Big Picture of Big Data Software Engineering

With example research challenges

Nazim H. Madhavji
Dept. of Computer Science
University of Western Ontario
London, Canada
madhavji@gmail.com

Andriy Miranskyy
Dept. of Computer Science
Ryerson University
Toronto, Canada
avm@ryerson.ca

Kostas Kontogiannis
Dept. of Electrical & Computer Eng.
National Technical University of Athens
Athens, Greece
kkontog@softlab.ntua.gr

***Abstract*— In the rapidly growing field of Big Data, we note that a disproportionately larger amount of effort is being invested in infrastructure development and data analytics in comparison to applications software development – approximately a 80:20 ratio. This prompted us to create a context model of Big Data Software Engineering (BDSE) containing various elements — such as development practice, Big Data systems, corporate decision-making, and research — and their relationships. The model puts into perspective where various types of stakeholders fit in. From the research perspective, we describe example challenges in BDSE, specifically requirements, architectures, and testing and maintenance.**

***Index Terms*—Big Data, Applications, Software Engineering, Context model, Research challenges.**

I. Introduction

The Big Data technology and services market is expected to grow at a compound annual rate of 27% from $9.8 billion in 2012 to $32.4 billion in 2017, leading to multi-fold increase in the amount of technical data generated [1]. In this hurricane of data, the technical and business communities are getting sucked into the eye of data analytics and Big Data infrastructures [1] blurring the inevitable need to develop end-user applications utilising Big Data in a wide varieties of application domains – the so-called Big Data Software Engineering (BDSE). This bias is evident in the frequency count of vacant Big Data positions in the first one hundred job entries on Google search: 76% in *infrastructures* and *analytics* against only 22% in *application* software (and approx. 10% other)[1] – let us say a 80:20 ratio.

There is no theoretical reason to support this bias given that BDSE systems can also yield new insights that can possibly aid in furthering institutional or corporate goals and objectives. This lopsided situation, in part, has prompted us to hold our position that there is a need to depict a Big Data environment as a context model (the so-called Big Picture of BDSE), containing elements (such as application development) and relationships with data analytics among others, so as to better understand the dynamics involved.

Such an understanding is anticipated to help in at least two ways: (a) in organisational structuring with appropriate agent roles, processes, and relationships among the interacting elements in the Big Data environment, and (b) in precipitating research in targeted areas of the Big Data environment. In this paper, we focus on (b) because this is a workshop paper and the field is in its infancy.

The context model is described in the next section. We then pick example areas of the model to highlight research challenges (in requirements, architecture, and testing) in the development of Big Data applications.

II. A Context Model of BDSE

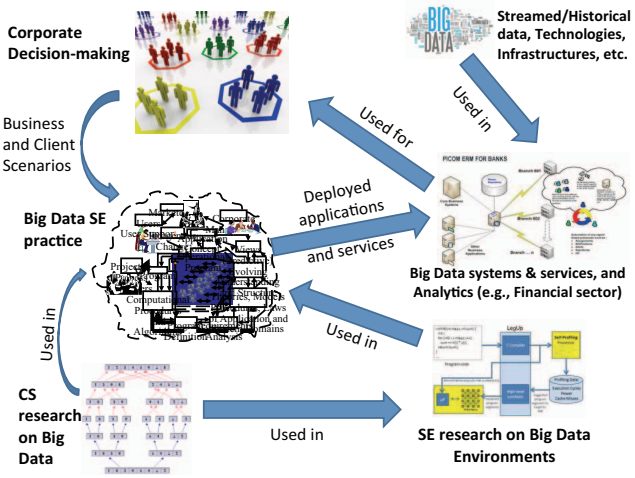

Figure 1 – A context model of Big Data Software Engineering

Figure 1 depicts a context model for BDSE. Let us start with the top right quadrant of the figure. Historical data or streamed data are used in analytics to address business objectives, the results of which are used for business decision-making. We assume that business objectives are relayed to the data analysts as requirements (not shown in the figure). Also, to carry out analytics, tools (such as Apache Mahout [2] and 0xdata H2O [3] data mining and machine learning frameworks) and infrastructure (such as one of the Amazon Web Services [4] or IBM Netezza appliances [5]) are utilised. This is where significant current-day excitement is in industry, judging by job demands.

[1] Total exceeds 100% due to overlapping areas of positions.

978-1-4673-7025-7/15 $31.00 © 2015 IEEE
DOI 10.1109/BIGDSE.2015.10

The centerpiece of the proposed model is the centre-left "blob" where BDSE practice takes place to develop, maintain and evolve end-user Big Data applications. Be it agile, iterative, or evolutionary process models, this is where requirements are discovered, analysed and prioritised, application systems are architected, designed, developed, tested and integrated into suitable products, systems, and services.

While methodologically this is all familiar from traditional software engineering (SE) (see SWEBOK [6]), new challenges emanate from having to deal with the characteristics of Big Data in specific SE tasks: e.g., data needs to be processed in real-time else incoming data could become lost and/or obsolete (*velocity*); mountain-ranges of historical data may exist (*volume*) and the end-user application system needs to be scalable to be able to cope with increasing size; data can be structured or unstructured (*variety*) and needs to be associated or aggregated in innovative ways to create new value for business; historical or streams of data may need to be cleaned up prior to analysing it (*veracity*); etc.

Given such a data centred environment, how do you specify requirements for an end-user system that utilise such data? How do you architect a system, or which reference architecture to select, for processing a deluge of wild data? Likewise for detailed design models and simulation, coding, testing, integration, etc. The software lifecycle is plagued with Big Data concerns at all stages of development, maintenance, and evolution of a Big Data application system. This presents new research challenges and, likewise, opportunities.

The model in Figure 1 shows that developed systems and services are deployed in the context of use (e.g., a financial sector), where a never-ending amount of data – historical or live -- is processed to yield value to the stakeholders (e.g., for corporate decision-making). Yet, wisdom tells us that the requirements for building end-user applications must come, in large measure, from stakeholder input (e.g., business scenarios); otherwise, the resultant systems (a) may not have desirable usability qualities and (b) when used may not yield desirable business or stakeholder value. We assume that customer and end-users, not explicitly shown, are represented in the BDSE "blob" (e.g., during requirements gathering, architecting, acceptance testing, etc.).

Of course, *practice* without *research* may soon run into trouble. This is why the big picture of BDSE in Figure 1 includes, for illustrative purposes, two core research areas (computer science (CS) and SE research) where experimentation on faster, better and cheaper Big Data concepts, models, methods, techniques, tools and processes takes place. The fruits of research are put into BDSE practice so as to create new or improved Big Data end-user applications, within budget and on time.

The reader is cautioned that this model is not to be taken as a dogma. It is meant to illustrate a theoretical context for BDSE and data analytics because neither of them exists in vacuum. There are immediate needs to operate on historical data (e.g., patient records) for which data analytics seem to offer a solution. Yet, there are possible new opportunities for which innovative end-user applications can transform the way we conduct businesses and operations. For this, new applications need to be developed, maintained and evolved -- data analytics alone may not suffice. Thus, BDSE and data analytics complement each other.

Furthermore, we consider that there are important analogies between software engineering for Big Data applications and *system* engineering issues. More specifically, software engineering of Big Data systems amalgamates with system engineering, as software on the system is often tailored to the hardware. This is especially pronounced with Big Data appliances (e.g., those offered by IBM Netezza, Oracle, and Teradata). For example, IBM Netezza appliances contain specialized hardware accelerators (S-Blades [5]) for data filtering and aggregation. Thus, any development of data processing software needs to take such accelerators into account. Another example is implementation of Deep Learning algorithms using cluster of GPU computation nodes; again software under development should take hardware constraints into account. Even if one builds a solution using infrastructure as a service (IaaS) offering, such as Amazon or Google web services, one has to take into account the topology of virtual computation nodes, limitations of available databases (such as Amazon Redshift or Google BigQuery), etc. Such services typically offer specialized API, which, on one hand ensures optimal performance of the system and, on the other, "locks" clients in a given business offering, making transition to a different vendor challenging.

III. ILLUSTRATIVE RESEARCH CHALLENGES

With reference to Figure 1, due to space limitation, we sketch selected research challenges that exist in BDSE. The purpose here is only one of raising awareness. While this paper does not offer solutions, we anticipate that recognition of challenges will promulgate solutions building. Section A focuses on requirements engineering (RE); Section B on reference architectures; and Section C on testing and maintenance.

A. Requirement Engineering

(1) *Big Data characteristics:* Engineering requirements for an application involves such key activities as: creating and understanding models of the application domain, eliciting use-case scenarios and requirements from stakeholders and from other sources, developing functional and behavioural models, analysis, prioritisation, and validation [7]. The characteristics of the real-world elements (e.g., dimensions, weight, cost, transfer rates, and loudness) just to name a few, that are relevant for the application being developed, should be representable in software along with the logic (operations) acting upon them.

For example, when eliciting behavioural scenarios of desirable system responses, the characteristics of Big Data elements (such as volume, velocity, variety, etc.) *must* be representable in requirements *notations* so that solution design can be created to meet the specifications. This is analogous to other relevant domain requirements, such as communication,

safety, privacy, performance, reliability, ease of use, personalisation, etc. [8].

Currently, however, RE for Big Data applications is an emerging area. A clearer understanding is needed, separating requirements for infrastructures, analytic tools and techniques, and end-user applications. Confusion abounds in job descriptions on the web. While the former two areas are becoming entrenched in the technology domain, relatively little is known about RE for end-user applications using Big Data.

For example, how should a requirements analyst exploit video footage of a recently arrived customer in a department store, in conjunction with possible previous experiences in the store, so as to offer, in real-time, personalised and time-sensitive discounts to *this* customer, at specific points of displays along the route where the customer would potentially be shopping for items? Here, we can clearly see that the software system needs to address all the popular Big Data characteristics – the so-called "4Vs".

(2) *Multi-Peak processes:* New underlying technologies (such as granular computing, cloud computing, biological computing and quantum computing [9]) have a variety of capabilities for information and knowledge processing (e.g., data abstraction, distribution, storage, high processing power, etc.), facilitating technologists to create specific libraries and frameworks for operating on Big Data elements (e.g., comparing multiple live videos concurrently), thereby, enabling requirements engineers to elicit (specify, analyse, prioritise, validate, negotiate, etc.) requirements more directly compatible with the end-user Big Data application domains.

For example, instead of specifying basic image requirements and detailed image-processing requirements, analysts can focus on more abstract or real-world requirements (e.g., "*if person walking into the store has been here before...*"). Accordingly, the solution can focus on the speed with which streamed images can be compared and be linked to behavioural data associated with matching images in the image-base. Note that this complements component-based requirements engineering (CARE) [10].

Such abstraction encourages or necessitates closer, agile cooperation between the end-user, requirements analysts, architects, technologists, testers and other stakeholders in decision-making at the front-end of a Big Data applications development project – the so-called *Multi-Peak processes* crossing traditional process boundaries. Clearly, this calls for recasting monolithic RE processes in traditional SE as an integrated set of agile processes (end-user, RE, SA, Testing, etc.) that exploit the underlying techniques and technologies for rapid, more accurate, and lower cost development.

B. Architectures

While there are robust infrastructures for Big Data analytics (e.g., cloud deployments, map-reduce frameworks), there is limited work on the investigation of reference architectures and patterns for Big Data analytics applications and on how these reference architectures can be reified into concrete architectures and deployments. Large software organisations such as IBM, Oracle, and Microsoft, institutes such as NIST have started to investigate this issue. However, there is still work to be done on *how these reference architectures can be mapped onto existing technologies, frameworks, and tools* to yield optimal application deployments.

Another issue that warrants investigation and poses research challenges is *techniques to assess the impact of architectural design decisions on functional and more importantly, non-functional requirements* [11] of Big Data analytics applications. Literature has a number of methods for assessing architectures, e.g., SAAM and ATAM by SEI. However, work is limited on techniques for assessing architectural design decisions in virtualised, distributed Big Data analytics environments.

C. Testing and Maintenance: Representative Test System

Big Data Systems (BDS) are complex solutions with many dynamic components, such as distributed computation nodes, networks, databases, middleware, and business intelligence layers. Any component can fail while interacting with others, resulting in crashing failure or operational quality degradation (e.g., performance, reliability, security) [12]. They are thus challenging to develop, test, and maintain.

To validate changes (induced by test or maintenance activities) to a production BDS, ideally, the test system needs to be identical to the production system. Unfortunately due to the scale of BDS, this is physically and economically unfeasible.

This raises a research question: *how to build a representative test system using hardware and software resources that account for a fraction of the BDS production system*? In order to address the question, we need to dissect the problem.

(1) *Scaling down resources:* The 1st sub-question (related to volume and velocity) is *how to scale down the resources to produce realistic BDS test system*? It is obvious that we need to reduce the amount of storage and computational resources. But how does this reduction affect software configuration? For example, if the test system has smaller amount of memory than production system, how much memory do we need to allocate to database engine installed on the test system? Certain practical hacks do exist: e.g., one can scale down database memory consumption while taking into account the ratio between production and test system size [13]. However, no general theory of scaling down software configurations (while retaining its representativeness to the production system) exists.

(2) *Scaling data representatively:* The 2nd sub-question (dealing with variety) is *how to scale data representatively*? Data representation is about ensuring that the data used for testing is representative of the data processed in the production BDS. You want to ensure that data representation in test maintains the same relationship and meaning as in production, even when the total amount of data is less in test than production. There exist tooling, such as IBM InfoSphere Optim Test Data Management [14], that can sample/extract a subset of structured data from a relational database while maintaining data integrity. There exist tools for generating representative synthetic structured data (see [15] for review).

However, no work seems to have been done on selecting or generating a representative subset of unstructured data (e.g., stored in NoSQL or NewSQL database engine). For example, if an analytic system is designed to extract topics from private messages posted via social network, can we synthesize representative messages, while still being able to obtain meaningful output from the analytic system?

(3) *Workload capture-replay:* Finally, the 3rd sub-question (associated with volume and velocity) is *how to capture workload on a production BDS and replay it on a test BDS*? In particular, in the database area, there exist tools (such as IBM InfoSphere Optim Workload Replay [16] and Oracle Database Replay [17]) for capturing workloads on production system and replaying them on a test system to ensure accurate system testing. However, they focus on relational databases. There is now a need for tools that can capture intensive production workloads of BDS and replaying them on the test system in the presence of data obfuscation. Likewise, tools for other components of BDS, such as business intelligence layer are required. There is also a need for a general strategy for scaling down the workload. If, say, our test system is ten times smaller than the production system, does this mean that we should reduce (in terms of: number of concurrent connections, operations per unit of time, etc.) our workload by ten times too?

IV. DISCUSSION AND SUMMARY

Big Data is here to stay [1]. However, analysis shows a bias in favour of infrastructure and analytical tool development than in applications development for Big Data systems. Judging by the history of the general software industry, it would appear that, as the field of Big Data matures, the (approximate) 80:20 ratio described in the introduction section, will likely shift in favour of Big Data software engineering (BDSE) as applications development business proliferates and as fewer players solidify their market hold in infrastructure and tool support.

This bias and the hurricane of activities in the area of Big Data technologies and analytics, unfortunately, has created a mental gap in the understanding of how various elements and relationships in the field of Big Data fit together. This motivated us to create a context model (see Figure 1) to fill the void. Among the benefits of such a model is that it gives a sense of belonging to various players (such as practitioners, organisations, clients, users, and researchers) as to where they fit in, in the field of Big Data.

Still, this model is a result of only general observations of current dynamics in the field. By no means do we claim its completeness. No doubt, time will uncover its flaws, hopefully leading to its gradual evolution and improved interpretation and representation of the practices, research, user benefits, and the business of Big Data.

Upon reflection on the model, this position paper has highlighted example research challenges in BDSE: requirements, architecture, and testing and maintenance sub-areas. Space did not permit to elaborate on these or include other research challenges (e.g.: debugging, implementation designs, parallel infrastructure, linkage with business, deployment of applications and services, and research-practice link). Thus, one take-away from this paper is that BDSE for applications development is a fertile area for new research and practice.

ACKNOWLEDGMENTS

The authors are very grateful to Ibtehal Noorwali for the groundwork on jobs data as well as on requirements scenarios.

REFERENCES

[1] D. Vesset, et al., “Worldwide Big Data Technology and Services 2013–2017 Forecast,” IDC Market Analysis, 244979, Dec. 2013.

[2] “Apache Mahout”, https://mahout.apache.org/.

[3] “0xdata, H2O”, http://0xdata.com.

[4] “Amazon Web Services”, http://aws.amazon.com/.

[5] P. Francisco, *IBM PureData System for Analytics Architecture: A Platform for High Performance Data Warehousing and Analytics*. IBM, 2014.

[6] P. Bourque and R.E. Fairley, eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society, 2014.

[7] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, Wiley, 2009.

[8] L. Zhang, “A framework to model big data driven complex cyber physical control systems,” in *20th Int. Conf. on Autom. and Comp. (ICAC)*, 2014, pp. 283–288.

[9] C. L. Philip Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data,” *Inf. Sci.*, vol. 275, pp. 314–347, Aug. 2014.

[10] L. Chung, K. Cooper, and S. Courtney, “COTS-Aware requirements engineering: The CARE process,” in *Proc. of Int. Wkshp. on Req. Eng. on COTS (RECOTS)*, 2004.

[11] R. Ferrari, J. A. Miller, and N. H. Madhavji, “A controlled experiment to assess the impact of system architectures on new system requirements,” *Requir. Eng.*, vol. 15, no. 2, pp. 215–233, Mar. 2010.

[12] A. Mockus, “Engineering Big Data Solutions,” in *Proc. of the on Future of Software Engineering*, 2014, pp. 85–99.

[13] A. V. Miranskyy and E. Cialini, “Scaling of DB2 for Linux, UNIX, and Windows memory-related configuration parameters on a test system.” IBM developerWorks, 2011.

[14] “IBM - Infosphere Optim Test Data Management”, http://www.ibm.com/software/products/en/infosphere-optim-test-data-management.

[15] A. Alexandrov, C. Brücke, and V. Markl, “Issues in Big Data Testing and Benchmarking,” in *Proc. of the 6th Int. Wkshp. on Testing Database Systems*, 2013, pp. 1:1–1:5.

[16] C. Whei-Jen, et al., *Getting Started with IBM InfoSphere Optim Workload Replay for DB2*. IBM Redbooks, 2015.

[17] Y. Wang, et al., “Real Application Testing with Database Replay,” in *Proc. of the 2nd Int. Wkshp. on Testing Database Sys.*, 2009, pp. 8:1–8:6.