

The state of big data reference architectures: a systematic literature review

Abstract

This template helps you to create a properly formatted L^AT_EX manuscript.

Keywords: `elsarticle.cls`, L^AT_EX, Elsevier, template

2010 MSC: 00-01, 99-00 s

1. Introduction

The rapid development of software technologies, the proliferation of digital devices and networking infrastructure of today, have by and large, augmented user's capability to generate data [1]. In the age of information, users are
5 unceasing generators of structured, semi-structured, and unstructured data that if collected and crunched correctly, may reveal game-changing patterns [2].

The unprecedented proliferation of data have emerged a new ecosystem of technologies; one of these ecosystems is big data (BD)[3]. BD is a term emerged to describe large amount of data that comes in various forms from different
10 channels. Within the years, BD has attained a lot of attention from academia and industry, and many strive to benefit from this new material. Howbeit, adopting BD requires the absorption of great deal of complexity and many traditional systems cannot cope with characteristics of this domain.

A recent survey published by Databricks in partnership with MIT Technol-
15 ogy Review Insights, stated that only 13% of companies excel at delivering on their data strategy [4]. In the same vein, Vintage Partners highlighted that only 24% of companies have successfully adopted BD [5]. Sigma computing report presented that 1 in 4 business experts have given up on getting insights they needed because the data processing took too long [6]. Moreover, Gartner

20 approximated that only 20% of companies have successfully adopted BD.

Some of the most highlighted challenges of BD is 'lack of business context', 'organizational challenges', 'BD architecture', 'data engineering', 'rapid technology change', and 'lack of talent' [7]. Whereas similar issues may exist in other domains, it is exacerbated when it comes to BD systems. This is due the
25 inherent complexity of BD engineering, the need for real-time processing, the scalability requirement of these systems, and the sensitivities around data.

Today, majority of BD systems are designed underlying ad-hoc and complicated architectural solutions [8], that do not seem to adhere to similar patterns. This will challenge software architects to design a suitable solution for any given
30 context, creates a foundation for an immature architectural decision, and does not promote the growth and development of BD systems as a whole.

Therefore, since the approach of ad-hoc design to BD systems is undesirable and leaves many engineers in the dark, there is a need for more software engineering research for BD systems. To this end, this study presents a systematic
35 literature review (SLR) on BD (BD) reference architectures (RAs).

2. Why reference architectures?

Conceptualization of the system as an RA, helps with understanding of the system's key components, behavior, composition and evolution of it, which in turn affect quality attributes such as maintainability, scalability and performance [9]. Therefore RAs can be a good standardization artefact and a commu-
40 nication medium that not only results in concrete architectures for BD systems, but also provide stakeholders with unified elements and symbols to discuss and progress BD projects.

This approach to system development is not new to practitioners of complex
45 system. In software product line (SPL) development, RAs are utilized as generic artifacts that are instantiated and configured for a particular domain of systems [10]. In software engineering, IT giants like IBM have referred to RAs as the 'best of best practices' to address complex and unique system design challenges

[9]. In other international standardization, RAs have been repeatedly used to
50 standardize an emerging domain, a good example of this is BS ISO/IEC 18384-1
RA for service oriented architectures [11].

3. Reference Architectures State of the art

Despite the benefits of utilizing RAs, and their potential to solve some of the
complex issues of BD systems, we think that this area is underdeveloped and
55 needs more attention from both academia and practice. This insight is derived
from our preliminary systematic review in academia, and a search for available
big data RAs ([2]).

One of the most comprehensive BD RA published, is the National Insti-
tute of Standards and Technology (NIST) BD RA, which published by Big
60 Data Public Working Group (NBD-PWG) with large set of contributors from
academia, industry, non-profit organizations, and others. This RA was an ini-
tiative from White house in March 2012, and was published in October 2019
with considerable amount of investment.

Given the substantial investment on BD RAs, one might infer the value of
65 these artifacts, and this can in turn highlights the necessity for more research
in this domain. Another factor that came to light, is how vaguely the phrase
'reference architecture' is defined and institutionalized. For instance, the differ-
ence between a 'concrete architecture' and a 'reference architecture' is hardly
discussed, and different domains seem to have defined the artifact slightly dif-
70 ferently.

For instance, Cloutier et al ([9]) presented that 'Reference Architectures
capture the essence of existing architectures, and the vision of future needs and
evolution to provide guidance to assist in developing new system architectures'.
This definition is derived from the system engineering domain and by the means
75 of collaborative forum from Steven's institute of technology. In another effort,
Muller et al ([12]) defines RA as 'artifacts that captures the essence of architec-
ture of a collection of systems. This definition is driven from the product line

engineering domain’.

In the same vein, Angelov et al ([13]) proposed that ‘A reference architec-
80 ture is a generic architecture for a class of information systems that is used as
a foundation for the design of concrete architectures from this class’. Although
different authors may have defined RAs with different syntax, the essence re-
mains the same: to reuse the software engineering knowledge for a class of
systems, particularly in relation to architecture.

85 Moreover, the difference between RAs and concrete architectures is rarely
discussed. Another definition by Bass et al ([14]) stated that ‘A reference archi-
tecture is a reference model mapped onto software elements (that cooperatively
implement the functionality defined in the reference model) and the data flows
between them’.

90 4. Objective of the study

Given the failure rate of BD projects, we posit RAs as potential solution to
facilitate system development and BD architecture, and aim to explore this area
through a systematic literature review. Up to date, there’s only one SLR that
explored this area ([2]), which is outdated, suffers from methodological clarity,
95 and is published as a conference paper, which implies lack of detail.

Based on this, the objective of this review is to find and collate the BD RAs
available from the body of evidence, highlight their architectural commonality
and point out the limitations. This study can be considered a useful primer for
practitioners or academics who are interested in adopting BD.

100 The research questions are formulated as the following;

1. **RQ1:** What are current BD RAs available in academia and industry?
2. **RQ2:** What are major architectural components of these BD RAs?
3. **RQ3:** What are the limitations of current BD RAs?

We found these questions relevant to our study firstly because they address a
105 gap in the body of knowledge (BD RAs), secondly because they aim to capture

the essence of practice on architecting big data systems and lastly because they shed lights on limitations of current BD RAs and common choke points.

5. Review Methodology:

This research follows the guidelines of PRISMA ([15]). In addition, we adopted PRISMA-S ([16]) to improve our search strategy and lastly we have used Barbara et al's guidelines for evidence based software engineering and systematic reviews [17]. Although PRISMA is a comprehensive guidelines on conducting a systematic literature review, it is derived from the healthcare community and sometimes makes assumptions that may not be relevant to software engineering and information system researchers. Barbara et al [17] has translated many of these assumptions to the domain of software engineering and included many guidelines for lone researchers and projects with small number of researchers.

We have therefore utilized PRISMA as the underpinning of our research design, with complementary studies to reduce bias, improve transparency and systematicity. SLR has been chosen because it is a qualitative research methodology that is aimed at driving knowledge and understanding about the subject matter and the elements surrounding it. Besides, SLR provides a transparent and reproducible procedure that elicits patterns, relationships, trends, and delineates the overall picture of the subject [18].

The main objective of this study is to assess the current state of BD RAs, identify their major architectural components, point out fundamental concepts and discuss their limitations. This objective is achieved in four phases. In first phase, research questions are stated, literature are identified and pooled, exclusion and inclusion criteria are defined, and the quality framework is developed. In second phase, the title of the studies are assessed based on the inclusion and exclusion criteria. After that, the filtered studies are once more assessed based on their title, abstract, introduction and conclusion. After this, full analysis of the studies took place by running each study against the criteria defined in

135 the quality framework. Thirdly, selected pool of literature is coded based on
research questions. Lastly, findings are synthesized by the means of thematic
synthesis, and themes realized are depicted.

This study builds on the SLR conducted by Ataei et al [2] and aims to
improve it by covering the years 2020 to 2022. Unlike Ataei’s work, this paper
140 aims to employ thematic synthesis, and provide a more detailed view of BD
RAs and their properties.

5.1. Identification

The first phase of the SLR began, by adoption of PRISMA-S ([16]) to develop
a robust multi-database search strategy. This extension of PRISMA provided
145 us with a framework of 12 items to increase transparency, systematicity, and
reduce bias. For the purposes of this study, following electronic databases were
searched: ScienceDirect, IEEE Explore, SpringerLink, AISel, JSTOR and ACM
library. To pursue to goal of finding all literature available on the topic, and to
avoid overlooking valuable research, abstract and citation databases and search
150 engines such as Google Scholar, and Research Gate was used.

We also searched the grey literature on the topic, using the search string
"big data" AND "reference architecture*" on Google (in June 2022). The first
40 results were selected for screening. This was done in 'incognito mode' to
avoid any personal customization of the google search pages. Reference lists of
155 included studies were manually screened to identify additional studies. This is
to achieve the critical component of 'completeness' as suggested by Kitchenham
et al [17].

The platform search capabilities varied, but our search strategy remained
uniform for most parts. For instance, if a platform did not support wildcards (
160 like asterisk), we just searched twice for the singular and plural version of the
word. The only exception that made the selection process longer was Springer-
Link, because it did not support bulk download of references in BibTex format.
The reproducible search for the chosen databases is as follows:

- ("Document Title":big data) AND ("Document Title":reference architecture) OR ("Document Title":big data architecture)

The reason we included architecture is due to the fact that terms *reference architecture* and *architecture* may have been used interchangeably, and an architecture that is at the abstraction level of an RA, might have been called just an architecture. Therefore it was critical for us to firmly define these terms and then categorize studies based on these definitions. These definitions and our findings are depicted in the findings section.

Our initial search was limited to year 2020 to year 2022, as the work of Ataei et al [2] covered the years 2010-2020. Nevertheless, we still included the years 2010 to 2020 to make sure no research is left out or overlooked. These years are chosen firstly because more contemporary researches are focused on the facilitation of big data system development, and secondly there's no SLR that has covered these years for BD RAs.

To achieve these limits, we have utilized databases features. All databases supported the selection of year range, and the language limit was automatically applied by doing an advanced search with the aforementioned keywords.

Our approach to systematic collection of evidence was to search databases using the keywords aforementioned and then bulk download the BibTex files. Majority of the databases supported bulk downloading of BibTex files except for SpringerLink, Google Scholar, and Research Gate. For SpringerLink we downloaded the studies in CSV format and then converted them to BibTex using a custom script. For Google Scholar and ResearchGate, unfortunately, we had to take the manual path of creating a bib file for the studies.

Once all the bib files have been created, we merged them into one large bib file and imported it to a software called JabRef ([19]) for deduplication. studies are pooled initially, out of which 6 duplicates have been identified. We removed the primary SLR that this study is based on, and also another paper that we could not find the citation for. In the other extreme, we found 5 white papers and 4 website blogs and added them to the selection pool. At the end

of this phase, 173 studies have been pooled.

195 5.2. Screening and Eligibility

Stage 1 of screening started with assessing the title, abstract, and keywords of the pooled studies. For grey literatures simply the title. This was achieved based on our inclusion and exclusion criteria. The inclusion criteria are as following;

- 200 • Primary and secondary studies (including grey literature) between Jan 1st 2010 and June 1st 2022 on the topics of BD RAs, BD models, and BD architectural components were included.
- Research that Indicates the current state of RAs in the field of BD and demonstrates possible outcomes
- 205 • Studies that are scholarly publications, book, book chapter, thesis, dissertation, or conference proceedings
- Grey literature such as white paper that includes extensive information on BD RAs

And the studies with the following topics were excluded:

- 210 • Informal literature surveys without any clearly defined research questions or research process
- Duplicate reports of the same study (a conference and journal version of the same paper)
- Short papers (less than 5 pages)
- 215 • Studies that are not written in English

Disagreement among researchers were resolved using Krippendorff's alpha ([20]). Our aim was not to get involved in a very complicated statistics model, so we've done most of the computations using SPSS, specifically with Hayes'

Macro. We made sure that a separate file is created for each variable, and
220 inserted coders as variables and not a constant value. Our α value was within
the acceptable range (above 80), and any disagreement was solved by inviting
a third person or a moderator. When α value was very low (indicating a low
reliability), we stopped the process, and tried to clarify fundamental concepts
and categories. The final computed α value was 89.9%.

225 In stage 2, After excluding papers based on inclusion and exclusion criteria,
and as suggested by Kitchenham et al [17], we assessed studies based on their
quality. Quality of the evidence collected as a result of this SLR has direct
impact on the quality of the findings, making quality assessment an important
undertaking.

230 However, this process comes with some well-known complexities. The most
fundamental ones are perhaps firstly defining the term 'quality', and secondly
trying to appraise the quality of conference papers that rarely provide enough
detail on research methodology and evaluation. Generally, a quality of a study
is tightly associated to its research method and the validity of its findings.
235 From this perspective, and inspired by the works of Noblit and Hare on meta-
ethnography ([21]), and Dyba et al ([22]), quality of studies is assessed by the
extent to which the conduct, design and analysis of a research is susceptible to
systematic errors or bias ([23]). That is, the more bias in the selected literature,
the more chance to create miss-leading conclusions.

240 Considering the rather heterogeneous nature of software engineering and
information systems (IS) papers, and difficulty of defining quality in studies with
varying nature, we first analyzed a few well-established checklists such as Critical
Appraisal Skills Programme (CASP [24]), and JBI's critical appraisal tool ([25]).
Whereas these checklists could potentially account for the requirements of this
245 study, we opted for something that is more specific to software engineering
and IS. We realized for example that, Runeson et al ([26]) provided a checklist
designated to help researchers reading and undertaking software engineering
case studies. In the same vein, Dyba et al ([22]) proposed a quality criteria based
on CASP checklist for qualitative studies in software engineering systematic

250 reviews.

Nevertheless, the challenge is that our study includes a large number of different study types that needs to go through a single checklist. To address this, we developed a criteria made up of 11 elements. These criteria are informed by those proposed by CASP for assessing the quality of qualitative research ([24]) and by guidelines provided by Kitchenham ([27]) on empirical research in software engineering. The 7 criteria tested literature on 4 major areas that can critically affect the quality of the studies. These categories and the corresponding criteria are as following;

1. *Minimum quality threshold:*

- 260 (a) Does the study report empirical research or is it merely a 'lesson learnt' report based on expert opinion ?
- (b) The objectives and aims of the study is clearly communicated, including the reasoning for why the study was undertaken ?
- (c) Does the study provide with adequate information regarding the context in which the research was carried out ?
- 265

2. *Rigour:*

- (a) Is the research design appropriate to address the objectives of the research ?
- (b) Is there any data collection method used and is it appropriate ?

270 3. *Credibility:*

- (a) Does the study report findings in a clear and unbiased manner ?

4. *Relevance:*

- (a) Does the study provides value for practice or research

Taken all together, these 7 criteria gave us a measure of the extent to which a particular study's findings could make a valuable contribution to the review. These criteria was disseminated as a checklist among researchers with value for each property being dichotomous, that is 'yes' or 'no' in two phases. In the first phase, researchers only assess the quality based on the first major area (minimum quality threshold). If the study passed the first phase, it would

275

280 then go into the second phase, where it was assessed for credibility, rigour and
relevance. The quality is agreed if 75% of the responses are positive for any
given study with at least 75% inter-rater reliability.

Disagreements regarding the quality was usually resolved through a meeting.
While, the meeting could not address the disagreements, a moderator has been
285 invited to the process. Lastly, it is worth mentioning that this quality framework
was not used for grey literature. Grey literature were only assessed through
inclusion and exclusion criteria.

In the first phase (identification) of this SLR, a total of 138 literature has
been pooled from academia, and 24 from grey literature. Some of this literature
290 has been added to the pool by the process of forward and backward searching.
For instance, by reading NIST RA, we found out about Oracle, Facebook, and
Amazon RAs and included those in the pool of the literature as well.

In the screening phase, the literature that were not in-line with our inclusion
and exclusion criteria have been eliminated. For example, if the paper was very
295 short and was not on the topic of BD RA, or its ecosystem or limitations, it
was excluded. As a result of this phase, 50 papers excluded. In the next phase,
by assessing studies against the quality framework, 21 studies from academia,
and 12 studies from grey literature pool has been eliminated. The flowchart is
depicted in figure 1.

300 By the result of this work, 79 articles have been selected comprising of pro-
ceedings, journal articles, book chapters, and white papers. Out of the pool of
articles, 33.3% are from IEEE Explore, 5.2% from ScienceDirect, 24.5% from
SpringerLink, 15.7% from ACM, and 21% from other sources such as Google
Scholar and Research Gate. 30 journal articles, 29 conference proceedings, 12
305 book chapters, 6 white papers, 1 Master's Thesis and 1 PhD thesis were selected.
55% of the articles were selected from the years 2016- 2022, 33% belonged to
years 2013-2016, and the rest to years 2010-2013. These stats are portrayed in
figure 2

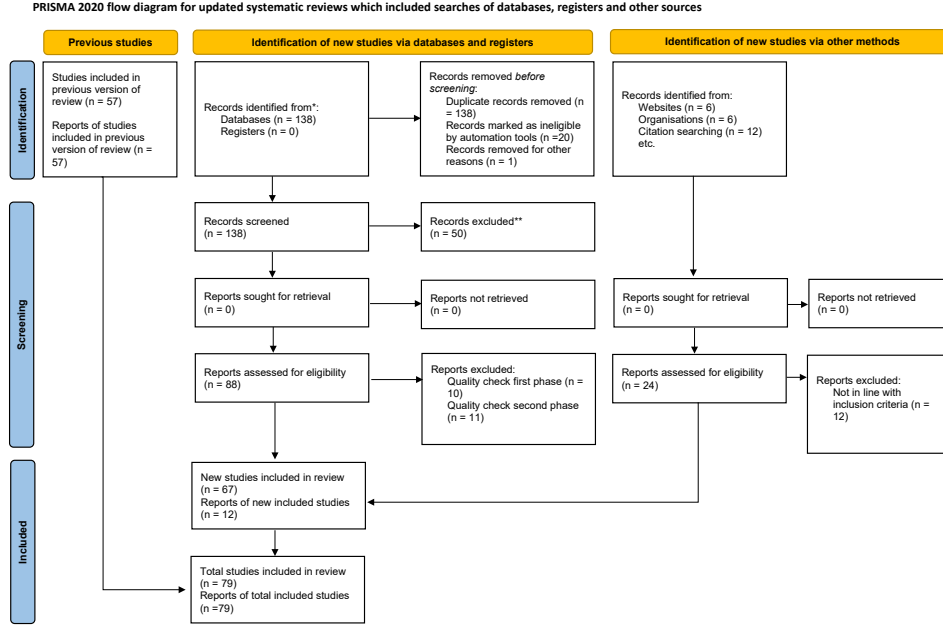


Figure 1: PRISMA flowchart

5.3. Data Extraction and Synthesis

By this stage, research questions have been set, inclusion and exclusion criteria are defined and applied, the quality assessment framework is developed and applied to the pool of studies, and the research embarked on actual synthesis of data. An integral element of this phase is data extraction, in which the essence of the studies are obtained in an explicit and consistent manner.

Precursor to synthesis of the actual data, we first followed the guidelines proposed by Dyba et al [28] for data extraction. Data extraction firstly began by reading the entire pool of literature in order to get immersed with the data [29]. From there on, we followed a structured reading approach and extracted three kind of data; 1) Publication Details (author, title, year, etc), 2) Contextual descriptions (industry, settings, technologies), and 3) Findings (results, the actual RA, events, etc ..)



Figure 2: SLR Statistics

This process was a bit challenging, as some studies did not describe the method adequately, contextual information were not detailed often, and evaluation methods varied. To overcome this challenge, majority of this process took place in a consensus meeting [30].

After data extraction, we began the coding process. For this step, we've had several approaches ahead of us. Either we could adopt a deductive or a prior approach ([31]) or an inductive or Grounded Theory approach ([32]). Neither of which could be as rigorous as we desired, thus we opted for an integrated approach ([33]). We used the software Nvivo to organize our files and created an initial set of a priori codes based on research questions. These codes are as followings;

1. BD RAs (RQ1)
2. BD RAs Architectural components (RQ2)
3. BD RAs limitations (RQ3)

As the coding progressed, we realized that there is a need to define some of the fundamental areas that seem to not have been well established in academia and practice. For instance, we've been looking for a comprehensive data to discuss the fundamental concepts of RA to further support our initiative, but
340 this was not standardized, and while there was mention of these concepts, they were usually lacking or very short. Furthermore, not many studies discussed the benefits and relevance of RAs for BD systems. We also could not find a study that thoroughly discusses common approaches to developing BD RAs, and the challenges of developing a BD RA.

345 Based on these, therefore, we added the following extra four codes;

1. Fundamental concepts of RAs
2. How can RAs help BD system development
3. Common approaches to creating BD RAs
4. Challenges of creating BD RAs

350 After having coded all the literature pooled, we began the process of turning them into themes. Themes helped us pull together segregated data into one meaningful whole that is above the sum of its constituents. This was not a single step process, and as we started to analyze codes, we have subsumed some first-cycle codes into higher-order codes. This also led to rearrangements and
355 reclassification of the codes. The end of this process was marked, when the emerging themes saturated, and we could not derive a new theme. Many of the themes emerged have been then categorized into higher-order themes.

The last step of data synthesis, was creation of a model based on the higher-order themes to explain relationships and to answer original research questions.
360 The final product of this phase, is a theory, connection with prior theories, and indication of relationships.

Of particular challenge we faced in this phase was the influence of heterogeneity, specifically given the inclusion of grey literature and cardinality of research methodologies in software engineering researches. Thus, to ensure the robustness of the higher-order themes we identified the main sources of variability as;
365

1) variability of outcomes (some RAs well evaluated in practice, while some other are just compared against other RAs), 2) variability in study designs (methodological diversity that exists in software engineering and specifically creation of RAs), and 3) variability in study settings (contextual factors are often not well reported).

Last, but not least, to increase the rigour, we assessed the trustworthiness of the synthesis from three aspects; 1) Credibility: is the focus of the research in-line with research questions, and does the thematic synthesis cover data well ? 2) Conformability: are data extracted and coded in the correct way? do all researchers agree on this? would readers agree with the approach ? 3) Transferability: are the findings generalizable, can the findings be applied in different context?

6. Findings

In this section, we map our findings against the research questions in a series of sub-sections. For increased clarity, these sub sections are driven by the research questions and models we created in the previous phase. We first begin by explaining fundamental concepts such as RAs and how they help BD system development (our deductive codes) and then progressively worked towards more specific topics such as current BD RAs and their limitations.

6.1. *What are the fundamental concepts of RAs?*

As the complexity of man-made systems grow, procedures, principles, and concepts of software architecture are increasingly applied to address those complexity faced by practitioners [2]. A system abstracted and expressed in terms of architectural concepts, facilitates the understanding of system's essence, properties revolving around it, and evolution of it, which in turn affects quality attributes such as performance, maintainability, and scalability.

In recent years, IT architectures played a pivotal role in the progress and evolution of system development and gained acceptance in maintenance, planning,

development, and cost reduction of complex systems [34]. To address ambiguity
395 about what should be developed to address what needs, an architecture can play
an overarching role by portraying the fundamental components of the system
and the means and ways in which these components communicate to achieve
the overall goal of the system [35]. This in turn creates manageable components
that can be used to address different aspect of the problem and provides stake-
400 holders with an abstract artefact to observe, reflect upon, contribute to, and
communicate with [36]

Many successful IT artefacts today stemmed from an effective RA. A few
good examples are the Open Systems Interconnection model or OSI [37], Open
Authentication or OATH [38], Common Object Request Broker Architecture or
405 CORBA [39], and WMS or workflow management systems [40]. In fact, every
system goes with an architecture, either known or unknown, and it is in the
architecture that the overall qualities of the system are defined

Whereas there are various definitions to what constitutes an RA, they all
share the same principle that the concept of patterns plays a significant role.
410 Some studies have defined RAs as “a predefined architectural pattern, or set
of patterns, possible, partially or completely instantiated, designed, and proven
for use in particular business and technical contexts, together with supporting
artifacts to enable their use” [9]. In Software Product Line (SPL) development,
RAs are defined as generic schema that can be instantiated and configured for
415 a particular class of systems [10].

In software engineering, RAs can be defined as an artefact that transfers
software engineering knowledge as a family of solutions to a problem domain
[41]. In another terms, RAs are artefacts that embody domain relevant concepts
and qualities, break down solutions and create a ubiquitous language to facilitate
420 effective communication, and inform various stakeholders.

Taking all into consideration, and based on the model derived from our
thematic synthesis, five major concept of RAs are identified as the following;

1. **RAs are at the highest level of abstraction:** RAs aim to capture the

essence of the practice as an abstraction that portrays elements necessary
for communication, standardization, implementation and maintenance of
certain class of systems. Hence, RAs aim to inject software engineering
knowledge as a set of high-level architectural patterns and do not provide
implementation details such as specific frameworks, vendors or environ-
ments. This makes RAs sit at a higher level of abstraction comparing to
concrete architectures.

2. **RAs emphasize heavily on architectural qualities:** RAs, sitting at
a higher level of abstractions are artifacts created for a wider audience and
a bigger context, and are usually used by solution architects to deduce a
concrete architecture in a specific environment ([42], [43]). As a result,
RAs pay more attention to architectural qualities.
3. **In RAs, stakeholders are not clearly defined:** Stakeholders are usu-
ally people of the same company involved in the actual design and im-
plementation of the system and do get involved in the product creation
in various phases. Different stakeholders have different concerns and are
crucial to the creation of the overall product [44]. A stakeholder can be
a developer, a designer, a product owner, a data scientist or a business
analyst. Notwithstanding, due to the generic nature of the RAs, it is not
feasible to include all stakeholders a priori. RAs are at a higher level of
abstraction and tend to provide a generic solution for a class of problems,
not a specific context. Therefore, defining and introducing stakeholders
into RAs can potentially decrease their effectiveness ([2], [45]).
4. **RAs promote adherence to common standards:** The design of an
RA is usually guided by existing architectural patterns, common pitfalls
in practice, the body of literature and models. For this reason, RAs con-
vey standard approaches and patterns that avoid known pitfalls, facilitate
reuse, and decrease complexity.
5. **5. RAs are effective artefacts for system development and com-
munication:** RAs are powerful artefacts that can be used by architects
that design, manage, and utilize complex system. Because RAs are created

455 as artifacts that codify the best practice and conventions of the industry
and often include architectural descriptions and standards, they can be
deemed effective artifacts for system development and communication.

7. How can RAs help BD system development?

Despite the high failure rate of BD projects, IT giants such as Google, Face-
460 book or Amazon have developed exclusive BD systems with complicated data
pipelines, data management, procurement and batch and real-time analysis ca-
pabilities [36]. Having the resources required, these companies attract the best
of talent from around the globe to manage the complexity involved in develop-
ment of big data systems. Notwithstanding, that's not the reality of majority
465 of organizations that are trying to benefit from big data analytics.

Big data systems sail away from traditional small data analytics paradigms
and bring various challenges including rapid technology change challenges [46],
system development and architecture challenges [47], and organizational chal-
lenges [3]. Moreover, big data systems are distributed in nature and need to
470 account for various kind of data processing usually batch and stream process-
ing. This combined with the complexity of maintaining and scaling data qual-
ity, metadata, data catalogs, data dimension modeling, and data evolvability,
makes big data system design a daunting task.

BD does not only mean 'big' amount of data, or just volume; other charac-
475 teristics of BD such as velocity, variety, veracity and variability bring significant
challenges to the practice. Although these challenges do not only belong to BD
systems, BD exacerbates these challenges because of the following reasons;

1. Distributed scaling is required to address batch and stream processing
demands
- 480 2. There is a need for real near-time performance (stream processing)
3. Complex technology orchestration is required to create effective commu-
nication channels between components and data flow

4. Continuous delivery is required to continually disseminate patterns and insights into various business domains
- 485 5. Two different approaches are required for data processing, stream and batch processing; or fast and delayed processing
6. Metadata should be managed at scale
7. Dimensional modeling for a rapidly changing schema is challenging

To provide a solution to these challenges, one has to realize the core fundamen-
490 tals of BD systems. Academic and practitioners of BD, describe BD as an interplay of methodology (workflow, organization), software engineering (data engineering, storage, etc.), and analysis (math, statistics) [48][7]. Therefore, one can deduce that technology orchestration is a focal matter in BD system development and maintenance.

495 Positioned on top of this rationale, and based on the result of the SLR synthesis, RAs can be considered an effective artefact that help with component delineation, interface definition, technology orchestration, variability management, scalability, and maintenance of BD systems [45][49]. The purpose of RAs is to create an integrated environment in which fragmented processes around
500 the system are optimized, responsiveness to change is assured, and delivery of architectural strategies is supported.

Most authors and practitioners agree that issues around BD software engineering and system development are severe and that this justifies the use of RAs for BD systems. Starting with a grounded RA means that the software architect
505 can refer to an already designed orchestration of components, interfaces, inter-communications, and variability points and map them against the organization's capability framework, desired quality attributes, and business drivers. This also means that the software architect or the architectural group is no longer challenged to model a new architecture from an array of independent
510 components that needs to be assembled.

Taking all into consideration, one can deduce that RAs are artefacts that facilitates development and homogenization of BD systems. Using RA to address

complex problems have been successfully applied for Database Management Systems (DBMS) [50] and Distributed Database Management Systems (DDBMS) [51].

8. What are some common approaches to creating BD RAs?

The findings gained from this study led to the understanding that there are not many frameworks available for design and development of RAs. One of the most commonly used approaches for developing RAs is ‘Empirically grounded Reference Architectures’ by Galster and Avgeriou ([52]). The research methodology is well-received because of its emphasis on empirical validity and empirical foundation. This methodology is comprising of 6 step process which are respectively 1) Selecting the type of the RA, 2) Selection of the design strategy, 3) Empirical acquisition of data, 4) Construction of the RA, 5) Enabling RA with variability, 6) Evaluation of the RA.

Another seminal work in this area is a framework for analysis and design of software RAs created by Angelov, Grefen, and Greefhorst ([53]). The framework utilizes a multi-dimensional classification space to classify RAs and as a result presents 5 major types. It is developed with the objective of supporting analysis of RAs in regards to their architectural specification/design, goal, and context. This is achieved through three major dimensions, each having their own corresponding subdimensions of design, goal, and context.

These dimensions and sub-dimensions are derived by interrogatives of ‘why’, ‘where’, ‘who’, ‘when’, ‘what’, and ‘how’. The interrogative why addresses the goal of the RA, who, when, where address the context, and how and what address the design dimensions. This framework categorizes RAs in two major groups: facilitation RAs and standardization RAs.

Volk, Bosse, Bischoff, and Turowski ([54]) utilized Software Architecture Comparison Analysis Method (SCAM) to compare and examine RAs based on their applicability. The result of this work was a decision-support process for selection of BD RAs. Two standards that have been observed the most

were ISO/IEC 25010 for choosing quality software products for RAs ([55]), and ISO/IEC 42010 for architecture description ([56]).

Surprisingly, based on the evidence gained from this SLR, most researchers
545 and practitioners use informal architectural description methods like boxes and
lines, except for the works of Geerdink ([44]). Geerdink used ArchiMate ([57]) as
the architectural description language which is a formal and standard language
that recommended in ISO/IEC 42010 as well. Informal methods of modeling
can introduce inconsistency issues between system design and implementation
550 of the system ([58]), do not adhere to a well-established standard and do not
promote the development of modeling approaches.

Therefore, one can argue that there is a need for more emphasis on usage
of standard architectural description languages to discuss and portray ontolo-
gies. Lastly, Hevner’s information systems research framework ([59]) has been
555 used for the development of RA presented by Geerdink ([44]), which is a suit-
able research design, since a BD RA is an information system artefact based on
existing literature and business needs.

9. Challenges of creating BD RAs

Among the challenges of developing RAs, perhaps evaluation is the most
560 significant [60]. According to Galster and Avgeriou ([52]), two fundamental
pillars of the evaluation is the correctness and the utility of the RA and how
efficiently it can be adapted and instantiated.

RAs and concrete architectures come with a different level of abstraction
and have divergent qualities. Whereas there are many well-established evalua-
565 tion methods for concrete architectures such as Architecture Level Modifiability
Analysis ([61]), Scenario-based Architecture Analysis Method ([62]), Architec-
ture Trade-off Analysis Method ([63]), and Performance Assessment of Software
Architecture ([64]), none of these can really be directly applied to RAs.

For instance, ATAM is reliant on participation of stakeholders in early stages
570 for creation of utility tree, and RAs, being highly abstract, do not have a clear

group of stakeholders at that stage. In addition, many of evaluation methodologies listed make use of scenarios, whereas RAs are highly abstract and are potentially adopted for various contexts, therefore making scenario creation difficult and sometimes invalid. Either a few general scenarios are developed to
575 cover all aspects, or a large number of specific scenarios are developed to cover various aspects of the RA. Each of which can pose threats to validity.

Based on three problems discussed above, available methods of architecture analysis are not sufficient for evaluating RAs. Various researched tried to address this problem. In one study, Angelov et al ([42]) modified ATAM and
580 extended it to resonate well with RAs. This process took place by invitation of representatives from leading industries for the evaluation process which included selection of various contexts and defined scenarios for these contexts. ATAM was extended to evaluate completeness, buildability and applicability. Howbeit the selection of the right candidate and involving them in the process is a daunting task and unfeasible at times. This can also poses threats to validity and
585 generalizability of the theory generated.

In Another study by Maier et al. ([60]) as a postgraduate dissertation in Eindhoven University of Technology, the evaluation of the RA has been conducted by mapping it against existing reference and concrete architectures described in industrial whitepapers and reports. Along the lines, Galster and
590 Avgeriou ([52]) suggested reference implementations, prototyping and incremental approach for the validation of the RA.

By the virtue of the findings from this SLR, and by studying the approaches from Bosch ([65]), Avgeriou ([66]), and Derras et al ([67]), an evaluation frame-
595 work for a RA can be done through architectural prototype evaluation, which means a concrete architecture of the RA is generated and then evaluated through a well-grounded method such as ATAM.

10. What are current BD RAs available in academia and industry?

As a result of this SLR and to answer RQ1, 22 BD RA has been found,
 600 among which 14 RAS are from academia, 8 from practice. These RA are listed
 in Table 1.

ID	Title	Domain	Year
s1	Lambda architecture ([68])	Practice	2011
s2	IBM - Reference architecture for high performance analytics in healthcare and life science ([69])	Practice	2013
s3	Microsoft - Big Data ecosystem reference architecture ([70])	Practice	2013
s4	Oracle - Information Management and Big Data: A Reference Architecture ([71])	Practice	2014
s5	Towards a big Data reference architecture ([60])	Academia	2013
s6	A reference architecture for Big Data solutions introducing a model to perform predictive analytics using Big Data technology ([44])	Academia	2013
s7	A proposal for a reference architecture for long-term archiving, preservation, and retrieval of Big Data ([72])	Academia	2014
s8	Questioning the Lambda architecture; Kappa Architecture ([73])	Academia	2014
s9	Accelerating Secondary Genome Analysis Using Intel Big Data Reference Architecture. ([74])	Practice	2014

s10	Reference architecture and classification of technologies, products and services for big data systems ([75])	Academia	2015
s11	SAP - NEC Reference Architecture for SAP HANA & Hadoop ([76])	Practice	2016
s12	Big data architecture for construction waste analytics (CWA): A conceptual framework ([77])	Academia	2016
s13	A reference architecture for Big Data systems in the national security domain ([41])	Academia	2016
s14	Managing Cloud-Based Big Data Platforms: A Reference Architecture and Cost Perspective ([78])	Academia	2017
s15	A software reference architecture for semantic-aware Big Data systems; Bolster Architecture ([49])	Academia	2017
s16	Simplifying big data analytics systems with a reference architecture ([79])	Academia	2017
s17	NIST Big Data interoperability framework ([45])	Practice	2018
s18	Extending reference architecture of big data systems towards machine learning in edge computing environments ([80])	Academia	2020
s19	A Big Data Reference Architecture for Emergency Management ([81])	Academia	2020
s20	ISO/IEC 20547-3:2020 BS ISO/IEC 20547 3:2020 Information technology. Big data reference architecture. Reference architecture ([82])	Practice	2020

s21	Phi: A Generic Microservices-Based Big Data Architecture ([83])	Academia	2021
s22	NeoMycelia: A software reference architecture for big data systems ([84])	Academia	2021

Table 1: BD RAs

Within the past years, there has been a considerable attention to the BD domain, and in specific BD system development. For instance, in March 2012, White House announced an initiative for BD research and development [85].
605 The goal of this initiative was to accelerate the speed of science and engineering discovery, to improve national security, and to improve the knowledge extraction from large and complicated sets of data [86]. This project has been supported by six federal departments and has been given more than \$200 million USD with the goal of substantial progress in the tools and techniques to handle big
610 data.

A year later, in June 2013, National Institute of Standards and Technology (NIST) Big Data Public Working Group (NBD-PWG) was launched with considerable participation from across the nation. Practitioners, researchers, agents, government representatives, and none-profit organizations joined in this
615 momentum.

One of the results of this project was NIST Big Data Reference Architecture (NBDRA). According to US Department of Defense, one of the main objectives of NBDRA was to provide with an authoritative source of information on big data that restraint and guides the overall practice. This is arguably one of the
620 most comprehensive and recent RAs available in the fields of big data. NBDRA is made up of two fabrics encompassing five functional logical components connected by various interfaces, representing intertwined nature of security and privacy and management.

Along the lines, other giant IT vendors published their own RAs for big

625 data. In this SLR, 8 BD RA has been collected from the practice, and mostly
through white papers. These white papers are from IBM, Microsoft, Oracle,
SAP, and a conference in which Lambda was discussed. Among these RAs,
arguably Lambda architecture is the most commonly discussed and studied. It
is also worth mentioning that there has been other BD RAs found in practice,
630 but they were rather too short or did not reflect the contemporary state of BD
analytics and has been eliminated as described in the research methodology
section.

We are well aware that Lambda is presented as an architecture and not an
RA, but we have grouped as an RA as it sits at an abstraction level of an RA.

635 In the realm of academia, there has been numerous efforts including a post-
graduate master's dissertation ([60]) and PhD thesis ([87]) for creating big data
RAs. In addition, few universities have published their own RA. For instance,
university of Amsterdam published a BD architecture framework [88].

Last but not least, there has been numerous reference architectures devel-
640 oped recently for specific domains. These studies have been usually published
as short journal papers, and many have promised future publication of the full
reference architecture as a book. For instance, Klein et al. ([41]) developed a
BD Ra in the national security domain, and Weyrich and Ebert ([89]) worked
on a BD RA in the domain of internet of things (IOT).

645 Through the process of literature review for this SLR, scarcity of big data
reference architectures has been witnessed. The studies listed above are promi-
nent research, with great potential to induce concrete architectures. But with
all, they are mostly published as short journals and provide with little infor-
mation about quality attributes, data quality, metadata management, security,
650 and privacy concerns. In another terms, they are notion or brief discussions on
reference architectures in very particular domains.

11. What are major architectural components of BD RAs?

To address RQ2, RAs listed in 1 was reviewed and compared to highlight common architectural components of BD RAs. Some of the RAs collected were in in the form of a short paper and provided with not much detail, whereas some of the other such as NIST were quite comprehensive.

Majority of RAs have been inspired or based on other RAs, and this signified the notion that “RAs can be perceived more effective when they are created from the body of available knowledge rather than from scratch”.

To answer this question in a systematic manner, and as a result of our data extraction, we listed all the components from all the BD RAs listed in the previous section. These components are described in table 2.

RA	Components
s1	Streaming layer, batch layer, serving layer
s2	Applicatons, Frameworks and platforms, Software defined infrastructure, Compute and storage servers
s3	Data sources, Data transformation, Data usage
s4	Data sources, Data Integration, Information Management, Information Access
s5	Data sources, Data Acquisition and Recording, Information Extraction and Cleaning, Data Integration, Aggregation and Representation, Query Processing, Data Modeling and Analysis, Interpretation
s6	Import Engine, Processing Engine, Management Engine, Analytics Engine, Visualisation Engine
s7	Big Data Layer, Archive Layer, Storage Layer, Presentation Layer
s8	Data Source, Real-Time Layer, Serving Layer
s9	Access Manager, Intel Big Data Analysis Platform, Data Ingestion, Dat Sources

s10	Data Sources, Data Extraction, Data Loading and Pre-Loading, Data Processing, Data Storage, Data Analysis, Data Loading and Transformation, Interfacing and Visualization
s11	Data Input sources, Data Processing Platform, Processed Data for Client
s12	Application Layer, Analytics Layer, Storage Layer, Data Sources
s13	Data Providers, Big Data Application Layer, Big Data Framework Provider, Data Consumers
s14	Data Generation, Data Streams, Data Storage, Stream Processing, Data Warehouse, Hadoop Cluster, Machine Learning, Presentation
s15	Batch Layer, Speed Layer, Semantic Layer, Serving Layer
s16	Data Source, Data Integration, Data Analysis and Aggregation, Interface/Visualization
s17	Data Provider, System Orcehstrator, Big Data Application Provider, Big Data Framework Provider, Security and Privacy Fabric, Management Fabric, Data Consumer
s18	Data Sources, Data Extraction, Data Loading and Preprocessing, Data Processing, Data Storage, Model Development and Interface, Data Transformation and Serving, Interacing and Visualization
s19	Data Provider, Big Data Application Provider, Big Data Framework Provider, System Orchestrator, Management Fabric, Security and Privacy Fabric, Data Consumer
s20	Big Data Application Provider, Big Data Processing Layer, Big Data Platform Layer, Big Data Infrastructure Layer, Integration, Security and Privacy, System Management, Big Data Provider, Big Data Consumer

s21	Acquisition Layer, Refinement Layer, Scrutiny Layer, Training Layer, Insight Layer
s22	Gateway, Stream Processing Service Mesh, Stream Processing Controller, Monitoring, Service Discovery, Query Controller, Batch Processing Controller, Batch Processing Service Mesh, Event Backbone, Data Lake, Query Engine, Event Archive, Semantic Layer, Control Tower, MicroService, Sidecar, Event Queue

Table 2: BD RAs Components

Different studies have chosen different phrases to describe their architectural components, and there seems to be no standard way of modeling BD RAs. The usage of architectural definition languages such as Archimate is scarce, and most studies have used boxes and lines with specifically defined ontologies. This made understanding and comparison of these RAs a difficult task that requires constant translation from one ontology to another.

Initially we’ve done an automated text analysis on these components’ names to try to highlight commonalities and word usage. The word cloud of these components are portrayed in figure 3.

Among the names authors used to name their components, ‘big data application provider’ seems to have been used the most (5 occurrences), and ‘big data framework provider’ the next (3 occurrences). This is due to the fact that a few of the RAs are built upon NIST BD RA (s17), and have therefore adopted the terminology. One term that all studies seems to have been using uniformly is ‘data consumer’ and ‘data provider’. Moreover, most studies have chosen the phrase ‘layer’ to logically group different components of the RA.

To achieve this, we paid clear attention to the description of these components and categorized them based on their functions. These categories are; 1) BD Management and Storage, 2) Data Processing and Application Interfaces, 3) BD Infrastructure.



Figure 3: BD RA Component Names Word Cloud

11.1. BD Management and Storage

One of the prominent characteristics of big data is ‘variety’, which rises
the need for distinct storage solutions. This is sometimes referred to as ‘poly-
glot persistence’ [90]. For instance, when it comes to dynamic data, NoSQL
databases such as MongoDB is a suitable choice because of their non-tabular
nature (Banker, Garrett, Bakkum, & Verch, 2016), and when there is a need for
complex relationship between entities, graph databases such as Neo4J are more
suitable because of their tree traversal performance (Van Bruggen, 2014).

Choosing the right database or databases, is an important architectural decision that can also include patterns for data access, storage and caching. For example, the practitioners of distributed system that are specialized in micro-services architecture may opt to use Command Query Responsibility Segregation (CQRS) pattern for high performance applications [91]. Therefore, the type of storage and the access pattern are two major architectural components of big data systems.

The current landscape of BD RAs seems to revolve around monolithic storage solutions such as data warehouse and data lake. While the traditional practice of staging data, dimensional modeling, storage in data warehouses, and data

marts as customized access layers, may seem ineffective in handling BD loads, we’ve been surprised to still witness some variations of this approach being proposed.

Another architectural component that is popular in BD RAs is data lake.
705 Data lake can be perceived as an ingestion framework that can be given various types of data including internal and external data. The data stored in the data lake is then usually retrieved for transformation. This is the LET (load, extract, transform) approach, comparing to old ETL (extract, transform, load) approaches.

710 Similar to the way that Business Intelligence (BI) and BD differ in their source data types both in terms of granularity and data structure of it, a data lake and data warehouse are different. In the case of a data warehouse, usually a relational database is used which decreases flexibility when it comes to analysis and can potentially cause considerable costs. In the case of data lake, data of
715 different kind can be stored without the engineer needing to define the schema in advance. This increases the flexibility.

Howbeit, this flexibility itself has its own downside and can be abused by data engineers. One can throw different data sets in the data lake, without much regard for how they’re structured and consumed, and this can in turn,
720 result in the creation of a data swamp. Data governance and active metadata can alleviate some of these issues [92].

Based on the results of this synthesis, we posit that BD RAs are driven by three main paradigms; 1) enterprise data warehouse paradigm, 2) data lake paradigm, and 3) multi-modal cloud based paradigm.

725 The first paradigm revolves around large monolithic enterprise data warehouses, with ETLs, staging environments and a data processing pipeline. A good example of the first paradigm is S5. The second paradigm is about monolithic data lakes containing data that follows a similar data processing pipelines as data warehouses, but happening at a later stage. A good example of the
730 second paradigm is S22. The third paradigm is not that far away from the second, but aims to incorporate more elements of distributed systems and cloud

features. A good example of this paradigm is the S8.

Moreover, some RAs sit at a higher level of abstraction. A good example is S21. For these kind of RAs, one cannot assume the nature of the pipelines and if the storage would be monolithic or not. In S21, there's a depiction of various kinds of storage in the 'Big Data Platform Layer', indicating that one may choose to opt in for polyglot persistence. Therefore, only the resulting system can help us realize what paradigm the RA may belong to.

When it comes to big data management, many of the cross-cutting concerns seems to be overlooked. For instance, we have realized that many BD RAs do not pay a clear attention to security, privacy, metadata management and data quality. While some RAs tend to revolve around security such as S13, and some other tend to revolve around metadata management such as S15, we could not find a comprehensive explication of BD cross-cutting concerns.

We could not understand how some of the RAs could scale to account for data source proliferation and how rapidly they could react to regional data privacy changes.

11.2. Data Processing and Application Interfaces

There are two major data processing activities that a BD system encompasses. These processes generally fall into stream processing and batch processing. Stream processing or fast processing is required for sensitive operations and time critical processes such as checking a fraudulent credit card, and batch processing required for a long-running continuum of data analysis such as regression analysis.

The decision on required type of processing for a context-specific architecture is determined by the characteristics of the data being analyzed, that is primarily variety, volume and velocity.

For instance, most algorithms for stream processing are using in-memory stateful data structures such Hyperloglog to compute values in real-time. A streaming component can be tailored to adopt specific windowing approaches such as tuple-at-a-time and a micro-batch processing. When in fact, these tech-

765 techniques are not required for batch processing. An architect may opt for MapReduce and Bulk Synchronous Parallel (BSM) processing for batch-oriented requirements or go for a streaming processing based on a specific performance requirement set to handle velocity and volume of data.

Various studies have provided different level of abstraction when it comes to describing data processing. While some studies like S19 describe the processes in the data processing pipeline, some others like s15 have just abstracted it to 'batch processing' or 'stream processing'.

770 Moreover, we realized two category of data processing. First category utilizes two different architectural constructs for batch and stream processing, while the latter tends to process both in one architectural component. This is a difference that can be seen between Lambda and Kappa, and the RAs that have been derived from the two. Some RAs such as S17 have used three architectural
775 constructs, namely batch node, streaming node and interactive node.

BD interfaces are communicated in two different ways, either the RA only presented a 'serving or access layer' (S21, S17), or several components that are each specific to a different requirement (ML, BI, etc). The latter can be witnessed in S16.

780 Some RAs tend to clearly distinguish to ingress and egress interfaces and inter-node interfaces such as s23, while others mostly annotated interfaces with just an arrow. Along the lines, some studies have created terms for their interfaces such as 'provider-request interface' in s15. While interfaces are quite important architectural constructs, current RAs don't seem to be vested a lot
785 of interest in clearly portraying them.

11.3. *BD Infrastructure*

Another major area that is discussed in the RAs is the concept of infrastructure. Different authors have taken different approaches to communicate this. Some have presented it through a standard architectural description language
790 such as s6, which clearly defines the technology layer, and some have not mentioned the concept of infrastructure, but it's rather implicitly conveyed such as

s22 and s23. Some other have presented with both infrastructure and platform layers such as s21 and s17.

For instance, one major component of a NIST BD RA (s17) is called big
795 data framework provider which includes ‘computing and analytics’, ‘data organization and distribution’, and ‘infrastructures such as networking, computing and storage’.

BD infrastructure is more of a layer than a component. A layer in which the RA lays out a possible computing and networking design of a BD system. This
800 is crucial, as practitioners of BD have been commonly architecting underlying distributed paradigms and horizontal scaling.

Therefore, CAP theorem, ACID and BASE transactions, data consistency, service discovery, and tail latency are potential architectural challenges one should consider. Should a BD system adopt an event-driven approach through
805 an event backbone such as Kafka that is discussed in s23? Or should it stick to REST based communication. What is the overhead of context switch and networking in the case of RPCs among services?

All and all, as a result of this SLR, a component of a BD infrastructure has been witnessed as a common pattern, in various forms and approaches. We consider platform layer of BD as a major architectural component of these systems.
810 Whereas one might argue that infrastructure is an architectural component of any system, the design challenge is more significant in the case of BD systems as these systems are usually distributed in nature.

Another area that seems to be neglected is DataOps. While DataOps can
815 play an important role in the automation and delivery of data engineering workloads in an agile manner, we could not find BD RAs that incorporate this aspect.

Lastly, our findings depicts the fact that many of RA presented are not designed underlying completely distributed architecture while BD systems can benefit from this paradigm. An exception to this is S23, which is absorbing
820 many patterns from event-driven microservices architecture.

12. What are the limitations of current BD RAs?

To answer the RQ3, RAS collected for this SLR have been appraised to point out limitations. To arrive at this objective systematically, we first described each RA with its limitation briefly. This is portrayed in table 3.

RA	Description/Limitations
S1	While Lambda is one of pioneers of BD architectures, and perhaps the oldest one, it does not address data quality issues, does not address seem to address changes in landscape of data, and seems to be providing with the bare minimum requirement for BD systems
S2	This RA is specifically designed for the domain of healthcare and life sciences, does not seem to account for data quality, security and privacy. Resembles to monolithic n-tier architectures, and pivots on IBM specific solutions such as LSF process manager, LSF application center, and LSF data manager.
S3	This RA seems to be covering a very general scenario of big data analytics, without any attention to either metadata or security. It flows from data sources to data transformation, and into data usage, but it's unclear how data quality is assured, how it will support changes in data landscape, and how it will respond to changes. In addition, the RA is heavy influenced by MapReduce, while MapReduce may not provide the best of performance in comparison to newer approaches such as acyclic direct graph used in Apache Spark and Tez.

S4	<p>A classically designed RA with 3 main phases of ingestion, processing and providing. This RA has underlying mechanisms that facilitates 'right-time' flow of data through the system. This is achieved by having two architectural constructs for strongly typed data and weakly typed data. While this was annotated as the data quality part of the RA, it does not seem to account for data ownership and changes in data landscape. In addition, this RA utilizes data mart for access and performance, which can affect the overall modifiability negatively.</p>
S5	<p>This RA is the augmentation of traditional data warehouse architectures with a few new components to account for stream processing. The RA defines three kind of storage, data warehouse, sandbox and raw data archive that communicate with each other through data load components. There seems to be many moving parts in this RA, and privacy and metadata is clearly addressed. Nevertheless, the RA is lacking in the area of security, data provenance, scalability and modifiability. We can't tell how data ownership is in place, and how data quality is addressed. It's also driven mostly by data warehouse way of thinking, like staging environments and data marts, which can be hard to maintain.</p>
S6	<p>This RA seems to be portraying the bare minimum requirements for big data analytics with no mention of security, privacy, metadata, or data quality. The RA is published as a short paper, thus not much detail is given.</p>
S7	<p>This RA provides with bare minimum components for data analytics, without any clear identification of stream processing. The RA seems to be done in quite a reductionist manner, with no attention to privacy, security, metadata, and data quality. Data storage seems to be only associated to hardware, and thus it's unclear how data is evolved and scaled.</p>

S8	Kappa is perhaps the predecessor of Lambda, aiming to address some of the limitations of it. The major difference between Kappa and Lambda is that Kappa has a unified processing layer for batch and stream process, which eliminates the complexity of maintaining two separate systems, and reduces costs. Nevertheless, this architecture, does not discuss meta-data, security, privacy, data quality, and maintainability in details.
S9	This BD RA is designed for healthcare application by Intel. This RA is not elaborated in detail, does not seem to account for cross-cutting concerns such as security, metadata, privacy, and data quality. The concept of access manager seems to be vague, as we could not understand how the access is managed. Moreover, the artefact seems to be a simple instance of Hadoop ecosystem with some extra components added
S10	A comprehensive RA that aim to cover many aspects of data engineering. Nevertheless, we did not find any notion of metadata management, neither was a discussion on security and privacy challenges. While this RA could work successfully for a regular BD workflows, we are unsure how data quality is met and how scalability is achieved.
S11	This RA is made up of three major phases, ingestion, processing and presentation. It is designed around Hadoop ecosystem, and provides with bare minimum necessary to conduct data analytics. We could not find a discussion on metadata management, security, privacy or data quality. The data pipeline is using a data warehouse, and uses it to communicate to the Hadoop side of things. We are not sure how unstructured data is handled, and how data lineage is achieved.
S12	This architecture is specifically designed for waste management, and seems to be using an approach similar to Kappa. The data takes a generic flow from data sources to application, without any clear identification of data quality, privacy and security concerns.

S13	This RA is specifically deigned for the security domain and seems to have a lot of inspiration from NIST BD RA. The RA is layed out in fabrics just like the NIST one, and unlike many others, does mention cross-cutting concerns such as security explicitly. However the concept of data ownership is not discussed, there's no mention of metadata or privacy, and the artefact evaluation is not extensive. That is, we cannot ensure how the derived solutions from this RA can scale.
S14	A generic RA that resembles to Lambda architecture, with stream and batch processing being processed in different nodes. This RA utilizes data warehouses and hadoop cluster for data processing. It was unclear how the security, privacy, metadata, and data quality is achieved. Maintainability aspects are not discussed as well.
S15	This RA extends the Lambda architecture by adding a semantic layer. It has a great focus on handling metadata in a right manner, but it does not seem to have any identification of other cross-cutting concerns such as privacy, security or data quality. It also adopts the idea of separate batch and stream layers, which can potentially affect modifiability negatively and increase cost.
S16	This RA clearly segregates stream data from other data, and defines clear interfaces for ingestion of different data types. It then passes the data directly to a distributed storage (Hadoop's HDFS), and retrieves it later for deduplication and cleaning. While the RA seems to have addressed the minimum requirements of data analytics, it does not seem to address cross-cutting concerns such as metadata, security and privacy. It is also unclear how data quality is achieved.

S17	This is perhaps the most comprehensive BD RA found in this SLR, and has been heavily funded by the government of the USA. While this RA is a good tool to facilitate open discussion, design structures, requirements, and operations inherent in BD, it is more of a high-level conceptual model of BD, rather than an effective BD RA. Some of the limitations witnessed in this RA is in its brief mention of metadata management (only discussed in lifecycle management), unclear approaches to attain data quality and data ownership, and potential monolithic coupling of components in big data application provider.
S18	This RA segregates data extraction and data loading phases, and tend to adopt the idea of segregating stream and batch processing layers. Nevertheless, we could not find the identification of cross-cutting concerns such privacy, security or metadata management. We could not understand how data quality is achieved. There are also three storage designed, but seems like all data will eventually be stored in one giant storage. This can potentially make modifiability harder and create a choke point.
S19	Driven from the NIST BD RA, this RA is more or less identical with BD RA, but tailored specifically for emergency management. We skip explanation for this RA, as the limitations discussed for NIST BD RA, can be applied to this one as well.
S20	This RA shares all the fundamental components with BD RA, and seems to be very similar. However, the phrase fabrics seem to be changed to multi-layer functions. Therefore this RA, just like NIST is too abstract and leaves many architectural decisions unknown such as data storage, data quality assurance, and data ownership. It is unclear on how storage should be approached, and the overall structure resembles to a monolithic data pipeline architecture.

S21	One of the few RAs that tend to absorb the concept of microservices into BD development. Nevertheless, the RA seems to be driven by the idea of one data lake for all data storage, which can be a daunting task to scale and maintain. The concept of metadata does not seem to be discussed, and other concerns such as security, privacy, data quality and data provenance are unclear.
S22	This RA absorbs a lot of patterns from microservices event driven architectures and reactive systems and tend to absorb them into the BD development. While there's been a clear attention to cross-cutting concerns such as metadata, and privacy, security does not seem to be well discussed in the study. The RA also tends to use data lake as the single source of storage which can be challenging to scale.

Table 3: BD RAs and Their Limitations

825 Except for one case (S22), all the architectures and RAs found as the result of this study, were designed underlying monolithic data pipeline architecture with four major components being data consumer, data processing, data infrastructure and data providers. To discuss the integral facets that embroil these architectures, one must look at the characteristics of these architectures and the
830 ways in which they achieve their ends.

The process of turning data into actionable insights in these architectures usually follow a similar lifecycle;

1. **Data Ingestion:** system begins to ingest data from all corners of the enterprise, including both transactional, operational and external data.
- 835 2. **Data Transformation:** data captured from the previous step is then cleansed for duplication, quality, and potentially scrubbed for privacy policies (scrubbing is hardly discussed in the studied RAs). This data then goes through a multifaceted enrichment process to facilitate data analysis.
3. **Data Serving:** at this stage, data is ready to be served to diverse array of

840 needs ranging from machine learning to marketing analytics, to business intelligence to product analysis and customer journey optimization.

The lifecycle depicted is indeed a high-level abstract view of prevalent BD systems. Howbeit, it highlights an important matter; these systems are all operating underlying monolithic data pipeline architecture that tends to account for 845 all sorts of data. This means, data that logically belong to different domains are now all lumped together and crunched in one architectural constructs, making maintainability and scalability a daunting task.

While architectures in software engineering have gone through series of evolution in the industry, adopting a more decentralized and distributed approaches 850 such as microservice architecture, event driven architectures, reactive systems, and domain driven design , the data engineering, and in specific BD ecosystems do not seem to be adopting many of these patterns. The whole idea of 'monolithic data pipeline architecture with no clearly defined domains and ownership' brings significant challenges to design, implementation, maintenance and scaling 855 of BD systems.

This architecture and system design if done underlying these approaches, can result in hard to maintain systems with high costs, and leave many managers disappointed. Nevertheless, we don't claim that all these architectures will fail, perhaps some have proven to be successful in a specific context. There are two 860 threats to maintainability and scalability of these systems;

- *Data source proliferation*: as the BD system grows and more data sources are added, the ability to ingest, process, and harmonize all these data in one place diminishes.
- *Data consumer proliferation*: organizations that utilize rapid experimentation approaches such Hypothesis-Driven Development and Continuous 865 Delivery constantly introduce new use cases for data to be consumed in different domains. This means that variability of the data rises, and the sum of aggregations, projections, and slices increases, which in turn adds

more work to the backlog of the data engineering team, slowing down the
870 process of serving the data to consumers.

Another limitation that we came across was that the concept of metadata
has been poorly discussed. For instance, s5 discussed the limitation of metadata
management systems, stating that most metadata solutions are ad-hoc. The
researcher then went ahead and introduced a layer for metadata management
875 in the RA, but as a non-integrated component. For instance, the author did not
discuss how data provenance can be achieved through the RA and underlying
which logical flow one can do linear analysis.

In another case, NIST BD RA only discusses metadata in a sentence, and
in sub-activity named ‘metadata management’. The RA only states what are
880 essential metadata information and how they are used. Except for s15 and s22,
metadata has not been accentuated enough and metadata layer is not thoroughly
discussed. This is a noticeable limitation in current BD RAs, as metadata plays
an important role in BD systems, addressing wide range of challenges such as
privacy, security, data provenance, and linear analysis.

885 Based on that, one can argue that any BD system can benefit from a well-
defined metadata layer as a means for bridging data stored in different plat-
forms such as on-premises or on cloud, reducing complexity, facilitating access
management, facilitating data governance, and potentially the creation of data
mesh.

890 Furthermore, white papers collected from IT giants tend to pivot the RA
around their services, which can potentially reduce its applicability, hinder RAs
openness, and even affect architectural qualities. In these white papers, al-
ternative technologies or vendors are typically not discussed which leaves the
architect with a small pool of options.

895 Lastly, privacy and security does not seem to have been discussed enough,
or it has been mostly marginalized. For instance, we have not found an archi-
tectural component that allows for data scrubbing, or we did not understand
how one can achieve security in-between data pipelines. Specially, in regard to

privacy and with recent global movements towards increased privacy, BD ar-
900 chitects are now increasingly challenged to design under the shadow of regional
data privacy policies such as GDPR. Placing this challenge next to security
challenges of package management, endpoint proliferation and DDOS handling
can further signify the necessity for more research on BD RAs.

Many of core architectural decisions revolving around security and privacy,
905 if not addressed in an initial phase, can result in massive losses and potential
bottlenecks.

13. Discussion

In this section, we provide a summary of main findings and the potential
implications for both industry and academia.

910 In this study, by adopting a rigorous research methodology, we aimed to
understand the state of the art in BD RAs. To the best of our knowledge, there
is no study in academia that has conducted a SLR on BD RAs, nor there is a
systematic mapping study. While RAs can play a pivotal role in development
and maintenance of BD systems, there does not seem to be enough attention
915 on this topic.

The closest study we could find to comparing and analysing BD architectures
was the work of Volk et al ([54]), which does not revolve around BD RAs,
but attempts to develop a decision support system for selecting BD reference
architectures. However, this study stays fairly light on BD architectures and
920 does not aim to systematically collect them.

NIST BD RA (S17) have collected a series of white paper from the big data
public working group, and used it as a foundation study. Howbeit, these white
papers are not detailed BD RAs, and are rather concepts that different memebrs
of the group have put together for the purposes of contrast and comparison.

925 Our findings from this study yielded the fact that progres is uneven in the
area of BD RAs. While there are many researches in the area of data ware-
housing, artificial inteligence, data science, and IOT, data engineering seems to

be needing more research. While, there are many well established approaches for crunching large volume of data, or handling dimensionality of complex data sets, the overall organization of BD technologies, which is the architecture, needs more attention from academia and industry.

Majority of the BD RAs that we have analyzed were running underlying some sort of a monolithic data pipeline with a central storage. This is a challenging architecture to scale and maintain. How does one takes preventive measures to stop a data lake from turning into a data swamp ? How a team of hyper-specialized siloed data engineers that are running the data pipelines, will be aware of the actual consumption of the data and therefore keep a certain level of quality of that data.

If a software engineers decides to, for instance, manipulate a certain field in an entity's schema for the development of a new feature, how will this affect the data engineering process and how is this communicated? as data becomes more and more available to the company, the ability to consume it all in one place diminishes.

On the other hand, the current state of BD RAs do not seem to be very far away from traditional data warehousing approaches. In fact, some of them have adopted the idea of data marts and propose them as BD solution, but using newer technologies. Moreover, some architectures have attempted to utilized data lake to serve data analysts and business intelligence.

We posit that neither the attempt to onboard BD analytics workloads to data warehouses, nor the attempt to serve business intelligence with data lakes is gonna result in a scalable and maintainable system. We therefore propose future research directions in the area of decentralised and distributed BD RAs.

We also felt that the quality of many of BD RAs published does not seems to be enough to meet the industrial expectations. This is due to the challenges of developing BD RAs and the cost and resources required to evaluate these artefacts. It is also worth mentioning that a rigorous methodology for evaluating reference architectures are quite rare, and while there are studies that have attempted to address these issues ([42]), there is a need for more research in

this area.

960 Given all, we posit that RAs can be considered an effective initial point
to design and development of BD systems. These artefacts helps facilitating
communication, capture requirements from various stakeholders, and catch de-
sign issues while they are still cheap. Based on this, therefore, more and more
attention needs to be given to this area and its foundational methodological
965 needs.

14. Threats to Validity

Just like any rigorous SLR, and based on our research methodology, it is
essential to conduct a 'threats to validity' assessment to transparently commu-
nicate any bias or flaws that research may suffer from. Therefore, the threats
970 to validity for this study are as following;

1. *Construct Validity*: This SLR is as good as the quality of the evidence
collected and the synthesis of it. Therefor we have taken extensive mea-
sures to maximize the rigour in this study. By following some of the
most prominent guidelines for SLRs, and a few complementary studies for
975 search strategy, inter-rate reliability, data synthesis, and quality assess-
ment, we ensured that at every step we are removing bias, and increase
transparency and systematicity of our study. We also opted for thematic
synthesis, and followed a well established approach to create themes and
models.
- 980 2. *Internal Validity*: To avoid losing on valuable information, we augmented
our SLR with grey literature, and used a rigorous method to incorpo-
rate those literature into the main pool of literature. We snowballed pa-
pers, looked for references, and even searched the works of well-known
researchers in the field.
- 985 3. *Conclusion Validity*: One potential threat to validity might be that some
of the studies we collected may not have been mature enough, and that
in turn might have impacted the generalizability of our conclusions. To

mitigate this threat, we have developed a rigorous quality framework and assessed each pooled literature based on it. We made sure that this process is conducted by different individuals so to achieve the quality desired.

15. Conclusion

This study sought to find all BD RAs available in practice and academia. The findings gained emerges the understanding that RAs can be an effective artefact to tackle complex BD system development. RAs at their core bring software engineering knowledge as a collection of patterns designed to address a class of problems with attention to specific requirements and context and do solve many of the prevalent architectural challenges that an architect might face.

As data proliferates further, there will be more BD systems created which in turn means more technology orchestration is required around data that can be effectively done through a well-established RA. RAs guide the evolution of the system both in terms of functional and non-functional requirements, and pinpoint variability points that can result in more successful BD projects and avoidance of common pitfalls.

Withal, BD RAs have yet to mature and become ubiquitous in industry and there is further research required in this area. These researches can be done in the area of micro-services RA for BD systems, event-driven paradigms for BD systems, security and privacy issues in BD systems, and metadata management.

References

- [1] B. Bashari Rad, N. Akbarzadeh, P. Ataei, Y. Khakbiz, Security and privacy challenges in big data era, *International Journal of Control Theory and Applications* 9 (43) (2016) 437–448.
- [2] P. Ataei, A. T. Litchfield, Big data reference architectures, a systematic literature review (2020).

- [3] B. B. Rada, P. Ataeib, Y. Khakbizc, N. Akbarzadehd, The hype of emerging technologies: Big data as a service (2017).
1015
- [4] Databricks.
URL <https://databricks.com/>
- [5] N. Partners, Big data and ai executive survey 2021 (2021).
URL https://www.supplychain247.com/paper/bi_data_and_ai_executive_survey_2021/pragmadik
1020
- [6] S. Computing, Bridging the gap between data and business teams (2020).
URL <https://www.sigmacomputing.com/resources/data-language-barrier/>
- [7] B. B. Rad, P. Ataei, The big data ecosystem and its environs, International Journal of Computer Science and Network Security (IJCSNS) 17 (3) (2017) 38.
1025
- [8] I. Gorton, J. Klein, Distribution, data, deployment, STC 2015 (2015) 78.
- [9] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, M. Bone, The concept of reference architectures, Systems Engineering 13 (1) (2010) 14–27.
1030
- [10] M. Derras, L. Deruelle, J.-M. Douin, N. Levy, F. Losavio, Y. Pollet, V. Reiner, Reference architecture design: A practical approach, in: IC-SOFT, pp. 633–640.
- [11] I. Iso, Information technology — reference architecture for service oriented architecture (soa ra) — part 1: Terminology and concepts for soa, International Organization for Standardization (2016) 51.
1035
URL <https://www.iso.org/standard/63104.html>
- [12] G. Muller, A reference architecture primer, Eindhoven Univ. of Techn., Eindhoven, White paper (2008).

- 1040 [13] S. Angelov, P. Grefen, D. Greefhorst, A classification of software reference architectures: Analyzing their success and effectiveness, in: 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, IEEE, 2009, pp. 141–150.
- [14] L. Bass, I. Weber, L. Zhu, DevOps: A software architect’s perspective, 1045 Addison-Wesley Professional, 2015.
- [15] M. J. Page, D. Moher, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, et al., Prisma 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews, *Bmj* 372 (2021).
- 1050 [16] M. L. Rethlefsen, S. Kirtley, S. Waffenschmidt, A. P. Ayala, D. Moher, M. J. Page, J. B. Koffel, Prisma-s: an extension to the prisma statement for reporting literature searches in systematic reviews, *Systematic reviews* 10 (1) (2021) 1–19.
- [17] B. A. Kitchenham, D. Budgen, P. Brereton, Evidence-based software engineering and systematic reviews, Vol. 4, CRC press, 2015. 1055
- [18] M. Borrego, M. J. Foster, J. E. Froyd, Systematic literature reviews in engineering education and other developing interdisciplinary fields, *Journal of Engineering Education* 103 (1) (2014) 45–76.
- [19] [link]. 1060 URL <https://www.jabref.org/>
- [20] K. Krippendorff, Computing krippendorff’s alpha-reliability (2011).
- [21] G. W. Noblit, R. D. Hare, R. D. Hare, Meta-ethnography: Synthesizing qualitative studies, Vol. 11, sage, 1988.
- 1065 [22] T. Dybå, T. Dingsøy, Empirical studies of agile software development: A systematic review, *Information and software technology* 50 (9-10) (2008) 833–859.

- [23] M. Cumpston, T. Li, M. J. Page, J. Chandler, V. A. Welch, J. P. Higgins, J. Thomas, Updated guidance for trusted systematic reviews: a new edition of the cochrane handbook for systematic reviews of interventions, *Cochrane Database Syst Rev* 10 (10.1002) (2019) 14651858.
- [24] [link].
URL <https://casp-uk.net/casp-tools-checklists/>
- [25] [link].
URL <https://jbi.global/critical-appraisal-tools>
- [26] P. Runeson, C. Andersson, T. Thelin, A. Andrews, T. Berling, What do we know about defect detection methods?[software testing], *IEEE software* 23 (3) (2006) 82–90.
- [27] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, J. Rosenberg, Preliminary guidelines for empirical research in software engineering, *IEEE Transactions on software engineering* 28 (8) (2002) 721–734.
- [28] D. S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: 2011 international symposium on empirical software engineering and measurement, IEEE, 2011, pp. 275–284.
- [29] V. Braun, V. Clarke, Using thematic analysis in psychology, *Qualitative research in psychology* 3 (2) (2006) 77–101.
- [30] T. Dyba, T. Dingsoyr, G. K. Hanssen, Applying systematic reviews to diverse study types: An experience report, in: First international symposium on empirical software engineering and measurement (ESEM 2007), IEEE, 2007, pp. 225–234.
- [31] M. B. Miles, A. M. Huberman, *Qualitative data analysis: An expanded sourcebook*, sage, 1994.

- [32] J. Corbin, A. Strauss, Basics of qualitative research: Techniques and procedures for developing grounded theory, Sage publications, 2014.
- 1095 [33] J. Lofland, L. H. Lofland, Analyzing social settings (1971).
- [34] M. A. Martínez-Prieto, C. E. Cuesta, M. Arias, J. D. Fernández, The solid architecture for real-time management of big semantic data, *Future Generation Computer Systems* 47 (2015) 62–79.
- 1100 [35] O. Sievi-Korte, I. Richardson, S. Beecham, Software architecture design in global software development: An empirical study, *Journal of Systems and Software* 158 (2019) 110400.
- [36] J. Kohler, T. Specht, Towards a secure, distributed, and reliable cloud-based reference architecture for big data in smart cities, in: *Big Data Analytics for Smart and Connected Cities*, IGI Global, 2019, pp. 38–70.
- 1105 [37] H. Zimmermann, Osi reference model-the iso model of architecture for open systems interconnection, *IEEE Transactions on communications* 28 (4) (1980) 425–432.
- [38] OATH, Oath reference architecture, release 2.0 initiative for open authentication, OATH (2007).
- 1110 URL <https://openauthentication.org/wp-content/uploads/2015/09/ReferenceArchitectureVersion2.pdf>
- [39] A. L. Pope, The CORBA reference guide: understanding the common object request broker architecture, Addison-Wesley Longman Publishing Co., Inc., 1998.
- 1115 [40] D. Greefhorst, Een applicatie-architectuur voor het web bij de bank—de pro’s en contra’s van toestandsloosheid, *Software Release Magazine* 2 (1999).
- [41] J. Klein, R. Buglak, D. Blockow, T. Wuttke, B. Cooper, A reference architecture for big data systems in the national security domain, in: 2016

- 1120 IEEE/ACM 2nd International Workshop on Big Data Software Engineering
(BIGDSE), IEEE, pp. 51–57.
- [42] S. Angelov, J. J. Trienekens, P. Grefen, Towards a method for the evaluation of reference architectures: Experiences from a case, in: European Conference on Software Architecture, Springer, 2008, pp. 225–240.
- 1125 [43] V. Stricker, K. Lauenroth, P. Corte, F. Gittler, S. De Panfilis, K. Pohl, Creating a reference architecture for service-based systems—a pattern-based approach, in: Towards the Future Internet, IOS Press, 2010, pp. 149–160.
- [44] B. Geerdink, A reference architecture for big data solutions introducing a model to perform predictive analytics using big data technology, in: 8th
1130 international conference for internet technology and secured transactions (ICITST-2013), IEEE, 2013, pp. 71–76.
- [45] W. L. Chang, D. Boyd, Nist big data interoperability framework: Volume 6, big data reference architecture, Report (2018).
- [46] H.-M. Chen, R. Kazman, J. Garbajosa, E. Gonzalez, Big data value engineering for business model innovation (2017).
1135
- [47] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, C. Shahabi, Big data and its technical challenges, Communications of the ACM 57 (7) (2014) 86–94.
- [48] P. Akhtar, J. G. Frynas, K. Mellahi, S. Ullah, Big data-savvy teams’ skills, big data-driven actions and business performance, British Journal of Management 30 (2) (2019) 252–271.
1140
- [49] S. Nadal, V. Herrero, O. Romero, A. Abelló, X. Franch, S. Vansummeren, D. Valerio, A software reference architecture for semantic-aware big data systems, Information and software technology 90 (2017) 75–92.
- 1145 [50] C. Piñeiro, J. Morales, M. Rodríguez, M. Aparicio, E. G. Manzanilla, Y. Koketsu, Big (pig) data and the internet of the swine things: a new paradigm in the industry, Animal frontiers 9 (2) (2019) 6–15.

- [51] S. K. Rahimi, F. S. Haug, Distributed database management systems: A Practical Approach, John Wiley & Sons, 2010.
- 1150 [52] M. Galster, P. Avgeriou, Empirically-grounded reference architectures: a proposal, in: Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS, 2011, pp. 153–158.
- 1155 [53] S. Angelov, P. Grefen, D. Greefhorst, A framework for analysis and design of software reference architectures, Information and Software Technology 54 (4) (2012) 417–431.
- [54] M. Volk, S. Bosse, D. Bischoff, K. Turowski, Decision-support for selecting big data reference architectures, in: International Conference on Business
1160 Information Systems, Springer, 2019, pp. 3–17.
- [55] I. Iso, Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models, International Organization for Standardization 34 (2011) 2910.
- 1165 [56] I. International Organization for Standardization (ISO/IEC), Iso/iec/ieee 42010:2011 (2017).
URL <https://www.iso.org/standard/50508.html>
- [57] A. Josey, M. Lankhorst, I. Band, H. Jonkers, D. Quartel, An introduction to the archimate® 3.0 specification, White Paper from The Open Group
1170 (2016).
- [58] H. Zhu, Software design methodology: From principles to architectural styles, Elsevier, 2005.
- [59] A. R. Hevner, S. T. March, J. Park, S. Ram, Design science in information systems research, MIS quarterly (2004) 75–105.

- 1175 [60] M. Maier, A. Serebrenik, I. Vanderfeesten, Towards a big data reference architecture, University of Eindhoven (2013).
- [61] P. Bengtsson, N. Lassing, J. Bosch, H. van Vliet, Architecture-level modifiability analysis (alma), *Journal of Systems and Software* 69 (1-2) (2004) 129–147.
- 1180 [62] R. Kazman, L. Bass, G. Abowd, M. Webb, Saam: A method for analyzing the properties of software architectures, in: *Proceedings of 16th International Conference on Software Engineering*, IEEE, 1994, pp. 81–90.
- [63] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, J. Carriere, The architecture tradeoff analysis method, in: *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems* (Cat. No. 98EX193), IEEE, pp. 68–78.
- 1185 [64] L. G. Williams, C. U. Smith, Pasasm: a method for the performance assessment of software architectures, in: *Proceedings of the 3rd international workshop on Software and performance*, pp. 179–189.
- 1190 [65] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*, Pearson Education, 2000.
- [66] P. Avgeriou, Describing, instantiating and evaluating a reference architecture: A case study, *Enterprise Architecture Journal* 342 (2003) 1–24.
- [67] M. Derras, L. Deruelle, J. M. Douin, N. Levy, F. Losavio, Y. Pollet, V. Reiner, Reference architecture design: a practical approach, in: *13th International Conference on Software Technologies (ICSOFT)*, SciTePress-Science and Technology Publications, 2018, pp. 633–640.
- 1195 [68] M. Kiran, P. Murphy, I. Monga, J. Dugan, S. S. Baveja, Lambda architecture for cost-effective batch and speed big data processing, in: *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 2785–2792.
- 1200

- [69] D. Quintero, F. N. Lee, et al., IBM reference architecture for high performance data and AI in healthcare and life sciences, IBM Redbooks, 2019.
- [70] B. Levin, Big data ecosystem reference architecture, Microsoft Corporation (2013).
1205
- [71] D. Cackett, Information management and big data, a reference architecture, Oracle: Redwood City, CA, USA (2013).
URL <https://www.oracle.com/technetwork/topics/entarch/articles/info-mgmt-big-data-ref-arch-1902853.pdf>
- [72] P. Viana, L. Sato, A proposal for a reference architecture for long-term archiving, preservation, and retrieval of big data, in: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2014, pp. 622–629.
1210
- [73] J. Kreps, Questioning the lambda architecture, Online article, July 2015 (2014).
1215
URL <https://www.oreilly.com/radar/questioning-the-lambda-architecture/>
- [74] W. Sikora-Wohlfeld, A. Basu, A. Butte, M. Martinez-Canales, Accelerating secondary genome analysis using intel big data reference architecture., Intel (09 2014).
- [75] P. Pääkkönen, D. Pakkala, Reference architecture and classification of technologies, products and services for big data systems, Big data research 2 (4) (2015) 166–186.
1220
- [76] Sap - nec reference architecture for sap hana & hadoop (2016).
URL <https://www.scribd.com/document/418835912/Whitepaper-NEC-SAPHANA-Hadoop>
1225
- [77] M. Bilal, L. O. Oyedele, O. O. Akinade, S. O. Ajayi, H. A. Alaka, H. A. Owolabi, J. Qadir, M. Pasha, S. A. Bello, Big data architecture for construction waste analytics (cwa): A conceptual framework, Journal of Building Engineering 6 (2016) 144–156.

- 1230 [78] L. Heilig, S. Voß, Managing cloud-based big data platforms: a reference architecture and cost perspective, in: Big data management, Springer, 2017, pp. 29–45.
- [79] G. M. Sang, L. Xu, P. d. Vrieze, Simplifying big data analytics systems with a reference architecture, in: Working Conference on Virtual Enterprises, Springer, 2017, pp. 242–249.
- 1235 [80] P. Pääkkönen, D. Pakkala, Extending reference architecture of big data systems towards machine learning in edge computing environments, Journal of Big Data 7 (1) (2020) 1–29.
- [81] C. A. Iglesias, A. Favenza, Á. Carrera, A big data reference architecture for emergency management, Information 11 (12) (2020) 569.
- 1240 [82] I. O. for Standardization (ISO/IEC), Iso/iec tr 20547-1:2020 (2020).
URL <https://www.iso.org/standard/71275.html>
- [83] A. Maamouri, L. Sfaxi, R. Robbana, Phi: A generic microservices-based big data architecture, in: European, Mediterranean, and Middle Eastern Conference on Information Systems, Springer, 2021, pp. 3–16.
- 1245 [84] P. Ataei, A. Litchfield, Neomycelia: A software reference architecture for big data systems, in: 2021 28th Asia-Pacific Software Engineering Conference (APSEC), IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 452–462. doi:10.1109/APSEC53868.2021.00052.
URL <https://doi.ieeecomputersociety.org/10.1109/APSEC53868.2021.00052>
- 1250 [85] Big data is a big deal.
URL <https://obamawhitehouse.archives.gov/blog/2012/03/29/big-data-big-deal>
- [86] W. L. Chang, N. Grady, et al., Nist big data interoperability framework: volume 1, big data definitions (2015).
- 1255

- [87] U. Suthakar, A scalable data store and analytic platform for real-time monitoring of data-intensive scientific infrastructure, Ph.D. thesis, Brunel University London (2017).
- 1260 [88] D. N. B. D. I. Framework, Draft nist big data interoperability framework: Volume 5, architectures white paper survey, NIST Special Publication (2015).
- [89] M. Weyrich, C. Ebert, Reference architectures for the internet of things, IEEE Software 33 (1) (2015) 112–116.
- 1265 [90] P. P. Khine, Z. Wang, A review of polyglot persistence in the big data world, Information 10 (4) (2019) 141.
- [91] G. Márquez, H. Astudillo, Actual use of architectural patterns in microservices-based open source projects, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Ieee, 2018, pp. 31–40.
- 1270 [92] Z. Dehghani, How to move beyond a monolithic data lake to a distributed data mesh (2019).
 URL <https://martinfowler.com/articles/data-monolith-to-mesh.html>