Highlights

Terramycelium: A Complex Adaptive Reference Architecture for Big Data Sytems

Author One, Author Two, Author Three

- Research highlight 1
- Research highlight 2

Terramycelium: A Complex Adaptive Reference Architecture for Big Data Sytems

Author One[a], Author Two[b], Author Three[a,b]

[a]*Department One, Address One, City One, 00000, State One, Country One*
[b]*Department Two, Address Two, City Two, 22222, State Two, Country Two*

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: keyword one, keyword two
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

The advent of the internet and widespread use of digital devices have sparked a profound shift in connectivity and data creation, leading to an era marked by a rapid growth in data. This period is characterised by the extensive expansion of data, which presents difficulties for traditional data processing systems and necessitates inventive methods in data architecture [6, 46]. The vast amount, variety, and rapid generation of data in the current digital environment necessitate innovative solutions, particularly in the field of Big Data (BD).

Data needs have dramatically evolved, transitioning from basic business intelligence (BI) functions, like generating reports for risk management and compliance, to incorporating machine learning across various organisational facets [9]. These range from product design with automated assistants to personalised customer service and optimised operations. Also, as machine learning becomes more popular, application development needs to change from rule-based, deterministic models to more flexible, probabilistic models that can handle a wider range of outcomes and need to be improved all the time with access to the newest data. This evolution underscores the need to reevaluate and simplify our data management strategies to address the growing and diverse expectations placed on data.

Currently, the success rate of BD projects is low. Recent surveys have identified the fact that current approaches to big data do not seem to be effectively addressing these expectations. According to a survey conducted by [21], only 13% of organisations are highly successful in their data strategy. Additionally, a report by NewVantage Partners reveals that only 24% of organisations have successfully converted to being data-driven, and a measly 30% have a well-established big data strategy. These observations, additionally corroborated by research conducted by McKinsey & Company (analytics2016age) and Gartner (Nash), emphasise the difficulties of successfully using big data in the industry. These difficulties include the lack of a clear understanding of how to extract value from data, the challenge of integrating data from multiple sources, data architecture, and the need for skilled data analysts and scientists.

Without a well-established big data strategy, companies may struggle to navigate these challenges and fully leverage the potential of their data. One effective artefact to overcome some of these challenges is Reference Architectures (RAs) [18]. RAs extract the essence of the practice as a series of patterns and architectural constructs and manifest it through high-level semantics. This allows stakeholders to refrain from reinventing the wheel and instead focus on utilising existing knowledge and best practices to harness the full potential of their data. While there are various BD RAs available to help practitioners design their BD systems, these RAs are overly cetnralised, lack attention to cross-cutting concerns such as privacy, security, and metadata, and may not effectively handle the proliferation of data sources and consumers.

To this end, this study presents TerrMycelium, a distributed RA designed specifically for BD systems with a focus on domain-driven design. TerrMycelium seeks to surpass the constraints of current RAs by utilising domain-driven and distributed approaches derived from contemporary software engineering. This method aims to improve the ability of BD systems to scale, be maintained, and evolve, surpassing the constraints of traditional monolithic data architectures.

The paper is structured as follows: Section 2 provides an overview of the foundational concepts and technologies pertinent to BD reference architecture, aiming to forge a conceptual framework that is required for this paper. An overview of the existing research on the topic is presented in Section 3. The significance of reference architectures in the context of big data is explored in Section 4. Section 5 details the software and system requirements necessary for implementing the proposed architecture. Section 6 delves into the theoretical foundation underpinning the challenges in contemporary big data systems. The design and development of the TerrMycelium artifact are described in Section 7. Section 8 examines the evaluation findings, their implications, limitations, and relevance to existing and future research. Finally, Section 9 summarizes the main contributions of the study, its practical implications, and suggests directions for future research.

2. Background

This section provides foundational definitions essential for comprehending the nuances of the research. This chapter aims to create the conceptual framework necessary to understand the terminology used in the thesis.

2.1. What is Big Data?

To define BD within the scope of this research, various academic definitions have been examined. Kaisler et al. [31] define BD as "the amount of data which is beyond technology's capability to store, manage and process efficiently". Srivastava [57] state that BD pertains to "the use of large data sets to handle the collection or reporting of data that serves various recipients in decision making".

Sagiroglu and Sinanc [53] describe BD as "a term for massive data sets having large, more varied and complex structure with the difficulties of storing, analyzing and visualizing for further processes or results".

Drawing from these definitions, BD in this research is conceptualised as the endeavour to discern patterns from vast amounts of data for the objectives of advancement, governance, and predictive analysis in domain-specific applications.

2.2. The Value of Big Data

The significance and value derived from BD remain pronounced [8]. Extensive discussions on the concept permeate reports, statistics, researches, and conferences [16]. Notably, prominent companies like Google, Facebook, Netflix, and Amazon have propelled this momentum with substantial investments in BD initiatives [48].

A compelling illustration of the tangible benefits that BD offers can be seen in the Netflix Prize recommender system. This system capitalized on a diverse array of data sources, including user queries, ratings, search terms, and various demographic indicators [4]. By implementing BD-powered recommendation algorithms, Netflix not only achieved a considerable increase in TV series consumption but also observed certain series experiencing up to a fourfold surge in viewership [4].

In a healthcare context, the Taiwanese government adeptly merged its national health insurance database with customs and immigration datasets as part of a BD strategy [63]. The resulting real-time alerts during clinical visits, informed by clinical symptoms and travel history among other factors, facilitated proactive identification of potential COVID-19 cases. Such strategic data-driven initiatives significantly bolstered Taiwan's effectiveness in managing the epidemic.

In the realm of energy exploration, Shell harnesses BD to optimise the decision-making process and reduce exploration costs [39]. By uploading and comparing data from various drilling sites globally, decisions pivot towards locations that mirror those with confirmed abundant resources. Prior to BD's integration, identifying energy resources presented formidable challenges. Traditional exploration methods, which relied heavily on deciphering waves of energy travelling through the earth's crust, were not only error-prone but also exorbitantly expensive and time-intensive.

Similarly, Rolls Royce capitalises on BD's potential by collecting intricate performance data from sensors fitted on its aircraft products [39]. Such data, transmitted wirelessly, provides insights into key operational phases, from take-off to maintenance. Leveraging this wealth of information, Rolls Royce can more accurately detect degradation, enhance diagnostic and prognostic accuracy, and effectively reduce false positives.

2.3. Reference Architectures

RAs have emerged as pivotal elements in contemporary system development, guiding the construction, maintenance, and evolution of increasingly complex systems [18]. They offer a clear depiction of the essential components of a system and the interactions necessary to realize overarching objectives.

This clarity fosters the creation of manageable modules, each addressing distinct aspects of complex problems, and provides a high-level platform for stakeholders to engage, contribute, and collaborate.

The significance of RAs in IT is underscored by the success of widely adopted technologies like OAuth [43] and ANSI-SPARC architecture [5], which have their origins in well-structured RAs. These RAs not only define the qualities of a system but also shape its evolution. While every system inherently possesses an architecture, RAs distinguish themselves by focusing on more abstract qualities and higher levels of abstraction. They aim to capture the essence of practice and integrate well-established patterns into cohesive frameworks, encompassing elements, properties, and interrelationships.

The significance of RAs in BD is multifaceted, encompassing aspects like communication, complexity control, knowledge management, risk mitigation, fostering future architectural visions, defining common ground, enhancing understanding of BD systems, and facilitating further analysis.

2.4. Microservices and Decentralised, Distributed Architectures

Microservices architecture, representing an evolution in software engineering, involves structuring applications as a collection of loosely coupled services [14]. This approach, emerging from the broader concept of Service Oriented Architectures (SOA), focuses on developing small, independently deployable modules that collaborate to form a comprehensive application. As Newman [41] elucidates, microservices enhance scalability, facilitate continuous deployment, and foster a more agile development environment. They enable teams to develop, deploy, and scale parts of a system independently, thus improving overall system resilience and facilitating rapid adaptation to changing demands.

Decentralised and distributed architectures are integral to the modern computing landscape, characterised by systems spread across multiple nodes, often in different geographic locations. This architectural style, as highlighted by Richards [51], mitigates the limitations of traditional monolithic structures, offering enhanced scalability, fault tolerance, and flexibility. In distributed systems, data and processing are dispersed across multiple nodes, which interact with each other to perform tasks, as discussed by Coulouris et al. [20]. Decentralisation in this context implies the lack of a single controlling node, instead opting for a more democratic and resilient network structure.

The convergence of microservices within these architectures represents a progressive step in software engineering. It reflects a move towards systems that are not only distributed in nature but also modular and adaptable. This architectural approach aligns well with contemporary demands for systems that are scalable, resilient, and capable of leveraging the distributed nature of modern computing environments. The adoption of microservices in decentralised, distributed architectures heralds a new era in software development, where flexibility, scalability, and resilience are paramount.

3. Related Work

This section reviews seminal works in BD RAs, delineating their scope, methodologies, and inherent limitations, thereby justifying the novel approach of this study's domain-driven distributed RA, *Terramycelium*.

Lambda architecture [35] and Kappa architecture [36] represent pivotal industry contributions to BD RAs, introducing foundational paradigms for data processing. However, these architectures have faced criticism for their lack of comprehensive data management strategies, particularly regarding data quality, security, and metadata [6]. This gap is further evidenced in domain-specific RAs like those proposed by [45] for healthcare, which, while addressing domain-specific needs, often overlook cross-cutting concerns such as privacy and interoperability.

Academic efforts, as in [61] and [44], have aimed to broaden the conceptual understanding of BD systems, proposing RAs that attempt to encapsulate more holistic views of data analytics ecosystems. Yet, these proposals frequently fall short in addressing the dynamic and distributed nature of modern data landscapes, particularly in terms of scalability and modifiability.

The limitations of current BD RAs, as summarized in the works of Ataei and Litchfield, highlight a common trend: a pronounced reliance on monolithic data pipeline architectures. This reliance is manifest in the inadequacy of existing RAs to effectively manage data quality, security, privacy, and metadata. Furthermore, the monolithic nature of these architectures often results in scalability and modifiability issues, as they struggle to adapt to the evolving data and technology landscapes.

Against this backdrop, *Terramycelium* emerges as a novel RA for BD systems. By embracing a domain-driven, distributed approach, *Terramycelium* addresses the critical limitations identified in existing RAs. Unlike its predecessors, which often encapsulate data management within rigid, monolithic structures, *Terramycelium* advocates for decentralized data stewardship and a modular architecture. This approach not only enhances scalability and flexibility but also ensures that cross-cutting concerns such as security, privacy, and data quality are inherently integrated into the system's design.

In essence, *Terramycelium* represents a significant departure from traditional BD RAs. By prioritizing domain-driven design and distributed processing, it offers a scalable and adaptable framework that aligns more closely with contemporary data management needs and the principles of modern software architecture. In doing so, *Terramycelium* not only addresses the limitations of existing RAs but also positions itself as a vanguard in the evolution of BD system design, paving the way for future research and development in this critical field.

4. Why Reference Architectures

Conceptualisation of the system as an RA, helps with understanding of the system's key components, behaviour, composition and evolution of it, which in turn affect quality attributes such as maintainability, scalability and performance [19].

Therefore, RAs can be a good standardisation artefact and a communication 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.

The practice of leveraging RAs for both system conceptualisation and as a standardisation artefact is not new to practitioners of complex systems. In Software Product Line (SPL) development, RAs are utilised as generic artifacts that are instantiated and configured for a particular domain of systems [24].

In software engineering, renowned IT corporations such as IBM have consistently advocated for RAs, considering them exemplary practices in addressing intricate system design challenges [18]. Similarly, in the realm of international standards, RAs frequently serve as tools to standardize emerging domains.

Morover, the BS ISO/IEC 18384-1 RA for service-oriented architectures [29] demonstrates the utility of RAs in creating standardized frameworks in specific fields.

5. Software and System Requirements

According to Wieringa[64], the requirement specification phase is an essential step in developing a new IS or artefact. This phase involves identifying the requirements that the artefact must satisfy to meet the needs of stakeholders and achieve the desired outcomes.

Wieringa's methodology distinguishes between functional requirements, which describe what the artefact should do, and non-functional requirements, which describe how the artefact should do it. Functional requirements are typically expressed as use cases, which describe the specific interactions between users and the system. Non-functional requirements may include performance requirements, security requirements, and usability requirements.

The requirement specification designed for this study is made up of the following phases:

1. Determining the type of the requirements
2. Determining the relevant requirements and
3. Identifying the right approach for categorisation of the requirements
4. Identifying the right approach for the presentation of the requirements

Determining the type of the requirements:. Defining and classifying software and system requirements is a common subject of debate. Sommerville[55] classify requirements as three levels of abstraction; user requirements, system requirements, and design specifications. These abstractions are then mapped against user acceptance testing, integration testing, and unit testing. Nevertheless, in this study, a more general framework provided by Laplante is adopted. The adopted approach provides three types of requirements: functional, non-functional, and domain requirements. The objective of this step is to define the high-level requirements of BD systems, therefore the main focus is on non-functional and domain requirements.

Determining the relevant requirements:. In an extensive effort, the NIST Big Data Public Working Group embarked on a large-scale study to extract requirements from a variety of application domains such as Healthcare, Life Sciences, Commercial, Energy, Government, and Defense [15]. The result of this study is the formation of general requirements under seven categories. In addition, Volk et al. categorize nine use cases of BD projects sourced from published literature using a hierarchical clustering algorithm.

Bashari Rad et al.[12] focus on security and privacy requirements for BD systems, Yu and Zhou present modern components of BD systems, using goal-oriented approaches, Eridaputra et al. created a generic model for BD requirements, and Al-Jaroodi and Mohamed investigate general requirements to support BD software development.

By analyzing the result of the first SLRs, the studies discussed above and by evaluating the design and requirement engineering required for BD RAs, a set of high-level requirements based on BD characteristics is established.

Identifying the right approach for categorisation of the requirements:. After clarifying the type of requirements and the relevant requirements, current BD RAs and their requirements have been assessed to increase understanding of the available BD requirement categorisation methods. A common theme among these studies connected the dot towards a common approach to classifying requirements. This approach is categorised through BD characteristics such as velocity, veracity, volume, variety, value, security and privacy [8, 11, 47, 17].

Identifying the right approach for the presentation of the requirements:. Then, a rigorous approach to present software and system requirements that offers informal methods of model verification is identified because such methods are well established in the industry and academia [33]. The approach for representing functional requirements follows the guidelines in *ISO/IEC/IEEE standard 29148* [30].

The requirements representation is organized in system modes, where the major components of the system and then the requirements are described. This approach is inspired by the requirement specification expressed for NASA Wide-field InfraRed Explorer (WIRE) system [38] and the Software Engineering Body of Knowledge (SEBoK) [2]. These requirements are described in Table 1.

Table 1: Terramycelium software and system requirements

Volume	**Vol-1)** System needs to support asynchronous, streaming, and batch processing to collect data from centralised, distributed, and other sources **Vol-2)** System needs to provide scalable storage for massive data sets

Velocity	**Vel-1)** System needs to support slow, bursty, and high throughput data transmission between data sources **Vel-2)** System needs to stream data to data consumers in a timely manner **Vel-3)** System needs to be able to ingest multiple, continuous, time-varying data streams **Vel-4)** System shall support fast search from streaming and processed data with high accuracy and relevancy **Vel-5)** System should be able to process data in a real-time or near real-time manner
Variety	**Var-1)** System needs to support data in various formats ranging from structured to semi-structured and unstructured data **Var-2)** System needs to support aggregation, standardization, and normalization of data from disparate sources **Var-3)** System shall support adaptations mechanisms for schema evolution **Var-4)** System can provide mechanisms to automatically include new data sources
Value	**Val-1)** System needs to able to handle compute-intensive analytical processing and machine learning techniques **Val-2)** System needs to support two types of analytical processing: batch and streaming **Val-3)** System needs to support different output file formats for different purposes **Val-4)** System needs to support streaming results to the consumers
Security & Privacy	**SaP-1)** System needs to protect and retain the privacy and security of sensitive data **SaP-2)** System needs to have access control, and multi-level, policy-driven authentication on protected data and processing nodes.
Veracity	**Ver-1)** System needs to support data quality curation including classification, pre-processing, format, reduction, and transformation **Ver-2)** System needs to support data provenance including data life cycle management and long-term preservation.

6. Theory

In mathematical terms, an inflection point is a significant juncture where the curvature of a curve changes direction, signifying the transition from one behaviour to another. This pivotal point is marked by a dissolution of the prior shape and the emergence of a novel form. Today, there are empirical signals and derivers that point in the direction of a seismic shift [1]. s A glimpse of the New Vantage Partners report in 2023 communicates that despite potential economic headwinds, the investments in data are growing and remain strong. Nevertheless, the report also communicates that only 23.9% of companies identify themselves as data-driven.

Before discussing these design theories, a clear communication method for architectural constructs is essential. This study employs the architecture definition from ISO/IEC 42010 [58], described as 'the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution'.

6.1. The monolith

While the tools and technologies of data engineering have reached the Cambrian explosion [49], the underlying assumptions that govern the data architectures have not been much challenged.

According to Richards and Ford there are two major categories of architectures; 1) monolithic (deployed as a single unit) and 2) distributed (the system is made up of sub-components that are deployed separately). Today, most data architectures are in the first category.

While starting as a monolith can be a simple and good approach for building a data-intensive system, it falls short as the solution scales. While this assumption is challenged in the software engineering world, data engineering seems to still be driven by monolithic designs [8]. These designs are enforced by enabler technologies such as data warehouses and data lakes. In addition, many organisations and books adopt the idea of a *single source of truth.*

6.2. The data chasm

Analytical data and operational data are two distinct types of data used in businesses. Operational data is used to manage day-to-day business operations, while analytical data is used to support strategic decision-making by identifying patterns and trends in historical data.

Many of the challenges of current BD architectures rely on its fundamental assumption of dividing operational and analytical data. While operational and analytical data have different properties and are processed differently, bringing operational data far from its original source can affect its integrity negatively, create organisational silos, and produce data quality issues.

These two different planes of data are usually operated underlying different organisational hierarchies. Data scientists, business intelligence analysts, machine learning engineers, data stewards, and data engineers are usually under the leadership of the Chief Data and Analytics Officer (CDAO) and are heavily involved in creating business value out of data. On the other hand, software engineers, product owners, and quality assurance engineers are usually working with the Chief Technology Officer (CTO).

This has resulted in two segregated technology stacks and heavy investments in bridging the two. This chasm has resulted in two different topologies and fragile integration architectures through ETLs (Figure 1). This is usually achieved by some sort of batch ETL job that aims to extract data from operational databases. These ETLs usually don't have any clearly defined contract with the operational database and are sheer consumers of its data. This highlights the fragility of this architecture, as the upstream changes from operational databases can affect downstream analytical applications. Over time, ETL job complexity increases, maintainability becomes harder, and data quality decreases.

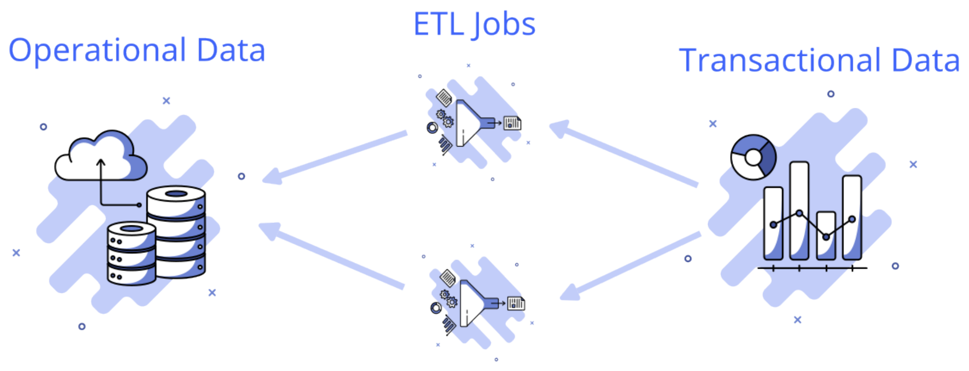

Figure 1: The great divide of data

Many of the technologies created over the years are developed underlying this very assumption. While these technologies are effective in handling the volume, velocity and variety of data, today's data challenges are about the proliferation of origins, data quality and data architecture.

Data are usually collected and consolidated through several systems. Some of these data may even go beyond the perimeters of the organisation. Herefore, based on the premises discussed hereinabove and the challenges explicated in Section **??**, it is posited that today's data architectures need a shift from the centralization of data in one big analytical database to connecting analytical data wherever it is.

Based on this, the artifact designed for this study learns from past solutions and addresses their shortcomings. This artifact aims to walk away from overly centralised and inflexible data architectures that act as a coordination bottleneck. Therefore, one of the objectives of this artifact is to bridge the gap between the point where data is generated and the point in which it is used, thus simplifying the process. This artifact aims to increase agility in the face of growth and respond effectively to organisational changes. Therefore the first architectural style of Metamycelium is as follows:

6.2.1. Localized autonomy through domain-driven decentralisation

Today's businesses are dealing with a great deal of complexity. A typical business is made up of various domains with different domains and structures. These domains change at different rates and tend to be quite isolated from each other. The overall synergy of the business is dictated by the relationship between these domains and the ways in which these domains evolve.

At the heart of these synergies sits volatility and rapid change in the market and an ever-increasing number of regulations. How do businesses today manage the impact of these changes to their data? should they constantly modify ETL jobs, create new ETL backlogs, and consolidate data into operational stores? how can businesses create quality and trustworthy data without slowing down? this boils down to embracing the change in today's data landscape.

One way to tackle this complexity is to align technology with business. Businesses break down their problem into smaller problems that are handled in each domain; technology and data can be incorporated into those domains too. This approach is well-established in microservices architectures [10].

6.2.2. Domain-driven design

Integral to Metamycelium is the distribution and decentralisation of services into domains that have a clear, bounded context. Perhaps one of the most challenging things one might face when it comes to architecting a distributed system is: based on what architectural quanta should the system be broken down? This issue has been repeatedly discussed, for example, among adopters of microservices architecture [23]. Metamycelium, inspired by the concept of domain-drive design (DDD), tends to sit data close to the product domain that relates to it. This implies that data is in the product domain and as a facet of it [37].

This is mainly driven by the fact that most organisations today are decomposed based on their products. These products are the capabilities of the business that are segregated into various domains. These domains usually define their bounded context, evolve at different rates, and are operated by cross-functional teams [54]. Incorporating data into these bounded contexts can result in a synergy that can improve the management of continuous change.

This can be micro, such as application developers communicating with data engineers about collecting user data in nested data structures, or macro, such as application developers thinking about redesigning their Graphql schema in an intermediary layer that may affect downstream analytical services. Therefore, the concept of DDD is incorporated into this study to facilitate communication and increase the adoption, rigour, and relevance of Metamycelium.

DDD is an approach to software development that focuses on understanding and modelling the problem domain of a software application. The goal of DDD is to create software that reflects the language, concepts, and behaviours of the problem domain rather than being based solely on technical considerations.

DDD can help Metamycelium by providing a systematic approach to modelling and managing data that is closely aligned with the problem domain of the application. By focusing on the language, concepts, and behaviours of the problem domain, DDD can help data architects gain a deeper understanding of the data that is needed and how it should be structured (eridaputra 2014 modelling, DataMesh Communication is a key component of any software development endeavour [60], and without it, essential knowledge sharing can be compromised. Often, data engineers and business stakeholders have no direct interaction with one another. Instead, domain knowledge is translated through intermediaries such as business analysts or project managers to a series of tasks to be done [34]. This implies at least two translations from two different ontologies.

In each translation, information is lost, which is essential domain knowledge, and this implies a risk to the overall quality of the data. In such a data engineering process, the requirement often gets distorted, and the data engineer has no awareness of the actual business domain or the problem being addressed.

Often, problems being solved through data engineering are not simple mathematical problems or riddles but rather have broader scopes. An organisation may decide to optimise workflows and processes through continuous data-driven decision-making, and a data architecture that is overly centralised and not flexible can risk project failure.

6.2.3. Complex adaptive systems

Architectures like Metamycelium share properties with complex adaptive systems [28]. The artifact designed for this thesis is especially inspired by the idea that powerful groups can emerge from an array of simple rules governing local agents. In one study, Reynolds analysed a synchronized flock of starling birds in autumn. This study presented the fact that every starling bird follows three simple rules: 1) alignment (following flockmates that are close by), 2) separation (so birds don't collide with each other), and 3) cohesion (keeping the same pace as the neighbouring flockmates). These rules can be mathematically expressed as follows:

Alignment:

$$v_i(t+1) = v_i(t) + \frac{1}{k}\sum_{j=1}^{N}(v_j(t) - v_i(t))$$

where $v_i(t)$ is the velocity vector of bird i at time t, k is a normalization factor, and N is the number of neighbouring birds.

Cohesion:

$$v_i(t+1) = v_i(t) + \frac{1}{k}(c_i(t) - p_i(t))$$

where $c_i(t)$ is the center of mass of the neighbouring birds, $p_i(t)$ is the position of bird i at time t, and k is a normalization factor.

Separation:

$$v_i(t+1) = v_i(t) + \sum_{j=1}^{N} \frac{(p_i(t) - p_j(t))}{d_{ij}^2}$$

where $p_i(t)$ is the position of bird i at time t, $p_j(t)$ is the position of bird j at time t, and d_{ij} is the distance between birds i and j.

Starling birds don't need a centralised orchestrator to create this complex adaptive system. In Metamycelium the aim is to promote a domain-driven distribution of data ownership. This architecture is modeled in a way that a domain does not provide only operational data through a standard interface, but does provide analytical data too. For instance, in practice management software for veterinaries, the animal domain provides operational APIs for updating animal attributes, but it can also provide analytical interfaces for retrieving animal data within a window of time. Every domain owns its data.

In this fashion, the domain can also choose to retrieve data from other domains with some sort of discovery mechanism, process the data and enrich its data. In some cases, there can be a creation of aggregate domains with the main concern of aggregating data from various domains and providing it for a specific use case.

This is to remove vertical dependency and allow teams to have their local autonomy while being empowered with the right level of discovery and APIs. This architecture promotes the idea of coequal nodes consolidated to achieve the overall goal of the system rather than a centralised database of all data owned by people who don't have domain knowledge. This concept is inspired by DDD in the book presented by Evans and Evans, data mesh [22], and microservices architecture [42].

6.2.4. Event driven services

Metamycelium's decentralized and distributed architecture introduces challenges in service communication as the network grows. Initially, simple point-to-point communication via REST API calls suffices, but this method proves inefficient with system expansion. Such synchronous interactions can lead to a 'distribution tax,' where one service's blocking state, often due to intensive processes, causes delays in dependent services [40].

The heavy network demands of distributed systems may further complicate matters, potentially causing *tail latency*, *context switching*, and *gridlocks* [56, 27, 32, 7]. This tight coupling is at odds with the distributed system's goals of autonomy and resilience.

To overcome these issues, Metamycelium adopts asynchronous event-driven communication. This model enables services to publish and respond to events, thus decoupling their interactions. In this *publish and forget* framework, services announce events to specific topics and move forward without awaiting direct responses. This is similar to restaurant staff responding to environmental cues instead of direct commands, promoting a smooth operational flow.

While event-driven architectures typically offer eventual consistency, which might not be suitable for certain real-time stream processing scenarios requiring immediate consistency, it is a safe assumption that the majority of data engineering workloads can efficiently operate within an event-driven paradigm.

According to Richards and Ford, event-driven architectures come in two major topologies: 1) broker topology and 2) mediator topology. Additionally, in the works of Stopford the concept of streaming platforms is elaborated. Since Metamycelium is a BD architecture that aims to process analytical data, a lot of challenges of achieving ACID transactions are eliminated. Therefore Metamycelium adopts the architectural concepts of *distributed asynchronous event-driven systems* using a hybrid topology. That is, Metamycelium is absorbing some elements of the broker topology, some elements of the mediator topology and many concepts from streaming platforms. Therefore in Metamycelium's CAP theorem [13] partition tolerance and availability are top guarantees.

There are five primary architectural components within this hybrid topology: the event, the event consumer, the event producer, the event backbone and the eventing interface. Events are initiated by the event producer and are dispatched to the topic of interest. This event then goes through the event broker and is stored there for retrieval in a queue-like indexed data structure. The event consumers that are interested in this topic will then listen to the topic of interest using the eventing interface. The event backbone itself is internally a distributed system that is made up of an arbitrary number of event brokers.

Event brokers are services that are spawned and provisioned to facilitate event communication through Metamycelia. These brokers are coordinated with a distributed service coordinator. Brokers do also allow for the replication of topics. Furthermore, to allow for fault tolerance and recoverability, the event backbone has a dedicated event archive. This archive aims to store all events that are going through the brokers so they can be restored to a correct state if a failure occurs.

These components work together to create a distributed, fault-tolerant, and scalable data system that can handle both batch and stream processing as portrayed in Figure 2.

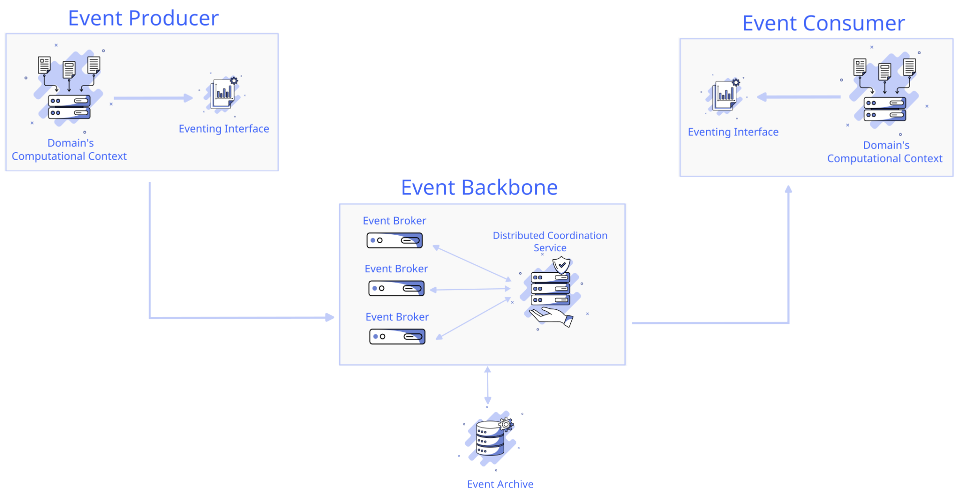

Figure 2: Metamycelium Event-driven Architecture

6.3. Architectural characteristics

The architectural characteristics of the Metamycelium architecture are notable for their focus on maintainability, scalability, fault tolerance, elasticity, and deployability. It aligns with modern engineering practices such as automated deployment and continous integration. The architecture emphasises the use of microservices, which are independently deployable and maintainable components.

Maintainability is a high-scoring characteristic of this architecture. The use of event-driven microservices architecture allows for modular development and independent scaling, making it easier to maintain and update individual components without affecting the entire system. Additionally, the architecture supports automated deployment practices, facilitating efficient updates and reducing manual intervention.

Scalability and elasticity are also prominent features. The architecture enables horizontal scalability, allowing for the addition or removal of services based on demand. This flexibility ensures that the system can handle varying workloads effectively.

Fault tolerance is another strength of the architecture. While interservice communication can impact fault tolerance, redundant and scalable design of Metamycelium, along with service discovery mechanisms, mitigate this issue. The independent, single-purpose nature of microservices generally leads to high fault tolerance.

Deployability is emphasised, thanks to the small deployment units and decoupled nature of Metamycelia. The architecture supports evolutionary change, aligning with modern business practices that require agility and adaptability. Small, independently deployable units allow for faster updates and iterations, keeping pace with the dynamic nature of business requirements.

However, the architecture may receive a lower score in terms of cost and simplicity. The distributed nature of Metamycelium and the potential for increased communication overhead can introduce complexities in managing and optimizing costs. Strategies such as intelligent data caching and replication can address performance challenges associated with network calls, but cost management remains an ongoing consideration.

Overall, the Metamycelium architecture embraces the strengths of microservices, prioritising maintainability, scalability, fault tolerance, elasticity, and deployability. It acknowledges the challenges inherent in distributed architectures and offers strategies to mitigate them. Architects must understand the rules of architecture to intelligently navigate and leverage its benefits effectively. An overview of architectural characteristics is portrayed in Table 2

7. Artifact

8. Discussion

9. Conclusion

References

[1] , 2023. Building a High-Performance Data and AI Organization. Technical Report. MIT Technology Review Insights. URL: `https://www.databricks.com/resources/whitepaper/mit-technology-review-insights`
accessed: 2023-06-03.

Table 2: Metamycelium Architecture Characteristics

Characteristic	**Score**
Maintainability	★★★
Scalability	★★★★
Fault Tolerance	★★★
Elasticity	★★★★
Deployability	★★★★
Cost	★★
Simplicity	★
Performance	★★★
Support for Modern Engineering Practices	★★★★

[2] Abran, A., Moore, J.W., Bourque, P., Dupuis, R., Tripp, L., 2004. Software engineering body of knowledge. IEEE Computer Society, Angela Burgess , 25.

[3] Al-Jaroodi, J., Mohamed, N., 2016. Characteristics and requirements of big data analytics applications, in: 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC), pp. 426–432.

[4] Amatriain, X., 2013. Beyond data: from user information to business value through personalized recommendations and consumer science, ACM. pp. 2201–2208. doi:10.1145/2505515.2514691.

[5] ANSI, A., 1975. X3/sparc study group on dbms, interim report. SIGMOD FDT Bull 7.

[6] Ataei, P., Litchfield, A., 2020. Big data reference architectures: A systematic literature review, in: 2020 31st Australasian Conference on Information Systems (ACIS), IEEE. pp. 1–11. doi:10.5130/acis2020.bf.

[7] Ataei, P., Litchfield, A., 2021. Neomycelia: A software reference architecturefor big data systems, in: 2021 28th Asia-Pacific Software Engineering Conference (APSEC), IEEE Computer Society, Los Alamitos, CA, USA. pp. 452–462. URL: `https://doi.ieeecomputersociety.org/10.1109/APSEC53868.2021.00052`, doi:10.1109/APSEC53868.2021.00052.

[8] Ataei, P., Litchfield, A., 2022. The state of big data reference architectures: a systematic literature review. IEEE Access .

[9] Ataei, P., Litchfield, A., 2023. Towards a domain-driven distributed reference architecture for big data systems, in: AMCIS 2023.

[10] Ataei, P., Staegemann, D., 2023. Application of microservices patterns to big data systems. Journal of Big Data 10, 1–49.

[11] Bahrami, M., Singhal, M., 2015. The role of cloud computing architecture in big data. Springer. pp. 275–295. doi:10.1201/9781315155678-8.

[12] Bashari Rad, B., Akbarzadeh, N., Ataei, P., Khakbiz, Y., 2016. Security and privacy challenges in big data era. International Journal of Control Theory and Applications 9, 437–448.

[13] Brewer, E.A., 2000. Towards robust distributed systems, in: International Conference on Dependable Systems and Networks, IEEE. pp. 398–405.

[14] Bucchiarone, A., Dragoni, N., Dustdar, S., Lago, P., Mazzara, M., Rivera, V., Sadovykh, A., 2020. Microservices. Science and Engineering. Springer .

[15] Chang, W.L., Boyd, D., 2018. NIST Big Data Interoperability Framework: Volume 6, Big Data Reference Architecture. Technical Report. National Institute of Standards and Technology (NIST). Gaithersburg, MD, USA.

[16] Chen, H., Chiang, R.H., Storey, V.C., 2012. Business intelligence and analytics: From big data to big impact. MIS quarterly 36, 1165. doi:10.2307/41703503.

[17] Chen, H.M., Kazman, R., Haziyev, S., 2016. Agile big data analytics development: An architecture-centric approach, in: 2016 49th Hawaii International Conference on System Sciences (HICSS), IEEE. pp. 5378–5387. doi:10.1109/hicss.2016.665.

[18] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M., 2010a. The concept of reference architectures. Systems Engineering 13, 14–27. doi:10.2514/6.2017-5118.

[19] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M., 2010b. The concept of reference architectures. Systems Engineering 13, 14–27.

[20] Coulouris, G., Dollimore, J., Kindberg, T., 2005. Distributed Systems: Concepts and Design. 4 ed., Addison-Wesley.

[21] technology review insights in partnership with Databricks, M., 2021. Building a high-performance data organization. URL: `https://databricks.com/p/whitepaper/mit-technology-review-insights-report`.

[22] Dehghani, Z., 2020. Data mesh: A new paradigm for data-driven organizations. ThoughtWorks .

[23] Dehghani, Z., 2022. Data Mesh: Delivering Data-Driven Value at Scale. O'Reilly Media.

[24] Derras, M., Deruelle, L., Douin, J.M., Levy, N., Losavio, F., Pollet, Y., Reiner, V., 2018. Reference architecture design: a practical approach, in: 13th International Conference on Software Technologies (ICSOFT), SciTePress-Science and Technology Publications. pp. 633–640.

[25] Eridaputra, H., Hendradjaya, B., Sunindyo, W.D., 2014. Modeling the requirements for big data application using goal oriented approach, in: 2014 international conference on data and software engineering (ICODSE), pp. 1–6.

[26] Evans, E., Evans, E.J., 2004. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional.

[27] Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., Bruno, A., Hu, J., Ritchken, B., Jackson, B., et al., 2019. 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, pp. 3–18.

[28] Holland, J.H., 1992. Complex adaptive systems. Daedalus 121, 17–30.

[29] ISO, I., 2016. Information technology — reference architecture for service oriented architecture (soa ra) — part 1: Terminology and concepts for soa. International Organization for Standardization , 51URL: `https://www.iso.org/standard/63104.html`.

[30] ISO/IEC, 2018. Iso/iec 29148:2018. systems and software engineering — life cycle processes — requirements engineering. URL: `https://www.iso.org/standard/72089.html`.

[31] Kaisler, S., Armour, F., Espinosa, J.A., Money, W., 2013. Big data: Issues and challenges moving forward, in: 2013 46th Hawaii International Conference on System Sciences, IEEE. pp. 995–1004. doi:10.1109/hicss.2013.645.

[32] Kakivaya, G., Xun, L., Hasha, R., Ahsan, S.B., Pfleiger, T., Sinha, R., Gupta, A., Tarta, M., Fussell, M., Modi, V., et al., 2018. Service fabric: a distributed platform for building microservices in the cloud, in: Proceedings of the thirteenth EuroSys conference, pp. 1–15.

[33] Kassab, M., Neill, C., Laplante, P., 2014. State of practice in requirements engineering: contemporary data. Innovations in Systems and Software Engineering 10, 235–241.

[34] Khrononov, S., 2021. Learning Domain-Driven Design: Aligning Your Architecture with the Business using Context Maps, Strategic Design, and Agile Techniques. Packt Publishing, Birmingham, UK. URL: `https://www.amazon.com/Learning-Domain-Driven-Design-Aligning-Architecture/dp/`

[35] Kiran, M., Murphy, P., Monga, I., Dugan, J., Baveja, S.S., 2015. Lambda architecture for cost-effective batch and speed big data processing, in: 2015 IEEE International Conference on Big Data (Big Data), IEEE. pp. 2785–2792.

[36] Kreps, J., 2014. Questioning the lambda architecture. Online article, July 205. URL: `https://www.oreilly.com/radar/questioning-the-lambda-architecture/`.

[37] Laigner, R., Zhou, Y., Salles, M.A.V., Liu, Y., Kalinowski, M., 2021. Data management in microservices: State of the practice, challenges, and research directions. arXiv preprint arXiv:2103.00170 .

[38] Laplante, P.A., 2017. Requirements engineering for software and systems. Auerbach Publications.

[39] Marr, B., 2016. Big data in practice: how 45 successful companies used big data analytics to deliver extraordinary results. John Wiley and Sons. doi:10.1109/bigdata.2018.8622333.

[40] Montesi, F., Weber, J., 2016. Circuit breakers, discovery, and api gateways in microservices. arXiv preprint arXiv:1609.05830 .

[41] Newman, S., 2015a. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Inc.

[42] Newman, S., 2015b. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Sebastopol, CA. URL: `https://www.amazon.com/Building-Microservices-Designing-Fine-Grained-Systems/d`

[43] OATH, 2007. Oath reference architecture, release 2.0 initiative for open authentication. OATH URL: `https://openauthentication.org/wp-content/uploads/2015/09/ReferenceArchitectur`

[44] Pääkkönen, P., Pakkala, D., 2015. Reference architecture and classification of technologies, products and services for big data systems. Big data research 2, 166–186.

[45] Quintero, D., Lee, F.N., et al., 2019. IBM reference architecture for high performance data and AI in healthcare and life sciences. IBM Redbooks.

[46] Rad, B.B., Ataei, P., 2017a. The big data ecosystem and its environs. International Journal of Computer Science and Network Security (IJCSNS) 17, 38.

[47] Rad, B.B., Ataei, P., 2017b. The big data ecosystem and its environs. International Journal of Computer Science and Network Security (IJCSNS) 17, 38.

[48] Rada, B.B., Ataeib, P., Khakbizc, Y., Akbarzadehd, N., 2017a. The hype of emerging technologies: Big data as a service. Int. J. Control Theory Appl 9, 1–18.

[49] Rada, B.B., Ataeib, P., Khakbizc, Y., Akbarzadehd, N., 2017b. The hype of emerging technologies: Big data as a service. Int. J. Control Theory Appl .

[50] Reynolds, C.W., 1987. Flocks, herds and schools: A distributed behavioral model, in: Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pp. 25–34.

[51] Richards, M., 2015. Microservices vs. Service-Oriented Architecture. O'Reilly Media, Inc.

[52] Richards, M., Ford, N., 2020. Fundamentals of software architecture: an engineering approach. O'Reilly Media.

[53] Sagiroglu, S., Sinanc, D., 2013. Big data: A review, in: 2013 International Conference on Collaboration Technologies and Systems (CTS), IEEE. pp. 42–47. doi:10.1109/cts.2013.6567202.

[54] Skelton, M., Pais, M., 2019. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution.

[55] Sommerville, I., 2011. Software Engineering, 9/E. Pearson Education India.

[56] Sriraman, A., Wenisch, T.F., 2018. μ suite: a benchmark suite for microservices, in: 2018 IEEE International Symposium on Workload Characterization (IISWC), IEEE. pp. 1–12.

[57] Srivastava, R., 2018. Big data: Issues and challenges. International Journal Of Scientific And Innovative Research .

[58] International Organization for Standardization (ISO/IEC), I., 2017. Iso/iec/ieee 42010:2011. URL: `https://www.iso.org/standard/50508.html`.

[59] Stopford, B., 2018. Designing Event-Driven Systems. O'Reilly Media, Inc.

[60] Sudhakar, G.P., 2012. A model of critical success factors for software projects. Journal of Enterprise Information Management .

[61] Viana, P., Sato, L., 2014. 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. pp. 622–629.

[62] Volk, M., Staegemann, D., Trifonova, I., Bosse, S., Turowski, K., 2020. Identifying similarities of big data projects–a use case driven approach. IEEE Access 8, 186599–186619.

[63] Wang, C.J., Ng, C.Y., Brook, R.H., 2020. Response to covid-19 in taiwan: big data analytics, new technology, and proactive testing. Jama 323, 1341–1342.

[64] Wieringa, R.J., 2014. Design science methodology for information systems and software engineering. Springer.

[65] Yu, J.H., Zhou, Z.M., 2019. Components and development in big data system: A survey. Journal of Electronic Science and Technology 17, 51–72.