DICE: Quality-Driven Development of Data-Intensive Cloud Applications

G. Casale[1], D. Ardagna[2], M. Artac[3], F. Barbier[4], E. Di Nitto[2], A. Henry[4], G. Iuhasz[7], C. Joubert[5], J. Merseguer[6], V. I. Munteanu[7], J. F. Pérez[1], D. Petcu[7], M. Rossi[2], C. Sheridan[8], I. Spais[9], D. Vladušič[3]

1: Imperial College London, UK; 2: Politecnico di Milano, Italy; 3: XLAB, Slovenia; 4: Netfective, France; 5: Prodevelop, Spain; 6: Universidad de Zaragoza, Spain; 7: Institute e-Austria Timisoara, Romania; 8: Flexiant Technologies, UK; 9: Athens Technology Center SA, Greece.

***Abstract*— Model-driven engineering (MDE) often features quality assurance (QA) techniques to help developers creating software that meets reliability, efficiency, and safety requirements. In this paper, we consider the question of how quality-aware MDE should support data-intensive software systems. This is a difficult challenge, since existing models and QA techniques largely ignore properties of data such as volumes, velocities, or data location. Furthermore, QA requires the ability to characterize the behavior of technologies such as Hadoop/MapReduce, NoSQL, and stream-based processing, which are poorly understood from a modeling standpoint. To foster a community response to these challenges, we present the research agenda of DICE, a quality-aware MDE methodology for data-intensive cloud applications. DICE aims at developing a quality engineering tool chain offering simulation, verification, and architectural optimization for Big Data applications. We overview some key challenges involved in developing these tools and the underpinning models.**

***Index Terms*— Big Data, quality assurance, model-driven engineering**

I. Introduction

Recent years have seen the rapid growth of interest for developing enterprise applications that use data-intensive technologies, such as Hadoop/MapReduce, NoSQL, and stream processing [Zak11]. These technologies are important in many application domains, from predictive analytics to environmental monitoring, from e-government to smart cities. However, quality assurance in the software engineering process for data-intensive applications is still in its infancy. We therefore believe that it is a timely moment to focus on this investigation and present a research agenda for DICE, a novel quality-aware MDE approach for data-intensive applications.

MDE is well-established as a paradigm for the design, development, deployment and evolution of complex software systems. Quality-aware MDE focuses in particular on defining software systems that meet service-level objectives. For example, UML MARTE[1] and PCM[2] add to MDE the ability to describe quality and timing requirements for a component-based application. The resulting models are processed by QA tools that automatically generate performance, scalability and reliability predictions for different deployment scenarios [Men2010, Mar10]. These predictions are useful to drive the early stages of the application design and to identify an optimal architecture.

Recently, quality-aware MDE approaches for component-based software systems have found application also in cloud software engineering [Ard11]. In this context, MDE relies on domain-specific languages (e.g., CloudML[3]**)** to capture cloud concepts and abstract the heterogeneity in the offerings of different providers. QA methods for cloud application extend the ones used in canonical MDE by explicitly accounting for pricing of cloud resources (e.g., on demand and spot prices) and by modeling quality risks in multi-tenant environments, such as performance variability throughout the business day [Fra13].

The growing importance of Big Data applications now calls to extend MDE and QA methods to better support Big Data technologies, which raises specific challenges. For example, data-intensive applications are often based on Hadoop/MapReduce. Therefore, functional and non-functional modeling annotations are needed to describe map and reduce tasks and relevant operations on network and data storage. In some classes of Big Data applications, direct acyclic graphs should also be supported to describe data transformations and data transfers. These abstractions are important to provide a complete description of the design-space of a Big Data application and thus enable automated reasoning on the best architectural and deployment choices. Unfortunately, the definition of models and QA tools that can address these needs is challenging. For example, modeling quality characteristics of Hadoop/MapReduce applications requires to: i) explicitly model the synchronization of the map and reduce processing phases; ii) characterize the impact of network latencies during the shuffle phases; iii) statistically characterize the execution times of each phase and its memory and storage requirements, which in turn depend on data volumes; iv) describe technology-specific queueing, scheduling and failure mechanisms. This puts a high barrier for implementing QA techniques in MDE for Big Data. However, automated tools are needed to bring QA to software developers not trained in quality engineering.

DICE aims at addressing this issue by making quality-aware MDE accessible to developers of Big Data applications

[1] http://www.omgmarte.org/

[2] http://www.palladio-simulator.com/science/palladio_component_model/

[3] http://cloudml.org/

through an automated MDE tool chain, DICE tools will rely on UML meta-models annotated with information about data, data processing and data transfers. The DICE QA tool chain will cover simulation, verification and architectural optimization. These tools will be coupled with feedback analysis methods to help the developer iteratively improve the application design, based on monitoring data obtained from test or production environments. In the rest of this paper, we discuss the challenges for realizing this vision and outline the research agenda for DICE. Section II gives a motivating example that illustrates the challenges of QA for Big Data applications. Section III outlines the DICE modeling approach. A technical approach is defined in Section IV. Section V briefly summarizes some related research initiatives. Concluding remarks are given in Section VI.

II. Towards Quality-aware MDE for Big Data

In this section, we identify and discuss in more details what we see as key challenges in quality-aware MDE for Big Data applications. We use a motivating example to provide context, then we evaluate the shortcomings of existing MDE approaches and discuss what additional features should be provided to address these limitations.

A. DICEnv Example

DataInc is a small software vendor selling cloud-based environmental software. The company managers have just signed a contract to develop DICEnv, a warning system for floods in rural regions. Local authorities will use DICEnv for hazard prediction by monitoring local environmental conditions collecting soil, weather, and water data through sensors and by fetching precipitations data from satellite image streams published by NASA. Hazard predictions on areas at risk will be shown to local authorities through a web interface. DICEnv exploits Big Data technologies and cloud capacity for online water simulations and MapReduce for batch processing of historical data.

As DICEnv is a critical system for citizen safety, local authorities impose strict quality requirements with the contractor. DICEnv is expected to remain up 24/7. Furthermore, in periods of heightened environmental hazards (e.g., during heavy rains), DICEnv should quickly ramp up data intake rates, as well as memory and compute capacities, to update more frequently the hazard management control room. Besides, risk-critical computations related to disaster hazards must meet deadlines at all times.

Unfortunately, the contract won by DataInc requires delivering an initial version of DICEnv within 3 months, capable of serving a small area, with the goal of increasing coverage of areas, sensors and compute capacity on a monthly basis. Yet, software developers are puzzled on how to implement a complex cloud application in such a short time. How could they satisfy all the quality requirements? What architecture should be adopted to take into account the future evolution of the system? How should they accelerate quality testing for this initial release?

B. Limitation of Existing MDE Approaches

The DICEnv example gives some insights on the challenges of offering a quality-aware MDE tool chain for Big Data applications. Here, we discuss these challenges using the reference OMG model-driven architecture, in particular the *Platform Independent Model* (PIM), which describes the behavior of the software while hiding the underlying technology platform, and the *Platform Specific Model* (PSM) level, which refines the PIM by mapping the design to a specific technology platform. Existing QA tools for quality-aware MDE tend to use information from both these layers.

PIM Layer Limitations. We argue that in the design of an application like DICEnv, existing MDE approaches would face limitations at the PIM layer, for example when expressing requirements for data transfer and data processing. Today, it is possible with MDE to express entity-relationship models, basic dependencies between components and data, field types and values, and data semantics. However, new MDE approaches are required to explicitly annotate at the PIM layer information such as:

- *static characteristics of data*: e.g., volumes, value, storage location, replication pattern, cost for accessing data via cloud storage services, known schedules of data transfers, data access control;
- *dynamic characteristics of data*, e.g., read rates, write rates, update rates, burstiness in data streams, caching;
- *data dependencies*, e.g., graph-based relationships between data archives and streams, for example to describe interdependencies in rates and characteristics due to complex-event processing.

In the DICEnv example, if the developers were to use a state-of-the-art MDE approach for cloud computing without the above annotations at the PIM layer, they would not be able to describe:

- individual dependencies between components and data streams, therefore it would be impossible for the QA tool chain receiving the PIM model to understand how a refactoring is going to affect latencies, costs and reliability for the data-intensive part of the application;
- the relationships between compute and memory requirements of individual software components and the volumes and I/O rates of the data, which would make it difficult to predict quality at design time.
- the lack of an explicit annotation for data characteristics would make it difficult to integrate in the QA tool chain a feedback analysis and performance anti-pattern detection capability, since the QA tool chain would not be in a condition to synchronize the models with monitoring data collected from the runtime.

These shortcomings call for enriching PIM with information about static and dynamic characteristics of data and data dependencies.

PSM Layer Limitations. Similar challenges arise at the PSM level, where the heterogeneity of Big Data technologies makes it difficult to identify common concepts across technology platforms. For the same reason, the automatic translation of PSM models into deployment plans is also challenging. For example, Hadoop-based clusters are highly configurable, with hundreds of available parameters ranging from distributed file system configuration to number of map and reduce tasks. Supporting this configuration complexity requires work to enrich the expressiveness of the PSM models and of the deployment plans, compared to those used in current MDE approaches, to encompass Big Data technologies and platforms. An emerging standard like TOSCA, the OASIS model for topology and orchestration specification in cloud applications, could be a candidate for these extensions. However, at present TOSCA is still agnostic of data. Also, TOSCA has no native support for Big Data applications, and no explicit notion of quality, but covers the abstractions needed to describe the deployment of cloud applications.

QA Tool chain limitations. Assuming that the PIM can provide the required data annotations and the PSM is sufficiently expressive to generate deployment and configuration plans for Big Data technologies, several challenges would then arise for the QA tool chain to ensure that these plans respect cost and quality constraints and they are optimal according to some objective function. In order to provide the predictions and decision-support features that are expected for quality-aware MDE, one would then need to develop transformations to automatically generate performance, reliability and safety models, and then analyze these models to extract quality metrics. Such metrics could then either be reported to the developer or used for exploring the design space of the application.

The key problem to address for optimal decision-making is that analytical models used today for performance and reliability evaluation, such as queueing networks, are meant to describe contention at processing resources, but they have limited expressiveness when it comes to correlating contention to memory consumption. This is a problem, because peak memory usage is a primary concern in Big Data applications. Furthermore, fork and joining of streams and phase synchronizations (e.g., map/reduce/shuffle) is complex to describe analytically in a way that preserves the tractability of the queueing models, although some initial works have been done in this area [Zha10]. Stochastic Petri nets appear more flexible in this respect, but their evaluation cost tends to be higher than queueing models since they often require simulation. However, simulation can be inefficient for optimal decision-making, since it is typically too slow for use in conjunction with non-linear programming algorithms.

Summarizing, several limitations and novel challenges exist in current MDE solutions that require major innovations in order to enable functional and quality modeling of Big Data application and defined effective QA tools.

III. DICE MDE Approach

The main goal of DICE is to define an MDE approach and a QA tool chain to continuously enhance data-intensive cloud applications with the goal of optimizing their service level. Summarizing the discussion in Section II, we believe that the methods and tools shown in Table 1 are required to provide a comprehensive quality-aware MDE approach for Big Data applications. The DICE IDE will guide the developer throughout this methodology. It will initially offer the ability to specify the data-intensive application through UML models and a novel DICE profile that will address the limitations outlined in Section II. From these models, the tool chain will guide the developer through the different phases of quality analysis (e.g., simulation and formal verification), deployment, testing, and acquisition of feedback data through monitoring. This data will then be processed and fed back to the IDE through the iterative quality enhancement tool chain, which will analyze runtime data to detect quality incidents and anti-patterns in the application design. This will provide feedbacks to guide the developer through cycles of iterative quality enhancement.

DICE profile	A novel data-aware UML profile to develop data-intensive cloud applications and annotate the design models with quality requirements.
DICE IDE	Integrated development environment with code generation to accelerate development.
Quality analysis	A tool chain to support quality-related decision-making composed by simulation, verification and optimization tools.
Iterative quality enhancement	A set of tools and methods for iterative design refinement through feedback analysis of monitoring data.
Deployment and testing	A set of tools to accelerate deployment and testing of data-intensive applications on private and public clouds.

Table 1. DICE Tools

A. DICE Profile: MDE for Data-Intensive Applications

Models in DICE should be formulated at three levels, called DPIM, DTSM, DDSM, which we discuss next.

DICE Platform Independent Model (DPIM). The DPIM model corresponds to the OMG MDA PIM layer and describes the behavior of the application as a directed acyclic graph that expresses the dependencies between computations and data. This model should also express source data formats, synchronization mechanisms in the computation logic, and quality requirements for both computation logic and data transfers.

Figure 2 shows a possible example of DPIM for an application including four Data Sources (DS1-DS4) and four Computational Logic elements (CL1-CL4). At the DPIM layer the designer can specify the data format (e.g., structured or semi-structured data, flat files, etc.) and indicate if the data is transferred between processing steps via a shared storage system (e.g., S1) or obtained from data streams (e.g., DS3 and DS4 flows). A computational logic element can process multiple flows both synchronously or asynchronously. Data locations, estimated size (e.g., 600-900 TB for DS1), computation logic workload (e.g., 200 requests/h for CL3) and service-level constraints (e.g., CL1 runtime less than 15 minutes) may also be specified.

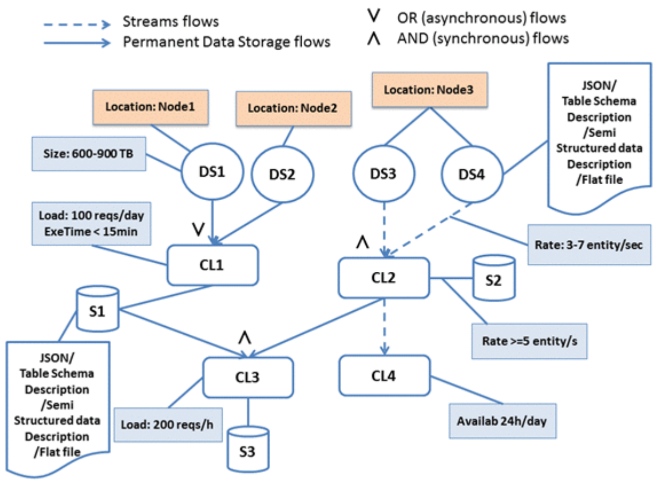

Figure 2. DICE platform independent model (DPIM)

DICE Platform and Technology Specific Model (DTSM). A DTSM, illustrated in Figure 3, consists of a refinement of the DPIM and includes some technology specific concepts, both for computational logic and data storage, but that are still independent of the deployment. For example, data and computational logic elements may be associated at the DTSM layer with specific technologies. DS1 and S1 may be required to be based on the Hadoop File System (HDFS), DS2 on a relational database (RDBMS), CL2 on complex event processing (CEP), and so forth.

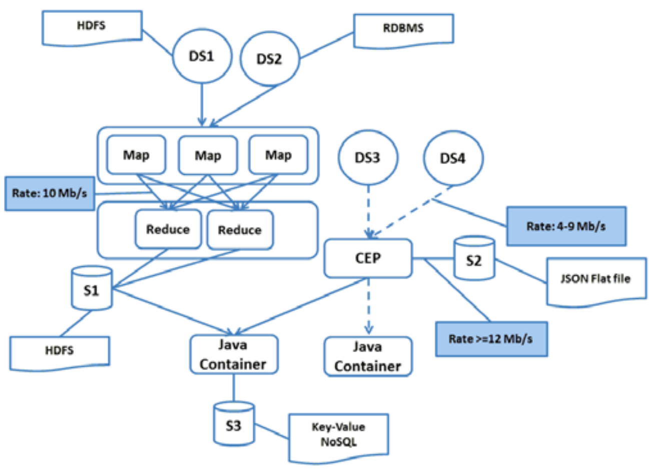

Figure 3 DICE Platform and Technology Specific Model (DTSM)

DICE Platform, Technology and Deployment Specific Model (DDSM). The DDSM, shown in Figure 4, is a specialization of the DTSM model which adds information about the technology in use and the application deployment characteristics. For example, the deployment may be specified at the DDSM layer with details on the system capacity (e.g., CL1 will be hosted on 50 EC2 Elastic MapReduce *xlarge* instances). DICE will help the developer deciding deployment characteristics by identifying through numerical optimization a deployment plan of minimum cost, subject to performance and reliability requirements. Additionally, deployment tools will be able to process the information provided by the DDSM to minimize the effort required to deploy the application. Transformations between DPIM, DTSM and DDSM models will be supported by the DICE tool chain.

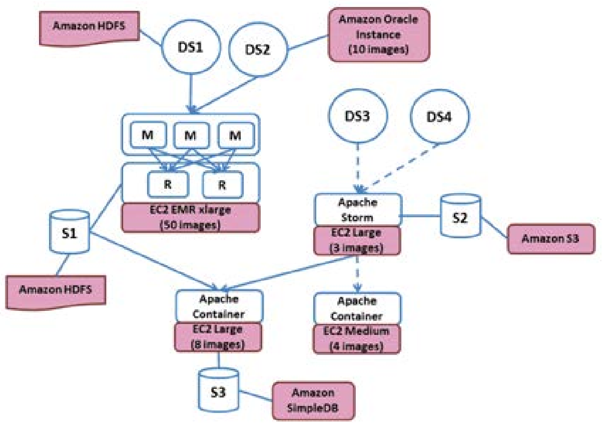

Figure 4 DICE Platform, Technology and Deployment Specific Model (DDSM)

B. Quality Annotations

The DICE profile will enable the design of data-intensive cloud applications. In particular, as highlighted in Section II, we envision that the DICE profile needs to include at least: (i) quantitative annotations on the availability of a data source or intermediate by-products resulting from a data transformation; (ii) annotations to specify rates, latencies and utilizations of resources, including the possibility to specify service level constraints on data transfers; (iii) annotations to specify costs of data-intensive applications; (iv) safety annotations that will be treated as hard constraints.

C. Deployment

The last set of requirements for the DICE approach to be effective concerns the development of appropriate tools to support the application deployment and initial testing. Ideally, the primary target of an MDE methodology for Big Data should be either private cloud applications or public cloud applications that can use cloud platform services for Big Data, such as Amazon Elastic MapReduce or cloud-based storage services. Automatic deployment and configuration from DDSM models could be achieved using extensions of tools such as Brooklyn, Puppet or Chef.

Continuous integration and QA testing approaches should be featured to validate and evolve the application code after initial deployment.

IV. QA Technical Approach

In this section, we identify and discuss some important quality dimensions for Big Data applications, followed by an initial proposal of a QA tool chain for MDE capable of analyzing these dimensions.

A. Quality Dimensions

In our view, a QA tool chain for data-intensive applications should focus at least on the following quality dimensions:

- *Data-aware reliability*: quantitative annotations on the availability of a data source and on the reliability of data transfer and data processing operations. The notions of data replication and integrity need to be explicitly correlated to the reliability requirements.
- *Data-aware efficiency in resource utilization*: data operations consume resources such as memory, network bandwidth, storage and cache space, CPU time, software pool resources, etc. Service requirements at these resources need to be expressible in the DICE model through rates and latencies. Annotations should also give the ability to express service level objectives, such as maximum acceptable resource utilization levels.
- *Data-aware efficiency in costs*: data efficiency also deals with costs, which are more complex to assess for data-intensive applications than for canonical web applications. One example is the quantification of network costs, which can vary if a stream transports some data between components operating on the same cloud or between a component and an end-user external to the cloud. This requires developing a novel annotation to relate deployment and data transfer characteristics with costs.
- *Data-aware safety*: annotations are needed to deal with constraints that must be guaranteed for safety reasons. QA tools need to support the specification of portions of the application that must logically and physically be isolated, together with the elicitation of formally correct requirements for a sequence of messages, states or events that relate application components.

UML profiles like MARTE and DAM[4] provide a suitable starting basis for extending the above dimensions in meta-models of Big Data applications. Other relevant baselines can be found in [Ber12].

B. Quality Tools

We now move to the problem of identifying a set of QA capable of fully exploiting the DICE profile and assess the quality dimensions we have described.

[4] https://bitbucket.org/mberenguer/marte-dam/wiki/Home

Quality analysis transformation tool. The goal is to map DICE profile models to quality analysis models. This entails the challenge of compensating uncertainties or ambiguities in the design specification, for example by restricting the use of ambiguous constructs or by interpreting them using predefined heuristics. A model-to-model transformation approach leveraging conceptual models should be used to define this tool.

Discrete-event simulation. The goal is to assess reliability and efficiency in Big Data applications. The main challenge is to assess cost and quality of given design scenarios, accounting for stochastic evolution of the environment (variability in the number of end-users, in the capacity offered by cloud resources, in the number of data sources and in their performance characteristics). The approach could leverage simulation of stochastic models of the DICE application based for example on stochastic Petri nets or queueing networks.

Formal verification tools. Verification tools are needed to assess safety risks in Big Data applications. The challenge is to find design flaws causing order and timing violations in message and state sequences. A possible approach to cope with this challenge involves formal verification of DICE models through transformation into metric temporal logic formulae and use of bounded model checking. The tools could then be used to verify, for example, that deadlines are met, requirements that a certain data value be observed at a given instant, or the correct ordering and timing of a sequence of computations or application states.
The underpinning models can be based on a precise and metric notion of time that is exploited to precisely express timing constraints, as for example the ones offered by UML MARTE. Models can then be automatically analyzed through tools based on state-of-the-art techniques such as satisfiability modulo theories solvers. Quantifier-elimination techniques may be investigated to extend currently available temporal logic-based verification mechanisms with the possibility to generalize verification results to systems made of an arbitrary number of components of the same kind, in order to describe parallelism in data processing.

Architecture optimization tool. The goal is to find architectural improvements to optimise costs and quality. The challenge is to define algorithms to quickly find good designs given new requirements, which is a difficult challenge since simulation tends to be slow. However, adopting a decomposition-based analysis approach, where compute and memory requirements are analyzed and optimized separately, simulation-based evaluation of performance metrics may still be used in optimization program, since decomposition would reduce the number of decision variables in the optimization problem.
A possibility to make optimization more efficient is to resort to so-called fluid approximation of stochastic models, which enable the simulating the behaviour of a system using ordinary differential equations instead of discrete-event systems. Initial work to illustrate the gains of the fluid

approach in architectural design can be found in [Per13, Per15]. Fluid methods can provide very large speedups in optimization programs, at the expense of low model approximation accuracy for subsystems where parallelism levels are small. Nicely, fluid models become more accurate as the system scale grows. Therefore, since large scale applications are becoming increasingly common in Big Data, fluid techniques may be promising to support decision making for architectural optimization activities.

Feedback analysis. Feedback analysis requires the automated extraction from the monitored data of key parameters required to define simulation and verification models. This is a novel challenge for Big Data applications. For example, there is a shortage of techniques for automated parameterization of stochastic models involving for example fork/join synchronizations. These abstractions are required to model MapReduce workloads, where map, shuffle and reduce phases need to synchronize. This challenge may be addressed by defining techniques capable of extracting model parameters through log mining and statistical estimation methods.

Another major issue to be addressed is the existence of different abstraction levels between design concepts (i.e., abstractions in the DICE profile) and runtime measurements, since the latter are implementation-dependent while the former are abstract models. This calls for defining novel statistical estimation techniques to breakdown resource consumption into its atomic components on the end-to-end path of requests, correlate these atomic components with the modeling abstractions in the DICE profile, and later analyze this information to identify bottlenecks and quantify the levels of reliability and availability offered by the application.

V. Related Initiatives

In the literature there is a variety of platforms to support the MDE for cloud applications. For example MODAClouds (www.modaclouds.eu) offers a quality-aware model-driven approach and offers basic tools to support DevOps. However, MODAClouds focuses on multi-cloud and in the Big Data domain only supports NoSQL databases.

The SeaClouds project (www.seaclouds-project.eu) aims at giving to organisations the capability of "Agility after Deployment". It takes care of different aspects of the cloud development life-cycle, such as an open, generic and interoperable foundation to orchestrate parts of cloud-based applications. Since DICE focuses on design-time and testing, as opposed to runtime management, the results of projects such as SeaClouds may be integrated with the DICE tools to cover the runtime operation aspects not developed within the DICE vision.

The main objective of U-QASAR (www.uqasar.eu) is to create a flexible Quality Assurance, Control and Measurement Methodology to measure the quality of Internet-related software development projects and their resulting products. The methodology is based on knowledge services, whereas DICE emphasizes the integration of MDE with stochastic and nondeterministic models for verification.

VI. Conclusion

We have described the research agenda of DICE, a vision for a novel model-driven engineering approach specifically tailored to Big Data applications. We have identified several challenges that arise in this area due to limitations in current models and quality analysis tools that arise from the inability to fully describe data operations and data characteristics. The authors are working towards implementing the vision described in this paper as part of a novel European research and innovation action started in February 2015 (www.dice-h2020.eu). This initiative will apply the DICE MDE approach to industrial demonstrators in the domains of news and media processing, maritime operations, and e-government. Challenges to be undertaken in these demonstrators include the ability to cover social media stream data (news/media domain), analysis of positional real-time data (maritime operations), and data-intensive applications that can cope with legacy systems and legacy data formats (e-government).

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644869 (DICE).

References

[Ard12] D Ardagna, E Di Nitto, et al. MODAClouds: A model-driven approach for the design and execution of applications on multiple Clouds, Proceedings of MiSE 2012, 50-56.

[Ber12] S. Bernardi, J. Merseguer, D. C. Petriu. Dependability modeling and analysis of software systems specified with UML. ACM Computing Surveys, 45(1), p. 2, 2012.

[Deb11] P. Debois. Devops: A software revolution in the making?, J. Information Technology Management, 2011

[Men10] D. A. Menascé, J. M. Ewing, H. Gomaa, S. Malek, J. P. Sousa. A framework for utility-based service oriented design in SASSY. Proceedings of ACM/SPEC WOSP/SIPEW 2010, 27-36.

[Mar10] A. Martens, H. Koziolek, S. Becker, R. Reussner. Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. Proceedings of ACM/SPEC WOSP/SIPEW 2010, 105-116

[Fra13] D. Franceschelli, D. Ardagna, M. Ciavotta, E. Di Nitto. Space4Cloud: A tool for system performance and cost evaluation of cloud systems. Proceedings of MultiCloud workshop, 27-34, 2013.

[Per13] J. F. Perez and G. Casale. Assessing SLA compliance from Palladio component models. Proceedings of the 2nd Workshop on Management of resources and services in Cloud and Sky computing (MICAS), IEEE Press, 2013.

[Per15] J.F. Pérez, G. Casale, and S. Pacheco-Sanchez. Estimating Computational Requirements in Multi-Threaded Applications. IEEE Transactions on Software Engineering, to appear in 2015.

[Zha10] H. Zhao, C. H. Xia, Z. Liu, D. F. Towsley. A unified modeling framework for distributed resource allocation of general fork and join processing networks. Proceedings of ACM SIGMETRICS 2010: 299-310.

[Zik11] P. Zikopoulos, C. Eaton. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Osborne, 2011.