Towards a UML Profile for Data Intensive Applications

Abel Gómez, José Merseguer
Departamento de Informática e
Ingeniería de Sistemas
Universidad de Zaragoza
María de Luna 1, 50018 Zaragoza, Spain
{ abel.gomez | jmerse } @unizar.es

Elisabetta Di Nitto, Damian A. Tamburri
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milan, Italy
{ elisabetta.dinitto |
damianandrew.tamburri } @polimi.it

ABSTRACT

Data intensive applications that leverage Big Data technologies are rapidly gaining market trend. However, their design and quality assurance are far from satisfying software engineers needs. In fact, a CapGemini research shows that only 13% of organizations have achieved full-scale production for their Big Data implementations. We aim at addressing an early design and a quality evaluation of data intensive applications, being our goal to help software engineers on assessing quality metrics, such as the response time of the application. We address this goal by means of a quality analysis tool-chain. At the core of the tool, we are developing a Profile that converts the Unified Modeling Language into a domain specific modeling language for quality evaluation of data intensive applications.

CCS Concepts

•**Software and its engineering** → **Unified Modeling Language (UML);** *Specification languages;* •**Information systems** → *Computing platforms;*

Keywords

Unified Modeling Language (UML), UML Profiles, Data-Intensive Applications, Model-Driven Engineering (MDE)

1. INTRODUCTION

Data intensive applications that use Big Data technologies such as Hadoop MapReduce or stream processing [18] are important in many application domains, for instance, predictive analytics or smart cities. However, a Capgemini research [4] shows that only 13% of organizations have achieved full-scale production for their Big Data implementations. The H2020 DICE project [3] aims at defining a quality-driven framework for developing data intensive applications. The DICE development approach follows the principles of model-driven development [9], being the Unified Modeling Language (UML) [17] the DICE choice for design. Therefore, providing the ability to continuously re-architect data intensive applications designs based on quality improvements such as performance or reliability.

The DICE approach allows reasoning about qualities in terms of data properties (e.g., data volumes) and data usage patterns (e.g., read rates or write rates). These data characteristics will be introduced in the UML designs by means of a new profile [11], called *DICE Profile*. This paper presents the DICE profile, its challenges, its architecture and its conceptualization.

The OMG Model-Driven Architecture [16] contemplates different abstraction layers for modeling: the *Platform Independent Model* describes the general architecture and behavior of the software while hiding the underlying platform; the *Platform Specific Model* refines the previous one by adding information related to a specific platform. Stemming from these principles, the DICE Profile will consider three abstraction layers, called DPIM, DTSM, DDSM. The *DICE Platform Independent Model* (DPIM) supports the specification of source data format, computation logic synchronisation mechanisms and quality requirements. The *DICE Platform and Technology Specific Model* (DTSM) is a refinement of the previous one and it includes some technology specific concepts, both for computation logic and data storage. Finally, the *DICE Deployment Specific Model* (DDSM) is a further specialisation of the DTSM that includes complete information of the technology in use and the application deployment.

For quality assessment, the DICE profile will rely on two already existing UML profiles, namely the standard MARTE profile (*Modeling and Analysis of Real-time and Embedded Systems*) [8] and the DAM profile (*Dependability Analysis and Modeling*) [2]. The MARTE profile will enable DICE to assess performance, while the DAM profile is its counterpart for enabling reliability assessment.

The rest of the paper is organised as follows. Section 2 overviews the approach carried out for constructing the profile. Section 3 summarizes the architecture of the DICE profile. Section 4 presents the conceptual model supporting the DPIM layer. Section 5 describes the DPIM level of the DICE profile. Section 6 illustrates the use of the DICE profile and its performance analysis capabilities. The paper finally offers a conclusion drawing next steps on the DICE profile development.

2. APPROACH OVERVIEW

For constructing a technically correct high-quality UML profile that covers the necessary concepts according to data intensive applications and corresponding Big Data technologies, several steps need to be followed. First, conceptual models for each abstraction level, i.e. DPIM, DTSM and DDSM, are needed. We have carried out this step by carefully reviewing the abstract concepts for modeling data intensive applications. Hence, we have obtained the abstractions for the DPIM level, which then conform the DICE domain model at DPIM level, Section 4 presents such model. Later, we have reviewed the different Big Data technologies addressed by DICE (e.g., Hadoop, Spark or Storm) and we have defined the abstractions of interest, consequently obtaining the DICE domain model at DTSM level, which is not reported in this paper for space reasons. Finally, the last level, DDSM, is ongoing work. An initial presentation of these models is provided in [6].

As a second step, we realized the need of introducing fresh concepts for quality assessment since the DICE DPIM domain model initially just offers concepts for describing an architectural view. Therefore, we searched in the literature for existing UML profiles that leverage quality concerns, and decided to incorporate MARTE and DAM. Our task was to select from the domain models of MARTE and DAM those metaclasses of interest for supporting our specific needs on assessment. Later, we studied how to integrate such metaclasses and the already developed DPIM domain model. Consequently, we gained a final domain model which integrates all needed features: applications abstractions at DPIM level and behavioral abstractions for quality assessment.

As a third step, the DICE profile at DPIM level was defined. Following technical advise on profile construction [7, 10], we needed to map the concepts of the DPIM domain model into proper UML profile constructors, i.e., stereotypes and tags. In particular, for the DPIM level profile we have designed: (i) the *DICE Library*, containing data intensive applications specific types; and (ii) the *DICE UML Extensions* (stereotypes and tags). The objective was to introduce a small yet comprehensive set of stereotypes for the software designer.

As a fourth and last step, we conducted a DPIM profile assessment by identifying a set of requirements based on three case studies from different application domains (fraud detection, acquisition of news from social sensors, vessel traffic management) and checking if they were met by the profile. If a requirement was not met, we went back to the previous step in order to refine it. Therefore, we followed an iterative process for the profile definition. This paper does not present the profile assessment step.

3. PROFILE ARCHITECTURE

Figure 1 offers a high-level view of the DICE profile organisation, which basically contains the *DICE Library* and the *DICE UML Extensions*. From MARTE we apply the *NFPs* and *VSL* sub-profiles (defined below), while from DAM we import the *DAM Library* which also imports the *MARTE Library*. The *DICE Library*, described in Section 5, contains basic and complex types to describe data intensive applications.

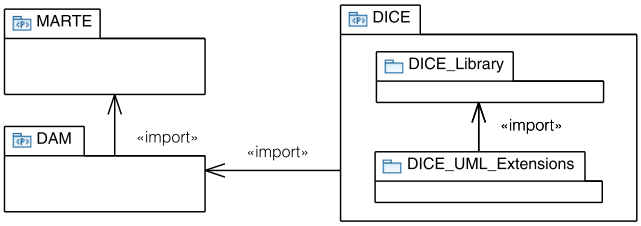

Figure 1: High-level view of the DICE profile

The *DICE UML Extensions* package, also described in Section 5, will provide the domain expert with a set of stereotypes to be applied at model specification level, i.e., the stereotypes necessary to represent the different system views using UML. We have organised the extensions according to the DICE modeling levels, as proposed in Fig. 2. Therefore, we are proposing three sub-profiles, DPIM, DTSM and DDSM. The DTSM sub-profile is organised in packages, one package per technology, all them inherit from a *DTSM-Core* that offers useful extensions for all technologies.

The MARTE profile consists of three main parts: *MARTE Foundations*, *MARTE Design Model* and *MARTE Analysis Model*. For our purposes the Analysis Model is of interest since it enables performance and schedulability analysis of a system, being the former our target. The Analysis Model consists of a *Generic Quantitative Analysis and Modelling* (GQAM) profile and its specialization, the *Performance Analysis and Modelling* (PAM) profile. In addition, MARTE owns an outstanding feature, the *Value Specification Language* (VSL), for specifying the values of constraints, properties, and stereotype attributes, particularly related to *Non Functional Properties* (NFP). Finally, MARTE also defines a library of primitive data types, a set of predefined NFP types and units of measures.

The DAM profile addresses the dependability (reliability, availability, safety and maintainability) modeling of real-time and embeded systems (RTES) with UML. It is important to note that the DAM profile is a specialization of the GQAM profile. As the DICE profile, DAM consists of a library and a set of extensions to be applied at model specification level.

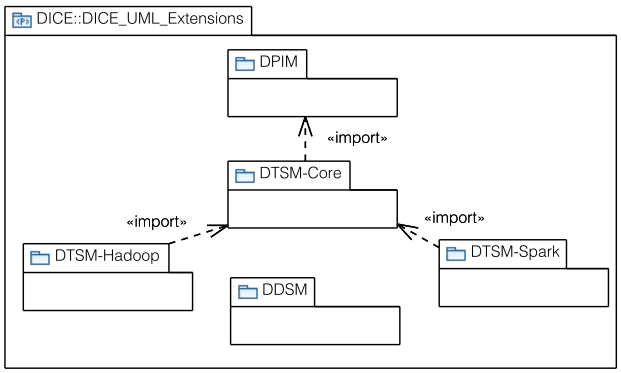

Figure 2: DICE UML Extensions package

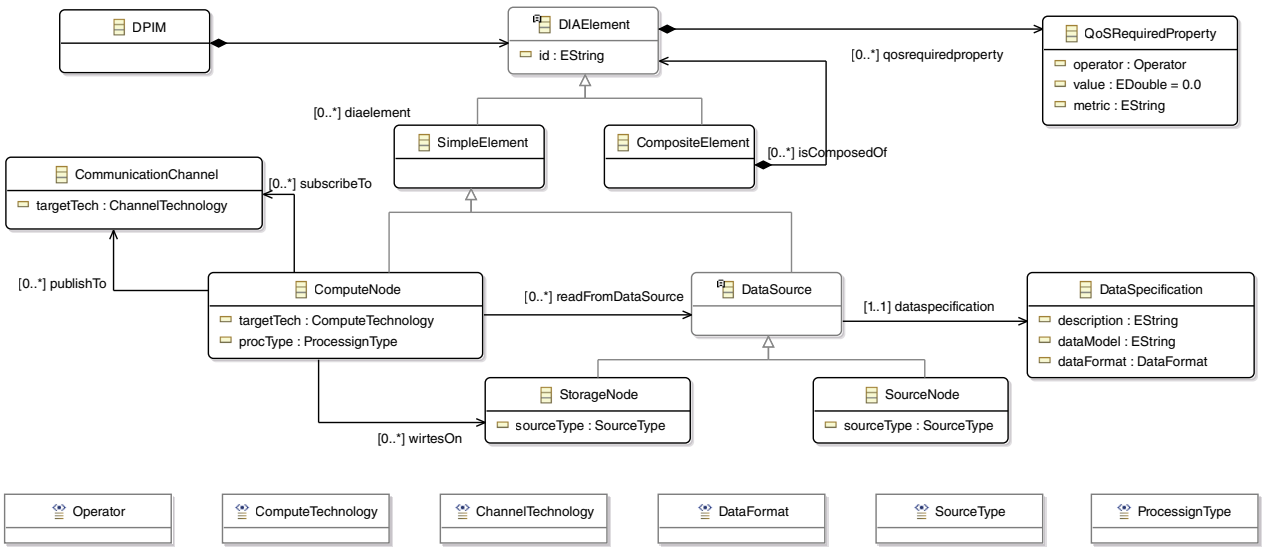

Figure 3: An excerpt of the DPIM Domain Model

4. DPIM DOMAIN MODEL

DICE identified a set of core concepts, at the DPIM layer, for designing data intensive applications, both at the architecture level and at the behavioral one. Fig. 3 depicts the fundamental concepts that constitute a data intensive applications. This model constitutes an initial asset and the first step towards defining the DICE profile. Moreover, designers may use the identified core architecture elements to quickly put together the structural view of their Big-Data application, highlighting and tackling concerns such as data flow and essential high-level processing properties (e.g., rate, properties provided and required by every component, etc.) as well as key data processing needs (e.g., batch, streaming, etc.).

The DPIM includes all concepts that are relevant for structuring a data intensive application, then allowing, to define the high level topology of the application. More in particular, the metamodel in Fig. 3 shows that elements of data intensive applications are essentially aggregates of two sets of components: *ComputeNode* and *DataSource*. The *ComputeNode* is basically responsible for carrying out computational tasks like *map*, or *reduce* in *MapReduce*. One important attribute of *ComputeNode* is *procType* that shows the processing type of Big Data, i.e., *batch processing* or *stream processing*. Moreover, the *ComputeNode* is associated to *CommunicationChannel* to represent communication channels of publication and subscription roles in the application. The *CommunicationChannel* in DPIM is a representation of *Governance and data Integration* which mainly includes the technologies responsible for transferring the data, like message broker systems. The *DataSource* itself, further specializes into *StorageNode* and *SourceNode*. The *SourceNode* role is to provide data for processing. In other words, the *SourceNode* represents the source of data which are coming into application in order to being processed. The attribute *sourceType* further specifies the characteristics of source. The ultimate goal of a Big Data application is to process the data that have high volume and velocity. So the *SourceNode*, and *ComputeNode* are in DPIM since there are the essential part of each and every data intensive application. The *SourceNode* is the entry point of data into application and the *ComputationNode* is where data would process. The third key element in the DPIM domain model is the *StorageNode*. As its name may suggest the *StorageNode* represents the element which is responsible for storing the data, either for long term or not. The concept of *StorageNode* in DPIM mainly corresponds to the *database*, in some case it could be *filesystem* also. The other salient elements in the model are *DataSpecification* and *QoSRequiredProperty*, which are annotation stubs to support the specification and subsequent evaluation of the type and format of data and the QoS of data intensive applications, respectively.

Our DPIM domain model is then useful for representing architectural views of data intensive applications. However, the capabilities for defining QoS and behavioral properties in UML designs are mainly inherited from aforementioned profiles, namely MARTE and DAM. Therefore, we need to combine the DPIM conceptual model with MARTE and DAM conceptual models for getting a profile that accounts for all design views: architectural and behavioral. Next section develops the DICE profile, at DPIM level, by leveraging the previous forces.

5. A PROFILE FOR DATA INTENSIVE APPLICATIONS AT THE DPIM LEVEL

Fig. 4 shows an excerpt of the DICE profile at DPIM level. For its construction, we have followed design principles and guidelines proposed by Selic [10] and Lagarde et al. [7]. In particular, we applied such principles to the domain model described in the previous section, enriched with MARTE and DAM abstractions for addressing behavioural and quality concerns.

For reading Fig. 4, elements annotated as «Stereotype» are DICE stereotypes. Those with white background were purposely created for the DICE profile, while the light gray

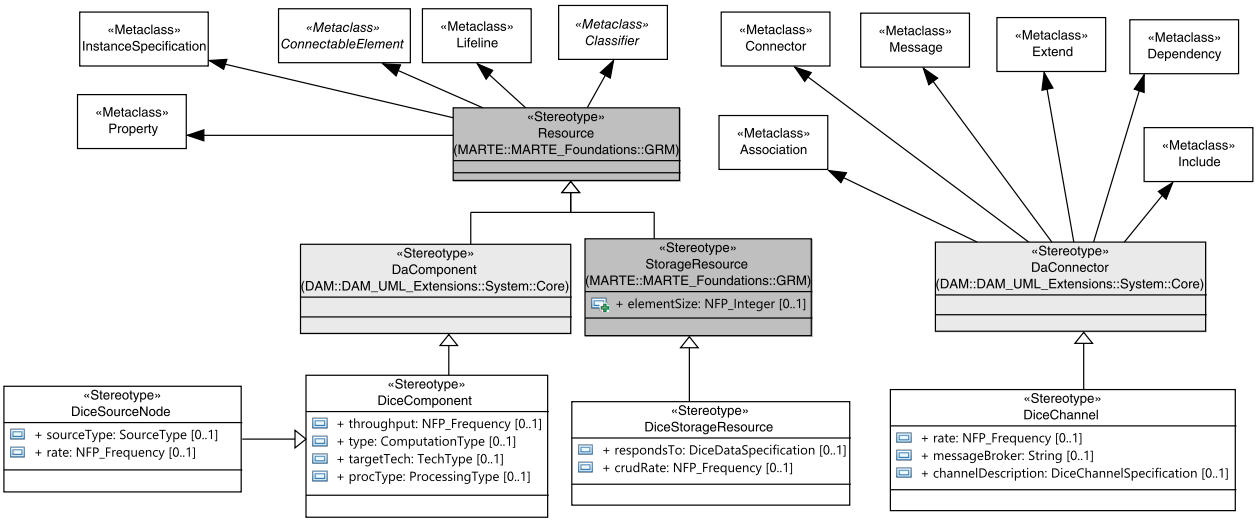

Figure 4: An excerpt of the DICE DPIM Profile

ones are inherited from DAM and the dark gray ones from MARTE. Finally, those classes annotated as «Metaclass» correspond to those standard UML metaclasses that may be extended by the DICE profile stereotypes.

In the following, we describe the rational behind some of the stereotypes to exemplify how the guidelines for creating a profile has been applied. Abstract classes, e.g., *DIAElement* and *DataSource*, do not map into stereotypes, nor the class representing the model itself, i.e., the *DPIM* domain class. On the other hand, classes of interest can be either transformed into stereotypes or datatypes. For example, *StorageNode* is transformed into *DiceStorageResource* stereotype. In this case, we also inherited from the *StorageResource* MARTE stereotype, since the resource storage concept was already present in MARTE, although not initially tailored for the domain of data intensive applications.

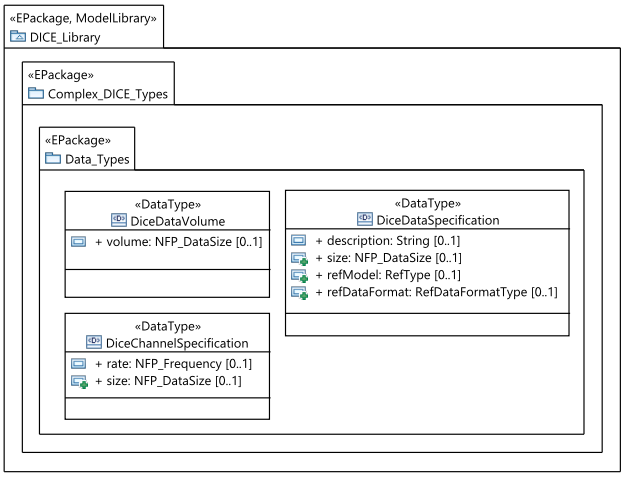

Figure 5: The *DICE_Library::Complex_DICE_Types::DataTypes* package

A case of a domain class that has been transformed into a data type is *DataSpecification*, then creating *DiceDataSpecification* in the *DICE_Library::Complex_DICE_Types::DataTypes* package (see Fig. 5). This data type will be useful for specifying tagged values related to data formats, data sizes and reference models, as in the case of the *respondsTo* tagged value of *DiceStorageResource*. In fact, following the design principles aforementioned, associations in the domain model can be transformed into tagged values. The *respondsTo* tagged value aims to represent the *dataspecification* association between *StorageNode* (inherited from *DataSource*) and *DataSpecification* in the domain model.

Finally, it is important to highlight how the *QoSRequiredProperty* domain concept has been represented in the DICE profile. As stated in Section 3, MARTE provides both the VSL language and the NFP framework to specify values of non-functional properties. Thus, QoS requirements that in the domain model are expressed using the *QoSRequiredProperty* class, are expressed in the profile using tagged values of some NFP type.

6. THE DICE PROFILE IN ACTION

This section illustrates the DICE profile capabilities for specifying data intensive applications at the DPIM level. Also, it illustrates how the annotated UML design, which represents such application and its specific data properties, provides the grounds to carry out performance assessment of the very same application. It is important to say that the DICE tool-chain implements the DICE profile and the assessment capabilities for easily guiding the developer through the different phases of the specification and quality analysis. In a further use case, the DICE tool-chain will also incorporate deployment, testing, and acquisition of feedback data through monitoring and data collection.

6.1 Using the DICE Profile for Modeling

Performance evaluation is traditionally carried out using scenarios, i.e., typical system paths of usage. Therefore, at DPIM level we need models specifying the system behavior

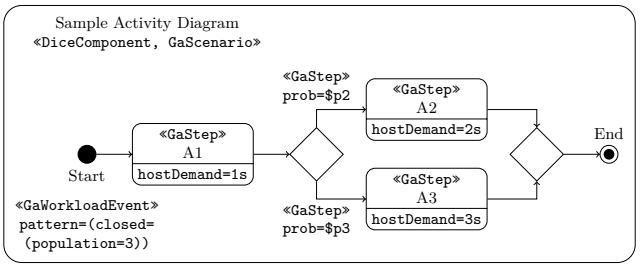

Figure 6: Sample activity diagram

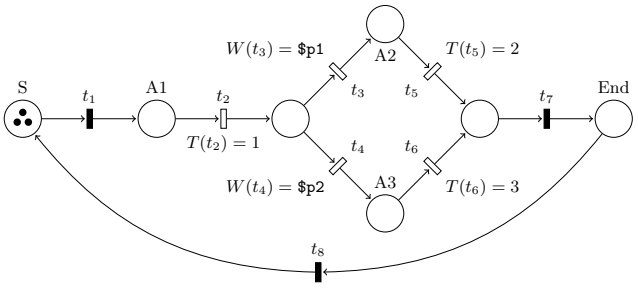

Figure 7: Generated Petri net

of the data intensive application. We can do this by modeling directed graphs that express dependencies between computation and/or data, for which the UML AD is suitable. At this DPIM level, the modelling is done independently of the target platform and technology, and, for this reason, the DICE profile heavily relies on the DAM and MARTE profiles.

As a simple example we model the internal behavior of a single component of a data intensive application by exploiting a UML activity diagram (AD), see Fig. 6. The diagram consists of an initial node (*Start*), a final node (*End*), and three activities (*A1*, *A2* and *A3*). Two of the activities (*A2* and *A3*) are executed alternatively after *A1*.

The example illustrates the use of four DICE stereotypes: *DiceComponent*, *GaScenario*, *GaWorkloadEvent* and *GaStep*. The *DiceComponent* stereotype specifies that the diagram represents a component behavior. The *GaScenario* stereotype, imported from MARTE, specifies that the diagram represents a *generic quantitive analysis* scenario and it enables the use of parameters for the performance analysis of the scenario, like the variables `$p2` and `$p3`, in fact this is a feature provided by the MARTE's VSL standard language. The *GaWorkloadEvent* stereotype is used to specify the system workload, it is applied to the initial node and here it determines that the system will receive a closed load with an initial population of 3 jobs. The *GaStep* stereotype is used to specify the service demand requirements for activities *A1*, *A2* and *A3*, they will require 1, 2 and 3 seconds of host demand in average, respectively. Finally, the application of *GaStep* to the flows between decision nodes, *A2* and *A3*, is useful for specifying the system routing rates. In this case, it indicates the probabilities of executing either *A2* or *A3*, which are `$p2` or `$p3` respectively.

6.2 Exploiting the DICE Profile for Quality Assessment

While the annotated UML model is useful for the engineer to specify both the workflow of the application and its data characteristics, it is not suitable for an assessment of its performance requirements. Thus, a strategy is to transform the annotated UML model into a suitable formalism, e.g., stochastic Petri nets [1], that allows this kind of analysis. Stochastic Petri nets are a graphical formalism for the modeling, analysis and evaluation of concurrent systems. Fig. 7 depicts the Petri net automatically obtained by the DICE tool-chain from the AD in Fig. 6. In the Petri net, rounded nodes represent *places*, while the rectangular ones represent *transitions*. Black transitions correspond to *immediate transitions*, i.e., those with no firing delay, while white transitions correspond to *timed transitions*, i.e., those with an exponentially distributed firing time.

As for the DICE annotations in Fig. 6, they are treated as follows: (i) the *closed pattern* of the system's workload defined for the *Start* initial node implies that the Petri net should be evaluated under a steady-state assumption, then we need to "close" the Petri net through transition, t_8; (ii) the workload *population* determines the *initial marking* of the Petri net, in this case associated to the place S; (iii) the *hostDemand* associated to each activity determines the *timing* – $T(t_i)$ – of its corresponding transitions, t_2, t_3 and t_4; and (iv) the *probability* associated to the flows between the decision node and *A2* and *A3* determines the *weight* – $W(t_i)$ – of transitions t_3 and t_4. Using this Petri net, an analysis tool, such as GreatSPN [5], may provide accurate results, e.g., response time or throughput, to reasoning about the performance characteristics of the system modeled in UML. We are currently carrying out work to validate the results obtained with our Petri net.

7. CONCLUSIONS AND FUTURE WORK

DICE is a project that, following the model-driven engineering paradigm, aims to define a quality-driven framework for developing data intensive applications leveraging *Big Data* technologies. A key asset of DICE is the so called DICE profile, which offers the ability to design such application using UML and a set of additional stereotypes to characterize specific data intensive features. DICE-profiled models are the cornerstone of the DICE framework, since they are exploited by the DICE tool-chain to guide developers through the whole application lifecycle (e.g., development, quality analysis, deployment, testing, monitoring, etc.).

In this paper, we have presented the DICE profile, its foundations and its architecture; and we have outlined how DICE-profiled models can be exploited in further software development phases. The DICE profile has deep roots on other two profiles, namely MARTE and DAM, and has been structured to fit different abstraction levels (DPIM, DTSM, DDSM) similarly to the MDA standard. For its construction, we have followed a guided process as recommended by state of the art works [7, 10] for building quality profiles.

The DICE profile has been implemented and integrated within *Papyrus UML* [14], a UML modeling tool based on the well-known Eclipse [15] integrated development environment. The DICE domain models and the DICE profile are publicly available under an open source license in their corresponding repositories, namely the *DICE-Models Repository* [12] and the *DICE-Profiles Repository* [13].

In the near future, we will focus on the completion and validation of the DICE profile. Specifically, the DTSM level, that still lacks of some technology-specific packages (such as those for *Storm* and *Oryx*), while the DDSM layer still needs to be addressed.

8. ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation framework programme under grant agreement No. 644869 (DICE), the Spanish Government (*Ministerio de Economía y Competitividad*) under project No. TIN2013-46238-C4-1-R and The Aragonese Goverment Ref. T27 – DIStributed COmputation (DISCO) research group.

9. REFERENCES

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modeling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1994.

[2] S. Bernardi, J. Merseguer, and D. Petriu. *Model-driven Dependability Assessment of Software Systems*. Springer, 2013.

[3] G. Casale et al. DICE: Quality-driven Development of Data-intensive Cloud Applications. In *Proceedings of the Seventh International Workshop on Modeling in Software Engineering*, pages 78–83, NJ, USA, 2015. IEEE Press.

[4] M. Colas, I. Finck, J. Buvat, R. Nambiar, and R. R. Singh. Cracking the data conundrum: How successful companies make big data operational. Technical report, Capgemini consulting, 2014. URL: `https://www.capgemini-consulting.com/cracking-the-data-conundrum`.

[5] Dipartamento di informatica, Università di Torino. GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets, Dec., 2015. URL: `www.di.unito.it/~greatspn/index.html`.

[6] M. Guerriero, S. Tajfar, D. Tamburri, and E. Di Nitto. Towards a model-driven design tool for Big Data architectures. In *Proceedings of the 2nd International Workshop on BIG Data Software Engineering*, 2016.

[7] F. Lagarde, H. Espinoza, F. Terrier, and S. Gérard. Improving UML profile design practices by leveraging conceptual domain models. In *22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), Atlanta (USA)*, pages 445–448. ACM, November 2007.

[8] UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems, June 2011. Version 1.1, OMG document: formal/2011-06-02.

[9] B. Selic. The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25, 2003.

[10] B. Selic. A Systematic Approach to Domain-Specific Language Design Using UML. In *Tenth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2007), 7-9 May 2007, Santorini Island, Greece*, pages 2–9. IEEE Computer Society, 2007.

[11] The DICE Consortium. Design and quality abstractions - Initial version. Technical report, European Union's Horizon 2020 research and innovation programme, 2016. URL: `http://wp.doc.ic.ac.uk/dice-h2020/wp-content/uploads/sites/75/2016/02/D2.1_Design-and-quality-abstractions-Initial-version.pdf`.

[12] The DICE Consortium. DICE Models Repository, Apr., 2016. URL: `https://github.com/dice-project/DICE-Models`.

[13] The DICE Consortium. DICE Profiles Repository, Apr., 2016. URL: `https://github.com/dice-project/DICE-Profiles`.

[14] The Eclipse Foundation. Papyrus, 2016. URL: `https://eclipse.org/papyrus/`.

[15] The Eclipse Foundation. Website, 2016. URL: `http://www.eclipse.org/`.

[16] The Object Management Group (OMG). Model-Driven Architecture Specification and Standardisation. Technical report, . URL: `http://www.omg.org/mda/`.

[17] Unified Modeling Language: Infrastructure, 2011. Version 2.4.1, OMG document: formal/2011-08-05.

[18] P. Zikopoulos and C. Eaton. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1st edition, 2011.