RESEARCH

Application of Microservices Patterns to Big Data Systems

Jane E. Doe[1*] and John R.S. Smith[1,2]

*Correspondence:
jane.e.doe@cambridge.co.uk
[1]Department of Science, University of Cambridge, London, UK
Full list of author information is available at the end of the article

Abstract

Keywords: big data; microservices; microservices patterns; big data architecture; data architecture; data engineering

1 Introduction

Since the dawn of internet and world wide web, humanity has witnessed an unprecedented degree of connection. The proliferation of digital devices pervaded with powerful software applications have created cyber communities that constantly mutate [1, 2]. In a world where we have network infrastructures that can support up to 250Mbps of data transmission, and smart phones and internet of things (IOT) devices that can have processing power of up to 3 Ghz, data becomes ubiquitous, the quantum that lays the foundation of the nexus [3]. According to InternetLiveStates.com [4], only in one second, there are 9,878 tweets sent, 1,138 instagram photos uploaded, 3,117,720 emails sent, 99,738 Google searches made, and 94,144 Youtube videos viewed. That is, if it has taken 5 second to read the preceding paragraph, during that time, 15,588,600 emails are sent.

Driven by the ambition to harness the power of this large amount of data, the term BD was coined [5]. BD initially emerged to address the challenges associated with various characteristics of data such as velocity, variety, volume and variability [2]. BD is the practice of extracting patterns, theories, and predictions from a large set of structured, semi-structured, and unstructured data for the purposes of business competitive advantage [6,7]. BD is a game-changing innovation, heralding the dawn of a new data-oriented industry. Nonetheless, BD is not a magical wand that can enchant any business process. While a lot of opportunities exist in BD, subsuming an emergent and rather high-impacting technology like BD to current state of affairs in organizations, is a daunting task.

According to recent survey from Databricks, only 13% of the organizations excel at delivering on their data strategy [8]. Another survey by NewVantage Partners indicated that only 24% organization have successfully gone data-driven [9]. This survey also states that only 30% of organizations have a well established strategy for their BD endeavour. In addition, surveys from McKinsey & Company [10] and Gartner [11] further support these numbers, which illuminates on the scarcity of successful BD implementations in the industry.

There are various challenge for adoption of BD such as 'data architecture', 'rapid technology change', 'lack of sufficiently skilled data engineers', and 'organization's cultural challenges of becoming data-driven' [2, 12]. Among these challenges, 'data

architecture' is highlighted. A successful BD system is built upon a solid data architecture, serving as a blue print that affects data lifecycle management, guides data integration, control data assets and handle change. Nevertheless, majority of BD systems are developed on-premise as ad-hoc complicated solutions that do not adhere to the practices of software engineering and software architecture [13,14]. As the data ecosystem grows and new technologies and data processing techniques are introduced, the software architect will have a harder time maintaining a solution that addresses the emergent requirements.

To this end, our study aims to address the challenges of data architecture by proposing a reference architecture (RA). RAs are suitable artifacts for development of complex systems as they capture the essence of knowledge for a class of systems and inject it into practices as high-level models that different stakeholders can reflect on. This can potentially create grounds for an immature architecture that is hard to scale, hard to maintain, and raise high-entry blockades [3]. Since the approach of ad-hoc design to BD system development is not desirable and may leave many architects in the dark, novel data architectures that are designed specifically for BD are required. To contribute to this goal, we explore the notion of RAs and present a distributed domain-driven software RA for BD systems. This RA is called Cybermycelium. The contribution of this work is threefold: 1) explicating design theories underlying current BD systems, 2) explicating design theories that generate the artefact's constructs, and 3) the artefact.

2 Background

In this section, we provide a brief discussion on what is known about BD architectures, articulate the research gap and problems that need addressing, and present with the objective of this research.

2.1 Big Data Architectures: State of the art

The available body of knowledge and the knowledge from practice highlight 3 generations of BD architectures;

1 **Enterprise Data Warehouse:** this is perhaps one of the oldest approaches to business intelligence and data crunching and has existed even before the term BD was coined [15]. Usually developed as proprietary software, this data architecture pivot on enterprise data warehouse, extract, transform and load (ETL) jobs, and a data visualization software such as Microsoft Power Business Intelligence (BI). As the data sources and consumers grow, this architecture suffers from hard to main ETL jobs, and visualizations that can be created and understood by a certain group of stakeholders, hindering positive impact of data on business. This also means, new transformations will take longer to be added to the workload, the system is monolithic and hard to scale, and only a few group of hyper-specialized individuals are able to operate the system. Moreover, data warehouses have been designed with different assumptions that cannot effectively handle the characteristics of BD.

2 **Data Lake**: to address the challenges occurred in the first generation of data architectures, a new BD ecosystem emerged. This new ecosystem revolved around a data lake, in a way that there isn't as much transformations on

the data initially, but rather everything is dumped into the data lake and retrieved when necessary. Although data lake architecture reached a higher level of success in comparison to the first generation of data architectures, they are still far from optimal. As data consumer and data providers grow, data engineers will be immensely challenged to avoid creation of data swamp [16], and because there is usually no concept of data owner, the whole stack is usually operated by a group of hyper-specialized data engineers, creating silos, and barriers for gradual adoption. This also means various teams' concerns will often go into data engineers backlog through an intermediary such as business analyst and they will not be in control of how and when they can consume the data they desire. Furthermore, data engineers are usually oblivious of the semantics and value of the data they are processing; they simply do not know how is that data useful or which domain it belongs to. This will overtime decrease the quality of data processing, results in haphazard data management, and make maintenance and data engineering a complicated task.

3 **Cloud Based Solutions:** Given the cost and complexity of running a data lake on-premise alongside the whole data engineering pipeline, and the substantial talent gap currently faced in the market [6], the third generation of BD architectures tend to revolve around as-a-service or on-demand cloud-based solutions. This generation of architecture tends to be leaning towards stream-processing with architectures such as Kappa or Lambda [17], or frameworks that unify batch and stream processing such as Apache Beam [18] or Databricks [19]. This is usually accompanied by cloud storage such as Amazon S3, and streaming technologies such as Amazon Kinesis. Whereas this generation tends to solve various issues regarding the complexity and cost of data handling and digestion, it still suffers from the same fundamental architectural challenges. It does not have clear data domains, a group of siloed hyper-specialized data engineers are running them, and data storage through a monolithic data pipelines soon becomes a choke-point.

To discuss the integral facets that embroil these architectures, one must look at the characteristics of these architectures and the ways in which they achieve their ends. Except for one case [3], 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.

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

1 **Data Ingestion:** system beings to ingest data from all corners of the enterprise, including both transactional, operational and external data. For instance, in a practice management software for veterinaries, data platform can ingest and persist transactional data such as 'user interaction with therapeutics', 'number of animals diagnosed', or 'number of invoices created' and 'medicines dispensed'.

2 **Data Transformation:** data captured from the previous step is then cleansed for duplication, quality, and potentially scrubbed for privacy policies. This data then goes through a multifaceted enrichment process to facilitate

data analysis. For instance, a journey of the veterinary nurse can be captured at every stage, enriched with demographics and animal breed for regression analysis and aggregate views.

3 **Data Serving:** at this stage, data is ready to be served to diverse array of needs ranging from machine learning to marketing analytics, to business intelligence to product analysis and customer journey optimization. In the 'veterinary practice management software' example, the data platform can provide real-time data through event backbone system such as Kafka about customers who have applied and have been dispensed restricted veterinary medicine (RVM) to make sure that these transactions comply with the conditions of the registration of these products.

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 all sorts of data in one architectural construct. This means, data that logically belong to different domains are now all lumped together and crunched in one place, making maintainability and scalability a daunting task [20].

While architectures in software engineering have gone through series of evolution in the industry, adopting a more decentralized and distributed approaches such as microservices architecture, event driven architectures, reactive systems, and domain-driven design [21], the data engineering, and in specific BD ecosystems do not seem to be adopting many of these patterns. Evidence collected from this study have proven that attention to decentralized BD systems, metadata, and privacy is deficient. Therefore, the whole idea of 'monolithic data pipeline architecture with no clearly defined domains and ownership' brings significant challenges to design, implementation, maintenance and scaling of BD systems.

To address these issues, we explore a domain-driven distributed RA for BD systems and propose a RA that addresses some of these challenges. This RA is inspired by the advances in software engineering architectures, and in specific microservices, domain-driven design, and reactive systems.

2.2 Why reference architecture?

To justify why we have chosen RAs as the suitable artefact, first we have to clarify two assumptions;

1 having a sound software architecture is essential to the successful development and maintenance of software systems [22]

2 there exist a sufficient body of knowledge in the field of software architecture to support the development of an effective RA [1]

One of the focal tenets of software architecture is that every system is developed to satisfy a business objective, and that the architecture of the system is a bridge between abstract business goals to concrete final solutions [22]. While the journey of BD can be quite challenging, the good news is that a software RA can be designed, analyzed and documented incorporating best practices, known techniques, and patterns that will support the achievement of the business goals. In this way, the complexity can be absorbed, and made tractable.

Practitioners of complex systems, software engineers, and system designers have been frequently using RAs to have a collective understanding of system components,

functionalities, data-flows and patterns which shape the overall qualities of system and help further adjust it to the business objectives [23,24]. In software product line (SPL) development, RAs are generic artifacts that are configured and instantiated for a particular domain of systems [25]. In software engineering, major IT giants like IBM has referred to RAs as the 'best of best practices' to address unique and complex system development challenges [23]. There is a fair amount of literature on RAs, and whereas different authors definition may vary, they all share the same tenets.

A RA is amalgamation of architectural patterns, standards, and software engineering techniques that bridge the problem domain to a class of solutions. This artefact can be partially or completely instantiated and prototyped in a particular business context together with other supporting artefact to enable its use. RAs are often created from previous RAs [1]. Based on the premises discussed and taking all into consideration, RAs can facilitate the issues of BD architecture and data engineering because of the following reasons;

1 RAs can promote adherence to best practice, standards, specifications and patterns
2 RAs can endow the data architecture team with openness and increase operability, incorporating architectural patterns that ensue desirable predefined quality attributes
3 RAs can be the best initial start to the BD journey, capturing design issues when they are still cheap
4 RAs can bring different stakeholders on the same table and help achieve consensus around major technological constructs
5 RAs can be effective in identifying and addressing cross-cutting concerns
6 RAs can serve as the organizational memory around design decisions, enlightening next subsequent decisions
7 RAs can act as a summary and blueprint in the portfolio of software engineers and software architects, resulting in better dissemination of knowledge

3 Research Methodology

Our research methodology is made up of two major phases. First we explore the body of knowledge in academia and industry to identify architecturally significant requirements (ASR) for BD systems, and secondly we discuss the chosen methodology for developing the artefact

3.1 Requirement Specification

Architecture aims to produces systems that are addressing specific requirements, and one cannot succeed in designing a successful architecture if requirements are unknown [22]. Therefore, in this section, we strive to firmly define the requirements necessary for the development of Cybermycelium. We present with three integral pieces of information:

1 Determining the type of the requirement
2 Identifying the right approach for categorization of the requirements
3 Identifying the right approach for presentation of the requirements

For maximum clarity, we've mapped the following sub-sections against the above mentioned elements.

3.2 Rationale for creating Cybermycelium

The underpinning of our rationale was the challenges that engulfed the current BD architectures discussed in 2.1. In this section, we expand on those issues and explain how our artifact would vary and what challenges it will address. The main goal of Cybermycelium is to introduce a new paradigm that shifts away from monolithic centralized data architectures, into a distributed mesh of data products that communicate through coherent interfaces. This implies change in various aspects. In Cybermycelium, centralized ownership of data by hyper-specialized individuals is broken down to decentralized domains where data is actually produced or used. Cybermycelium shifts away from storing data in monolithic data warehouses and data lakes to distributed data sets that can be connected through standardized interfaces. Cybermycelium aims to shift from centralized top-down data governance, to computer-aided federated construct with policies computationally stored. In Cybermycelium, data is more treated as product rather than an asset to be collected; that is, the provider of the data should be accountable for the quality of data.

3.2.1 Type of requirements

Precursor to theorizing about the potential of Cybermycelium, we needed to define what are the requirements. System and software requirements come in different flavours and can range from a sketch on a napkin to formal (mathematical) specifications. Therefore, we first needed to identify what kind of requirements is the most suitable for the purposes of this study. To answer this question, we first explored the body of evidence to understand the current classification of software requirements.

There's been various attempts to defining and classifying software and system requirements. For instance, Sommerville [26] classified requirements into three levels of abstraction that are namely 1) user requirements, 2) system requirements and 3) design specifications. The author then mapped these requirements against user acceptance testing, integration testing and unit testing.While this could satisfy the requirements of this study, we opted for a more general framework provided by Laplante [27]. In Laplante's approach, requirements are categorized into three categories of 1) functional requirements, 2) non-functional requirements, and 3) domain requirements.

Our objective is to define the high-level requirements of BD systems, thus we do not fully explore 'non-functional' requirements. Majority of non-functional requirements are emerged from the particularities of an environment, such as a banking sector or healthcare, and do not correlate to our study. Therefore, our primary focus is on functional and domain requirements and secondly on non-functional requirements.

3.2.2 Categorizing requirements

After having filtered out the right type of requirement, we then sought for a rigorous and relevant method to categorize the requirements. For this purpose, we followed the well-established categorization method based on BD characteristics, that is the 5Vs. These 5Vs are velocity, veracity, volume, variety and value [28, 29]. Nadal et al. [30] have underpinned their RA on these characteristics and requirements that goes with them. Moreover, NIST BD Public Working Group embarked on a large

scale study to extract requirements from variety of application domains such as healthcare, life sciences, commercial, energy, government, and defense. The result of this study was the formation of general requirements under seven categories. In another effort by Volk et al. [31], nine use cases for BD projects are identified by collecting theories and use cases from the literature and categorizing them using a hierarchical clustering algorithm. Bashari et al. [32] focused on the security and privacy requirements of BD systems, Yu et al. presented the modern components of BD systems [33], Eridaputra et al. [34] created a generic model for BD requirements using goal oriented approaches, and Al-jaroodi et al. [35] investigated general requirements to support BD software development.

We've also studied other RAs developed for BD systems to understand general requirements. In one study, Ataei et al. [36] assessed the body of evidence and presented with a comprehensive list of BD RAs. This study helped us realize the spectrum of BD RAs, how they are designed and the general set of requirements. By analyzing these studies and by evaluating the design and requirement engineering required for BD RAs, we adjusted our initial categories of requirements and added security and privacy to it.

3.2.3 Present requirements

After knowing the type and category of requirements, We looked for a rigorous approach to present these requirements. There are numerous approaches used for software and system requirement representation including informal, semiformal and formal methods. For the purposes of this study, we opted for an informal method because it is a well established method in the industry and academia [37]. Our approach follows the guidelines explained in ISO/IEC/IEEE standard 29148 [38] for representing functional requirements. We have also taken inspiration from Software Engineering Body of Knowledge [39]. However, our requirement representation is organized in term of BD characteristics.

3.3 The artefact development methodology

There are a few studies that have addressed the systematic development of RAs. Cloutier et al. [23] present a high-level model for RA development through collection of contemporary information and capturing the essence of architectural advancements. In another effort, PuLSE-DSSA is proposed by Bayer et al. [40] in the context of product line development and domain engineering. PulSE-DSSA emphasizes on capturing the existing architectural knowledge. Stricker et al. [41] propose a pattern-based approach for creating an RA. This study revolves around software engineering patterns motivated by the work of Gamma et al. [42]; and proposes a structural approach that includes three layers of patterns with well-defined hierarchical relationships. Nakagawa, Martins, Felizardo, and Maldonado [43] propose an approach to RA design outside of product line management context that is concentrated towards aspect-oriented systems.

Galster and Avgeriou [44] propose an empirically grounded RA based on two main facets; Existing RAs in practice and available literature on RAs. Along the same vein, Nakagawa et al. [45] presented ProSA-RA which is a 4 phase methodology that unlike many other methodologies do provide a more comprehensive instructions on

RA evaluation. In addition, this methodology benefits from an ecosystem of complementary constructs that aid in RA design and evaluation such as RAModel [46] and a framework for evaluation of RAs (FERA) [47].In a recent study, Derras et al. [48] propose a schema of practical RA development in the context of software product line and domain engineering. This study is based on capturing knowledge from architectures in practice with attention to variability, configurability and product line development. The findings provide a four-phase process to develop quality driven RAs. This approach is influenced by ISO/IEC 26550 [49].

By analysis and study of all these approaches for design and development of RAs, a common pattern has been witnessed. Whereas some of them are more recent and some belong to years ago, there are commonalities that has been observed. All these approaches are grounded on three main pillars, 1) Existing RAs 2) RAs in literature 3) Architectures in practice. Taking this into consideration and by analyzing the results of the systematic literature review (SLR) conducted by Ataei et al. [1] we found 'Empirically-grounded RAs' proposed by Galster and Avgeriou [44] a suitable methodology, because firstly it's been adopted by many studies, and secondly it's in-line with the goal of our study.

Nevertheless, we did not fully adopt this methodology and rather customized to the needs of this particular research. This is due to some inherent limitations that has been witnessed with the methodology. For instance we could not find a comprehensive guideline on how to identify data sources and how data could be categorized and synthesized into creation of the RA (in the third step of the methodology). Therefore we employed the Nakagawa's information source investigation guidelines and the overall idea of the RAModel. Another limitation we've faced was with evaluation of the RA. As evaluation, second to a sound research methodology is one of the key elements of any good design science research, we had to look for a stronger and more systematic evaluation approach than what is discussed in 'empirically grounded RAs' methodology. For this purpose, and inspired by the works of Angelov et al. [50,51], we first created a prototype of the RA in practice, and then used 'The architecture tradeoff analysis method' (ATAM) [52] to evaluate the artefact.

This research methodology is constituent of 6 phases which are respectively; 1) Decision on the type of the RA 2) Design strategy 3) Empirical acquisition of data 4) Construction of the RA 5) Enable RA with variability 6) Evaluation of the RA. The phrase 'empirically grounded' refers to two major elements; firstly the RA should be grounded in well-established and proven principles; secondly, the RA should be evaluated for applicability and validity. These don't only belong to Galster and Avgeriou's methodology, and other researchers such as Cloutier [23] and Derras et al. [25] have promoted the same ideas.

It is worth mentioning that this methodology is iterative, meaning that the results gained from the evaluation phase (6th phase) determines the subsequent iterations until the design reaches saturation.

3.4 Step1: Decision on type of the RA

Precursor to any effective RA development, is the decision on type of it. The type of the RA is significant, as it illuminates on information to be collected and the construction of the RA in later phases. The selection on the type of RA for the

purposes of this study is based on two dimensions; the classification framework proposed by Angelov et al. [53] and the usage context [54].

Based on the classification framework proposed by Angelov et al. [53], five types of RA are defined. This framework has been developed with the goal of supporting analysis of RAs with regards to context, goal, and the architecture specification/design relationships. It is based on 3 major dimensions namely context, goals, and design, each having their own corresponding sub-dimensions. These dimensions and sub-dimensions are derived by the means of interrogatives (the usage of interrogates is a well-established practice for problem analysis).

The interrogatives 'When', 'Where', and 'Who' have been used to address the 'context', 'Why' has been used to address 'goal', and 'How' and 'What' have been used to address 'design' dimension. The outcome of the study categorizes RAs in two major groups; 1) standardization RAs and 2) Facilitation RAs. This framework has been chosen because it is completely in-line with the purposes of this study and aims to demarcate a clear domain for the RA to be developed. The comprehensive classification of the RAs with examples in practice illuminates on how different RAs are playing roles in the industry and how they are classified. This brings clarity on what should be developed and what boundaries should be drawn.

By reading the results of the recent SLR conducted by Ataei et al. on BD RAs [1], we've added more examples of the RAs on top of what was provided by Angelov [53], and provided an updated list of RA classifications with examples. This list can be found at **??**.

The domain-driven distributed BD RA chosen for the purposes of this study pursues two major goals; 1) supporting the development of BD systems 2) enabling an effective and scalable data architecture. Therefore, the outcome artefact will be a BD RA that is a classical standardization RA designed to be implemented in multiple organizations.

3.5 Step2: Selection of Design Strategy

Angelov et al. [50] and Galster et al.[44] have both presented that RAs can have two major design strategies to them; 1) RAs that are designed from scratch (practice driven), 2) RAs that are based on other RAs (research driven). Designing RAs from scratch is rare, and usually takes place in an emergent domain that have not perceived a lot of attention. On the other hand, most RAs today are the amalgamation of a priori concrete architectures, models, patterns, best practices, and RAs, that together provide a compelling artefact for a class of problems.

RAs developed from scratch tend to create more prescriptive theories, whereas RAs developed based on available body of knowledge tends to provide with more descriptive design theories. The RA designed for the purposes of this study is a research-based RA based on existing RAs, concrete architectures, and best practices.

3.6 Step 3: Empirical Acquisition of Data

As aforementioned, due to the limitation witnessed by this research methodology, we have augmented this phase, and increased the systematicity and transparency of data collection and synthesis through various academic methods such as SLR.

This phase is made up of three major undertakings; 1) identification of data sources; 2) capturing data sources; 3) synthesis of data sources.

3.6.1 Identification of data sources

To identify suitable data sources, we've employed the first step of ProSA-RA methodology 'information source investigation'. This step is an endeavour to capture focal and ancillary knowledge and theories that revolve around the target domain, and lay the ground of RA development.

To unearth the architectural quanta, and to highlight gradations between various approaches to BD system development, we've selected most relevant sources as the followings;

1 **Practice-led conferences:** given that majority of recent advancements for emerging technologies such as microservices architecture [55, 56, 56] and BD are coming from virtually hosted practice-led conferences, we've chosen some of the best conferences hold world-wide for the purposes of data collection. These conferences are 1) Qcon [57] 2) State of Data Mesh by ThoughtWorks [58] 3) Worldwide Software Architecture Summit'21 [59] and 4) Kafka Summit Europe 2021 [60]. Our objective was to capture the frontiers of software architecture and emerging approaches currently being practiced in IT giants such as Google, Facebook and Netflix. Among all the speech in these conferences, we looked for topics that entailed or were related to the keywords 'emergent software architecture trends', 'distributed software architecture', 'BD software architecture' and 'domain-driven design'. We used the software Nvivo to code the transcripts from the conference videos. We used the aforementioned keywords as the codes and associated different texts, summative, essence-capturing sentences, and evocative attributes to them.

2 **Publications:** in order to capture evidence from the body of knowledge, we conducted a SLR, following the guidelines of PRISMA presented by Moher et al. [61], and Kitchenham et al. [62]. Although PRISMA is a comprehensive guidelines for conducting a SLR, it is derived from the healthcare community and is driven by assumptions that may not be thoroughly relevant to software engineering and information system researchers. To this end, we adopted some guidelines from Kitchenham et al. for evidence based software engineering.
The main objective of this SLR was to highlight common architectural constructs found among all the BD RAs. This SLR is build on top of our recent work [1] that covered all the RAs by 2020.
The initial SLR included IEEE Explore, ScienceDirect, SpringerLink, ACM library, MIS Quarterly, Elsevier, AISel as well as citation databases such as Scopus, Web of Science, Google Scholar, and Research Gate. The SLR search keywords used were 'Big Data Reference Architectures', 'Reference Architectures in the domain of Big Data', and 'Reference Architectures and Big Data'. We followed the same methodology, but this time for the years 2021 and 2022. Our aim was to find out if there has been any new BD RA published.
By the result of this SLR, we've found 3 more BD RAs [3, 63, 64] and we've added two new standards [65, 66] to further solidify our study. Converging these new SLR with the old, covering the years 2010-2022, we've pooled 89 literature in the primary phase, and another 10 by snowballing and citation searching. These 99 literature then went through our inclusion, exclusion criteria. These criteria are as blow;
Inclusion criteria:

(a) Primary and secondary studies between Jan 1st 2020 and Sep 1st 2022 focused on the topics of BD RA, BD architecture, and BD architectural components
(b) Research that indicates the current state of RAs in the field of BD and demonstrates possible outcomes
(c) Studies that are scholarly publications, books, book chapters, thesis, dissertations, or conference proceedings
(d) Grey literature such as white paper that includes extensive information on BD RAs

exclusion criteria:

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

The screening process was conducted by each researcher separately. Disagreement among researchers were resolved using Krippendorff's alpha [67]. 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. Our α value was within the acceptable range (above 80).

After excluding papers based on inclusion and exclusion criteria, and as suggested by Kitchenham et al. [62], we assessed studies based on their quality. For this purposes, and inspired by Critical Appraisal Skills Programme CASP [68], we developed our quality framework based on 7 criteria. These 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:*
   (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 ?
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 ?
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 were 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 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.

3.6.2 Data Synthesis

After pooling the studies, 13 studies have been removed based on inclusion and exclusion criteria. From there on, 76 studies have been assessed for eligibility based on the quality framework and the inclusion criteria. The result of this process handed over 63 studies from this branch (identification of studies visa database and registers). From the other branch (identification of data via other methods), 10 records identified through citation searching. These reports have been assessed through the same quality framework, inclusion and exclusion criteria, yielding 5 studies. Together 68 studies pooled for this SLR as depicted in Figure 1.

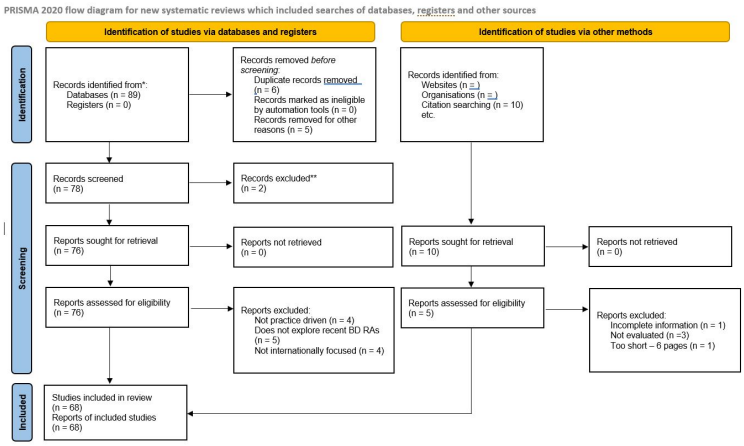

Figure 1: PRISMA flowchart

These 68 studies are comprising of journal papers, conference papers, book chapters, tech reports, tech surveys white papers, standards, master dissertation, and PhD thesis. Out of the pool of these studies, 39.4% are from IEEE Explore, 4.4% are from ScienceDirect, 23.5% are from Springerlink, 13.2% are from ACM, and 29.4% are from other sources such as citation search, Google Scholar and Research Gate. 30 journal articles, 14 conference papers, 6 whitepapers, 2 ISO standards, 14 book chapters, and 2 postgraduate studies have been selected. 26% of these studies are from the year 2010-2013, 33% are from the years 2013-2015, and 51% are from the years 2016-2022. These stats are portrayed in Figure 2.

By this stage, the research objective is set, studies are pooled, assessed and refined, thus the research embarked on the actual synthesis of data. For this purpose, we employed thematic synthesis presented by Cruzes et al. [69]. An integral element of this phase is data extraction, in which the essence of the studies are obtained in an explicit and consistent manner. We opted for an integrated approach to coding [70] using the software Nvivo [71]. All the keywords aforementioned has been created as

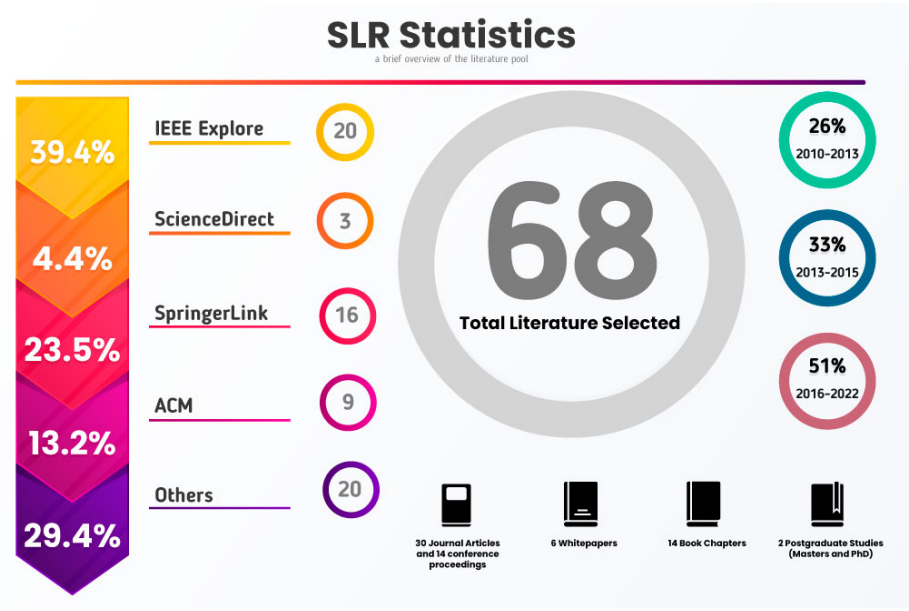

Figure 2: SLR statistics

nodes in the software, which are then associated to relevant sentences in studies. After coding all the studies, the findings have been synthesized to create theories, which in turn emerged themes and patterns. The findings gained from this SLR grounded the foundation for various aspect of the SLR development. To increase transparency, RAs and standards found as the result of this SRL is presented in **??**.

Appendix

Author details
[1]Department of Science, University of Cambridge, London, UK. [2]Institute of Biology, National University of Sciences, Kiel, Germany.

References

1. Ataei P, Litchfield AT. Big Data Reference Architectures, a systematic literature review [Journal Article]. Australasian Conference on Information Systems. 2020.
2. Rad BB, Ataei P. The big data Ecosystem and its Environs [Journal Article]. International Journal of Computer Science and Network Security (IJCSNS). 2017;17(3):38.
3. Ataei P, Litchfield A. NeoMycelia: A software reference architecturefor big data systems. In: 2021 28th Asia-Pacific Software Engineering Conference (APSEC). Los Alamitos, CA, USA: IEEE Computer Society; 2021. p. 452-62. Available from: https://doi.ieeecomputersociety.org/10.1109/APSEC53868.2021.00052.
4. Stats IL. Stats IL, editor. Internet live stats. internetLiveStats; 2022. Available from: https://www.internetlivestats.com/.
5. Lycett M. Lycett M, editor. 'Datafication': making sense of (big) data in a complex world. Taylor & Francis; 2013.
6. Rada BB, Ataeib P, Khakbizc Y, Akbarzadehd N. The Hype of Emerging Technologies: Big Data as a Service [Journal Article]. International Journal of Control Theory and Applications. 2017.
7. Huberty M. Awaiting the second big data revolution: from digital noise to value creation [Journal Article]. Journal of Industry, Competition and Trade. 2015;15(1):35-47.
8. technology review insights in partnership with Databricks M. technology review insights in partnership with Databricks M, editor. Building a high-performance data organization. Databricks; 2021. Available from: https://databricks.com/p/whitepaper/mit-technology-review-insights-report.
9. Partners N. Partners N, editor. Big Data and AI Executive Survey 2021. NewVantage Partners; 2021. Available from: https://www.supplychain247.com/paper/big_data_and_ai_executive_survey_2021/pragmadik.
10. Analytics M. The age of analytics: competing in a data-driven world. Technical report, San Francisco: McKinsey & Company; 2016.
11. Nash H. CIO SURVEY 2015 [Journal Article]. Association with KPMG. 2015.

12. Singh N, Lai KH, Vejvar M, Cheng T. Big data technology: Challenges, prospects and realities [Journal Article]. IEEE Engineering Management Review. 2019.
13. Gorton I, Klein J. Distribution, Data, Deployment [Journal Article]. STC 2015. 2015:78.
14. Nadal S, Herrero V, Romero O, Abelló A, Franch X, Vansummeren S, et al. A software reference architecture for semantic-aware Big Data systems [Journal Article]. Information and software technology. 2017;90:75-92.
15. Leonard EM. Design and implementation of an enterprise data warehouse. Marquette University; 2011.
16. Brackenbury W, Liu R, Mondal M, Elmore AJ, Ur B, Chard K, et al. Draining the data swamp: A similarity-based approach. In: Proceedings of the workshop on human-in-the-loop data analytics; 2018. p. 1-7.
17. Lin J. The lambda and the kappa. IEEE Internet Computing. 2017;21(05):60-6.
18. Beam A. Beam A, editor. Apache Beam. Apache; 2022. online. Available from: https://beam.apache.org/.
19. Databricks. Databricks. Databricks; 2022. online. Available from: https://databricks.com/.
20. Dehghani Z. Fowler M, editor. How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. martinfowler.com; 2019. Available from: https://martinfowler.com/articles/data-monolith-to-mesh.html.
21. Alshuqayran N, Ali N, Evans R. A systematic mapping study in microservice architecture. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA). IEEE; 2016. p. 44-51.
22. Len Bass RK Dr Paul Clements. Software Architecture in Practice (SEI Series in Software Engineering) 4th Edition. Addison-Wesley Professional; 4th edition; 2021.
23. Cloutier R, Muller G, Verma D, Nilchiani R, Hole E, Bone M. The concept of reference architectures [Journal Article]. Systems Engineering. 2010;13(1):14-27.
24. Kohler J, Specht T. 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. p. 38-70.
25. Derras M, Deruelle L, Douin JM, Levy N, Losavio F, Pollet Y, et al. Reference Architecture Design: A Practical Approach. In: ICSOFT; 2018. p. 633-40.
26. Sommerville I. Software Engineering, 9/E. Pearson Education India; 2011.
27. Laplante PA. Requirements engineering for software and systems. Auerbach Publications; 2017.
28. Bughin J. Big data, Big bang? [Journal Article]. Journal of Big Data. 2016;3(1):2.
29. Rad BB, Ataei P. The big data ecosystem and its environs. International Journal of Computer Science and Network Security (IJCSNS). 2017;17(3):38.
30. Nadal S, Herrero V, Romero O, Abelló A, Franch X, Vansummeren S, et al. A software reference architecture for semantic-aware Big Data systems. Information and software technology. 2017;90:75-92.
31. Volk M, Staegemann D, Trifonova I, Bosse S, Turowski K. Identifying similarities of big data projects–a use case driven approach. IEEE Access. 2020;8:186599-619.
32. Bashari Rad B, Akbarzadeh N, Ataei P, Khakbiz Y. Security and privacy challenges in big data era. International Journal of Control Theory and Applications. 2016;9(43):437-48.
33. Yu JH, Zhou ZM. Components and development in Big Data system: A survey. Journal of Electronic Science and Technology. 2019;17(1):51-72.
34. Eridaputra H, Hendradjaya B, Sunindyo WD. Modeling the requirements for big data application using goal oriented approach. In: 2014 international conference on data and software engineering (ICODSE). IEEE; 2014. p. 1-6.
35. Al-Jaroodi J, Mohamed N. Characteristics and requirements of big data analytics applications. In: 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC). IEEE; 2016. p. 426-32.
36. Ataei P, Litchfield AT. Big data reference architectures, a systematic literature review. Australasian Conference on Information Systems. 2020.
37. Kassab M, Neill C, Laplante P. State of practice in requirements engineering: contemporary data. Innovations in Systems and Software Engineering. 2014;10(4):235-41.
38. 29148:2018 I. 29148:2018 I, editor. ISO/IEC 29148:2018. ISO/IEC/IEEE; 2018. Available from: https://www.iso.org/standard/72089.html.
39. Abran A, Moore JW, Bourque P, Dupuis R, Tripp L. Software engineering body of knowledge. IEEE Computer Society, Angela Burgess. 2004:25.
40. Bayer J, Forster T, Ganesan D, Girard JF, John I, Knodel J, et al. Definition of reference architectures based on existing systems. Fraunhofer IESE, March. 2004.
41. Stricker V, Lauenroth K, Corte P, Gittler F, De Panfilis S, Pohl K. Creating a reference architecture for service-based systems–a pattern-based approach. In: Towards the Future Internet. IOS Press; 2010. p. 149-60.
42. Gamma E, Helm R, Johnson R, Johnson RE, Vlissides J, et al. Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH; 1995.
43. Nakagawa EY, Martins RM, Felizardo KR, Maldonado JC. Towards a process to design aspect-oriented reference architectures. In: XXXV Latin American Informatics Conference (CLEI) 2009; 2009. p. 0.
44. Galster M, Avgeriou P. 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. p. 153-8.
45. Nakagawa EY, Guessi M, Maldonado JC, Feitosa D, Oquendo F. Consolidating a process for the design, representation, and evaluation of reference architectures. In: 2014 IEEE/IFIP Conference on Software Architecture. IEEE; 2014. p. 143-52.
46. Nakagawa EY, Oquendo F, Becker M. Ramodel: A reference model for reference architectures. In: 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture. IEEE; 2012. p. 297-301.
47. Santos JFM, Guessi M, Galster M, Feitosa D, Nakagawa EY. A Checklist for Evaluation of Reference Architectures of Embedded Systems (S). In: SEKE. vol. 13; 2013. p. 1-4.
48. Derras M, Deruelle L, Douin JM, Levy N, Losavio F, Pollet Y, et al. Reference Architecture Design: a practical approach. In: 13th International Conference on Software Technologies (ICSOFT). SciTePress-Science and Technology Publications; 2018. p. 633-40.

49. WG I. ISO/IEC 26550: 2015-Software and systems engineering–Reference model for product line engineering and management. ISO/IEC, Tech Rep. 2015.
50. Angelov S, Trienekens JJ, Grefen P. Towards a method for the evaluation of reference architectures: Experiences from a case. In: European Conference on Software Architecture. Springer; 2008. p. 225-40.
51. Angelov S, Trienekens JJ, Grefen P. Extending and adapting the architecture tradeoff analysis method for the evaluation of software reference architectures. Information Systems IE&IS. 2014.
52. Kazman R, Klein M, Barbacci M, Longstaff T, Lipson H, Carriere J. The architecture tradeoff analysis method. In: Proceedings. fourth ieee international conference on engineering of complex computer systems (cat. no. 98ex193). IEEE; 1998. p. 68-78.
53. Angelov S, Grefen P, Greefhorst D. 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. p. 141-50.
54. Angelov S, Grefen P. An e-contracting reference architecture. Journal of Systems and Software. 2008;81(11):1816-44.
55. Gan Y, Zhang Y, Cheng D, Shetty A, Rathi P, Katarki N, et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems; 2019. p. 3-18.
56. Laigner R, Zhou Y, Salles MAV, Liu Y, Kalinowski M. Data Management in Microservices: State of the Practice, Challenges, and Research Directions. arXiv preprint arXiv:210300170. 2021.
57. QCON. QCON, editor. QCon Software Conferences. QCON; 2022. Available from: https://qconferences.com/.
58. ThoughtWorks. ThoughtWorks, editor. State of Data Mesh 2022. ThoughtWorks; 2022. Available from: https://www.thoughtworks.com/about-us/events/state-of-data-mesh-2022.
59. Geekle. Geekle, editor. Worldwide Software Architecture Summit'21. Geekle; 2021. Available from: https://events.geekle.us/software_architecture/.
60. KafkaSummit. KafkaSummit, editor. Kafka Summit Europe 2021. Confluent; 2021. Available from: https://www.confluent.io/events/kafka-summit-europe-2021/.
61. Moher D, Shamseer L, Clarke M, Ghersi D, Liberati A, Petticrew M, et al. Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement. Systematic reviews. 2015;4(1):1-9.
62. Kitchenham BA, Budgen D, Brereton P. Evidence-based software engineering and systematic reviews. vol. 4. CRC press; 2015.
63. Castellanos C, Perez B, Correal D. Smart transportation: A reference architecture for big data analytics. In: Smart Cities: A Data Analytics Perspective. Springer; 2021. p. 161-79.
64. Sang GM, Xu L, Vrieze Pd. Simplifying big data analytics systems with a reference architecture. In: Working Conference on Virtual Enterprises. Springer; 2017. p. 242-9.
65. for Standardization (ISO/IEC) IO. for Standardization (ISO/IEC) IO, editor. ISO/IEC 20546:2019. ISO/IEC; 2019. Available from: https://www.iso.org/standard/68305.html.
66. for Standardization (ISO/IEC) IO. for Standardization (ISO/IEC) IO, editor. ISO/IEC TR 20547-1:2020. ISO/IEC; 2020. Available from: https://www.iso.org/standard/71275.html.
67. Krippendorff K. Computing Krippendorff's alpha-reliability. Scholary commons. 2011.
68. Programme CAS. Programme CAS, editor. Critical Appraisal Skills Programme. Critical Appraisal Skills Programme; 2022. Available from: https://casp-uk.net/casp-tools-checklists/.
69. Cruzes DS, Dyba T. Recommended steps for thematic synthesis in software engineering. In: 2011 international symposium on empirical software engineering and measurement. IEEE; 2011. p. 275-84.
70. Lofland J, Lofland LH. Analyzing social settings. Wadsworth Pub Belmont, CA, USA; 1971.
71. NVIVO. NVIVO, editor. Unlock insights in your data with the best qualitative data analysis software. NVIVO; 2022. Available from: https://www.qsrinternational.com/nvivo-qualitative-data-analysis-software/home.