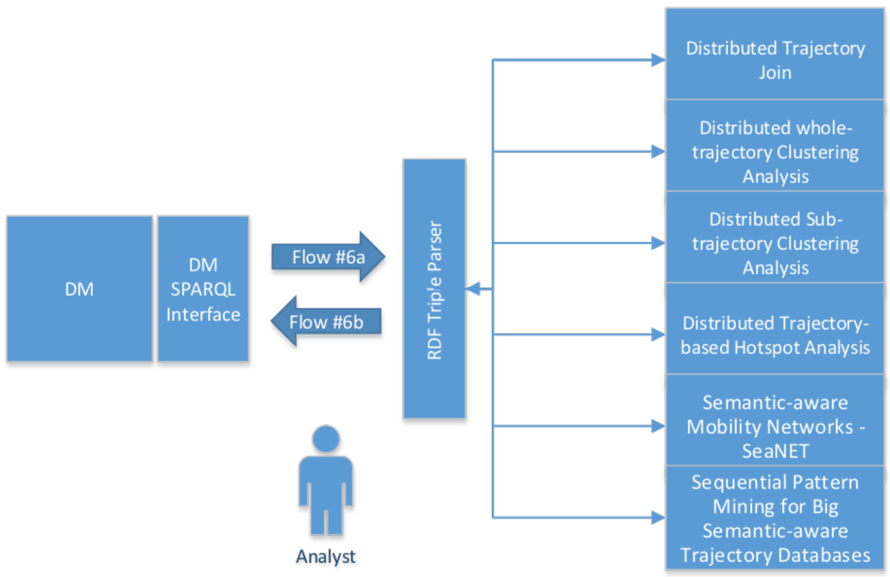

DM
DM SPARQL Interface
Flow #6a
Flow #6b
RDF Triple Parser
Analyst
Distributed Trajectory Join
Distributed whole-trajectory Clustering Analysis
Distributed Sub-trajectory Clustering Analysis
Distributed Trajectory-based Hotspot Analysis
Semantic-aware Mobility Networks - SeaNET
Sequential Pattern Mining for Big Semantic-aware Trajectory Databases

Figure 5: The offline Trajectory Data Analytics (TDA) module.

the desired data in the form of RDF triples. Then, the RDF triple parser transforms these triples to the appropriate format for the specific component. In fact, this component receives one record per point and must be of the form: (Id: the object id, t: the timestamp of the point, lon: the longitude of the point, lat: the latitude of the point, alt: the altitude of the point (optional)).

Finally, the output of this component is a representative sub-trajectory for each cluster, along with the sub-trajectories belonging to the cluster. This output can be converted to RDF, stored to the DM and become available to the analyst via the SPARQL interface provided by DM.

3.7 Complex Event Recognition / Forecasting Offline (Flow #7)

This flow realizes the offline complex event recognition and forecasting functionality. This is as example of a datAcron flow that is only technically possible in the architecture, but actually not required from the semantics of the applications considered in the project.

In the second half of the project, we have extended the capabilities of the online CER/CEF, such that there is no need to work in some preparatory manner off-line. This practically eliminated the need for explicitly developing Flow #7.

3.8 Interactive Visual Analytics Offline (Flow #8)

The IVA component provides facilities for the visual exploration of data and visual-interactive support for building and refining models and their parameter settings. It therefore targets analysis experts working on the strategic level using data at rest, but may in suitable cases also provide visualizations with limited interactivity on the tactical or even operational level. Therefore, the IVA component mostly operates in batch mode (offline), and can be fed with data from a wide variety of data sources, including the DM.

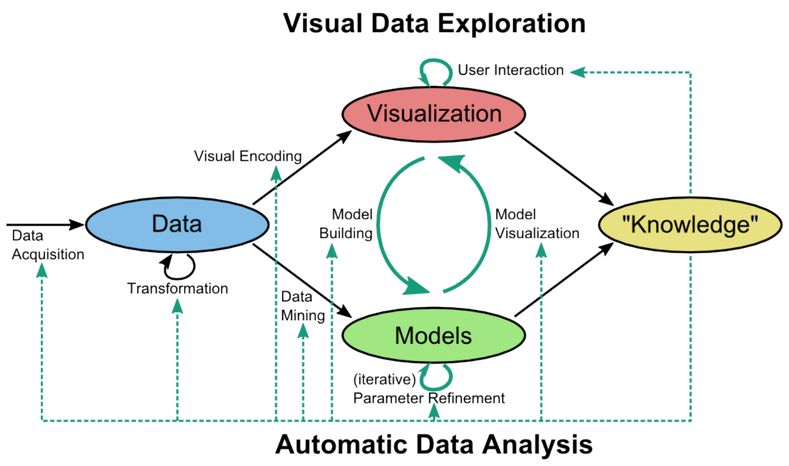


Figure 6: The Visual Analytics Loop supported by datAcron's IVA component, adapted from [2].

The purpose of the Visual Analysis approach is to combine algorithmic analysis with the human analyst's insight and tacit knowledge in the face of incomplete or informal problem specifications and noisy, incomplete, or conflicting data. Visual Analysis therefore is an iterative process where intermediate results are visually evaluated to ascertain and inform subsequent analysis steps based on prior knowledge and gathered insights. The underlying conceptual model is the Visual Analytics Loop (Fig. 6). Specifically, it is worth noting that due to the exploratory focus, VA does not prescribe a rigid pipeline of algorithmic processing steps, nor does it prescribe a fixed composition of specific visualizations, as opposed to typical KPI dashboards.

To cope with these requirements in an efficient and scalable way, the *Visual Analytics* component within the integrated datAcron architecture is itself of a modular, extensible design, as shown in Figure 7. It comprises four principal component groups - data storage, analysis methods, data filtering and selection tools, and of course, visualization techniques. Different components are typically composed in an ad-hoc fashion, through visual-interactive controls, to facilitate the workflow required by the human analyst's task at hand. In particular, this allows creating linked multiple views to simultaneously visualize complementary aspects of complex data or analytical models. Figure 7 indicates by color marks matching colors from Figure 6 what components are typically involved in which phases of the VA loop.

Data Storage The data storage component serves three functions. First, it provides an interface to the Data Manager (DM). This allows read access to historical data, specifically, from the distributed spatio-temporal RDF store. In cases where analysis results in data artifacts that are worth persisting, for example interactively defined area boundaries, "typical" vessel trajectories, or analytical models for later use such as spatio-temporal flow graphs of aircraft, these are pushed down to the Data Manager for long-term storage and retrieval.

Second, this component provides management of intermediate analysis results. This is necessary as interactive analysis frequently requires data representations that are different from archival storage for efficiency reasons, e.g. by denormalizing data kept in a relational schema.

In addition, the explorative and iterative nature of analysis often results in intermediate data attributes that are almost immediately discarded for a refined result (e.g., cluster associations of entities after interactive parameter changes to the algorithm). Such data is never persisted and so is not handled by the distributed RDF store.

Third, ad-hoc analyses might often have the need to integrate external data not yet ingested by the *Data Manager*. Typical examples include data and models generated by scripts or other tools in standard formats (CSV files, Shape files, XML files, or local databases) by an analyst. Enabling this loose, file-based integration into the VA platform has proven essential in maintaining flexibility and extensibility in terms of the analyst's tool capabilities.

Data Selection and Grouping Similar to the data storage component, functionality of this module is divided between the central *Data Manager* and the VA components internal selector module. Complex queries to large historical or synopsis data are handed off to the query engine of the distributed RDF store (see Deliverable D1.10).

One key feature of Visual Analytics, however, is the ability to directly manipulate data and algorithm parameters through visual interaction. Therefore, interactive selection of data elements across multiple views allows the analyst to define complex, multi-faceted filters on data before analytical processing. An example is the simultaneous specification of a geospatial region in a map display, a specific time range from a time graph, and a subset of entities according to some cluster visualization (e.g., see [1]). Such compound queries are mostly executed on the in-memory representation held by the VA module's data storage component.

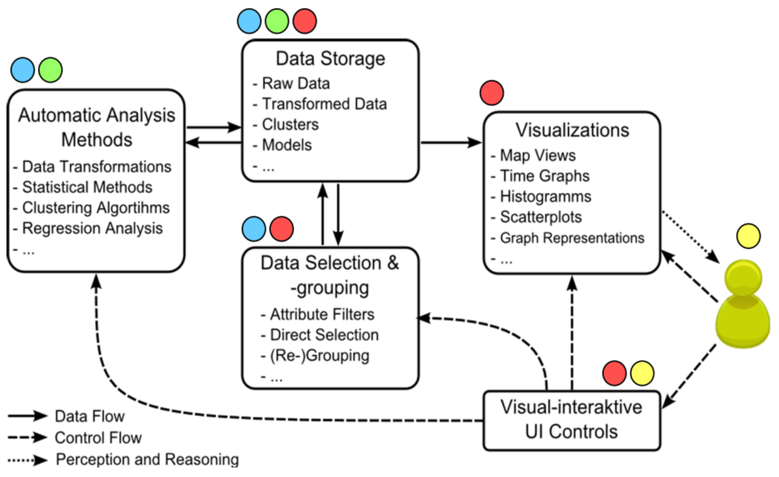


Figure 7: The Visual Analytics module architecture with its principal components to support the VA loop.

Analysis Methods From the perspective of the IVA offline module, analysis methods fall into the category those applied in exploratory analysis on the strategic level: selection and parameter tuning of algorithms prior to their application at operational (real-time) or tactical latency levels. The VA component facilitates the integration of a wide range of algorithms by loose coupling on the level of tabular and graph-structured data handled by the data storage component.

Visualizations Outwardly the core component of a visual analysis system, this component will provide the set of interactive visualization techniques needed for – primarily exploratory –

analysis. The final set of visualizations is not prescribed at this point of time, however. Rather, it is derived from use case requirements (see documents D5.3, D6.3), and is likely to involve research in designing novel visualization techniques. Available visualizations are expected to include established base techniques such layered map displays, time graph displays, but also specialized and complex representations such as dynamic flow graph visualizations that are one current focus of VA research in datAcron. Similar to integrated analysis techniques, visualization technique will be integrated on the level of internal data representation, i.e., based on how types of entities and their attribute structure are mapped to specific visual encodings by a given visualization technique.

4 Demonstration of the Integrated Prototype

This section provides a demonstration of the integrated prototype and its functionality, in relation with the individual constituent modules. Selected illustrations are used in order to show the functionality offered to the end-user, as well as how the outputs of individual modules are combined in meaningful and comprehensive visualizations.

Section 4.1 focuses on the online part of the integrated prototype, showing visualizations produced in (near) real-time, serving operational needs at low latency.

Section 4.2 places emphasis on the offline functionality, demonstrated mainly by data analytics performed over integrated data from diverse data sources.

4.1 Online Functionality

Figure 8 provides a general overview of the user interface. To the right, a legend with all MSIs and synopsis events is depicted. The events are added to the current location of each vessel (see Figure 9). In the upper left, a button for entering the settings menu can be seen. Additionally to the regular map, the Natura2000 (green) and fishing areas (pinkish-white) are added as WMS layers.

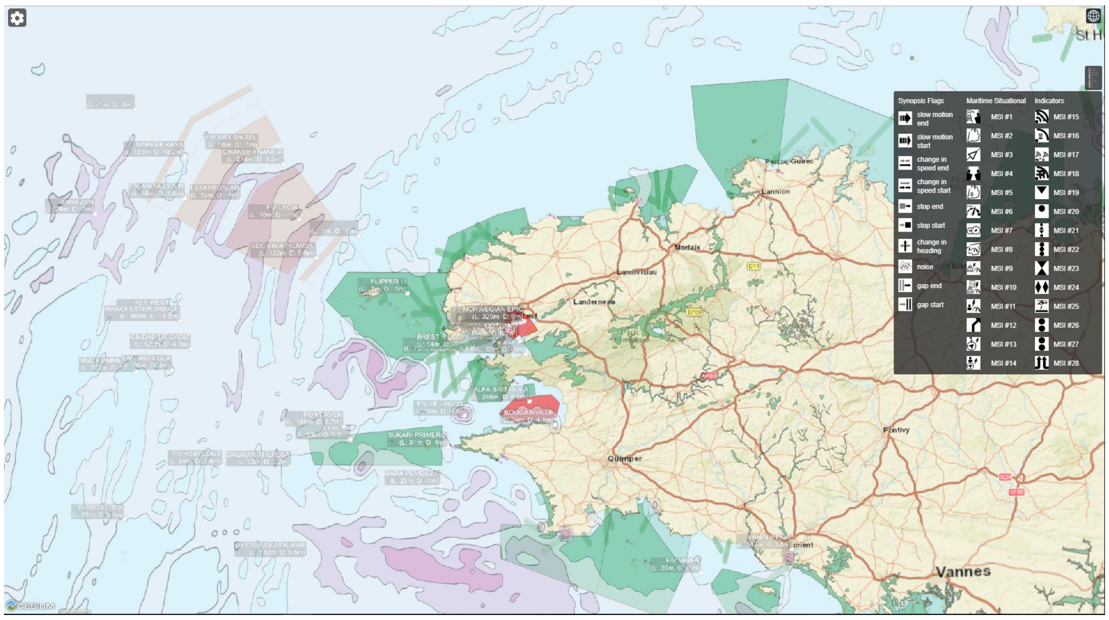


Figure 8: Overview of main user interface for online visualizations.

The data shown in Figure 8 are produced by all online modules of the integrated prototype, and comprise results of several information flows (those addressing online scenarios, namely: #1, #3, #4, and #5). In more detail:

- Vessel positions have been processed and annotated by LED and SG modules, as well as integrated with contextual and weather data by the SI module.
- MSIs have been detected as a result of different components: entry/exit to areas is produced by LED, critical points are produced by SG, simple spatial and spatio-temporal events are the results of the link discovery component within SI, complex events are produced by CER, whereas future location predictions are produced by FLP.
- Spatial areas (such as Natura2000) are fetched and stored locally at the Viz component, while they are stored persistently in the distributed RDF store.

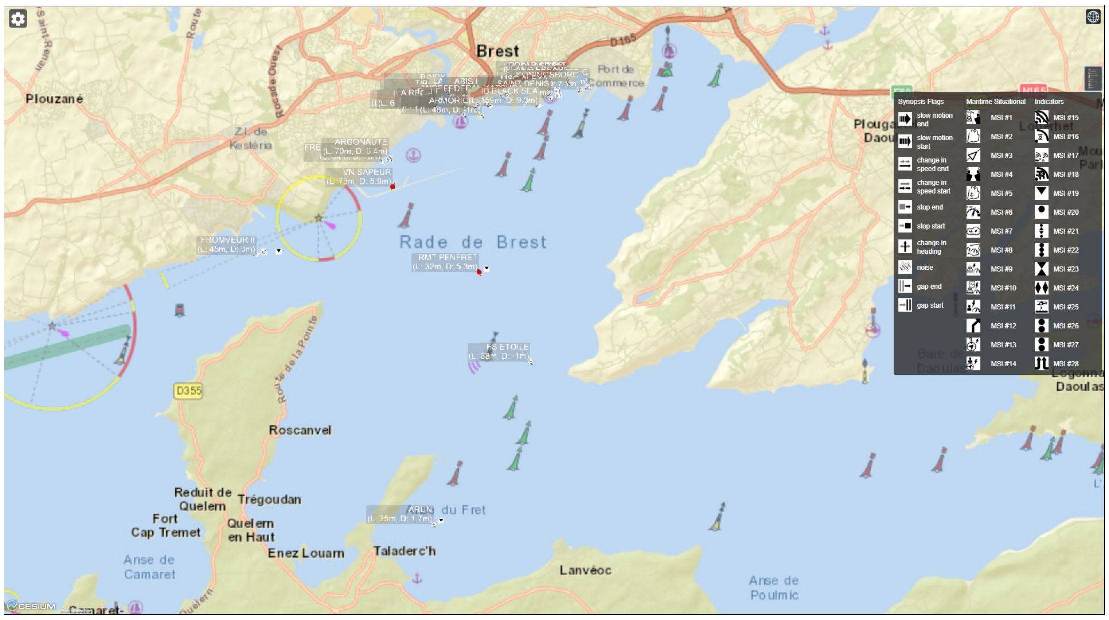


Figure 9: Closer zoom to Brest harbour area.

In Figure 10, four vessels are shown and each vessel has a box with name and some general vessel information (e.g., its length). This information is the result of combining live positional data with archival data that contain static information about vessels (vessel registry). If an event is detected by CER or one of the other modules, the vessel position is marked with the according flag from the legend. Any module that detects an event produces the corresponding output to its Kafka output topic. The Viz component uses a mechanism that combines the output Kafka topics of individual modules, and visualizes the corresponding events, based on the interaction with the end-user and the user settings.

In Figure 11 , the same image as above (Figure 10) is depicted, but with settings options and filter activated. Observe that two vessels are now highlighted because the were selected by their respective vessel id. The output of each component (producing a Kafka topic) is implemented as a single layer, which can be turned on and off (first 6 sliders in the Settings menu).

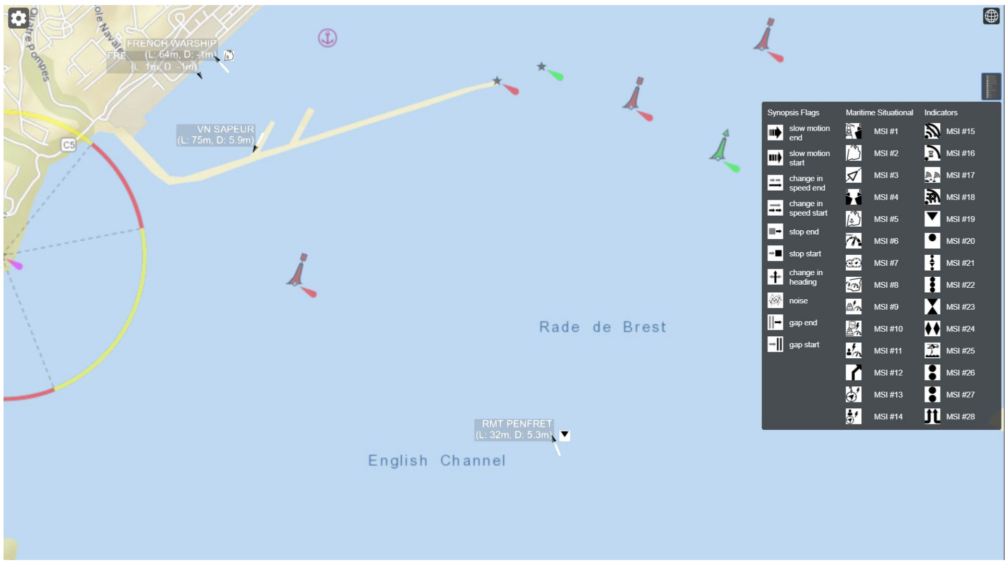


Figure 10: Focusing on four vessels.

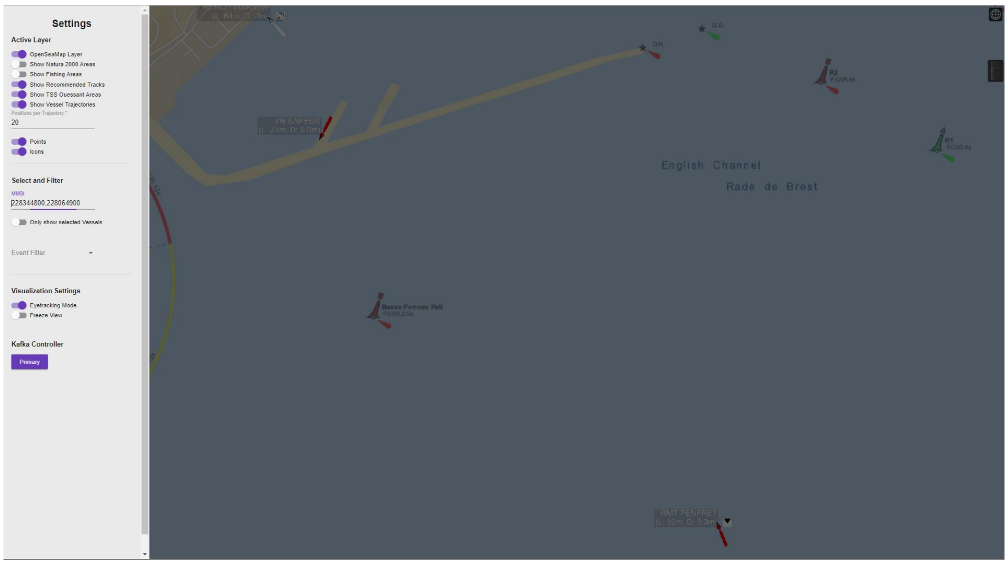
4.2 Offline Functionality

The offline functionality offered by the integrated prototype consists mainly of the distributed RDF store implemented in the Data Manager component, and the different data analytics modules. Essentially, this functionality can be demonstrated by flows: #6, #7, and #8, which concern trajectory analytics, complex event recognition/forecasting, and visual analytics. In addition, flow #2 is supporting the offline functionality, as it describes the data loading procedure of data that has been transformed in RDF to the distributed RDF store.

In all cases of offline analytics, two types of modules are involved. On the one hand, the distributed RDF store serves as batch processing layer and provides integrated data that is subject to advanced data analytics [3]. On the other hand, the different data analytics modules receive input data from the distributed RDF store, ingest this data internally in order to fill their data and index structures, and perform data analysis operations. To demonstrate the results of offline data analytics, we use visualizations from the trajectory analytics module and the visual analytics module.

Figure 12 presents an example of results of two runs of the sub-trajectory clustering algorithm (S^2T-Clustering) [4, 5], developed in the trajectory analytics module, for comparison purposes. The cluster representatives from the two runs are viewed in a 3D display. The user can either put both sets of results in the same 3D display or create two 3D displays, each showing one set of results. In the first case, the user can interactively switch on and off the visibility of each set of results. The same can be done with a map display.

Moreover, the user experiences in discovering and visualizing other interesting patterns, such as the holding patterns typically performed by aircrafts as they approach to their destination, in our case London airports (as it is illustrated in Figure 13).



Settings

Active Layer

- [✓] OpenSeaMap Layer
- [ ] Show Natura 2000 Areas
- [ ] Show Fishing Areas
- [✓] Show Recommended Tracks
- [✓] Show TSS Ouessant Areas
- [✓] Show Vessel Trajectories

Positions per Trajectory *: 20

- [✓] Points
- [✓] Icons

Select and Filter

MMSI: 228344800,228064900

- [ ] Only show selected Vessels

Event Filter

Visualization Settings

- [✓] Eyetracking Mode
- [ ] Freeze View

Kafka Controller

Primary

Figure 11: Settings options.

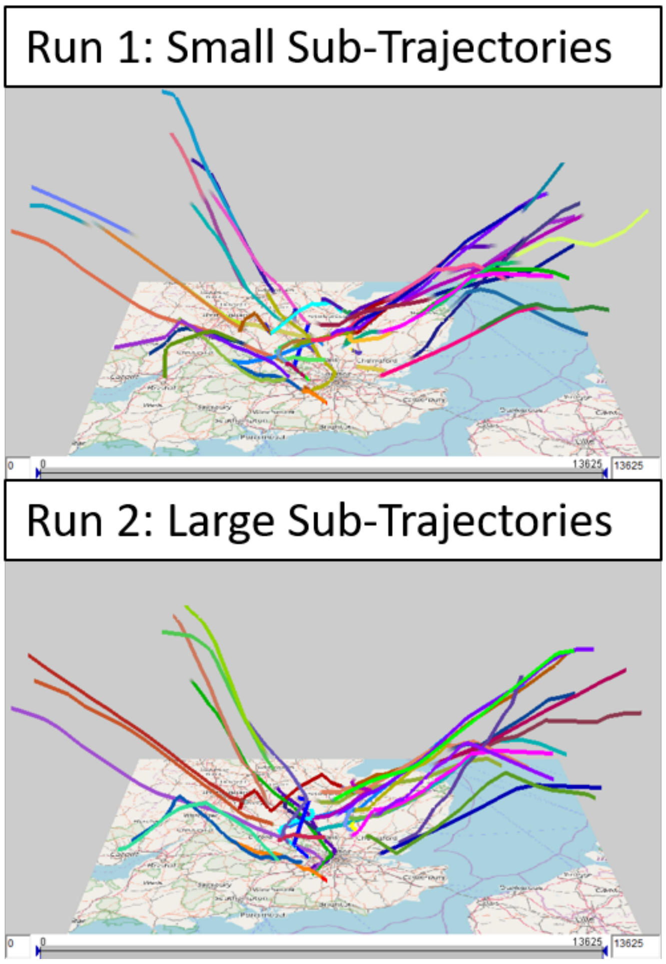


Figure 12: Time-aware sub-trajectory clustering in action: cluster representatives from two different runs of sub-trajectory clustering are visually compared by means of a 3D display.

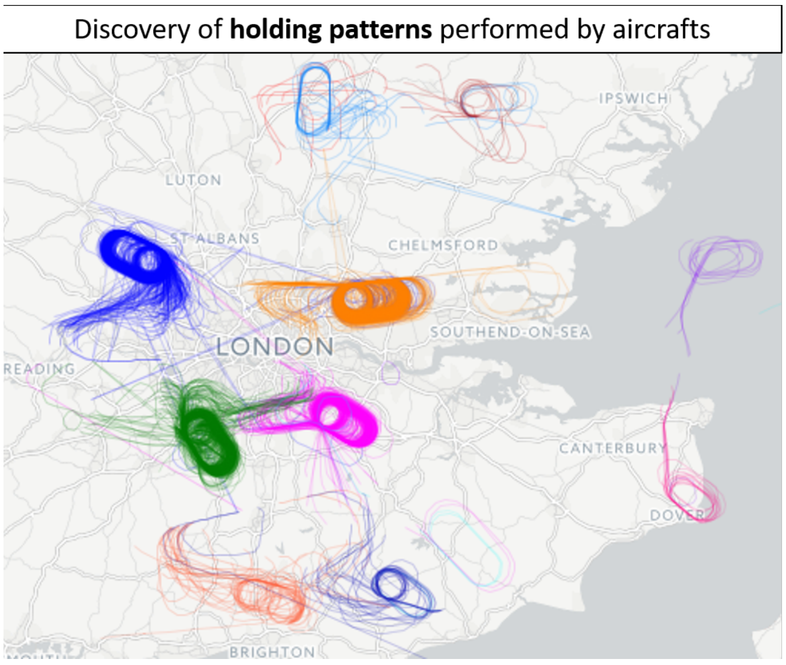


Figure 13: Time-aware sub-trajectory clustering in action (cont.): holding patterns performed by aircrafts are discovered and visualized.

5 Concluding Remarks

This report accompanies the demonstration of the integrated prototype (D1.12) and provides a short description of it. It presents different flows of information in datAcron, the modules that comprise the integrated prototype, their intercommunication, and implementation details.

Perhaps more importantly, it presents indicative visualizations provided to the end-user, thus demonstrating the functionality of the integrated prototype, which is further analyzed in terms of the individual modules and their functionality. As such, the integrated prototype addresses research challenges such as those described in [6] for the maritime domain, but also applicable to the air-traffic management domain.

References

[1] Gennady L. Andrienko, Natalia V. Andrienko, Christophe Claramunt, Georg Fuchs, and Cyril Ray. Visual analysis of vessel traffic safety by extracting events and orchestrating interactive filters. In *Proceedings of Maritime Knowledge Discovery And Anomaly Detection Workshop (MKDAD)*, 2016.

[2] Daniel A. Keim, Gennady L. Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. Visual analytics: Definition, process, and challenges. In *Proceedings of Information Visualization - Human-Centered Issues and Perspectives*, pages 154–175. 2008.

[3] Panagiotis Nikitopoulos, Akrivi Vlachou, Christos Doulkeridis, and George A. Vouros. Distrdf: Distributed spatio-temporal RDF queries on spark. In *Proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26, 2018.*, pages 125–132, 2018.

[4] Nikos Pelekis, Panagiotis Tampakis, Marios Vodas, Costas Panagiotakis, and Yannis Theodoridis. In-dbms sampling-based sub-trajectory clustering. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017.*, pages 632–643, 2017.

[5] Panagiotis Tampakis, Nikos Pelekis, Natalia Andrienko, Gennady Andrienko, Georg Fuchs, and Yannis Theodoridis. Time-aware sub-trajectory clustering in hermes@postgresql. In *35th IEEE International Conference on Data Engineering (ICDE)*, 2019.

[6] George A. Vouros, Akrivi Vlachou, Georgios M. Santipantakis, Christos Doulkeridis, Nikos Pelekis, Harris V. Georgiou, Yannis Theodoridis, Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Christophe Claramunt, Cyril Ray, David Scarlatti, Georg Fuchs, Gennady L. Andrienko, Natalia V. Andrienko, Michael Mock, Elena Camossi, Anne-Laure Jousselme, and Jose Manuel Cordero Garcia. Big data analytics for time critical mobility forecasting: Recent progress and research challenges. In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 612–623, 2018.