Summary: A domain specific-model and DevOps approach for big data analytics architectures

Camilo Castellanos[a], Carlos A. Varela[b] and Dario Correal[a]

[a]System Engineering and Computing Department, Universidad de Los Andes, Bogota, Colombia
[b]Department of Computer Science Rensselaer Polytechnic Institute, Troy, NY, USA

Abstract
Big data analytics (BDA) applications use machine learning algorithms to extract insights from large, fast, and heterogeneous data sources. Software engineering challenges for BDA applications include ensuring performance levels of data-driven algorithms even in the presence of large data volume, velocity, and variety (3Vs). BDA software complexity frequently leads to delayed deployments and challenging performance assessments. This paper[1] proposes a domain-specific modeling (DSM) and DevOps practices to design, deploy, and monitor performance metrics for BDA architectures. Our proposal includes a design process and framework to define architectural inputs, functional, and deployment views via integrated high-level abstractions to monitor the achievement of quality scenarios. We evaluate our approach with four use cases from different domains. Our results show shorter deployment and monitoring times and a higher gain factor per iteration than similar approaches.

Keywords
Software architecture, Big data analytics deployment, DevOps, domain specific modeling, quality scenarios, performance monitoring

1. Introduction

Big data analytics (BDA) applications use Machine Learning (ML) algorithms to extract valuable insights from large, fast, and heterogeneous data. These BDA applications require complex software design, development, and deployment to deal with big data characteristics: volume, variety, and velocity (*3Vs*), to maintain expected performance levels. However, the complexity involved in BDA application development frequently leads to delayed deployments [2][3] and hinders performance monitoring (e.g., throughput or latency) [4]. Besides, heavy workloads imply batch processing over big data, demand high scalability and fault tolerance for achieving deadlines. A software architecture specifies the system's structures and their relationships to achieve expected quality properties. The BDA solutions development exhibits a high cost and error-prone transition between development and production environments given the lack of techniques and tools to enable articulation and integration [2, 5]. Despite the growing interest in big data adoption, real deployments are still scarce ("Deployment Gap" phenomenon) [6].

[1]Use the original publication when citing this work[1]

ECSA'21: European Conference on Software Architecture, September 13–17, 2021, Växjö, Sweden
cc.castellanos87@uniandes.edu.co (C. Castellanos); cvarela@cs.rpi.edu (C. A. Varela); dcorreal@uniandes.edu.co (D. Correal)

We propose ACCORDANT (An exeCutable arChitecture mOdel foR big Data ANalyTics), a DevOps and DSM approach to develop, deploy, and monitor BDA solutions bridging the gap between analytics and IT domains. This paper highlights the contributions of this proposal presented in [7] [1]. ACCORDANT allows designing BDA applications driven by quality scenarios (QS), functional, and deployment views. A QS specifies a quality attribute requirement for a software artifact to support the design and quality assessment. The functional view defines the architectural elements that deliver the application's functionality. The deployment view describes how software is assigned to technology infrastructure. Our deployment strategy incorporates containers to promote portability and continuous deployment.

ACCORDANT is validated with four use cases from different domains by designing functional and deployment models and assessing performance QS. This validation aims to reduce the time of design, deployment, and QS monitoring of BDA solutions. In summary, the contributions of this paper are: i) A DSM framework to formalize and accelerate the development and deployment of BDA solutions by specifying functional iteratively and deployment views aligned to QS. ii) Three integrated DSLs to specify architectural inputs, component-connector models, and deployments. iii) A containerization approach to promote automation delivery and performance metrics monitoring aligned to QS. iv) The evaluation of this proposal with four use cases from diverse domains and using different deployment strategies and QS.

The rest of this paper is organized as follows. Section 2 presents our methodology and proposal overview. Section 3 presents evaluation and discusses the results. Finally, Section 4 summarizes the conclusions.

2. ACCORDANT: A DevOps and Domain Specific Model Approach for BDA

Our proposal comprises a design and deployment method and a DSM framework. This proposal includes architectural inputs, containerization, and serverless deployments. Fig. 1 depicts the ACCORDANT's process. The steps performed using the ACCORDANT modeling framework are framed in solid lines, while the steps made with external tools are represented by dotted lines. The ACCORDANT process is iterative and composed of seven steps: Definition of 1.1) business goals and 1.2) QS. 2) The data scientist develops data transformations and analytics models and exports them as PMML files (Predictive Model Markup Language). 3) The IT architect designs the software architecture using ACCORDANT Metamodel in *Functional Viewpoint* (FV) and *Deployment Viewpoint* (DV). FV model makes use of PMML models to specify the software behavior. 4) FV and DV models are interweaved to obtain an integrated model. 5) Code generation is performed from integrated models. 6) The generated code is executed to provision infrastructure and install the software. 7) QS are monitored in operation to be validated.

Architectural Inputs: The architecture design is driven by predefined quality scenarios (QS) according to architecture design methods such as Attribute-Driven Design (ADD) [8]. These QS are achieved through design decisions compiled in well-known catalogs of architectural patterns and tactics. Both QS and tactics are inputs of the architecture design. Therefore we include these initial building blocks in the ACCORDANT metamodel along with other concepts defined in ADD. The main input building blocks are grouped by the architectural input package

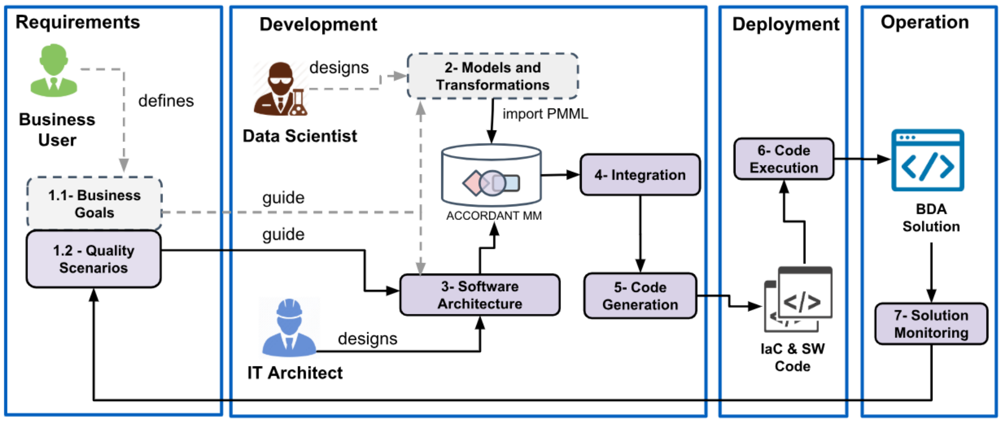

Figure 1: ACCORDANT Process overview

(*InputPackage*) which contains the elements required to start the architectural design: Quality Scenario (*QScenario*), Analyzed QS (*AnalyzedQS*), *SentivityPoint* and *Tactic*.

The **functional viewpoint (FV)** specifies analytics pipelines, including ingestion, preparation, analysis, and exporting components. *FV* describes functional requirements of the BDA solution, and the constructs are described in a technology-neutral way. FV is expressed in a component-connector structure. Sensitivity points as architectural inputs can be associated to components and connectors to represent where architectural decisions regarding the QS. Component metaclasses are specialized in *Ingestor*, *Transformer*, *Estimators* and *Sink*. *Estimator* and *Transformer* are software component realizations of PMML models and data transformer respectively, and the PMML file defines their behavior. A *Component* exposes required and provided *Port*. *Connectors* metaclasses transfer data or control flow among components through their *Roles*. The *Connector* types are defined based on the classification proposed by Taylor et al. in [9]: *Procedure Call*, *Event*, *Stream*, *Adaptor*, *Distributor*, and *Arbitrator*.

The **deployment viewpoint (DV)** integrates DevOps practices encompassing containerization and infrastructure as code (IaC). The DV specifies how software artifacts (components and connectors) are deployed on a set of computating nodes. DV metamodel comprises *Pod*, *ExposedPort*, and *Deployment* metaclasses to operationalize BDA applications in a specific technology. It is noteworthy that a *FV* model can be deployed in different *DV* models either to use a different strategy or to test the fulfillment of predefined QScenarios. DV contains *Devices*, *Services*, *Deployments*, serverless environments (*ServerlessEnv*), and *Artifacts*.

Once PMML, FV, and DV models are designed and integrated, code generation takes place by means of model-to-text transformations. Code generation is twofold: software and infrastructure (IaC) code. In the last step, the performance metrics of the BDA application are gathered to evaluate them against QS. This process can take several iterations, and this is the whole cycle that we expect to accelerate and using ACCORDANT.

3. Evaluation

Our experimentation aims to compare development and deployment time per iteration using accordant and other two frameworks: FastScore and SpringXD. We chose these frameworks because they are the closest to our approach regarding the related work detailed in [1], and they support portable analytics models. We validated our proposal through four use cases: UC1) Transport delay prediction, UC2) Near mid-air collision detection, UC3) Near mid-air collision risk analysis, and UC4) El Nino/Southern Oscillation cycles.

To compare ACCORDANT, SpringXD, and FastScore, we measured the time invested in development and deployment phases per use case. These phases are performed iteratively since improvements and refinements are made in each iteration until QS is achieved. Therefore, we measure the time invested in each iteration. Also, we calculated the gain factor $GF(uc, f)$, as a metric to estimate the cumulative average of time reduction ratio for a use case *uc*, using framework *f*. We define the gain factor as a form to measure the incremental improvement of using high-level abstractions to evolve an application.

To design, develop, and deploy the four use cases, we followed the ACCORDANT process detailed previously in Figure 1. The development time using ACCORDANT is higher (between 23% and 47%) compared to SpringXD and Fastscore, but the deployment time is lower (between 50% and 81%). The higher development time can be explained by the effort required to specify ACCORDANT models. The highest time differences arise from UC2, which is the most complex pipeline. The high-level reuse of previous architectural decisions reduced development time, as shown by the decrease between use cases and the growing gain factor among iterations. These results suggest that ACCORDANT is most suitable for applications with multiple iterations, or in subsequent applications, reusing architectural models reduces development times.

ACCORDANT's gain factor was higher for all use cases in the development phase, which suggests that the high-level abstractions promote the reduction of development time among consecutive iterations. The highest gain factor was 0.46 in the UC3, reducing in 46% the development time between iterations. The greatest gain factor difference over the other approaches was 0.13 in the UC3. Regarding the deployment gain factor (Fig. 2b), ACCORDANT also exhibited the highest gain factor, on a higher proportion, up to 0.75 in UC4. Each deployment iteration reduces the time by 75% compared to the previous one. The gain factor in the deployment phase is greater in ACCORDANT because the IaC generation is not offered in other approaches.

4. Conclusions

The results have shown that ACCORDANT can accelerate iterative development and deployment phases. The greatest time reduction was reported in the deployment phase, achieving up to 81% compared to other approaches. In contrast, the development times offered by ACCORDANT were greater. Despite the longer development time, deployment time is significantly reduced by using QS, FV, and DV alignment. ACCORDANT's gain factor was higher, which implies a higher reduction time in each iteration. In contrast, some limitations have emerged from experimentation. The development phase is slower than the other approaches due to the current ACCORDANT's prototype requires additional manual coding. ACCORDANT requires detailed

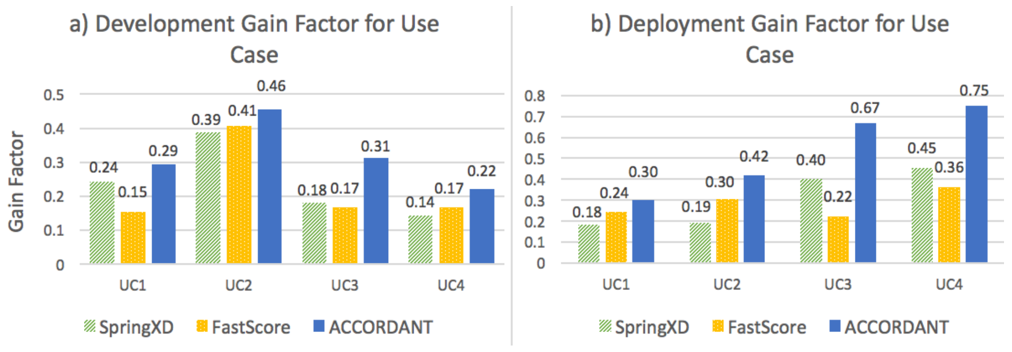

Figure 2: Gain Factor for Use Case

design initially, which is then rewarded in consecutive iterations. So, ACCORDANT is most suitable for applications with multiple iterations. Finally, our approach exploits the reuse of architectural decisions and models. Hence, first-time or one-time applications may not be benefited from our proposal.

References

[1] C. Castellanos, C. A. Varela, D. Correal, ACCORDANT: A domain specific-model and DevOps approach for big data analytics architectures, Journal of Systems and Software 172 (2021) 110869. doi:https://doi.org/10.1016/j.jss.2020.110869.

[2] H.-M. Chen, R. Kazman, S. Haziyev, Agile Big Data Analytics for Web-Based Systems: An Architecture-Centric Approach, IEEE Transactions on Big Data 2 (2016) 234–248. doi:10.1109/TBDATA.2016.2564982.

[3] C. Castellanos, B. Pérez, C. A. Varela, M. d. P. Villamil, D. Correal, A survey on big data analytics solutions deployment, in: T. Bures, L. Duchien, P. Inverardi (Eds.), Software Architecture, Springer International Publishing, Cham, 2019, pp. 195–210.

[4] R. Ranjan, Streaming big data processing in datacenter clouds, IEEE Cloud Computing 1 (2014) 78–83. doi:10.1109/MCC.2014.22.

[5] D. Wegener, S. Rüping, On reusing data mining in business processes-a pattern-based approach, in: International Conference on Business Process Management, Springer, 2010, pp. 264–276.

[6] H.-M. Chen, R. Schütz, R. Kazman, F. Matthes, How Lufthansa Capitalized on Big Data for Business Model Renovation, MIS Quarterly Executive 1615 (2017) 299–320.

[7] C. Castellanos, D. Correal, J.-D. Rodriguez, Executing Architectural Models for Big Data Analytics, in: C. E. Cuesta, D. Garlan, J. Pérez (Eds.), Software Architecture, Springer International Publishing, Cham, 2018, pp. 364–371.

[8] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, B. Wood, Attribute-driven design (ADD), version 2.0, Technical Report, Carnegie Mellon University-SEI, 2006.

[9] R. N. Taylor, N. Medvidovic, D. E. M., Software Architecture: Foundations, theory and practice, John Wiley and Sons, Inc, 2010.