

**Developing Data-Intensive Cloud
Applications with Iterative Quality
Enhancements**



Demonstrators implementation plan

Deliverable 6.1

Version: Final

Deliverable:	D6.1
Title:	Demonstrators implementation plan
Editor(s):	George Giotis (ATC), Christophe Joubert (PRO), Youssef Ridene (NETF)
Contributor(s):	Ilias Spais (ATC), Matej Artac (XLAB).
Reviewers:	Giuliano Casale (IMP), Matteo Giovanni Rossi (PMI)
Type (R/P/DEC):	Report
Version:	1.0
Date:	31/05/2016
Status:	Final version
Dissemination level:	Public
Download page:	http://www.dice-h2020.eu/deliverables/
Copyright:	Copyright © 2016, DICE consortium – All rights reserved

DICE partners

ATC:	Athens Technology Centre
FLEXI:	Flexiant Limited
IEAT:	Institutul E Austria Timisoara
IMP:	Imperial College of Science, Technology & Medicine
NETF:	Netfective Technology SA
PMI:	Politecnico di Milano
PRO:	Prodevelop SL
XLAB:	XLAB razvoj programske opreme in svetovanje d.o.o.
ZAR:	Unversidad De Zaragoza



The DICE project (February 2015-January 2018) has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869

Executive summary

Stemming from the initial requirements in T1.1, this deliverable defines a concrete implementation plan for the demonstrators. Building on top of D1.2 “Requirement specification” results, DICE demonstrators revised their real-life industrial scenarios with respect to the objectives of WP6. In addition to, DICE demonstrators have the chance to familiarize themselves with DICE tools, identify advantaged and disadvantages and subsequently revise the list of demonstrator’s requirements define in M6.

The current report is also addressing some comments received by project’s EC reviewers during the 1st official Review Meeting that was performed at 18/05/2016 in Brussels. DICE demonstrators analysed the useful comments/suggestions perceived and addressed them from their case study point of view. The following recommendations were addressed:

- **Recommendation 1: Privacy and Security**. As it is detailed in Section A.3, DICE will deal with security aspects primarily in terms of access control
- **Recommendation 2: Requirement Analysis**. DICE demonstrator requirements were updated and the latest version is included in Sections B.4, C.4 and D.4.
- **Recommendation 4: Measuring project impact**. DICE consortium defined several KPIs to ensure that DICE objectives are met. Three of them (KPI-RI-2, KPI-RI-3, KPI-RI-7) will be validated by DICE demonstrators. In order to ensure that project’s impact would be measured, DICE demonstrators and DICE technical team defined metrics (one per DICE tool) with respect to these the KPIs (Section A.2)

The current report consist of the following sections:

- **Section A**, which provides an overview of WP6 objectives and activities, and positions demonstrators in the context of DICE. Furthermore, details the metrics that will be used to validate DICE framework and refers to privacy and security aspects related to DICE. Finally, the sections describes the four milestones that have been set by DICE demonstrators.
- **Section B**, which details the ATC demonstrator focusing on: a) its business and technical goals, b) the current and future status, c) how the demonstrator is mapped to DICE architecture, d) the revised scenario and requirements update. Finally, the section provides an implementation plan that will guide the technical team of the demonstrator.
- **Section C**, which details the Prodevelop demonstrator focusing on the same aspects as in the previous section and
- **Section D**, which details the NETF demonstrator focusing on the same aspects as in the previous section.

Table of contents

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS.....	4
LIST OF FIGURES	5
LIST OF TABLES	5
A. INTRODUCTION.....	6
A.1. Demonstrator's overview and challenges.....	6
A.2. Indicators of evaluation	8
A.3. Security and privacy aspects	9
A.4. Roadmap.....	9
B. THE ATC DEMONSTRATOR: NEWSASSET	11
B.1. Introduction	11
B.1.1 Business goals	11
B.1.2 Technical goals.....	11
B.2. Mapping DICE tools to ATC demonstrator	12
B.2.1 Current status.....	14
B.2.2 Future plans	15
B.3. Scenarios revision – The News Orchestrator Application.....	16
B.3.1 Current status.....	16
B.3.2 Proposed scenario.....	18
B.4. Requirements update	19
B.5. Implementation plan.....	20
C. PRODEVELOP DEMONSTRATOR: POSIDONIA OPERATIONS.....	25
C.1. Introduction	25
C.1.1 Business goals	25
C.1.2 Technical goals.....	26
C.2. Mapping DICE tools to Prodevelop demonstrator	26
C.2.1 Current status.....	28
C.2.2 Future plans	29
C.3. Scenario revision	30
C.4. Requirements update	37
C.5. Implementation plan.....	40
D. NETF DEMONSTRATOR: EGOV TAX FRAUD DETECTION - BIG BLU	45
D.1. Introduction	45
D.1.1 Business goals	45
D.1.2 Technical goals.....	45
D.2. Mapping DICE tools to NETF demonstrator	46
D.2.1 Current status.....	46
D.2.2 Future plans	50
D.3. Scenarios revision.....	54
D.4. Requirements update	57
D.5. Implementation plan.....	61

List of figures

Figure 1. WP6 activities	6
Figure 2. DICE demonstrators and challenges	8
Figure 3. DICE demonstrators roadmap.....	10
Figure 4. The ATC demonstrator.	13
Figure 5. DICE toolset in the ATC demonstrator.....	13
Figure 6. DMON platform operated by ATC demonstrator.....	14
Figure 7. ATC demonstrator: the DySCO workflow	17
Figure 8. ATC demonstrator: The News Orchestrator Application	18
Figure 9. ATC demonstrator: The news-orchestrator components/modules.....	18
Figure 10. ATC demonstrator: Roadmap	21
Figure 11. DICE toolset in the Prodevelop demonstrator	27
Figure 12. Prodevelop Demonstrator: Posidonia Operations general architecture.....	30
Figure 13. Prodevelop Demonstrator: A Storm – like model.....	31
Figure 14. Prodevelop Demonstrator: The port operator console	32
Figure 15. Prodevelop Demonstrator: Events detection.....	33
Figure 16. Prodevelop Demonstrator: Workflow configuration interface.....	34
Figure 17. Prodevelop demonstrator: Roadmap	40
Figure 18. NETF demonstrator: General architecture of the Big Blu	46
Figure 19. NETF demonstrator: Homepage	47
Figure 20. NETF demonstrator: Fraud detection list (status and results).....	48
Figure 21. NETF demonstrator: Taxpayers generation	48
Figure 22. NETF demonstrator: Taxpayers metamodel	49
Figure 23. NETF demonstrator: Part of our DPIM model built using the DICE profile.....	50
Figure 24. DICE toolset in the NETF demonstrator.....	51
Figure 25. NETF demonstrator: Overview.....	55
Figure 26. NETF demonstrator: Social aid fraud	55
Figure 27. NETF demonstrator: Relocation fraud.....	56
Figure 28. NETF demonstrator: Detailed architecture	57
Figure 29. NETF demonstrator: Roadmap	62

List of tables

Table 1. DICE KPIs and tool metrics addressed by demonstrators.....	8
Table 2. DICE tools and ATC demonstrator	15
Table 3. DICE tools and Prodevelop demonstrator.....	27
Table 4. DICE tools and NETF demonstrator	51

A. Introduction

The current report is the first deliverable of Work Package 6 “Validation and Demonstrators” focusing on the demonstrator’s implementation plan. The ultimate goal of WP6 activities is the realization of real industrial demonstrators by operating DICE framework. Three data intensive applications (a media system, an e-Government application and a Geo-fencing framework) are using the DICE framework in order to validate its capability of facilitating the production of high-quality real-life applications in a variety of business domains.

Being the core part of WP6 (Figure 1 presents WP6.), the selected industrial applications will not only act as the proof of concept demonstrators of what DICE is offering, but will validate and verify project’s objectives by continuously providing rapid feedback and thus steering the research direction of DICE. In addition to, demonstrators are bringing in the business and industrial requirements ensuring project’s long-term sustainability.

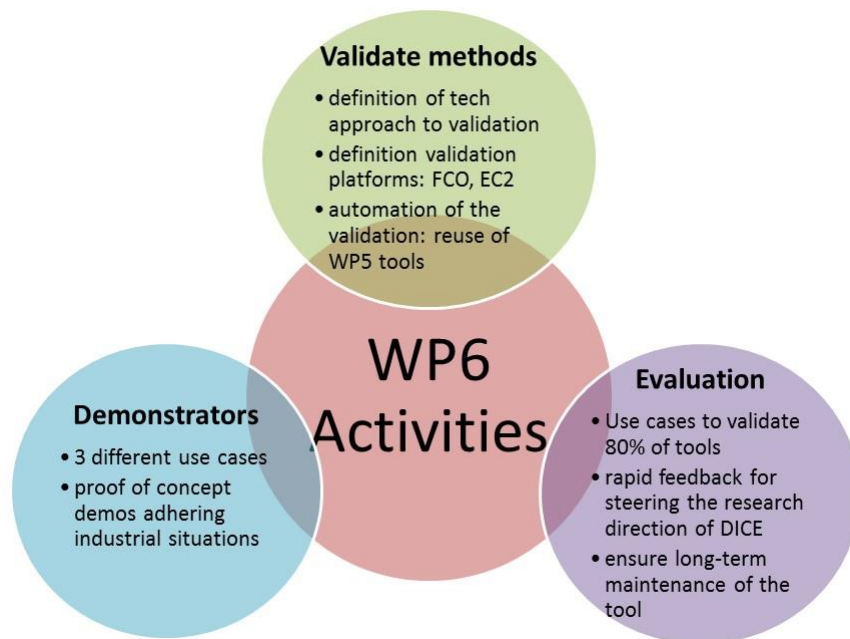


Figure 1. WP6 activities

Work Package 6 objectives are the following:

- Provide multiple feedback to the project:
 - Business requirements with respect to the selected industrial scenarios;
 - Functional and non-functional requirements;
 - Evaluation remark, ensuring that DICE will not end up with an obsolete framework. The goal for the demonstrators is to validate 80% of tools.
- Ensure that DICE can quickly react to the requirements of the market:
 - Desirable and usable offerings for the stakeholders.
- Long-term sustainability:
 - Proof of concept realization of the “DICE innovation” ecosystem.

A.1. Demonstrator’s overview and challenges

There are three demonstrators (case studies) in the context of DICE project:

- **Distributed data-intensive media system: News as an Asset - NewsAsset** (Section B) is an end-to-end multimedia cross-channel solution for evolving news agencies, broadcasters and

publishers, branded by ATC (Athens Technology Centre). To keep up with the demands of the news and media domain, the services and application provided by NewsAsset need a constant expansion to incorporate new data sources and new distribution channels from social media, mobile phones or sensor networks. DICE is focusing on this scenario on addressing the challenge **of managing the complexity of large software and data-intensive systems** like NewsAsset, by handling features like **data velocity, volume, rate of update and granularity**.

- **DICE-based Geo-fencing for the Maritime Sector: Posidonia Operations** (Section C). Posidonia Operations is an Integrated Port Operation Management System highly customizable that allows a port to optimize its maritime operational activities related to the flow of vessels in the port service area, integrating all the relevant stakeholders and computer systems. In technical terms, Posidonia Operations **is a real-time and data intensive platform** able to connect to AIS (Automatic Identification System), VTS (Vessel Traffic System) or radar, and automatically detect vessel operational events like port arrival, berthing, unberthing, bunkering operations, tugging, etc. This use case is aimed at providing a DICE-based geo-fencing enabler with certain **complex event processing and streaming capabilities** for the maritime sector, using location awareness technologies to extend them for geo-location and geo-fencing. This is a data-intensive scenario for DICE **as the streaming and complex event processing engine** needs to analyse and offer a reactive decision support system for hundreds of position messages per second with **complex event processing based on geographical positions**.
- **Big Data for e-Government: Big Blu** (Section D). NETF use case aims to facilitate the task of **filtering and gathering data for fiscal agents in order to increase productivity**. Utilizing Big Data and Cloud processing technologies through DICE will be the key feature of the demonstrator. In fact, exploring and analyzing high volumes of **data from various heterogeneous sources** should be scalable in order to **address the complexity of this data-intensive configuration**. Our implementation plan, requirements, current and future plans are discussed in this document.

The objectives of the DICE demonstrators are:

- Address the challenge to manage the complexity of large software and data-intensive systems
 - e.g. analyze and evaluate different architecture alternatives for different quality and performance indicators
- Provide a realization of quality-driven processes in the Big Data domain
 - throughput, availability, security, fault tolerance and response time
 - give insights on current quality and performance metrics to iteratively improve them

The following figure provides an overview of the case study challenges that will be addressed by DICE framework.




Case study	Domain	Features & Challenges
Distributed data-intensive media system: NewsAsset 	<ul style="list-style-type: none"> News & Media Social media 	<ul style="list-style-type: none"> Large-scale software Data velocities Data volumes Data granularity Multiple data sources and channels Re-engineering
Big Data for e-Government 	<ul style="list-style-type: none"> E-Gov application 	<ul style="list-style-type: none"> Data volumes Legacy data Data consolidation Data stores Privacy & security Forecasting and data analysis
Geo-fencing 	<ul style="list-style-type: none"> Maritime sector 	<ul style="list-style-type: none"> Vessels movements Safety requirements Streaming & CEP Geographical information

Figure 2. DICE demonstrators and challenges

A.2. Indicators of evaluation

In order to operate experiments that will result in useful evaluation remarks for the DICE technical team, DICE consortium defined at the beginning of the project a number of Key Performance Indicators (KPIs) to guide the validation activities. In sequence, DICE technical team and case study providers defined metrics for all DICE tools so as to ensure that KPIs will be addressed and DICE objectives will be achieved.

The following three DICE KPIs will be addressed by the DICE demonstrators.

- **KPI-RI-2:** Productivity increase by at least 30% compared to today, verified on at least 1 demonstrator.
- **KPI-RI-3:** Reduction of quality issues by 50% after quality enhancement iterations
- **KPI-RI-7:** Implementation of 3 demonstrators, together validating at least 80% of tools.

The following table

Table 1. DICE KPIs and tool metrics addressed by demonstrators

<u>DICE Tool</u>	<u>Metrics</u>	<u>Target</u>	<u>KPI</u>
Delivery tool	Time from deployment modelling to deployment the DIA	<=50% of no-modelling approach	Productivity
Verification	Violations of timing constraints identified by verification tool	>=2	Quality (Verification)
Simulation	Prediction error of response time	=30% (median)	Quality (Performance)
Optimization	Cost prediction error	=30% (median)	Quality (Performance)
Monitoring	Time to configure the monitoring across the DIA; monitoring overhead	<=30% decrease; <5% overhead	Productivity

Anomaly detection	False positives	<10%	Quality
Trace checking	Number of violations found in traces	>=2	Quality (Verification)
Enhancement	Demand estimation error with respect to complete information traces; Number of antipatterns detected in one demonstrator	<20%; >=1	Quality
Quality testing	Manual time required in a test cycle	>30% reduction	Quality (Performance)
Configuration optimization	Difference in latency or throughput compared to default config	>30%	Quality (Performance)
Fault injection	Manual time required in a test cycle	>30% reduction	Quality (Reliability)
DICE IDE	Number of DICE tools used in each demonstrator	4	Productivity

Regarding **KPI-RI-7** it has already been planned that the three demonstrators will use and subsequently validate more than 80% of DICE tools (Figure 5, Figure 11 and Figure 24)

A.3. Security and privacy aspects

Data privacy, also called information privacy, is the aspect of information technology that deals with the ability an organization or individual has to determine what data in a system can be shared with third parties, how that information is used, or if that information is used to track users. Privacy concerns exist wherever personally identifiable information or other sensitive information is collected, stored, used, and finally destroyed or deleted. Privacy is not an option in Data-Intensive Applications. Indeed, it must be considered both at the infrastructure and the software levels. This stored and/or processed data is by essence a valuable data which may interest hackers especially when it is related to personal information. The challenge of data privacy is to utilize data while protecting individual's privacy preferences and their personally identifiable information. Although Big Data technologies have some built-in privacy enhancing features, we still need to deal with privacy at the design level since the Big Data ecosystem is highly evolving and new platforms are emerging and cannot provide the same level of privacy guarantees. From the point of view of a DIA designer some basic requirements must be shared at the design level such as data encryption for some databases or some nodes.

Moreover DICE will deal with security aspects primarily in terms of access control. The developer will be able through DICE models to design roles/permissions with various accreditation levels. All DICE tools will be deployed in such a way that access control lists will be automatically generated at instantiation time so to fulfil the security of the DIA designed in the DICE IDE.

DICE plans to investigate the applicability of such methods primarily in the NETF demonstrator, which has important privacy/security requirements related to the tax data.

A.4. Roadmap

This section provides an overview of the milestones that DICE demonstrators have defined for the following months. Milestones will guide the activities for the four releases that are planned for all demonstrators, M20 internal release and M24, M30 and M36 official submissions. Besides the immediate actions which are detailed in Sections B.5, C.5 and D.5, the DICE demonstrators have defined the following four milestones:

- **M20 – Architecture Validation.** This release will focus on the architecture validation. The demonstrators have designed an initial DPIM version for their case studies. This Papyrus UML model will be the central element in order to validate their DIA architecture, detect anti-pattern design and simulate the correctness of the whole system. DICE tools was and will be utilized in the future in order to validate the architecture of each application and the expected outcomes will be a validated, corrected and enhanced architecture model.
- **M24 – Quality Assurance:** In M24 the initial implementation of the three demonstrators prototypes will be released with respect to the scenarios defined in D1.2 and in the current document (Sections B.3, C.3 and D.3). Indicative activities are:
 - The demonstrators will deliver an augmented version of the validated DPIM, after using this DPIM to automatically generate the DTSM and DDSM models. These models will be annotated using the DICE Profile properties mainly related to data (source, restrictions...) and using the MARTE properties related to quality properties (MTTR, response time...). These profiled-UML models will be simulated, verified and checked by other DICE tools in order to assess and analyze each application from a quality point of view. This quality will be made at the technological and deployment levels. The expected outputs of this initial draft implementation are mainly new models with much focus on quality requirements.
- **M30 – DICE advanced features.** In M30 the demonstrators will release the second version of their prototypes. At this stage an iterative enhancement of the application quality will be performed with respect to the initial version released at M24. The demonstrators will obtain an enhanced, validated and corrected architecture which puts the quality at the center of the whole models. M30 deliverable will put in evidence the iterative quality enhancement, while each application will be continuously updated, i.e. adding new features.
- **M36 – Productivity Enhancement:** In M36 the demonstrators will release the final version of their prototypes. The final prototype will include all the results achieved in the demonstrators and will highlight the industrial impact arising from them. M36 release will put in evidence the productivity growth inherited from the automation offered by the DICE tools.



Figure 3. DICE demonstrators roadmap

DICE demonstrators will of course be releasing minor versions which will include bug fixes and new features. Their goal is to obtain, by month 30, a highly available running DIA (Sections B.5, C.5 and D.5).

B. The ATC demonstrator: NewsAsset

B.1. Introduction

NewsAsset is a commercial product positioned in the news and media domain, branded by Athens Technology Center, an SME located in Greece. The NewsAsset suite constitutes an innovative management solution for handling large volumes of information offering a complete and secure electronic environment for storage, management and delivery of sensitive information in the news production environment. The platform proposes a distributed multi-tier architecture engine for managing data storage composed by media items such as text, images, reports, articles, videos, etc.

Innovative software engineering practices, like Big Data technologies, Model-Driven Engineering (MDE) techniques, Cloud Computing processes and Service-Oriented methods have penetrated in the media domain. News agencies are already feeling the impact of the capabilities that these technologies offer (e.g. processing power, transparent distribution of information, sophisticated analytics, quick responses, etc.) facilitating the development of the next generation of products, applications and services. Especially considering interesting media and burst events which is out there in the digital world, these technologies can offer efficient processing and can provide an added value to journalists.

At the same time, heterogeneous sources like social networks, sensor networks and several other initiatives connected to the Internet are continuously feeding the world of Internet with a variety of real data at a tremendous pace: media items describing burst events, traffic speed on roads; slipperiness index for roads receiving rain or snowfall; air pollution levels by location; etc. As more of those sources are entering the digital world, journalists will be able to access data from more and more of them, aiding not only in disaster coverage, but being used in all manner of news stories. As that trend plays out, when a disaster is happening somewhere in the world, it is the social networks like Twitter, Facebook, Instagram, etc. that people are using to watch the news ecosystem and try to learn what damage is where, and what conditions exist in real-time. Many eyewitnesses will snap a disaster photo and post it, explaining what's going on. Subsequently, news agencies have realized that social-media content are becoming increasingly useful for disaster news coverage and can benefit from this future trend only if they adopt the aforementioned innovative technologies. Thus, the challenge for NewsAsset is to catch up with this evolution and provide services that can handle the developing new situation in the media industry.

B.1.1 Business goals

The following three business goals have been defined for NewsAsset use case:

1. Keep up with the demands of the news and media domain. Constantly upgrade NewsAsset to meet the challenges of nowadays
2. Incorporate new data sources and new distribution channels, like social media, sensor networks/IoT, etc.
3. Monitor what is discussed in the social media in real time and on various levels of granularity

B.1.2 Technical goals

The following technical goals have been defined for NewsAsset use case:

1. Facilitate the implementation of services that cope with high rates of data
 - a. handle efficiently features like data velocity, volume, rate of update, granularity

2. Adopt runtime scalability methods to manage temporal peaks of high computational demand (i.e. during the peaks of a bust event)
3. Realization of quality-driven processes in the Big Data domain
 - a. throughput, availability, fault tolerance, response time

B.2. Mapping DICE tools to ATC demonstrator

The NewsAsset demonstrator will validate and verify how the DICE framework can be utilized to design and deploy a Data Intensive Application that efficiently gather big streams of media data (text, images, web links, etc.) from heterogeneous sources (Social Networks, web sources, RSS feeds, etc.), processes them to relax redundancy and digest information so as to discover trends, burst events and interesting media content.

While the news item is the main entity in the news editing workflow, the upgraded version of the application to be developed in DICE wishes to enhance it in order to incorporate state-of-the-art technologies, such as the ones offered by Big Data technologies, Model-Driven Engineering (MDE) techniques, Cloud Computing processes and Service-Oriented methods. It is envisioned that NewsAsset will be able to monitor what is discussed in the social media (Twitter, Facebook etc.) in real time and on various levels of granularity. The enhancements for NewsAsset will be to monitor trending topics and events, as well as to extract other relevant information discussed in their context, such as locations etc.

The ATC demonstrator builds on of the achievements of another EU-funded project, namely SocialSensor, with the ultimate goal to modernize the existing commercial product that is branded by ATC (NewsAsset). SocialSensor deployed an open source platform capable of collecting, processing, and aggregating big streams of social media data and multimedia. However, the ATC team identified a lot of functional and non-functional quality-driven requirements that are not addressed by the current status of the platform.

Thus, the following three high level challenges (a detailed analysis can be seen in “D1.2. Requirement Specification”) will be addressed by DICE framework:

- Refactoring of the old-fashioned engine
 - related to cloud processing and Big Data technologies
- Reconfiguration
 - revise obsolete architecture with respect to quality-driven metrics
- Manage complexity
 - real-time responsiveness for temporal peaks of high computational demand

The following figure (Figure 4) provides an overview of ATC demonstrator

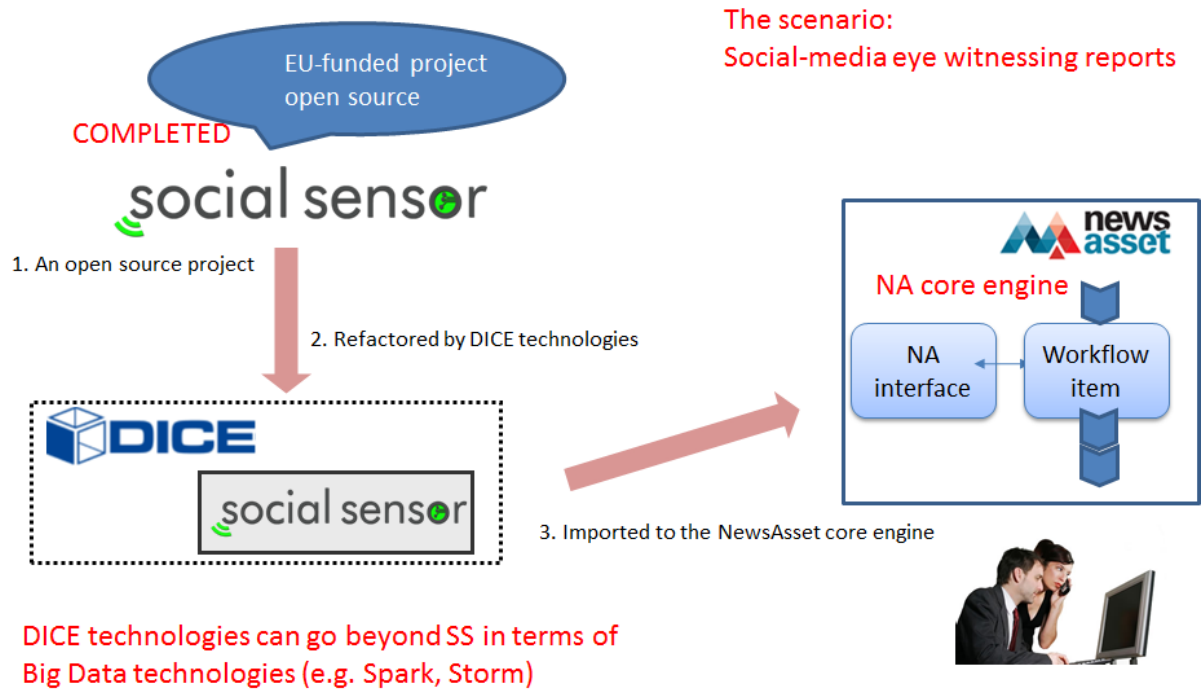


Figure 4. The ATC demonstrator.

The DICE framework will be used to achieve the aforementioned three challenges. The following figure (Figure 5) presents the DICE toolset and highlights which parts have been used so far (M1-M12 of DICE lifecycle, solid circles) and which will be used in the future (dashed circles). As it can be seen it is envisioned that almost the DICE framework as a whole will be utilized.

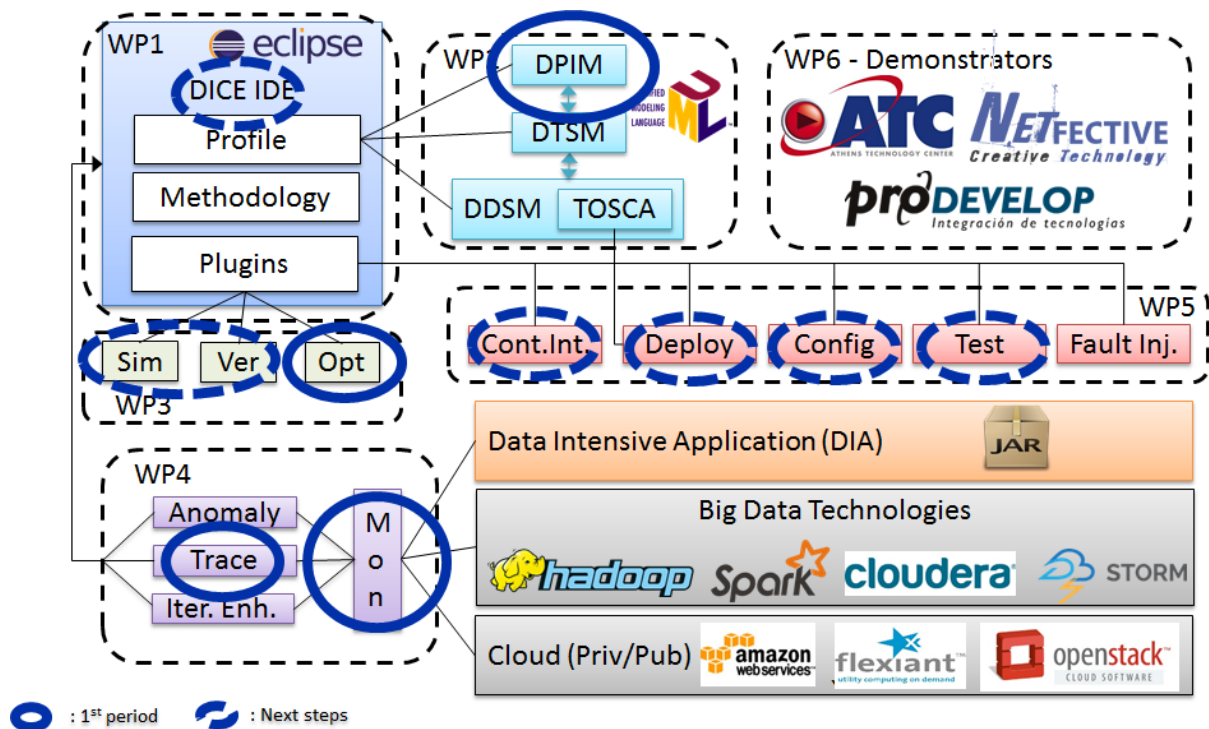


Figure 5. DICE toolset in the ATC demonstrator

B.2.1 Current status

With respect to demonstrator's challenges ("D1.2. Requirement Specification") we defined a list of requirements that are guiding ATC implementation activities. During the period M1-M12 of project's lifecycle the following have been achieved:

- ATC team analyzed the existing SocialSensor platform, run performance tests and identified quality-oriented bottlenecks and malfunctions in terms of availability, fault tolerance and performance. The part of SocialSensor's framework (Orchestrator Application creates a major bottleneck) that must be revised;
- WP4: Integration of DMon platform in operative SocialSensor environment. Due to absence of any Big Data technology in operational Social Sensor environment we focused on collecting and analyzing system resources (CPU, memory, network usage). We Identified which nodes are memory intensive and which are CPU-bound as evidences for potential code refactoring/optimization (Figure 6);
- WP2: we created a topology and component diagram of SocialSensor framework as a whole. Thus, we identified the key components and we run experiments to identify gaps and bottlenecks;
- WP2: The WP2 models showed the need to elaborate further on Storm Focused Crawler, which wasn't considered in the existing operational environment;
- WP3: Verification experiments on Storm Focused Crawler;
- WP5: Experimenting with Configuration Optimizer by utilizing it for various deployment set-ups of Storm Focused Crawler, one node Storm cluster and multi node Storm cluster;

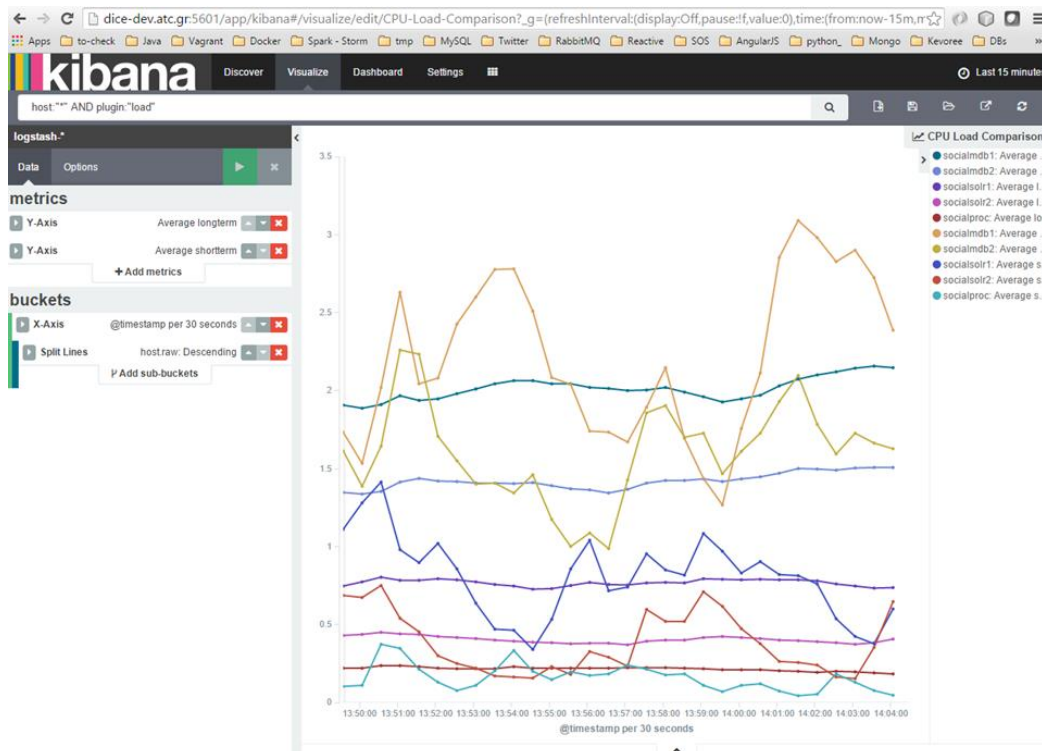


Figure 6. DMON platform operated by ATC demonstrator

B.2.2 Future plans

The following table provides an overview of DICE tools that are mapped to ATC demonstrator. We are referring to the current status of the implementation (described in details in the previous sub-section) and the future plans according to the status of the tools development. It also includes which demonstrator requirements are being addressed by a specific tool (ATC.X).

Table 2. DICE tools and ATC demonstrator

Tool	Current status	Future plans
IDE	The DICE IDE has been used mainly for designing the various models using the Papyrus plugin.	The DICE IDE will be heavily used in the design as well as the implementation process of the topic detection algorithm by creating DPIM, DTSM and DDSM models.
DICE Profile	We used DICE Profile for creating topology and component DPIM and DTSM diagram models, regarding the News Orchestrator, in order to identify any bottlenecks and better understand which parts should be re-engineered.	The same process will be performed for the topic detection algorithm.
Simulation	Not used yet	Measure the impact of different architecture alternatives based on quality driven metrics defined by the demonstrator. ATC.6
Optimization	Not used yet	Measure the impact of different architecture alternatives based on quality driven metrics defined by the demonstrator. ATC.6
Verification	Some preliminary experiments with regard to the bottleneck analysis have been carried out, related to ATC.9 requirement	The plan is to integrate the Verification Tool more systematically in the development not only of the re-engineered version of News Orchestrator but in the topic detection algorithm too.
Monitoring	DMON platform has been utilized to collect and analyze system resources (CPU, memory, network usage) of the existing News Orchestrator operational environment.	The same platform will be used to collect logs of the updated deployment and compare them with values of the old one. ATC.1, ATC.9
Anomaly detection	Not used yet	N/A
Trace checking	Not used yet	The plan is to check the consistency of the model used by the Verification Tool with the actual implementation by running trace checking on executions of the News Orchestrator application. Related to ATC.9.
Enhancement	Not used yet	The Enhancement Tool will help us with regard to the bottleneck detection in the components of the News Orchestrator application. ATC.9.

Quality testing	Not used yet	ATC plans to use the Quality Testing tool to test the real performance of our stack and also to give us a sense regarding the scalability of our deployment.
Configuration optimization	Utilized to optimize two deployment set-ups of Storm Focused Crawler (an existing News Orchestrator component), one node Storm cluster and multi node Storm cluster.	The plan is to evaluate the performance outcome by using the tool, provide feedback and involve it in News Orchestrator workflow. Related to ATC.8.
Fault injection	Not used yet	With the help of Fault Injection tool ATC wants to evaluate how the News Orchestrator application behaves when an unexpected loss of a node on our cluster happens and if the cluster is still available when part of it goes dark.
Repository	Not used yet	ATC plans to implicitly use the Repository through adoption of DICE methodology.
Delivery tool	Not used yet	We expect from DICE Delivery tool to help us improve the DevOps process in our use case development and provide us with a deployed environment to experiment on. With regard to ATC.7 we expect to be able to include in our workflow a step of evaluating progress throughout application development by visually inspecting the history of performance metrics in Jenkins CI. Regarding ATC.8 the plan is to take advantage of the WP2's model transformation tool by creating a TOSCA document usable by the Delivery tool. Should test if the models have the expressiveness that we need and if it deploys all the necessary components.

B.3. Scenarios revision – The News Orchestrator Application

The News Orchestrator Application is part of SocialSensor project which managed to develop a new framework for enabling real-time multimedia indexing and search in the Social Web. To do this, the SocialSensor project has developed and introduced the concept of Dynamic Social COntainers ([DySCOs](#)), a new layer of online multimedia content organisation with particular emphasis on the real-time, social and contextual nature of content and information consumption. Through the proposed DySCOs-centered media search, SocialSensor integrates social content mining, search and intelligent presentation in a personalized, context and network-aware way, based on aggregation and indexing of both UGC and multimedia Web content.

B.3.1 Current status

The news-orchestrator acts as the monitoring and controlling entity of the analysis and indexing phase of the workflow. Its role is to trigger in a sequential way the various modules that participate in the DySCO formulation. DySCOs can be considered as sets of social stream items related to some topic/entity of interest. It starts by synchronizing the stream manager's output to mongoDB with the analysis workflow

(Figure 8), acting as an intermediate buffer that pushes content in batches to the various analysis modules. Once the analysis modules interact to fill in the different metadata fields of items and DySCO objects, the orchestrator encodes the objects into Solr-compatible documents and feeds them into the Solr server. After this point, all DySCOs and Items are available to be retrieved and visualized at the presentation layer through the available User Interfaces.

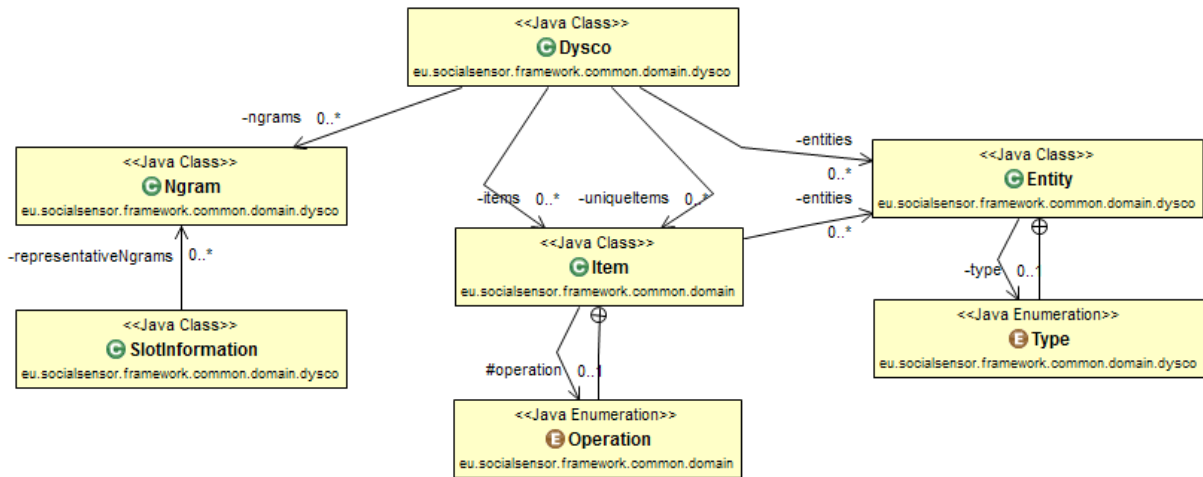


Figure 7. ATC demonstrator: the DySCO workflow

The storage layer acts as either temporary or permanent storage for all the data retrieved or generated by the News Orchestrator system. It consists of a centralized MongoDB instance and a Solr full-text-search engine. Each of these storages serves different purposes of the platform: MongoDB acts as object storage, maintaining all the generated and retrieved (enriched by the system) JSON files. Moreover, in certain cases, it plays the role of the intermediate buffer, used for synchronising the various processes and modules of the backend. Finally, the Solr server is a powerful full-text-search engine used for browsing and searching among the millions of indexed documents that reside in the system (Figure 8).

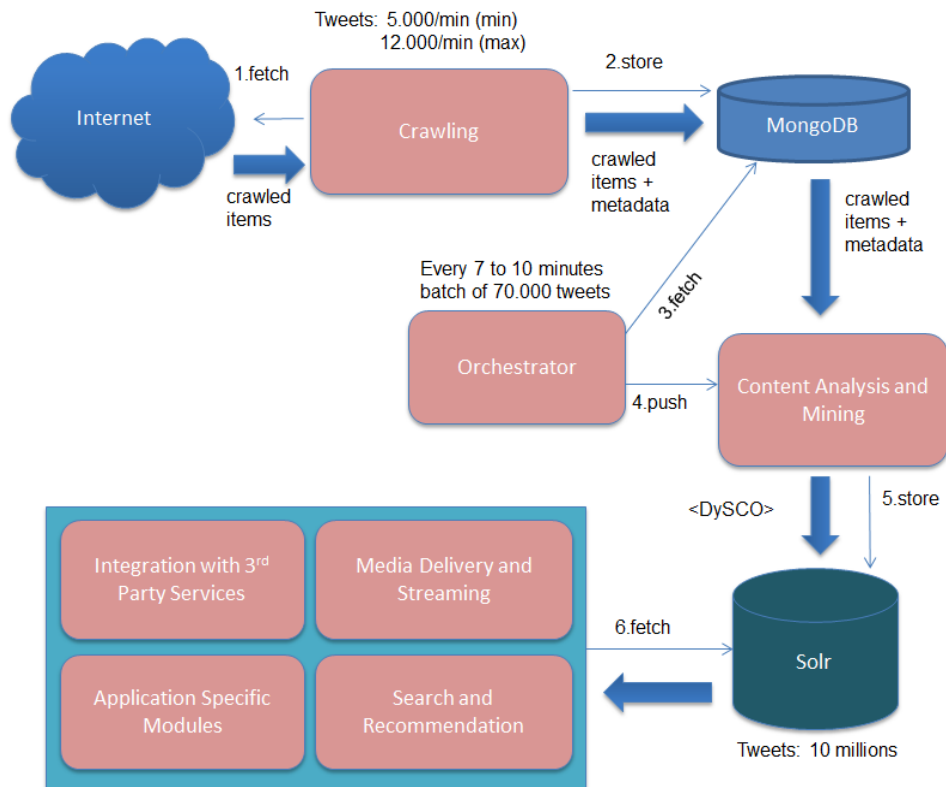


Figure 8. ATC demonstrator: The News Orchestrator Application

The backend of the system consists of several processes running continuously on a 24/7 basis as persistent loops. These processes perform a set of steps for the three following basic stages: a) retrieval of data from social networks and other sources, b) enrichment and processing of this data and c) indexing of data in visualisable and searchable format.

More specifically, the following components are operated by the news-orchestrator (Figure 9):

- **Entity Extractor:** For each incoming Item, the Entity Extractor detects references to named entities. This is based on the Stanford CoreNLP library.
- **Sentiment Analyzer:** The Sentiment Analyzer is responsible for the detection of sentiment labels (positive/neutral/negative) for each incoming Item.
- **TopicClusterer:** The DySCO clusterer clusters incoming Items based on the BN-gram method.
- **DyscoMatcher:** This matches the newly created DySCOs with DySCOs created in previous timeslots (provided their similarity exceeds a certain threshold).
- **Aggregator:** This aggregates the different elements that were extracted per Item (n-grams, keywords, named entities) on a per DySCO basis.
- **Title Extractor:** This uses a set of business rules and heuristics to extract a human readable title for each new DySCO.
- **Ranker:** This component will associate importance weights to the discovered DySCOs.
- **Query Creator:** This will be responsible for (a) forming appropriate SolrQueries that are used for the retrieval of Items, MediaItems and WebPages related to a DySCO of interest, and (b) forming appropriate queries that are used by the stream-manager to fetch (from the wrapped online social networks) additional Items and MediaItems that are related to the newly created DySCO.

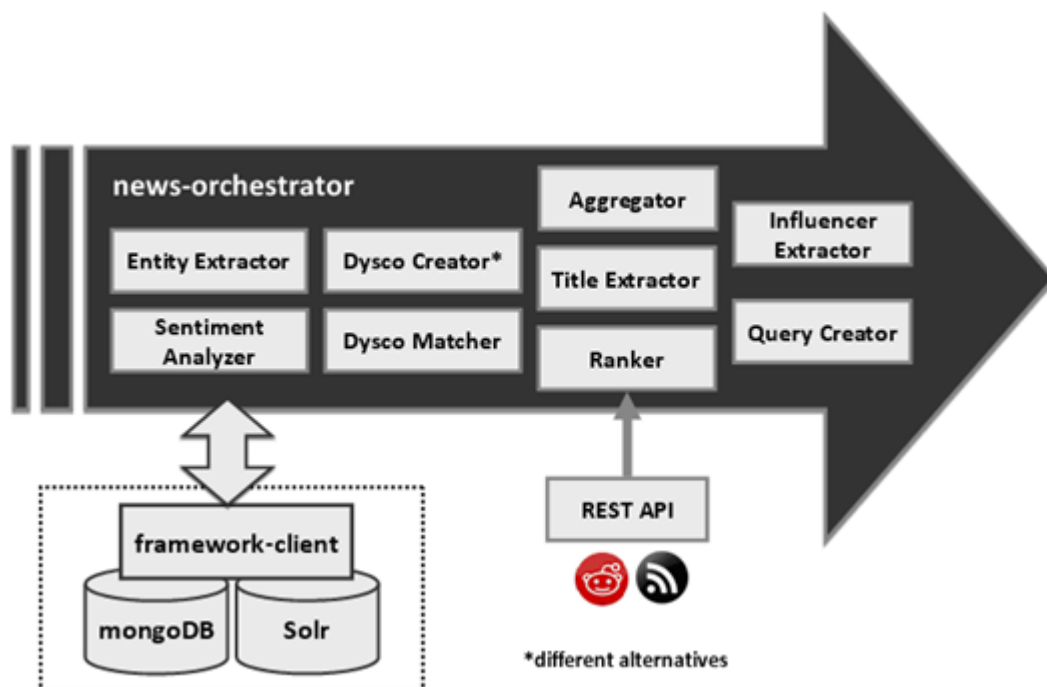


Figure 9. ATC demonstrator: The news-orchestrator components/modules

B.3.2 Proposed scenario

In the architecture described it is worth mentioning that although the News Orchestrator application deals with big streams of social networks data the use of Big Data technologies in the processing layer is quite

limited. The idea is to re-engineer the architecture and introduce Big Data technologies where this is possible. By identifying and addressing quality-driven metrics we expect to isolate bottlenecks in the architecture and revise/redesign those parts by introducing Big Data technologies. More specifically, the revised architecture should satisfy the following requirements:

- High Availability
 - The system should be stable on a 24/7 basis
- Fault tolerance
 - The system should recover automatically in case of failure without losing significant data
- Performance
 - The system should be able to scale up in terms of throughput

The time behavior of the News Orchestrator application is quite critical since the analysed information regarding the trending topics extracted should be indexed and exposed by the User Interface in almost real time, enhancing in this way the importance of the identified news topics. The idea of replacing the batch processing, that is now the basic and only approach, with stream processing would definitely affect the time accuracy of the system. The goal is to optimize the existing processing time slot of 7 minutes by means of not only minimizing the timeslot duration to reflect almost real time processing but also by maximizing the crawling capacity of the social networks crawler, giving us the potential to collect and analyze as much social networks content as possible. Another limitation that we focus on overcoming is the fact that the license of the topic detection module that is currently used, called “DyscoCreator”, does not allow for any use and modification.

In order to address the aforementioned requirements the ATC team will focus on two directions (sub-scenarios):

- **Sub - scenario 1**: parallelization of the current operational workflow of News Orchestrator application, which is now taking place sequentially for each user’s list.
- **Sub - scenario 2**: implementation of a topic detection module from scratch, utilizing most of the tools offered by the DICE tools chain.

A number of activities have been identified for both the aforementioned sub-scenarios that are presented in the following section.

B.4. Requirements update

Following the recommendations of the 1st year review report, the ATC team revised the ATC demonstrator’s requirements that were defined in the first months of project’s lifetime. In addition to, it was decided that the team will be continuously monitoring the requirements in order to first and for most guide the technical activities, secondly revise following project’s DICE objectives and vision and finally make sure that they reflect the real life industrial DICE scenarios. With respect to D.1.2. “Requirements Specification” which was submitted at M6, the following two tables include requirements ATC.8 which was revised, and a new requirements ATC.13.

<u>ID:</u>	ATC.8
<u>Title:</u>	Cloud Deployment models
<u>Priority of accomplishment:</u>	Could have
<u>Type:</u>	Requirement
<u>Description:</u>	As an ARCHITECT/DEVELOPER I want to model cloud deployment configuration to automatically deploy through generated scripts
<u>Rationale:</u>	N/A
<u>Supporting material:</u>	D6.1 (M16)
<u>Other comments:</u>	Involves TRANSFORMATION_TOOLS, DEPLOYMENT_TOOLS

<u>ID:</u>	ATC.13
<u>Title:</u>	Data availability
<u>Priority of accomplishment:</u>	Must have
<u>Type:</u>	Requirement
<u>Description:</u>	As an Architect/Developer i want to design/implement a system by eliminating any single point of failure and by ensuring at the same time that the system is resilient in a network partition case.
<u>Rationale:</u>	Data availability and network partition tolerance
<u>Supporting material:</u>	N/A
<u>Other comments:</u>	N/A

B.5. Implementation plan

In order to achieve the envisioned scenario, ATC has created a number of high-level activities to be executed with respect to the information aggregated so far. ATC team will continuously monitor the DICE tool's evolution and update the activities accordingly. This will be an on-going task.

The following figure (Figure 10) provides an overview of the short-term activities that are planned for the following months.

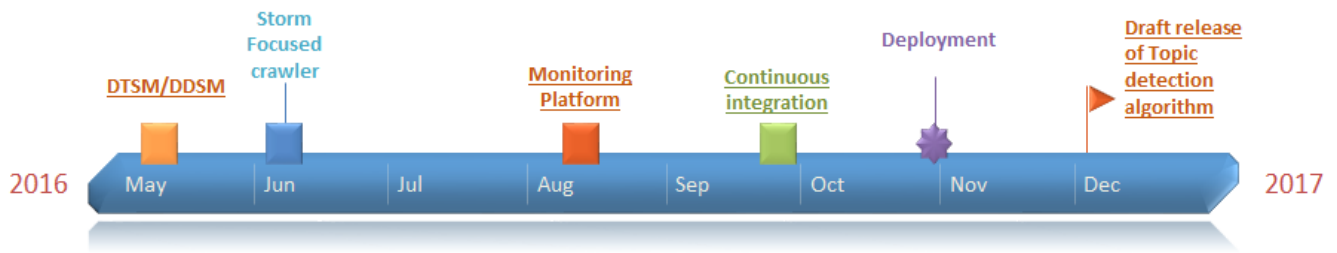


Figure 10. ATC demonstrator: Roadmap

Following the model-driven analysis of SocialSensor platform (May 2016), the ATC team identified the Orchestrator application as the most critical one to be re-factored in the DICE concept. Subsequently, the modelling analysis of the Orchestrator application revealed the need first and foremost to parallelize the process (sub-scenario 1) and secondly to implement a topic detection module from scratch (sub-scenario 2).

The first step is to optimize the topology configuration of storm-focused-crawler. In sequence, use monitoring platform to identify if the updated topology is better than the old one. Continuous integration and automated deployment of the refactored News Orchestrator application will follow (November 2016). It is envisioned to have a draft release of the new topic detection algorithm end of December and the revised by DICE tools News Orchestrator Application at the end of January 2017 (1st release).

The following tables provide a detailed view of the implementation activities defined by ATC team in the context of News Orchestrator Application scenario.

Activity/Task short name	Refactoring of the old-fashioned commercial engine (NewsAsset)
Description	SocialSensor existing operation environment (News Orchestrator Application) must be refactored to incorporate Big Data and Cloud computing technologies. Functional and non-functional challenges envisioned to be addressed by the updated engine are reported in D1.2 “Requirement Specification”.
DICE tools utilized	DICE IDE
Position in DICE methodology	DICE methodology as a whole
Part of scenario	The Orchestrator Application scenario (both sub-scenarios)
Requirements addressed	ATC.1.
Envisioned outcome	A NewsAsset plugin/service capable of pushing/importing news/media processed (not raw) data to the NewsAsset core engine
No. of releases	1
Delivery dates	M30

Activity/Task short name	Reconfiguration of SocialSensor architecture
Description	<p>Quality-driven metrics will be used to revise the obsolete architecture of News Orchestrator Application.</p> <p><u>Performance</u> Being able to process in real time more than 2000 tweets/min and 84.000 item batch each timeslot.</p> <p><u>Time behavior</u></p>

	<p>Batch Processing: optimize existing processing timeslot in terms of minimizing duration to reflect almost real-time processing maximizing the crawling capacity (number of items collected) UI - Slow response times due to Solr capacity</p> <p><u>Fault tolerance</u> Due to sequential pipeline, timeslot discarded in case of failure in initial steps no parallelization in batch processing</p> <p><u>Availability</u> Single point of failure – Data not available if a node is down (we only use sharding but not replication) No load balancing</p> <p><u>Scalability</u> real-time responsiveness for temporal peaks of high computational demand From 1 list of users to 3 lists of users From 5.000 users to 15.000 users From max 2000 tweets/min to 12000 tweets/min From 28.000 item batch each timeslot to 84.000 item batch each timeslot</p>
DICE tools utilized	DICE IDE
Position in DICE methodology	DICE methodology as a whole
Part of scenario	The Orchestrator Application scenario (both sub-scenarios)
Requirements addressed	ATC.1
Envisioned outcome	Manage the complexity of an existing DIA, handle (aggregate and process) streams of data produced by Social Networks
No. of releases	2
Delivery dates	M24-M30

Activity/Task short name	Implementing topic detection algorithm
Description	Implementing an algorithm for identifying trending topics from a corpus of social stream items in real time. The input is the content collected by the crawler that monitors the social streams (Twitter, Facebook etc).
DICE tools utilized	DICE IDE, DICE profile, Simulation/Optimization/Verification, Anomaly Detection/Trace Checking, Quality Testing, Configuration Optimization and DICE Delivery tools
Position in DICE methodology	DICE methodology as a whole
Part of scenario	The News Orchestrator Application scenario (sub-scenario 2)
Requirements addressed	ATC.1
Envisioned outcome	A topic detection module implemented with Apache Spark/Storm technologies as part of the Orchestrator application that can handle arbitrary big streams of social networks data.
No. of releases	2
Delivery dates	M23-M30

Activity/Task short name	Optimize topology configuration of storm-focused-crawler module
Description	Override default values in topology configuration with optimal values to boost performance.
DICE tools utilized	Monitoring Platform, Configuration Optimization and Configuration and Optimization tool
Position in DICE methodology	Interaction with WP4 and WP5 DICE tools
Part of scenario	The News Orchestrator Application scenario (both sub scenarios)
Requirements addressed	ATC.13
Envisioned outcome	Analysis performed by storm-focused-crawler module should be fine-tuned to reduce response time.
No. of releases	1
Delivery dates	M17

Activity/Task short name	Redesign Orchestrator application in terms of scalability
Description	The processing layer of News Orchestrator Application should be parallelized on a per user list basis. Currently the execution of processing modules is performed sequentially for all user lists thus limiting the near real time requirement (milliseconds) for trending topics extraction. Moreover, News Orchestrator Application should be able to cope with temporal peaks in the input rate of crawler.
DICE tools utilized	DICE IDE, DICE profile, Simulation/Optimization/Verification, Anomaly Detection/Trace Checking, Quality Testing and Configuration Optimization tool
Position in DICE methodology	DICE methodology as a whole
Part of scenario	The News Orchestrator Application scenario (sub-scenario 1)
Requirements addressed	ATC.1, ATC.2, ATC.9, ATC. 5, ATC.13
Envisioned outcome	A refactored version of Orchestrator application in terms of scalability and performance.
No. of releases	2
Delivery dates	M24-M30

Activity/Task short name	Enhance the testing of Orchestrator application by adopting Continuous integration/Continuous deployment practices.
Description	A build job should be created in the Jenkins CI environment to support the continuous integration of News Orchestrator Application, automatically triggered on each commit.
DICE tools utilized	Delivery Tool
Position in DICE methodology	Interaction with WP5 tools
Part of scenario	The News Orchestrator Application scenario (sub scenario 1)
Requirements addressed	ATC.6, ATC.7, ATC.8

Deliverable 6.1. Demonstrators implementation plan.

Envisioned outcome	An automatic build mechanism for continuous integration purposes.
No. of releases	2
Delivery dates	M20-M30

Activity/Task short name	Monitor Orchestrator logs through DICE Monitoring platform
Description	Feed Orchestrator application logs into DICE Monitoring platform using logstash-forwarder. This would allow developers to create graphs based on runtime performance metrics with regard to response time and crawling input rate.
DICE tools utilized	Monitoring platform
Position in DICE methodology	Interaction with WP4 tools
Part of scenario	The News Orchestrator Application scenario (both sub-scenarios)
Requirements addressed	ATC.7, ATC.8
Envisioned outcome	The ability to monitor Orchestrator application logs at runtime focusing on identifying potential bottlenecks.
No. of releases	2
Delivery dates	M19-M30

C. Prodevelop demonstrator: POSIDONIA OPERATIONS

C.1. Introduction

This section provides a detailed implementation plan of the Posidonia Operations use case.

Posidonia Operations is an Integrated Port Operation Management System highly customizable that allows a port to optimize its maritime operational activities related to the flow of vessels in the port service area, integrating all the relevant stakeholders and computer systems.

In technical terms, Posidonia Operations is a real-time and data intensive platform able to connect to AIS (Automatic Identification System), VTS (Vessel Traffic System) or radar, and automatically detect vessel operational events like port arrival, berthing, unberthing, bunkering operations, tugging, etc.

Posidonia Operations is a commercial software solution that is currently tracking maritime traffic in Spain, Italy, Portugal, Morocco and Tunisia, thus providing service to different port authorities and terminals.

Having this scenario, several business and technical goals have been identified as a result of the future application of the DICE methodology and tools to the Posidonia Operations use case.

C.1.1 Business goals

Three main business goals have been identified for the Posidonia Operations use case.

1. Lower deployment and operational costs.

Posidonia Operations is offered in two deployment and operational modes: on-premises and on a virtual private cloud.

When on-premises, having a methodology and tools to ease the deployment process will result in a shortened time to production, thus saving costs and resources.

In the case of a virtual private cloud deployment it is expected that the monitoring, analysis and iterative enhancement of our current solution, will result in better hardware requirements specifications that in the end are translated in lower operational costs..

2. Lower development costs

Posidonia Operations is defined as a “glocal” solution for maritime operations. By “glocal” we mean that it offers a global solution for maritime traffic processing and analysis that can be configured, customized and integrated according to local requirements.

In addition it is a solution that operates in real-time making tasks like testing, integration, releasing, etc. more complicated.

By the application of the DICE methodology an improvement of different phases of the development process is expected, thus resulting in shortened development lifecycles and lower development costs.

3. Better service quality policies

Several quality and performance metrics have been considered of interest for the Posidonia Operations use case.

Monitoring, predictive analysis or reliability among versions will end in an iterative enhancement of the service quality policies to our current clients.

C.1.2 Technical goals

DICE aims at providing a methodology and tools that applied to a data intensive application (DIA) result in an enhancement of the application development lifecycle. As a use case provider our main goal is the knowledge acquisition that will allow us to build a DICE-based version of Posidonia Operations and at a higher level any DIA.

Apart from this general topic, the Posidonia Operations use case technical goals can be mapped to the business goals and are summarized in these two aspects:

1. Adopt a DevOps methodology

One of the DICE core values is the DevOps approach. This emphasizes the collaboration of different stakeholders during the software development process, automating delivery of software and infrastructure and promoting an environment for frequent and reliable development lifecycle.

Posidonia Operations lacks of processes for automating deployment and infrastructures, either for production, test, development, etc. so adopting the DICE methodology will produce a technical benefit at different stages.

2. Iterative enhancement of the product development lifecycle

This technical goal is a direct outcome from some of the DICE tools when applied to the Posidonia Operations use case.

Being able to simulate and predict some functional properties for different configurations, monitor, extract and analyze performance metrics, identify bottlenecks, etc. will help to enhance at the end the Posidonia Operations development lifecycle.

C.2. Mapping DICE tools to Prodevelop demonstrator

This section aims at describing the current work done for each DICE tool regarding the Posidonia Operations use case and the future plans in order to fulfil some of the tools' and use case requirements.

As a reference, the next figure shows the general DICE architecture diagram. It shows the different artifacts that compose the DICE Profile, the DICE tools (plugins) and their relationships among them, the big data technologies of interest and the use case providers. In solid lines are represented the DICE tools that have already started to be applied to the POSIDONIA OPERATIONS use case after year 1 of the project. In dashed lines are represented the DICE tools that will be tested in the context of the POSIDONIA OPERATIONS use case.

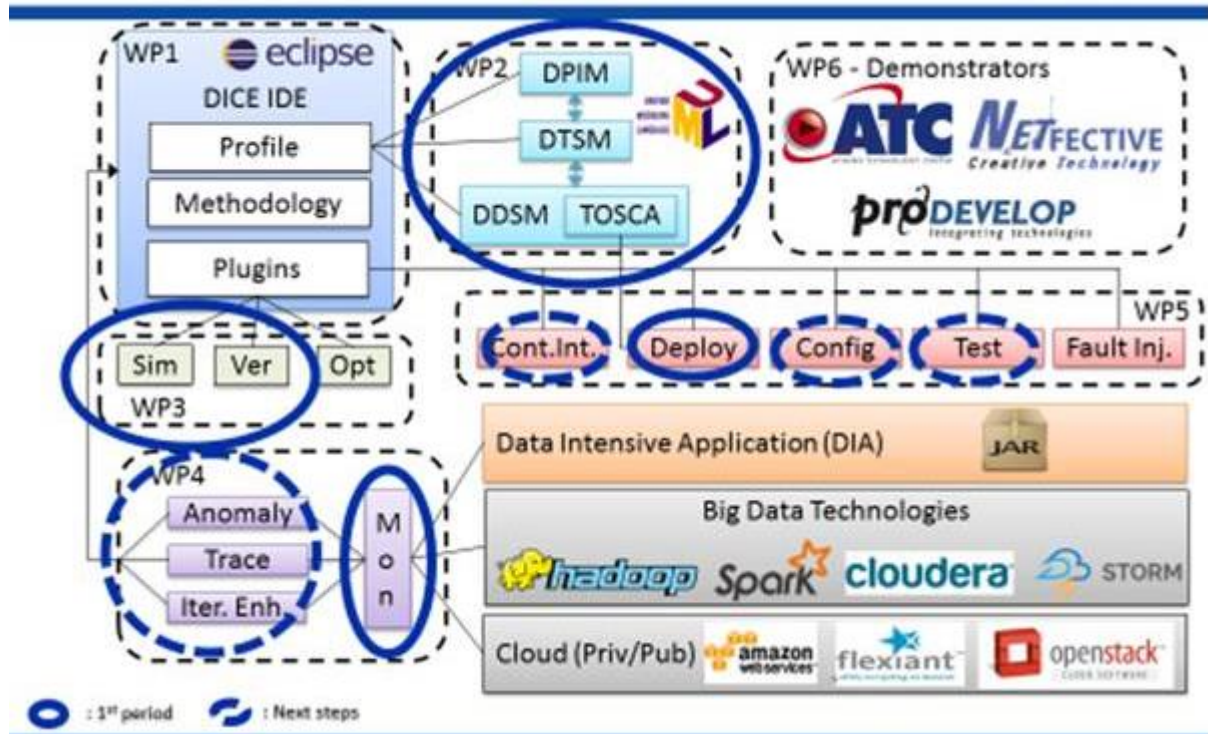


Figure 11. DICE toolset in the Prodevelop demonstrator

Next table summarizes the current status (solid circles) and future plans (dashed circles) for each DICE tool regarding the Posidonia Operations use case.

Table 3. DICE tools and Prodevelop demonstrator

Tool	Current status	Future plans
IDE	Not used yet	Test and enhance IDE to model Posidonia Operations PO.4
DICE Profile	DPIM diagram DTSM diagram Identified technologies of interest Validated Storm meta-model Storm-based model of Posidonia Operations Provided real-world deployment diagram	Test, validate and enhance models to support Posidonia Operations PO.4
Simulation	AIS parser sequence diagram AIS parser activity diagram Provided execution logs for analysis/simulation tools Provided sequence diagram for analysis/simulation tools	Test tool to support requirements (PO.1, PO.2)
Optimization	Not used yet	Collaborate to support PO.8

Verification	Provided spout and bolt parameters for verification tools	Build the POSIDONIA topology model and apply the verification tool to support PO.5
Monitoring	Focus group + questionnaire	Integrate POSIDONIA infrastructure with monitoring tool and provide access to logs (simulated or runtime) to support PO.2, PO.3, PO.12, PO.13
Anomaly detection	Not used yet	PO.13 - Anomaly tool first release is due M18
Trace checking	Not used yet	PO.13
Enhancement	Not used yet	PO.5
Quality testing	Not used yet	PO.3, PO.7, PO.9, PO.10, PO.11, PO.12, PO.13
Configuration optimization	Not used yet	Collaborate for PO.8
Fault injection	Not used yet	N/A
Repository	Not used yet	Test and validate
Delivery tool	Provided a description of our development and deployment lifecycle Focus group + questionnaire Deployment of TOSCA documents transformed from deployment diagrams (DDSM) Support for deploying Storm topologies in the FCO Continuous Integration jobs for deployment	Create deployment model, test and validate PO.4, PO.9, PO.16 <ul style="list-style-type: none"> • Enable pool of nodes deployment • Graphical representation in the Continuous Integration of performance metrics vs. versions

C.2.1 Current status

Posidonia Operations use case has focused its efforts during this period on 4 DICE tools: DICE profile, simulation, verification and delivery tools.

Regarding the DICE profile tool: DPIM, DTSM and deployment diagrams of Posidonia Operations have been done. On the other hand, several big data technologies of interest for Posidonia Operations have been identified: Storm for real time streaming processing, Cassandra and HDFS for big data storage and MapReduce or Spark for batch processing.

Since at this moment Posidonia Operations is not based on Storm, a mapping between a Storm topology model and the Posidonia Operations architecture components has been done thus producing a Storm-like model of Posidonia Operations.

Posidonia Operations' execution logs have been provided to the Simulation tool in order to their leaders start working on them. The Simulation Tool team derives some models directly from the log files since they provide timestamps for the start/end of each relevant phase in the execution. As a result, an activity and sequence diagram have been produced to have a more detailed description of the internals of Posidonia Operations streaming process.

Having mapped Posidonia Operations as a Storm-like topology model; spot and bolt parameters (those that define inputs and outputs of the topology) of Posidonia Operations streaming process have been provided to the Verification tool.

Regarding the Delivery tool, a real world deployment diagram has been provided.

Finally, several focus groups have been held among use case providers and tools' leaders in order to better map use case requirements and tools' features and extract common needs.

C.2.2 Future plans

Future plans for Posidonia Operations use case can be summarized on three iterative stages:

1. Collaborate with DICE tools' leaders to provide the inputs necessary for their tools
2. Test, validate and enhance DICE tools
3. Adapt if necessary Posidonia Operations technologies to match the selected ones for DICE tools

Our future plans contemplate the initial long list of use case requirements that will be refined in order to focus on the most profitable for both tools leaders and use case providers or those that fulfill a common need for different use cases.

At this moment, a map between every requirement and each tool has been done (see table above) and some of the necessary inputs from tools leaders have been collected in order to start working on fulfilling those initial inputs.

A dissection of future plans per tool is as follows:

IDE and DICE Profile

As soon as tools will be integrated in the IDE we will test and validate them to produce the needed models or give the inputs necessary for Posidonia Operations.

As stated in the general architecture diagram, DPIM, DTSM, DDSM and TOSCA models will be produced from the DICE Profile tool in order to benefit from the DICE tools integrated in the IDE.

Simulation and Optimization

We plan to use the Simulation tool to support two requirements that are related to simulation of the behaviour of the system for different configurations and the predictive analysis of new business rules in Posidonia Operations.

Some of these requirements overlap Simulation and Optimization tool.

Monitoring

We will work on connecting the POSIDONIA Operations infrastructure and logs with the monitoring tool in order to get performance and execution metrics along with other deployment metrics, availability, etc.

Anomaly detection, trace checking, verification and quality testing tools

We are interested in obtaining reliability metrics for different versions and configurations of Posidonia Operations, that means, given an input (a streaming dataset) ensure that the outputs are the same (the same events have been detected).

Configuration optimization, continuous integration and delivery tool

We plan to improve our delivery and continuous integration processes. We will provide as much inputs necessary to automate our current configuration, integration and deployment processes in order to adopt a DevOps approach.

C.3. Scenario revision

Posidonia Operations is an integrated port operations management system. Its mission consists on “glocally” monitor vessels’ positions in real time to improve and automatize port authorities operations.

Figure 12 shows the general architecture of Posidonia Operations. The architecture is based on independent Java processes that communicate with each other by means of a middleware layer that gives support to a Message Queue, a Publication and Subscription API and a set of Topics to exchange data among components.

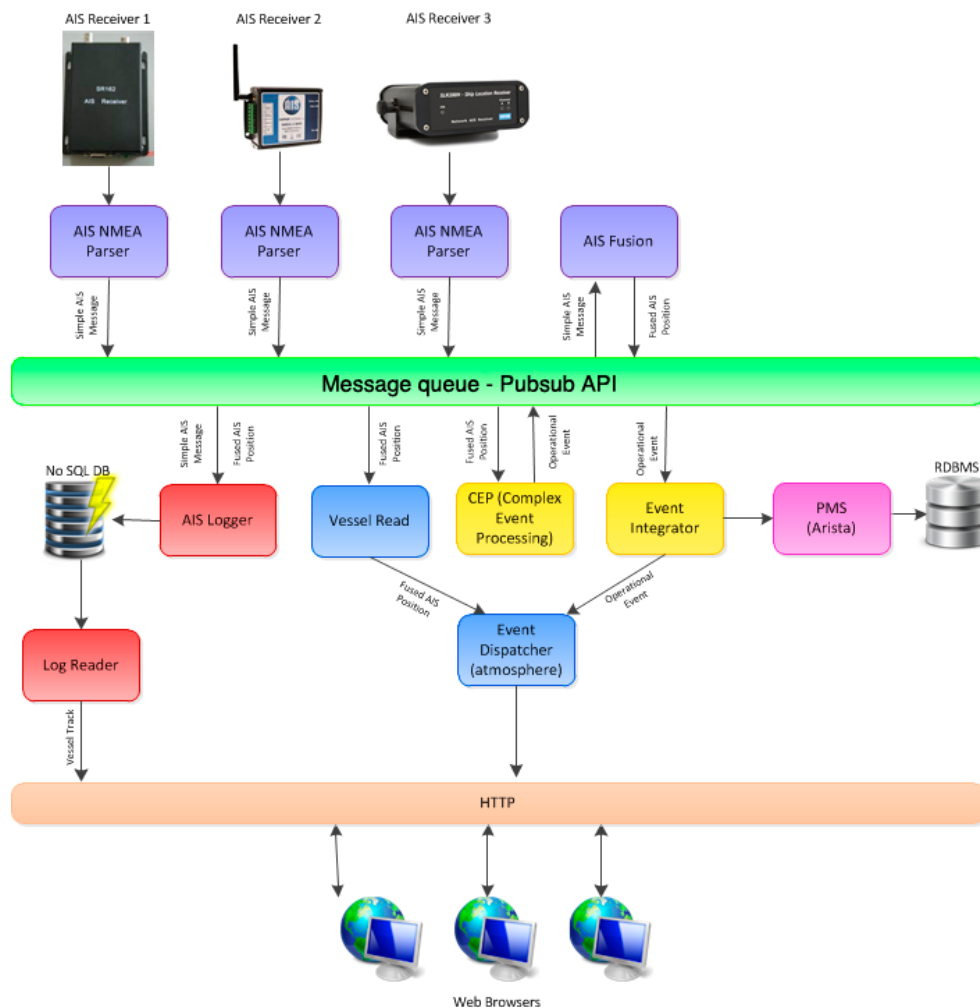


Figure 12. Prodevelop Demonstrator: Posidonia Operations general architecture

For the implementation plan we will focus our efforts on three core components where DICE tools can be of interest:

1. The stream processor or AIS Parser
2. The message broker
3. The complex event processing engine or CEP

These core components follow patterns that can be matched to some of the big data technologies that are of interest for the DICE project.

- AIS Parser: Storm spout
- CEP: Storm bolt
- Message broker (RabbitMQ): Apache Kafka

Figure 13 shows the general architecture diagram expressed as a Storm-like model in which a Spout is a streaming node, a Bolt a processing node and the RabbitMQ Exchange corresponds to the middleware layer.

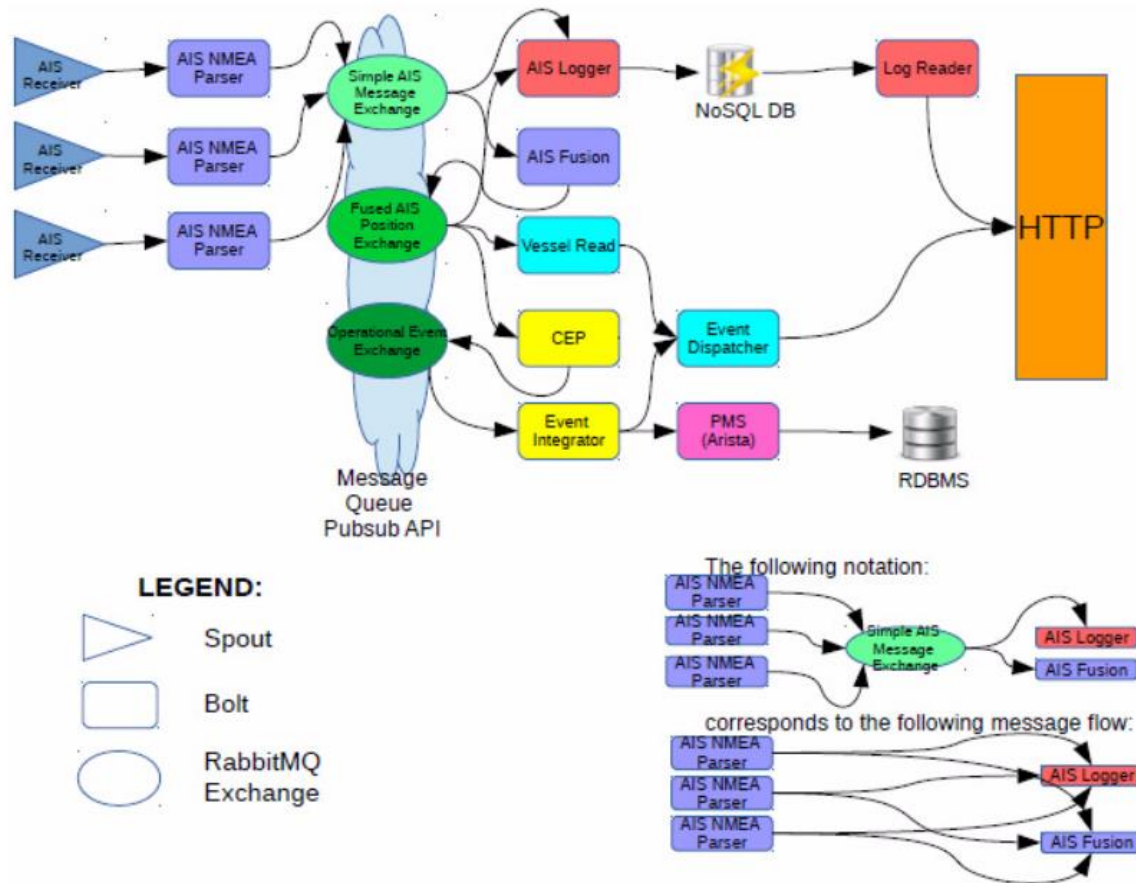


Figure 13. Prodevelop Demonstrator: A Storm – like model

A summary of the main scenario for Posidonia Operations would be:

1. Vessels in the service area of a port send AIS messages that include their location and other metadata to a central station. (This is out of the scope of the architecture diagram)
2. An AIS Receiver (a spout) receives those messages and emits them through a streaming channel (usually a TCP connection)
3. The AIS Parser (a bolt) is connected to the streaming channel, parses the AIS messages into a middleware topic and publishes it to a RabbitMQ exchange.
4. Other components (bolts) subscribe to the RabbitMQ exchange to receive messages for further processing. As an example, the Complex Event Processing engine receives AIS messages in order to detect patterns and emit events to a different RabbitMQ exchange.

Use cases

Posidonia Operations is a commercial product already in production in several port authorities and terminals in Europe.

Given the general scenario of the previous section, several use cases are offered to port operators for different purposes:

1. Traffic visualization

Description

This functionality allows port operators to visualize the traffic of a port in real time in a web application or reproduce historical situations.

How it works?

The stream of AIS messages is processed and finally sent to a web client application. The web application represents the current maritime traffic in the port service area.

On the other hand, AIS messages are logged and daily tracks of each vessel are calculated for visualizing in the web application.

Components involved

AIS Parser, RabbitMQ (middleware), CEP, AIS logger, Event dispatcher

Figure 14 shows a real screenshot of the port operator console in which he/she can visualize the current maritime traffic and list of vessels in the port's service area, operations detected and to be attended and the track of vessels.

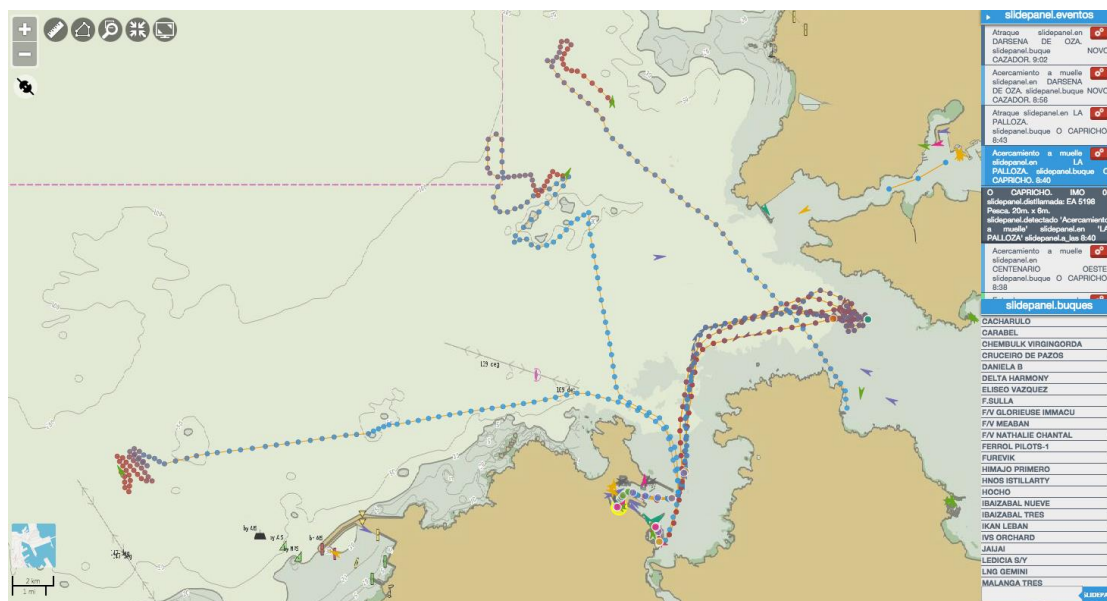


Figure 14. Prodevelop Demonstrator: The port operator console

2. Events detection

Description

Real world events related to vessels operations are detected and emitted to the message broker: Bunkering, berthing, anchorage, etc. for being integrated into the port management service.

How it works?

The complex event processing engine subscribes to the RabbitMQ AIS exchange, receives the stream of AIS messages and applies a set of rules to identify patterns that can be matched to port operations.

Components involved

AIS Parser, RabbitMQ (middleware), CEP

Figure 15 shows in green dots the arrival track of a vessel to a berth. The red circle represents the instant of detection of a berthing operation. The track in blue dots is the vessel's leaving track and the blue square represents the detection of the unberthing operation.



Figure 15. Prodevelop Demonstrator: Events detection

3. Integration with the port operations information system

Description

Real world events detected by the CEP are mapped to the operations and billing information system of a port authority to automate them.

How it works?

Once events are detected by the CEP, they are emitted to a RabbitMQ exchange. An event integrator is subscribed to that exchange to apply different business workflows that integrate the information from the real world to the port management system.

Components involved

CEP, RabbitMQ (middleware), Event Integrator

Figure 16 is a screenshot of the workflow configurator user interface. It allows users to configure different business processes that will be triggered by a particular event detected by the CEP.

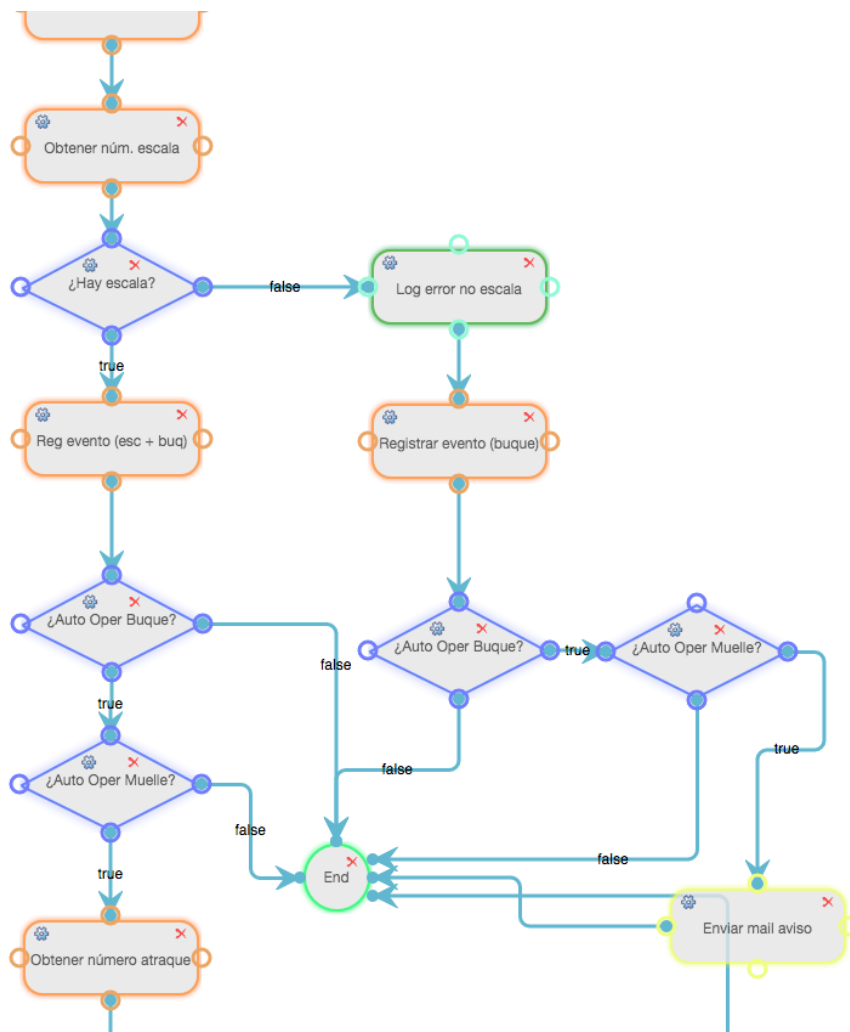


Figure 16. Prodevelop Demonstrator: Workflow configuration interface

4. Berth planning and occupation

Description

A web application to visualize the current and historical occupation of berths in order to better plan operations resources

How it works?

Deliverable 6.1. Demonstrators implementation plan.

Once AIS messages have been streamed, parsed, processed, events have been detected and integrated; different tools can be developed in order to extract insights from the maritime traffic and operations of a port authority.

One example among other is the tool depicted below for berth planning and occupation visualization. This tool provides a web interface to visualize the distribution of vessels along a berth (x axis) and time (y axis).

This tool allows port operators to have a real time and historical view of the occupation of berths, to detect empty time slots and improve future planning.



Scenarios

Once described the general scenario for Posidonia Operations, we have found different usual scenarios where our current product development lifecycle can benefit from DICE. These scenarios are based on a small subset of real world use cases and our current experience delivering a data intensive application to port authorities and terminals.

1. Deployment

Currently Posidonia Operations can be deployed in two fashions:

- On-premises: The port authority provides its own infrastructure and the platform is deployed on Linux virtual machines
- On the cloud: Posidonia Operations is also offered as a SaaS for port terminals. In this case we use the Amazon Virtual Private Cloud (VPC) to deploy an instance of Posidonia Operations that gives support to different port terminals.

Apart from this, configuration varies depending on the deployment environment:

Deployment environment	Stream speed (messages per second)	Artifacts deployed
On-premises #1	40	10
On-premises #2	5	7
On-premises #3	7	7

On the cloud #1	8	3
On the cloud #2	15	10

The Posidonia Operations deployment lifecycle presents different issues that DICE tools can help to improve:

- Hardware requirements for a deployment of Posidonia Operations (number of nodes, CPU, RAM, DISK) is based on the team experience rather than on real needs based on data. For each deployment the team knows the input data speed (messages per second) and the algorithms (rules) to be applied for each message. DICE tools can help to tune hardware requirements to deploy Posidonia Operations.
- Posidonia Operations deployment and configuration is done by a system administrator and a developer and it varies depending on the port authority. Although deployment and configuration is documented DICE tools can help to adopt a DevOps approach, where deployment and configuration can be modeled in order not only to better understand the system by different stakeholders, but also to automate some tasks.
- A DevOps approach can help to provide also test and simulation environments that will improve our development lifecycle.

2. Support vessels traffic increase for a given port

Posidonia Operations core functionality is based on analyzing a real time stream of messages that represent vessels positions to detect and emit events that occur on the real world (a berthing, an anchorage, a bunkering, etc.).

Different factors can make the marine traffic of a port increase (or decrease), namely:

- Weather conditions
- Time of the day
- Season of the year
- Current port occupation
- etc.

That means that the number of messages per second to be analyzed is variable and can affect performance and reliability of the events detected if the system is not able to process the streaming data as it arrives. When this is not possible, messages are queued and this is a situation to be avoided.

We currently have tools to increase the speed of the streaming data in order to validate the behaviour of the system in a test environment. However the process of validate and tune the system for a traffic increase is a tedious and time consuming process where DICE tools can help to improve our current solution.

3. Add new business rules (CEP rules) for different ports

Analysis of the streaming data is done by a Complex Event Processing engine. This engine can be considered as a “pattern matcher”, for each vessel position that arrives it computes different conditions, that when satisfied produce an event.

The number of rules (computation) to be applied to each message can affect on the overall performance of the system. Actually, the number and implementation of rules vary from one deployment to other.

DICE tools can help on different quality and performance metrics, simulation and predictive analysis, optimization, etc. in order to tune our current solution.

4. Give support to another port in the cloud instance of Posidonia Operations

Give support to another port (or terminal) in the cloud instance of Posidonia Operations usually means:

- An increase of the streaming speed (more messages per second)
- An increase on computation (more CEP rules executed per second)
- Deployment and configuration of new artefacts and/or nodes

In this case DICE tools can help improve Posidonia Operations also on estimating the monetary cost of introducing a new port on the cloud instance.

5. Run a simulation to validate performance and quality metrics among versions

CEP rules (business rules) evolve from one version of Posidonia Operations to another. That means that performance and quality of the overall solution could be affected by this situation among different versions.

Some examples of validations we currently do (manually):

- Performance: New version of CEP rules don't introduce a performance penalty on the system
- Performance: New version of CEP rules don't produce queues
- Reliability: New version of CEP rules provide the same output as prior version (they both detect the same events)

DICE tools can help to improve validation of performance and quality metrics.

C.4. Requirements update

With respect to D.1.2. "Requirements Specification" which was submitted at M6, the following changes have been made to the requirements related to Posidonia Operations. Requirements PO.6, PO.10, PO.14 and PO.15 were deleted.

ID:	PO.5
Title:	Bottleneck detection
Priority of accomplishment:	Could have
Type:	Requirement
Description:	As a developer I want to know the bottlenecks of my CEP rules, AIS data parsing implementation so that I can fix them for better performance
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)

<u>ID:</u>	PO.7
<u>Title:</u>	Test simulation
<u>Priority of accomplishment:</u>	Should have
<u>Type:</u>	Requirement
<u>Description:</u>	As a DEVELOPER I want to simulate my implementation with different datasets to validate the correctness of the results
<u>Supporting material:</u>	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
<u>Tools affected:</u>	Involves CI_TOOLS, MONITORING_TOOLS

<u>ID:</u>	PO.8
<u>Title:</u>	Performance impact
<u>Priority of accomplishment:</u>	Must have
<u>Type:</u>	Requirement
<u>Description:</u>	As a DEVELOPER I want to know the impact on the performance metrics when I change the implementation of a CEP business rule so that I can improve the implementation for better performance
<u>Supporting material:</u>	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
<u>Tools affected:</u>	Involves MONITORING_TOOLS, TRACE_CHECKING_TOOL

<u>ID:</u>	PO.9
<u>Title:</u>	Model continuous integration jobs
<u>Priority of accomplishment:</u>	Could have
<u>Type:</u>	Requirement
<u>Description:</u>	As a QA_ENGINEER I want to model continuous integration to automatically generate and configure continuous integration jobs

<u>Supporting material:</u>	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
<u>Tools affected:</u>	Involves DEPLOYMENT_TOOLS, CI_TOOLS

<u>ID:</u>	PO.11
<u>Title:</u>	Run simulation environments
<u>Priority of accomplishment:</u>	Must have
<u>Type:</u>	Requirement
<u>Description:</u>	As a QA_ENGINEER I want to automatically run isolated simulation environments to validate integration tests
<u>Supporting material:</u>	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
<u>Tools affected:</u>	Involves CI_TOOLS

<u>ID:</u>	PO.13
<u>Title:</u>	Reliability results comparison
<u>Priority of accomplishment:</u>	Must have
<u>Type:</u>	Requirement
<u>Description:</u>	As a QA_TESTER I want to know the reliability of the results of the system among versions testing with different datasets so I can validate the correctness of the development
<u>Supporting material:</u>	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
<u>Tools affected:</u>	Involves MONITORING_TOOLS, ANOMALY_TRACE_TOOLS, TRACE_CHECKING_TOOLS

ID:	PO.16
Title:	Deployment scripts
Priority of accomplishment:	Should have
Type:	Requirement
Description:	As an ADMINISTRATOR I want to get deployment scripts for a given cloud environment. It has to be possible to have a full POSIDONIA Operations deployment environment. This includes: VM infrastructure, middleware deployment (Java Virtual Machine, RabbitMQ, others...), application jars and configuration
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Tools affected:	Involves DEPLOYMENT_TOOLS

C.5. Implementation plan

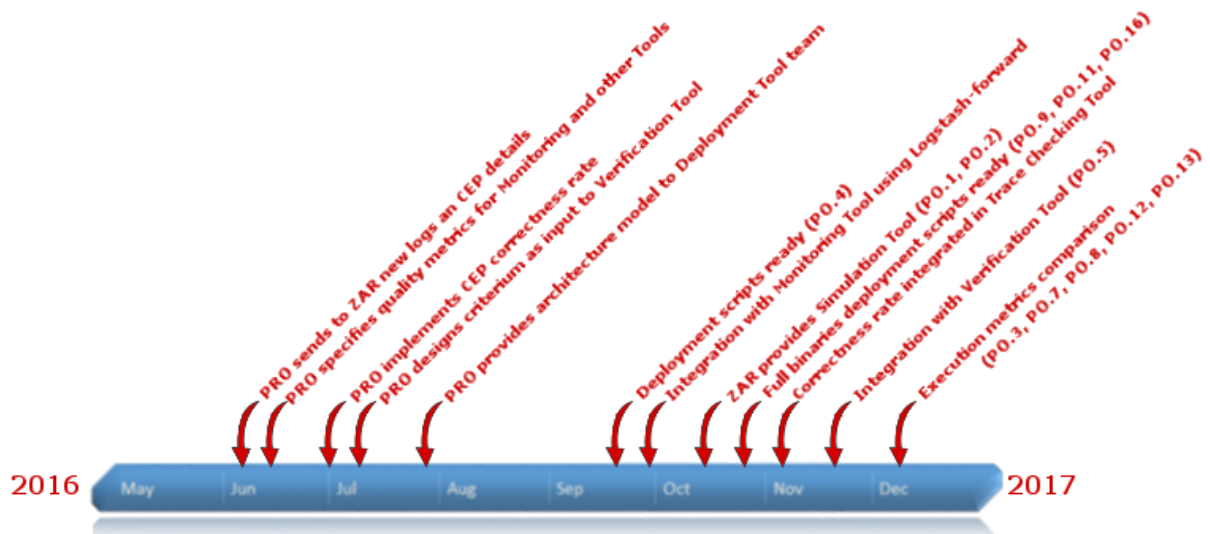


Figure 17. Prodevelop demonstrator: Roadmap

Activity/Task short name	Extracting KPIs from Posidonia Operations log files
Description	Determining which KPIs can be extracted from the POSIDONIA OPERATIONS main components (CEP and parser), mostly from the log files. Some modifications in the source code will be needed in order to print the relevant values in the logs files. Some examples of these KPI are: AIS messages processed per hour; average time needed to parse an AIS message; correctness rate (0-1 index indicating how correct the results are in terms of port events detected); cost of each rule in terms of computational time; amount of time when the component is not working (downtime); % of messages that are processed by each CEP (due to geographic filters).
DICE tools utilized	The goal of this task is being able to use the following DICE components: Simulation Tool, Monitoring Platform, Continuous Integration Tool, Verification Tool and Enhancement Tool
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Adding new business rules (CEP rules) for different ports", "Run a simulation to validate performance and quality metrics among versions", "Support vessels traffic increase for a given port"
Requirements addressed	PO.1, PO.2, PO.3, PO.5, PO.7
Envisioned outcome	A list of relevant KPI, their significance and where and how they can be computed

Activity/Task short name	Integration of log files into DICE Monitoring Platform
Description	Using logstash-forwarder or FileBeat to convert our parser/CEP log files into a format usable by the DICE Monitoring Platform.
DICE tools utilized	The goal of this task is being able to use the following DICE components: Monitoring Platform.
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Run a simulation to validate performance and quality metrics among versions"
Requirements addressed	PO.3, PO.15
Envisioned outcome	A simple converter to dump the log files into a database or format that is readable by the Monitoring Platform

Activity/Task short name	Choosing and analyzing real-world data to conduct performance test
Description	Choosing a particular port(s) and date(s) and visually determine the events and their timestamps. The resulting event lists will be used to compute the correctness rate of each CEP execution. We will probably use an example using the ports of the Balearic Islands: Palma, Ibiza, Mahón, Alcudia and La Savina, so the model will include the geographic filters applied by each CEP.
DICE tools utilized	The goal of this task is being able to use the following DICE components: Simulation Tool, Monitoring Platform, Continuous Integration Tool, Quality Testing Tool.

Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Adding new business rules (CEP rules) for different ports", "Run a simulation to validate performance and quality metrics among versions", "Support vessels traffic increase for a given port"
Requirements addressed	PO.1, PO.2, PO.3, PO.6, PO.7
Envisioned outcome	A list of ports, dates and events (including their details) that must be detected by the different Parser/CEP configurations. The comparison will result in a "correctness rate"

Activity/Task short name	Implementing correctness rate algorithm
Description	Implementing an algorithm which computes the correctness index of a CEP execution. It will take as input the CEP log file and a list of events that should have been detected. The comparison will be based on: Presence/absence of expected events in log file; Time accuracy (when the event happened); Spatial accuracy (where the event happened). Ideally, this algorithm will be implemented within the Trace Checking Tool.
DICE tools utilized	The goal of this task is being able to use the following DICE components: Simulation Tool, Monitoring Platform, Continuous Integration Tool.
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Run a simulation to validate performance and quality metrics among versions"
Requirements addressed	PO.1, PO.2, PO.3, PO.6, PO.7
Envisioned outcome	A simple Java application that computes the correctness rate after reading the CEP log file and the list of real-world events that must be detected.

Activity/Task short name	Preparing script for Continuous Integration Tool
Description	Preparing the script in the DICE Continuous Integration Tool (Jenkins-based) to execute tests and obtain performance metrics. Ideally, this process will be triggered by a commit to the source code repository and the result will be a report indicating some key performance indicators (KPIs).
DICE tools utilized	The goal of this task is being able to use the following DICE components: Continuous Integration Tool
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Deployment", "Run a simulation to validate performance and quality metrics among versions"
Requirements addressed	PO.7, PO.8, PO.9
Envisioned outcome	A Jenkins-based mechanism to easily perform test deployments which provides performance reports after each deployment.

Activity/Task short name	Describe the Posidonia Operations performance and quality metrics
Description	Several requirements aim to monitor or analyze different performance and quality metrics, but there is no clear definition of exactly which metrics are going to be measured.
DICE tools utilized	The goal of this task is being able to use the following DICE components: Simulation Tool, Monitoring Platform
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Adding new business rules (CEP rules) for different ports", "Run a simulation to validate performance and quality metrics among versions", "Support vessels traffic increase for a given port"
Requirements addressed	PO.1, PO.2, PO.3, PO.7
Envisioned outcome	In this task we will define a list of metrics to be monitored and analyzed.

Activity/Task short name	Describe the Posidonia Operations reliability properties
Description	Describe the reliability properties we want to analyze
DICE tools utilized	The goal of this task is being able to use the following DICE components: Qtesting tool, Monitoring Platform
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Run a simulation to validate performance and quality metrics among versions"
Requirements addressed	PO.13
Envisioned outcome	In this task we will define reliability properties

Activity/Task short name	Generate deployment models of Posidonia Operations
Description	Study if the current deployment configuration of Posidonia Operations fits with the deployment tools and create the deployment of Posidonia Operations
DICE tools utilized	The goal of this task is being able to use the following DICE components: Deployment tools, transformation tools, Continuous Integration tools
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "Deployment"
Requirements addressed	PO.4, PO.11, PO.14, PO.16
Envisioned outcome	A deployment model of Posidonia Operations that can be transformed in deployment recipes and hardware requirements for the given model

Deliverable 6.1. Demonstrators implementation plan.

Activity/Task short name	Connect our infrastructure with the Monitoring Tool
Description	Connect our infrastructure with the Monitoring Tool in order to extract performance, execution and system metrics
DICE tools utilized	The goal of this task is being able to use the following DICE components: Monitoring Platform.
Position in DICE methodology	This is one of the tasks needed to obtain a demonstrator. It is focused on the interaction with WP5 (DICE Deployment and Quality Testing Tools)
Part of scenario	The scenarios involved include: "All"
Requirements addressed	PO.3, PO.12, PO.13, PO.15
Envisioned outcome	Performance, execution and system metrics to monitor Posidonia Operations

D. NETF demonstrator: eGov Tax Fraud Detection - Big Blu

D.1. Introduction

Tax frauds represent a huge problem for governments, causing them a big loss of money each year. The European Union has estimated the fiscal loss lost due to tax evasion to be of the order of 1 trillion euros. In France, this loss represented approximately between 60 billion and 80 billion euros in 2015; which is huge since the state deficit is about 85 billion euros. For more than 145 governments impacted by this phenomenon—most of them suffering from repetitive economic crisis—the issue is the protection of billions of euros of revenue streams. However, when we step back to see the overall picture, we realize that tax fraud is not only about money. It is just the tip of the iceberg which hides many threats. Governments need to have a more efficient control on how money circulates, and, to a greater extent, how it is used.

Governments are increasingly using Big Data in multiple sectors to help their agencies manage their operations, and to improve the services they provide to citizens and businesses. In this case, Big Data has the potential to make tax agencies faster and more efficient. However, detecting fraud is actually a difficult task because of the high number of tax operations performed each year and the differences inherent in the way the taxes are calculated.

D.1.1 Business goals

Today the Big Data market become mature and our customers started asking NETF for prototypes. That's why the company is working to internally acquire new skills in order to release this new product dealing with tax fraud detection. NETF is also working in parallel to propose new architectures in Blu Age such as modernizing applications from legacy to Big Data targets. It's time for governments to start capitalizing on proven Big Data technologies which are already revolutionizing business efficiency across industries from healthcare to education and retail.

A survey recently made by Information Week¹ shows that the main barrier SMEs are facing about using Big Data Software is that the expertise is scarce and expensive. And NETF does not make exception! Hadoop, MapReduce and Cassandra are not point-and-click technologies. There is quite a bit of Linux configuration, some Java coding and a set of frameworks to make smoothly work together. Unless you get hands-on experience with each of those parts in a use-case context, the climb will be steep. And actually, DICE aims to relieve users of a big part of this burden by proposing a set of tools in order to facilitate the adoption of Big Data technologies and accelerate the time to market.

D.1.2 Technical goals

The appellation 'Big Data' is used to name the inability of traditional data systems to efficiently handle new datasets, which are too big (**volume**), too diverse (**variety**), arrive too fast (**velocity**), change too rapidly (**variability** and **volatility**), and/or contain too much noise (**veracity**, **validity**, and **value**).

-- ISO (Big data – Preliminary Report 2014)

The above definition of the appellation Big Data fits well our use case. It actually highlights 8 Vs related to datasets which are:

- Volume when we deal with too much data for instance millions of tax operations made daily and combined within historical information,
- Variety when data is diverse – in our use case data are coming from different sources (incomes, VAT, banks...),

¹ <http://www.informationweek.com/big-data/software-platforms/big-data-worries/d/d-id/1316857>

- Velocity when data arrive too fast – for example during tax declaration periods,
- Variability and volatility when data change too rapidly – in our case we have to deal with relocations, updated banking details, etc.
- And finally Veracity, Validity and Value when data contains too much noise and in our use case we have to filter the data and take into account for example the authorized exemptions.

Because of the complexity of our use case, we have a lot of requirements to consider while building our demonstrator. In fact:

- System must be reliable especially in terms of availability because we are targeting data store behind applications that manage a country wide taxes (1% error is still billions €),
- The Security and Privacy of data are of course fundamental for us and for our future customers,
- We also need an iterative design process which is actually available in DICE thanks to the adoption of DevOps paradigm. For instance, we are looking for quick design time and quick production feedback.

Last but not least, today governmental systems have reached the limit both technically and functionally and they absolutely need to move to fault tolerant and horizontally scalable systems.

D.2. Mapping DICE tools to NETF demonstrator

D.2.1 Current status

NETF demonstrator, aka “**Big Blu**”, is being developed from scratch, i.e., there is no existing solution to be directly used in the DICE context. We divided the technical realization of our demonstrator in 3 main parts (technical layers):

- **bGUI**: a web based application which will be the unique interface between the user and the whole system.
- **bServer**: a Web Service acting as a bridge between the user interface and the Big Data environment.
- **bBig**: the engine of the application dealing with data storage and processing.

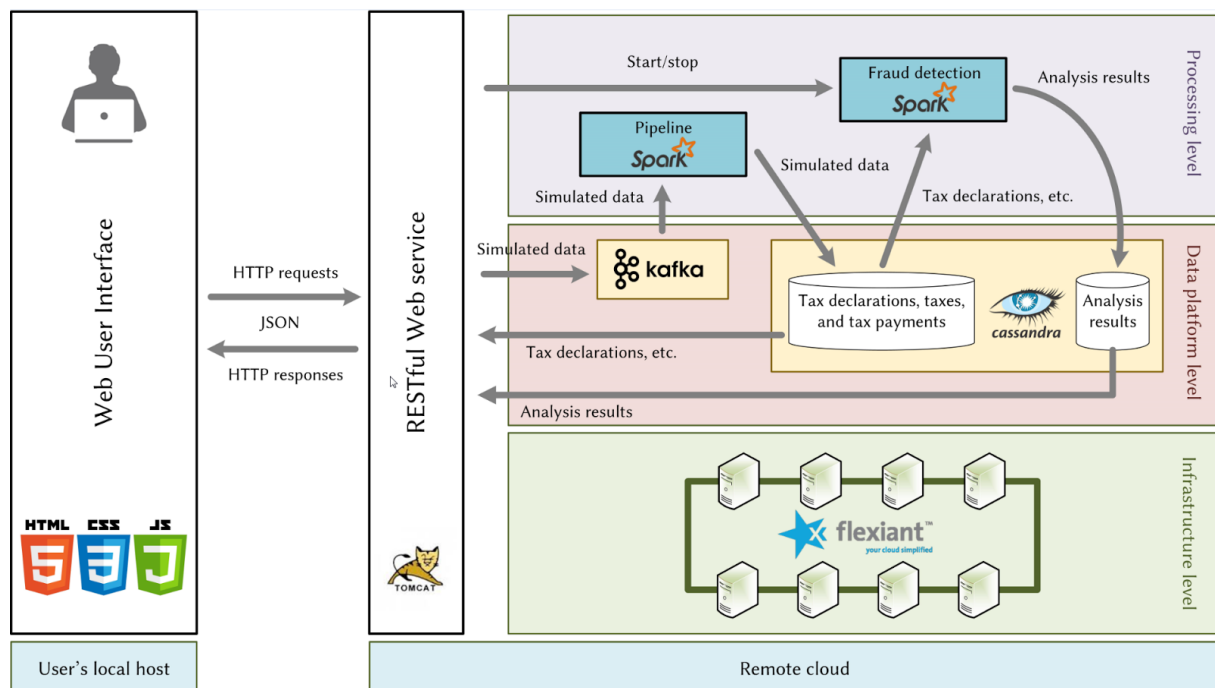


Figure 18. NETF demonstrator: General architecture of the Big Blu

Deliverable 6.1. Demonstrators implementation plan.

The implementation of our demonstrator is made using the DICE IDE (Integrated Development Environment) built on the Eclipse Platform. Indeed, we managed to get a complete IDE allowing the design and the implementation of any Big Data application using common frameworks and programming languages. It's also possible to enrich the IDE by adding any external tool or feature using the update site built-in mechanism.

bGUI - Frontend (User Interface):

Our end-users will be tax agents working in various treasury departments. In order to provide them a user-friendly tool, we started building a solution with a rich web GUI (Graphical User Interface) using HTML5, CSS and JQuery technologies. This user interface includes menus, navigation/exploration pages, configuration tools, etc. So far, we implemented some key features:

- The main pages and menus of the user interface were designed and implemented (Figure 19)
- The fraud detection page (Figure 20) is now available in order allows the user:
 - Launch Apache Spark jobs (i.e, fraud detection processings) above the Cassandra Cluster. So far, the jobs are simply making trivial requests to the database and we are working on more advanced algorithms dealing with concrete identified fraud indicators,
 - Obtain details about the jobs' status (running, killed, finished),
 - Visualize the results of each detection job. Results are presented as a list of potential fraudsters extracted from the database.

We have identified a plethora of features with high added-values which will be added in future release iterations.

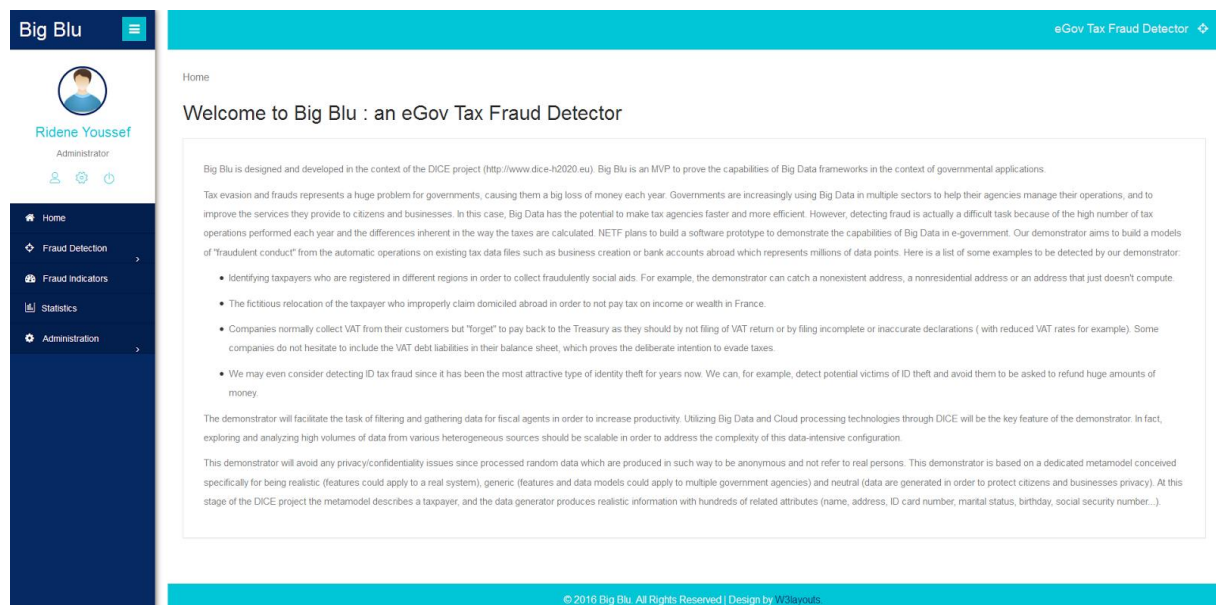
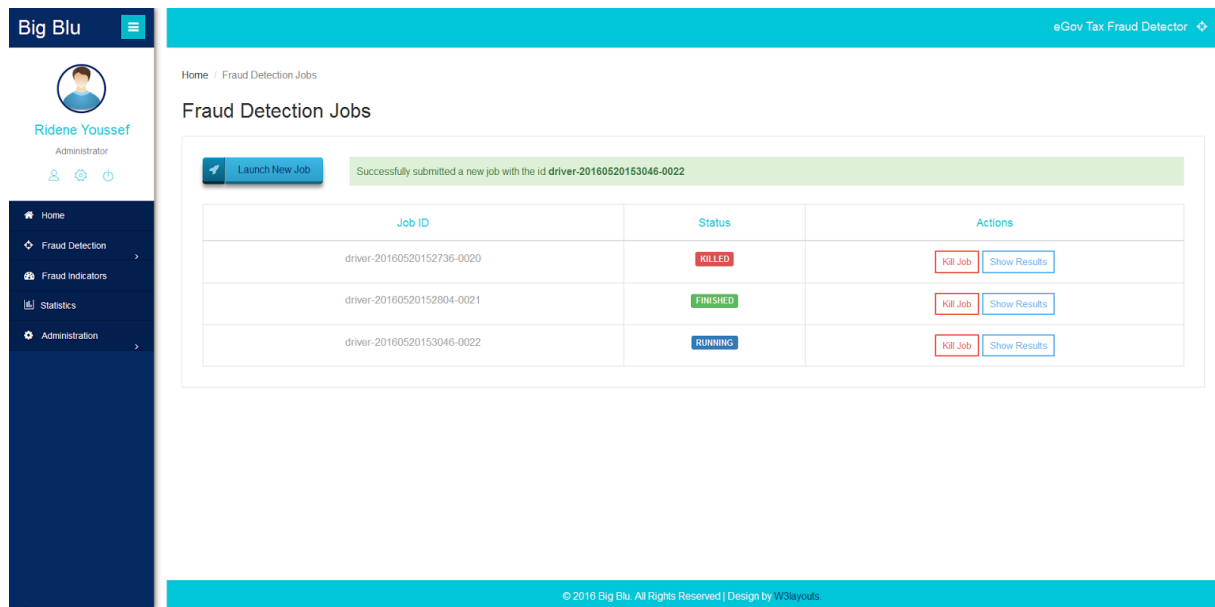


Figure 19. NETF demonstrator: Homepage



The screenshot shows the 'Big Blu' interface for 'eGov Tax Fraud Detector'. The user is 'Ridene Youssef', an Administrator. The page title is 'Fraud Detection Jobs'. A green notification bar states: 'Successfully submitted a new job with the id driver-20160520153046-0022'. Below this is a table with columns 'Job ID', 'Status', and 'Actions'.

Job ID	Status	Actions
driver-20160520152736-0020	KILLED	Kill Job Show Results
driver-20160520152804-0021	FINISHED	Kill Job Show Results
driver-20160520153046-0022	RUNNING	Kill Job Show Results

© 2016 Big Blu. All Rights Reserved | Design by W3layouts

Figure 20. NETF demonstrator: Fraud detection list (status and results)

bServer - Web Service (Front-End <-> Back-End):

The glue between the user interface (bGUI) and the Big Data application (bBig) is made thanks to a web service based on JSON. This web service is deployed on a Tomcat server running on Flexiant's testbed. This technical component is mainly made of custom implementation of RESTful HTTP verbs (POST, DELETE...). This service is enriched according to the new implementations made mainly at the level of the back-end.

bBig - Back-End (Databases and Data Processing Units):

This part includes the biggest and most important part of the demonstrator. In fact, it includes the core feature of the tool which is in charge of the data processing in order to detect fraudulent conducts. In order to avoid any privacy and/or confidentiality issue, we decided to process imaginary but realistic data. Thus, we implemented a piece of software we called "Taxpayers Random Generator Module" which is able to generate, according to our needs, information describing millions of taxpayers (Figure 21).

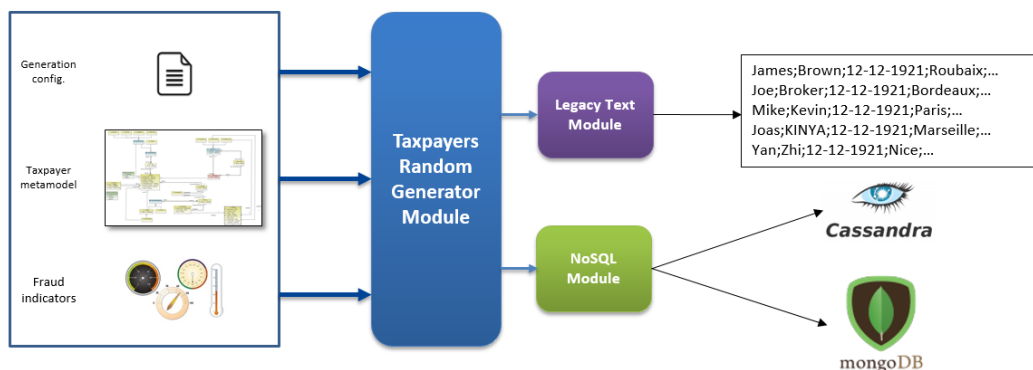


Figure 21. NETF demonstrator: Taxpayers generation

The taxpayer's random generator module produces realistic information using 3 main inputs:

- The first input is a "Generation configuration File" which contains various kinds of parameters such as the number of taxpayers, the percentage of single or married taxpayers, the data, encoding,

data structure, schema, database locations, etc. This file will be replaced later in order to be managed using the user interface.

- The second input is the taxpayer metamodel and its underlying constructs (discussed later).
- The last input for this module is a model of fraud indicators which is actually a list with known fraudulent behaviours (for example: a huge change in incomes compared to last years). This fraud indicators model will be extensible with any new identified fraud patterns.

Based on these 3 inputs, the **Taxpayers Random Generator Module** is able to generate millions of inputs (i.e., taxpayers) to fill our Cassandra cluster. The generated data are the “raw material“ for the whole demonstrator. The latter is based on a dedicated metamodel (Figure 22) conceived specifically for being:

- realistic (features could apply to a real system),
- generic (features and data models could apply to multiple government agencies),
- neutral (data are generated in order to protect the privacy of citizens and businesses).

At this stage of the DICE project the metamodel describes a taxpayer, and the data generator produces realistic information with hundreds of related attributes (name, address, ID card number, marital status, birthday, social security number...).

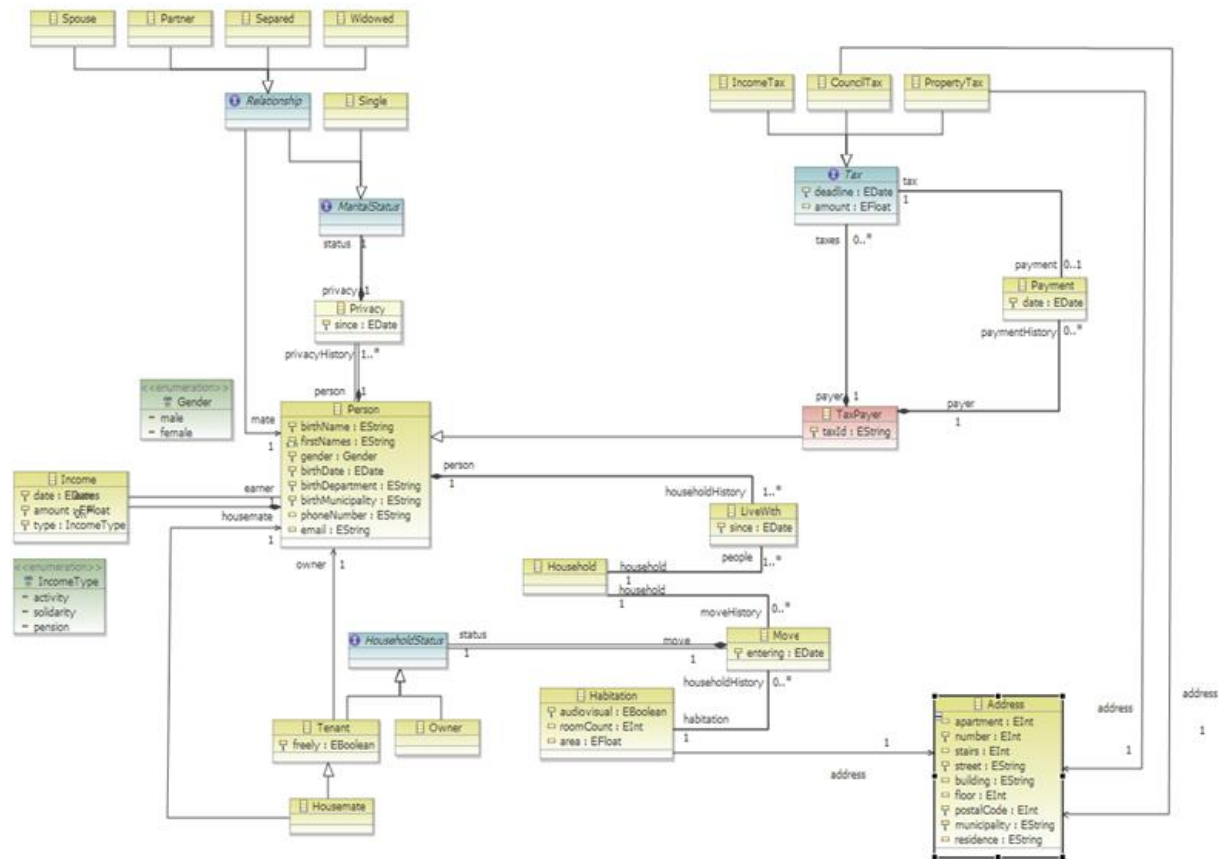


Figure 22. NETF demonstrator: Taxpayers metamodel

For our use case, we opted for the Lambda Architecture (already supported by the DICE tools) which is actually a reference architecture for Batch and Real-Time processing in the Big Data ecosystem. This architecture relies on a Data Input Stream which is in our case a Cassandra Cluster filled with various kind of information related to taxpayers. We started designing and implementing our use case dealing with tax fraud detection. So far,

- We designed our Data-Intensive Application architecture.
- The technologies to be used have been identified (Cassandra, Spark, Kafka...).

- The technical infrastructure (clusters, frameworks...) has been installed and configured on Flexiant's testbed. This basic configuration will be of course fine-tuned later using the DICE tools in order to get an optimal one for our specific needs. In fact, most requirements related to deployment have already been fulfilled and this allows the definition of deployment-specific views of the architecture containing technical infrastructure details.
- We have advanced prototypes both of the graphical user interface and the back-end solutions.
- The data generator is already implemented and tested.
- NETF built a complete DPIM (DICE Platform-Independent Model) using the beta version of the DICE IDE and Profile. In fact, DICE requirement R2.10 has already been fulfilled and allows the definition of a DIA architecture at DPIM and DTSM::Core level (i.e., with properties specification).

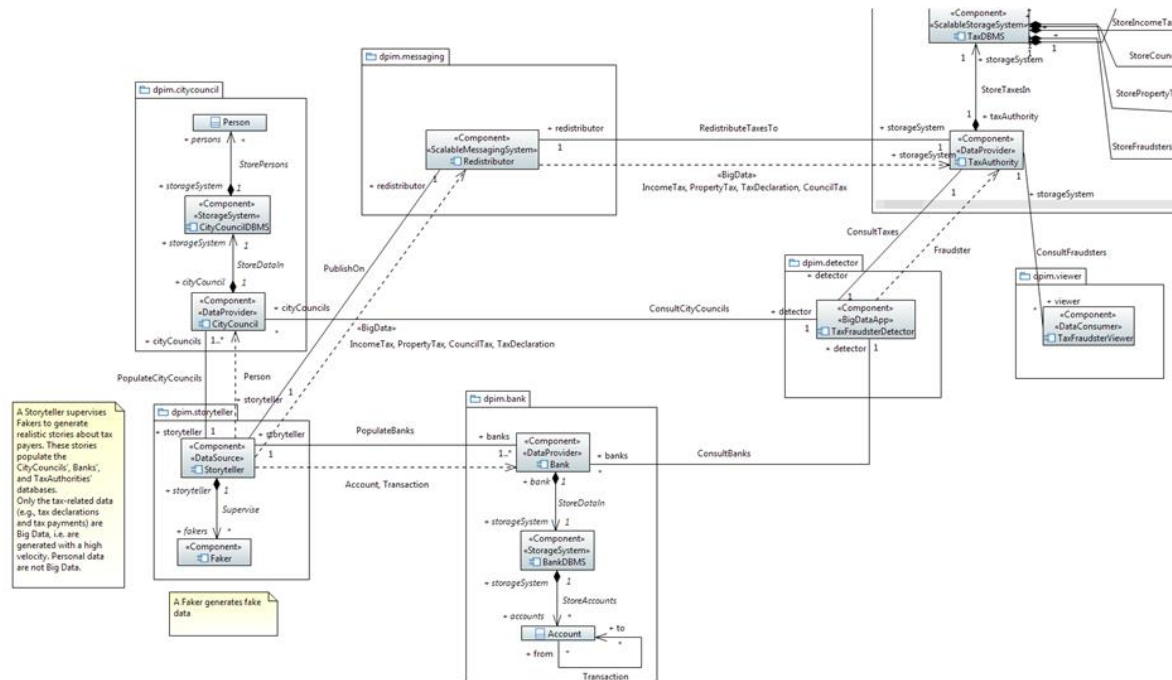


Figure 23. NETF demonstrator: Part of our DPIM model built using the DICE profile

At this stage of the project, we have designed and deployed the minimum required building blocks of our demonstrator. This version will be used as the basis of future planned versions to be enhanced and optimized using the DICE ecosystem. In fact, in order to be able to use the DICE tools, we need a running prototype (a minimal valuable product, MVP) using the chosen Big Data frameworks. So far, we have successfully designed, implemented and deployed such a MVP which includes the whole chain. This demonstrator will be now optimized and enhanced using the DICE tools especially for some already identified issues (time-consuming tasks such as the packaging and deployment, complex infrastructure configuration...). The DICE tools will be using the produced DPIM for early stage assessments, optimization and configuration enhancement.

D.2.2 Future plans

During the initial phase of the prototype implementation, NETF was as any other novice SME in the field of data-intensive applications. In fact, this prototype is our first Big Data application using such frameworks and paradigms. Getting hands-on experience highlights concrete and actual issues which must be addressed by the DICE tools. Furthermore, we have added-value inputs to fine-tune the DICE methodology in order to capitalize on our learned lessons. We will continue this incremental work in order to lead 3 parallel activities:

1. **UCN-1:** Enhance the DICE methodology in order to get it as simple and complete as possible. In fact, we are convinced that the methodology, using the DICE tools, can facilitate the life of any developer aiming to a rapid prototyping of a Data-Intensive Application.
2. **UCN-2:** Use, test and give feedbacks to tool providers is one of our priorities as a use case provider in the DICE project. Like a beta-tester of the DICE ecosystem, NETF wants to support our partners (tools providers) to release first-class useful products.
3. **UCN-3:** Add new features to our demonstrator and push the system to its limits to get real execution situations. We will keep iterating internally in NETF in order to get feedbacks from our stakeholders and especially partners interested in the project. In fact, we have planned some demonstrations sessions to occur before the end of this year.

In order to achieve the aforementioned tasks, we have adopted an incremental development strategy based on a feature-based approach. In fact, we release a new version each week by adding a new use case scenario. The technical underlying implementations and configurations are made accordingly.

Task UCN-2 is one of our priorities in which we have already realized meaningful achievements. We can see in that we have already used some of the DICE available tools and we plan to use, in near future, other tools.

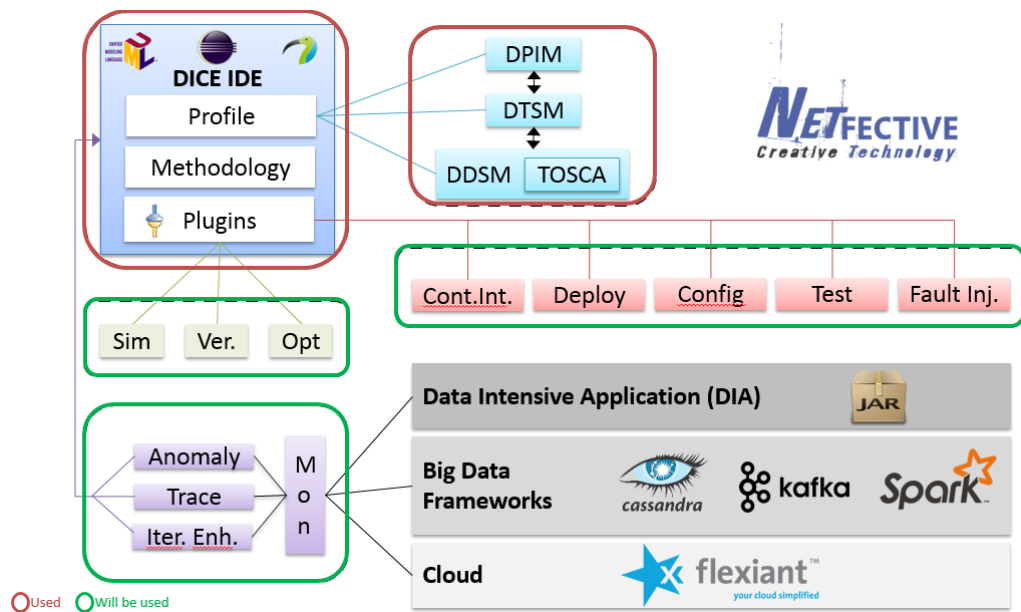


Figure 24. DICE toolset in the NETF demonstrator

The table below gives more details on our interaction with the DICE ecosystem.

Table 4. DICE tools and NETF demonstrator

Tool	Current status	Future plans
IDE	We are using the DICE IDE in order to develop our demonstrator using the development features inherited from the Eclipse environment (Java, Scala, Web Services...)	We will keep using the DICE IDE for the whole project development. We have identified a list of interesting features which must be included in order to rise the level of automation and relieve the developer from some technical tuning, e.g. creating a DICE

	and also to design our models using Papyrus.	project with already filled DPIM, DTSM and DDSM using Papyrus, etc.
DICE Profile	<p>We produced a DPIM diagram using the beta version of the DICE Profile. This DPIM was shared with our partners (especially tool providers) mainly to start discussing about the scope of the concepts we must deal with and get early-stage verification and optimization regarding our primary architecture.</p> <p>This DPIM is also used in order to identify how privacy and security must be managed by the DICE Profile and especially at which level of abstraction.</p>	We are now looking for feedbacks from our partners in order to enhance this DPIM and especially extract valuable outcomes in terms of architecture refining, suitable tools mapping, bottleneck identification, etc.
Simulation	Not used yet	The simulation tool will be an excellent rapid way to test the application without spending time in clusters configuration. We will be able to validate our architecture at early-stage at very low cost in terms of time spent in implementation and operations.
Optimization	Not used yet	We have already identified issues with the hand-made clusters configuration (memory leaks, performance degradation...). We will be using the optimization tool in order to automatically get the best (most suitable) cluster configuration for our Data-Intensive Applications. We are mainly interested in Spark and Cassandra.
Verification	Not used yet	N/A
Monitoring	The DICE Monitoring Platform (DMon) is for sure the cornerstone of DICE. Without relevant information regarding the in-situ environment (running Data-Intensive Application in situation), we won't be able to concretely enhance, change, optimize. We need to get realtime details on how things happen especially to react accordingly.	We started working with our partners from IeAT in order to be autonomous regarding the use of the tool. We started configuring the environment on our cluster and we planned to use the DMon provided API in order to integrate the relevant information in our demonstrator. This feature will be an added-value addons for our end-users who have not to deal with monitoring in general but will be for sure interested to know how their environment (including only their tools) is behaving.

Anomaly detection	Not used yet	Our use case will be constantly evolving especially in terms of fraud detection algorithms which must be adapted to several changes (laws, exemptions...). This results in continuous delivery of new versions of the whole system which must be of course reliable and operates at a high level in terms of performance and correctness. This requires to perform statistical analysis to compare monitoring data across versions which exactly the goal of the anomaly detection tool.
Trace checking	Not used yet	While implementing our use case, we have rapidly seen that the debugging of a Big Data application can be a nightmare. As a developer, you have no choice except logging and then read and extract valuable details among the tremendous quantity of information. The trace checking tool will do it for us automatically and verify the application behaviour.
Enhancement	Not used yet	Building Big Data applications is, without doubt, a complicated exercise during which the developer can involuntary insert anti-patterns which may highly impact the application's performance. Identifying such anti-patterns and proposing concrete enhancement strategies mainly in terms of runtime monitoring measurements is welcome to ensure a high quality application
Quality testing	Not used yet	For our classical Java applications, we have been used to automatically run JMeter in order to load test functional behavior and measure performance. For our Data-Intensive Application, JMeter will not be sufficient. We absolutely need tools which cover all the frameworks we are using. This will be definitely the role of the quality testing tool
Configuration optimization	Not used yet	While testing our prototype we spent a huge time setting-up a default configuration environments for Cassandra, Spark, etc. They are, for sure, not point-and-click technologies and these vendor-provided configurations are not the optimal ones which can be made in order to make the best use of these frameworks. This is exactly what we are expecting from this tool, i.e., an optimal configuration at the cost of some computation and time in order obtain the best configuration in terms of performance without having to rely on vendor-provided defaults or (rare and expensive) experts to tweak them.

Fault injection	Not used yet	Today we cannot talk about Big Data, Cloud computing, Data-Intensive Applications... if the loss of a node in the cluster or a high memory usage or any other “abnormal” infrastructure behavior, will lead to a whole system breakdown. System disruption due to faults is not an option, our application must be resilient and must be able to recover and continue to work correctly. This latter will be tested using the fault injection tool.
Repository	Not used yet	The repository will be used by tools that we absolutely need for our demonstrator (deployment...).
Delivery tool	Not used yet	We are convinced that this tool will change our lives since we are spending a huge time on manual delivery. When the developer needs to debug his application, he has to package, copy, compile and deliver the application tens of times per day. So if he can only push a button to get all of this done, he will for sure get a huge productivity growth.

The DICE tools will be used in order to release our application with a high level of quality. As explained earlier, we will be intensively changing, enhancing, enriching and modifying our prototype in order to release rapidly new augmented versions. We will heavily rely on the DICE ecosystem in order to be able to proceed in an agile-DevOps approach without taking unmeasured risks regarding the reliability and the performance of our application.

D.3. Scenarios revision

NETF plans to build a software prototype to demonstrate the capabilities of Big Data in e-government. Our demonstrator aims to build a model of "fraudulent conduct" from the automatic operations on existing tax data files such as business creation or bank accounts abroad which represent millions of data points.

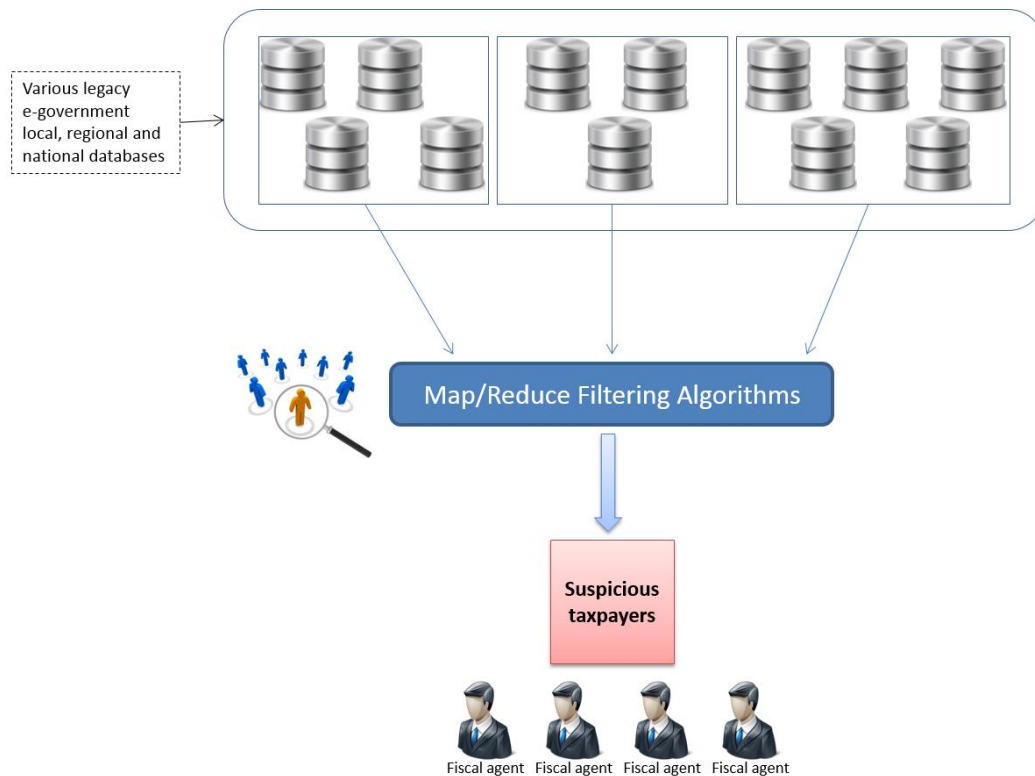


Figure 25. NETF demonstrator: Overview

Here is a list of some examples to be detected by our demonstrator:

- Identifying taxpayers who are registered in different regions in order to collect fraudulently social aids (Figure 26). For example, the demonstrator can catch a non-existent address, a non-residential address or an address that just does not compute.

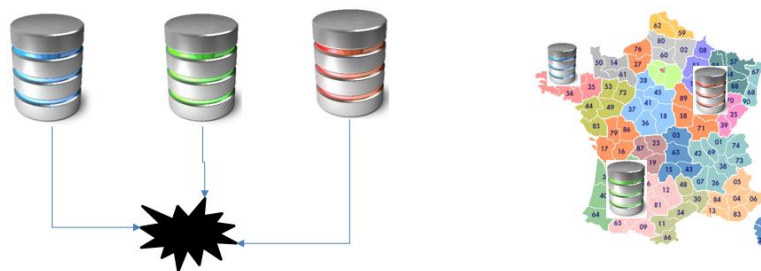


Figure 26. NETF demonstrator: Social aid fraud

- The fictitious relocation of the taxpayer who improperly claims to be domiciled abroad in order to not pay tax on income or wealth in France (Figure 27).

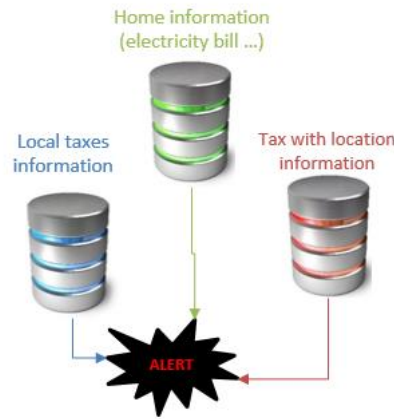


Figure 27. NETF demonstrator: Relocation fraud

- Companies normally collect VAT from their customers but "forget" to pay back to the Treasury as they should by not filing for VAT return or by filing incomplete or inaccurate declarations (with reduced VAT rates for example). Some companies do not hesitate to include the VAT debt liabilities in their balance sheet, which proves the deliberate intention to evade taxes.
- We may even consider detecting ID tax fraud since it has been the most attractive type of identity theft for years now. We can, for example, detect potential victims of ID theft and avoid them to be asked to refund huge amounts of money.

The demonstrator will facilitate the task of filtering and gathering data for fiscal agents in order to increase productivity. Utilizing Big Data and Cloud processing technologies through DICE will be the key feature of the demonstrator. In fact, exploring and analyzing high volumes of data from various heterogeneous sources should be scalable in order to address the complexity of this data-intensive configuration.

This demonstrator will process data record instances (millions) complying with the aforementioned metamodel. All instances will be automatically generated using various algorithms. Each algorithm will generate specific textual datasets using heterogeneous formats (flat files, legacy proprietary databases, etc.). Therefore it is expected that confidentiality constraints will not be considered. Furthermore data will be distributed both in spatial and temporal manners and will be heterogeneous:

- **Spatial:** data coming from different data stores (both physical storage and data access technology).
- **Temporal:** new data input must be related to either oldest data or results of processed data, i.e., incremental map/reduce.
- **Heterogeneous:** different data structure must be correlated (RDBMS schema vs sequential indexed files).

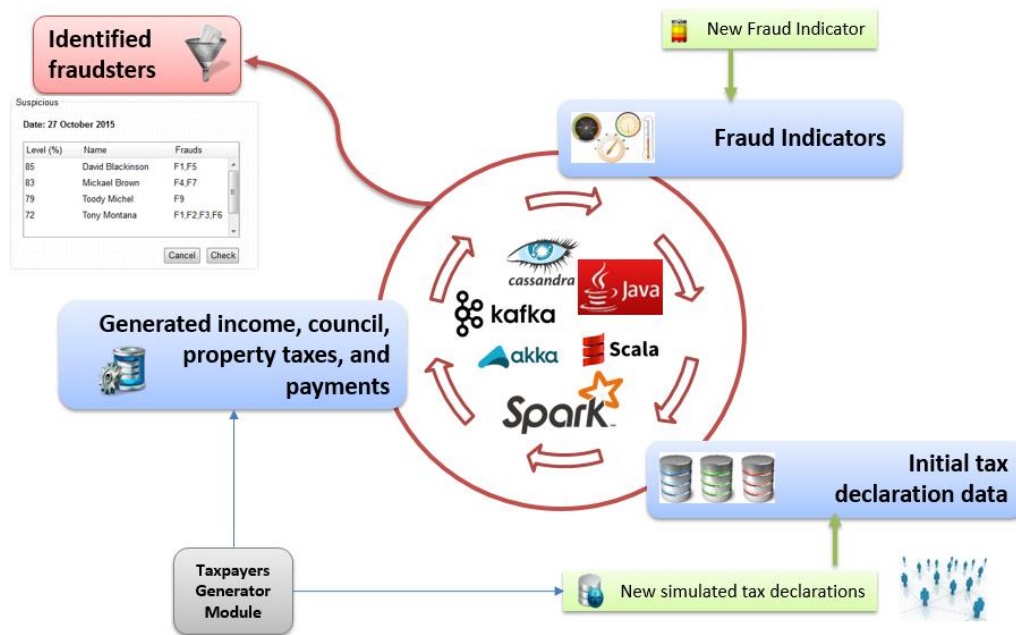


Figure 28. NETF demonstrator: Detailed architecture

We deliberately want to avoid “classical” batch processing, we need real time processing on already processed data (Figure 28). For example, when a new detail about a taxpayer is updated, we need to process it using already processed data. A kind of real time processing which use the processed data as an input for the filtering algorithms. This functional need is the core concept behind the Lambda Architecture we adopted. Our architecture is made of several layers:

- A Cassandra cluster to store data,
- A Kafka Cluster which is a scalable messaging system to handle massive amounts of data published with a high flow rate.
- An Akka Cluster to exploit the actors programming paradigm in order to simplify the implementation.
- A spark Core for the data processing which is the central element of our use case.
- And finally a Machine Learning module which aims to propose to the end-users (tax agents) new potential fraud indicators in order to anticipate frauds.

Our Big Data application made of these technologies will be continuously running on Flexiant cloud. The Cassandra databases will be filled with taxpayers detail, historical tax declarations, etc. The application will be performing computation on all data including new generated inputs. These data have to be processed using Fraud indicators. In the case a new Fraud Indicator, we have to proceed to a new batch processing phase on all data, but we need to be able to answer any query using a merge between old batch results and new real-time computations. The user will be notified on the graphical user interface about the taxpayers who may be fraudulent.

D.4. Requirements update

Getting hands-on concrete experience while developing our prototype gave us the opportunity to discover concrete issues which may be faced by any Big-Data application’s developer, and for which DICE tools can do a lot to help the user; mainly by automating tasks, simplifying processes, or guiding him through identified steps. Here are some updated expectations regarding DICE tools for our demonstrator:

<u>ID:</u>	NETF.1
<u>Title:</u>	Design
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	DIA design guidelines through DICE: a kind of graphical representation of the workflow - the methodology (showing achieved/remaining steps). It can be either a specific editor or a technical view (Eclipse part) comparable to the GMF Dashboard ² . This component must be interactive, i.e., can be used to navigate automatically through diagrams, etc.

<u>ID:</u>	NETF.2
<u>Title:</u>	Performance impact
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	We want to know the impact on the performance metrics when using different architecture alternatives for different quality and performance indicators.

<u>ID:</u>	NETF.3
<u>Title:</u>	Storage
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	The key requirement of Big Data storage technologies is that they can handle very large amounts of data. They have to be easily scalable to accommodate data growth, and must deliver sufficiently rapidly data to analytics tools. DICE must provide a way to model them and express requirements about them in the model.

² https://wiki.eclipse.org/Graphical_Modeling_Framework/Tutorial/Part_4

<u>ID:</u>	NETF.4
<u>Title:</u>	Deployment models
<u>Priority of accomplishment:</u>	Could have
<u>Description:</u>	We want to model deployment configuration to automatically generate deployment scripts. Indeed, manual deployment is time-consuming and needs a plethora of Linux and Apache frameworks tuning.

<u>ID:</u>	NETF.5
<u>Title:</u>	DIA analysis and assessment
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	Analyse and validate the application architecture using various data sources and computational logic without spending time in building runtime platforms which may have to be changed later.

<u>ID:</u>	NETF.6
<u>Title:</u>	Quality, performance and other metrics monitoring
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	<p>Automatically extract quality and performance metrics iteratively to improve those metrics on following versions:</p> <ul style="list-style-type: none"> • monitoring data and logs to detect candidate anomalies and report to user, • detecting data design anti-patterns, • estimate root-causes of quality anomalies

Deliverable 6.1. Demonstrators implementation plan.

<u>ID:</u>	NETF.7
<u>Title:</u>	Scalability analysis
<u>Priority of accomplishment:</u>	Should have
<u>Description:</u>	<p>Cloud deployment and scalability</p> <ul style="list-style-type: none"> • Evaluate cloud alternatives for deployment: Cost versus performance. • Automatically create cloud deployment configurations.

<u>ID:</u>	NETF.8
<u>Title:</u>	Design
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	<p>The DICE Profile must provide validation constraints in order to help the user build “correct” models (DPIM, DTSM, DDSM) which can be fully by DICE tools unless producing wrong models, i.e., which can’t be used by tools, will be worthless.</p>

<u>ID:</u>	NETF.9
<u>Title:</u>	Fault tolerance
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	<p>We identified a killer issue for any Big Data applications while “playing” with our prototype: if a node in our cluster is no more responding, the whole application is down. We need to express such properties in the model which must be verified later in-situ.</p>

<u>ID:</u>	NETF.10
<u>Title:</u>	Data Consistency
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	I need to be sure that my data is always consistent and any occurring fault/error related to human or hardware failure will not corrupt my data.

<u>ID:</u>	NETF.11
<u>Title:</u>	Privacy (Design)
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	I need to express privacy on my data. For example in order to set roles and permissions for different users with different accreditation levels.

<u>ID:</u>	NETF.12
<u>Title:</u>	Security (Design)
<u>Priority of accomplishment:</u>	Must have
<u>Description:</u>	Some data must be encrypted so I need to explicitly mention this feature in the model.

D.5. Implementation plan

As mentioned in the Future plan section, we have identified three main tasks which will be led in parallel:

- **UCN-1:** Enhance the methodology in order to get it as simple and complete as possible.
- **UCN-2:** Use, test and give feedbacks to tool providers.
- **UCN-3:** Add new features to our demonstrator and push the system to its limits to get real execution situations.

UCN-1 and UCN-2 will be performed during all the project life with respect to the DICE deliverables scheduling and the tool providers' availabilities. Regarding UCN-3, we have planned a set of tasks mainly for the short and mid-terms:

- **May 2016:** Building the Spark core features dealing with batch processing (connection with the Cassandra database, run user's queries using the GUI).
- **June 2016:** Building the real-time features of Spark in order to deal with streaming processing.
- **July & August 2016:** Generate and validate the DTSM and DDSM models obtained from the validated DPIM.
- **September 2016:** Deploy the application using the deployment tool.
- **October 2016:** Offers to the end-users more features regarding fraud indicators modeling.
- **December 2016:** Release of a beta version of the demonstrator which is able to run at least 2 kinds of fraud detections in order to start monitoring and optimization.

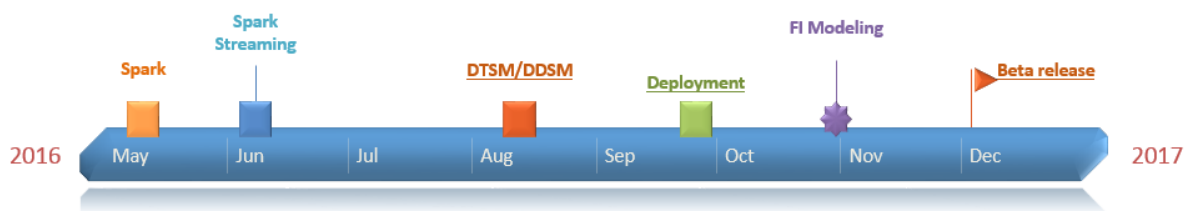


Figure 29. NETF demonstrator: Roadmap

Once the Beta release made and according to the primary results, we will prioritize added-value features such as Deep Learning. In fact we will be looking for an intelligent solution which will propose, to the end-users, identified fraudulent conducts. With respect to UCN-2, we already started using some DICE tools and we have identified future needs which will be covered by other tools. The tables below give more details about the identified implementation iterations.

Activity/Task short name	Spark Batch Layer
Description	Instantiating the batch layer of the lambda architecture with Spark. This Spark application will analyze the full dataset created by the taxpayer random generator module, and identifies frauds.
DICE tools utilized	IDE, DICE Profile.
Position in DICE methodology	This corresponds to the platform and technology specific implementation activity of the methodology.
Part of scenario	Running Spark SQL queries over Cassandra, combining the results thereof with Spark joins, filter, map, and reduce transformations.
Requirements addressed	NETF.1, NETF.3
Envisioned outcome	Fraudulent tax declarations and tax fraudsters.

Activity/Task short name	Spark Speed Layer
Description	Instantiating the speed layer of the lambda architecture with Spark. This Spark application will analyze every second the fresh data newly created by the taxpayer random generator module, and use its knowledge of previously identified frauds to classify them in real-time. This is achieved by using machine learning algorithms.
DICE tools utilized	IDE, DICE Profile
Position in DICE methodology	This corresponds to the platform and technology specific implementation activity of the methodology.
Part of scenario	Running machine learning algorithms that will be trained on a generated training dataset. Running these algorithms with the Spark streaming framework.
Requirements addressed	NETF.1, NETF.3
Envisioned outcome	Fresh data about fraudulent tax declarations and tax fraudsters.

Activity/Task short name	DTSM and DDSM
Description	Generate and validate the DTSM and DDSM models obtained from the validated DPIM. Indeed, if we want take advantage of most of the DICE tools, we have to produce such models.
DICE tools utilized	DICE Profile and M2M Transformations
Position in DICE methodology	The model-to-model transformations offered by the methodology facilitates the creation of the DTSM and DDSM models by using the DPIM.
Requirements addressed	NETF.1
Envisioned outcome	The models are the cornerstone of DICE. They will be used by all the tools as an input in order to assess the quality of the application, analyze its reliability, optimize its environment, etc. The production of such models will allow us to shift from functional POC (proof-of-concept) realization to high quality application delivery.
Delivery dates	August 2016

Activity/Task short name	Continuous Deployment
Description	We need to automate the deployment process in order to be able to repeat it as many as needed. In fact, today we manually deploy the application between 10 and 30 times per day using third party tools (ssh client, jar...). We will be using the deployment tool in order to automate this step for rapid feedback and quick debugging.
DICE tools utilized	Deployment tool
Position in DICE methodology	The DICE methodology is quality-centric and encourage the end-users to deploy and run quality assessment tools (configuration, optimization...) as much as possible in order to make an iterative enhancement of the application.

Requirements addressed	NETF.4
Envisioned outcome	We envision an important increase of our productivity and above all in reducing the time required for manual deployment.
No. of releases	Does not apply
Delivery dates	Does not apply

Activity/Task short name	Fraud Indicator Modeling
Description	Design a graphical or textual modeling language allowing users to define new fraud indicators. It will be possible to integrate the models done with this specific language into the Spark batch layer and speed layer described above.
DICE tools utilized	DICE IDE
Position in DICE methodology	This task is related to the business modeling and requirement analysis activity of the DICE methodology.
Part of scenario	Defining the modeling language with a metamodeling language such as Ecore or MOF. Defining a graphical or textual way to build models conform to it.
Requirements addressed	NETF.3
Envisioned outcome	A fully integrated tax indicator modeling language.

Activity/Task short name	Monitoring and Optimization
Description	Release of a beta version of the demonstrator which is able to run at least 2 kind of fraud detections in order to start monitoring and optimization.
DICE tools utilized	Monitoring platform & Optimization
Position in DICE methodology	This task corresponds to the runtime feedback analysis activity of the DICE methodology.
Part of scenario	Running DICE monitoring and configuration optimization tools.
Requirements addressed	NETF.6, NETF.7

Activity/Task short name	Analysis and Assessment
Description	Analyse and validate the application architecture using various data sources and computational logic.
DICE tools utilized	Simulation, Verification
Position in DICE methodology	This task corresponds to the DTSM simulation and verification activity of the DICE methodology.
Requirements addressed	NETF.5

