Iran Journal of Computer Science (2018) 1:3–10
https://doi.org/10.1007/s42044-017-0003-0

ORIGINAL ARTICLE

Big Data and software engineering: prospects for mutual enrichment

Timothy Arndt[1]

Received: 26 October 2017 / Accepted: 28 November 2017 / Published online: 11 December 2017

Abstract
Software engineering has evolved over the last 50 years, initially as a response to the so-called software crisis (the problems that organizations had producing quality software systems on time and on budget) of the 1960s and 1970s. Software engineering (SE) has been defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software". Software engineering has developed a number of approaches to areas such as software requirements, software design, software testing, and software maintenance. Software development processes such as the waterfall model, incremental development, and the spiral model have been successfully applied to produce high-quality software on time and under budget. More recently, agile software development has gained popularity as an alternative to the more traditional development methods for development of complex systems. Within the last decade or so, advances in technologies such as mobile computing, social networks, cloud computing, and the Internet of things have given rise to massive datasets which have been given the name Big Data (BD). Big Data has been defined as data with 3Vs—high volume, velocity, and variety. Big Data contains so much data that low probability events are captured in the data. These events can be discovered using analytics methods and turned into actionable intelligence which can be used by businesses to gain a competitive advantage. Unfortunately, the very scale of BD often renders inadequate SQL-based relational database systems which have formed the backbone of data intensive systems for the last 30 years, requiring new NoSQL technologies to be effective. In this paper, we will explore how well-established SE technology can be adapted to support successful development of BD projects, as well as how BD techniques can be used to increase the utility of SE processes and techniques. Thus, BD and SE may mutually support and enrich each other.

Keywords Big Data · Software engineering · Software analytics · Data mining

1 Introduction

1.1 Software engineering

Software engineering has evolved over the last 50 years (the term coming into widespread use after it was used in the title of a conference on the topic sponsored by NATO in 1968), initially as a response to the so-called software crisis (the problems that organizations had producing quality software systems on time and on budget) of the 1960s and 1970s. The software crisis was famously described in a book by Fred Brooks published in 1975 in which he described his experiences as a project manager at IBM in the preceding decade [1].

Software engineering (SE) has been defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" [2].

As the field has matured, SE has developed a number of approaches to areas such as software requirements, software design, software testing, and software maintenance. Software development processes such as the waterfall model, incremental development, and the spiral model have been successfully applied to produce high-quality software on time and under budget [3]. More recently, agile software development has gained popularity as an alternative to the more traditional development methods for development of complex systems [4]. Generally accepted best practices for software engineering have been collected by the ISO/IEC and published as the Software Engineering Body of Knowl-

✉ Timothy Arndt
t.arndt@csuohio.edu

[1] Department of Information Systems and Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH, USA

edge (SWEBOK) [2,5]. Organizational standards such as COBIT (Control Objectives for Information and Related Technologies), a framework for IT management and IT governance developed by ISACA and the Capability Maturity Model Integration (CMMI), a process-level improvement and training program originally developed by Carnegie Mellon University's Software Engineering Institute and now administered by ISACA have become widely known (if somewhat less widely used).

1.2 Big Data

Within the last decade or so, advances in technologies such as mobile computing, social networks, cloud computing, and the Internet of things have given rise to massive datasets which are beyond the capabilities of traditional databases to handle which have been given the name Big Data.

Big Data has been defined by industry analyst Doug Laney as data with 3Vs [6]—high volume, velocity, and variety. The data that organizations are collecting in such volume are coming from social networking sites such as Facebook, sensor data often associated with the Internet of things (IoT) [7], business transactions, historical data, and other sources. Velocity reflects the fact that data are generated in real time from those social network sites, RFID tags, transactions, etc. The data thus generated must be handled in an expeditious manner in order to result in actionable intelligence for an organization. The third V—variety—is in recognition that the data can arrive in many different formats—the traditional structured format that relational databases are built to handle, as well as semi-structured and unstructured data such as text documents, XML documents, and video.

Several other Vs have been added [8]: veracity, value, volatility, and validity. Due to the huge amount of data generated from uncontrolled sources, it is necessary to clean up the data in order to make it useful since it may come from untrustworthy or fundamentally dishonest sources (veracity). Once we have eliminated the untrustworthy data, we still need to be sure that the data are relevant to the problem we want to apply the data to. We may need to remove large amounts of irrelevant data to get to that core of valuable, relevant data (validity). A related concept is volatility. Data in the Big Data context can quickly become outdated or obsolete, especially in such time-critical domains as short-term weather prediction and stock market trading. Finally, it is necessary to use the Big Data not just for its own sake, but in order to generate value for the organization. Turning data into actionable intelligence is the end point that must be reached in order for Big Data to be of value to an organization.

As Big Data was increasingly produced and seen as a valuable resource, the Apache Hadoop [9] distributed processing framework was developed as a way to run analytics-oriented Big Data applications on clustered commodity hardware. Unfortunately, the very scale of BD often renders inadequate SQL-based relational database systems which have formed the backbone of data intensive systems for the last 30 years, requiring new NoSQL (non-SQL or not only SQL) technologies [10] to be effective. NoSQL systems include HBase, BigTable, Cassandra, and MongoDB.

Big Data contains so much data that low probability events are captured in the data. These events can be discovered using analytics methods and turned into actionable intelligence which can be used by businesses to gain a competitive advantage. Analytics is the application of analysis, data, and systematic reasoning to make decisions [11]. It helps users to move from answering simple questions such as "What happened?" to more complex ones such as "How did it happen and why?" Analytics allows for summarizing, filtering, modeling, and experimenting. Techniques used include A/B testing, machine learning, natural language processing, and multilinear subspace learning.

1.3 Overview of the paper

In this paper, we will explore how well-established SE technology can be adapted to support successful development of BD projects, as well as how BD techniques can be used to increase the utility of SE processes and techniques. In particular, we will highlight how BD and SE may mutually support and enrich each other.

The rest of this paper is structured as follows. In Sect. 2, we survey research in applying software engineering methodologies in BD projects. Section 3 turns the question around and surveys research in applying Big Data technology to the software engineering process. Section 4 reviews related surveys of work at the confluence of software engineering and BD. Section 5 reflects on how the two disciplines can mutually enrich each other and points out areas of future research.

2 Software engineering for Big Data

Much research has been done in recent years on how software engineering processes can be used to improve the production of Big Data systems. Such systems present many challenges. Big Data systems need to support rapid and elastic scaling, whenever and whatever needed, across multiple datacenters [12]. Design for scalability needs to be built into such systems. A large part of the challenge is the construction of larger BD solutions (such as services) from composable elements in order to reduce the cost of construction. As the volume, velocity, and variety of data in such systems grow, it becomes more difficult to reduce the required human intervention needed to extract the required information since such approaches aren't sustainable.

2.1 Software architecture for Big Data

One of the main focuses of research in applying software engineering methods to the production of Big Data systems has been in the area of developing appropriate software architectures for such systems. A sampling of research in this area will give an idea of the flavor of the research.

Sena et al. [13] have explored a basic issue in the area of software architectures for BD—namely how software architectures of these systems can be characterized in terms of their modules and organizations, basic requirements, and main characteristics. In order to present the state of the art for BD software architectures, they conducted a systematic mapping study. They were able to identify an essential set of eight requirements for Big Data architectures (including scalability and modularity) as well as a collection of five fundamental modules for data flow (including data sources, data integration, and data storage) in order to guide architects in the development of new Big Data systems.

Big Data systems present many challenges to the software architect [14]. Among them, the fact that software architectures become tightly coupled to data and deployment architectures is of particular concern. Designs must be harmonized across software, data, and deployment architectures in order to meet quality requirements. In order to meet these requirements, one well-known approach is to select and apply a sequence of architectural tactics—elemental decision decisions which embody architectural knowledge about how to satisfy a particular concern of a quality attribute. Extended performance and availability tactics which meet the specific needs of BD and which span the data, distribution, and deployment architectures include replicated stateless Web servers, database partitioning to distribute read load, and replicated database across clusters.

Various approaches to software architecture for Big Data systems have been proposed. Chen et al. [15] propose an architecture-centric methodology to address the technical, organizational, and rapid technology change challenges of Big Data system development for Big Data Web analytics using an agile approach. This architecture-centric agile approach was able to cost-effectively achieve business goals and architecture agility in 11 Web-based systems presented by the authors. A different approach to Big Data architectures is represented by the application of the model-driven approach. Model-driven engineering (MDE), i.e., software engineering by means of models and their automated manipulation can be used to support the construction of Big Data applications. The architecture of a tool to support MDE in this context is given in [16]. A related effort is [17] which presents a practical approach based on MDE to support the semi-automated development of Big Data software systems which use the MapReduce model. The approach is composed of a framework, metamodels, transformation definitions, and visual Alf which is used to enhance UML to describe the behavior of the models.

Architectures for specific types of Big Data software systems have also been proposed. DBSA [18] is a device-based software architecture for data mining. Each processing task is modeled as a component, and the components are linked together to form a data mining application. Each processing task can be thought of as a device, giving rise to the name of the architecture. Bolster [19] is a software reference architecture for semantic-aware Big Data systems. Such systems have components to simplify data definition and data exploitation; in particular, they can leverage metadata in order to aid the user in their decision-making process. A framework incorporating an architecture to support stream processing, batch processing, and deep learning to reveal knowledge hidden in video data via online video processing has been described in [20]. A pipeline-based architecture for heterogeneous execution environments in Big Data systems has been presented by Wu and co-authors [21].

Finally, a model for cost-effective, systematic risk management in agile Big Data system development called Risk-Based Architecture-Centric Strategic Prototyping (RASP) model [22] has been developed to help software architects of BD systems deal with strategic prototyping to manage risks involved in developing such systems.

2.2 Testing and debugging Big Data systems

The testing, debugging, and benchmarking of BD systems have also been an active field of research.

Big Data systems are complex, with numerous dynamic components. Any of these components can fail, leading to a system crash or degraded performance, reliability, or security. Given the interdependent nature of the systems components, finding the root cause of the failure can be a challenge. One of the methods that analysts employ to pinpoint a problem's root cause is examination of operational data—logs and traces—which is generated by the components of the BD system [23]. Such a log is a record of temporal events which has been captured during an operation of the system. Such a log might contain program execution paths, events, or user activities. The logs of Big Data systems themselves exhibit the 5V characteristics of such systems—velocity, volume, variety, veracity, and value. This makes analyzing logs of the systems problematic. Miranskyy et al. [23] describe issues which arise in analyzing these logs and suggest solutions for each. The issues they identify are: scarce storage; unscalable log analysis; inaccurate capture and replay of logs; inadequate log-processing tools; incorrect log classification; a variety of log formats; and inadequate privacy of sensitive data.

Another approach which has been suggested over the years for proving the correctness of systems is formal verification.

Given the huge amounts of data generated by BD these systems, new formal verification techniques are required for them [24]. The powerful computing resources associated with cloud computing can fundamentally transform formal verification techniques by parallelizing and distributing the verification techniques themselves. Camilli [24] introduces a distributed approach to formal verification which exploits Big Data techniques to enable the verification of very large, complex systems using Big Data approaches and cloud computing facilities.

In order for Big Data systems to be considered to be correct, they must meet performance standards, due to the huge datasets to be processed. Unfortunately, reliable performance benchmarking of Big Data systems is problematic [25]. Shapira and Chen [25] document experiences at Cloudera, a leading Big Data vendor, in the area of benchmarking Big Data systems. They identify a number of important principles of conducting performance benchmarking and assessing others' results for Big Data systems. Among these principles, workload and hardware choices should be relevant to expected use; modifications of benchmark standards should be documented and justified; Big Data systems should be tested along multiple dimensions of big scale; care should be taken to ensure that only one parameter is changed when testing is meant to compare systems on that parameter alone; having a model of expected system behavior is mandatory; benchmark results should contain sufficient information to be reproducible; results, tables, and charts should be clear, meaningful, and not misleading.

2.3 Software engineering process for Big Data systems

Several researchers have investigated how software engineering processes can be tailored to fit the specific needs of Big Data projects.

Saltz [26] looked at how a capability and maturity model (such as the CMMI mentioned in Sect. 1) can be adopted for Big Data projects which he characterizes as operating at a low level of process maturity. Thirteen factors were identified which would affect adoption of a Big Data CMM. These can be grouped as relative advantage, complexity, compatibility, and observability. There is a perceived relative advantage of using a Big Data CMM in terms of project cost and duration as well as having the results be more useful to the project champions. It was often perceived to be compatible with the current organization on several axes. Increased observability was another plus, while complexity was a negative factor.

Researchers have also examined how the software engineering life cycle processes can be specialized for Big Data analytics projects [27,28]. Such specialization is required since the project goals, scope, and functional requirements are less explicit in Big Data projects than in more traditional software projects. Lin and Huang [27] note that the incompletion rate for Big Data projects, at 55%, is much higher than more traditional software projects, at 38%. Considering the 4V characteristics—volume, velocity, variety, and variability—variety adds the most complexity to the software process since it leads to higher uncertainty and adds unexpected elements to the data relations. Lin [27] proposes a number of software processes be created and added to the software life cycle for BD projects to handle these unique characteristics. The new processes are data inventory process, data requirement analysis process, data clean process, and data innovation process.

Agile methodologies are rapidly gaining popularity in the software industry, including such methodologies as Scrum and Extreme Programming. These methodologies help developers quickly deliver a system that meets functional requirements while being able to adapt to changes in customer requirements as well as to customer feedback. Non-functional requirements, on the other hand, are often ignored. Sachedva and Chung [29] propose a novel approach to handling security and performance non-functional requirements for Big Data projects. They present a case study showing how their approach helps deal with these types of requirements in an agile methodology.

2.4 Managing Big Data projects

Much of the research surveyed in this section deals with providing tools and techniques for developers of Big Data projects. There is also the need, however, to consider the managers of Big Data projects. (The CMM mentioned by Saltz [26] might fit in this category.)

Companies are increasingly looking at adopting Big Data analytics in order to better understand customers and to provide those customers with customized services. Dutta and Bose [30] present a framework which can provide organizations a holistic roadmap in conceptualizing, planning, and successfully implementing Big Data projects and validate this framework through their observation of a descriptive case study of an organization—Ramco Cements Limited—which has implemented such a project.

3 Big Data for software engineering

The amount of research which has been done on how Big Data techniques can be integrated into the software engineering process seems to be somewhat more limited than the research in applying software engineering to the production of Big Data products. In this section, we will survey some representative research in this area to give an idea of the areas that researchers are exploring.

3.1 Analytics for software engineering

Thomas Zimmerman at Microsoft Research and several colleagues have published several papers exploring how Big Data analytics can be of use to software developers and software managers. In the first paper we will look at, Buse and Zimmerman carried out a survey of 110 software developers and software managers at Microsoft to determine the data and analysis needs of professional software engineers [11]. The survey explored their decision-making process, their need for artifacts and indicators, and various scenarios in which they would use analytics. Based on the survey, they propose several guidelines for analytics tools in software development. The tools should be simple to use, fast, and produce concise output. Due to the diverse analyses of software engineers, along with the fact that they consider most indicators to be important (not favoring one or a few), software analytics tools should support diverse types of artifacts and multiple indicators. Also, developers and managers want to be able to drill down into data based on time, organizational structure, and system architecture. A proof-of-concept analytics tool was implemented to elicit further feedback from developers on how future analytics tools should be designed.

A second paper, this one by Begel and Zimmerman [31], reports the result from two surveys of Microsoft software professionals. The first survey solicited questions that software engineers would like to have data scientists investigate and software, software processes and practices, about software engineers. In total, 1500 randomly chosen Microsoft engineers were surveyed. Of these, 203 software engineers responded with a total of 679 questions that they wanted to have investigated. An open card sort was used to group the questions into 12 categories and identify 145 questions. A second survey asked a different pool of Microsoft engineers—2500 in total—to prioritize the 145 questions from the first survey. In total, 16,765 ratings were received from 607 Microsoft engineers. These results were used to identify differences of opinion between various demographic groups (e.g., software engineering subdiscipline, management role, geographic region, years at company). The results can help researchers, practitioners, and educators to better focus their efforts on topics of importance to a large company (Microsoft).

A third paper by Zimmerman and co-workers [32] reports a large-scale survey of Microsoft employees, but this time instead of surveying software developers and managers about their analytics needs, they reached out to data scientists at the company in order to understand their educational background, problem topics that they work on, tool usages, and activities. A total of 793 professional data scientists at Microsoft were surveyed out of 2397 initially contacted. (599 of these were data scientists by discipline, while 1798 were data science enthusiasts with interests in the area, but not specialists in it.) Among the results, they found that the problem topics that most engaged the respondents included user engagement, software productivity and quality, domain-specific problems, and business intelligence. The tools they reported using were SQL, Excel, R, and Python. The most popular types of data they worked with were customer usage, business data, and execution behavior. The types of data scientists identified included polymath, data evangelist, data preparer, data shaper, and data analyzer. Among the challenges the data scientists face involving data are data quality, data availability, and data preparation. Challenges related to analysis included scale and machine learning. Asked how they ensure the quality of their work, they gave responses including cross-validation is multidimensional, check data distribution, type and schema checking, and repeatability. More structured tool support is needed for these approaches to validation.

3.2 Data mining software repositories

The mining software repositories field analyzes the huge amounts of rich data which have accumulated over the years in the software repositories of large organizations, as well as open-source software development projects, in order to glean interesting and hopefully actionable information about software systems and processes [33] using analytics methods from BD. Topics which have been examined include software evolution, models of software development processes, characterization of software developers and their activities, use of machine learning techniques on software project data, and bug and failure prediction.

A few examples of such studies will give an idea of the type of research done in mining software repositories. Choetkiertikul et al. [34] have developed a method to predict whether an issue is at risk of being delayed, and, if it is, the extent of the delay. Open-source projects are used to identify 19 risk factors which are used to train prediction models. Evaluation indicates that the method is effective—likelihood of delay predicted with 9% precision and 61% recall and the extent of delay are shown to be predicted with some accuracy as well. Exception handling bug hazards in Android are examined by Coelho et al. [35] who describe an empirical study of bug hazards in more than 6,000 Java exception stack traces extracted from over 600 open-source Android projects. The bug hazards were subsequently further assessed by developers through a survey. The results of the research are a call for tool support to help developers better understand exception handling and wrapping logic. Batarseh and Gonzalez [36] take a somewhat different approach to the problem of predicting failures in agile software development through data analytics. Rather than mine a software repository for the actionable information needed for the stated goal, they look at the data derived during the previous stages of the agile

software development process for a single project. The previous individual sprints in the agile development process are examined to determine mean time between failure (MTBF) for those sprints. Regression modeling is then performed as part of a methodology called analytics-driven testing in order to predict future errors and their locations in a sprint-based life cycle (scrum).

Three software engineering datasets [37–39] are also presented in the special section of Empirical Software Engineering on mining software repositories [33]. These datasets are given a thorough description in order that they might be of use to future researchers and practitioners in mining software repositories and in putting these valuable resources to the test.

Finally, Hentschel et al. [40] consider whether Big Data approaches can benefit the software measurement community. They looked at project data available on GitHub and gave examples of how Big Data applications can benefit software measurement practices and communities.

3.3 Visual analytics for software engineering

One of the main techniques which are applied to BD nowadays is visual analytics. Visual analytics is the science of analytical reasoning facilitated by visual interfaces. It combines data mining and visualization techniques in order to support reasoning about the phenomena in a dataset. Visual analytics allows one to combine the strengths of both machine learning and human intuition for the purpose of knowledge discovery.

Visual analytics has been applied to the software engineering process in several different ways. Telea and Voinea [41] present an adaptation of the visual analytics framework to the context of software understanding for maintenance. In particular, they present an instance of a visual software analytics application for the build optimization of large-scale code bases. Data mining and visualization techniques are combined and adapted to answer questions asked by developers in reducing the build cost of large-scale code bases. The application has been used by software architects on an industrial C code base of 17.5 million lines of code. User reports were positive. Another application of visual analytics in software engineering has been reported in the area of management of human resources during the development and maintenance of a software system [42]. Software development projects often are beset by problems associated with rotation and distributed software development. In such a context, project managers and project leads often lack necessary prior knowledge for decision-making on human resources and must, anyway, perform such a task in an empirical manner. In this area, visual analytics can be applied to software evolution to support project leads and managers in decision-making via visualization and interaction techniques. For example, the approach allows one to determine which programmer has led a project or contributed more to the development and maintenance of system by revisions. Using this and other similar information discoverable by visual analytics, software project leads and managers can make appropriate decisions regarding task assignment to developers and staff substitutions due to staff turnover or other reasons. Experimentation on using data from SCM logs and source code associated with revisions with visual analytics techniques was carried out in the context of a novel architecture.

3.4 Self-adaptive systems

As a final example of the use of Big Data analytics in a software engineering context (software design), consider the paper by Schmid and co-authors [43]. They look at how to make large-scale software-intensive distributed systems self-adaptive. An example would be a city traffic management system, which collects data from sources such as cars and traffic lights in order to optimize traffic guidance. Such systems are difficult to make self-adaptive since their adaptation must consider the current situation of the system as a whole (possibly including predictions of future situations). In order to solve the problem, they use a very simple system model with essential input and output parameters. Their model relies on Big Data analytics to evaluate large-scale datasets at run time to effectively optimize systems based on adaptation. A tool for adaptation of a system based on such analysis is presented.

4 Related research

In this section, we will look at related research regarding the interplay of Big Data and software engineering.

The possibilities of applying SE techniques to BD projects (or BD techniques to SE processes) have certainly not been overlooked by researchers in the two fields. As an example of the growing interest in the area, the 1st International Workshop on Big Data Software Engineering (BDSE) was held as part of the IEEE/ACM sponsored 37th IEEE International Conference on Software Engineering in May 2015 in Florence, Italy, and a second workshop (BIGDSE 2016) was held in May 2016 in Austin, Texas. A special issue of IEEE Software on software engineering for Big Data systems was published in 2015 [44].

Given the growing interest in the intersection of BD and SE, as well as the large number of recent research papers in the area, it is no surprise that several authors have surveyed this area.

Bagriyanik and Karahoca [45] have performed a systematic literature review of the intersection of BD and SE published between January 2010 and October 2015. They

aimed to answer two research questions: one concerning the main areas where BD and SE are interacting and to what extent (which subareas of SE are using BD) and the second asking which are the SE artifacts which are most frequently used for BD processing in the research surveyed. The methodology they used relied on keyword search and analysis to answer the questions. They found an increasing number of papers as they moved closer to the current year and that the most popular SE areas for applying BD according to the keywords matched in their search for papers are software quality, development, project management, and human–computer interaction. The software artifacts most frequently used as the subjects of BD analysis in the papers they retrieved were source code, source code changes, bug-related data, and operational data.

Rouhani and co-authors [46] investigated the effects of Big Data on the design and development of information systems by a systematic literature review. Their findings indicated that the volume attribute of Big Data affects software engineering tools, while the variety and velocity attributes affect software engineering methods, indicating that both of these components of software engineering will need to be adapted in order to support Big Data projects.

Kumar and Alencar [47] (see also Kumar's Master's Thesis on the same topic [48]) used a methodology similar to [45]—a systematic literature review—to study how researchers are using the different phases of the software development life cycle to support BD system development (a somewhat more limited search than [45]). The results looked at the application domain reported. The most highly cited was a somewhat catch-all "information technology" domain. Others popular domains included health care, geographic information systems, and transport. As far as the SE subfields, the three most widely reported were architecture, framework, and design. The subareas don't exactly match those of [45], so no exact comparison of the two results is possible.

Otero and Peter [49] attempt to set a research agenda for engineering Big Data analytics software. They look at several problems in the engineering of BD software: the requirements problem, the design problem, the construction problem, and the testing problem. The authors give a simple formalization of each of these problems in the general SE context and then show how the specific BD requirements complicate the problem. Their efforts are oriented toward making the traditional SE goals of creating software on time and within budget apply to BD systems.

Another attempt to set a research agenda is given by Madhavji et al. [50] in a paper presented at BIGDSE 2015. The authors begin by noting that a preponderance of the effort being expended in BD systems development is in the areas of infrastructure development and analytics rather than in applications software development. In an attempt to remedy this imbalance, the authors present a context model of Big Data software engineering. Based on this context model, the authors then sketch research challenges in requirements engineering, architectures, and testing and maintenance. The results are similar to those in [49], although the models are quite different (conceptual versus more mathematical/formal).

5 Discussion and future research

Our survey of the intersection of Big Data and software engineering shows that this is very much a lively topic for researchers around the globe. While relatively more research has been done on how software engineering techniques, methodologies, etc. can be applied to Big Data projects than has been done on how Big Data analytics and techniques can help improve the software engineering process, both sides of the coin are well represented.

Our hope is that, in the future, both sides of this relation can benefit. In other words, Big Data helps to improve software engineering processes, and the improved software engineering process can in turn develop better Big Data projects. This approach was not widely seen in the works we surveyed—perhaps it is best represented in the paper by Camilli [24]. We will be turning our attention to several approaches to try to spur development of this mutual enrichment idea in the near future.

References

1. Brooks, F.P.: The Mythical Man-Month. Addison-Wesley, (1975)
2. Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L.L.: Guide to the Software Engineering Body of Knowledge, IEEE, (2004)
3. Sommerville, I.: Software Engineering, 10th edn. Pearson, (2015)
4. Agile Manifesto, http://agilemanifesto.org
5. Bourque, P., Fairley, R.E. eds.: SWEBOK: Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society Press, (2014)
6. Laney, D.: "3-D Data Management: Controlling Data Volume, Velocity and Variety", META Group Research Note, February 6, (2001)
7. Kopetz, H.: "Internet of Things", Real-Time Systems, Real-Time Systems Series. Springer, (2011)
8. Khan, M.A., Uddin, M.F., Gupta, N.: "Seven V's of Big Data Understading Big Data to Extract Value", Proceedings of the 2015 Zone 1 Conference of the American Society for Engineering Education, pp. 1-5, (2014)
9. Zikopoulos, P., Eaton, C.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 1st edn. McGraw-Hill Osborne Media, (2011)
10. Leavitt, N.: Will NoSQL Databases Live Up to Their Promise? Computer **43**(2), 12–14 (2010)
11. Buse, R.P.L., Zimmerman, T.: "Information Needs for Software Deveopment Analytics", Proceedings 34th International Conference on Software Engineering – ICSE 2012, pp. 987-996, (2012)

12. Szyperski, C., Peticlerc, M., Barga, R.: Three Experts on Big Data Engineering. IEEE Software **33**(2), 68–72 (2016)
13. Sena, B., Allian, A.P., Nakagawa, E.Y.: "Characterizing Big Data Software Architectures: A Systematic Mapping Study", Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse, (2017)
14. Gorton, I., Klein, J.: Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems. IEEE Software **32**(3), 78–85 (2015)
15. Chen, H.M., Kazman, R., Haziyev, S.: Agile Big Data Analytics for Web-Based Systems: An Architecture-Centric Approach. IEEE Transactions on Big Data **3**(2), 234–248 (2016)
16. Guerriero, M., Tajfar, S., Tamburri, D.A., Di Nitto, E.: "Towards a Model-Driven Design Tool for Big Data Architectures", Proceedings of the 2nd International Workshop on BIG Data Software Engineering (BIGDSE '16), ACM, New York, NY, USA, pp. 37-43, (2016)
17. Osvaldo, S.S., Lopes, D., Silva, A.C., Abdelouahab, Z.: Developing Software Systems to Big Data Platform Based on MapReduce Model: An Approach Based on Model Driven Engineering. Information and Software Technology **92**, 30–48 (2017)
18. Kätevä, J., Laurinen, P., Rautio, T., Suutala, J., Tuovinen, L., Röning, J.: "DBSA: a Device-Based Software Architecture for Data Mining", Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), pp. 2273-2280, (2010)
19. Nadal, S., Herrero, V., Romero, O., Abelló, A., Franch, X., Vansummeren, S., Valerio, D.: A Software Reference Architecture for Semantic-Aware Big Data Systems. Information and Software Technology **90**, 75–92 (2017)
20. Zhang, W., Xu, L., Li, Z., Lu, Q., Liu, Y.: A Deep-Intelligence Framework for Online Video Processing. IEEE Software **33**(2), 44–51 (2016)
21. Wu, D., Zhu, L., Xu, X., Sakr, S., Sun, D., Lu, Q.: Building Pipelines for Heterogeneous Execution Environments for Big Data Processing. IEEE Software **33**(2), 60–67 (2016)
22. Chen, H., Kazman, R., Haziyev, S.: Strategic Prototyping for Developing Big Data Systems. IEEE Software **33**(2), 36–43 (2016)
23. Miranskyy, A., Hamou-Lhadj, A., Cialini, E., Larsson, A., Liu, Y.: Operational-Log Analysis for Big Data Systems: Challenges and Solutions. IEEE Software **33**(2), 52–59 (2016)
24. Camilli, M.: "Formal Verification Problems in a Big Data World: Towards a Mighty Synergy", Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion 2014). ACM, New York, NY, USA, pp. 638-641, (2014)
25. Shapira, G., Chen, Y.: Common Pitfalls of Benchmarking Big Data Systems. IEEE Transactions on Services Computing **9**(1), 152–160 (2016)
26. Saltz, J.: "Acceptance Factors for Using a Big Data Capability and Maturity Model", Proceedings of the 25th European Conference on Information Systems (ECIS), pp. 2602-2612, (2017)
27. Lin, Y., Huang, S.J.: The Design of a Software Engineering Life Cycle Process for Big Data Projects. IT Professional (2017). https://doi.org/10.1109/MITP.2017.265105546
28. Al-Jaroodi, J., Hollein, B., Mohamed, N.: "Applying software engineering processes for big data analytics applications development", Proceedings IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), (2017)
29. Sachdeva, V., Chung, L.: "Handling Non-Functional Requirements for Big Data and IOT Projects in Scrum", Proceedings 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, (2017)
30. Dutta, D., Bose, I.: Managing a Big Data project: The case of Ramco Cements Limited. International Journal of Production Economics" **165**, 293–306 (2015)
31. Begel, A., Zimmerman, T.: "Analyze This! 145 Questions for Data Scientists in Software Engineering", Proceedings 36th International Conference on Software Engineering – ICSE 2014, pp.12-23, (2014)
32. Kim, M., Zimmermann, T., DeLine, R., Begel, A.: Data Scientists in Software Teams: State of the Art and Challenges. IEEE Transactions on Software Engineering (2017). https://doi.org/10.1109/TSE.2017.2754374
33. Robbes, R., Kamei, Y., Pinzger, M.: Guest Editorial: Mining Software Repositories. Empirical Software Engineering **22**, 1143–1145 (2017)
34. Choetkiertikul, M., Dam, H.K., Tran, T., Ghose, A.: Predicting the Delay of Issues with Due Dates in Software Projects. Empirical Software Engineering **22**, 1223–1263 (2017)
35. Coelho, R., Almeida, L., Gousios, G., et al.: Exception Handling Bug Hazards in Android: Results From a Mining Study and an Exploratory Survey. Empirical Software Engineering **22**(3), 1264–1304 (2017)
36. Batarseh, F., Gonzalez, A. J.: "Predicting Failures in Agile Software Development through Data Analytics", Software Quality Journal, pp. 1-18, 2015, https://doi.org/10.1007/s11219-015-9285-3
37. Sawant, A., Bachelli, A.: fine-GRAPE: Fine-Grained APi Usage Extractor - An Approach and Dataset to Investigate API Usage. Empirical Software Engineering **22**(3), 1348–1371 (2017)
38. Spinellis, D.: A repository of Unix history and evolution. Empirical Software Engineering **22**(3), 1372–1404 (2017)
39. Caneill, M., Germán, D.M., Zacchiroli, S.: The Debsources Dataset: Two Decades of Free and Open Source Software. Empirical Software Engineering **22**(3), 1405–1437 (2017)
40. Hentschel, J., Schmietendorf, A., Dumke, R.R.: "Big Data Benefits for the Software Measurement Community", 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), (2016), https://doi.org/10.1109/IWSM-Mensura.2016.025
41. Telea, A., Voinea, L.: Visual Software Analytics for Build Optimization of Large-Scale Software Systems. Computational Statistics **26**(4), 635–654 (2011)
42. González-Torres, A., García-Peñalvo, F.J., Therón-Sánchez, R., Colomo-Palacios, R.: Knowledge Discovery in Software Teams by Means of Evolutionary Visual Software Analytics. Science of Computer Programming **121**(1), 55–74 (2016)
43. Schmid, S., Gerostathopoulos, I., Prehofer, C., Bures, T.: "Self-Adaptation Based on Big Data Analytics: A Model Problem and Tool", Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 102-108, (2017)
44. Gorton, I., Bener, A.B., Mockus, A.: Software Engineering for Big Data Systems. IEEE Software **33**(2), 32–35 (2016)
45. Bagriyanik, S., Karahoca, A.: Big Data in Software Engineering: A Systematic Literature Review. Global Journal of Information Technology **6**(1), 107–116 (2016)
46. Rouhani, S., Rotbei, S., Shamizanjani, M.: "Meta-Synthesis of Big Data Impacts on Information Systems Development", Journal of Management Analytics vol. 4, no. 2, (2017)
47. Kumar, V.D., Alencar, P.: "Software Engineering for Big Data Systems: Domains, Methodologies and Gaps", Proceedings of IEEE International Conference on Big Data, (2016)
48. Kumar, V.D.: "Software Engineering for Big Data Systems", Masters Degree Thesis, University of Waterloo, (2017)
49. Otero, C.E., Peter, A.: Research Directions for Engineering Big Data Analytics Software. IEEE Intelligent Systems **30**(1), 13–19 (2015)
50. Madhavji, N.H., Miranskyy, A., Kontogiannis, K.: "Big Picture of Big Data Software Engineering: With Example Research Challenges", Proceedings of the First International Workshop on BIG Data Software Engineering (BIGDSE '15), pp. 11-14, (2015)