Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements


# Requirement Specification

## Deliverable 1.2

| | |
|---|---|
| **Deliverable:** | D1.2 |
| **Title:** | Requirement Specification |
| **Editor(s):** | Ilias Spais (ATC) |
| **Contributor(s):** | Giuliano Casale (IMP), Tatiana Ustinova (IMP), Pooyan Jamshidi (IMP), Marc Gil (PRO), Christophe Joubert (PRO), Alberto Romeu (PRO), José Merseguer (ZAR), Raquel Trillo (ZAR), Matteo Giovanni Rossi (PMI), Elisabetta Di Nitto (PMI), Damian Andrew Tamburri (PMI), Danilo Ardagna (PMI), José Vilar (ZAR), Simona Bernardi (ZAR), Matej Artač (XLAB), Madalina Erascu (IEAT), Daniel Pop (IEAT), Gabriel Iuhasz (IEAT), Youssef Ridene (NETF), Josuah Aron (NETF), Craig Sheridan (FLEXI), Darren Whigham (FLEXI) |
| **Reviewers:** | José Merseguer (ZAR), Matej Artač (XLAB) |
| **Type (R/P/DEC):** | Report |
| **Version:** | 1.0 |
| **Date:** | 31-July-2015 |
| **Status:** | Final version |
| **Dissemination level:** | Public |
| **Download page:** | http://www.dice-h2020.eu/deliverables/ |
| **Copyright:** | Copyright © 2015, 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:** | Universidad 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

D1.2 presents the requirements analysis for the DICE project. The outcome of the analysis is a consolidated list of requirements that define the project's goals and that will guide the DICE technical activities. Rational behind requirements is to develop appropriate methods and tools matching industry real-life needs. This deliverable achieves the first step, i.e., milestone MS-1 "Baseline and requirements", of the DICE project.

A critical success factor in the development of high quality software applications is a deep understanding of the projects' requirements, both technological and user-centric ones, as opposed to perceived requirements. DICE consortium has conducted a requirements analysis that focused on both, business/demonstrator requirements and pure technical ones. DICE objectives stated in the project's proposal served as the baseline of the analysis process, so to ensure that all DICE interests will be completely covered and also to avoid requirements out of the scope of DICE demonstrators. Hence, the resulting requirements specification envisions the foundations for a successful implementation of a research innovative prototype and for creating applications that meet industrial real needs.

For business requirements gathering, use case providers were strongly engaged and focus groups were organized within each WP. These open discussions were extremely helpful not only for the use case providers but for the technical team as well, as a result, the needs of each tool demonstrator were defined. Descriptions of the demonstrators, including business and technical details of the current and envisioned situations, were presented to the technical team. At the same time, WP leaders distributed questionnaires in order to aggregate feedback in a more structured way. Furthermore, use case providers had the chance to raise questions and ask for clarifications and thus several issues were brought to the technical teams and corresponding actions were defined when needed. From these open discussions a list of high level business and technical requirements related to the demonstrators was created.

The elicitation of technical requirements started with a thorough review of the state of the art (D1.1 "State of the art analysis"), a summary is given in this document. In sequence, WP leaders organized weekly conference calls for discussing requirements. Thus, more than fifty detailed technical requirements were defined, covering all DICE aspects completely. At the same time, several questionnaires were created and distributed to several stakeholders (including use case providers), which are identified in this report. Then, WP leaders analysed the extended list in order to come up with a consolidate one that will serve as the baseline for each WP's activities from now on.

It must be mentioned that the requirements analysis process  not in DICE is intended to be a continuous process, hence this report does not coincide with the end of requirement analyses. A requirement repository was defined and will be continuously updated as part of task T1.1 "State of the art analysis and requirements specification" until month 24 (M24). Furthermore, an intermediate update on the status of the demonstrator requirements will be added as a chapter in deliverable D6.1 (M16) in WP6.

 

# Table of contents



 

 

# List of figures



# List of tables



 

 

# Introduction

This deliverable provides a description of the requirements analysis carried out in the first six months of the DICE project. The objective of this exercise was to refine the vision outlined at proposal stage into a concrete set of requirements, driven by

- internal technical discussions,
- input from focus groups within the companies that act as use case providers, and
- impact considerations.

The DICE consortium (both use case and technology providers) participated in several activities (e.g., open discussions, preparing, distributing and filling in questionnaires to the project participants) for creating a consolidated list of DICE requirements that will serve as baseline for the technical activities from now on. It is worth noting that a requirements repository was created and will be continuously updated following DICE achievements.

The resulting document provides a comprehensive description of the project expected outcomes. Furthermore, early technical discussions leading to this deliverable have been published in the form of a vision paper, co-authored by all the consortium partners:

> DICE: Quality-Driven Development of Data-Intensive Cloud Applications. Proceedings of the 7th International Workshop on Modelling in Software Engineering (MiSE 2015) [1].

This document is structured as follows. Section A sets the baselines of the project (vision and objectives) that served as the boundaries of the requirements analysis. The ultimate goal was to avoid requirements out of project's scope, but at the same time address all DICE interests. Section B explains the general approach followed to elicit the requirements. It focuses on the rationale of the process and the methodology followed (open discussions and questionnaires). Furthermore, all the stakeholders of DICE were defined, both from the demonstrator's point of view (actors and profiles that are engaged in DICE workflows) and external initiatives that will be interested in DICE achievements. Section C provides an initial description of the three demonstrators, focusing on the current and envisioned situations. Several aspects are discussed, like the motivation of participating in DICE, high level conceptual architectures of the system AS-IS, and the TO-BE vision, i.e., how the system will evolve throughout the project. For each demonstrator, a consolidated list of requirements is presented. Section D discusses the technical requirements per WP and concludes with a consolidated list that will serve as the baseline for future work. Finally, Section 2 identifies suggested future steps for the next requirements iteration.

The deliverable comes with a companion document made up of tables that describe the comprehensive list of requirements and scenarios that has been defined in the first six months of activities. Out of this list, a subset of these requirements has been prioritized and reported in the present document.

# A. DICE Initial Vision

## A.1. Overview

As mentioned before, the vision was presented at MiSE [1], an established workshop co-located with the 37th International Conference on Software Engineering (ICSE 2015), co-sponsored by IEEE and ACM. We point the reader to [1] for a detailed description of the technical needs that justify the focus of DICE on quality-driven development for data-intensive applications (DIAs) in relation to the current state-of-the-art. As part of the project deliverable, such investigation as been primarily confided to DICE deliverable D1.1 – "State of the Art", which gives a detailed overview of the terminology and the state-of-the-art in each technical area of the project.

## A.2. Aims

The goal of the DICE project is to deliver a methodology and a tool chain to help independent software vendors to develop Big Data applications without compromising on quality. DICE proposes innovations concerning both functional and non-functional properties (NFPs) of DIAs.

For what concerns **functional properties**, DICE wants to extend UML with a novel profile to annotate properties of data. Therefore, model driven engineering approaches (MDE) will benefit from the new profile, which encompasses annotations ranging from platform-independent to technology and deployment-specific issues. Moreover, the profile will also consider technologies and architecture styles that are specific to Big Data, such as the Lambda architecture we have extensively described in deliverable D1.1. The main challenge of this endeavour is to develop a UML profile, a consistent methodology, and the underpinning model-to-model transformations to support MDE for Big Data applications. Furthermore, DICE aims at translating such high-level design models into a concrete TOSCA-compliant deployment plan and execute it by means of a model-driven deployment and configuration tool chain.

On the side of modelling and predicting **non-functional properties**, UML designs will be annotated with performance, reliability and privacy requirements using existing profiles, such as the UML MARTE and UML DAM, but also with novel annotations that describe distinguishing features of DIA affecting NFPs, for example the data volume or velocity used by the application. Then, tools will be developed to predict the fulfilment of these requirements before and during application development. In particular, DICE envisions the co-existence of multiple simulation, verification, and testing tools that can guide the developer through the quality assessment of early prototypes of the Big Data application. For example, a developer could initially describe the application architecture, expected user behaviour and the technologies to be used; based on this specification, he could then explore the forecasted response times under increasing volumes or rates of data intakes. This information would be helpful to assess if a given architecture design can meet service level agreements (SLAs). The novelty is accounting for data properties, such as volumes or rates, in the predictions.

The DICE development environment will offer a general methodology, which can be useful in a number of **application domains**. The project plans to develop demonstrators in the areas of News & Media, e-Government, and Maritime Operations. In News & Media, streaming solutions that connect to social platforms will need to be modelled, together with Hadoop/MapReduce processing of the acquired social data. The case of e-Government provides a test scenario for the DICE methodology to apply in an environment with legacy data systems, where decision-making related to the best Big Data technologies to adopt is complex. Lastly, Maritime Operations is a sector where streaming data related to vessel movement needs to be processed and analyzed in real-time to guarantee safe and correct port operations.

## A.3. Assumptions

The DICE vision outlined above is fairly broad, therefore one aim of the requirement analysis phase was to clarify the classes of applications, technologies and software engineering phases on which the project will be focused on. We here review the outcomes of these analyses.

### A.3.1 Application classes

A key decision for the project was concerned with the precise definition of the classes of applications that will be supported and analysed. After careful analysis, we have concluded that including support for classic multi-tier enterprise applications, or hybrids between such applications and Big Data technologies, would effectively come at the expense of research innovation, since these applications are now rather mature and their integration with Big Data technologies appears less challenging to achieve than developing, using and configuring the Big Data technologies themselves. **We have therefore decided to focus on data-intensive applications developed using Big Data technologies, taking the Lambda architecture as the reference architecture style.** It is felt that, given the very limited availability of MDE methods for this class of systems, the potential impact of the project can be maximized by focusing on Big Data technologies, as opposed to canonical multi-tier applications. It is also expected that the project will focus on Java applications, although some aspects related to deployment and testing may assume knowledge of other languages that are more popular across system administrators (e.g., Python).

### A.3.2 Technology Focus

After internal consultation with the use cases providers and based on impact considerations, the DICE project has identified an initial set of Big Data technologies and platforms that we plan to use as an initial baseline for the investigation. These technologies are presented in the Table 1:

**Table 1: Initial set of Big Data technologies and platforms planned to be used as a baseline in DICE.**

| **Technology** | D.1. **Type** | D.2. **Comments** |
|---|---|---|
| D.3. Apache Hadoop[2] | Batch processing | We are primarily motivated to model this open source implementation of MapReduce since it is available for free both in private and public clouds. |
| Apache Spark[3] | Batch & real-time processing | Spark is seen by many as the successor of Hadoop and therefore its support is important for long-term sustainability of the project results. |
| Apache Storm[4] | Real-time processing system | Streaming is a fundamental technology in Big Data and Storm is a widely adopted solution for continuous querying of data streams. |
| Apache Cassandra[5] | NoSQL database | Cassandra has emerged in recent years as a reliable column-oriented NoSQL database. The DICE project is currently exploring with the Horizon 2020 project MIKELANGELO the possibility of collaborating on this technology. |
| Amazon S3[6] | Cloud storage | S3 is the market leader for cloud-based blob storage. Virtually all public cloud applications on Amazon AWS relying to some extend on S3. |
| Cloudera Oryx 2[7] | Machine learning | Oryx 2 will be exploited in DICE both as toy application, to |

[2] https://hadoop.apache.org/
[3] http://spark.apache.org/
[4] https://storm.apache.org/
[5] http://cassandra.apache.org/
[6] http://aws.amazon.com/s3/

| | | |
|---|---|---|
| | platform. | demonstrate the applicability of the methodology to lambda architecture styles, but also as a suitable library for machine learning. Thus we will also offer native support for specifying Oryx-based applications in the DICE profile, and deploying, testing and configuring Oryx 2. |

The initial plan is to focus on building support in the DICE methodology for DIAs that rely on the above technology stack or a subset thereof. We will leave the door open in further stages of the project for integrating additional technologies, when needed. Clearly, one of the aims of DICE is to provide an extensible modelling approach for DIAs, therefore our main goal is to illustrate the applicability of the approach on popular technologies, but community participation is needed to be able to cover a wider range of technologies.

### A.3.3 Software engineering phases

The project has a strong focus on development and initial prototyping. Therefore, a potential source of misunderstanding is on why the project encompasses monitoring, configuration and deployment tools. This is illustrated in Figure 1, which summarises the scope of the project within common development and service delivery activities.

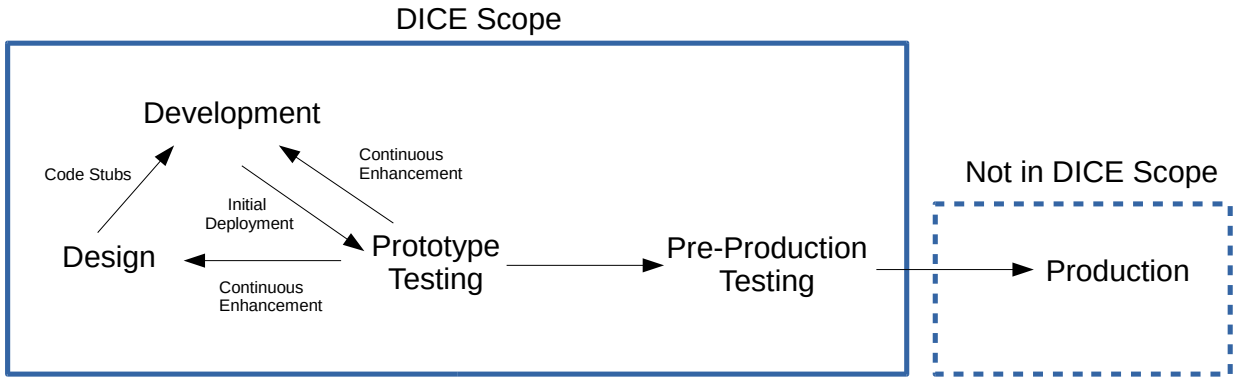


**Figure 1. Software engineering activities and the DICE scope.**

As can be seen, the **project intends to accelerate the initial prototyping, testing and deployment for the application,** so to empower DICE developers with tools that can increase their productivity and reduce the time to market. However, DICE does not focus on managing the application at runtime (e.g., monitoring, adaptation, software upgrades, etc), although certain tools developed to help the ISV developers, such as the DICE monitoring tools, are in principle exploitable also by operators in a production environment.Big Data.

### A.3.4 Public vs Private Cloud Deployments

Our initial vision aims at supporting applications developed on top of the Infrastructure-as-a-Service (IaaS) model, which is popular in both private and public clouds. It is recognized that most of the enterprise market is still focusing on private cloud deployments for Big Data systems, especially because the data itself is often sensitive business data, therefore we will perform testing especially on private premises, exploiting the availability of a dedicated private testbed offered by Flexiant, which acts in the project as testbed provider, but several DICE tools will also be applicable for describing and testing in public cloud deployment.

[7] https://github.com/cloudera/oryx

### A.3.5 Positioning in Model-Driven Engineering & DevOps

A fundamental assumption of the DICE project is that the models co-existing in our MDE methodology (UML & TOSCA models) will act as a vehicular language to integrate different tools across the DICE tool chain. It will also allow different actors to see a complex Big Data systems at different level of abstractions, such as abstract levels, architecture levels, and deployment levels. Therefore, the DICE approach is fully compliant with the MDE philosophy, and should be deemed as a possible extension of MDE to the realm of Big Data.

In addition to embracing MDE, DICE aims at contributing to the DevOps movement, which is gaining increasing attention in the last few years. As stated in D1.1, *DevOps is a hybrid software development and operation paradigm that predicates the intermixed co-operation, collaboration and communication between Dev-Teams (i.e., development teams) and Ops-Teams (i.e., Operations Teams)*.

DevOps is fairly heterogeneous at the moment and tends to emphasize quality aspects in terms of both functional correctness and QoS. As highlighted in Figure 1, DICE does not cover the operation (production) activities in a direct way, but, indeed, it offers tools for deployment and monitoring that, even though are functional in DICE to the deployment and analysis of the system behavior in the testing environment, could be used in operation as well. Moreover, such tools, as well as the analysis and testing features offered by DICE, are beneficial to cover the gap from Dev to Ops offering to developers a way to be aware of non-functional and resource allocation aspects that otherwise are invisible to them and of competence of other stakeholders in the software lifecycle. The MDE approach underpins this goal by proposing to use UML models as a way for Dev and Ops to share a global view of the system. Such global view of the system is a key element of the DevOps vision. DICE therefore wants to emphasize the convergence of MDE and DevOps as a way to achieve an integrated, harmonized system view and orchestration between Dev and Ops.

# B. DICE Requirements Elicitation Approach

## B.1. Strategic Rationale and Methodology

The scope of this deliverableis to specify business and technical requirements for developing methods and tools matching with industry needs. Due to the need of adopting research prototypes to real case scenarios, the approach should be the design of DIAs taking into account technical and business quality. In this direction, a requirement analysis is based on a combination of state of the art technological achievements and a user-centred approach, with the ultimate goal of creating DIAs that meet requirements and provide precise descriptions of the content, functionality and quality demanded by prospective users.

The strategic rationale of DICE requirements analysis is to:

- Analyse business needs to understand how to shape the DICE IDE and tools for increasing chance of industry adoption.
- Consult stakeholders of the DICE outcomes to understand how they expect to interact with a quality analysis tool chain for developing DIA.
- Identify quality services (quality engineering) indicators/metrics that will be adopted throughout the software development lifecycle.
- Determine how to leverage MDE to capture special data characteristics and usage patterns of DIAs.
- Identify requirements related to how quality assessment and architecture enhancement using a tool chain with a high degree of automation can be upgrade DIAs.
- Identify requirements related to how a quality-driven MDE approach can effectively influence the design and/or refactor of a DIA.

To analyse business and technical requirements two complementary approaches are followed in parallel. For initiating business requirements gathering, interviews (focus groups) with the three DICE demonstrators, namely, ATC, PRODEVELOP (PRO) and NETFECTIVE (NETF), were organized in order to provide a comprehensive description of their cases and their current situation (AS-IS description). Out of these discussions, an initial set of high level requirements (business and technical) for the three Industry demonstrators were inferred. To broaden the scope of applicability of DICE results, a series of questionnaires (one per technical WP) were shared between the partners and used by the consortium to derive a set of general business goals and expected achievements by applying the DICE tools to the existed prototypes (TO-BE descriptions). Finally, the requirements defined by the demonstrators were validated by WP leaders in order to ensure correctness with respect to DICE objectives and vision.

The elicitation of technical requirements starts with a thorough analysis of the state of the art survey covering related technologies (D1.1. “State of the art analysis”). A summary is given in this document. Following this analysis, the technical requirements are defined per WP, based on internal discussions among the members of the consortium and on questionnaires distributed to the demonstrators .

The results of this initial requirements analysis will be later refined and extended in a new deliverable D6.1 that will be developed at month sixteen of the project.

## B.2. DICE Actors

Central to the approach on requirements engineering is the identification of agents such as **human stakeholders,** who will interact with the overall or part of the system and therefore who can be considered responsible for specifying requirements and using the DICE tools, and the **software**, that are

 

the non-human agents in the DICE ecosystem (DICE tools, application to be developed, deployment infrastructure). Table 2, details the actors of the DICE project mentioning the respective profiles. Such list is consistent with the description of work and the DICE vision paper [1].

**Table 2: DICE actors - human stakeholders and software**

| Actors | Acronym | Type | Description |
|---|---|---|---|
| Data-Intensive Application | APPLICATION or DIA | Software | The data-intensive application to be developed or extended with DICE. |
| Requirements Engineer | REQ_ENGINEER | Human | A technical person capable of eliciting and maintaining the requirements of the DIA. |
| Software Architect | ARCHITECT | Human | The designer of the application architecture, also capable of modelling it. |
| Software Developer | DEVELOPER | Human | A technical person tasked with implementing parts of the application code. |
| Administrator | ADMINISTRATOR | Human | System manager capable of performing deployment, installation and configuration actions on the application. |
| Quality Assurance Engineer | QA_ENGINEER | Human | Technical person tasked with evaluating, using prediction and analysis, the performance of the application and continuously assessing impact of changes in the application. |
| Quality Assurance Tester | QA_TESTER | Human | Technical person tasked with evaluating, using measurements and prototype and pre-production testing, the performance of the application and continuously assessing impact of changes in the application. |
| Testbed | TESTBED | Software | Cloud testbed on which the DIA runs or will run. |
| Continuous integration tools | CI_TOOLS | Software | Tools to support continuous integration of the application code and artefacts with the application deployment. These are part of WP5. |
| Transformation tools | TRANSFORMATION_TOOLS | Software | Tools implementing the model-to-model transformations. These are part of WP3. |
| Simulation tools | SIMULATION_TOOLS | Software | Tools to simulate the system's quality (e.g., performance, reliability) from model-based specifications. These are part of WP3. |
| Verification tools | VERIFICATION_TOOLS | Software | Tools to assess the formal properties of the system from model-based specifications. These are part of WP3. |
| Optimization tools | OPTIMIZATION_TOOLS | Software | Tools to optimize the application cost and deployment architecture using numerical optimization. These are |

| | | | |
|---|---|---|---|
| | | | part of WP3. |
| Monitoring tools | MONITORING_TOOLS | Software | Tools to monitor quality metrics in the application and in its underpinning software stack and infrastructure as the application runs. |
| Iterative Enhancement tools | ENHANCEMENT_TOOLS | Software | Tools to support the developers and architects with the task of evolving the application quality after tests on prototypes. |
| Anomaly detection and trace checking tools | ANOMALY_TRACE_TOOLS | Software | Tools to verify if properties in the monitoring data concerning performance, reliability or safety are consistent with expectation or show deviations. |
| Quality testing tools | QTESTING_TOOLS | Software | Tools to generate artificial workloads that are sequentially submitted to the running application according to a testing plan. |
| Initial deployment tools | DEPLOYMENT_TOOLS | Software | Tools to help administrators to deploy and initially configure the application. |

Besides the DICE demonstrators and in order to ensure wider adoption of project's results, DICE consortium identified in Table 3 external initiatives that are very close to DICE research and business areas of interest and could be targeted in the future for additional input for requirement analysis. Such external stakeholders include many developers, architects and administrators who fit the profiles described in Table 2. In particular, as reported in Deliverable D7.1, we have already presented DICE to some of these groups (e.g., SPEC RG DevOps Performance WG, Tata Consulting Services) and the requirements presented in this deliverable have considered the feedback received during these presentations.

**Table 3: Example of external initiatives bringing together DICE stakeholders**

| **Topic/research areas** | **Target Group** |
|---|---|
| Quality-aware DevOps | SPEC RG DevOps Performance WG [3]<br>DevOps [4] |
| MDE for DIA | E2 working cluster<br>OMG [5]<br>Tata Consulting Services [6] |
| Agile software development and delivery methodologies | Agile Alliance [7]<br>Agile practitioners communities (e.g. LeanAgileTraining.com SME from US, Romanian Agile Community)<br>OSLC [8] |
| Cloud based application/service | OASIS TOSCA<br>Chef recipes<br>Cloud Security Alliance (CSE) [9]<br>E2 working cluster<br>Network for Sustainable Ultrascale Computing (ESUS) [10] |
| Open source software | CloudWatch [11]<br>OW2 [12]<br>AppHub<br>PIN-SME [13] |

| | |
|---|---|
| Optimisation of quality of Big Data applications | Big Data Community<br>Big Data Europe Project [14]<br>Big Data Value [15]<br>NESSI organization [16] |

## B.3. Continuously updating the requirements - The DICE repository

As part of task "T1.1: State of the art analysis and requirements specification" a requirements repository has been defined and it will be continuously updated until M24 of the project. The repository is stored on Google Drive since it speed ups the collaborative work between multiple partners editing at the same time. The requirement repository is fairly extensive and as of now includes about 250 entries, including both requirements and scenarios. **A detailed record of these requirements and scenarios has been included in the companion document to this deliverable – "D1.2 Companion Document".** Given the large number of entries in the repository, which has been subject of multiple reviewing stages, a prioritization has been made to ensure that each partner can keep the focus first on the most important requirements.

The repository is currently composed of:

- A requirements template.
- A scenarios template.
- A table of DICE actors.
- For each Work Package:
  - A requirements list. Each requirement described according to the aforementioned template.
  - A list of use cases. Each use case described according to the aforementioned scenario template.
- A set of consolidated requirements. They are a subset of requirements extracted from the WP requirements. The rational for creating this restricted list was to simplify the reviewing process and to keep a distinction between the essential requirements from the low priorities requirements.
- Related textual documents.

The templates for requirements and scenarios are filled with textual descriptions inspired by a standardized formal language in RFC2119 [17] to describe among others: pre-conditions and post-conditions, rationale, flow of events and keywords. In particular, we often highlight in block letters MUST, SHOULD and COULD/MAY. These should however not be confused with the similar keywords that we use for the priority of accomplishment of a requirements, i.e., "Must have", "Should have", "Could have".

The requirements template we have defined is as follows:

| **Field** | **Description** |
|---|---|
| **ID** | A unique ID for this requirement/assumption |
| **Title** | A title/short name for this requirement/assumption |
| **Priority of accomplishment** | One of the following:<br>Must have: The system must implement this requirement to be accepted.<br>Should have: The system should implement this requirement: some deviation from the requirement as stated may be acceptable.<br>Could have: The system should implement this requirement, but may be accepted without it. |

| | |
|---|---|
| **Type** | One of the following:<br>Domain Assumption, Requirement |
| **Description** | Specify the intention of the requirement/assumption |
| **Rationale** | If the description is not descriptive enough, this entry gives a justification of the requirement/assumption. Otherwise this entry will be filled with N/A. |
| **Supporting materials** | If applicable, give a pointer to documents that illustrate and<br>explain this requirement/assumption. Otherwise this entry will be filled with N/A. |
| **Tentative scheduling** | Tentative scheduling of accomplishment. |

The use case scenarios template is instead as follows:

| **Field** | **Description** |
|---|---|
| **ID** | A unique ID for this use case (name of partner + UseCase + number) |
| **Title** | A title or short name for this scenario |
| **Task** | The work package taks(s) that can address this scenario |
| **Priority** | "REQUIRED", "RECOMMENDED", "OPTIONAL" |
| **Actor 1/2/3/4/...** | The actors involved in the use case |
| **Flow of events** | Describe the flow of event characterizing the use case |
| **Pre-conditions** | Describe known pre-conditions. Whenever possible, make sure they are measurable or verifiable. |
| **Post-conditions** | Describe expected post-conditions. Whenever possible, make sure they are measurable or verifiable. Include satisfaction criteria if applicable. |
| **Exceptions** | Specify possible exceptions that should be handled |
| **Data Exchanges:** | If notable data exchanges happen in this scenario, they should be recorded here. |

# C. Business goals and requirements - DICE demonstrators

## C.1. The NewsAsset demonstrator. A distributed data – intensive media system

### C.1.1 Current solution (AS IS)

#### *C.1.1.1. Business and technological domain of the case study*

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 been already utilized in several domains of our society. As such, they have already penetrated in the media domain as well. News agencies are feeling the impact of the capabilities that these technologies offer (e.g. processing power, transparent distribution of information, sophisticated analytics, quick responses, facilitating the development of the next generation of products, applications and services). Interesting media and burst events is out there in the digital world and if processed efficiently they 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 in 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 eco-system 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.

#### *C.1.1.2. The vision of NewsAsset – The DICE demonstrator*

The ultimate vision of the NewsAsset product is to leverage on environments spanning from cloud computers and social networks to handheld smartphones and sensor platforms situated in the field. Subsequently, enhancements should focus on addressing several heterogeneity and distribution aspects. In terms of heterogeneity, different types of platforms/data sources will be utilized to produce and collect the media data, while in order to process very large amounts of data, highly robust and scalable computations must run in parallel across a cluster of machines, implemented and executed in a distributed way. From a bird's eye view, the NewsAsset vision will be realized by the implementation and deployment of four layers, namely the nodes, the network, the operation/management analysis and the application/services layer.

Figure 2 presents all the proposed layers and the respective components of the NewAsset vision, while the red block presents the part that will be managed in the context of DICE. More specifically, in the context of DICE, ATC is interested in developing services that will be able to aggregate, store, process

 

and digest media items (text, images, web links, etc.) from social networks and RSS feeds that subscriptions are valid (e.g. Twitter, Athens News Agency). Real-time streaming data and historical data stored to archives are being processed by an existing NewsAsset prototype. ATC's vision is to enhance and refactor prototype's architecture by exploiting the capabilities offered by Big Data and Cloud Computing technologies without damaging the quality requirements of the media domain.

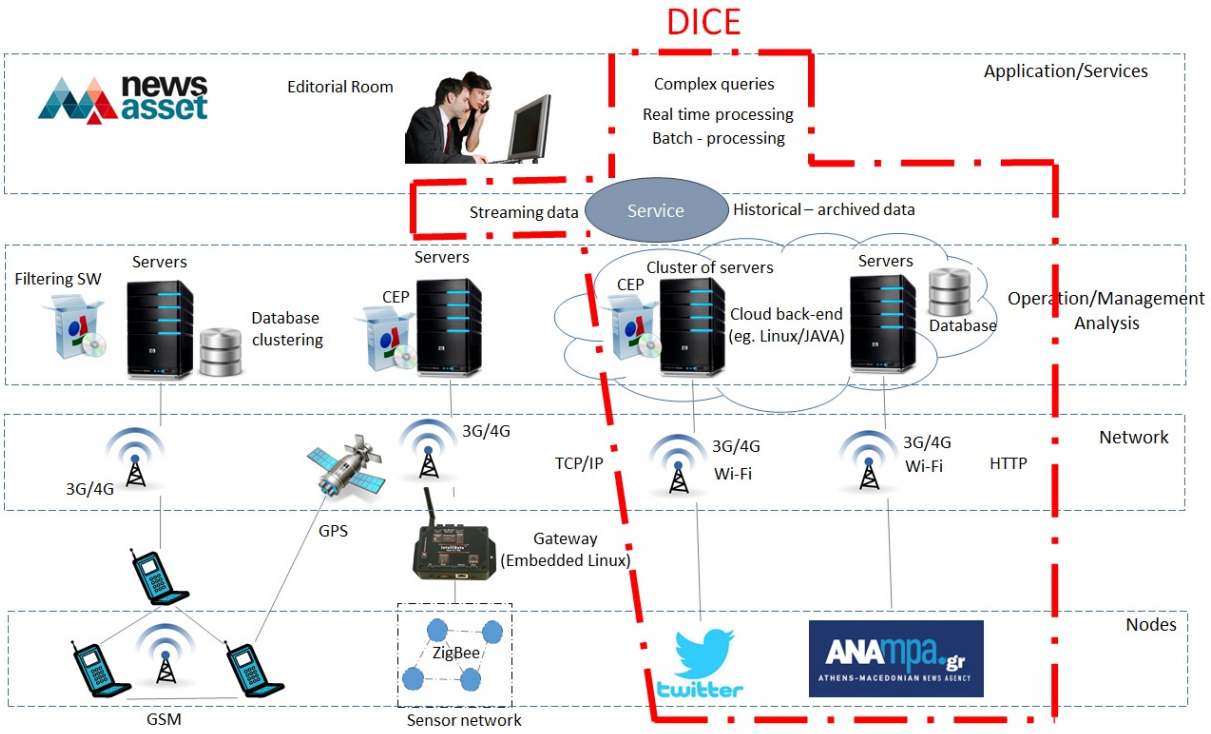


**Figure 2. Positioning DICE in the NewsAsset vision.**

### *C.1.1.3. The use case scenario*

NewsAsset use case will demonstrate how the application designer will be able to design DIA that efficiently gather data from heterogeneous nodes, utilize services to aggregate them to relax redundancy and compose services to associate relevant data coming from different sources. To do so, ATC will use as a baseline the achievements an EU-funded project, namely SocialSensor [19] that produced a platform capable of collecting, processing, and aggregating big streams of social media data and multimedia to discover trends, events and interesting media content. Figure 3 presents a high level conceptual architecture of the platform that ATC will develop by utilizing DICE. It demonstrates that a series of modules are exploiting the capabilities of the two prototypes and several user interfaces are exposing the outcome to the end users.

 

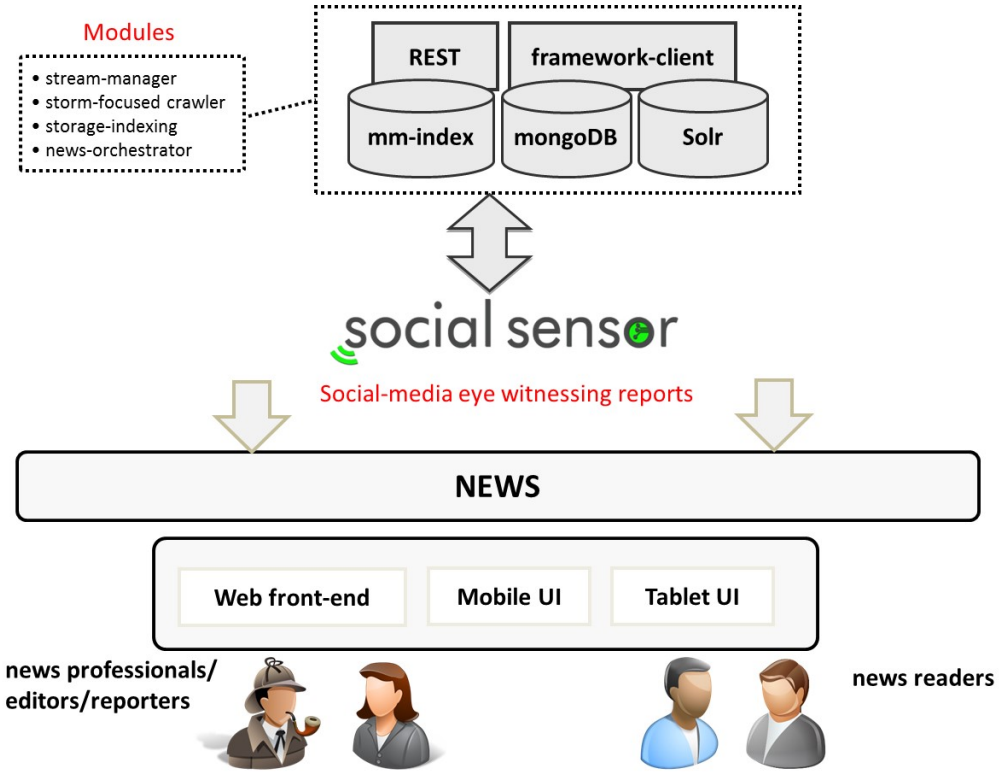


**Figure 3. NewsAsset in DICE: conceptual architecture.**

NewsAsset platform is **a single tool that quickly extracts material from social media, with context**. The above statement translates to the following desired properties for the tool of interest:

- "single tool": a common interface for all required functionalities;
- "quickly": information discovery and search should happen, in real time;
- "surfaces": automatically discovers and organises (clusters) relevant content;
- "material": any piece of multimedia content (text, image, video);
- "social media": across all relevant social media platforms (or a combination of platforms relevant to journalists);
- "with context": provides information on mainstream media sources.

Figure 4 depicts NewsAsset use case scenario.

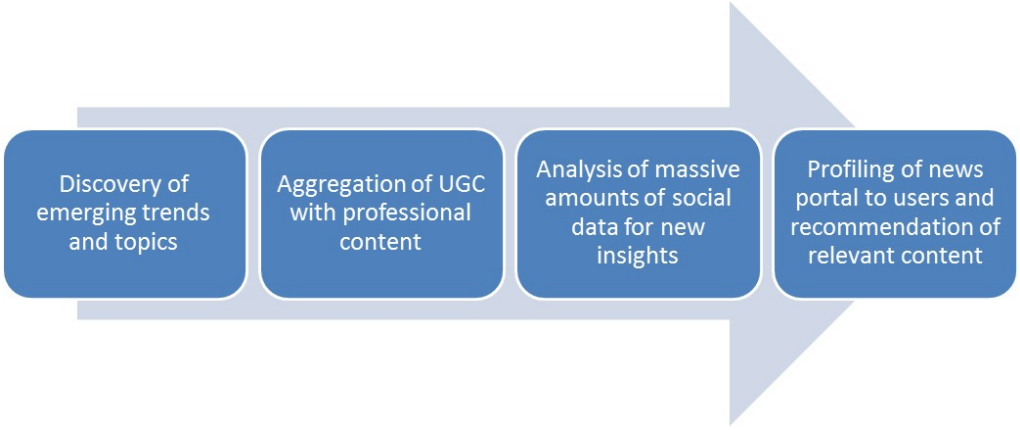


**Figure 4. The NewsAsset use case scenario.**

Functionalities:

- **identifies and visualises events and trends** across social media sources in real time;
- **identifies key sources** and opinion formers around any event (news portals and similar sources);
- supports journalists in **verifying publically available user generated content** (text, images, video and audio) from social media sources by suggesting sources that refer to similar content and have been annotated as trustworthy by the users in the past.

More details related to the NewsAsset demonstrator will be given in deliverable D6.1 “Demonstrators implementation plan” due to M16 in the project.

### C.1.2 The DICE vision (TO BE)

#### *C.1.2.1. Overview*

While the news item is the main entity in the news editing workflow, the modernized 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 cloud-computing and Big Data. 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.

#### *C.1.2.2. Challenges*

NewsAsset journalism demonstrator will have to address several challenges:

**Functional challenges**:

- Utilize cloud processing and Big Data technologies to enhance existing services and to:
  - support the collection of high volumes of data from various heterogeneous sources
  - address the challenge to manage the complexity of large software and data-intensive systems
  - incorporate and digest heterogeneous resources, and
  - handle efficiently features like data velocity, volume and rate of update or granularity.
- Handle streams of data fed from social network and web-based platforms. The architecture for this type of real time stream processing must deal with ingest, processing, storage and analysis of a large number of events per minute.
  - Services that aggregate data and relax redundancy both in terms of content and storage.
  - Services that extract and generate meta-data from aggregated content.
  - Services that cope with high rates of data.
  - Services that associate relevant data coming from different sources
- Store the data in a highly scalable distributed storage system.
- Analyze real time streaming data, for instance filtering, pre-processing and validation, and provide insights. Merge the outcome with a batch processing analysis using historical data.

**Non-functional challenges**:

- Scale out to serve a wide range of workloads and use cases, while providing low-latency access to the stored information. The system should be linearly scalable, and it should scale out rather than up, meaning that throwing more machines at the problem will do the job.
- Cloud deployment and scalability

    - Evaluate cloud alternatives for deployment: cost versus performance.
    - Automatically create cloud deployment configurations.
- Support the need for a robust system that is fault-tolerant, simple and comprehensive.
- Realization of quality-driven processes in the Big Data domain
    - Throughput, availability, prvacy, fault tolerance and response time.
- Simulation and predictive analysis.
    - Analyze and evaluate impact on quality metrics of business logic changes: consider metrics like performance, cost and privacy.
    - Analyze and evaluate different architecture alternatives for different quality and performance indicators. Are there any other architecture alternatives for better results?
    - Analyze and evaluate impact on quality and performance metrics when bursts of events occur and the load is high: Which are the thresholds for the architecture that is capable of efficiently managing high loads?
    - Automatically create and run controlled simulation environments. For example, deploy and run NewsAsset on an isolated simulation environment with historical data to verify that integration tests pass.
- Analyze execution logs to identify bottlenecks.
    - Detect bottlenecks.
    - Give insights on current quality and performance metrics to iteratively improve them.
- Availability of the system
    - 24/7.
    - Monitor components and their status.
    - Collect execution data in a production-like environment to analyze and improve metrics and for improving simulations.

### C.1.3 Initial list of requirements

Tables from Table 4 to Table 15 contain specifications for initial requirements developed in the process of internal DICE consultations.

**Table 4: Handling of data streams from social network and web-based platforms.**

| | |
|---|---|
| **ID:** | D.4.ATC.1 |
| D.5.**Title:** | Handle streams of data fed from social network and web-based platforms |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to be able to design a system that will include services that cope with high rates of data which vary in terms of size and format |
| **Rationale:** | Address the challenge to manage the complexity of large software and data-intensive systems. The architecture design of such a system must deal with a workload that is unpredictable due to its nature (social media items vary in number and size in an unpredictable way |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | N/A |

**Table 5: Scaling requirement.**

| | |
|---|---|
| **ID:** | D.6. ATC.2 |
| D.7. **Title:** | Scaling |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to be able to design a system that will be able to scale out to serve a wide range of workloads. The system should be linearly scalable, and it should scale out rather than up, meaning that throwing more machines at the problem will do the job |
| **Rationale:** | Analyze real time streaming data (for instance filtering, pre-processing and validation) and provide insights. |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | This requirement is connected to ATC.3 |

**Table 6: Auto-scaling requirement.**

| | |
|---|---|
| **ID:** | D.8. ATC.3 |
| D.9. **Title:** | Auto - Scaling |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to be able to set monitoring parameter values as thresholds for triggering automatic scaling. At the same time I want to be able to monitor performance and quality metrics and evaluate the impact of the data rate in order to re-configure auto-scaling policy details and desired performance rates. Which are the thresholds for efficiently managing high loads? |
| **Rationale:** | The two goals of auto-scaling are to optimize resources used by an application (which saves money), and to minimize human intervention (which saves time and reduces errors) |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | This requirement is connected to ATC.2.<br>Involves SIMULATION_TOOLS, MONITORING_TOOLS, QTESTING_TOOLS |

**Table 7: Requirement for the selection of public cloud providers.**

| | |
|---|---|
| **ID:** | D.10. ATC.4 |
| D.11. **Title:** | Selection of public cloud providers |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | As an ADMINISTRATOR I want to obtain models of well – known cloud offerings (e.g. Amazon EC2) related to data management. For instance location, protections (e.g. encryption), who has access to it, both in the short-term and long-term |
| **Rationale:** | The rationale of modelling public cloud providers is to identify offerings related to data security and privacy. Where are the data located, which is the data management policy offered, etc. |

| | |
|---|---|
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | N/A |

**Table 8: Distributed processing requirement.**

| | |
|---|---|
| **ID:** | **D.12.** ATC.5 |
| D.13. **Title:** | Distributed processing |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to be able to model a system that distributes its processing functionalities to different components/modules. |
| **Rationale:** | Avoid a centralized architecture. Push processing functionalities to several different components in order to perform quick and accurate computations. These components may serve different business logics or could be instances of the same one hosted on different nodes. |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | N/A |

**Table 9: Requirement for the simulation and predictive analysis.**

| | |
|---|---|
| **ID:** | D.14. ATC.6 |
| D.15. **Title:** | Simulation and predictive analysis |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | As a QA_ENGINEER, I want to measure the impact of different architecture alternatives based on performance and cost. For example, deploy and run the application on an isolated simulation environment with historical data to verify that quality tests pass |
| **Rationale:** | Identify the best architecture alternatives according to the workload managed. Give insights on current quality and performance metrics to iteratively improve them |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | Involves SIMULATION_TOOLS |

**Table 10: Requirement for the quality metrics monitoring.**

| | |
|---|---|
| **ID:** | D.16. ATC.7 |
| D.17. **Title:** | Quality metrics monitoring |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As a QA_ENGINEER_I want to automatically extract quality metrics iteratively (from |

| | |
|---|---|
| | current version to compare with previous and next versions) to improve those metrics on following versions |
| **Rationale:** | Monitor throughput, fault tolerance, response time and availability |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | Involves MONITORING_TOOLS, QTESTING_TOOLS |

**Table 11: Requirement for deployment models.**

| | |
|---|---|
| **ID:** | D.18. ATC.8 |
| D.19. **Title:** | Deployment models |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to model deployment configuration to automatically generate deployment scripts |
| **Rationale:** | N/A |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | Same as PO.4.<br>Involves TRANSFORMATION_TOOLS, DEPLOYMENT_TOOLS |

**Table 12: Requirement for the bottleneck detection.**

| | |
|---|---|
| **ID:** | D.20. ATC.9 |
| D.21. **Title:** | Bottleneck detection |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As a DEVELOPER I want to know the bottlenecks of my NLP processing so that I can fix them for better performance |
| **Rationale:** | N/A |
| **Supporting material:** | D6.1 (M16) |
| **Other comments:** | Same as PO.5<br>Involves VERIFICATION_TOOLS, ENHANCEMENT_TOOLS, MONITORING_TOOLS, TESTING_TOOLS |

**Table 13: Expression of non-functional requirements.**

| | |
|---|---|
| **ID:** | D.22. ATC.10 |
| D.23. **Title:** | Express non-functional requirements |
| **Priority of accomplishment:** | Should have |

| | |
|---|---|
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT/ REQ_ENGINEER I want to have tools to express non-functional requirements that will be considered when modeling my system |
| **Rationale:** | In the current version of the system non-functional requirements are defined in a textual way making their adoption extremely difficult |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 14: Common-model vocabularies.**

| | |
|---|---|
| **ID:** | D.24. ATC.11 |
| D.25. **Title:** | Common model vocabularies |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to use/exploit common domain-independent and domain-dependent vocabularies (meta-models, UML profiles, etc.) that can be used to describe application's components and express requirements (if possible at component level), enabling a common understanding |
| **Rationale:** | Common vocabularies can be beneficiary when describing similar components. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 15: Methodology blueprint requirement.**

| | |
|---|---|
| **ID:** | D.26. ATC.12 |
| D.27. **Title:** | Methodology blueprint |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT/DEVELOPER I want to obtain a graphical representation (outline) of the methodology process integrated within the main tooling suite, and easily accessible throughout the suite. A cheat sheet explaining the overall process and details for each task. Links to launch required tools for each task from the cheat sheet or the graphical outline |
| **Rationale:** | Can be used as a tutorial for the end user |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

## C.2. Big Data for e-Government

### C.2.1 Current solution (AS IS)

Tax evasion and frauds represents a huge problem for governments, causing them a big loss of money each year. The U.S. Treasury registered an increase of the number of identified fraudulent federal returns by more than $4 billion (40% increase) from 2011 to 2012 [20]. The EU estimates sums due to tax evasion and avoidance in the order of € 1 trillion [21]. In Italy alone, an estimated €285 billion taxes remained unpaid in 2012, about 18% of the GDP [22]. 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 in each country, especially for business entities which may have many justifiable exemptions and expenses that are harder to track or verify.

With the availability of massive amounts of structured and unstructured information, data organization and analytics are a promising solution to find anomalies and uncover violations, and even predicting frauds that may happen in the future. 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.

Unfortunately, most public organizations are still using legacy information systems that do not support the recent Big Data platforms and where the data is highly duplicated, scattered, even diluted (e.g., resulting from COBOL programming style) without any consolidated view, no correlation, no explicit semantics and with ineffective naming conventions. Documentation is often inadequate or missing. On top of that, the huge amount of digital data must be collected from all sorts of sources including non-classical databases and “flat files”, sometimes with odd structuring, such as configuration files and comma-delimited text files.

The point to be made here is that migrating data from such systems into a modern platform is undoubtedly a tedious task, and the process can be risky, expensive and disruptive. Because of those limitations, Big Data plans are often held back by governments and companies.

Using Big Data technologies implies an unavoidable step which consists on migrating the legacy database schema and data into a relational database. NETF released Blu Age Data and Database Modernization [23], a modernization tool set that automates the migration of databases and data (see Figure 5) allowing them to be standardized. This product allows for joint modernization of both the data format and the database schema.

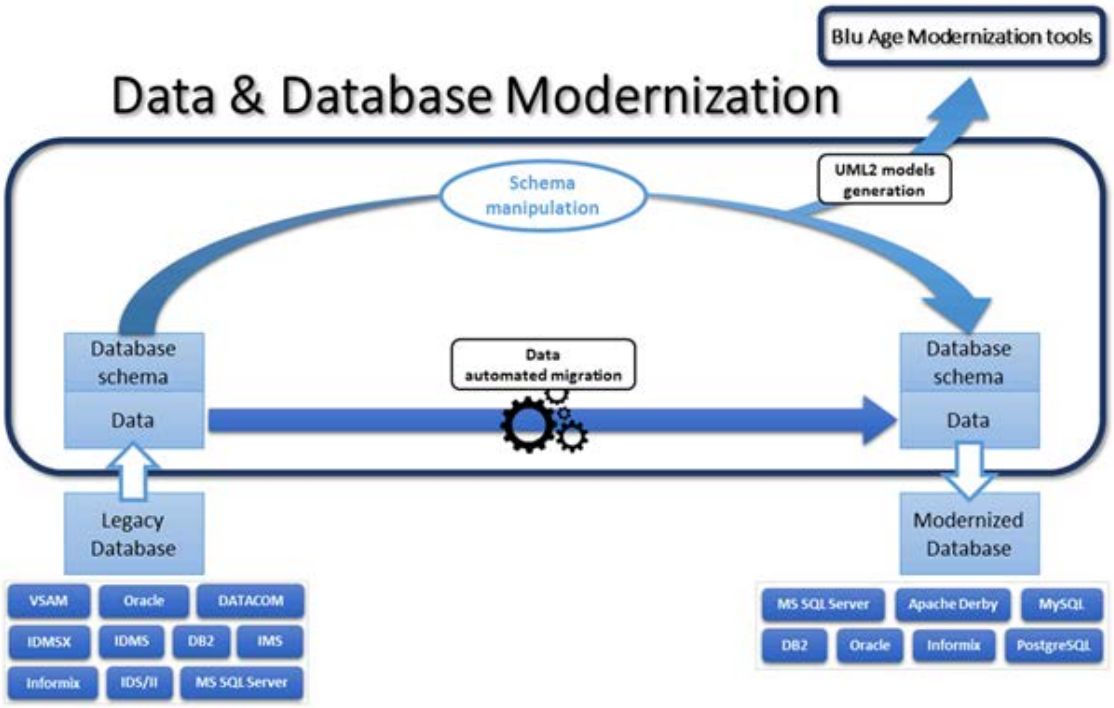


**Figure 5. Data & Database migration using Blu Age**

Figure 6 presents the modernization process which is done in 9 phases:

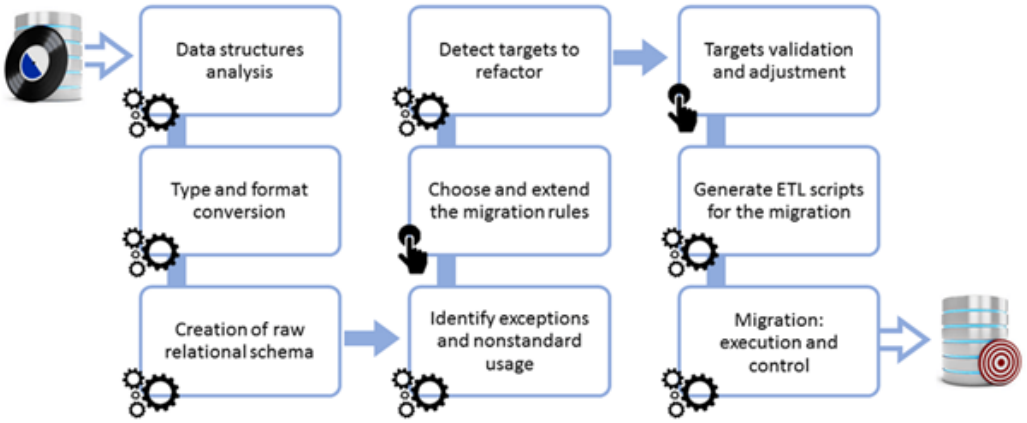


**Figure 6. Blu Age modernization process**

Making the transition to NoSQL is more common as more and more businesses are doing the same to keep up with the growing volumes of data to process. The possibility of rebuilding these flat files and non-classical databases in a unified manner with Big Data approach for high-performance processing capabilities (MapReduce/NOSQL) is a way of unanticipated consequences.

### C.2.2 The DICE vision (TO BE)

NETF plans to build a software prototype to demonstrate the capabilities of Big Data in e-government applications and to integrate Big Data platforms into the list of supported Blu Age target technologies (target in MMD scope). The prototype is expected to use randomly generated data.

NETF will produce a specific model of the description of a taxpayer in French. This model will be based on the CERFA forms which are publically available on the Web [24]. Based on CERFA forms,

we will be able to build a huge model with hundreds of attributes describing a taxpayer (name, address, ID card number, married/single/..., birthday, car owned, social security number...). This intrinsically raises no privacy/confidentiality issues since the model will not contain any actual data of any real person, but carry random data instead. All instances will be automatically generated using various algorithms. Each algorithm will generate specific text datasets on heterogeneous formats (flat files, legacy proprietary databases, etc.). Therefore it is expected that confidentiality constraints wo not be considered.

Legacy information systems of many organisations are huge brittle data sets (flat files with odd structuring, non-classical databases…). For example, tax fraud detection for a public institution is the timely processing of complex requests across Big Data environments, which is only possible with computing infrastructure and technologies supported by DICE.

The focus is the direct valorisation of known pattern of fraud by deducing new data with richer semantics. This amounts to processing ill-explained financial data whose interpretation may lead to further investigation towards tax fraud management. In this case, identifying fraudulent people and organisations (among the tremendous volume of data to gather, correlate and queries to execute) is challenging. With DICE technology, a novel application will be developed that will consolidate data from different data stores and provide a technical infrastructure to help users move away from using data mining to visualize information toward forecasting and making up decision between suggested options, as in the spirit of Big Data applications (see Figure 7).

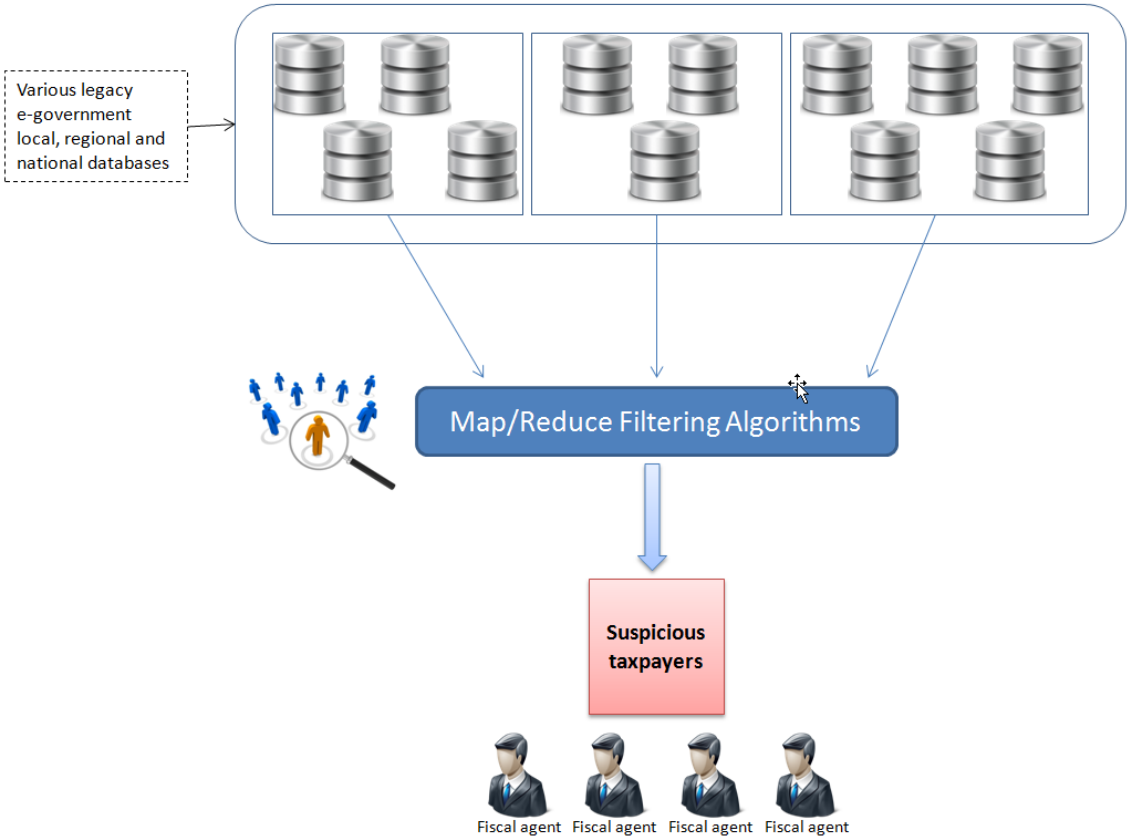


**Figure 7. The DICE vision for tax fraud detection**

Our demonstrator aims to build models of "fraudulent conduct" from the automatic operations on generated tax data files such as business creation or bank accounts abroad which represents millions of data points. Here is a list of some examples to be detected by our demonstrator:

- Identifying taxpayers who are registered in different French departments in order to collect fraudulently social aids. For example, the demonstrator can catch a non-existent address, a non-residential address or an address that just does not make sense.
- The fictitious relocation of the taxpayer who improperly claims domiciled abroad in order to not pay tax on income or wealth in France.
- Companies normally collect VAT from their customers, but "forget" to pay back to the Treasury as they should by not filing of 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.

Provided results will mainly have a record of value among others and in no case lead to an automatic condemnation. The demonstrator will just 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.

### C.2.3 Initial list of requirements

Tables from Table 16 to Table 22 contain specifications for initial requirements developed in the process of internal DICE consultations.

**Table 16: Design requirement.**

| | |
|---|---|
| **ID:** | D.28. NETF.1 |
| D.29. **Title:** | Design |
| **Priority of accomplishment:** | Must have |
| **Description:** | 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) such GMF Dashboard. This component must be interactive, i.e. can be used to automatically navigate through diagrams, etc. |

**Table 17: Performance impact requirement.**

| | |
|---|---|
| **ID:** | D.30. NETF.2 |
| D.31. **Title:** | Performance impact |
| **Priority of accomplishment:** | Must have |
| **Description:** | I want to know the impact on the performance metrics when using different architecture alternatives for different quality and performance indicators. |

**Table 18: Storage requirement.**

| | |
|---|---|
| **ID:** | D.32. NETF.3 |
| D.33. **Title:** | Storage |
| **Priority of accomplishment:** | Must have |
| **Description:** | The key requirement of Big Data storage is that it can handle very large amounts of data and it has to be easily scalable to accommodate data growth, and that it can provide the input/output operations per second (IOPS) necessary to deliver data to analytics tools. In fact, DICE must provide a way to model such technology and express this requirement in the model. |

**Table 19: Requirement for deployment models.**

| | |
|---|---|
| **ID:** | D.34. NETF.4 |
| D.35. **Title:** | Deployment models |
| **Priority of accomplishment:** | Could have |
| **Description:** | As an ARCHITECT I want to model deployment configuration to automatically generate deployment scripts. |

**Table 20: DIA analysis and assessment.**

| | |
|---|---|
| **ID:** | D.36. NETF.5 |
| D.37. **Title:** | DIA analysis and assessment |
| **Priority of accomplishment:** | Must have |
| **Description:** | Analyse and validate the application architecture using various data sources and computational logic. |

**Table 21: Requirement for the monitoring of quality and performance metrics.**

| | |
|---|---|
| **ID:** | D.38. NETF.6 |
| D.39. **Title:** | Quality, performance and other metrics monitoring |
| **Priority of accomplishment:** | Must have |
| **Description:** | Automatically extract quality and performance metrics iteratively to improve those metrics on following versions:<br>• monitoring data and logs to detect candidate anomalies and report to user<br>• detecting data design anti-patterns<br>• estimate root-causes of quality anomalies |

**Table 22: Requirement for the scalability analysis.**

| | |
|---|---|
| **ID:** | D.40. NETF.7 |

| | |
|---|---|
| D.41. **Title:** | Scalability analysis |
| **Priority of accomplishment:** | Should have |
| **Description:** | Cloud deployment and scalability<br>• Evaluate cloud alternatives for deployment: Cost versus performance.<br>• Automatically create cloud deployment configurations |

# C.3. DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)

## C.3.1 Current solution (AS IS)

### *C.3.1.1. Posidonia Operations*

#### *C.3.1.1.1. Summary*

Posidonia Operations is an integrated port operations management system. Its mission is "glocally" monitor vessels position in real time to improve and automate port authorities operations.

Posidonia Operations is a data-intensive application (DIA) implemented in Java. It processes streamed data from Automatic Identification System (AIS) receivers [25], [26], a system that gets vessels position and metadata in real time. The encoding protocol of an AIS sentence can be found in [27]. Figure 8 shows a schematic example of an AIS network operation.


**Figure 8. Example of an AIS network operation [28].**

Once an AIS stream is parsed, it is published to a message queue for further processing: analysis, complex event processing, data integration, visualization, etc.

AIS messages are binary encoded sentences that can be decoded into key-value objects. Its size is usually under 100 bytes.

#### *C.3.1.1.2. AIS streaming*

To get data from an AIS Network we have two options:

a) A TCP connection to the port AIS receiver.

b) Make use of AISHub [29], a service that provides global access to AIS streams. Usually the AIS stream provided by AISHub is filtered, so the velocity and volume of data is lower than a direct connection to an AIS receiver.

#### *C.3.1.1.3. Use cases*

Typical use cases of Posidonia Operations are:

- Visualise the traffic of a port in real time in a web application or reproduce historical situations.
- Detect real world events related to vessels operations such as bunkering, berthing or anchoring.
- Integrate real world events with the management system of a port authority.

#### *C.3.1.1.4. Core components*

The core components of interest for DICE are:

- A **streaming processor or AIS parser** that collects the data from the AIS receiver and parses it. Currently this is a custom Java implementation.
- A **message queue** for subscribing/publishing data such as AIS messages or events detected. We are currently supporting the RabbitMQ system.
- A **complex event processing** engine that subscribes to AIS messages and correlates them in time and space to identify events. Current implementation is based on JBoss Drools.

#### *C.3.1.1.5. Current status*

Posidonia Operations is a commercial product developed by Prodevelop already deployed and operated in 5 port authorities in Spain.

Currently it can be deployed in two ways:

- In the port authority private cloud. All Posidonia Operations infrastructure has to be installed, configured and monitored in the port servers. In this case the challenge is to replicate or adapt production environments, manage continuous delivery, monitoring, etc.
- In a public cloud. A public cloud allows us to configure Posidonia Operations as a service in order to provide access to different port authorities on demand. In this case the challenge is more related to scalability

#### *C.3.1.1.6. Challenges*

We are facing three challenges:

a) **Velocity and volume of data**: For a single port we usually have a velocity of 100 AIS messages per second with a volume of about 5 million messages per day. This is approximately 200MB of data per day, that we store indefinitely and we plan to use for batch analysis. In our public cloud instance these numbers can be multiplied by the number of ports we offer support as an upper limit.

In this case the challenge is related to the scalability of our solution and data processing, storage and analysis.

**Table 23: Velocity and volume of data for increasing number of ports**

| | D.42. **1 port** | D.43. **3 ports** | D.44. **10 ports** | D.45. **100 ports** | D.46. **AISHub (filtered)** |
|---|---|---|---|---|---|

| D.47. **Velocity** | 100 msg/second | 200 msg/second | 500 msg/second | 3000 msg/second | 300 msg/second |
|---|---|---|---|---|---|
| **Volume - 1 day** | 200MB | 400MB | 1GB | 5GB | 600MB |
| **Volume - 1 month** | 6GB | 12GB | 36GB | 150GB | 20GB |
| **Volume - 1 year** | 70GB | 140GB | 400GB | 2TB | 200GB |

b) **Continuous delivery:** Posidonia Operations is a product in continuous evolutionary maintenance, so that we have to face continuous deployments, different port configurations, evaluate regressions, ensure availability, etc. Everything related to create and run development, test and deployment environments and having replicas of environments is a challenge for us.

c) **Non-functional properties:** Since this is a product already in production we have to ensure the safety of the components, reliability of the results, performance, etc. so we need monitoring tools but also tools to evaluate these properties as the product evolves.

#### *C.3.1.1.7. Issues*

Some of the issues we have on our product lifecycle:

- Difficulty to have coherent development, testing and production environments.
- Rapidly configure, run and monitor environments.
- Difficulty to evaluate performance through stress test.
- Difficulty to evaluate streaming and Big Data solutions and focus on our business.
- Scale.
- Identify hardware requirements.
- Automate deployments in the infrastructure of our clients.
- Launch tests to evaluate regressions.

#### *C.3.1.1.8. Overview of the business benefits expected from DICE*

Having tools to improve some of our current challenges will allow us to:

a) Improve the scalability and deployment of our current solution in order to be able to give support to more port authorities.
b) Improve our current development processes in terms of testing, continuous integration, configuration and execution of development and testing environments, in order to be more agile.
c) Improve our service quality policies in order to ensure safety, reliability and monitoring, also to offer a better solution to our clients.

A more detailed view of the benefits expected from DICE can be found in the section C.3.2 of this document.

#### *C.3.1.1.9. Users*

Posidonia Operations has three different operation modes: automatic, assisted and manual.

In the automatic mode there is no need for users to interact with the system. It is deployed and configured as a black box daemon that processes the streaming data and integrates the events detected in the Port Management System.

In the assisted and manual modes, Posidonia Operations provides a web interface in order the port operators to visualize and manage the events detected by the system. The port operators are administrative users that have the required knowledge to transform the events detected by Posidonia Operations into information that has to be integrated into their Port Management System. In any case the number of users of Posidonia Operations will not exceed of 10 concurrent users.

#### *C.3.1.1.10. Modelling tools*

Our current solution does not make use of any modelling language to model the streaming process. Although we are used to use UML models to describe different parts of our applications whenever we can make profit of the models by transforming them to other models, generate code that can be used in production or simply having a diagram that help us to understand better the system.

Having said that we are open to use DICE modelling tools to model parts of Posidonia Operations to face our current challenges.

### *C.3.1.2. Development lifecycle and continuous integration*

#### *C.3.1.2.1. Lifecycle*

- We do iterative and incremental development. Feature delivery each 3-4 weeks.
- Integration is done on every commit.
- Release of artifacts is done on every delivery.

#### *C.3.1.2.2. Quality assurance*

- Low unit tests coverage. Only the critical parts.
- Higher integration tests coverage: tests that use two or more components of the architecture.

#### *C.3.1.2.3. Environments*

We use different development and deployment environments:

- Developers' local machines.
- CI server/testing.
- Internal staging.
- Pre-production.
- Production.

#### *C.3.1.2.4. Tools and technologies*

- Posidonia Operations is built on Java with Maven
- Jenkins as a CI engine
- VN as a source code repository
- FTP as a Maven repository
- Sonar to static analysis of code
- Nagios to monitor the execution of Posidonia Operations

#### *C.3.1.2.5. Continuous integration*

- Every single Posidonia operations component has a continuous integration job that is created manually once.
- Every component is built on every commit.

- A typical configuration of a build job executed by Jenkins:
  - When a developer commits.
  - Jenkins downloads source code from SVN.
  - Compiles.
  - Executes tests (unit, integration, etc.).
  - Builds binary files.
  - Deploys on the internal staging server.
  - Restarts Posidonia Operations services.
  - When something fails an e-mail is sent to the developers to fix it.

### C.3.2 The DICE vision (TO BE)

Here are some achievements foreseen with the application of DICE to Posidonia Operations:

- Simulation and predictive analysis.
  - Analyze and evaluate impact on performance metrics of business logic changes: What happens if we add new CEP rules or change their implementation, will it affect the quality characteristics?
  - Analyze and evaluate impact on performance metrics when adding new publishers and subscribers to Posidonia Operations: What happens if we add more components reading and writing messages in the message queue?
- Analyze execution logs to identify bottlenecks.
  - Detect bottlenecks, anti-patterns.
  - Give insights on current performance metrics to iteratively improve them.
- Identify optimal deployment architecture (number of AIS parsers, CEPs, etc.)
  - Given a set of requirements for a port authority: number of AIS messages per second, geographical zones to monitor, extension, number of vessels, etc. propose optimal deployment architecture.
- Availability of the system
  - 24/7 availability.
  - Monitor components and their status.
  - Collect execution data in a production similar environment to analyze and improve metrics, to improve simulations, etc.
- Ensure reliability. Evaluate regressions.
  - Given the same port situation and a change in the implementation of CEP rules, are the results of the new implementation still reliable/correct?
- Scalability. Analyze and evaluate impact on quality and performance metrics when increasing the data input (number of AIS messages, number messages on the message queue, etc.)
  - What happens if we duplicate the number of AIS messages, what if we multiply per 10. This is the same as what if we want to give support to more port authorities with the same deployment. What are the hardware requirements to scale?
- Improve testing capabilities
  - Automatically create and run controlled simulation environments. For example, deploy and run Posidonia Operations on an isolated simulation environment with historical data to verify that integration tests pass.
  - Automatically generate test fixtures.
- Evaluate cloud alternatives for deployment: Cost, performance, etc.
- Automatically create cloud deployment configurations: Chef, etc.

### C.3.3 Initial list of requirements

Tables from Table 24 to Table 38 contain specifications for initial requirements developed in the process of internal DICE consultations.

**Table 24: Simulation and predictive analysis of new business rules.**

| | |
|---|---|
| **ID:** | D.48. PO.1 |
| D.49. **Title:** | Simulation and predictive analysis of new business rules |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As a REQ_ENGINEER I want to know the impact of a new CEP business rule on the quality and performance metrics so that I can evaluate different alternative requirements with a lower impact on quality and performance |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves SIMULATION_TOOLS |

**Table 25: Scalability analysis requirement.**

| | |
|---|---|
| **ID:** | D.50. PO.2 |
| D.51. **Title:** | Scalability analysis |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to know the impact of changes in the data stream (more data) or in the areas of the port to monitor (more zones, bigger, more complex) on the performance and quality metrics so that I can make changes in the architecture |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves SIMULATION_TOOLS, MONITORING_TOOLS |

**Table 26: Requirement for the quality metrics monitoring.**

| | |
|---|---|
| **ID:** | D.52. PO.3 |
| D.53. **Title:** | Performance and other metrics monitoring |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to automatically extract performance metrics iteratively (from current version to compare with previous and next versions) to improve those metrics on following versions |

| | |
|---|---|
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves MONITORING_TOOLS, QTESTING_TOOLS |

**Table 27: Requirement for deployment models.**

| | |
|---|---|
| **ID:** | D.54. PO.4 |
| D.55. **Title:** | Deployment models |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | As an ARCHITECT I want to model deployment configuration to automatically generate deployment scripts |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves TRANSFORMATION_TOOLS, DEPLOYMENT_TOOLS |

**Table 28: Requirement for the bottleneck detection.**

| | |
|---|---|
| **ID:** | D.56. PO.5 |
| D.57. **Title:** | Bottleneck detection |
| **Priority of accomplishment:** | Should 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 |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves VERIFICATION_TOOLS, ENHANCEMENT_TOOLS, MONITORING_TOOLS, TESTING_TOOLS |

**Table 29: Requirement for the testing and load stressing scenarios.**

| | |
|---|---|
| **ID:** | D.58. PO.6 |
| D.59. **Title:** | Testing and load stressing scenarios |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | As a DEVELOPER I want to configure testing and load stressing scenarios so I can debug concrete CEP business rules on different situations |

| | |
|---|---|
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves DEPLOYMENT_TOOLS, CI_TOOLS, QTESTING_TOOLS, TESTBED |

**Table 30: Performance impact requirement.**

| | |
|---|---|
| **ID:** | D.60. PO.7 |
| D.61. **Title:** | Performance impact |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | 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 |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves OPTIMIZATION_TOOLS, ENHANCEMENT_TOOLS |

**Table 31: Continuous integration requirement.**

| | |
|---|---|
| **ID:** | D.62. PO.8 |
| D.63. **Title:** | Model continuous integration jobs |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | As a QA_ENGINEER, I want to model continuous integration to automatically generate and configure continuous integration jobs |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves DEPLOYMENT_TOOLS, CI_TOOLS, TRANSFORMATION_TOOLS |

**Table 32: Requirement for the text fixtures generation.**

| | |
|---|---|
| **ID:** | D.64. PO.9 |
| D.65. **Title:** | Test fixtures generation |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | As a QA_ENGINEER I want to generate test fixtures to build simulation environments |

| | |
|---|---|
| **Rationale:** | In the context of Posidonia Operations a test fixture is a portion of the data extracted from a real time execution. This test fixture (data) is then injected into the system to reproduce concrete scenarios |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves QTESTING_TOOLS, CI_TOOLS |

**Table 33: Requirement for the running of simulation environments.**

| | |
|---|---|
| **ID:** | D.66. PO.10 |
| D.67. **Title:** | Run simulation environments |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As a QA_ENGINEER I want to automatically run isolated testing environments to validate integration tests |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves SIMULATION_TOOLS, DEPLOYMENT_TOOLS, CI_TOOLS |

**Table 34: Requirement for the execution metrics.**

| | |
|---|---|
| **ID:** | D.68. PO.11 |
| D.69. **Title:** | Execution metrics |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As a QA_TESTER I want to get real time execution metrics so I can improve them |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves MONITORING_TOOLS, QTESTING_TOOLS |

**Table 35: Requirement for the reliability results comparison.**

| | |
|---|---|
| **ID:** | D.70. PO.12 |
| D.71. **Title:** | Reliability results comparison |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | 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 |

| | |
|---|---|
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves MONITORING_TOOLS, ANOMALY_TRACE_TOOLS, QTESTING_TOOLS |

**Table 36: Deployment requirements.**

| | |
|---|---|
| **ID:** | D.72. PO.13 |
| D.73. **Title:** | Deployment requirements |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ADMINISTRATOR given some port requirements (number of vessels, number of messages per second, number of CEP rules, areas to monitor) I want to get insights on hardware deployment requirements I want to learn how much RAM, CPU, etc. will be required to get a certain performance |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves DEPLOYMENT_TOOLS |

**Table 37: Deployment monitoring requirement.**

| | |
|---|---|
| **ID:** | D.74. PO.14 |
| D.75. **Title:** | Deployment monitoring |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | As an ADMINISTRATOR I want to monitor the status of the deployed system so I can take actions when something fails or is about to failing |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves MONITORING_TOOLS, ANOMALY_TRACE_TOOLS |

**Table 38: Requirement for the deployment scripts.**

| | |
|---|---|
| **ID:** | D.76. PO.15 |
| D.77. **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 |
| **Rationale:** | N/A |
| **Supporting material:** | See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS) |
| **Other comments:** | Involves DEPLOYMENT_TOOLS |

### C.3.4 Initial list of use case scenarios

Based on the previous list of requirements, we can foresee the following use case scenarios of DICE for POSIDONIA OPERATIONS.

Possible usages scenarios that may arise in concrete situations include, but are not limited to the following:

- Install Posidonia Operations on the private infrastructure of a port authority
- Install Posidonia Operations on the Cloud
- Give support to another port in the cloud instance of Posidonia Operations
- Add new business rules (CEP rules) for different ports
- Support vessels traffic increase for a given port
- Run a simulation to validate integration tests, avoid regressions, etc.

# D. Overview of Technical Requirements

## D.1. IDE Technical Requirements (Work Package 1 Requirements)

### D.1.1 Overview

#### *D.1.1.1. Work package technical objectives.*

Work Package 1 (WP1) focuses on the definition of the DICE architecture, requirements and integrated framework. The goals of this WP are as follows: it will first analyse relevant state of the art of software engineering for data-intensive cloud applications, followed by requirement analysis, architecture definition, and integration (planning and execution). A large amount of effort is put on developing the DICE IDE as front-end to the DICE tool-chain components and releasing the integrated framework. This WP will implement the integrated DICE framework, including the Integrated Development Environment and the integration with it of all the tools developed in the technical WPs according to the patterns developed in D1.2. Baseline for the IDE will be the open-source (Eclipse Public License) MOSKitt CASE tool maintained by PRO and developed on top of the Eclipse framework.

#### *D.1.1.2. Work package stakeholders*

The correlation between WP1-WP6 is star-shaped and centered around WP1: the interaction between all work packages will be mediated as part of the WP1 activities on integration. Since all partners participate to WP1, this work package provides the ideal forum for synchronisation and updates.

The main stakeholders are:

- ARCHITECT: A technical person capable of designing and modelling a DIA through Platform-Independent UML Model stereotyped with the DICE profile.
- DEVELOPER: A technical person capable of developing a DIA is guided through the DICE methodology to accelerate development and deployment of the DIA with quality iteration.
- QA-TESTER: A quality-assessment expert that may also run and examine the output of the QA testing tools in addition to the developer.

Tools that will be created in the WP:

- Eclipse-based Rich Client Platform (RCP) that will include all the needed features to facilitate the work for analysts.

How the stakeholders will use the tools created in this WP:

- The DICE IDE will give support to the methodology defined in WP2, in order to ease the use of tools defined in WP3-5 by ARCHITECT, DEVELOPER, and QA-TESTER.

### D.1.2 Requirement Analysis

Tables from Table 39 to Table 46 contain specifications for technical requirements developed in the process of internal DICE consultations.

**Table 39: Requirement for the stereotyping of UML diagrams and DICE profile.**

| **ID:** | D.78. R1.1 |
|---|---|
| D.79. **Title:** | Stereotyping of UML diagrams with DICE profile |
| **Priority of accomplishment:** | Must have |

| | |
|---|---|
| **Type:** | Requirement |
| **Description:** | Open-source modelling tool with XMI and UML2.X (2.4 or 2.5) support |
| **Rationale:** | Support quality-related decision-making |
| **Supporting material:** | N/A |
| **Other comments:** | Stereotypes of the DICE profile will be applied in Papyrus UML models |

**Table 40: Requirement for the DICE methodology guidance.**

| | |
|---|---|
| **ID:** | D.80. R1.2 |
| D.81. **Title:** | Guides through the DICE methodology |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | Dashboard tool with DICE functional covering workflow support |
| **Rationale:** | The DICE IDE will guide the developer through the DICE methodology. Integrated development environment to generate Java code and accelerate development |
| **Supporting material:** | N/A |
| **Other comments:** | MOSKitt Dashboard diagram is proposed as workflow dashboard application. Generation of Java Code should be designed and implemented |

**Table 41: Continuous integration tools requirement.**

| | |
|---|---|
| **ID:** | D.82. R1.7 |
| D.83. **Title:** | Continuous integration tools IDE integration |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The CI_TOOLS MUST be integrated with the IDE. |
| **Rationale:** | The continuous integration tools must provide the means to be invoked remotely, with an option of controls and status display built into the IDE. |
| **Supporting material:** | N/A |
| **Other comments:** | A plugin to connect Eclipse with Jenkins will be provided on the IDE. This plugin allows to execute Continuous Integration (e.g., Jenkins) Tasks from Eclipse. Configuration should be done on Jenkins. This plugin allows to execute them from Eclipse, and see the results from there |

**Table 42: Requirement for the loading of annotated UML model.**

| | |
|---|---|
| **ID:** | D.84. R3IDE.4 |
| D.85. **Title:** | Loading the annotated UML model |
| **Priority of accomplishment:** | Must have |

| | |
|---|---|
| **Type:** | Requirement |
| **Description:** | The DICE IDE MUST include a command to launch the SIMULATION_TOOLS and VERIFICATION_TOOLS for a DICE UML model that is loaded in the IDE |
| **Rationale:** | The verification phase is launched from the DICE IDE, it is not meant to be independent, even though it involves launching an external tool (see R3.9.1). |
| **Supporting material:** | N/A |
| **Other comments:** | IDE will allow to execute external tools providing as a parameter the desired annotated UML model. A Papyrus UML model can be annotated with EAnnotation (from Ecore) in order to extend the Metamodel properties. |

**Table 43: Property verification requirement.**

| | |
|---|---|
| **ID:** | D.86. R3IDE.4.2 |
| D.87. **Title:** | Loading of the property to be verified |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The VERIFICATION_TOOLS MUST be able to handle the verification of the properties to be checked that can be defined through the IDE and the DICE profile |
| **Rationale:** | The properties to be checked are defined in the DICE UML model (possibly using templates). The requirement on the VERIFICATION_TOOLS is to be able to handle them. |
| **Supporting material:** | N/A |
| **Other comments:** | Properties to be verified can be listed in a custom model understandable by the VERIFICATION_TOOLS, where all the properties to be verified can be listed there. Both this model and the UML model will be used as input for the verification tools. |

**Table 44: Graphical output requirement.**

| | |
|---|---|
| **ID:** | D.88. R3IDE.5 |
| D.89. **Title:** | Graphical output |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | Whenever needed (for better understanding of the response), the IDE SHOULD be able to take the output generated by the VERIFICATION_TOOLS (i.e., execution traces of the modeled system) and represent it graphically, connecting it to the elements of the modeled system. |
| **Rationale:** | The output of the VERIFICATION_TOOLS (i.e., traces of the modeled system) should be presented in a user-friendly way to help the user better understand the outcome of the verification task. |
| **Supporting material:** | N/A |
| **Other comments:** | One way to do that is to create a metamodel that supports to define all the traces and relates them to an element from the UML model. The easiest way is to annotate the Papyrus UML model with EAnnotations (from Ecore) and, programmatically, colorate |

| | |
|---|---|
| | elements if desired. Also the traces (a string) can be added as annotation and show it within a popup or similar. |

**Table 45: Requirement for the visualisation of analysis results.**

| | |
|---|---|
| **ID:** | D.90. R4IDE5 |
| D.91. **Title:** | Visualization of analysis results |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | ENHANCEMENT_TOOLS SHOULD be capable of visualizing analysis results |
| **Rationale:** | N/A |
| **Supporting material:** | R4.25 |
| **Other comments:** | One way to do that is to create a metamodel that supports to define all the traces and relates them to an element from the UML model. The easiest way is to annotate the Papyrus UML model with EAnnotations (from Ecore) and, programmatically, colorate elements if desired. Also the traces (an string) can be added as annotation and show it within a popup or similar. |

**Table 46: Requirement for the loading of safety and privacy properties.**

| | |
|---|---|
| **ID:** | D.92. R4IDE6 |
| D.93. **Title:** | Safety and privacy properties loading |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The ANOMALY_TRACE_TOOLS MUST allow the DEVELOPER/ARCHITECT to choose and load the safety and privacy properties from the Model of the application described through the DICE profile |
| **Rationale:** | The properties to be analyzed are application-dependent, and they must come from somewhere in the DICE model of the application. The user knows what properties are to be monitored, so he/she should select those that most interest him/her |
| **Supporting material:** | R4.28 |
| **Other comments:** | A wizard where properties to be analyzed can be selected before launching the external tool. So the configuration model and the UML model will be passed as input to these tools. |

### D.1.3 Discussion of the requirements.

The selected requirements will conform into an Eclipse-based RCP that will include all the needed features to facilitate the work for analysts.

Papyrus is chosen as UML modeller. As Papyrus is open source, we can provide external functionality to their diagrams and models. These include, for instance, trying to modify the visualization of the validation results, adding actions for the diagram elements to show the traces, etc.

Furthermore, as Eclipse is also open source, allows to provide plugins to enhance the behaviour or provide functionality or tools to make the work easier. In this way, wizards to configure the external executions will be done.

We will be always limited to Papyrus to show the results. As far as possible, provided actions, wizards, diagram modification or contributions to Papyrus will be performed.

### D.1.4 Initial Roadmap

Below is the work allocation for the first year (up to M12) within task T1.3, plus prospects beyond it. For the first year we are planning to introduce support for different technologies and tools using an iterative approach. Details are given in the month and Milestone-based list below:

M7: Start of the task T1.3 “Integrated framework”

a) Creation of a code repository for DICE IDE

b) Definition of DICE IDE basic software components

MS2:

a) Development of DICE IDE first version including basic software components

b) Basic IDE functionalities for modelling and stereotyping models (WP2)

c) Basic IDE controls to link with the simulation and verification tools (WP3)

d) Definition of basic IDE control for monitoring and anti-patterns functionalities (WP4)

e) Basic IDE controls for Continuous-Integration (CI) tools (WP5)

f) Identify a suitable release format for the individual tools (e.g., Eclipse plugins) and integrated solution (e.g., VM ISO).

MS4:

a) Integration in the DICE IDE of the modelling abstractions (DICE profiles) introduced in WP2

b) Integration of DICE IDE with the methodological workflow defined in T2.4

c) Integration of metrics from WP3-5 tools

d) Implementation of a suitable release format for the individual tools (e.g., Eclipse plugins) and integrated solution (e.g., VM ISO).

## D.2. Methodology and Data-Aware Models (WP2 Technical Requirements)

### D.2.1 Overview

WP2 aims at developing systematic methods for the design as well as representational support models to aid the Model-Driven Engineering of DIAs. As established and dictated by the OMG standard guidelines for Model-Driven Engineering (as represented by the MDA standard) we decided to arrange the DICE methodology and supporting profiles framework to accommodate the threefold nature of the Model-Driven Engineering paradigm. More in particular the DICE methodology and profile will

support software architects and designers at three levels of abstraction: (a) the Data-Intensive Platform-Independent level (i.e., supporting the definition of a DPIM model), essentially allowing designers to state requirements for their Data-Intensive endeavour; (b) the Data-Intensive Technology-Specific level (i.e., supporting the definition of a DTSM model), essentially allowing designers/architects and developers to brainstorm on technological implementation alternatives with the support of ready-made DICE-supported technologies (e.g., Hadoop, Spark, Storm, etc.); (c) finally, the Data-Intensive Deployment-Specific level (i.e., supporting the definition of a DDSM model), essentially allowing, on one side, software architects to brainstorm on the deployment needs and possibilities of the realised DIA and, on the other side, allowing software developers to deploy successfully the Data-Intensive application by means of TOSCA-enabled blueprints.

As previously mentioned, the most likely stakeholders, also envisioned in our methodology, are: (a) software architects, since they are the main targets for heavy-weight modelling and model analysis technologies in general; (b) software developers, since they are the main potential stakeholders to benefit from automated and deployment-enabled modelling facilities.

Although it is not strictly part of this WP to develop an extensive modelling tool and tool-support beyond configuring ready-made Model-Driven Engineering Facilities (such as EMF or Papyrus), we are considering to provide ad-hoc, self-contained and model-driven tool-support to the DICE profile, in line with the DICE methodology, along two possible support lines:

a) *DICE-Profile Modeller:* this tool shall enable the modelling of DPIM and DTSM models and model views, in line with their future transformation towards a DDSM model, ready to be translated in a TOSCA-ready format. Initially the tool may consist of auto-generated model instantiation facilities, e.g., following the EMF reflective modelling we are considering to adopt as part of the EMF/ATL + Eclipse Papyrus toolchain;
b) *DICE-Profile Deployability Checker:* this tool shall assist the deployability transformation (e.g., semi-assisted transformation from a DTSM model into a DDSM model) and deployability refinement (i.e., the analysis of multiple deployable configurations based on known stakeholders' concerns) eventually leading to a TOSCA-ready recipe. Initially the tool may consist of mapping rules between DICE profile notations and views with TOSCA standard version 1.0;

The DICE methodology assumes that the ARCHITECT may be supported in at least three ways:

- When ARCHITECT may not know the best fitting technologies to be used for their DIA needs, DICE profile and methodology will help in providing multiple alternatives, e.g., as a brainstorming basis but also as a basis for analysis and comparison, e.g., instrumental to applying approaches such as the SEI's Architecture Tradeoff Analysis Method (ATAM) [31];
- When ARCHITECT already aware of technological needs may start unassisted at the DPIM layer and proceed by modelling requirements at that layer; DICE profile and methodology will help in refining said requirements providing valid and consistent technological packages for effectively deploying the known technological needs (e.g., ready-made packages for Hadoop, Spark, Storm, etc.);
- When ARCHITECT with stringent deployment or budgetary constraints may require assistance in striking a balance with requirements specified at the DPIM layer and implementation possibilities at the DTSM and DDSM layer; DICE profile and methodology will help by providing methods in which analyses can be carried out to assist and quantify the tackling of said constraints;

 

The scenarios above were used to devise constructs and views to support the specification and the application of the DICE profile in practice. Said scenarios will also be used for validating that the DICE profile meets the theoretical assumptions upon which it was based.

### D.2.2 Requirement Analysis

Tables from Table 47 to Table 56 contain specifications for technical requirements developed in the process of internal DICE consultations.

**Table 47: Requirement for the DICE methodological paradigm.**

| | |
|---|---|
| **ID:** | D.94. R2.1 |
| D.95. **Title:** | DICE Methodological Paradigm |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DICE profile and methodology shall support the incremental specification of Data-Intensive Applications (DIAs) following a Model-Driven Engineering approach, as defined in standard OMG guidelines. |
| **Rationale:** | The DICE profile and Methodology both follow the MDE paradigm and the models envisioned thereto. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 48: Requirement for the origin of the abstraction layer.**

| | |
|---|---|
| **ID:** | D.96. R2.2 |
| D.97. **Title:** | Abstraction Layer Origin |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | Every abstraction layer (namely, DPIM, DTSM and DDSM) of the DICE profile MUST stem from UML. |
| **Rationale:** | The DICE profile shall mimic the standard assumptions behind Model-Driven Engineering, including the separation of concerns across three disjoint but related layers (Platform-Independent, Platform-Specific and Deployment-Specific). |
| **Supporting material:** | [5] |
| **Other comments:** | N/A |

**Table 49: DICE Constraints Specification Requirement**

| | |
|---|---|
| **ID:** | D.98. R2.4 |
| D.99. **Title:** | DICE Constraints Specification |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |

| | |
|---|---|
| **Description:** | The DICE Profile MUST allow definition of values of constraints (e.g., maximum cost for the DIA), properties (e.g., outgoing flow from a Storage Node) and stereotype attributes (batch and speed DIA elements) using the MARTE VSL standard. |
| **Rationale:** | VSL is a part of the MARTE standard dedicated specifically to the (semi-)formal specification of quality attribute values across profiles for quality properties definition and their analysis. DICE shall make use of these modelling facilities inherited from MARTE |
| **Supporting material:** | [30] |
| **Other comments:** | N/A |

**Table 50: Requirement for the DICE profile technology-specific constraints.**

| | |
|---|---|
| **ID:** | D.100. R2.10 |
| D.101. **Title:** | DICE Profile Tech-Specific Constraints |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DICE Profile MUST define structural and behavioral constraints typical in targeted technologies (e.g., Hadoop, Storm, Spark, etc.). |
| **Rationale:** | Many technologies have different possible structural or behavioral concerns and consequent constraints. These must be explicitly supported across the DICE profile. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 51: Requirement for the DICE profile separation-of-concerns.**

| | |
|---|---|
| **ID:** | D.102. R2.11 |
| D.103. **Title:** | DICE Profile Separation-of-Concerns |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DICE Profile MUST use packages to separately tackle the description of targeted technologies in the respective profile abstraction layers (e.g., DTSM and DDSM). Said packages shall be maintained consistently |
| **Rationale:** | Separation of concerns is one of the basic principles behind model-driven engineering and related technologies. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 52: Requirement for the data-intensive Quality of Service (QoS).**

| | |
|---|---|
| **ID:** | D.104. R2.4 |

| D.105. **Title:** | Data-Intensive QoS |
|---|---|
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DPIM MUST be generic enough so as not to require any specialization, e.g., for domain-specific DIAs. Conversely, the DPIM layer shall contain generic constructs with which to instantiate all possible DIAs together with all relevant QoS and Data-intensive analyses. |
| **Rationale:** | The first layer of abstraction of the DICE profile shall at least address the quality annotations as well as the safety & privacy characteristics (cfr. WP3) needed to further the design of a DIA in a QoS-Aware way. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 53: Requirement for the DICE topologies.**

| **ID:** | D.106. R2.9 |
|---|---|
| D.107. **Title:** | DICE Topologies |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DTSM layer MUST support the definition of Technology-specific DIA Topologies (e.g., Namenode-Datanode-SecondaryNamenode vs. Master-Region-Zookeeper, etc.). |
| **Rationale:** | Similarly to other modelling technologies (e.g., TOSCA) DICE shall support the definition and design of DIA as topologies of connected services/components/nodes. Given that different technologies require different topologies, this concern is especially relevant at the DTSM layer and shall be supported as such. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 54: Requirement for DICE extension points.**

| **ID:** | D.108. R2.7 |
|---|---|
| D.109. **Title:** | DICE Extension-Points |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DTSM MUST include extension facilities. These facilities shall be used to “augment” the DICE profile with technologies beyond the DICE project assumptions (e.g., Storm, Spark, Hadoop/MR, etc.). Similarly, every technological space embedded within the DICE profile shall exist in the form of such extensions, e.g., as conceptual packages (at the DTSM layer) and refined implementation-specific packages (at the DDSM layer). |
| **Rationale:** | Because Big-Data Applications and their domain are extremely rich with technology and very highly evolving, the DICE profile shall define extension points where possible, i.e., points where further technologies may be specified and "plugged-in" within the profile |

| | |
|---|---|
| | itself. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 55: Requirement for the DICE deployment-specific views.**

| | |
|---|---|
| **ID:** | D.110. R2.12 |
| D.111. **Title:** | DICE Deployment Specific Views |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DDSM layer MUST support the definition of an Actionable deployment view (TOSCA-ready): this view offers conceptual mappings between the technological layers defined in the DTSM and concepts in the TOSCA meta modeling infrastructure such that one-way transformation between the technological layer and the actionable deployment view is possible. |
| **Rationale:** | Because the instantiation for execution of different technologies may be optional and supported via TOSCA, the DDSM layer shall allow designers to use or not use the TOSCA-based deployment model for execution. This requirement assumes that further standards may be presented beyond TOSCA in the future. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 56: IDE support to the use of profile requirement.**

| | |
|---|---|
| **ID:** | D.112. R2.17 |
| D.113. **Title:** | IDE support to the use of profile |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The DICE IDE MUST support the development of DIA exploiting the DICE profile and following the DICE methodology. This means that it should offer wizards to guide the developer through the steps envisioned in the DICE methodology |
| **Rationale:** | An adoption of the DICE profile not supported by a user friendly IDE can be quite cumbersome and limit the benefits of our approach. The more the IDE is user friendly the more the potential of a positive impact of the DICE profile on practitioners increases |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 57: DICE Analysis Focus.**

| | |
|---|---|
| **ID:** | D.114. R2.18 |
| D.115. **Title:** | DICE Analysis Focus |
| **Priority of** | Must have |

| | |
|---|---|
| **accomplishment:** | |
| **Type:** | Domain Assumption |
| **Description:** | The DICE profile and its design shall work under the assumption that their focus of application is limited to providing facilities and methodological approaches to support those properties that are relevant to perform analysis (e.g., for fine-tuning, load-estimation, etc.), testing (e.g., for run-time verification and adaptation towards continuous integration), monitoring (e.g., for flexible continuous improvement, etc.). |
| **Rationale:** | Being an emerging field, DIAs design and analysis may entail a great variety of possible analyses and venues for research and development. Our assumption however, is that DIAs are either modelled to analyse and estimate their properties, test these estimations in practice or monitor their auctioned behaviour for continuous improvement. Other endeavours, however connected to DIAs, are out of the scope of DICE. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

### D.2.3 Discussion of the requirements

The requirements outlined above stem from a twofold analysis.

First, we conducted a systematic study of three top technologies, namely, Hadoop, Storm and Spark, highlighting: (a) learning-curve and conceptual needs; (b) relevant issues to be tackled in average application scenarios as well as concerns emerging in more complex and elaborate scenarios; (c) relevant configuration tweaks and overrides. This study was loosely based on a systematic mapping protocol (freely available) and was targeted at top tutorials and support sites related to technologies involved.

Second, requirements were refined using input and focus-groups with case-study partners. This refinement campaign emerged into an additional set of requirements plus a refinement of previously existing ones, taking into account: (a) newly emerging stakeholders' concerns for modelling and automated tool-support; (b) newly emerging technological aids (e.g., automated consistency checks and ad-hoc technology configuration constraints) that the DICE profile could provide; (c) new perspectives for the intended usage of the DICE profile by case-study partners and relevant stakeholders.

As a result, most if not all the requirements reflect, either: (a) peculiar overrides or critical configurations that the DICE profile and methodology could support in making explicit or even automating, thus reducing the learning-curve needed to familiarise and operate with supported technologies; (b) peculiar views highlighting relevant stakeholder-specific (e.g., for architects, designers and developers) aspects of supported technologies - these aspects could discriminate the usage of said technologies in the first place; (c) future evolution and DevOps trends represented by a continuous improvement of tools supported by data-intensive applications developed according to the DICE profile - for example, profiles that envision the extension of the DICE profile with additional technologies (e.g., requirement MR2.7) or requirements for designing certain technologies following the "core-subset packages" meta-modelling pattern (e.g., requirement MR2.4) are examples of said set of requirements.

From a validation point of view, the requirements will be subject to inquiry using standard automated consistency analysis applied to meta-modelling facilities. Eclipse Modelling Facilities (e.g., EMF) already provide said basic meta-model consistency validation techniques - these allow to verify if produced meta-models can actually be reproduced into re-deployable profiles. An extensive body of

 

knowledge into conceptual modelling and model validation facilities exists that could avail said validation campaign.

From a risks-management perspective, the design of the DICE profile will regularly undergo technology consistency checks, to make sure the profile covers all constructs to tentatively support new or evolving DIA-specific technologies (E.g., evolutions of Kafka, TEZ, etc.). Said technological consistency checks will take place every time DICE profile technological extensions are proposed and discussed. In addition, risks connected to tool-support will be mitigated considering possible alternative implementation venues while carrying out decided pursuits. For example, we will consider alternative tool-support options while thoroughly exploring the usage of Eclipse Papyrus and related tool-support. Finally, risks connected to misalignment with other work-packages will be mitigated by regularly involving other WP-leaders in profile-critical alignment decisions.

### D.2.4 Initial roadmap

MS2: Architecture Definition:

- First consolidation of DICE profile;
- First definition of DICE methodology;
- Preliminary validation of DICE profile vs. Methodology and vs. WP3 alignment;
- Initial technological support workout;
- Release of Deliverable D2.1.

After month 12, the DICE profile will be consolidated incrementally based on emerging needs (e.g., new constructs emerging from augmented partner or market needs). Particular care will be devoted to preserving the internal and external consistency of the DICE profile so as not to compromise its internal and external validity as a research and practical contribution. Also, particular care will be devoted to maintaining wherever possible the alignment between evolving versions of supported technologies. This notwithstanding, the application of the DICE methodology and supporting profile will make clear that their design and application are tied to the specific technologies and versions considered during the design and development of the DICE profile and methodology.

## D.3. Data-Aware Quality Analysis (WP3 Technical Requirements)

### D.3.1 Overview

The data-aware quality analysis aims at assessing quality requirements for DIAs (Data Intensive Applications) and at offering an optimized deployment configuration for the application. The assessment starts from the DIA UML design, which includes not only the functionality of the system but the quality requirements and corresponding parameters. In particular, the UML design will be produced using the tools developed by WP2, i.e., such design is expressed at three abstraction levels (DPIM, DTSM and DDSM) and annotated according to the DICE profile. Regarding quality requirements for assessment we are focussed on performance, reliability and safety. For the optimization the DDSM level will be the input target.

For accomplishing the aforementioned objectives WP3 will develop the following tools:

- TRANSFORMATION_TOOLS - The transformation tools will be developed according to the model-driven software engineering paradigm, i.e., using model to model (M2M) transformation techniques. These tools take as input a UML DICE-profiled design representing a DIA and will produce suitable formal models, depending on the kind of analysis to be carried out (e.g., stochastic Petri nets, temporal logics, and possibly other formalisms).

- SIMULATION_TOOLS - The simulation tools will take as input the models produced by the TRANSFORMATION_TOOLS, such models apply mainly at DPIM and DTSM abstraction levels. The simulation tools, using well-established analytical and simulation techniques, will validate the performance and reliability requirements of the DIA, requirements originally expressed in the UML DIA design.

- VERIFICATION_TOOLS - The verification tools aim at checking the so-called safety properties for the DIA (for example, that the sizes of queues do not exceed certain thresholds, or that the time to process data does not exceed desired bounds). They take as input a suitable formal representation of the UML DIA design, and the property to be analyzed, and determine whether the property holds for the modeled system or not. The verification tools can be applied at different levels of abstraction, and in particular at DPIM and DTSM levels.

- OPTIMIZATION_TOOLS - The optimization tools apply at DDSM level. According to different deployments, these tools will evaluate the corresponding Petri net models for deciding which deployment, among those satisfying QoS, is the optimal one, say regarding to a predefined criteria (cost versus usage).

These tools will be useful for the project teams optimizing the deployment, validating the DIA design and assessing the DIA quality requirements. Therefore, these tools will assess the proposed DIA design, at different abstraction levels, for accomplishing the quality requirements established for the DIA. The tools will provide feedback on the adequateness of the application's design for meeting efficiency, reliability and safety requirements. In particular, the following stakeholders will use the tools directly:

- QA_ENGINEER - The application quality engineer will use the SIMULATION_TOOLS and VERIFICATION_TOOLS through the DICE IDE. For a given UML design a dashboard baseline will present results for each requirement to be assessed. For example, at DPIM level a quality requirement (e.g. response time) will be highlighted as fulfilled or not.
- ARCHITECT - The designer of the DIA uses the OPTIMIZATION_TOOLS through the DICE IDE. For a given DDSL the tool will output the number of resources and architecture at minimum cost provided that the quality requirements are fulfilled.
- SIMULATION_TOOLS, VERIFICATION_TOOLS and OPTIMIZATION_TOOLS - The TRANSFORMATION_TOOLS are not used by human actors directly but internally for the rest of the tools in this WP. These tools, as actors, need to invoke the transformation tools for getting the model to be evaluated.

### D.3.2 Requirement Analysis

Tables from Table 58 to Table 64 contain specifications for technical requirements developed in the process of internal DICE consultations.

**Table 58: Requirement for the Model to Model (M2M) transformation.**

| | |
|---|---|
| **ID:** | D.116. R3.1 |
| D.117. **Title:** | M2M Transformation |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The TRANSFORMATION_TOOLS MUST perform a model-to-model transformation taking the input from a DPIM or DTSM DICE annotated UML model and returning a |

| | |
|---|---|
| | formal model (e.g. Petri net model or a temporal logic model). |
| **Rationale:** | This is the main functionality needed to perform simulations and verification activities |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 59: Requirement for the annotations.**

| | |
|---|---|
| **ID:** | D.118. R3.2 |
| D.119. **Title:** | Taking into account relevant annotations |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The TRANSFORMATION_TOOLS MUST take into account the relevant annotations in the DICE profile (properties, constraints and metrics) whether related to performance, reliability, safety, privacy, and transform them into the corresponding artifact in the formal model |
| **Rationale:** | N/A |
| **Supporting material:** | A property is a characteristic of a system's element (e.g. transfer rate of a disk) |
| **Other comments:** | N/A |

**Table 60: Requirement for the generation of traces from the system model.**

| | |
|---|---|
| **ID:** | D.120. R3.7 |
| D.121. **Title:** | Generation of traces from the system model |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The VERIFICATION_TOOLS MUST be able, from the UML DICE model a system, to show possible execution traces of the system, with its corresponding time stamps. This sequence SHOULD be used by the QA_ENGINEER to determine whether the system model captures the behavior of the application or not, for model validation purposes. |
| **Rationale:** | One way to validate whether the actual system has been sufficiently captured by the model is to produce traces of the model, and see whether they are consistent with the expected behavior of the system. |
| **Supporting material:** | N/A |
| **Other comments:** | The checking of whether the trace is "reasonable" or not can only be done by the user, it cannot be done automatically by the tool. In fact, the tool will always produce traces that are compatible with the system model; the question is whether the system model is reasonable or not. |

**Table 61: Requirement for the cost/quality balance.**

| | |
|---|---|
| **ID:** | D.122. R3.8 |

| D.123. **Title:** | Cost/quality balance |
|---|---|
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The OPTIMIZATION_TOOLS will minimize deployment costs trying to fulfill reliability and performance metrics (e.g., map reduce jobs execution deadlines) |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 62: Requirement for the Service Level Agreement (SLA) specification and compliance.**

| **ID:** | D.124. R3.10 |
|---|---|
| D.125. **Title:** | SLA specification and compliance |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | All three tool types, SIMULATION_TOOLS, VERIFICATION_TOOLS and OPTIMIZATION_TOOLS MUST permit users to check their outputs against SLA's included in UML model annotations. If an SLA is violated the tools will inform the user. |
| **Rationale:** | The DICE Profile inherits from MARTE how to specify non-functional properties, i.e., how to specify SLA's as requirements. Then, the WP3 TOOLS must read these SLA's and compute in the formal model results that help to verify them. For example, the UML model could specify a performance requirement of 1 sec. as the response time of a given service. Then, the SIMULATION_TOOLS must analyze the Petri net performance model to tell the response time of such service, according to the current model input parameters. The tool could highlight those SLA's that are not fulfilled. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 63: Requirement for the optimisation timeout.**

| **ID:** | D.126. R3.11 |
|---|---|
| D.127. **Title:** | Optimization timeout |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | The OPTIMIZATION_TOOLS MUST explore the design space and accept the specification of a timeout and return results gracefully when this timeout is expired |
| **Rationale:** | The user should not be waiting for a response indefinitely |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 64: Requirement for the white/black box transparency.**

| | |
|---|---|
| **ID:** | D.128. R3.13 |
| D.129. **Title:** | White/black box transparency |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | For the TRANSFORMATION_TOOLS and the SIMULATION_TOOLS there will be no difference between white box and black box model elements. |
| **Rationale:** | In both cases, black or white model elements, the processes remain the same. First, annotations will come from well-known sources for some components while others will be guessed by the ARCHITECT. Later, the reasoning about the system through the formal model will lead to improvements of some attributes, parameters or constraints. Finally, the analysis of the logs coming from WP4 will provide information from real application execution. It does not matter whether the improved parameter refers to a black box model element (e.g., MP job or any other Hadoop framework executed in the cloud) or an ad hoc well known algorithm modeled as a white-box component. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

### D.3.3 Discussion of the requirements

The TRANSFORMATION_TOOLS have three main requirements: R3.1, R3.2 and R3.13. Obviously none of them refers to the DICE IDE since they are just internally used by the rest of the tools developed in WP3. Regarding R3.1, it highlights the need of these tools for constructing the SIMULATION_TOOLS and VERIFICATION_TOOLS. On the other hand, R3.2 is meant for the interface of the WP3 tools with WP2 tools, i.e., the UML designs, such design gather both, the functional requirements and the quality requirements, properties and constraints.

Requirements R3.7 and R3.10 describe how the SIMULATION_TOOLS and VERIFICATION_TOOLS assess the quality requirements defined in the UML designs. Requirements R3.8, R3.10 and R3.11 describe how the OPTIMIZATION_TOOLS will select the deployment that minimizes costs. Finally R3.13 describes how the tools manage the differences between white and black box models.

### D.3.4 Initial roadmap

MS2: Architecture Definition:

- Initial version of the M2M transformation (functional for a UML diagram);
- Prototype of the M2M transformation (functional for a UML diagram);
- Implementation of the first version of the IDE architecture;
- Functional prototype of the SIMULATION_TOOLS and VERIFICATION_TOOLS;
- Release of Deliverables D3.2 and D3.5.

After month 12:

- Development of the initial version of the OPTIMIZATION_TOOLS will be accomplished;
- For TRANSFORMATION_TOOLS we need to consolidate the transformation of the UML activity diagram and get an initial version of the transformation of the sequence diagram;

 

- For the DICE IDE concerning WP3 tools we need to include support for at least two metrics;
- For SIMULATION_TOOLS we need to implement the analysis of the models for the selected requirements;
- For VERIFICATION_TOOLS we will expand on the set properties to be analyzed, and on the set of considered kinds of components.

## D.4. Iterative Quality Enhancement (WP4 Technical Requirements)

### D.4.1 Overview

#### *D.4.1.1. WP technical objectives*

The goal of this WP is to define tools and techniques to support the iterative improvement of quality characteristics in data-intensive applications obtained through feedback to the developers. This feedback will guide architectural design changes of the application being developed. This WP will design and implement the monitoring platform that will collect and store traces and logs produced during the execution of data-intensive applications. Relying on monitoring platform, techniques based on classification, statistical analysis and trace analysis using formal methods to detect quality incidents and quality anti-patterns from the data collected at runtime will be developed. These analyses will be correlated to the application's design models in order to provide visual and report feedback to the developer on the quality offered by the application at runtime and thus guide quality evolution.

#### *D.4.1.2. WP stakeholders.*

DEVELOPER / ARCHITECT - The ENHANCEMENT_TOOLS and ANOMALY_TRACE_TOOLS developed in this package are specifically designed to help developers and software architects in finding bottlenecks in their data-intensive application, in adjusting the design to better answer functional and non-functional requirements (together with WP2 and WP3 tools).

QUALITY ENGINEER - The MONITORING_TOOLS allows quality engineers to collect and view performance metrics for data-intensive applications. They can also benefit of ANOMALY_TRACE_TOOLS to get insights on the performance issues of data-intensive applications.

SYSTEM ADMINISTRATOR - The MONITORING_TOOLS offers a centralized repository for monitoring data that can help system administrators in the monitoring and tuning of the infrastructure

#### *D.4.1.3. Tools that will be created in the WP*

Monitoring and data warehousing platform (MONITORING_TOOLS) - retrieves monitoring data produced by the data-intensive application, stores it and provides an easy to use interface to query it. Monitoring data refers to either (i) logs produced by Big Data frameworks (e.g. Apache Hadoop (MapReduce and HDFS), Apache Spark, Apache Storm, Cassandra) or (ii) logs produced by an application modelled using DICE profile (R4.1), provenance and versioning information being recorded in both cases (R4.2, R4.2.1, and R4.2.2). These are application-specific logs and can be either directly generated by the application, or produced by (byte-code) instrumented wrappers. Clients of MONITORING_TOOLS will be provided with a REST API to update/query the data warehouse (R 4.7). An OSLC Performance Metrics Specification compliant interface will allow clients to consume monitoring data from the platform (R4.35).

Quality incident detection and trace checking tools (ANOMALY_TRACE_TOOLS) perform analysis of monitoring data collected by the MONITORING_TOOLS in order to empower developers (architects) with information about the quality of the application. Given a trace of the events, ANOMALY_TRACE_TOOLS will check whether that trace is compatible with the desired safety and privacy properties and will offer feedback to the developer regarding the application. They will also

 

apply unsupervised and supervised machine learning algorithms on monitoring data in order to detect deviations in application quality.

Enhancement tools (ENHANCEMENT_TOOLS) are the main tools to close the loop between runtime (monitoring data) and design time (models and tools). They will implement the correlation between monitoring data and UML models (WP2), will infer initial input parameters for simulation and optimization tools (WP3) and will breakdown resource consumption into atomic components. The integration of ENHANCEMENT_TOOLS with SIMULATION and OPTIMIZATION tools is implemented through the UML models, which will be updated by the ENHANCEMENT_TOOLS.

### D.4.2 Requirement Analysis

Tables from Table 65 to Table 76 contain specifications for technical requirements developed in the process of internal DICE consultations.

**Table 65: Requirement for the monitoring data extraction.**

| | |
|---|---|
| **ID:** | D.130. R4.3 |
| D.131. **Title:** | Monitoring data extractions |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | MONITORING_TOOLS MUST perform monitoring data pre-processing (extraction) before storing the data in the data warehouse in order to facilitate usage by other tasks. |
| **Rationale:** | Different actors have different /expectations from the monitoring data stored in DW, such that aggregations over time periods, different granularities etc. |
| **Supporting material:** | N/A |
| **Other comments:** | Pre-processing refers to extraction and validation operations in order to extract (parse) log files and validate the obtained data (e.g. valid email address, valid IP address etc.). |

**Table 66: Requirement for access restriction to the monitoring data.**

| | |
|---|---|
| **ID:** | D.132. R4.6 |
| D.133. **Title:** | Monitoring data access restrictions |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The data warehouse MUST provide the ability to prevent unauthorised access to the monitoring data. |
| **Rationale:** | The monitored data may contain sensitive and private data. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 67: Monitoring visualisation requirement.**

| | |
|---|---|
| **ID:** | D.134. R4.8 |
| D.135. **Title:** | Monitoring Visualization |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | MONITORING_TOOLS SHOULD support interactive visualization of monitoring data |
| **Rationale:** | Visualization will give human actors an initial overview over the monitoring data available for their APPLICATION. |
| **Supporting material:** | N/A |
| **Other comments:** | This will reuse an existing Web-based visualization tool available for the data warehouse platform (e.g. Kibana Web tool for Elastic platform) |

**Table 68: Requirement for the refactoring methods.**

| | |
|---|---|
| **ID:** | D.136. R4.14 |
| D.137. **Title:** | Refactoring methods |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | Once correlation between anomalies in runtime and anti-patterns has been detected, the ENHANCEMENT_TOOLS SHOULD propose methods for refactoring the design leveraging parameters extracted from the traces. |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 69: Enhancement tools version difference requirement.**

| | |
|---|---|
| **ID:** | D.138. R4.16 |
| D.139. **Title:** | Enhancement tools version difference |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | The ENHANCEMENT_TOOLS COULD compare two versions of the application to identify relevant changes. |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 70: Requirement for the parameterization of simulation and optimization models.**

| **ID:** | D.140. R4.19 |
|---|---|
| D.141. **Title:** | Parameterization of simulation and optimization models. |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The ENHANCEMENT_TOOLS MUST extract or infer the input parameters needed by the SIMULATION_TOOLS and OPTIMIZATION_TOOLS to perform the quality analyses. |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | Input parameters inferred as a result of this requirement may be completed by additional parameters provided by end-user or other tools (e.g. configuration recommender). |

**Table 71: Requirement for the time-based ordering of monitoring data entries.**

| **ID:** | D.142. R4.22 |
|---|---|
| D.143. **Title:** | Time-based ordering of monitoring data entries |
| **Priority of accomplishment:** | Must have |
| **Type:** | Domain Assumption |
| **Description:** | Monitoring data MUST support the reconstruction of a sequence of events and the identification of the time when things occurred (for example a consistent timestamp in a distributed system) |
| **Rationale:** | While in general data is application-dependent, for running trace checking it is important that data is time-based ordered. |
| **Supporting material:** | N/A |
| **Other comments:** | In case of data collected from multiple nodes of a distributed system, MONITORING_TOOLS must ensure data is consistently ordered when providing answer to actors' queries. |

**Table 72: Requirement for the data size trends.**

| **ID:** | D.144. R4.23 |
|---|---|
| D.145. **Title:** | Data size trends |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | The MONITORING_TOOLS and ENHANCEMENT_TOOLS SHOULD capture the growth in the data size for the APPLICATION. |
| **Rationale:** | Data size cannot be predicted at design time, the APPLICATION behavior usually depend on it since the DB gets slower the larger the data gets. |
| **Supporting material:** | N/A |

| | |
|---|---|
| **Other comments:** | N/A |

**Table 73: Requirement for the propagation of changes/automatic annotation of UML models.**

| | |
|---|---|
| **ID:** | D.146. R4.27 |
| D.147. **Title:** | Propagation of changes/automatic annotation of UML models |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | ENHANCEMENT_TOOLS MUST be capable of automatically updating UML models with analysis results (new values) |
| **Rationale:** | Increase efficiency of iterative enhancement process |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 74: Requirement for the loading of safety and privacy properties.**

| | |
|---|---|
| **ID:** | D.148. R4.28 (R4IDE6) |
| D.149. **Title:** | Safety and privacy properties loading |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The ANOMALY_TRACE_TOOLS MUST allow the DEVELOPER/ARCHITECT to choose and load the safety and privacy properties from the Model of the application described through the DICE profile |
| **Rationale:** | The properties to be analyzed are application-dependent, and they must come from somewhere in the DICE model of the application. The user knows what properties are to be monitored, so he/she should select those that most interest him/her |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 75: Requirement for the monitoring of safety and privacy properties.**

| | |
|---|---|
| **ID:** | D.150. R4.30 |
| D.151. **Title:** | Safety and privacy properties monitoring |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | The ANOMALY_TRACE_TOOLS MUST be able to check, given a trace of the events of interest of the application, whether that trace is compatible with the desired safety and privacy properties |
| **Rationale:** | This is the main functionality of the trace checking tool |
| **Supporting material:** | N/A |

| | |
|---|---|
| **Other comments:** | The check is performed off-line, i.e., in batch mode (a trace is retrieved from the DW, then analysed by the trace checking tool) |

**Table 76: Requirement for the correlation between data stored in the DW and DICE UML models.**

| | |
|---|---|
| **ID:** | D.152. R4.32 |
| D.153. **Title:** | Correlation between data stored in the DW and DICE UML models |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | There MUST be a way to link the information that is stored in the data warehouse with the features and concepts of the DICE UML models (operations, attributes, objects, etc.) |
| **Rationale:** | The properties analyzed by the ANOMALY_TRACE_TOOLS through trace checking are expressed in terms of the elements of the DICE UML model. Hence, to run the trace checking the events stored in the data warehouse must be correlated with what is described by the UML model. A similar need arises for the ENHANCEMENT_TOOLS, which need to reason on the UML model to infer input parameters. |
| **Supporting material:** | N/A |
| **Other comments:** | It is unclear which component should bear responsibility of this fundamental part. Would it be ENHANCEMENT_TOOLS or a dedicated component? |

### D.4.3 Discussion on the requirements

Technical requirements at WP4 level reflect the needs of other technical WPs (WP2, WP3) in DICE, augmented with the needs expressed by software developers, architects and quality engineers. Thus, a first set of requirements defines the expectations with respect to the monitoring platforms:

- scalability to be able to face the large volumes, and high speed of monitoring data produced in data-intensive applications,
- ETL (Extract-Transform-Load), visualization and controlled access to the repository
- REST interface to update / query the repository, OSLC-compliant interface
- ability to consistently handle timestamp in distributed environments
- ability to handle trends in monitoring data

These requirements are then exploited by ENHANCEMENT_TOOLS, ANOMALY_TRACE_TOOLS, SIMULATION_TOOLS, or OPTIMIZATION_TOOLS, tools that will query the repository to retrieve runtime data they need.

The correlation between runtime environment and design time models is supported by ENHANCEMENT_TOOLS, which will update the UML models with relevant information inferred from monitoring data, for example input parameters needed by the simulation and optimization tools. Iterative quality enhancement process will be further supported by ENHANCEMENT_TOOLS by offering a breakdown of resource consumption into its atomic components and displaying this in design time models. In order to achieve this, DICE application's bytecode may need to be instrumented, or code probes will be inserted at code generation time in order to log the required monitoring information in the repository.

Further quality insights will be derived by the ANOMALY_TRACE_TOOLS, such as whether the trace is compatible with the desired safety and privacy properties, and anomalies in quality using unsupervised and supervised techniques.

The execution of DICE application and its tests, which are responsible for the creation of runtime information, will be controlled by CI_TOOLS developed in WP5. They will also maintain versioning information needed by the MONITORING_TOOLS.

### D.4.4 Initial roadmap

Below is the per task work allocation for the first year (up to M12), plus prospects beyond it. For the first year we are planning to introduce support for different technologies using an iterative approach. Thus, the first platform we target is Apache Oryx 2, comprising Hadoop, Kafka and Spark, subsequently followed by other technologies (Storm, NoSQL etc). Details are given in the Milestone-based list below.

MS2 - Architecture definition (M12), as follows:

- Initial deployment of monitoring platform (ELK standing for Elastic Search, Logstash and Kibana) on Flexiant Cloud Orchestrator
- Oryx2 platform: deployment, relevant metrics collection and integration in monitoring DW, integration with DICE profile, query in JSON format
- Deployment of Apache Storm: deployment, relevant metrics monitoring and integration in monitoring DW, integration in DICE profile, query in JSON format
- Monitoring and data warehousing initial version
- Investigate the anomaly detection in Oryx2 applications using unsupervised methods
- Investigate the correlation between runtime, monitoring data and design-time models for Oryx2

MS3 - DICE Tools Initial Release (M18), will add support for monitoring and anomaly detection for other technologies, such as NoSQL datastores (e.g. MongoDB, Apache Cassandra). This MS will also include work on safety properties and data trends analysis.

MS4 - Integrated Framework. First release (M24), will mainly be focused on integration of monitoring platform tools with demonstrators' case studies.

## D.5. Deployment and Quality Testing (WP5 Technical Requirements)

### D.5.1 Overview

The Deployment and quality testing WP aims at developing tools, which help put the DICE tools users' application to the actual environment and evaluate its runtime. The DDSM models describe in a low-level way the whole application that the users want to deploy, test and use. The tools created in the Deployment and quality testing WP take this model as a TOSCA blueprint, then they set up the target (test or pre-production) environment, deploy the application, and run the application against a number of tests to obtain feedback on the application's behaviour.

The tools from this WP will be developed with a DevOps ecosystem in mind, and will consist of the following tools:

- CI_TOOLS - Continuous Integration (and Deployment) tools will represent the interface through which the users will be able to interact with the deployment and testing process.
- DEPLOYMENT_TOOLS - Deployment tools will include the orchestration tools for executing TOSCA compliant blueprints, and a configuration recommender tool, which will help the user decide on the best set of parameters for certain conditions.

- QTESTING_TOOLS will subject the deployed application to various levels of input data feeds to be able to assess the reaction of the application, and to obtain quantitative feedback on the performance and reliability of the application.
- TESTBED - The environment of choice for testing and validating the DICE tools will be the Flexiant Cloud Orchestrator.

These tools will be highly useful for the project teams, who do planning and develop data intensive applications. The tools will produce deployment configuration recommendations, and perform the actual deployment to a functioning application, which will be ready to be used. They will provide also the feedback on the application's level of efficiency and reliability. In particular, the following stakeholders will use the tools directly:

- ARCHITECT - System designer/architect may use the tools to verify the bare minimum skeleton of the application performs as expected and required.
- DEVELOPER - The tools are invaluable to System developers/programmers for continuously deploying and testing the functionality of the application as it is being developed.
- QA_TESTER - Quality assurance expert takes advantage of the results of the quality tests and indications of the performance history across versions of the application
- ADMINISTRATOR - System manager is mostly involved in the setting up of the DICE tools in the target environment. May also decide on the acceptance of the application builds based on the feedback from the quality testing tools.

### D.5.2 Requirement Analysis

Tables from Table 65 to Table 76 contain specifications for technical requirements developed in the process of internal DICE consultations.

**Table 77: Requirement for the continuous integration and versioning.**

| | |
|---|---|
| **ID:** | D.154. R5.34 |
| D.155. **Title:** | Continuous Integration records and versioning |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | CI_TOOLS MUST record the results of each test, mapping them to the version number. |
| **Rationale:** | Versioning of the application marks the progress of development and enables dependencies. Associating version numbers to outputs is crucial for performance analysis and history bookkeeping. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 78: Requirement for the graphical user interface for continuous integration.**

| | |
|---|---|
| **ID:** | D.156. R5.35 |
| D.157. **Title:** | Graphical user interface for Continuous Integration |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |

| | |
|---|---|
| **Description:** | CI_TOOLS SHOULD offer a dashboard to consolidate the view of the application deployment, and the access SHOULD be restricted to only the authorized users. |
| **Rationale:** | A graphical user interface dashboard is where the users find the data of the latest and past quality testing tools runs, but this information is sensitive for access. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 79: Requirement for the quality testing scope.**

| | |
|---|---|
| **ID:** | D.158. R5.36 |
| D.159. **Title:** | Quality Testing basic scope |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | QTESTING_TOOLS MUST test the application for efficiency and reliability. |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 80: Requirement for the extended quality testing scope.**

| | |
|---|---|
| **ID:** | D.160. R5.37 |
| D.161. **Title:** | Quality Testing extended scope |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | QTESTING_TOOLS COULD test the application for safety. |
| **Rationale:** | N/A |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 81: Requirements for the quality testing results.**

| | |
|---|---|
| **ID:** | D.162. R5.38 |
| D.163. **Title:** | Results of the Quality Testing |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | QTESTING_TOOLS MUST provide the test outcome: success or failure, and the result MUST be independent of any other test runs. |
| **Rationale:** | When running Continuous Integration, the tests need to indicate if they encountered an |

| | |
|---|---|
| | error, an invalid application state or violation of the set quality constraints. The QTESTING_TOOLS will perform many tests of varying parameters, and each result has to be independent. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 82: Requirement for the autonomy of deployment tools.**

| | |
|---|---|
| **ID:** | D.164. R5.39 |
| D.165. **Title:** | Autonomy of the deployment tools |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | DEPLOYMENT_TOOLS MUST be able to run automatically and autonomically. |
| **Rationale:** | Manual interventions into the deployment process are not trackable, and reduce control and repeatability of the deployment process. A recommended pattern is to use automatic and autonomic tools. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 83: Requirement for the scope of deployment tools.**

| | |
|---|---|
| **ID:** | D.166. R5.40 |
| D.167. **Title:** | Scope of the Deployment Tools |
| **Priority of accomplishment:** | Must have |
| **Type:** | Requirement |
| **Description:** | DEPLOYMENT_TOOLS MUST be able to deploy and install any application and the related monitoring tools from a valid topology of the supported DICE building blocks. |
| **Rationale:** | DICE will support an essential set of building blocks able to support the selected use cases. The deployment tools will make this support effective. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 84: Requirement for the extendibility and flexibility of the deployment tools.**

| | |
|---|---|
| **ID:** | D.168. R5.41 |
| **D.169. Title:** | Extendibility and flexibility of the Deployment Tools |
| **Priority of accomplishment:** | Should have |
| **Type:** | Requirement |
| **Description:** | DEPLOYMENT_TOOLS SHOULD be extendible and support multiple IaaS. |

| | |
|---|---|
| **Rationale:** | The DICE consortium aims to support a reasonable number of technologies and to demonstrate the functionality on a select platform. This, however, must not be a limiting factor for anyone in the community and industry to extend the tools and use them in other contexts and with new technologies. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

**Table 85: Support of deployment tools for Platform-as-aService (PaaS) requirement.**

| | |
|---|---|
| **ID:** | D.170. R5.42 |
| D.171. **Title:** | Support of Deployment Tools for PaaS |
| **Priority of accomplishment:** | Could have |
| **Type:** | Requirement |
| **Description:** | DEPLOYMENT_TOOLS COULD support selected PaaS. |
| **Rationale:** | The DICE deployment tools will serve as an orchestration, which is a functionality similar to the PaaS functionality, but with a wider reach and platform support. This support could be extended to some of the PaaS offerings. |
| **Supporting material:** | N/A |
| **Other comments:** | N/A |

### D.5.3 Discussion on the requirements

The majority of requirements arise from the need for the DICE tools to fit into an existing development environment. The integration should not only be seamless, but it also has to support an improvement in development efficiency and a higher quality of the results, at least compared to the traditional development approaches. The DICE survey shows that the use case teams and the surveyed private start-ups use some of the tools and patterns such as the Continuous Integration (mostly Jenkins), and they automate unit tests. The reported number of unit tests is small, however. Any load and scalability tests are done manually at most, and there is little or no reported tests of reliability of the applications.

The majority of the surveyed teams expressed an interest in a tool, which would help them automatically deploy and test for efficiency and reliability of the applications they develop. The implementation of such solutions is currently missing, because the teams feel that they do not have sufficient time or available effort to properly implement it.

DICE needs to ascertain a good internal integration of the deployment and testing tools. The CI_TOOLS act as a glue for these tools, so it needs to offer the full support for the other DICE tools. Additionally, it needs to also accommodate for any specific and custom requirements that the users of the tools have, providing them with a simple way of adding custom steps in the Continuous Integration jobs.

The DICE deployment tools should also exhibit extendibility and customisability. In this way it will be possible to adapt the DICE tools for projects, which use technologies other than the ones natively supported by the DICE. This will also make sure that the DICE deployment tools can be updated to support future versions of the technologies covered by the core building blocks.

### D.5.4 Initial roadmap

MS2:

- initial set of building blocks supported
- initial integration of the configuration management tools with WP4 services
- first prototype of the loop using recommender engine
- basic IDE controls
- basic integration and testing protocol in place
- first prototype of stress testing, configuration tuning, performance unit tests
- prototype of code injection
- toy example deployment and testing supported

MS3:

- additional building blocks support for deployment and quality testing
- initial methodology for incremental deployment
- testbed available on Flexiant's testbed for DICE developers and use cases
- fault injections demonstrated

 

# Conclusion

The main objective of the current document concerns to detail an outcome of the DICE task T1.1. “State of the art analysis and requirements specification”, namely the requirements specification. A requirements analysis was conducted from both the business and technical partners of the consortium. Use case providers detailed real-life industrial requirements that if addressed by DICE will enhance existing products/solutions. At the same time, technology providers defined a list of requirements that will a) ensure that DICE will utilize innovative algorithms/modules, b) technical obstacles will be avoided, c) limited the possibility of deviating from project’s objectives and initial vision and d) apply DICE assets to industrial needs.

The following table summarizes the objectives of the current report and the achievements.

| **Objective** | D.172. **Achievement** |
|---|---|
| | D1.2 report is submitted according to plan. |
| D.173. Address DICE MS-I | |
| Conduct a process aggregating requirements | Use case providers ran internal discussions and organized open meetings with people external to their organization as well. On the other hand, technology providers created questionnaires that were distributed to different stakeholders. They also ran internal meetings for each WP to conclude with detailed requirements lists covering all DICE areas of interest. |
| Conduct a requirement’s analysis | DICE consortium analysed the detailed lists of requirements and came up with two consolidated lists, one depicting demonstrators needs and requests and one presenting technical specifications that must be addressed to ensure a successful outcome. |
| Define a concrete list of stakeholders | Some of the weekly WP1 calls, carried out during these first six months of the project, were allocated to define and analyse the involved DICE stakeholders. The consortium’s expertise in the technologies addressed by DICE were evident during these calls and helped a lot preparing this list. Furthermore, D1.1 “State of the art analysis” provided the baseline of this activity. |
| Define a requirement’s repository | A requirement repository was defined and will be continuously updated until M24 of the project. The current requirement repository is stored on Google Drive, but can be easily exported to a Excel file maintained in the subversion system of the DICE project. So far, we keep this repository online (Google document) since it speeds up the collaborative work between multiple partners editing at the same time. |

DICE requirement’s list will be continuously updated until M24 of the project. Requirements repository will be the basis for this activity. Furthermore, in the context of T6.2, while use case providers will be defining and designing their demonstrators it is certain that requirements will be revised and an enhanced list will be created.

# References


[1] http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2014/11/mise15dice-position-paper.pdf
[2] http://ec.europa.eu/research/participants/data/ref/h2020/other/gm/h2020-guide-comm_en.pdf
[3] SPEC Research Group. DevOps Performance Work Group http://research.spec.org/working-groups/devops-performance-working-group.html
[4] DevOps http://devops.com/
[5] OMG http://www.omg.org/spec/UML/
[6] Tata Consulting Services http://www.tcs.com/Pages/default.aspx
[7] Agile Alliance http://www.agilealliance.org/
[8] OSLC http://www.oasis-oslc.org/
[9] Cloud Security Alliance https://cloudsecurityalliance.org/
[10] Network for Sustainable Ultrascale Computing http://www.nesus.eu/
[11] CloudWatch http://www.cloudwatchhub.eu/
[12] OW2: The Open Source Community for Infrastructure Software http://www.ow2.org/bin/view/Main/
[13] PIN-SME http://www.it-sme.eu/
[14] Big Data Europe Project http://www.big-data-europe.eu/
[15] Big Data Value Association http://www.bdva.eu/
[16] NESSI http://www.nessi-europe.eu/default.aspx?page=home
[17] https://www.ietf.org/rfc/rfc2119.txt
[18] http://open-services.net/bin/view/Main/OslcCoreSpecification
[19] SocialSensor http://socialsensor.atc.gr/socialsensor/site/results.xhtml
[20] http://www.governing.com/columns/tech-talk/gov-states-big-data-tax-fraud.html
[21] http://ec.europa.eu/taxation_customs/taxation/tax_fraud_evasion/a_huge_problem/index_en.htm
[22] http://www.economist.com/blogs/schumpeter/2013/01/tax-evasion-italy
[23] http://www.bluage.com/en/en_product/en-ba-db-modernization.html
[24] https://mon.service-public.fr/portail/app/cms/public/les_formulaires
[25] http://www.uscg.mil/hq/cg5/TVNCOE/Documents/links/AIS.EncodingGuide.pdf
[26] http://www.comarsystems.com/brochures/Installation%20SLR200-G%20Guide%20.pdf
[27] http://catb.org/gpsd/AIVDM.html
[28] http://www.hollandmarinehardware.nl/index.php?page=Beschrijving%20AIS
[29] http://www.aishub.net/
[30] http://www.omg.org/omgmarte/Documents/tutorial/part2.pdf
[31] Rick Kazman, Mark H. Klein, Paul C. Clements, ATAM: Method for Architecture Evaluation, CMU/SEI-2000-TR-004 Technical Report, Software Engineering Institute, available at http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5177


 