Investigating the Applicability of Architectural Patterns in Big Data Systems

JOAO RAFAEL VARELA, University of São Paulo
LINA GARCÉS, University of São Paulo
ANA PAULA ALLIAN, University of São Paulo
ELISA YUMI NAKAGAWA, University of São Paulo

Context: Architectural patterns are well-proven solutions to solve software systems issues related to quality attributes. They define a set of high level structure properties, a vocabulary to describe them, and assumptions and constraints on how they should be employed to correctly solve the common problem. The use of architectural patterns has been considered a good practice to design software architectures with quality, improving the reuse and understanding of rationale of architectural decisions. In another perspective, big data systems have gained attention mainly by their inherent 5Vs characteristics of velocity, veracity, volume, value, and variety, which requires the accomplishment of several quality attributes, such as scalability, performance, security, integrity, flexibility, interoperability, and of business goals. **Problem:** Despite of big data systems are increasing in importance for different domains, few studies have investigated the quality of such systems from a perspective of software architecture. For the best of our knowledge, there are no work investigating which of the existing architectural patterns have been used to design software architectures of big data systems with a focus on quality. **Objective:** In this paper, we investigate the applicability of well-established architectural patterns by analyzing their benefits and liabilities when used to architect big data systems. **Method:** We analyzed studies reporting software architectures of big data systems. Studies were identified through the conduction of a systematic mapping study that was reported in a previous work of authors. We used those studies to identify architectural patterns, requirements of quality attributes, and problems and liabilities addressed by those solutions. **Results:** We identified that the common architectural patterns used for big data systems (i.e., layers, pipe and filters, shared repository, and broker architecture) do not differ from those used in other systems, however, they have important impacts on big data systems qualities and on their characteristics that are not reported in others studies. When used independently, those patterns partially allow accomplishment of big data issues and quality attributes requirements, such as scalability and performance. As future works, we intend to study the combination of these patterns besides investigate patterns used in industry to support the design and documentation of software architecture for big data systems.

Categories and Subject Descriptors: D.2.11 [**Software Architectures**] Patterns; E.0 [**Data**] General; H.4.0 [**Information Systems Applications**] General

General Terms: Big Data

Additional Key Words and Phrases: Software Architectures, Architectural Patterns

1. INTRODUCTION

Architectural patterns facilitate reasoning and understanding about system's design, by providing a set of high level structural properties, a vocabulary to describe them, and assumptions and constraints on how they should be employed [Ingram et al. 2015]. They represent a pre-defined set of components, responsibilities and relationship among them and can help define the basic characteristics and behavior of an application [Richards 2015; Meunier

Author's address: Av. Trabalhador São-Carlense, 400 - Parque Arnold Schimidt, São Carlos - SP, Brazil, 13566-590.
Email: joao.rafael.varela@gmail.com; linamgr@icmc.usp.br; ana.allian@usp.br; and elisa@icmc.usp.br

et al. 1996]. They have been gained attention in systems architecture field because applications lacking a formal and well-established architecture are generally tightly coupled, brittle, difficult to change, and without a clear vision or direction [Richards 2015]. These patterns indicate the first place in the software architecture design in which quality requirements should be addressed. Such quality requirements are thus satisfied by the various structures designed into the architecture, and the behaviors and interactions of the elements that populate those structures [Bass et al. 2013].

In another perspective, the concept of big data has emerged and can be defined as a high volume, high velocity, and high variety of data that demands cost-effective, innovative forms of information processing for enhanced insight and decision making [Beyer and Laney 2012]. With the rapid increase in data generation and processing, software and hardware resources have became overloaded and require new innovative approaches to effectively manage and use the data [Al-Jaroodi and Mohamed 2016]. Such issues led to the establishment of the 5 V's model (i.e., volume, velocity, variety, veracity, and value), which characterizes the main concerns when dealing with these data complexities.

Due to the difficulty of managing large data sets, attention to software architecture is required to support big data innovations [Sena et al. 2017]. Particularly, to assure efficient management of data generated in different domains, big data architectures should be well structured and guarantee critical quality requirements [Begoli and Horey 2012]. In this scenario, architectural patterns can be used to guide architects during the big data design, but selecting an appropriate pattern is a challenge task because each one impacts on several factors in big data systems, including their quality attributes. Understanding the way existing patterns have been used to design big data systems would facilitate the interactions between patterns and quality attributes.

Thus, the driving motivation for this study is to understand what architecture patterns have been used in big data systems and in which level such patterns impact the accomplishment of quality attributes in those systems. The definition of such attributes is complex because it needs to consider the mapping of the intrinsic characteristics of the big data context (5V's) to attributes that can be approached and measured architecturally. Therefore, in this study we intend to answer the following research questions: **RQ1 -** What architecture patterns are most commonly used in big data systems?, and **RQ2 -** How the architectural patterns used those systems impact their 5 V's characteristics?".

To solve these research questions, in this work, we studied architectural patterns used to support the design of these architectures. Moreover, the impact (positive or negative) of those patterns on quality attributes of big data systems was investigated. We identified those patterns through the conduction of a systematic mapping study reported in a preliminary work [Sena et al. 2017].

The remainder of this paper is organized as follows. In Section 2, background about architectural patterns, big data concepts, and big data architectural requirements are described. Section 3 presents the methodology for conducting this study. Section 4 analyses each architectural pattern considering big data requirements. Section 5 discusses related to the applicability of the selected patterns for the big data context. Section 6 presents the main threats to the validity of our study, while Section 7 presents the limitations of obtained results. Finally, our conclusions and perspectives for future work are presented in Section 8.

2. BACKGROUND

In this section are presented important concepts considered in this work. Firstly, we presented the definition of architectural patterns and their relationship with quality attributes. After, an overview of Big Data domain is detailed highlighting its most important characteristics. Finally, architectural requirements of Big Data systems are described.

2.1 Architectural Patterns

In its most establish definition, "A pattern describes a problem which occurs over and over again in the environment, and then describes the core of the solution of that problem so that it can be applied a million times over" [Alexander 1977]. Architectural patterns constitute a mechanism for capturing domain experience and knowledge to allow it to be reapplied when a new architectural problem is encountered [Pressman and Bruce R. Maxim 2014]. According to [Bass et al. 2013], an architectural pattern establishes a relationship between:

A context. A recurring, common situation in the world that gives rise to a problem.

A problem. The problem, appropriately generalized, that arises in the given context. The pattern description outlines the problem and its variants, and describes any complementary or opposing forces. The description of the problem often includes quality attributes that must be met.

A solution. A successful architectural resolution to the problem, appropriately abstracted. The solution describes the architectural structures that solve the problem, the responsibilities of and static relationship among elements, besides the run-time behavior of and interaction between elements. Such solution includes (i) a set of element types; (ii) a set of interaction mechanisms or connectors; (iii) a topological layout of the components; and (iv) a set of semantic constraints covering topology, element behavior, and interaction mechanisms.

One of the most important aspects of the architectural patterns is to make clear what quality attributes are provided by the static and run-time configurations of elements, since software architectures are mostly driven or shaped by quality attributes requirements, which determine and constrain the most important architectural decisions [Clements and Bass 2010]. These patterns represent the first place in the software creation process in which quality requirements should be addressed. Such quality requirements are thus satisfied by the various structures designed into the architecture, and the behaviors and interactions of the elements that populate those structures [Bass et al. 2013].

2.2 Big Data

Big Data is defined as a set of techniques and technologies that require new forms of integration to uncover largely hidden values from large datasets that are diverse, complex, and of a massive scale [Hashem et al. 2015]. Early definitions focused on its characterizing based on the three V's model [Laney 2001], which concerns to the volume, velocity and variety of data. However, a more commonly accepted characterization now relies upon the following 5V's [Beyer and Laney 2012]: volume, velocity, variety, veracity and value. In this sense, different V's have been proposed such as volatility, and validity [Uddin et al. 2014], but they have not yet been consolidated in the literature.
Following, we present a brief description of the intrinsic characteristics of this 5V's.

Volume: big data first and foremost has to be "big" in size, which is measured as volume [Jain 2016]. This characteristic refers to the huge amount of data generated every second, which makes datasets too large to store and analyze using traditional database technology. Such characteristic requires distributed and parallel processing that most of regular systems were not designed for [Chen et al. 2015];

Velocity: data velocity measures the speed of data creation, streaming, and aggregation [Kaisler et al. 2013]. It is addressed in two different perspectives in the big data context: (i) the rapidly increasing speed at which new data is being created by technological advances, and (ii) the corresponding need for that data to be digested and analyzed in near real-time;

Variety: it is a measure of the richness of data representation, i.e., text, images, video, and audio. Such characteristics brings a challenge related to the effectiveness of using use large volumes of data, because first the data must be processed in order to handle incompatible data formats, non-aligned data structures, and inconsistent data semantics [Kaisler et al. 2013];

Veracity: this characteristic refers to the trustworthiness of the data. It must be handled due to the many forms of data, which turns quality and accuracy less controllable. It brings some challenges in data validation, modeling and governance, due to the need to keep its reliability [Chen et al. 2015];

Value: data value measures the usefulness of data in support the decision making by the domain stakeholders [Kaisler et al. 2013]. By investing in an infrastructure to collect and interpret data on a system-wide scale, it is important to make sure that the insights that are generated were based on accurate data and lead to improvements in the domain decisions [Jain 2016]. Although value is added as the fifth V, value is defined as the desired outcome of big data processing, and not a defining characteristic of big data itself [Uddin et al. 2014; L heureux et al. 2017]. For this reason, due to the well-established definition, value is considered only as a concern to big data and then characterizes the 5V's.

2.3 Big Data Architectural Requirements

It is crucial to design and to develop well-designed software systems capable of effectively and efficiently satisfying the various requirements, related to the need of data analysis, processing, and storage [Al-Jaroodi and Mohamed 2016]. Following, we present a brief description of each architectural requirement identified. Such requirements are to identify the critical points and forces in the architecture, and guide the architectural patterns selection.

- **Scalability:** Big data systems must be scalable, that is, must be able to increase and support different amounts of data, processing them uniformly, allocating resources without impacting costs or efficiency. To meet this requirement, it is required to distribute data sets and their processing across multiple computing and storage nodes;
- **High Performance Computing:** Big data systems must be able to process large streams of data in a short period of time, thus returning the results to users as efficiently as possible. In addition, the system should support computation intensive analytics, i.e,. support diverse workloads, mixing request that require rapid responses with long-running requests that perform complex analytics on significant portions of the data collection [Gorton and Klein 2015];
- **Modularity:** Big data systems must offer distributed services, divided into modules, which can be used to access, process, visualize, and share data, and models from various domains, providing flexibility to change machine tasks;
- **Consistency:** Big data systems must support data consistency, heterogeneity, and exploitation. Different data formats must be also managed to represent useful information for the system [Cecchinel et al. 2014]. Besides that, systems must be flexible to accommodate this multiple data exchange formats and must achieve the best accuracy as possible to perform these operations;
- **Security:** Big data systems must ensure security in the data and its manipulation in the architecture, supporting integrity of information, exchanging data, multilevel policy-driven, access control, and prevent unauthorized access [Demchenko et al. 2017];
- **Real-time operation:** Big data systems must be able to manage the continuous flow of data and its processing in real time, facilitating decision making;
- **Interoperability:** Big data systems must be transparently intercommunicated to allow exchanging information between machines and processes, interfaces, and people [Santos et al. 2017];
- **Availability:** Big data systems must ensure high data availability, through data replication horizontal scaling, i.e., spread a data set over clusters of computers and storage nodes, avoiding bottlenecks [Gorton et al. 2015]. For this, system should allow replication and it should handle failures.

3. METHODS

To identify the patterns that have been commonly used in big data software architectures, we performed a systematic mapping study (SMS), detailed in [Sena et al. 2017]. In short, SMS can provide a detailed overview of a research area by categorizing and investigating studies related to the topics of interest [Petersen et al. 2015]. For conducting our SMS, we followed the guidelines recommended by [Kitchenham and Charters 2007] and [Petersen et al. 2015]. We defined the search string *Big Data AND (Software Architecture OR Reference Architecture OR Reference Model)* that was executed in the following data libraries: ACM Digital Library[1], El Compendex[2], IEEE Digital Library[3], ISI Web of Science[4], Science Direct[5], Scopus[6], and Springer Link[7].

As results, 39 studies were considered for data extraction [Sena et al. 2017]. These studies were completely read and relevant information to their architectures was extracted. We focused on requirements, forces, architectural design, standards, and architectural solutions that were considered in each study. From those 39 studies, only 11 reported the use of architectural patterns for big data systems. We just considered information that was explicit in the text or in the architecture about the patterns. We avoided to imply any information that was not detailed by the authors of the studies, because it could cause a threat to validity.

In table I, we present the 11 studies considered to this analysis, the patterns identified in each one, and their references. We extracted information from those studies to identify how the reported architectural patterns affect or benefit the accomplishment of requirements of quality attributes in Big Data systems, as presented in next section. Table I. Final list of primary studies

ID	Title and Reference	Architectural Patterns
S1	A big data analytics architecture for Industry 4.0 [Santos et al. 2017]	Layered
S2	A proposal for a reference architecture for long-term archiving, preservation, and retrieval of big data [Viana and Sato 2015]	Layered
S3	A reference architecture for big data solutions introducing a model to perform predictive analytics using big data technology [Geerdink 2015]	Pipe and Filters, Layered
S4	A reference architecture for big data systems in the national security domain [Klein et al. 2016]	Pipe and Filters, Shared-repository, Layered
S5	A reference architecture for supporting secure big data analytics over Cloud-Enabled relational databases [Cuzzocrea 2016]	Broker, Layered
S6	An architecture to support the collection of big data in the Internet of Things [Cecchinel et al. 2014]	Broker
S7	Big data analytics architecture for Agro Advisory systems [Shah et al. 2016]	Pipe and Filters, Layered
S8	Building a big data platform for Smart Cities: Experience and lessons from Santander [Cheng et al. 2015]	Broker, Shared-repository, Layered
S9	Expanding global big data solutions with innovative analytics [Dayal et al. 2014]	Layered
S10	Managing cloud-based big data platforms: A reference architecture and cost perspective [Heilig and Voß 2017]	Layered
S11	Reference architecture and classification of technologies, products and services for big data systems [Paakonen and Pakkala 2015]	Pipe and Filters, Layered

4. ANALYSIS OF APPLYING ARCHITECTURAL PATTERNS FOR BIG DATA SYSTEMS

Based on the analyzed studies, presented in Table I, we identified that the common architectural patterns used for Big Data systems are layers, pipe and filters, shared repository, and broker architecture. In this section we analyze each of those

[1] http://portal.acm.org

[2] http://www.engineeringvillage.com

[3] http://ieeexplore.ieee.org

[4] http://www.isiknowledge.com

[5] http://www.sciencedirect.com

[6] http://www.scopus.com

[7] http://link.springer.com

patterns to verify their applicability in the big data context based on how they benefit and affect quality attributes. Architectural patterns are presented following the structure proposed by [Bass et al. 2013].

4.1 Layered Pattern

Context: Big data systems require the division of responsibility among their components. Such division must be established at both the module level (i.e., data storage, data visualization, data integration) and at the abstraction level (i.e., communication, security, management). Besides that, these systems must orchestrate the flow of data to execute business processes, and hence to ensure that the system's objectives are fulfilled and that data is not lost.

Problem: Big data architectures must allow the division of responsibilities, by grouping them based on goals and functionalities. Such architectures must provide means to process data at real time, because sometimes specific work flows depend on the results of the previous one. Moreover, the communication protocols must guarantee high performance and data scalability, avoiding bottlenecks.

Solution: The layered pattern provides a horizontal division of software responsibilities by grouping a set of functionalities in encapsulated and independent layers. The specific partitioning criteria can be defined along various dimensions, such as abstraction, granularity, hardware distance, and rate of change. Layers of an architecture generally communicate with adjacent layers through implemented interfaces. Therefore, one layer is able to make requests to the next layer, and wait for the response while responding to requests from the previous layer. Such interfaces must be scalable and implement communication protocols to ensure efficiency [Hanmer 2012].

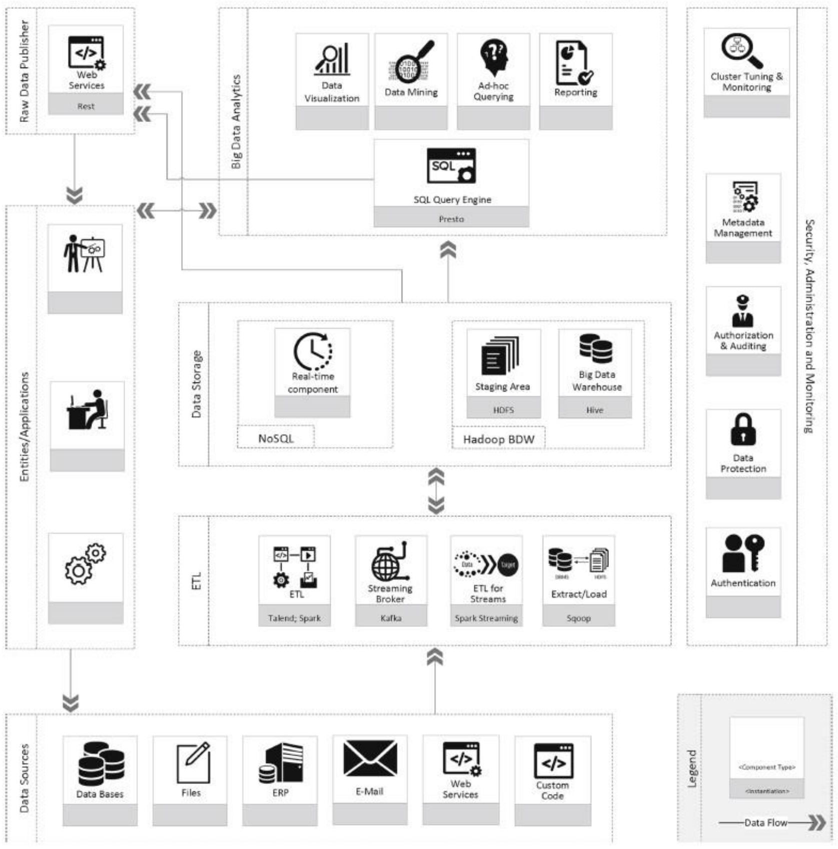
Example: Several big data architectures have been proposed based on this solution. [Santos et al. 2017] proposed a big data architecture for Industry 4.0, as presented in Figure 1. Such architecture divides the functionalities in layers according to the different steps of data analytics (e.g., storage, publisher, sources, and applications). Moreover, [Santos et al. 2017] proposed a common layer in big data architectures, focusing in security, management and monitoring, which includes components that provide base functionalities needed in the other layers, ensuring the proper operation of the whole infrastructure. Some components needed are destined to monitoring, bottlenecks detection, performance improvement by adjusting parameters of the adapted technologies, authentication, data protection, among others. In a different abstraction perspective, [Geerdink 2015] proposed a three layered architecture that separates the concerns in technology, application, and business, as depicted in Figure 2. In both layers, arrows represent data flow and can be implemented through interfaces and APIs.

Trade-offs: The waiting time and the execution of communication protocols may delay processing and cause the response to not be as fast as expected. Besides that, these communication protocols impact scalability, since the high volume of data must be processed at the same time and can cause a bottleneck between the layers.
Finally, because data have to go through different layers, the processing performance is directly impacted.

Regarding big data requirements, this pattern supports the modularity by providing a division of concerns in the application. The independence of each layer enables the implementation of different protocols for keeping the data safe to be sent only when necessary, allowing a positive impact on the security requirement of Big data.

Considering weaknesses, three of the most important big data requirements are minimized by this pattern (i.e., real-time operation, scalability, and performance). Big data requires the processing of data at run-time, because an specific layer depends on the result of the previous one. Hence, the wait time and the execution of communication protocols may delay processing and cause the response to not be as fast as expected. These communication protocols impact scalability, since the high volume of data must be processed at the same time

Fig. 1. A big data architecture for Industry 4.0. Extracted from Santos et al. (2017)



and can cause a bottleneck between the layers. Finally, because data have to go through different layers, the processing performance is directly impacted.

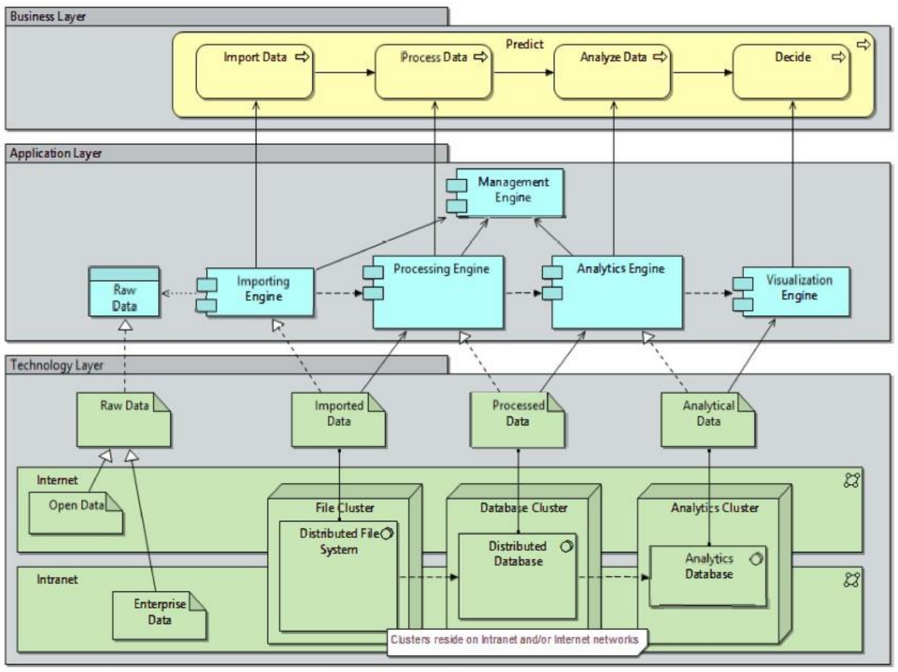
4.2 Shared-repository Pattern

Context: Big data is related to massive volumes of data that must be handled, managed and processed for being used in different industrial systems. These systems must manage a huge amount of data types, which must be shared for providing information for different people and domains.

Problem: In this perspective, preparing big data hosting repository often involves more effort and challenges when such data must be shared among different partners and domains. Shared repositories also increase a control issue related to how and when data will be available in order to be read or written in the shared memory.

Solution: The shared repository pattern consists of using a data repository as communication among different software components crossing domain boundaries. Information produced by a component is stored in the shared repository and other components will retrieve it if they request such data. The repository manages the common data handling application's operation and their permissions to access and modify the repository. Thus, this pattern reduces the communication flows by allowing only requested data, which entails a gain in performance in big data systems. Reuse of data clearly benefits the third party, who gets an information from a common shared repository. The access to the repository enables to check the system evolution and coherence at run time and the evolution and replacement of data is relatively easy. This pattern must be scalable to meet the clients requirements, and it

Fig. 2. A reference architecture for big data solutions. Extracted from Geerdink (2015)



must ensure data consistency. It must handle problems of resource contention, for example by locking accessed data [Avgeriou and Zdun 2005].

Example: In the work presented in [Chen et al. 2015] it was proposed a big data architecture for smart cities focusing on the city data and analytics platform, as illustrated in Figure 2. In this architecture there is a shared repository to communicate to all the architecture components, i.e., components for big data processing, technological modules, the broker that process data generated by several sources, and the server connected to the domain applications. This repository is responsible for receiving requests from all these components and maintain such data available and consistent.

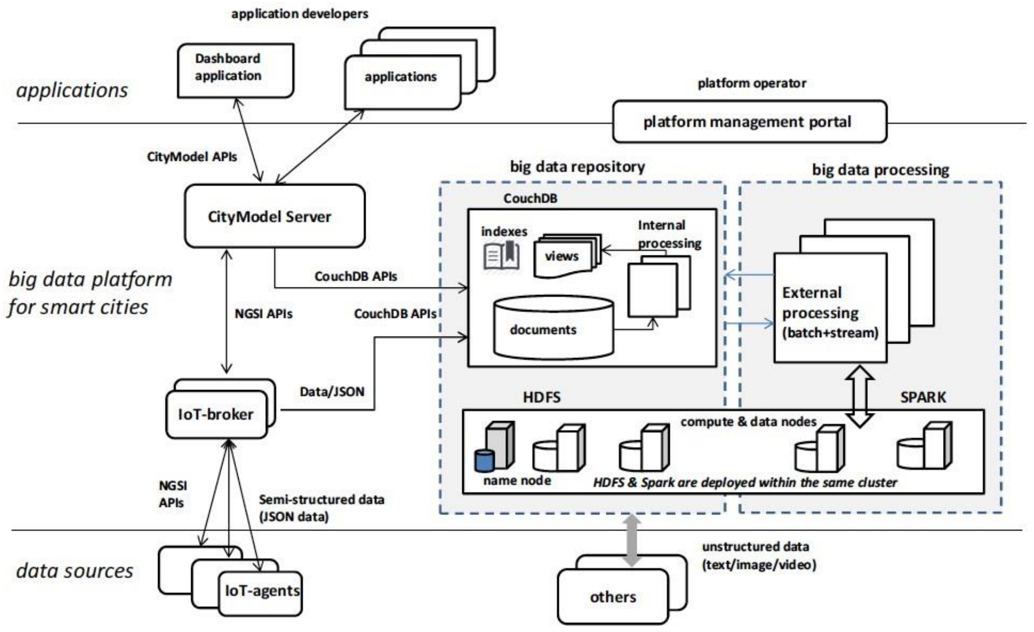
Trade-offs: The main benefits of this pattern include the combination with different storage patterns for improving the way large volumes of data are stored using cloudy and distributed systems. However, shared repository has some drawbacks in the performance and scalability that can be seriously affected in distributed environments by concurrent access to the data. Shared repository transmission can fail and the whole system is immediately affected by losing the synchronization with other services as well as real-time response. In addition, changing the structure of the shared repository has a negative impact on big data systems resulting in modifications in the interfaces and software communication systems.

4.3 Pipe and Filters Pattern

Context: One of the main goals of big data systems is to extract useful value from the massive volume of data generated at real time. For such extraction, data must pass through several steps (e.g., data collection, data integration, pre-processing, processing, and data visualization), which could support the overall data mining process. These steps have to be executed in a specific order to complete the correct data processing.

Problem: In this scenario, big data architectures must support the execution of efficient data flows. The communication between the steps must be scalable, avoiding bottlenecks that could result in failures or delays in

Fig. 3. An architecture for city data and analytics platform. Extracted from Chen et al. (2015)



the following steps. Data must be processed independently in each step, and thus sent to the next to maintain real-time operation. Moreover, security protocols to keep data consistency must be a key concern.

Solution: This pattern structures the processing of a stream of data [Geerdink 2015]. Each processing step is implemented as a filter, with information flowing between the filters through a pipe. Filters can be combined in many ways to provide a family of solutions [Hanmer 2012]. Its importance is justified because the sequence of data analysis is essential, e.g., it is not possible to extract value from data that have not been still pre-processed. This pattern increases complexity and could support data transformation, implying on high dependence between the filters [Buschmann et al. 2007].

Example: [Paakonen and Pakkala 2015] and [Klein et al. 2016] proposed architectures that use pipe and filter pattern to guide data flow, as presented in Figure 4 and 5, respectively. In both approaches data come from an external source or producer and is managed through several steps, such as data collection, preparation, processing, and visualization. Finally, the results of such processing are sent to consumers and domain applications.

Trade-offs: This is one of the closest related pattern to the big data context, because it was proposed to solve data processing issues. However, only such data flow is not sufficient to ensure that there will be quality in performance. Pipe and filters implementation usually achieve scalability and performance but several big data specificities impact its execution and may minimize its benefits.

Firstly, high volume of data is generated at real-time and should be consumed by the filters and processed effectively. However, some of the pipe steps require that the entire dataset be loaded before processing and since there is no alternative way, processing queues are generated, thus impacting scalability, response time, and directly performance [Buschmann et al. 2007; Hanmer 2012].

The veracity of the data can also take risks in this pattern. There are no intermediary sources of input and output between the filters, that is, the data is the input of the first, and the extracted value of the data is the output of the last one. This increase the complexity (i.e., volume, variety) of the data causing some failures in the middle of the

Fig. 4. Pipe and filter pattern in a high-level design of reference architecture. Extracted from Paakonen and Pakkala (2015)

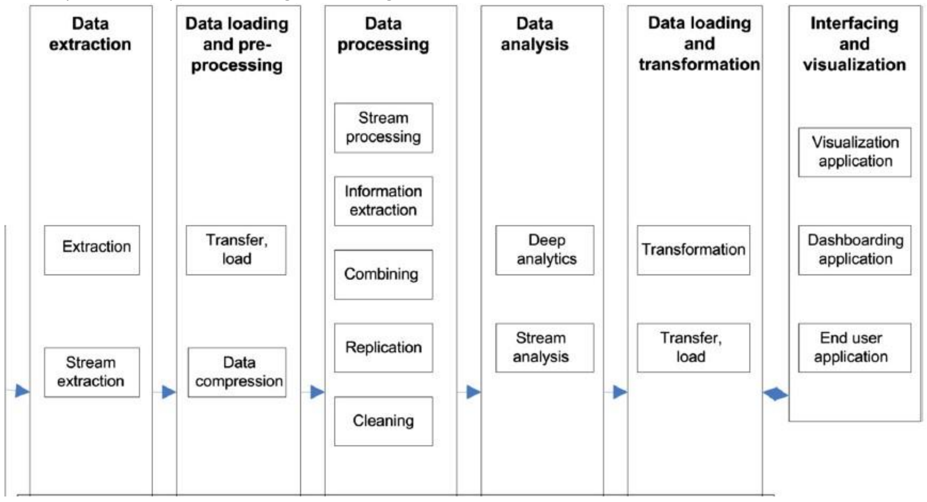
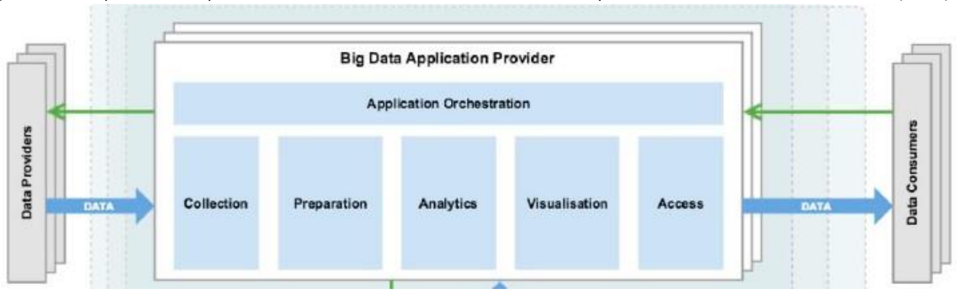


Fig. 5. Pipe and filter pattern in a architecture for the national security domain. Extracted from Klein et al. (2016)



execution, and such failures can be propagated resulting in changes or even data lost [Buschmann et al. 2007]. This indicates a central point of failure and at the same time a liability to safety [Geerdink 2015].

4.4 Broker Pattern

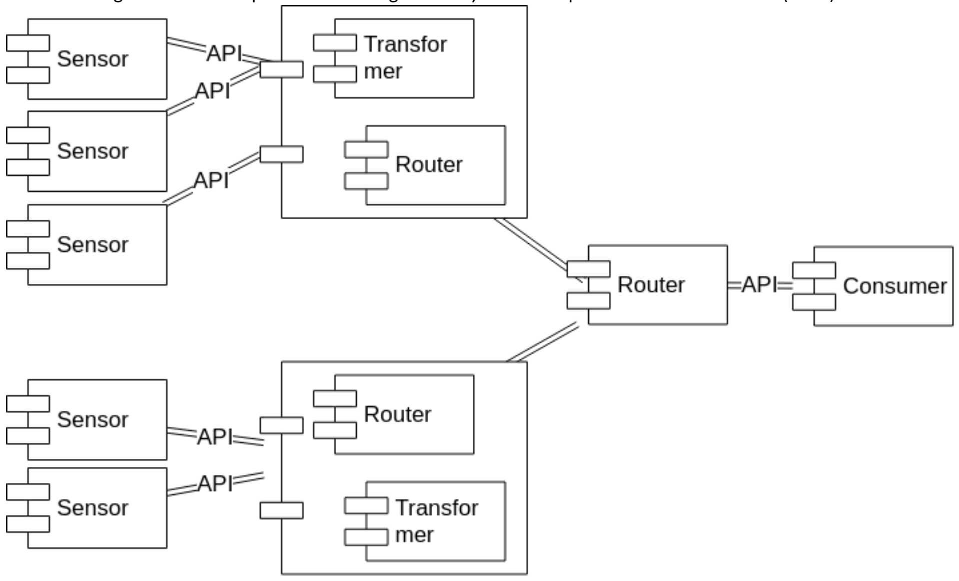
Context: Big data systems must allow transparent communication of heterogeneous, incompatible, inconsistent and undetermined amount of data provided every second by components distributed over a network. Moreover, those systems must integrate a vast amount of data types (e.g., text, images, video, audio) to provide consolidated information to people and other systems, and to execute business processes.

Problem: In this perspective, architectures of Big data systems must be designed to address interoperability of large-amount of distributed and heterogeneous data and processes. Additionally, those architectures must address scalability, that is, those systems must be able to increase and support different amounts of data, processing them uniformly, and allocating resources without impacting costs or efficiency.

Solution: The broker architectural pattern is commonly used in distributed systems to achieve better decoupling of clients and servers. In the context of Big Data systems, this architectural pattern is composed by the following elements:

consumers, providers, routers, and transformers. Routers act as bridges between consumers and providers. Routers must implement interfaces to allow bidirectional and interoperable communications between consumers and providers. Most of the times, each provider has a dedicated router implementing its interface to send data requested by the consumer. Because the heterogeneity of data types, communication protocols,

Fig. 6. Example of broker usage in IoT systems. Adapted from Cecchinel et al. (2014)



programming languages, that are involved in the operation of Big Data systems, the broker architectural pattern includes the translator element to allow interoperability (sometimes executing transformations) between data, protocols, and languages.

Example: A great amount of Big Data systems have used broker architectural patterns for improving interoperability and scalability, such as in IoT systems. Figure 6 depicts an excerpt of a software architecture for allow the collection of Big Data in IoT systems, originally presented in [Cecchinel et al. 2014]. Different sensors send measures in different formats to the broker, which transforms sensors data formats in common formats (e.g., json objects). Further, data aggregation is made and routed to consumers. The router connected to consumers receive normalized data from different providers routers [Cecchinel et al. 2014].

Trade-offs: Interoperability among services is managed and maintained by the broker, which also ensures the flow of data to the correct services and tries to avoid the occurrence of bottlenecks, thus increasing the scalability of the pattern. In addition, with the increase of technologies developed for big data, updates are always necessary. This pattern supports portability by having low coupling services.

However, due to the use of the broker to control the entire data flow, the performance is not as high as it would be if the consumers and providers are connected directly, even more considering the amount of data that is managed. This pattern also implies a central point of failure, because whether complexity causes the broker to fail, the communication between consumer and provider stops.

5. DISCUSSIONS

Big data systems must be designed considering their 5V's intrinsic characteristics of volume, velocity, variety, veracity,and value [Beyer and Laney 2012]. From a perspective of software architecture is important to define requirements of quality attributes that must be overcome by those systems. In Table II, we relate important requirements of quality attributes with the 5V's characteristics of those systems. Requirements were identified and reported in our previous work[Sena et al. 2017]. Scalability, performance and real-time operation are related to the velocity at which data is generated. The system needs to be scalable to handle this velocity, as well as processing this data in real time to avoid bottlenecks and not impact performance. In terms of characteristic of volume, the related attributes are scalability, performance, and modularity. Big Data systems need to adapt to the high and low volumes of data that are generated at run-time. In addition, these data need to be executed with high performance, and if possible, in different modules, to increase the modularity and division of responsibilities. Data veracity has aspects of security and consistency. The data processed must remain consistent throughout the processing flow, and backups must be guaranteed so that data can be recovered if the system fails. In addition, communication between the modules must be secure to avoid data losses and data alteration during this process. The variety of data is addressed by interoperability requirements, which must ensure that data provided by different systems and different formats are processed for value acquisition. Finally, availability is related to value acquisition for ensuring that the generated knowledge and data is available to stakeholders and to execute desired business processes.

In Table II, we summarize the results, mapping the big data characteristics into quality attributes and therefore, indicating which architectural patterns meet specifically the big data requirements.

Table II. Quality Requirements x Architectural Patterns

Characteristics of Big Data Systems	**Quality Attributes – Forces**	**Layered**	**Shared Repository**	**Pipe and Filters**	**Broker**
,Volume, Velocity	*Scalability*	–	–	–	+
Volume, Velocity	*Performance*	–	–	–	–
Volume	*Modularity*	+	–	+	
Veracity	*Consistency*		+	–	
Veracity	*Security*	+	+	–	–
Velocity	*Real-time*	–	–	–	
Variety, Value	*Interoperability*				+
Value	*Availability*		–		

As expected, none of the patters solve directly all big data requirements, mostly because each of them was proposed to contribute for an specific problem. According to the SMS in our previous work[Sena et al. 2017], performance, scalability, and availability are one the most critical requirements for big data systems. Performance and real-time operation are not supported by any of the patterns detailed in this work. This is due to we did not find evidence in studies presented in Table I about how software architectures proposed for big data systems address performance, availability, and real-time requirements. In this perspective, it is of utmost importance to investigate which architectural solutions can be integrated with patterns discussed in this work to successfully address those requirements. The broker pattern supports scalability, but fails to meet others. In the same perspective, modularity, security, interoperability, consistency, and availability are addressed individually by different patters, but at aiming to achiee them, different quality requirements are affected.

The main disadvantage related to patterns offering partial solutions, is that architects must prioritize some requirements more than others (e.g., scalability, performance, security), causing adverse effects between the 5V's characteristics of big data system. Therefore, it is identified that the union of more than one of the patterns could add the benefits of each and then support the design of big data software architectures with more quality. However, such union must be investigated carefully, in order to address the new requirements and guarantee that the crucial requirements are still considered. Moreover, the implementation of each pattern could be adapted in order to support different characteristics. For instance, the communication between filters in the pipe and filters pattern could be implemented to support real-time operation.

It is important to highlight that the information from each architectural pattern presented in this study was based on the data extracted from the SMS. Thus, some important characteristics from patterns that were not clear in the studies from the SMS may not have been extracted, for instance, we found shared repository explicitly twice, but every big data system is almost by definition a shared repository.

6. THREATS TO VALIDITY

Missing of important primary studies:. A threat to the validity is that important studies were not found through the systematic search in databases. To minimize this threat, the result was validated with a control group of studies, which were selected with the support of experts in the domain. Furthermore, we followed a predefined protocol proposed by Kitchenham et al. [Kitchenham and Charters 2007] and Petersen et al. [Petersen et al. 2015] to avoid bias in the study search. Additionally, we searched in seven most important digital databases in software engineering area aiming to find all studies available about software architectures in the context of big data systems.

Study Selection:. As the first author selected the studies by reading each one, it is possible that some studies were erroneously assessed. To mitigate this threat, a set of 100 studies were randomly chosen and the title, abstract, and metadata analysis were again analyzed. After that, the results of both analyses were compared and no difference between them was found.

Data Extraction:. During data extraction, we created a data sheet to fill and save all answers from each study. However, since not all information was explicitly available in the studies, some data had to be interpreted. To minimize this threat, discussions among the authors were developed.

Results Analysis:. The analysis of the results may have been influenced by previous knowledge of the authors and also by the lack of design in several of the identified works. To mitigate this, the authors discussed the extraction of data and architectures to identify patterns that may not have been highlighted by the authors or patterns that were used with different names.

7. LIMITATIONS

We have conducted a literature review eliciting primary studies from 7 digital libraries, namely ACM Digital Library, EI Compendex, IEEE Digital Library, ISI Web of Science, Science Direct, Scopus, and Springer Link. However, with the increasing number of studies related to big data and software architectures, we can not guarantee to have recoverd all relevant material for our research.

Another concern is related to keywords used at searching on architectural patterns. We used relevant keywords to identify primary studies; however, some patterns used different nomenclatures which harnesing their search in data libraries. In addition, some architectures did not provide enough information about used patterns. In this case, we avoided to include information unclear in studies.

Other limitation is concerned about the type of studies retrieved by our systematic review. Most studies were developed in academia and published in relevant journals and conferences related to big data and software architecture. However, we might have missed important studies and research developed by industry, which usually are not published in academic venues. For a complete state-of-the-art analysis of architectural patterns in big data, a survey with industry experts could be conducted to identify which patterns they have used and which attributes are being achieved with such patterns.

Finally, most retrieved studies combined different patterns to achieve domain requirements in the context of big data systems. We conclude, that most patterns were not applied individually in real context. However, a survey with industrial patterns could provide a better accuracy on this statement and clarify how patterns could maximize the quality of software architectures.

8. CONCLUSIONS

The big data context is promising due to the fact that data complexity is increasing, and therefore tends to bring more challenges to software architectures. In this paper, we investigated the applicability of four architectural patterns, namely, layered, pipe and filters, broker, and shared repository to the big data context. Those patterns were identified through a SMS conducted and reported in a previous work [Sena et al. 2017]. Each of them has aspects that can positively or negatively impact Big Data systems. As results, we identified that none of these patterns can successfully meet the expected requirements for big data, since each pattern was conceived to solve an specific problem. Using a specific pattern to achieve some requirements, cause trade-offs between others desired requirements for big data systems. In this context, the integration of patterns to consolidate a software architecture for those systems must be carefully investigated. It is required the preliminary identification of trade-offs to define alternatives to mitigate negative impacts imposed by architectural patterns selected to conform the software architecture of big data systems. This work can be used as a basis for such trade-off analysis.

As main contributions, we presented an analysis of well-known architectural patterns in the context of big data systems, and related the 5V's model with software architectures through the description of requirements of quality attributes, and hence, the description of architectural patterns to address challenges presented in big data systems. Moreover, this study is a first step towards the consolidation of an architectural pattern language for those systems.

As future works, we intend to overcome limitations presented in Section 7. A study on combination of reported architectural patterns in industry will be conducted. Additionally, if necessary, new patterns will be investigated, aiming at supporting the design and documentation of software architectures for big data systems.

Acknowledgments

The authors would like to thank the Brazilian funding agency FAPESP (Grants: 2018/07437-9, 2017/22237-3, 2016/05919-0, and 2017/06195-9). Our greatest thank and gratitude to our shepherd, Neil Harrison, for his valuable suggestions and comments to improve this paper. Thanks also to all members of the PLoP'18 writers workshop, their feedback offered new valuable insights.

REFERENCES

Jameela Al-Jaroodi and Nader Mohamed. 2016. Characteristics and Requirements of Big Data Analytics Applications. In *2nd International Conference on Collaboration and Internet Computing, (CIC)*. IEEE Computer Society, Pittsburgh, PA, USA, 426–432.

Christopher Alexander. 1977. *A pattern language: towns, buildings, construction*. Oxford university press.

Paris Avgeriou and Uwe Zdun. 2005. Architectural patterns revisited-a pattern language. (2005), 1–39.

L Bass, P Clements, and R Kazman. 2013. *Software Architecture in Practice*. Vol. 3. Addison-Wesley Pearson Education, Boston.

Edmon Begoli and James Horey. 2012. Design principles for effective knowledge discovery from big data. In *6th Joint Working Conference on Software Architecture and European Conference on Software Architecture, (WICSA/ECSA)*. IEEE, Helsinki, Finland, 215 – 218.

Mark A Beyer and Douglas Laney. 2012. The importance of big data: a definition. *Stamford, CT: Gartner* (2012), 2014–2018.

Frank Buschmann, Kelvin Henney, and Douglas Schimdt. 2007. *Pattern-oriented Software Architecture: on patterns and pattern language*. Vol. 5. John wiley & sons.

Cyril Cecchinel, Matthieu Jimenez, Sebastien Mosser, and Michel Riveill. 2014. An Architecture to Support the Collection of Big Data in the Internet of Things. In *IEEE World Congress on Services (SERVICES)*. IEEE, Anchorage, Alaska, USA, 442–449.

Hong-Mei Chen, Rick Kazman, Serge Haziyev, and Olha Hrytsay. 2015. Big data system development: An embedded case study with a global outsourcing firm. In *Proceedings of the First International Workshop on BIG Data Software Engineering*. IEEE Press, 44–50.

Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer, and Ernoe Kovacs. 2015. Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander. In *IEEE International Congress on Big Data (IEEE BigData)*. IEEE, Santa Clara, CA, USA, 592–599.

Paul Clements and Len Bass. 2010. *Relating business goals to architecturally significant requirements for software systems*. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

A. Cuzzocrea. 2016. A Reference Architecture for Supporting Secure Big Data Analytics over Cloud-Enabled Relational Databases. In *40th Annual Computer Software and Applications Conference, (COMPSAC)*. IEEE Computer Society, Atlanta, GA, USA, 356–358.

Umeshwar Dayal, Masaharu Akatsu, Chetan Gupta, Ravigopal Vennelakanti, and Massimiliano Lenardi. 2014. Expanding global big data solutions with innovative analytics. *Hitachi Review* 63, 6 (2014), 333–339.

Yuri Demchenko, Fatih Turkmen, Cees de Laat, Ching-Hsien Hsu, Christophe Blanchet, and Charles Loomis. 2017. Cloud Computing Infrastructure for Data Intensive Applications. In *Big Data Analytics for Sensor-Network Collected Intelligence*, Hui-Huang Hsu, Chuan-Yu Chang, and Ching-Hsien Hsu (Eds.). Academic Press, Taiwan, 21–62.

Bas Geerdink. 2015. A reference architecture for big data solutions - introducing a model to perform predictive analytics using big data technology. *International Journal of Big Data Intelligence (IJBDI)* 2, 4 (2015), 236–249.

Ian Gorton and John Klein. 2015. Distribution, data, deployment: Software architecture convergence in big data systems. *IEEE Software* 32, 3 (2015), 78 – 85.

Ian Gorton, John Klein, and Albert Nurgaliev. 2015. Architecture Knowledge for Evaluating Scalable Databases. In *12th Working IEEE/IFIP Conference on Software Architecture, (WICSA)*. IEEE Computer Society, Montreal, QC, Canada, 95–104.

Robert Hanmer. 2012. *Pattern-oriented software architecture for dummies*. John Wiley & Sons.

Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. 2015. The rise of big data on cloud computing: Review and open research issues. *Information Systems* 47 (2015), 98–115.

Leonard Heilig and Stefan Voß. 2017. Managing Cloud-Based Big Data Platforms: A Reference Architecture and Cost Perspective. In *Big Data Management*. Springer International Publishing, Cham, Switzerland, 29–45.

Claire Ingram, Richard Payne, and John Fitzgerald. 2015. Architectural Modelling Patterns for Systems of Systems. In *INCOSE International Symposium*, Vol. 25. Wiley Online Library, 1177–1192.

Anil Jain. 2016. The 5 Vs of Big Data. https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/, (2016).

Stephen Kaisler, Frank Armour, J Alberto Espinosa, and William Money. 2013. Big data: Issues and challenges moving forward. In *System sciences (HICSS), 2013 46th Hawaii international conference on*. IEEE, 995–1004.

Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Technical Report. Keele University and Durham University Joint Report. 57 pages.

John Klein, R. Buglak, D. Blockow, T. Wuttke, and B. Cooper. 2016. A Reference Architecture for Big Data Systems in the National Security Domain. In *2nd International Workshop on BIG Data Software Engineering, (BIGDSE@ICSE)*. ACM, Austin, Texas, USA, 51–57.

Alexandra L heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. 2017. Machine learning with big data: Challenges and approaches. *IEEE Access* 5 (2017), 7776–7797.

Doug Laney. 2001. 3D data management: Controlling data volume, velocity and variety. *META Group Research Note* 6, 70 (2001).

Buschman Frank R Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, and Frank Buschmann. 1996. Pattern Oriented Software Architecture: A System of Patterns. *Chichester et al* (1996).

Pekka Paakonen and Daniel Pakkala. 2015. Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems. *Big Data Research* 2, 4 (2015), 166 – 186.

Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information & Software Technology* 64, C (2015), 1–18.

R.S. Pressman and D. Bruce R. Maxim. 2014. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.

Mark Richards. 2015. *Software architecture patterns*. O'Reilly Media, Incorporated.

Maribel Yasmina Santos, Jorge Oliveira e Sá, Carlos Costa, João Galvão, Carina Andrade, Bruno Martinho, Francisca Vale Lima, and Eduarda Costa. 2017. A Big Data Analytics Architecture for Industry 4.0. In *Recent Advances in Information Systems and Technologies (WorldCIST)*. Springer, Porto Santo Island, Madeira, Portugal, 175–184.

Bruno Sena, Ana Paula Allian, and Elisa Yumi Nakagawa. 2017. Characterizing big data software architectures: a systematic mapping study. In *Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse*. ACM, Fortaleza, Brazil, 1–10.

Purnima Shah, Deepak Hiremath, and Sanjay Chaudhary. 2016. Big Data Analytics Architecture for Agro Advisory System. In *23rd International Conference on High Performance Computing Workshops, (HiPCW)*. IEEE Computer Society, Hyderabad, India, 43–49.

Muhammad Fahim Uddin, Navarun Gupta, and others. 2014. Seven V's of Big Data understanding Big Data to extract value. In *American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the*. IEEE, 1–5.

Phillip Viana and Liria Sato. 2015. A proposal for a reference architecture for long-term archiving, preservation, and retrieval of big data. In *13th International Conference on Trust, Security and Privacy in Computing and Communications, (TrustCom)*. IEEE Computer Society, Beijing, China, 622–629.