Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements

Requirement Specification – Companion Document

Deliverable 1.2

Table of Contents

List of Tables

Appendix A. DICE Demonstrators Requirements

A.1. The NewsAsset demonstrator Requirements

Table 1: Handling of data streams from social network and web-based platforms.

ID:	ATC.1
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 2: Scaling requirement.

ID:	ATC.2
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 3: Auto-scaling requirement.

ID:	ATC.3
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. Involves SIMULATION_TOOLS, MONITORING_TOOLS, QTESTING_TOOLS

Table 4: Requirement for the selection of public cloud providers.

ID:	ATC.4
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:	

Table 5: Distributed processing requirement.

ID:	ATC.5
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:	

Table 6: Requirement for the simulation and predictive analysis.

ID:	ATC.6

Title:	Simulation and predictive analysis
Priority of accomplishment:	Should have
Type:	Requirement
Description:	As a REQ_ENGINEER/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 7: Requirement for the quality metrics monitoring.

ID:	ATC.7
Title:	Quality metrics monitoring
Priority of accomplishment:	Must have
Type:	Requirement
Description:	As an 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 8: Requirement for deployment models.

ID:	ATC.8
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:	
Supporting material:	D6.1 (M16)
Other comments:	Same as PO.4. Involves TRANSFORMATION_TOOLS, DEPLOYMENT_TOOLS

Table 9: Requirement for the bottleneck detection.

ID:	ATC.9
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:	
Supporting material:	D6.1 (M16)
Other comments:	Same as PO.5 Involves VERIFICATION_TOOLS, ENHANCEMENT_TOOLS, MONITORING_TOOLS, TESTING_TOOLS

Table 10: Expression of non-functional requirements.

ID:	ATC.10
Title:	Express non-functional requirements
Priority of accomplishment:	Should have
Type:	Requirement
Description:	As an ARCHITECT 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:	
Other comments:	

Table 11: Common-model vocabularies.

ID:	ATC.11
Title:	Common model vocabularies
Priority of accomplishment:	Should have
Type:	Requirement
Description:	As an ARCHITECT I want to obtain 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:	
Other comments:	

Table 12: Methodology blueprint requirement.

ID:	ATC.12
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:	
Other comments:	

A.2. Big Data for e-Government

Table 13: Design requirement.

ID:	NETF.1
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 14: Performance impact requirement.

ID:	NETF.2
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 15: Storage requirement.

ID:	NETF.3
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 16: Requirement for deployment models.

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

Table 17: DIA analysis and assessment.

ID:	NETF.5
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 18: Requirement for the monitoring of quality and performance metrics.

ID:	NETF.6
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: • monitoring data and logs to detect candidate anomalies and report to user • detecting data design anti-patterns • estimate root-causes of quality anomalies

Table 19: Requirement for the scalability analysis.

ID:	NETF.7
Title:	Scalability analysis
Priority of accomplishment:	Should have
Description:	Cloud deployment and scalability • Evaluate cloud alternatives for deployment: Cost versus performance. • Automatically create cloud deployment configurations

A.3. DICE-based Geo-fencing for the Maritime Sector

Table 20: Simulation and predictive analysis of new business rules.

ID:	PO.1
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 on the quality and performance metrics of a new CEP business rule so that I can evaluate different alternative requirements with a lower impact on quality and performance
Rationale:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves SIMULATION_TOOLS

Table 21: Scalability analysis requirement.

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

Table 22: Requirement for the quality metrics monitoring.

ID:	PO.3
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves MONITORING_TOOLS, QTESTING_TOOLS

Table 23: Requirement for deployment models.

ID:	PO.4
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves TRANSFORMATION_TOOLS, DEPLOYMENT_TOOLS

Table 24: Requirement for the bottleneck detection.

ID:	PO.5
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:	
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 25: Requirement for the testing and load stressing scenarios.

ID:	PO.6
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves DEPLOYMENT_TOOLS, CI_TOOLS, QTESTING_TOOLS, TESTBED

Table 26: Performance impact requirement.

ID:	PO.7

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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves OPTIMIZATION_TOOLS, ENHANCEMENT_TOOLS

Table 27: Continuous integration requirement.

ID:	PO.8
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves DEPLOYMENT_TOOLS, CI_TOOLS, TRANSFORMATION_TOOLS

Table 28: Requirement for the text fixtures generation.

ID:	PO.9
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 29: Requirement for the running of simulation environments.

ID:	PO.10

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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves SIMULATION_TOOLS, DEPLOYMENT_TOOLS, CI_TOOLS

Table 30: Requirement for the execution metrics.

ID:	PO.11
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves MONITORING_TOOLS, QTESTING_TOOLS

Table 31: Requirement for the reliability results comparison.

ID:	PO.12
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves MONITORING_TOOLS, ANOMALY_TRACE_TOOLS, QTESTING_TOOLS

Table 32: Deployment requirements.

ID:	PO.13
Title:	Deployment requirements
Priority of	Must have

accomplishment:	
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves DEPLOYMENT_TOOLS

Table 33: Deployment monitoring requirement.

ID:	PO.14
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves MONITORING_TOOLS, ANOMALY_TRACE_TOOLS

Table 34: Requirement for the deployment scripts.

ID:	PO.15
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:	
Supporting material:	See: DICE-based Geo-fencing for the Maritime Sector (POSIDONIA OPERATIONS)
Other comments:	Involves DEPLOYMENT_TOOLS

Appendix B. Technical Requirements

B.1. WP1 Requirements

B.1.1 Consolidated requirements

Table 35: Requirement for the stereotyping of UML diagrams and DICE profile.

ID:	R1.1
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 36: Requirement for the DICE methodology guidance.

ID:	R1.2
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 37: Continuous integration tools requirement.

ID:	R1.7
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	N/A

material:	
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 38: Requirement for the loading of annotated UML model.

ID:	R3IDE.4
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 39: Property verification requirement.

ID:	R3IDE.4.2
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 40: Graphical output requirement.

ID:	R3IDE.5
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 41: Requirement for the visualisation of analysis results.

ID:	R4IDE5
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 42: Requirement for the loading of safety and privacy properties.

ID:	R4IDE6
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.

B.1.2 Detailed requirements

Table 43: The Stereotyping of UML diagrams with DICE profile Requirement.

ID:	R1.1
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 44: The Guides through the DICE methodology Requirement.

ID:	R1.2
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 45: The Quality testing tools IDE integration Requirement.

ID:	R1.6
Title:	Quality testing tools IDE integration
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The IDE SHOULD provide the means to configure the QTESTING_TOOLS execution
Rationale:	Quality tests may come with parameters such as the number of tests to run or the

	duration of each tests, which the user should be able to change.
Supporting material:	N/A
Other comments:	N/A

Table 46: The Continuous integration tools IDE integration Requirement.

ID:	R1.7
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 47: The Running tests from IDE without committing to VCS Requirement.

ID:	R1.7.1
Title:	Running tests from IDE without committing to VCS
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The CI_TOOLS COULD provide an integration with the IDE that enables deployment and execution of tests on the user's local changes without committing the code into the VCS.
Rationale:	In some cases the DEVELOPER may want to run a test without committing the code into the repository.
Supporting material:	N/A
Other comments:	N/A

Table 48: The IDE support to the use of profile Requirement.

ID:	R2IDE.1
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 widzards 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 49: The Metric selection Requirement.

ID:	R3IDE.1
Title:	Metric selection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE IDE MUST allow to select the metric to compute from those defined in the DPIM/DTSM DICE annotated UML model. There are efficiency and reliability related metrics
Rationale:	N/A
Supporting material:	The metrics supported will be all those defined in WP2. Examples of them are Throughput or response time when talking about performance; or MTTF o MTBF, and so on regarding reliability
Other comments:	N/A

Table 50: The Timeout specification Requirement.

ID:	R3IDE.2
Title:	Timeout specification
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The IDE SHOULD allow the user to set a timeout and a maximum amount of memory (2) to be used when running the SIMULATION_TOOLS and the VERIFICATION_TOOLS. Then, when the timeout expires or when the memory limit is exceeded, the IDE SHOULS notify the user
Rationale:	N/A
Supporting material:	(2) The timeout should be set by the user considering the hardware configuration and the space of the model
Other comments:	N/A

Table 51: The Usability Requirement.

ID:	R3IDE.3
Title:	Usability
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The TRANSFORMATION_TOOLS and SIMULATION_TOOLS MAY follow some usability, ergonomics or accesibility standard such as ISO/TR 16982:2002,

	ISO 9241, WAI W3C or similar
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 52: The Loading the annotated UML model Requirement.

ID:	R3IDE.4
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 53: The Usability of the IDE-VERIFICATION_TOOLS interaction Requirement.

ID:	R3IDE.4.1
Title:	Usability of the IDE-VERIFICATION_TOOLS interaction
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The QA_ENGINEER SHOULD not perceive a difference between the IDE and the VERIFICATION_TOOL; it SHOULD be possible to seamlessly invoke the latter from the former
Rationale:	In a sense the IDE and the VERFICATION_TOOLS reside in a sort of continuum, where the former invokes the latter, but the user should not feel the difference in the environment
Supporting material:	N/A
Other comments:	N/A

Table 54: The Loading of the property to be verified Requirement.

ID:	R3IDE.4.2
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 55: The Graphical output Requirement.

ID:	R3IDE.5
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 mod
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 56: The Graphical output of erroneous behaviors Requirement.

ID:	R3IDE.5.1
Title:	Graphical output of erroneous behaviors
Priority of accomplishment:	Could have
Type:	Requirement
Description:	In case the outcome of the verification task is "the property does not hold", the VERIFICATION_TOOLS COULD provide, in addition to the raw execution trace of the system that violates the desired property, an indication of where in the trace lies the probl
Rationale:	In case of a property not holding, the VERIFICATION_TOOLS return a trace of the system model that violates the property. Understanding *why* the property is violated (e.g., which part of the trace is the one where the property is violated) is not always an easy task. The output of the VERIFICATION_TOOLS might help in this regard, by highlighting where the problem lies.
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 57: The Loading a DDSM level model Requirement.

ID:	R3IDE.6
Title:	Loading a DDSM level model
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The OPTIMIZATION_TOOLS as part of the IDE MUST provide an interface to load (not design) a DDSM DICE annotated model
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 58: The Resource consumption breakdown Requirement.

ID:	R4IDE1
Title:	Resource consumption breakdown
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEVELOPER MUST be able to see via the ENHANCEMENT_TOOLS the resource consumption breakdown into its atomic components.
Rationale:	Existence of different abstraction levels between design concepts (e.g., abstractions in the DICE profile) and runtime measurements hides the details on what high-level request effectively generated the request for data.
Supporting material:	R4.11
Other comments:	N/A

Table 59: The Bottleneck Identification Requirement.

ID:	R4IDE2
Title:	Bottleneck Identification
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS MUST indicate which classes of requests represent bottlenecks for the application in a given deployment.
Rationale:	N/A
Supporting material:	R4.12
Other comments:	N/A

Table 60: The Model parameter uncertainties Requirement.

ID:	R4IDE3
Title:	Model parameter uncertainties
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The REQ_ENGINEER COULD express uncertainty on some performance/reliability input parameters (e.g., execution times) in the DICE profile by means of a prior distribution or an interval. The ENHANCEMENT_TOOLS COULD take into account these parameters to esti
Rationale:	DoW mentions Bayesian estimation techniques. These techniques can explicitly account for the uncertainty provided by the REQ_ENGINEER.
Supporting material:	R4.20
Other comments:	This requirement may be alternatively stated as part of WP2 or WP3, since it also affects the DICE profile. The requirement would expand the scientific impact of the tool, but if too complex to implement it might be ignored without major consequences.

Table 61: The Model parameter confidence intervals Requirement.

ID:	R4IDE4
Title:	Model parameter confidence intervals
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS COULD return confidence intervals for each inferred parameter of the performance and reliability models.
Rationale:	The WP3 models require to provide a number of parameters, such as CPU speeds. These will be inferred by the ENHANCEMENT_TOOLS of WP4 from the monitoring data. However, the estimation is subject to uncertainties so confidence intervals could be provided to the WP3 tools to quantify such uncertainty. If the CI is too wide, we might issue a warning in SIMULATION_TOOLS that the prediction is not robust.
Supporting material:	R4.21
Other comments:	N/A

Table 62: The Visualization of analysis results Requirement.

ID:	R4IDE5
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 63: The Safety and privacy properties loading Requirement.

ID:	R4IDE6
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

Table 64: The Feedback from safety and privacy properties monitoring to UML models concerning violatedtime bounds Requirement.

ID:	R4IDE7
Title:	Feedback from safety and privacy properties monitoring to UML models concerning violated time bounds
Priority of accomplishment:	Could have
Type:	Requirement
Description:	In the feedback provided by the ANOMALY_TRACE_TOOLS to the DEVELOPER/ARCHITECT, the tools COULD highlight when a timing requirement is violated, and what is the value of the violation
Rationale:	The specific feedback about timing violations might help the DEVELOPER/ARCHITECT adjust the parameters of the models/properties
Supporting material:	R4.31.1
Other comments:	N/A

Table 65: The Relation between ANOMALY_TRACE_TOOLS and IDE Requirement.

ID:	R4IDE8
Title:	Relation between ANOMALY_TRACE_TOOLS and IDE
Priority of accomplishment:	Should have
Type:	Requirement

Description:	It SHOULD be possible to launch the ANOMALY_TRACE_TOOLS from the IDE
Rationale:	The idea is that the trace checking is performed starting from the elements that are described in the DICE UML model (see requirement R4.32). Hence, it makes sense that the tool is invoked from the UML IDE. The idea could be that the IDE has a link to the DW, and when the user asks for performing trace checking, the IDE queries the DW, retrieves the information for the trace checking, then feeds the ANOMALY_TRACE_TOOLS with the traces to be checked.
Supporting material:	R4.33
Other comments:	N/A

B.2. WP2 Requirements

B.2.1 Consolidated requirements

Table 66: Requirement for the DICE methodological paradigm.

ID:	R2.1
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 67: Requirement for the origin of the abstraction layer.

ID:	R2.2
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:	
Other comments:	N/A

Table 68: DICE Constraints Specification Requirement

ID:	R2.4

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:	
Other comments:	N/A

Table 69: Requirement for the DICE profile technology-specific constraints.

ID:	R2.10
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 70: Requirement for the DICE profile separation-of-concerns.

ID:	R2.11
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 71: Requirement for the data-intensive Quality of Service (QoS).

ID:	R2.4
Title:	Data-Intensive QoS
Priority of	Must have

accomplishment:	
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 72: Requirement for the DICE topologies.

ID:	R2.9
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 73: Requirement for DICE extension points.

ID:	R2.7
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 74: Requirement for the DICE deployment-specific views.

ID:	R2.12

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 75: IDE support to the use of profile requirement.

ID:	R2.17
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 76: DICE Analysis Focus.

ID:	R2.18
Title:	DICE Analysis Focus
Priority of accomplishment:	Must have
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

B.2.2 Detailed requirements

Table 77: Requirement for the DICE methodological paradigm.

ID:	R2.1
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 78: The Abstraction Layer Origin Requirement.

ID:	R2.2
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:	
Other comments:	N/A

Table 79: The Relation with MARTE UML Profile Requirement.

ID:	R2.3
Title:	Relation with MARTE UML Profile
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile MUST define required and provided properties of a DIA as well as metrics (estimated, measured, calculated and requirements) to monitor them. Said metrics will be specifed following the MARTE NFP framework.
Rationale:	MARTE provides valuable foundations for specifying non-functional properties and shall be considered for extension
Supporting material:	http://www.omgmarte.org/
Other comments:	N/A

Table 80: The Constraints Definition Requirement.

ID:	R2.4
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:	
Other comments:	N/A

Table 81: The DICE Profile Performance Annotations Requirement.

ID:	R2.5
Title:	DICE Profile Performance Annotations
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define annotations for performance based on the MARTE::GQAM framework.
Rationale:	Relevant part inherited from MARTE for the specifcations of performance values.
Supporting material:	N/A
Other comments:	N/A

Table 82: The DICE Profile Reliability Annotations Requirement.

ID:	R2.6
Title:	DICE Profile Reliability Annotations
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define annotations for reliability based on the DAM profile.
Rationale:	DAM is a profile designed to extend MARTE in support of reliability, and therefore shall be considered within DICE and the profile specification.
Supporting material:	N/A
Other comments:	N/A

Table 83: Requirement for DICE extension points.

ID:	R2.7

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 84: The DICE Profile Main DIA Concerns - Flow and Behavior Requirement.

ID:	R2.8
Title:	DICE Profile Main DIA Concerns - Flow and Behavior
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define annotations that address behavioral and flow concerns behind DIAs. Also, the DICE Profile shall define annotations for flow-control across DIAs.
Rationale:	Many of the characteristics behind DIAs are sensibly influenced by the flow of information, its management and the application's behavior in managing and handling data. These aspects shall be made explicit for DICE-supported analysis.
Supporting material:	N/A
Other comments:	N/A

Table 85: Requirement for the DICE topologies.

ID:	R2.9
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 86: The DICE Profile Tech-Specific Constraints Requirement.

ID:	R2.10
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 87: The DICE Profile Separation-of-Concerns Requirement.

ID:	R2.11
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 88: Requirement for the DICE deployment-specific views.

ID:	R2.12
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 89: The DICE Profile Data Structure Requirement.

ID:	R2.13
Title:	DICE Profile Data Structure
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define QoS annotations for data structure and its specification.
Rationale:	Data-Structure is a big concern in Data-Intensive Applications. Also, said concern must be explicitly supported with ad-hoc constructs such that its relations with DIAs is properly analysed and supported at Design time.
Supporting material:	N/A
Other comments:	N/A

Table 90: The DICE Profile Data Communication Requirement.

ID:	R2.14
Title:	DICE Profile Data Communication
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define annotations to elaborate on structural and behavioral details concerning the channeling and marshalling of information across specified DIAs.
Rationale:	the flow of information across a DIA, e.g., for further processing or visualization shall be supported at both structural (i.e., nodes involved) and behavioral (i.e., behavior of said nodes) level. Thsi is because data flow and manipulation of data can vary sensibly depending on the kind of DIA being designed (e.g., for the purpose of analysing streaming data).
Supporting material:	N/A
Other comments:	N/A

Table 91: The DICE Profile Sub-Structures Requirement.

ID:	R2.15
Title:	DICE Profile Sub-Structures
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall provide annotations for specifying node nesting and replication across the structure of DIAs.
Rationale:	DIAs often are requried to be designed as nested applications. For example, compute nodes may hide internal logic from multiple possible technological specification within them. Therefore, the ability to support nesting and sub-structure across DIAs shall be supported.
Supporting material:	N/A
Other comments:	N/A

Table 92: The DICE Analysis Focus Requirement.

ID:	R2.16
Title:	DICE Analysis Focus
Priority of accomplishment:	Must have
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 methdological approaches to support those properties that are relevant to perform analysis (e.g., for fine-tuning, load-
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 actioned behavior for continuous improvement. Other endeavours, however connected to DIAs, are out of the scope of DICE.
Supporting material:	https://docs.google.com/presentation/d/1aAeoGJox42pHBpmLCDDhwGtmb-J7RmzFobqm-QB7tV8/edit#slide=id.gb6c695009_2_115
Other comments:	N/A

Table 93: The DICE Methodological Paradigm Requirement.

ID:	MR2.1
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 94: The DICE Methodology support Diagrams Requirement.

ID:	MR2.2
Title:	DICE Methodology support Diagrams
Priority of accomplishment:	Should have
Type:	Domain Assumption
Description:	Every abstraction layer (namely, DPIM, DTSM and DDSM) of the DICE profile shall stem from UML.
Rationale:	several notations are being considered in the scope of DICE (e.g., MDA, MDE, MARTE, SecureML) - these notations already provide diagramming facilities that may be assumed as directly related to the needs and requirements of the DICE profile.

Supporting material:	N/A
Other comments:	N/A

Table 95: The DICE Design Process Requirement.

ID:	PR2.16
Title:	DICE Design Process
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE profile and methodology shall support the design of DIAs across three layers of abstractions: The DPIM, the DTSM and the DDSM, addressing platform-independent, technology-specific and deployment-specific details respectively.
Rationale:	Designing DIAs via the DICE profile shall also follow the MDE paradigm.
Supporting material:	N/A
Other comments:	N/A

Table 96: The DICE Profile Views Requirement.

ID:	MR2.3
Title:	DICE Profile Views
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE profile framework MUST envision that the designer obtains views using the DICE profile and following the methodology. Said views shall isolate separately all and only elements necessary to perform DICE quality evaluations. To this purpose, the DP
Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 97: The DICE Component View: this view allows designers to elaborate on the organizational structure ofthe components and possibly the responsible entities involved in the DIAinteractions for the purpose of realising the DIA's intended use; (4) A QoS Cross-Cutti Requirement.

ID:	MR2.3a
Title:	DICE Component View: this view allows designers to elaborate on the organizational structure of the components and possibly the responsible entities involved in the DIAinteractions for the purpose of realising the DIA's intended use; (4) A QoS Cross-Cutti
Priority of accomplishment:	Must have
Type:	Requirement
Description:	this view allows designers to elaborate on the organizational structure of the components and possibly the responsible entities involved in the DIA

Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 98: The DICE State-Behavioral View Requirement.

ID:	MR2.3b
Title:	DICE State-Behavioral View
Priority of accomplishment:	Must have
Type:	Requirement
Description:	this view allows designers to elaborate on the internal components behavior rather than high-level components interactions across the DIA
Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 99: The DICE Sequence-Behavioral View Requirement.

ID:	MR2.3c
Title:	DICE Sequence-Behavioral View
Priority of accomplishment:	Must have
Type:	Requirement
Description:	this view allows designers to elaborate on components interactions for the purpose of realising the DIA’s intended use
Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 100: The DICE QoS Cross-Cutting View Requirement.

ID:	MR2.3d
Title:	DICE QoS Cross-Cutting View
Priority of accomplishment:	Must have
Type:	Requirement
Description:	this view shall consist of cross-cutting annotations to elements in views “a”, “b” and “c”. The purpose of this view is to elaborate on the QoS constraints, limitations or requirements specified for annotated elements. The DICE profile shall focus on QoS
Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to

	be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 101: The A Usage Cross-Cutting View; Requirement.

ID:	MR2.3e
Title:	A Usage Cross-Cutting View;
Priority of accomplishment:	Must have
Type:	Requirement
Description:	this view shall consist of cross-cutting annotations or graphical notations containing information related to the expected entrance load for the DIA and its composing elements.
Rationale:	the views in the requirement emerged from a preliminary analysis of concerns to be addressed at design time for DIAs.
Supporting material:	N/A
Other comments:	N/A

Table 102: The Data-Intensive QoS Requirement.

ID:	MR2.4
Title:	Data-Intensive QoS
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DPIM shall 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-inten
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 103: The DICE DPIM Relations Requirement.

ID:	MR2.5
Title:	DICE DPIM Relations
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DPIM shall inherit notations and concepts from conceptual notations intended for similar purposes. For example, ModaCloudML offers modeling facilities to reason on cloud-based applications from multiple, functionally-complete perspectives (e.g., data,

Rationale:	there exist a number of profiles that alaready (partially) cover the needs behind the DICE profile. Rather than reinventing new concepts, DICE may well inherit from said notations reusing where possible.
Supporting material:	N/A
Other comments:	N/A

Table 104: The DICE DPIM Concern - Data and I/O Logic Requirement.

ID:	MR2.6
Title:	DICE DPIM Concern - Data and I/O Logic
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DPIM shall provide annotations to specify data-retrieval (i.e., where does the data come from and how is it transferred to its destination). Hence, I/O logic shall also be specified at the DPIM layer. Therefore, the DICE profile has to provide annotat
Rationale:	the DPIM layer shall be conceived for requirements engineering of DIAs. In so doing, data and I/O shall be equally covered in the first layer of DIA abstraction.
Supporting material:	N/A
Other comments:	N/A

Table 105: The DICE Extension-Points Requirement.

ID:	MR2.7
Title:	DICE Extension-Points
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DTSM shall 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
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 106: The DICE Splits Requirement.

ID:	MR2.8
Title:	DICE Splits
Priority of accomplishment:	Must have
Type:	Requirement

Description:	The DTSM layer shall support the definition and reasoning of "Splits", i.e., computable portions of data for the DIA at hand.
Rationale:	The DICE profile shall support the design of logically processable portions of information, i.e., "splits". This construct is technology-specific and is therefore needed starting from the DTSM layer. For example, if the designer is interested in knowing or manipulating/configuring the data processing policy he may want to vary the size, shape and processing for splits in his ad-hoc DIA.
Supporting material:	N/A
Other comments:	N/A

Table 107: The DICE Topologies Requirement.

ID:	MR2.9
Title:	DICE Topologies
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DTSM layer shall 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 108: The DICE Access Policies Requirement.

ID:	MR2.10
Title:	DICE Access Policies
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DTSM layer shall support the definition of Access Policies, e.g., to data or to DIA frameworks.
Rationale:	normally a designer is also required to specify which access policies will be used across the DIAs. Given that different tchnologies require different access policies and related mechanisms, reasoning on Access policies shall take place initailly at the DTSM layer.
Supporting material:	N/A
Other comments:	N/A

Table 109: The DICE Functional Definition Requirement.

ID:	MR2.11

Title:	DICE Functional Definition
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DTSM layer shall support Technology-specific functions definition (Map-Reduce-Combine vs. Transformation-Action-Filter etc.).
Rationale:	The technological compound within DIAs consists of functional definitions which are specific for certain technologies. This means that functional specification for said technologies shall take place initially at the DTSM layer.
Supporting material:	N/A
Other comments:	N/A

Table 110: The DICE Deployment Specific Views Requirement.

ID:	MR2.12
Title:	DICE Deployment Specific Views
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DDSM layer shall support the definition of an Actionable deployment view (TOSCA-ready): this view offers conceptual mappings between the technological layer defined in the DTSM and concepts in the TOSCA metamodeling infrastructure such that one-way t
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 111: The DICE Framework Overrides Requirement.

ID:	MR2.13
Title:	DICE Framework Overrides
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DDSM layer shall support the definition of framework overrides. This allows designers to provide ad-hoc tweaks to framework settings based on specific constraints or design concerns.
Rationale:	many applications require ad-hoc configuration of the frameworks on which they are based. These tweaks are, by design, only allowed to change execution and deployment dynamics. Therefore, this ability shall be given to designers at the DDSM layer.
Supporting material:	N/A
Other comments:	N/A

Table 112: The DICE Resource Control Requirement.

ID:	MR2.14
Title:	DICE Resource Control
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DDSM layer shall support the management of VMs and similar resources as well as the necessary environmental setup connected to the application of specific frameworks (e.g., Hadoop/MapReduce).
Rationale:	many DIAs require fine-grained handling and management of resources beyond transparent resource-provisioning. Designers shall be given the ability to govern said aspects of deployment at the DDSM layer.
Supporting material:	N/A
Other comments:	N/A

Table 113: The DICE Scripting Support Requirement.

ID:	MR2.15
Title:	DICE Scripting Support
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DDSM layer shall allow the support for linking ad-hoc config. scripts or default config. scripts within the DIA.
Rationale:	a big part in specifying and deploying/running DIAs consists in the definition/reuse of configuration scripts. The DICE profile shall allow designers to link scripts to modelling elements specific to their designed DIA.
Supporting material:	N/A
Other comments:	N/A

Table 114: The DIA Application Bundling Requirement.

ID:	MR2.16
Title:	DIA Application Bundling
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The Actionable Deployment View within the DDSM layer shall support DIA application bundling, e.g., using the CSAR formalism adopted by the TOSCA notation.
Rationale:	Container technologies are the de-facto standard for deploying DIAs. The TOSCA reference format for DICE deployment models already pre-defines a deployment bundle possibly for reuse within the DICE profile itself.
Supporting material:	N/A
Other comments:	N/A

Table 115: The IDE support to the use of profile Requirement.

ID:	MR2.17
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 widzards 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 116: The DICE Deployment Constructs Origin Requirement.

ID:	PRD2.1
Title:	DICE Deployment Constructs Origin
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The DICE Profile shall define deployment-specific construct contiguously to TOSCA-specific constructs and their relations.
Rationale:	TOSCA is the key reference format to be supported for deployment-ready DIAs - reference to its constructs shall be constant in the definition of the DICE profile.
Supporting material:	N/A
Other comments:	N/A

Table 117: The DICE Deployment Required and Provided Properties Requirement.

ID:	PRD2.2
Title:	DICE Deployment Required and Provided Properties
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define technology-specific properties in terms of required- and provided-properties.
Rationale:	Provided- and required-properties are an essential concept behind TOSCA-ready cloud applications. TOSCA-ready orchestrators use said constructs as requirements to drive the deployment process of parsed specifications. As a consequence, said constructs shall be used massively across the definition of DICE profile and its modeling elements.
Supporting material:	N/A

Other comments:	N/A

Table 118: The DICE Deployment Required and Provided Execution Platforms Requirement.

ID:	PRD2.3
Title:	DICE Deployment Required and Provided Execution Platforms
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE Profile shall define annotations support the specification of required- and provided-execution platforms for the deployment of DIAs.
Rationale:	execution platforms are coherent specifications that describe the environment atop which the DIA needs to be processed. DIAs specified within DICE shall include said specifications since they are required to map DICE-specified DIAs into TOSCA-ready executable CSAR bundles.
Supporting material:	N/A
Other comments:	N/A

Table 119: The DICE Deployment - NFV Requirement.

ID:	PRD2.4
Title:	DICE Deployment - NFV
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The DICE Profile shall provide facilities to model virtualized network-functions and their respective relations in an NFV topology.
Rationale:	Network-Function Virtualization shall be an integral part to DICE profile definition. Also, in defining TOSCA-compliant specifications, DIAs specified within DICE shall need to elaborate on NFV constructs to be possibly expressed using TOSCA-YAML syntax.
Supporting material:	N/A
Other comments:	N/A

B.3. WP3 Requirements

B.3.1 Consolidated requirements

Table 120: Requirement for the Model to Model (M2M) transformation.

ID:	R3.1
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 121: Requirement for the annotations.

ID:	R3.2
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 122: Requirement for the generation of traces from the system model.

ID:	R3.7
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 123: Requirement for the cost/quality balance.

ID:	R3.8
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 124: Requirement for the Service Level Agreement (SLA) specification and compliance.

ID:	R3.10
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 125: Requirement for the optimisation timeout.

ID:	R3.11
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 126: Requirement for the white/black box transparency.

ID:	R3.13
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

B.3.2 Detailed requirements

Table 127: The M2M Transformation Requirement.

ID:	R3.1
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 128: The Taking into account relevant annotations Requirement.

ID:	R3.2
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 form
Rationale:	N/A
Supporting material:	A property is a characteristic of a system's element (e.g. tranfer rate of a disk)
Other comments:	N/A

Table 129: The Transformation rules Requirement.

ID:	R3.3
Title:	Transformation rules
Priority of accomplishment:	Could have

Type:	Requirement
Description:	The TRANSFORMATION_TOOLS MAY be able to extract, interpret and apply the transformation rules from an external source(1).
Rationale:	An external source joined to a declarative style make it possible to extend the behavior of the system without having to modify source code. In the last term, these two requirements, will permit to provide an extension mechanism to the DICE profile (e.g. to support the impact of new parameters coming from new technologies or algorithms).
Supporting material:	1) External source: Probably a repository with the transformation rules in declarative format to be processed by QVT (Query/View/Transformation) or a similar tool
Other comments:	N/A

Table 130: The Simulation solvers Requirement.

ID:	R3.4
Title:	Simulation solvers
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The SIMULATION_TOOLS will select automatically and acording to the metric selected, the right SOLVER whether simulation or analytical solvers (e.g. Markov sollution)
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 131: The Simulation of hosted big data services Requirement.

ID:	R3.5
Title:	Simulation of hosted big data services
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The SIMULATION_TOOLS MUST be able to describe the execution characteristics of hosted big data services.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 132: The Transparency of underlying tools Requirement.

ID:	R3.6
Title:	Transparency of underlying tools
Priority of accomplishment:	Must have

Type:	Requirement
Description:	The TRANSFORMATION_TOOLS and SIMULATION_TOOLS MUST be transparent to users. From their point of view the user is analyzing metrics from and making simulations over an enriched UML Model.
Rationale:	N/A
Supporting material:	The whole process must be atomic to the user. s/he just need to know that is simulating the behaviour of an UML model. Any tranformation or analysis we are doing to compute the metrics doesn't need to be explicited to the user (or even better expressed,
Other comments:	N/A

Table 133: The Generation of traces from the system model Requirement.

ID:	R3.7
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
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 134: The Cost/quality balance Requirement.

ID:	R3.8
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 135: The Relaxing constraints Requirement.

ID:	R3.9

Title:	Relaxing constraints
Priority of accomplishment:	Could have
Type:	Requirement
Description:	Being not possible to fulfill all requirements (SLA vs cost), the OPTIMIZATION_TOOLS COULD suggest what constraints should be relaxed (whether cost related or SLA related) to obtain a compliant model
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 136: The SLA specification and compliance Requirement.

ID:	R3.10
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 137: The Optimization timeout Requirement.

ID:	R3.11
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 138: The Modelling abstraction level Requirement.

ID:	R3.12
Title:	Modelling abstraction level
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The user should not be waiting for a response indefinitely
Rationale:	We are not talking here about layers or tiers in the sense we do when talking about DPIM, DTSM or DDSM models, but about levels in the degree of understanding/knowledge (or desired granularity) of the system. Some elements will be treated as black boxes.
Supporting material:	N/A
Other comments:	N/A

Table 139: The White/black box transparency Requirement.

ID:	R3.13
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 doesn't 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

Table 140: The Ranged or extended what if analysis Requirement.

ID:	R3.14
Title:	Ranged or extended what if analysis
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The SIMULATION_TOOLS will be able to cover a range of possible values for a parameter and run a simulation for every different scenario (according to a gap parameter that splits the range to cover in a list of discrete values to evaluate)
Rationale:	N/A
Supporting material:	N/A

Other comments:	N/A

Table 141: The Verification of temporal safety/privacy properties Requirement.

ID:	R3.15
Title:	Verification of temporal safety/privacy properties
Priority of accomplishment:	Must have
Type:	Requirement
Description:	Taking the DICE annotated UML model (which must include the property to be verified) as an input, the VERIFICATION_TOOLS MUST be able to answer questions related to whether the specified property holds for the modeled system or not.
Rationale:	This is the main role of the VERIFICATION_TOOL: to be able to verify the properties defined in the DICE UML model
Supporting material:	N/A
Other comments:	N/A

Table 142: The Metric selection Requirement.

ID:	R3IDE.1
Title:	Metric selection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DICE IDE MUST allow to select the metric to compute from those defined in the DPIM/DTSM DICE annotated UML model. There are efficiency and reliability related metrics
Rationale:	N/A
Supporting material:	The metrics supported will be all those defined in WP2. Examples of them are Throughput or response time when talking about performance; or MTTF o MTBF, and so on regarding reliability
Other comments:	N/A

Table 143: The Timeout specification Requirement.

ID:	R3IDE.2
Title:	Timeout specification
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The IDE SHOULD allow the user to set a timeout and a maximum amount of memory (2) to be used when running the SIMULATION_TOOLS and the VERIFICATION_TOOLS. Then, when the timeout expires or when the memory limit is exceeded, the IDE SHOULD notify the user
Rationale:	N/A
Supporting material:	(2) The timeout should be set by the user considering the hardware configuration

	and the space of the model
Other comments:	N/A

Table 144: The Usability Requirement.

ID:	R3IDE.3
Title:	Usability
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The TRANSFORMATION_TOOLS and SIMULATION_TOOLS MAY follow some usability, ergonomics or accesibility standard such as ISO/TR 16982:2002, ISO 9241, WAI W3C or similar
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 145: The Loading the annotated UML model Requirement.

ID:	R3IDE.4
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:	N/A

Table 146: The Usability of the IDE-VERIFICATION_TOOLS interaction Requirement.

ID:	R3IDE.4.1
Title:	Usability of the IDE-VERIFICATION_TOOLS interaction
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The QA_ENGINEER SHOULD not perceive a difference between the IDE and the VERIFICATION_TOOL; it SHOULD be possible to seamlessly invoke the latter from the former
Rationale:	In a sense the IDE and the VERFICATION_TOOLS reside in a sort of continuum, where the former invokes the latter, but the user should not feel the difference in the environment
Supporting material:	N/A

Other comments:	N/A

Table 147: The Loading of the property to be verified Requirement.

ID:	R3IDE.4.2
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:	The properties that can be defined at the level of the DICE UML model should actually only be those that can be analyzed.

Table 148: The Graphical output Requirement.

ID:	R3IDE.5
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 mod
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:	N/A

Table 149: The Graphical output of erroneous behaviors Requirement.

ID:	R3IDE.5.1
Title:	Graphical output of erroneous behaviors
Priority of accomplishment:	Could have
Type:	Requirement
Description:	In case the outcome of the verification task is "the property does not hold", the VERIFICATION_TOOLS COULD provide, in addition to the raw execution trace of the system that violates the desired property, an indication of where in the trace lies the probl
Rationale:	In case of a property not holding, the VERIFICATION_TOOLS return a trace of

	the system model that violates the property. Understanding *why* the property is violated (e.g., which part of the trace is the one where the property is violated) is not always an easy task. The output of the VERIFICATION_TOOLS might help in this regard, by highlighting where the problem lies.
Supporting material:	N/A
Other comments:	N/A

Table 150: The Loading a DDSM level model Requirement.

ID:	R3IDE.6
Title:	Loading a DDSM level model
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The OPTIMIZATION_TOOLS as part of the IDE MUST provide an interface to load (not design) a DDSM DICE annotated model
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

B.4. WP4 Requirements

B.4.1 Consolidated requirements

Table 151: Requirement for the monitoring data extraction.

ID:	R4.3
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 152: Requirement for access restriction to the monitoring data.

ID:	R4.6
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 153: Monitoring visualisation requirement.

ID:	R4.8
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 154: Requirement for the refactoring methods.

ID:	R4.14
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 155: Enhancement tools version difference requirement.

ID:	R4.16
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 156: Requirement for the parameterization of simulation and optimization models.

ID:	R4.19
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 157: Requirement for the time-based ordering of monitoring data entries.

ID:	R4.22
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 158: Requirement for the data size trends.

ID:	R4.23
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 159: Requirement for the propagation of changes/automatic annotation of UML models.

ID:	R4.27
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 160: Requirement for the loading of safety and privacy properties.

ID:	R4.28 (R4IDE6)
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 161: Requirement for the monitoring of safety and privacy properties.

ID:	R4.30
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 162: Requirement for the correlation between data stored in the DW and DICE UML models.

ID:	R4.32
Title:	Correlation between data stored in the DW and DICE UML models
Priority of	Must have

accomplishment:	
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?

B.4.2 Detailed requirements

Table 163: The Monitoring data warehousing Requirement.

ID:	R4.1
Title:	Monitoring data warehousing
Priority of accomplishment:	Must have
Type:	Requirement
Description:	There will be multiple 'monitoring data collector' tools that will retrieve monitoring data from different platforms and store it under the monitoring data warehouse. The data warehouse will support different data types, providing near real-time access.
Rationale:	We expect that the monitoring agents will produce a high number of monitoring data. This data needs to be stored in the application's test and runtime environment, capable of handling the bulk of data.
Supporting material:	In the early stage, the monitoring data refers to logs produced by the Big Data applications (Hadoop, NOSQL).
Other comments:	In the early stage, the monitoring data refers to logs produced by the Big Data applications (Hadoop, NOSQL)

Table 164: The Monitoring data warehouse schema Requirement.

ID:	R4.2
Title:	Monitoring data warehouse schema
Priority of accomplishment:	Must have
Type:	Requirement
Description:	MONITORING_TOOLS storing the monitoring data MUST use a schema that lets identify the sources of the monitoring data, but is general enough to permit adding new sources.
Rationale:	The monitoring data warehousing needs to accommodate for any monitoring data input format and content without losing any relevant data. The monitoring entries need to be equipped with metadata, but the contents need to stay intact.
Supporting material:	N/A
Other comments:	N/A

Table 165: The Monitoring data versioning Requirement.

ID:	R4.2.1
Title:	Monitoring data versioning
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The metrics records MUST include the information on the version of the APPLICATION's build.
Rationale:	Association between the monitored application's version and the monitoring data is crucial for quality enhancement and configuration recommendation engine.
Supporting material:	N/A
Other comments:	N/A

Table 166: The Supplying the version number Requirement.

ID:	R4.2.2
Title:	Supplying the version number
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MUST supply the APPLICATION's current version number when starting the MONITORING_TOOLS
Rationale:	The version number has to arrive from tools external to monitoring tools.
Supporting material:	N/A
Other comments:	N/A

Table 167: The Monitoring data extractions Requirement.

ID:	R4.3
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 168: The Monitoring data format transformations Requirement.

ID:	R4.4
Title:	Monitoring data format transformations
Priority of accomplishment:	Must have
Type:	Requirement
Description:	MONITORING_TOOLS MUST perform data transformation when the data is retrieved from the data warehouse.
Rationale:	Tools may require data in different formats in order to function. This transformation from the DW internal format to the required format is done at data retrieval.
Supporting material:	N/A
Other comments:	cleaning, normalization, projection, windowing in time series,

Table 169: The Monitoring data retention policy Requirement.

ID:	R4.5
Title:	Monitoring data retention policy
Priority of accomplishment:	Must have
Type:	Requirement
Description:	MONITORING_TOOLS MUST be able to set and enforce a policy on how long any monitoring entry may be preserved before deletion.
Rationale:	The solution has to observe both the space restrictions and any legal requirements for preserving or deleting the monitoring records.
Supporting material:	N/A
Other comments:	N/A

Table 170: The Monitoring data access restrictions Requirement.

ID:	R4.6
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 171: The Monitoring tools REST API Requirement.

ID:	R4.7
Title:	Monitoring tools REST API

Priority of accomplishment:	Must have
Type:	Requirement
Description:	MONITORING_TOOLS MUST expose their functionality using simple REST API.
Rationale:	This interface will facilitate querying, data transformation and extraction tasks.
Supporting material:	N/A
Other comments:	The REST interface will support monitoring data storage, retrieval, transformation, versioning etc.

Table 172: The Monitoring Visualization Requirement.

ID:	R4.8
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 173: The Data Warehouse replication Requirement.

ID:	R4.9
Title:	Data Warehouse replication
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The data warehouse COULD have replication capabilities.
Rationale:	Replication will offer increased availability and storage size in case monitoring data collected will be very large. Initially, we will adopt a centralized deployment.
Supporting material:	N/A
Other comments:	N/A

Table 174: The Resource consumption breakdown Requirement.

ID:	R4.11
Title:	Resource consumption breakdown
Priority of accomplishment:	Must have

Type:	Requirement
Description:	The DEVELOPER MUST be able to see via the ENHANCEMENT_TOOLS the resource consumption breakdown into its atomic components.
Rationale:	Existence of different abstraction levels between design concepts (e.g., abstractions in the DICE profile) and runtime measurements hides the details on what high-level request effectively generated the request for data.
Supporting material:	R4IDE1
Other comments:	N/A

Table 175: The Bottleneck Identification Requirement.

ID:	R4.12
Title:	Bottleneck Identification
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS MUST indicate which classes of requests represent bottlenecks for the application in a given deployment.
Rationale:	N/A
Supporting material:	R4IDE2
Other comments:	N/A

Table 176: The Semi-automated anti-pattern detection Requirement.

ID:	R4.13
Title:	Semi-automated anti-pattern detection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS MUST feature a semi-automated analysis to detect and notify the presence of anti-patterns in the application design.
Rationale:	N/A
Supporting material:	N/A
Other comments:	Anti-patterns will most probably use both UML information combined with monitoring data.

Table 177: The Refactoring methods Requirement.

ID:	R4.14
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 178: The Enhancement tools version difference Requirement.

ID:	R4.16
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 179: The Enhancement tools data acquisition Requirement.

ID:	R4.17
Title:	Enhancement tools data acquisition
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS SHOULD perform its operations by retrieving the relevant monitoring data from the MONITORING_TOOLS.
Rationale:	Local data processing appears more flexible than processing directly inside the data warehouse.
Supporting material:	N/A
Other comments:	N/A

Table 180: The Enhancement tools model access Requirement.

ID:	R4.18
Title:	Enhancement tools model access
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS MUST be able to access the DICE profile model associated to the considered version of the APPLICATION.
Rationale:	Parameter inference and anti-pattern detection need UML model.

Supporting material:	N/A
Other comments:	N/A

Table 181: The Parameterization of simulation and optimization models. Requirement.

ID:	R4.19
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 182: The Model parameter uncertainties Requirement.

ID:	R4.20
Title:	Model parameter uncertainties
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The REQ_ENGINEER COULD express uncertainty on some performance/reliability input parameters (e.g., execution times) in the DICE profile by means of a prior distribution or an interval. The ENHANCEMENT_TOOLS COULD take into account these parameters to esti
Rationale:	DoW mentions Bayesian estimation techniques. These techniques can explicitly account for the uncertainty provided by the REQ_ENGINEER.
Supporting material:	R4IDE3
Other comments:	This requirement may be alternatively stated as part of WP2 or WP3, since it also affects the DICE profile. The requirement would expand the scientific impact of the tool, but if too complex to implement it might be ignored without major consequences.

Table 183: The Model parameter confidence intervals Requirement.

ID:	R4.21
Title:	Model parameter confidence intervals
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The ENHANCEMENT_TOOLS COULD return confidence intervals for each inferred parameter of the performance and reliability models.

Rationale:	The WP3 models require to provide a number of parameters, such as CPU speeds. These will be inferred by the ENHANCEMENT_TOOLS of WP4 from the monitoring data. However, the estimation is subject to uncertainties so confidence intervals could be provided to the WP3 tools to quantify such uncertainty. If the CI is too wide, we might issue a warning in SIMULATION_TOOLS that the prediction is not robust.
Supporting material:	R4IDE4
Other comments:	N/A

Table 184: The Time-based ordering of monitoring data entries Requirement.

ID:	R4.22
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 185: The Data size trends Requirement.

ID:	R4.23
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 186: The Anomaly detection in APPLICATION quality Requirement.

ID:	R4.24
Title:	Anomaly detection in APPLICATION quality
Priority of accomplishment:	Must have
Type:	Requirement

Description:	MONITORING_TOOLS MUST provide means to detect anomalies in APPLICATION's quality after deployment
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 187: The Unsupervised Anomaly Detection Requirement.

ID:	R4.24.1
Title:	Unsupervised Anomaly Detection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The MONITORING_TOOLS must be able to detect anomalies from the APPLICATION using unsupervised methods. It is assumed that normal data instances lie closer to their closest centrid while anomalies are far away.
Rationale:	Monitored data may come in unlabeled (training dataset hard to create) form thus it is important to detect anomalies based on unsupervised methodology. It is assumed that normal data instanes are more frequent than anomalies.
Supporting material:	N/A
Other comments:	N/A

Table 188: The Supervised Anomaly Detection Requirement.

ID:	R4.24.2
Title:	Supervised Anomaly Detection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The MONITORING_TOOLS must be able to detect anomalies from the APPLICATION using supervised methods.
Rationale:	Creation of training dataset can be created thus it is posible to train predictive models based in supervised methodology.
Supporting material:	N/A
Other comments:	N/A

Table 189: The Contextual Anomalies Requirement.

ID:	R4.24.3
Title:	Contextual Anomalies
Priority of accomplishment:	Should have
Type:	Domain Assumption
Description:	The MONITORING_TOOLS should be able to detect that data instances of a given APPLICATION are anomalouse in a specific instance but not otherwise.

Rationale:	This is induced by the structure of the dataset and has to be specified as part of the problem formulation using the MONITORING_TOOLS. Data instances must be defined using: contextual attributes and behavioural attributes. Time-series data.
Supporting material:	N/A
Other comments:	N/A

Table 190: The Collective anomalies Requirement.

ID:	R4.24.4
Title:	Collective anomalies
Priority of accomplishment:	Should have
Type:	Domain Assumption
Description:	The MONITORING_TOOLS must be able to detect that a collection of related data instances of a given APPLICATION can be anomalouse with respect to the entire colleted dataset.
Rationale:	Data instances might not be anomalouse by themselves.This type of anomalies occur when the data instances are related. Sequence data.
Supporting material:	N/A
Other comments:	N/A

Table 191: The Predictive Model saving for Anomaly Detection Requirement.

ID:	R4.24.5
Title:	Predictive Model saving for Anomaly Detection
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The MONITORING_TOOLS must be able to save the predictive model trained using monitored APPLICATION data. These models can be reused and serve as a bootstrap for future predictive models.
Rationale:	Two APPLICATIONS can be similar or a single APPLICATION can have many versions thus a trained predictive model can be reused or can serve as a starting point. Can use PMML format.
Supporting material:	N/A
Other comments:	N/A

Table 192: The Semi-automated data labelling Requirement.

ID:	R4.24.6
Title:	Semi-automated data labelling
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The MONITORING_TOOLS COULD have the capability to insert labeled anomalous data instances in order to create training datasets for supervised

	training for Anomaly detection.
Rationale:	As anomalouse instances are far fewer than normal data instances (unbalanced class distribution) the insertion of labeled anomalies can help create a more viable predictive model. Optaining fully labeled data is most often unfeasible.
Supporting material:	N/A
Other comments:	N/A

Table 193: The Adaptation of thresholding Requirement.

ID:	R4.24.7
Title:	Adaptation of thresholding
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The MONITORING_TOOLS (anomaly detection tool) COULD ask feedback to the user about the predefined threshold used to detect an outlier and adjust based on the feedback received.
Rationale:	A given anomaly detection result could be scored by the user. A simple algorithm could interpret this to refine the threshold.
Supporting material:	N/A
Other comments:	N/A

Table 194: The Visualization of analysis results Requirement.

ID:	R4.25
Title:	Visualization of analysis results
Priority of accomplishment:	Should have
Type:	Requirement
Description:	ENHANCEMENT_TOOLS SHOULD be capable of visualizing analysis results
Rationale:	N/A
Supporting material:	R4IDE5
Other comments:	N/A

Table 195: The Report generation of analysis results Requirement.

ID:	R4.26
Title:	Report generation of analysis results
Priority of accomplishment:	Should have
Type:	Requirement
Description:	Both ANOMALY_TRACE_TOOLS and ENHANCEMENT_TOOLS SHOULD be able to generate reports with analysis results
Rationale:	This feature is needed: a) for when DEVELOPER/ARCHITECT needs to make a decision and make changes manually, b) to create history of changes (may be

	useful)
Supporting material:	N/A
Other comments:	N/A

Table 196: The Propagation of changes/automatic annotation of UML models Requirement.

ID:	R4.27
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 197: The Safety and privacy properties loading Requirement.

ID:	R4.28
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:	R4IDE6
Other comments:	N/A

Table 198: The Definition of time window of interest for safety/privacy properties Requirement.

ID:	R4.28.1
Title:	Definition of time window of interest for safety/privacy properties
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ANOMALY_TRACE_TOOLS MUST allow the DEVELOPER/ARCHITECT to choose the time window of interest, which must be considered when choosing the traces to be analyzed.
Rationale:	We do not want to analyze the whole history of the application, but only a slice, which is selected by the user

Supporting material:	N/A
Other comments:	Trace checking is not a real-time analysis of a stream of events; it is done in batch mode (see also R4.30), so the user should select the window of interest

Table 199: The Mechanisms for the definition of the time window of interest for safety/privacy propertiesRequirement.

ID:	R4.28.1.1
Title:	Mechanisms for the definition of the time window of interest for safety/privacy properties
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The ANOMALY_TRACE_TOOLS COULD offer the DEVELOPER/ARCHITECT different ways to choose the time window of interest; the time window could be indicated though a size (to computed in the past from the current instant), or using a starting and ending event.
Rationale:	We might want to give the user some flexibility in how the slice of the runtime history of the application to be analyzed is chosen.
Supporting material:	N/A
Other comments:	N/A

Table 200: The Event occurrences detection for safety and privacy properties monitoring Requirement.

ID:	R4.29
Title:	Event occurrences detection for safety and privacy properties monitoring
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ANOMALY_TRACE_TOOLS MUST be able to retrieve, depending on the properties to be checked, the relevant data stored in the DW, and translate them into traces of relevant events for the trace checking
Rationale:	The ANOMALY_TRACE_TOOLS, and the trace checking tool in particular, requires as input traces of events of interest, which must be identified before they are fed to the tool. There is probably a translation to be performed from what is stored in the DW into the input format for the trace checking tool.
Supporting material:	N/A
Other comments:	This is similar/related to R4.4, but it is probably worth it to highlight this this issue. It is also linked to R4.32

Table 201: The Safety and privacy properties monitoring Requirement.

ID:	R4.30
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 cheking 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 202: The Safety and privacy properties result reporting Requirement.

ID:	R4.30.1
Title:	Safety and privacy properties result reporting
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The ANOMALY_TRACE_TOOLS MUST be able to notify the DEVELOPER/ARCHITECT when a safety/privacy property is violated by the application.
Rationale:	The trace checking tool must be able to give feedback to the developers
Supporting material:	N/A
Other comments:	This requirement is linked to R4.26, maybe it is a sub-requirement

Table 203: The Feedback from safety and privacy properties monitoring to UML models Requirement.

ID:	R4.31
Title:	Feedback from safety and privacy properties monitoring to UML models
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The ANOMALY_TRACE_TOOLS COULD provide feedback about safety/privacy properties violated at runtime in the UML DICE models
Rationale:	Providing feedback in the UML DICE models might help the DEVELOPER/ARCHITECT get a picture of where the problems are in the application
Supporting material:	N/A
Other comments:	N/A

Table 204: The Feedback from safety and privacy properties monitoring to UML models concerning violatedtime bounds Requirement.

ID:	R4.31.1
Title:	Feedback from safety and privacy properties monitoring to UML models concerning violated time bounds
Priority of accomplishment:	Could have
Type:	Requirement
Description:	In the feedback provided by the ANOMALY_TRACE_TOOLS to the DEVELOPER/ARCHITECT, the tools COULD highlight when a timing

	requirement is violated, and what is the value of the violation
Rationale:	The specific feedback about timing violations might help the DEVELOPER/ARCHITECT adjust the parameters of the models/properties
Supporting material:	R4IDE7?
Other comments:	N/A

Table 205: The Correlation between data stored in the DW and DICE UML models Requirement.

ID:	R4.32
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?

Table 206: The Relation between ANOMALY_TRACE_TOOLS and IDE Requirement.

ID:	R4.33
Title:	Relation between ANOMALY_TRACE_TOOLS and IDE
Priority of accomplishment:	Should have
Type:	Requirement
Description:	It SHOULD be possible to launch the ANOMALY_TRACE_TOOLS from the IDE
Rationale:	The idea is that the trace checking is performed starting from the elements that are described in the DICE UML model (see requirement R4.32). Hence, it makes sense that the tool is invoked from the UML IDE. The idea could be that the IDE has a link to the DW, and when the user asks for performing trace checking, the IDE queries the DW, retrieves the information for the trace checking, then feeds the ANOMALY_TRACE_TOOLS with the traces to be checked.
Supporting material:	R4IDE8
Other comments:	N/A

Table 207: The Monitoring for quality tests Requirement.

ID:	R4.34

Title:	Monitoring for quality tests
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The MONITORING_TOOLS MUST support and collect all the metrics relevant for the QTESTING_TOOLS
Rationale:	The quality testing tools rely on the data obtained by monitoring the runtime of the application during the test runs.
Supporting material:	N/A
Other comments:	N/A

Table 208: The Tag monitoring data with OSLC tags Requirement.

ID:	R4.35
Title:	Tag monitoring data with OSLC tags
Priority of accomplishment:	Must have
Type:	Requirement
Description:	MONITORING_TOOLS MUST tag monitoring data with OSLC tags
Rationale:	DICE tools need to show compliance with OSLC standard
Supporting material:	N/A
Other comments:	N/A

B.5. WP5 Requirements

B.5.1 Consolidated requirements

Table 209: Requirement for the continuous integration and versioning.

ID:	R5.34
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 210: Requirement for the graphical user interface for continuous integration.

ID:	R5.35
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 211: Requirement for the quality testing scope.

ID:	R5.36
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 212: Requirement for the extended quality testing scope.

ID:	R5.37
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 213: Requirements for the quality testing results.

ID:	R5.38
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 214: Requirement for the autonomy of deployment tools.

ID:	R5.39
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 215: Requirement for the scope of deployment tools.

ID:	R5.40
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 216: Requirement for the extendibility and flexibility of the deployment tools.

ID:	R5.41
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 217: Support of deployment tools for Platform-as-aService (PaaS) requirement.

ID:	R5.42
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

B.5.2 Detailed requirements

Table 218: The Versioning Requirement.

ID:	R5.1
Title:	Versioning
Priority of accomplishment:	Must have
Type:	Domain Assumption
Description:	Everything in the user’s project MUST be treated as code. All code MUST be versioned and the DEPLOYMENT_TOOLS and CI_TOOLS tools MUST involve the version information in their process.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 219: The Testing project Requirement.

ID:	R5.2
Title:	Testing project
Priority of accomplishment:	Must have
Type:	Domain Assumption
Description:	An ADMINISTRATOR MUST configure a project or an account in the TESTBED with resource quotas set to accommodate application tests.
Rationale:	The DICE tools will deploy and test the application in the TESTBED running either in the private or the public cloud. As a pre-requiste of the tests, the TESTBED needs to be pre-configured to allow provisionning of resources without going over the set quotas.
Supporting material:	resources: CPU, RAM, hard drive space, network connectivity project or account: an environment in the cloud permitting provisioning of a

	limited or an unlimited set of virtual machines
Other comments:	In the context of DICE development, we assume this will be in a testbed. Otherwise the development team has a private data centre or a community cloud computing account to be used.

Table 220: The Continuous integration tools deployment Requirement.

ID:	R5.3
Title:	Continuous integration tools deployment
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The ADMINISTRATOR MUST manually install and configure CI_TOOLS MUST upon installation of the CI_TOOLS and can be updated later on. The configuration MUST enable CI_TOOLS to access the TESTBED.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 221: The Translation of TOSCA models Requirement.

ID:	R5.4
Title:	Translation of TOSCA models
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MUST be able to translate TOSCA models from WP2 into the supported target configuration manager's DSL for orchestration
Rationale:	The specialised tools for configuring the environment and orchestrating applications (e.g., Chef) use their own DSL other than TOSCA.
Supporting material:	DSL: domain-specific language
Other comments:	N/A

Table 222: The Deployment plan support Requirement.

ID:	R5.4.1
Title:	Deployment plan support
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MUST be able to deploy all the DICE supported core building blocks.
Rationale:	DICE will provide support for the initial set of services that support use cases and basic needs.
Supporting material:	N/A

Other comments:	N/A

Table 223: The Translation tools autonomy Requirement.

ID:	R5.4.2
Title:	Translation tools autonomy
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MUST take all of its input from the TOSCA model and therefore MUST NOT require any additional user's input.
Rationale:	The DEPLOYMENT_TOOLS have to operate transparently for the users.
Supporting material:	N/A
Other comments:	N/A

Table 224: The Deployment plan contents Requirement.

ID:	R5.4.3
Title:	Deployment plan contents
Priority of accomplishment:	Must have
Type:	Domain Assumption
Description:	An automated deployment plan of the DEPLOYMENT_TOOLS MUST consist of an executable list of operations to deploy, configure, and start the application on the TESTBED.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 225: The Deployment plans execution tools Requirement.

ID:	R5.4.4
Title:	Deployment plans execution tools
Priority of accomplishment:	Should have
Type:	Domain Assumption
Description:	The DEPLOYMENT_TOOLS SHOULD rely on third-party runtime configuration and deployment tools.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 226: The Deployment tools transparency Requirement.

ID:	R5.4.5

Title:	Deployment tools transparency
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS SHOULD NOT expose to the ADMINISTRATOR details on the tools used to deploy and configure the application.
Rationale:	For ease of use and extensibility, the DEPLOYMENT_TOOLS should hide their inner details to the external world
Supporting material:	N/A
Other comments:	N/A

Table 227: The Deployment plans extendability Requirement.

ID:	R5.4.6
Title:	Deployment plans extendability
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MAY be extended by the ADMINISTRATOR with other building blocks not in the core set.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 228: The Deployment plans portability Requirement.

ID:	R5.4.9
Title:	Deployment plans portability
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS SHOULD be able to support more than one vendor's IaaS.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 229: The Deployment of the application in a test environment Requirement.

ID:	R5.4.7
Title:	Deployment of the application in a test environment
Priority of accomplishment:	Must have
Type:	Requirement

Description:	The DEPLOYMENT_TOOLS MUST provision the resources required by the application
Rationale:	Assuming that there is an application, its model and a set of quality test, a dedicated set of resources need to exist and be assigned to the tests.
Supporting material:	resources: CPU, RAM, hard drive space, network connectivity
Other comments:	N/A

Table 230: The Starting the monitoring tools Requirement.

ID:	R5.4.8
Title:	Starting the monitoring tools
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS MUST start the MONITORING_TOOLS for the application.
Rationale:	Monitoring tools are an essential part of the DICE quality testing tools.
Supporting material:	
Other comments:	N/A

Table 231: The User-provided initial data retrieval Requirement.

ID:	R5.5
Title:	User-provided initial data retrieval
Priority of accomplishment:	Must have
Type:	Requirement
Description:	CI_TOOLS MUST retrieve from the artifact repository or use input from the code versioning system any user-provided initial data
Rationale:	Applications may require initial data prepared by the DEVELOPER to be loaded in the databases. If the DEVELOPER prepares them in a dedicated place, the CI_TOOLS are responsible to retrieve them and have them loaded in the databases.
Supporting material:	artifact repository: a dedicated repository for built application programs and libraries and any additional data such as bulk data
Other comments:	N/A

Table 232: The Test data generation Requirement.

ID:	R5.6
Title:	Test data generation
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The QTESTING_TOOLS COULD be able to generate the initial input data for the APPLICATION

Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 233: The Data loading support Requirement.

ID:	R5.7
Title:	Data loading support
Priority of accomplishment:	Must have
Type:	Requirement
Description:	DEPLOYMENT_TOOLS and QTESTING_TOOLS MUST support bulk loading and bulk unloading of the data for the core building blocks.
Rationale:	DICE should support the core building blocks (e.g., technologies such as CEPH/HDFS, SQL, NoSQL) with the ability to load the inital data in a standard and documented form (eg SQL scripts, files, etc). DICE should also allow to unload that data (delete files, drop table, etc).
Supporting material:	N/A
Other comments:	N/A

Table 234: The Data loading hook Requirement.

ID:	R5.7.1
Title:	Data loading hook
Priority of accomplishment:	Must have
Type:	Requirement
Description:	DEPLOYMENT_TOOLS and QTESTING_TOOLS MUST provide a well-defined way to accept the initial bulk data that they can load.
Rationale:	This requirement provides to the DEVELOPER a way to prepare the initial data, which either DEPLOYMENT_TOOLS or QTESTING_TOOLS (depends on the use case) load into the databases.
Supporting material:	N/A
Other comments:	N/A

Table 235: The Definition of quality test Requirement.

ID:	R5.8
Title:	Definition of quality test
Priority of accomplishment:	Must have
Type:	Domain Assumption
Description:	A quality test of the QTESTING_TOOLS MUST include at least executable code to generate the workload for the application, a timeout, an experimental design that assign the levels of the factors, and a set of target monitoring metrics to be collected by the

Rationale:	N/A
Supporting material:	Workload may be artificial or from real-traces collected by the MONITORING_TOOLS.
Other comments:	N/A

Table 236: The Representative test configurations generation Requirement.

ID:	R5.8.1
Title:	Representative test configurations generation
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The QTESTING_TOOLS SHOULD avoid a full factorial design testing by means of experimental design methods
Rationale:	The space of possible combinations of parameters to test may become prohibitively large, requiring to long a time to test them all. The QTESTING_TOOLS must select a feasible, but representative subset.
Supporting material:	N/A
Other comments:	N/A

Table 237: The Starting the quality testing Requirement.

ID:	R5.8.2
Title:	Starting the quality testing
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MAY be invoked by the CI_TOOLS or by the QA_TESTER
Rationale:	Addresses the responsibility of executing the programs or scripts, which implement the quality assurance runs.
Supporting material:	N/A
Other comments:	N/A

Table 238: The Test run independence Requirement.

ID:	R5.8.3
Title:	Test run independence
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST ensure that no side effects from past or ongoing tests leak into the runtime of any other test.
Rationale:	Each test needs to be run independently from the other test runs. The test results should be as repeatable as possible.

Supporting material:	N/A
Other comments:	N/A

Table 239: The Test outcome Requirement.

ID:	R5.8.5
Title:	Test outcome
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST provide the test outcome to CI_TOOLS: success or failure
Rationale:	The outcome of each test must be a clear "success" of "failure". The tests with clear criteria of success or failure must provide the decision. The tests, which run a survey, benchmark or stress-test always succeed unless there is an error in the runtime.
Supporting material:	N/A
Other comments:	Relates to R5.16

Table 240: The User's unit and regression tests code execution inclusion Requirement.

ID:	R5.9
Title:	User's unit and regression tests code execution inclusion
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The CI_TOOLS MUST offer the ability to run unit tests and regression tests. The unit tests and regression tests SHOULD be written by the DEVELOPER, who SHOULD have the ability of choosing which ones to run.
Rationale:	Addresses the responsibility of executing the programs or scripts, which implement the quality assurance runs.
Supporting material:	N/A
Other comments:	N/A

Table 241: The Continuous integration tools dashboard Requirement.

ID:	R5.10
Title:	Continuous integration tools dashboard
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The CI_TOOLS SHOULD offer a dashboard that consolidates the view on the state of the application and the deployed components.
Rationale:	N/A
Supporting material:	N/A

Other comments:	N/A

Table 242: The Quality testing tools IDE integration Requirement.

ID:	R5.11
Title:	Quality testing tools IDE integration
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The IDE SHOULD provide the means to configure the QTESTING_TOOLS execution
Rationale:	Quality tests may come with parameters such as the number of tests to run or the duration of each tests, which the user should be able to change.
Supporting material:	N/A
Other comments:	N/A

Table 243: The Testing results feedback Requirement.

ID:	R5.12
Title:	Testing results feedback
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The CI_TOOLS MUST provide feedback to the DEVELOPER on the results of the unit tests.
Rationale:	The CI_TOOLS invoke the testing on the user's behalf. Therefore they must indicate what the QTESTING_TOOLS returned as their outcome.
Supporting material:	N/A
Other comments:	N/A

Table 244: The Test the application for efficiency Requirement.

ID:	R5.13
Title:	Test the application for efficiency
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST test the application's performance across various configurations.
Rationale:	N/A
Supporting material:	Reference metrics for performance and costs should be defined project-wise.
Other comments:	N/A

Table 245: The Test the application for reliability Requirement.

ID:	R5.14
Title:	Test the application for reliability
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST be tested for the application's ability to maintain the functionality and data integrity even when there are outages and faults in the supporting system.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 246: The Test the behaviour when resources become exhausted Requirement.

ID:	R5.14.1
Title:	Test the behaviour when resources become exhausted
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST provide the ability to saturate and exhaust resources used by the application.
Rationale:	DICE tools must enable getting a feedback on what happens when a resource is exhausted. The application may crash, corrupt data, request scale-up of infrastructure or stop gracefully.
Supporting material:	Source literature: The Pragmatic Programmer
Other comments:	N/A

Table 247: The Trigger deliberate outages and problems to assess the application’s behaviour under faultsRequirement.

ID:	R5.14.2
Title:	Trigger deliberate outages and problems to assess the application’s behaviour under faults
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST use the TESTBED's fault injection functionality to test the application's resilience.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 248: The Test the application for safety Requirement.

ID:	R5.15
Title:	Test the application for safety

Priority of accomplishment:	Could have
Type:	Requirement
Description:	The QTESTING_TOOLS COULD test the application for safety properties.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 249: The Test the application for data protection Requirement.

ID:	R5.15.1
Title:	Test the application for data protection
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The QTESTING_TOOLS COULD test the application for its ability to protect the data from unauthorized access.
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 250: The Provide monitoring of the quality aspect of the development evolution (quality regression)Requirement.

ID:	R5.16
Title:	Provide monitoring of the quality aspect of the development evolution (quality regression)
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The CI_TOOLS MUST record the results of each test and map them to the momentary project's (model, code etc.) version.
Rationale:	While the QTESTING_TOOLS produce the direct results of success or failure, it must be CI_TOOLS that ensure these results are stored and available for inspection of history.
Supporting material:	results: success/failure, quality indicators
Other comments:	See also R5.1 and R5.8.4

Table 251: The Quick testing vs comprehensive testing Requirement.

ID:	R5.17
Title:	Quick testing vs comprehensive testing
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The QTESTING_TOOLS MUST receive as input parameter the scope of the

	tests to be run.
Rationale:	Speed is important when designing and developing code. DICE should provide two (or more) profiles for testing: a quick one running only the representative tests, and a long one (for "overnight" tests) giving a more comprehensive assessment.
Supporting material:	N/A
Other comments:	N/A

Table 252: The Deployment configuration review Requirement.

ID:	R5.19
Title:	Deployment configuration review
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The CI_TOOLS MUST enable that ADMINISTRATOR assigns one or more users (including self) for reviewing the deployment configuration
Rationale:	Automated quality tests have to be complemented with the input from humans, who must be able to review the model, the parameters affecting the deployment, and also possibly the results of the quality tests.
Supporting material:	N/A
Other comments:	N/A

Table 253: The Build acceptance Requirement.

ID:	R5.20
Title:	Build acceptance
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The CI_TOOLS MUST NOT run the deployment of the application to pre-production if the quality test fail or the reviewers have not provided a positive score.
Rationale:	No build should be promoted to pre-production accidentally. ADMINISTRATOR or other actor has to have the means to block harmful updates.
Supporting material:	N/A
Other comments:	N/A

Table 254: The Deployment plans reuse Requirement.

ID:	R5.21
Title:	Deployment plans reuse
Priority of accomplishment:	Could have
Type:	Requirement

Description:	The DEPLOYMENT_TOOLS COULD detect the deployments plans that have been tried before and IDE COULD flag this to the user.
Rationale:	The ADMINISTRATOR may use this information to establish that a deployment plan is mature and error-free.
Supporting material:	N/A
Other comments:	N/A

Table 255: The Continuous integration tools access control Requirement.

ID:	R5.22
Title:	Continuous integration tools access control
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The access to CI_TOOLS SHOULD be protectable with good credentials (e.g., username and password or a single sign-on token)
Rationale:	In the environments where the access to code and the builds need to be restricted to only the authorised staff, the CI_TOOLS should enable setting up of accounts, roles of accounts, and prevent access to unauthorised users.
Supporting material:	N/A
Other comments:	N/A

Table 256: The Continuous integration tools IDE integration Requirement.

ID:	R5.23
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:	N/A

Table 257: The Running tests from IDE without committing to VCS Requirement.

ID:	R5.23.1
Title:	Running tests from IDE without committing to VCS
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The CI_TOOLS COULD provide an integration with the IDE that enables deployment and execution of tests on the user's local changes without committing the code into the VCS.

Rationale:	In some cases the DEVELOPER may want to run a test without committing the code into the repository.
Supporting material:	N/A
Other comments:	N/A

Table 258: The Flexiant platform simulated or induced faults Requirement.

ID:	R5.24
Title:	Flexiant platform simulated or induced faults
Priority of accomplishment:	Must have
Type:	Requirement
Description:	The TESTBED MUST enable simulating or inducing at least the following platform faults: High CPU usage, High Memory usage, Node Power outage, Network outage/ fault, Lack of resources
Rationale:	One set of problems an application may encounter is that a part of the host's resources are exhausted. The TESTBED in DICE will provide a controled and reliable way of inducing resource ourages.
Supporting material:	N/A
Other comments:	N/A

Table 259: The Recommender Engine and Optimization Requirement.

ID:	R5.27
Title:	Recommender Engine and Optimization
Priority of accomplishment:	Must have
Type:	Requirement
Description:	DEPLOYMENT_TOOLS (recommender engine) MUST retrieve from the OPTIMIZATION_TOOLS initial deployment parameters and recommend the vaules of the parameters that have not yet been set.
Rationale:	OPTIMIZATION_TOOLS from WP3 handle subset of deployment parameters, while this requirement is meant to augment/add additional deployment parameters.
Supporting material:	N/A
Other comments:	A requirement for configuration recommender engine

Table 260: The Brute-force approach for deployment configuration deployment Requirement.

ID:	R5.27.1
Title:	Brute-force approach for deployment configuration deployment
Priority of accomplishment:	Could have
Type:	Requirement
Description:	DEPLOYMENT_TOOLS (recommender engine) COULD employ trial and error approach to recommend the 'best' one, where different configurations would be tried on a data sample in order to evaluate the performance

	achieved, the best one being chosen in the end
Rationale:	Alternative to ML approach
Supporting material:	N/A
Other comments:	N/A

Table 261: The Recommender Engine API Requirement.

ID:	R5.27.2
Title:	Recommender Engine API
Priority of accomplishment:	Must have
Type:	Requirement
Description:	DEPLOYMENT_TOOLS MUST provide APIs to access recommender system (push data, get recommendations, etc)
Rationale:	N/A
Supporting material:	N/A
Other comments:	A command-line interface can probably work at the first release.

Table 262: The Induced faults in the guest environment Requirement.

ID:	R5.30
Title:	Induced faults in the guest environment
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The TESTBED COULD enable simulating or inducing at least the following VM Level faults: High CPU usage, High Memory usage, Network fault
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 263: The Reactions to problems in the runtime Requirement.

ID:	R5.31
Title:	Reactions to problems in the runtime
Priority of accomplishment:	Could have
Type:	Requirement
Description:	The DEPLOYMENT_TOOLS COULD provide the means to trigger special actions such as reconfiguration or problem notifications when problems are detected
Rationale:	N/A
Supporting material:	N/A
Other comments:	N/A

Table 264: The Testbed problem notifications Requirement.

ID:	R5.32
Title:	Testbed problem notifications
Priority of accomplishment:	Should have
Type:	Requirement
Description:	The TESTBED SHOULD output notifications of faults to at least one of the regular channels (RESTful URL subscription, e-mail, queue...)
Rationale:	The TESTBED needs to provide the means for sending notifications when it detects faults regardless of whether they occur deliberately or accidentally.
Supporting material:	N/A
Other comments:	N/A

Appendix C. Technical Scenarios

C.1. WP1 Scenarios

Table 265: The Stereotyping a UML diagram with the DICE profile to obtain a Platform-Indep. Model Scenario.

ID:	UC1.1
Title:	Stereotyping a UML diagram with the DICE profile to obtain a Platform-Indep. Model
Task:	T1.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	A technical person capable of designing and modelling a data intensive application models the Platform-Indep. UML Model stereotyped with the DICE profile
Pre-conditions:	UML diagram of domain model
Post-conditions:	Stereotyped diagram with DICE profile
Exceptions:	N/A
Data Exchanges:	N/A

Table 266: The Analysis, simulation, verification, feedback, and transformations until obtaining a deployment model Scenario.

ID:	UC1.2
Title:	Analysis, simulation, verification, feedback, and transformations until obtaining a deployment model
Task:	T1.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	IDE
Actor 3:	QA_TESTER
Actor 4:	N/A
Flow of Events:	The developer is a technical person capable of developing a data intensive application is guided through the DICE methodology to accelerate development and deployment of the data-intensive application with quality iteration. A Quality-Assessment expert may also run and examine the output of the QA testing tools in addition to the developer
Pre-conditions:	Stereotyped diagram with DICE profile
Post-conditions:	Architecture model, platform-specific model, QA models
Exceptions:	N/A

Data Exchanges:	N/A

C.2. WP2 Scenarios

Table 267: The Workflow Specification Scenario.

ID:	DS1
Title:	Workflow Specification
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	definition of the behavioral flow to be supported by the DIA. A possible view for a workflow specification consists of workflow specification items as sequences of actions to be performed on input data
Pre-conditions:	architect has clear ideas in mind on which processing type needs to be carried out
Post-conditions:	DIA is specified thorugh the DICE profile and does not contain inconsistent or undeployable Data-Intensive Elements
Exceptions:	N/A
Data Exchanges:	N/A

Table 268: The Cost Analysis Scenario.

ID:	DS2
Title:	Cost Analysis
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	OPTIMIZATION_TOOLS
Actor 3:	TRANSFORMATION_TOOLS
Actor 4:	N/A
Flow of Events:	actual monetary/resource value produced by the DIA against the tentative cost calculations for the DIA itself
Pre-conditions:	architect has a number of possible configuration alternatives to be analysed for architecture-level trade-off analysis.
Post-conditions:	Architect finds out the right compromise between deployment and management

	cost against desired performance or other non-functional requirements
Exceptions:	N/A
Data Exchanges:	N/A

Table 269: The Data-flow specification Scenario.

ID:	DS3
Title:	Data-flow specification
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEPLOYMENT_TOOLS
Actor 4:	N/A
Flow of Events:	the way in which the DIA retrieves, stores and forwards data to its internal or external environment
Pre-conditions:	architect has many possible sources from which data needs to be harvested and elicited or possibly filtered.
Post-conditions:	architect has defined a DIA with a specific policy embedded within it. The policy is consistent with provisioned infrastructure and able to process needed data.
Exceptions:	N/A
Data Exchanges:	N/A

Table 270: The Function Specification Scenario.

ID:	DS4
Title:	Function Specification
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	the way in which DIA main Map/Reduce functions are implemented and how other concerns influence the implementation choices
Pre-conditions:	architect has to define DIA and job contained therein. Architect knows elements and technologies involved but needs to specify the flow of involved functions and study the policies involved.

Post-conditions:	architect has specified the DIA in terms of the exchanges of control between DIA functions (e.g., map / reduce, etc.)
Exceptions:	N/A
Data Exchanges:	N/A

Table 271: The Framework Override Scenario.

ID:	DS5
Title:	Framework Override
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	the need for overriding default values used by target DIA frameworks such that ad-hoc setup may be performed
Pre-conditions:	architect needs to override specific framework parameters but overrides may reflect architectural modifications on the DIA under specification
Post-conditions:	Architect finds the right compromise between having a deployable DIA and framework optimisation profile
Exceptions:	N/A
Data Exchanges:	N/A

Table 272: The Defining Data-Splits Scenario.

ID:	DS6
Title:	Defining Data-Splits
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	architects need to specify how data is split, filtered and arranged across the processing nodes
Pre-conditions:	Architect knows or has chosen DIA general architecture, including and workflow function specification but needs to specify data-splits that are required for processing in the DIA

Post-conditions:	architect has defined which node processes which types of data or what chunk of data is to be processed in terms of size, time and maximum expense of resources
Exceptions:	N/A
Data Exchanges:	N/A

Table 273: The Topology Specification Scenario.

ID:	DS7
Title:	Topology Specification
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	Architects need to specifically model the structural view of the DIA architecture, intended as an interconnected series of components responding to structural constraints (e.g., the lambda architecture)
Pre-conditions:	architects are aware of constraints (e.g, social, organizational, technical, governamental) to designing and deploying certain topologies - these constraints are cost and scale drivers for building the DIA topology
Post-conditions:	architects specified a topology consistent with known constraints; constraints are perfectly mapped in the architecture topological specification and can be analysed for architectural refinement
Exceptions:	N/A
Data Exchanges:	N/A

Table 274: The Framework Control Scenario.

ID:	DS8
Title:	Framework Control
Task:	T2.1
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	IDE
Actor 3:	DEVELOPER
Actor 4:	N/A
Flow of Events:	architects and developers need to specify and agree upon the dynamics that regulate the operational behavior behind the target framework (e.g., Hadoop/MR). For example, framework control includes configuration details

	about controlling software (e.g., framework daemons, task controllers, etc.)
Pre-conditions:	architect is familiar with the constraints (e.g., cost) and how those constraints reflect on the architecture; developer is familiar with tweaks to be applied for framework control that might avail architecture constraints; a compromise can be sought for
Post-conditions:	constraints and operational tweaks for framework control are perfectly specified in the architecture model; further reasoning on the same model may be applied via further optimisation
Exceptions:	N/A
Data Exchanges:	N/A

C.3. WP3 Scenarios

Table 275: The Verification of reliability or performance properties from a DPIM/DTSM DICE annotated UMLmodel Scenario.

ID:	UC3.1
Title:	Verification of reliability or performance properties from a DPIM/DTSM DICE annotated UML model
Task:	
Priority:	REQUIRED
Actor 1:	QA_ENGINEER
Actor 2:	IDE
Actor 3:	TRANSFORMATION_TOOLS
Actor 4:	SIMULATION_TOOLS
Flow of Events:	1.- The QA_ENGINEER selects the model to be evaluated. 2.- IDE loads it (the model is loaded successfully). 3.- The QA_ENGINEER selects the metric(s), he/she wants to evaluate. 4.- IDE submits the evaluation request. 5.- TRANSFORMATION_TOOLS takes the DICE annotated UML model and, by applying the transformation rules, converts it into a formal model (e.g., a Stochastic Petri net). 6.- SIMULATION_TOOLS selects the appropriate solver (e.g., a simulator), analyzes the formal model and calculates the metric(s) value(s). 7.- IDE presents the results of the metric prediction and let the QA_ENGINEER to store them.
Pre-conditions:	There exists a DPIM/DTSM level UML annotated model.
Post-conditions:	The QA_ENGINEER gets information about the predicted metric value in the technological environment being studied
Exceptions:	N/A
Data Exchanges:	N/A

Table 276: The Verification of throughput from a DPIM DICE annotated UML model Scenario.

ID:	UC3.1.1
Title:	Verification of throughput from a DPIM DICE annotated UML model
Task:	
Priority:	REQUIRED
Actor 1:	QA_ENGINEER
Actor 2:	IDE
Actor 3:	TRANSFORMATION_TOOLS
Actor 4:	SIMULATION_TOOLS
Flow of Events:	1.- The QA_ENGINEER selects a DPIM model to evaluate. 2.- IDE loads it (the model is loaded successfully). 3.- The QA_ENGINEER enters into performance metrics and selects throughput. 4.- IDE submits the evaluation request. 5.- TRANSFORMATION_TOOLS takes the DICE annotated UML model and, by applying the transformation rules, converts it into Stochastic Petri net. 6.- SIMULATION_TOOLS selects the appropriate solver (in this case probably ¿simulation?), analyzes the formal model and calculates the value for throughput. 7.- IDE presents the results of the throughput prediction and let the QA_ENGINEER to store it joint to the model itself.
Pre-conditions:	N/A
Post-conditions:	N/A
Exceptions:	N/A
Data Exchanges:	N/A

Table 277: The Verification of safety and privacy properties from a DICE UML model Scenario.

ID:	UC3.2
Title:	Verification of safety and privacy properties from a DICE UML model
Task:	
Priority:	REQUIRED
Actor 1:	QA_ENGINEER
Actor 2:	IDE
Actor 3:	TRANSFORMATION_TOOLS
Actor 4:	VERIFICATION_TOOLS
Flow of Events:	1.- The QA_ENGINEER selects the model to be verified. 2.- IDE loads the model (the model is loaded successfully).

	3.- The QA_ENGINEER selects the (safety) property to be checked (possibly using templates). 4.- IDE submits the verification request. 5.- TRANSFORMATION_TOOLS takes the DICE UML model and the property to be verified and, by applying the transformation rules, converts them into a formal model that is suitable for verification (e.g., a temporal logic model). 6.- VERIFICATION_TOOLS selects the appropriate solver, analyzes the formal model against the desired property and determines whether the property holds for the modeled system or not. 7.- IDE presents the result, which indicates whether the property is fulfilled or not. If the property is violated, the IDE presents the outcome of the verification activity to QA_ENGINEER in the form of a trace of the modeled system that violates the property.
Pre-conditions:	There exists a UML model built using the DICE profile. A property to be checked has been defined through the DICE profile, or at least through the DICE IDE, by instantiating some pattern.
Post-conditions:	The QA_ENGINEER gets information about whether the property holds for the modeled system or not
Exceptions:	N/A
Data Exchanges:	N/A

Table 278: The Optimization of the deployment from a DDSM DICE annotated UML model with reliability andperformance constraints Scenario.

ID:	UC3.3
Title:	Optimization of the deployment from a DDSM DICE annotated UML model with reliability and performance constraints
Task:	T3.4
Priority:	REQUIRED
Actor 1:	ARCHITECT
Actor 2:	OPTIMIZATION_TOOLS
Actor 3:	SIMULATION_TOOLS
Actor 4:	N/A
Flow of Events:	1.- The ARCHITECT selects the DDSL model and the OPTIMIZATION_TOOLS loads it 2.- The ARCHITECT specifies the target deployment to be considered: private or public cloud 3.- In case of public cloud deployment, the ARCHITECT specifies the set of candidate Cloud providers to be considered in the design space exploration 4.-The ARCHITECT specifies the application SLAs that will be translated into optimization constraints (e.g. minimum reliability or jobs deadlines). 5.- The ARCHITECT specifies the set of candidate resource containers to be considered in the design space exploration or alternatively specifies minimum

	RAM and CPU capacity for individual resource containers. 6.- For each resource container, the ARCHITECT possibly provides profiling data (e.g. tasks duration on different VMs types) 7.- The OPTIMIZATION_TOOLS start the design space exploration and compute the reliability and performance metrics for candidate solutions through the SIMULATIONS_TOOLS (providing multiple DDSMs of the candidate solutions to be evaluated in parallel) 8.- The OPTIMIZATION_TOOLS output the number of resources and architecture if minimum cost that fulfil SLAs, providing the complete DDSL for the data intensive application. 9.- DDSL is pushed back to the DICE IDE
Pre-conditions:	There exists a DDSM level UML annotated model (where the number and possibly type of VMs are not specified). Cost are stored in the OPTIMIZATION_TOOLS internal resource DB
Post-conditions:	The ARCHITECT starts from a partial DDSM model and reasons about the optimal resource allocation considering the trade off cost/requirements. Multiple technology are analysed by providing multiple DDSMs throught what-if scenarios
Exceptions:	N/A
Data Exchanges:	N/A

C.4. WP4 Scenarios

Table 279: The Monitor a big data framework Scenario.

ID:	UC4.1
Title:	Monitor a big data framework
Task:	T4.1
Priority:	REQUIRED
Actor 1:	OPTIMIZATION_TOOLS
Actor 2:	MONITORING_TOOLS
Actor 3:	SIMULATION_TOOLS
Actor 4:	N/A
Flow of Events:	N/A
Pre-conditions:	Existing deployment of big data framework to be monitored.
Post-conditions:	Measured metrics stored and available in a DW
Exceptions:	N/A
Data Exchanges:	N/A

Table 280: The Metrics Specification Scenario.

ID:	UC4.1.1
Title:	Metrics Specification
Task:	T4.1
Priority:	REQUIRED
Actor 1:	OPTIMIZATION_TOOLS
Actor 2:	ENHANCEMENT_TOOLS
Actor 3:	SIMULATION_TOOLS
Actor 4:	N/A
Flow of Events:	N/A
Pre-conditions:	A UML model for the application has been defined.
Post-conditions:	The application UML model is annotated with requirements on the metrics to be collected.
Exceptions:	N/A
Data Exchanges:	N/A

Table 281: The Monitoring tools registration Scenario.

ID:	UC4.1.2
Title:	Monitoring tools registration
Task:	T4.1
Priority:	REQUIRED
Actor 1:	MONITORING_TOOLS
Actor 2:	N/A
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	Each Monitoring tool will: 1. Discover the the Data Warehousing component 2. Send its identifier and the list of available metrics to the Data Warehousing component 3. Negotiate the monitored metrics and acknowledge
Pre-conditions:	Test application MUST be successfully deployed on test environment. DW is installed.
Post-conditions:	N/A
Exceptions:	N/A
Data Exchanges:	N/A

Table 282: The Monitored Data Storage Scenario.

ID:	UC.4.1.3
Title:	Monitored Data Storage
Task:	T4.1
Priority:	REQUIRED
Actor 1:	MONITORING_TOOLS
Actor 2:	N/A
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. Monitoring tools connects to the DW. 2. Compute the metrics by performing ETL (extracts-tranform-load) type jobs. 3. Store resulting metrics in the DW.
Pre-conditions:	Existence of data collection tools.
Post-conditions:	Recorded and computed metrics stored and available in a DW
Exceptions:	N/A
Data Exchanges:	N/A

Table 283: The Data Warehouse query Scenario.

ID:	UC4.2
Title:	Data Warehouse query
Task:	T4.1
Priority:	REQUIRED
Actor 1:	MONITORING_TOOLS
Actor 2:	SIMULATION_TOOLS
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	N/A
Flow of Events:	1. Actors send a query to the DW for specific metrics and timeframe 2. DW sintactically validates the query
Pre-conditions:	N/A
Post-conditions:	N/A
Exceptions:	3. If the query is malformed then the actor receives an error message 4. If the query is correct then the actor receives a dataset as a result
Data Exchanges:	N/A

Table 284: The Data Cleaning Scenario.

ID:	UC4.3
Title:	Data Cleaning
Task:	T4.1
Priority:	RECOMMENDED
Actor 1:	ANOMALY_TRACE_TOOLS
Actor 2:	N/A
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. ANOMALY_TRACE_TOOLS connect to the DW 2. ANOMALY_TRACE_TOOLS specifies an event window 3. ANOMALY_TRACE_TOOLS specifies a data cleaning algorithm to be applied on selected window 4. ANOMALY_TRACE_TOOLS lauches the data cleaning task
Pre-conditions:	N/A
Post-conditions:	N/A
Exceptions:	N/A
Data Exchanges:	N/A

Table 285: The Metrics Visualization Scenario.

ID:	UC4.4
Title:	Metrics Visualization
Task:	T4.1
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ADMINISTRATOR
Actor 4:	N/A
Flow of Events:	1. Actors access the WUI 2. Actors choose what metrics have to be displayed 3. Actors choose the visualization form
Pre-conditions:	Web User Interface (WUI) formetrics visualization is accessible DW accessible Metrics defined
Post-conditions:	Metrics are displayed in WUI
Exceptions:	N/A
Data Exchanges:	N/A

Table 286: The Anomaly detection model training Scenario.

ID:	UC4.5
Title:	Anomaly detection model training
Task:	T4.2
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ANOMALY_TRACE_TOOLS
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. The DEVELOPER defines the training dataset. 2. DEVELOPER selects anomaly detection method 3. The ANOMALY_TRACE_TOOLS apply transformations based on ML method chosen for the dataset 4. The DEVELOPER validates the model 5. Model is saved in the DW for later predictions.
Pre-conditions:	Monitoring data available in the DW. Application open in the IDE. UC4.1.2 and UC4.1.3. Labelled monitoring dataset.
Post-conditions:	Predictive model available in the DW
Exceptions:	N/A
Data Exchanges:	N/A

Table 287: The Offline Anomaly detection Scenario.

ID:	UC4.6
Title:	Offline Anomaly detection
Task:	T4.2
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ANOMALY_TRACE_TOOLS
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. The DEVELOPER defines the dataset window (query data) 2. The DEVELOPER selects anomaly detection method

	- Predictive model - Unsupervised method 3. ANOMALY_TRACE_TOOLS performs the detection 4. ANOMALY_TRACE_TOOLS stores the result of the detection 5. ANOMALY_TRACE_TOOLS inform DEVELOPER of the outcome
Pre-conditions:	DW is accessible In case Predictive method is used, Predictive Models need to be available in the DW (as outcome of UC4.5)
Post-conditions:	Results stored in the DW
Exceptions:	N/A
Data Exchanges:	N/A

Table 288: The Detect safety/privacy properties violation Scenario.

ID:	UC4.7
Title:	Detect safety/privacy properties violation
Task:	T4.2
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ANOMALY_TRACE_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT activates the monitoring, and selects from the UML DICE model the propeties to be monitored 2. ANOMALY_TRACE_TOOLS load from the DW the traces to be checked, depending on the propeties to be monitored and on the chosen time window 3. ANOMALY_TRACE_TOOLS analyze the loaded traces 4. If a violation of one (or more) of the properties is detected, the ANOMALY_TRACE_TOOLS report what properties are violated to the user
Pre-conditions:	Monitoring data available in the DW. Application open in the IDE. Existence of links between data stored in the DW with elements of the UML model.
Post-conditions:	N/A
Exceptions:	N/A
Data Exchanges:	N/A

Table 289: The Anti-pattern detection Scenario.

ID:	UC4.8
Title:	Anti-pattern detection
Task:	T4.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT requests to detect anti-patterns in the current design 2. ENHANCEMENT_TOOLS analyzes the current UML models and returns an indication of possible anti-patterns 3. Possibly, some visualization of the anti-pattern is given
Pre-conditions:	1. Knowledge of the software architecture (if it is simulated through PCM) or 2. Experimental data and user profiles to obtain performance metrics. 3. Performance metrics (e.g. response time, throughput, CPU utilisation etc.) 4. Metamodel wit
Post-conditions:	1. Detected antipatterns with a rank (the possibility of their influence on the performance degradation). 2. Possibly set of refactoring solutions (new architecture configurations/designs) for the DEVELOPER/ARCHITECT to choose from
Exceptions:	N/A
Data Exchanges:	N/A

Table 290: The Anti-pattern driven architectural refactoring Scenario.

ID:	UC4.8.1
Title:	Anti-pattern driven architectural refactoring
Task:	T4.3
Priority:	OPTIONAL
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT requests to detect anti-patterns in the current design 2. ENHANCEMENT_TOOLS analyzes the current UML models and returns an

	indication of possible anti-patterns 3. Possibly, some visualization of the anti-pattern is given 4. From the DICE IDE, DEVELOPER/ARCHITECT requests to ENHANCEMENT_TOOLS to suggest a possible refactoring to address the anti-pattern 5. A refactoring plan is shown inside the IDE and, if confirmed by the DEVELOPER/ARCHITECT, executed
Pre-conditions:	N/A
Post-conditions:	N/A
Exceptions:	N/A
Data Exchanges:	N/A

Table 291: The Bottleneck detection based on testing data Scenario.

ID:	UC4.9
Title:	Bottleneck detection based on testing data
Task:	T4.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT requests to detect anti-patterns in the current design 2. ENHANCEMENT_TOOLS analyzes the current UML models and highlights software or hardware bottlenecks based on the testing results 3. Possibly, some visualization of the bottlenecks is given
Pre-conditions:	Monitoring data available in the DW
Post-conditions:	1. Performance model (and software model?), annotated with the results of performance analysis 2. Notifications to the DEVELOPER about the presence of bottlenecks.
Exceptions:	N/A
Data Exchanges:	N/A

Table 292: The Automatic extraction of model parameters Scenario.

ID:	UC4.10

Title:	Automatic extraction of model parameters
Task:	T4.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT requests to update the model parameters (e.g., expected execution times), with an indication of the time granted to the ENHANCEMENT_TOOLS for the analysis 2. A dialog window is shown to select the model parameters to update 3. The IDE invokes ENHANCEMENT_TOOLS 4. An updated DICE profile is returned to the IDE
Pre-conditions:	Monitoring data available in the DW
Post-conditions:	DICE profile populated with new parameters
Exceptions:	N/A
Data Exchanges:	N/A

Table 293: The Quality regression Scenario.

ID:	UC4.11
Title:	Quality regression
Task:	T4.3
Priority:	RECOMMENDED
Actor 1:	DEVELOPER
Actor 2:	ARCHITECT
Actor 3:	ENHANCEMENT_TOOLS
Actor 4:	IDE
Flow of Events:	1. From the DICE IDE, DEVELOPER/ARCHITECT requests to examine quality regressions in two versions of the application 2. The IDE invokes ENHANCEMENT_TOOLS 3. ENHANCEMENT_TOOLS analyses quality differences between versions by operating directly on the monitoring data 4. Results are returned to the IDE
Pre-conditions:	Monitoring data available in the DW for versions to be compared
Post-conditions:	Quality regression results returned to IDE
Exceptions:	N/A
Data Exchanges:	N/A

C.5. WP5 Scenarios

Table 294: The Building the configuration description Scenario.

ID:	U5.1
Title:	Building the configuration description
Task:	T5.1
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	DEPLOYMENT_TOOLS
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. The DEVELOPER (or CI_TOOLS) requests building of the application 2. The IDE or the CLI invoke DEPLOYMENT_TOOLS with the TOSCA model of the configuration as the input 3. The DEPLOYMENT_TOOLS return the recipes in the DSL of the configuration management, mapping the TOSCA model
Pre-conditions:	1.1. TOSCA model of the configuration available. Or 1.2. tools for building TOSCA model from UML model installed
Post-conditions:	1. Configuration of the application available in the selected DSL.
Exceptions:	A configuration manager such as Chef is not compatible with TOSCA. The configurations are considered an artifact of building an application.
Data Exchanges:	DSL: Domain-Specific Language VCS: version control system (e.g., git or Subversion)

Table 295: The Continuous deployment sequence Scenario.

ID:	U5.3
Title:	Continuous deployment sequence
Task:	T5.1, T5.2, T5.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	QA_TESTER
Actor 3:	ADMINISTRATOR
Actor 4:	N/A
Flow of Events:	1. Actor designs and develops or corrects a feature and make them into a build 2. Actor use IDE to indicate the the type of tests to run: quality tests or unit tests

	or both. Also, the actor selects the scope of the test to run. 3. Actor request runing the tests in the test environment 4. CI_TOOLS, DEPLOYMENT_TOOLS and QTESTING_TOOLS execute the requested model's deployment and application testing 5. Actor receive the results of the tests in the IDE 6. Actor review the build 7. The build is checked and approved by the actors, it is ready to be deployed to pre-production
Pre-conditions:	1. Application’s artifacts (compiled libraries and programs) available in the package repository 2. DICE tools for continuous delivery installed at the test bed 3. Testbed project or account available for hosting the application and tests
Post-conditions:	1. Results of the QTESTING_TOOLS 2. Review from QA_ENGINEER and ADMINISTRATOR on the acceptability of the build, containing acceptance or rejection with comments on suggested improvements and corrections 3. Build ready for pre-production
Exceptions:	DICE supports Continuous integration and continuous deployment by letting the deployment and testing tasks be fully automated, while giving human users the final say in promoting experimental builds to pre-production ones. This scenario provides a user's
Data Exchanges:	project or account: an environment in the cloud permitting provisioning of a limited or an unlimited set of virtual machines

Table 296: The Continuous integration sequence Scenario.

ID:	U5.4
Title:	Continuous integration sequence
Task:	T5.2
Priority:	REQUIRED
Actor 1:	CI_TOOLS
Actor 2:	DEPLOYMENT_TOOLS
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. The CI_TOOLS receive DEVELOPER's request from IDE or a time-based tigger to execute a deployment-testing job 2. CI_TOOLS retrieve the code of the application and the models from the VCS 3. CI_TOOLS call DEPLOYMENT_TOOLS to translate TOSCA model into target configuration tool DSL blueprint 4. CI_TOOLS provide the DSL blueprint to DEPLOYMENT_TOOLS, which provision or reconfigure the TESTBED, install and configure the application, install MONITORING_TOOLS

	5. CI_TOOLS call QTESTING_TOOLS to execute the tests 6. CI_TOOLS collect the results of the tests 7. DEVELOPER inspects the results in the IDE or in the CI_TOOLS dashboard
Pre-conditions:	1. A job in the CI_TOOLS configured to run with a specific scope and tests 2. Application's artifacts (compiled libraries and programs) available in the package repository 3. TOSCA model of the configuration available.
Post-conditions:	1. Application deployed in the TESTBED 2. Results of the quality tests are available
Exceptions:	N/A
Data Exchanges:	DSL: Domain-Specific Language

Table 297: The Obtaining configuration recommendation Scenario.

ID:	U5.5
Title:	Obtaining configuration recommendation
Task:	T5.1
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	DEPLOYMENT_TOOLS
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. DEVELOPER provides the model, fixed parameters and free parameters as an input to DEPLOYMENT_TOOLS 2. DEPLOYMENT_TOOLS provide recommended values for the free parameters, optionally quantified with the quality criteria (reliability, efficiency, safety) 3. DEVELOPER selects from the recommended values to fix all of the parameters
Pre-conditions:	1. Model of the application (WP2) 2. Free/fixed parameters in the model (WP2) 3. Ouput of OPTIMIZATION_TOOLS proposing additional fixed parameters (WP3)
Post-conditions:	1. Deployment configuration with parameters set to optimal and recommended values
Exceptions:	OPTIMIZATION_TOOLS and DEPLOYMENT_TOOLS help assign a complimentary set of parameter values (e.g., number of Hadoop mappers and reducers)
Data Exchanges:	

Table 298: The One-click deployment and testing Scenario.

ID:	U5.12
Title:	One-click deployment and testing
Task:	T5.2
Priority:	OPTIONAL
Actor 1:	DEVELOPER
Actor 2:	IDE
Actor 3:	CI_TOOLS
Actor 4:	N/A
Flow of Events:	1. DEVELOPER clicks a "Deploy and test now" button in the IDE 2. IDE sends to the CI_TOOLS the DEVELOPER's current project and code changes without making a commit into the VCS 3. CI_TOOLS trigger the deployment and quality testing as normal 4. IDE displays the test outcome
Pre-conditions:	1. IDE configured with the parameters of the CI_TOOLS (host, port, user credentials, ...)
Post-conditions:	1. The application deployed and tested 2. Test outcome available to the DEVELOPER 3. Monitoring data of the test run available 4. If the outcome is positive, the developer can decide to commit the changes into the VCS and have the commit verified and revi
Exceptions:	The scenario addresses the situations where the developers want to test the build before committing the changes into the current branch of the VCS. Also they enable quick execution of tests useful for exploring the effect of small changes in the models, p
Data Exchanges:	

Table 299: The Configuration recommender engine training Scenario.

ID:	U5.6
Title:	Configuration recommender engine training
Task:	T5.1
Priority:	REQUIRED
Actor 1:	DEPLOYMENT_TOOLS
Actor 2:	
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. The actor obtains the monitoring data from the MONITORING_TOOLS 2. The actor uses the model from the previous version (not build!) as the initial

	model in training 3. The actor updates the training model for future recommendations
Pre-conditions:	1. Application available in the artifact repository 2. Collected metrics from MONITORING_TOOLS obtained during the most recent QTESTING_TOOLS execution OR intial defaults are available.
Post-conditions:	1. The training model of the recommender engine has been updated for the current version
Exceptions:	Training of the recommender engine's machine learning model has to depend on the initial model from the previous version and the monitoring data of the latest tests.
Data Exchanges:	

Table 300: The Initial data preparation Scenario.

ID:	U5.7
Title:	Initial data preparation
Task:	T5.1, T5.3
Priority:	REQUIRED
Actor 1:	DEVELOPER
Actor 2:	DEPLOYMENT_TOOLS
Actor 3:	CI_TOOLS
Actor 4:	N/A
Flow of Events:	1. DEVELOPER prepares the database schemas (where applicable) and initial data in a documented way 2. DEVELOPER indicates in IDE or configuration which phase the data should be loaded at: deployment or testing 3. DEVELOPER commits the data into the code versioning system (small datasets), uploads them to a package/artifact/binary data repository or provides an URL where the data is available (datasets of any size). 4. DEVELOPER or CI_TOOLS invoke the deployment of the application 5. DEPLOYMENT_TOOLS load the data if so required in the step 2 6. QTESTING_TOOLS load the data if so required in the step 2
Pre-conditions:	1. Application model available in the code versioning system repository
Post-conditions:	1. Application deployed 2. Initial data loaded into the application
Exceptions:	Database schemas and initial data are often crucial parts of the application deployment. The data prepared by the designer or developer needs to be loaded during or right after the application's installation in order for the application to function correc
Data Exchanges:	code versioning system: subversion, git or other CVS maintaining the versioned progression of the project development package/artifact/binary data repository: any repository (web access, network file

	system, ftp etc.) which may contain built, pre-built o

Table 301: The Provisioning of the test resources Scenario.

ID:	U5.9
Title:	Provisioning of the test resources
Task:	T5.2
Priority:	REQUIRED
Actor 1:	DEPLOYMENT_TOOLS
Actor 2:	TESTBED
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. DEPLOYMENT_TOOLS retrieve the configuration in the DSL from the artifact repository 2. DEPLOYMENT_TOOLS assign existing available Virtual Machiens in the TESTBED for use in the deployment 3. DEPLOYMENT_TOOLS request the TESTBED to provision any additional Virtual Machines if the existing availble ones are not sufficient for the model 4. DEPLOYMENT_TOOLS install, configure and run the applications configured to run.
Pre-conditions:	1. Application's artifacts (compiled libraries and programs) are avaialable in the package repository. 2. Resources are available in the TESTBED to host the application
Post-conditions:	1. The application runs in the TESTBED's provisioned VMs.
Exceptions:	The models of the application need to be transformed into an actual set of running virtal machines, hosting the application to be tested.
Data Exchanges:	N/A

Table 302: The Performing the quality testing Scenario.

ID:	U5.10
Title:	Performing the quality testing
Task:	T5.3
Priority:	REQUIRED
Actor 1:	QTESTING_TOOLS
Actor 2:	MONITORING_TOOLS
Actor 3:	TESTBED
Actor 4:	N/A
Flow of Events:	1. QTESTING_TOOLS starts injecting load in the APPLICATION after

	signaling to MONITORING_TOOLS the start of a test 2. The test plan is executed taking into consideration the dimensions to be explored (eg. performance, reliability, etc) 3. If requested, QTESTING_TOOLS may access TESTBED APIs to perform the test
Pre-conditions:	1. A quality test has been requested in some scenario 2. Test resources have been provisioned
Post-conditions:	1. Test data has been collected by MONITORING_TOOLS
Exceptions:	N/A
Data Exchanges:	N/A

Table 303: The Testing the application against external faults Scenario.

ID:	U5.11
Title:	Testing the application against external faults
Task:	T5.3, T5.4
Priority:	REQUIRED
Actor 1:	QTESTING_TOOLS
Actor 2:	TESTBED
Actor 3:	N/A
Actor 4:	N/A
Flow of Events:	1. QTESTING_TOOLS select a pre-programmed or a random fault to occur 2. QTESTING_TOOLS request from the TESTBED to inject the selected fault 3. TESTBED executes the request by causing a fault 4. QTESTING_TOOLS run the quality tests to check if the application still responds with expected results
Pre-conditions:	1. Application deployed in the testbed and running 2. Monitorong tools active
Post-conditions:	1. Outcome of the test, which is any of the following: application ok, application responds with unexpected results, application no longer works 2. Monitoring data of the history before, during and after the fault.
Exceptions:	N/A
Data Exchanges:	N/A