Author(s): Loukiala, Antti; Joutsenlahti, Juha-Pekka; Raatikainen, Mikko; Mikkonen, Tommi; Lehtonen, Timo

Title: Migrating from a Centralized Data Warehouse to a Decentralized Data Platform Architecture

Please cite the original version:

Loukiala, A., Joutsenlahti, J.-P., Raatikainen, M., Mikkonen, T., & Lehtonen, T. (2021). Migrating from a Centralized Data Warehouse to a Decentralized Data Platform Architecture. In L. Ardito, A. Jedlitschka, M. Morisio, & M. Torchiano (Eds.), PROFES 2021 : 22nd International Conference on Product-Focused Software Process Improvement, Proceedings. Product-Focused Software Process Improvement (pp. 36-48). Springer International Publishing. Lecture Notes in Computer Science, 13126. https://doi.org/10.1007/978-3-030-91452-3_3

Migrating from a Centralized Data Warehouse to a Decentralized Data Platform Architecture

Antti Loukiala[1], Juha-Pekka Joutsenlahti[2], Mikko Raatikainen[3][0000−0002−2410−0722], Tommi Mikkonen[3,4][0000−0002−8540−9918], and Timo Lehtonen[1][0000−0001−8833−1725]

[1] Solita Ltd, Finland
`first.last@solita.com`
[2] TietoEVRY, Tampere, Finland
`first.last@tietoevry.com`
[3] University of Helsinki, Finland
`first.last@helsinki.fi`
[4] University of Jyväskylä, Finland
`tommi.j.mikkonen@jyu.fi`

Abstract. To an increasing degree, data is a driving force for digitization, and hence also a key asset for numerous companies. In many businesses, various sources of data exist, which are isolated from one another in different domains, across a heterogeneous application landscape. Well-known centralized solution technologies, such as data warehouses and data lakes, exist to integrate data into one system, but they do not always scale well. Therefore, robust and decentralized ways to manage data can provide the companies with better value give companies a competitive edge over a single central repository. In this paper, we address why and when a monolithic data storage should be decentralized for improved scalability, and how to perform the decentralization. The paper is based on industrial experiences and the findings show empirically the potential of a distributed system as well as pinpoint the core pieces that are needed for its central management.

Keywords: Data warehousing · data platform architecture · distributed data management · data decentralization

1 Introduction

Upon becoming the driving force for digitization, data is regarded as a central key asset by many companies. Used at an ever-increasing scale in decision making and applications, there is a constant demand for more data, at better quality and availability, and from a wider time horizon.

State of the practice technological solutions, such as data warehouse and data lake, have been introduced as a solution that integrates data from disparate sources into a central repository for analytic and reporting purposes. Thus, the solution can be considered as a central, monolithic solution that they are typically hard to scale [5]. Monolithic solutions tend to become hard to scale from

the development point of view as multiple developers are working on the same code base making it hard to parallelize the work and in the larger systems code base can become very complex. Monolithic solutions are usually built around single technology that might not support all the use cases in an optimal way. There are also other complicating factors. In particular, central data warehouse teams require several special data management skills [1]. Furthermore, as the data warehouse is working with business domain-specific data, the team also requires deep domain knowledge.

The situation can be simplified with modern data engineering practices, including the use of version control, continuous integration and deployment, and metadata-driven data warehouse development, which automate many steps of data modelling, making the data available for consumption faster. Similarly, today's technologies typically do not create bottlenecks for data management – they usually scale well both vertically and horizontally. However, even with these improvements, integrating enterprise data, which may originate from numerous source systems, into a single model can lead to complex models that are hard to build, understand, use, and maintain. Moreover, the process of distilling data from the source to the final data warehouse model requires many, often time-consuming steps and wide technical expertise as well as deep domain knowledge. With this premise, centrally run data solutions even with the latest technology become easily a bottleneck for data management. A common way to scale is to bring in more expertise to the central team that is accountable for the solutions. This is not always feasible because the expertise required to build, improve and maintain these kinds of systems is very broad and hard to find in the labour market.

In contrast to data management, scaling centralised, monolithic applications have been addressed in software development, where sophisticated solutions have been introduced. For instance, large systems use distributed, service-oriented architectures, built using technologies, such as microservices, that also introduce other benefits as a side-effect. In addition, new innovations, such as the cloud, helps in scaling. However, while these technologies have been around for a while, data management related activities and operations have not seen a similar paradigm change.

This paper proposes decentralizing data management architecture, where different business functions take larger responsibility of exposing their data for general analytical use. The solution is based on experiences from a centralized data management architecture in a large Nordic manufacturing company where Solita[5], a Nordic midcap consultancy company, has been consulting on data related development. The Nordic manufacturing company has recently undergone a data decentralizing process, resulting in a decentralized data platform architecture. Our findings empirically show the potential of such architecture as well as indicate the core pieces that need central management.

While we have developed the approach and the data platform architecture implementation independently, the recently introduced concept called *data mesh*

---

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

is also used to refer to a similar decentralised approach to data management. The closest concrete data mesh-based approaches to ours, also building on designing a scalable data architecture, have been suggested by Zhamak Deghani [2] and ThoughtWorks [13]. However, to the best of our knowledge, neither one has reported uses in scientific literature. For the sake of simplicity, in this paper, we have retained our original terminology, as data mesh is such a new concept that several variations exist in its terminology.

The rest of the paper is structured as follows. In Section 2, we present the necessary background for the paper. In Section 3, we introduce the case context and its setup. In Section 4, we provide an insight into the drivers of the modernization process for the data architecture. In Section 5, which forms the core of the paper, we introduce the new data architecture in the case system. In Section 6, we discuss experiences gained in the process. In Section 7, we draw some final conclusions.

2   Background

Next, we introduce the key concepts of the paper. These include data warehousing, data lakes, data platform architecture, service-oriented architecture, and microservices. Each of these concepts is discussed in a dedicated subsection below.

2.1   Data warehousing

Data warehousing [7] refers to building data storage used for reporting and analytics. A resulting data warehouse is a central repository of data extracted from different source systems. The data is integrated and remodelled to allows easy usage of data. Separating the analytical database in a data warehouse from the transactional databases used in daily operations allows one to apply heavy aggregations without compromising the operational work. Furthermore, a data warehouse typically stores the history of the operational data. In general, successful data warehousing requires an ecosystem of tools and capabilities to make this possible.

Data is typically published from the data warehouse in a form of a dimensional data model, which is a widely accepted way of modelling data for reporting and analytical usage. A dimensional data model creates a data presentation layer that simplifies the data access patterns.

To represent data from multiple sources in a common enterprise-wide format in a data warehouse, data is remodelled into a *canonical data model* for better accommodating the analytical needs. Forming a canonical model in a complex environment consisting of numerous sources is a complex task, because of contextual dependencies between domains. Therefore, in order to integrate data into a canonical data model requires heavy upfront data modelling, contextual mapping, and deep domain knowledge of the different data sources.

In order to make data warehouse development more efficient, highly specialized teams are formed that can deliver data for analytical applications. This often creates bottlenecks, as adding new data into the data warehouse as well as making new data assets available a huge amount of work.

2.2   Data lakes

A *data lake* is a central repository of structured and unstructured data that can be of any format [12] unlike in a data warehouse, where a uniform data model is used. A data lake can speed up data availability because data is often stored in a raw format and the schema of data is defined at the read time allowing more flexible usage of data.

This form of storage is very suitable for data analytics and data science, as discoveries on the data assets can be made without the requirement of heavy data modelling upfront. A data lake also allows raw historical data storage at lower costs, taking off some of the burdens from a data warehouse based approach.

Modern big data tools allow easy usage of the data stored in a data lake, allowing even running SQL queries on the stored data, making it viable to store more curated and transformed data sets within it [8]. A data lake is separated from operational systems and therefore has little or no effect on source systems.

On the downside, even though a data lake allows more flexible access to data, a data lake is often built in a central manner creating similar bottlenecks as a data warehouse.

2.3   Data Platform Architecture

Modern usage of data in an organisation usually requires multiple different components to match all the needs. We use the term *data platform architecture* to refer to the architecture that defines how a data platform connects the different data sources to a bigger whole, enabling the use of data in a similar fashion one expects in a centralized approach. This includes fundamental structures, elements, and their relations to each other that are designed to match the strategic goals set by the company on data management.

Data platform architecture needs to fit the organisation and its strategic goals. Even though there is no single way of realizing a data platform architecture, a data platform is typically seen as a central service, maintained by a single team. Even in modern data platforms that are composed out of multiple functional elements, the platforms are very centrally built. Even though the platform approach can tackle demanding needs for diverse data analysis needs, a complex and large realization of centrally managed architecture suffers from an ability to scale.

The data platform architecture implementation we have composed is based on design principles from software design. The overall framework follows principles of service-oriented architecture, and microservices are used as the underlying implementation technique.

2.4   Service oriented architecture

Scaling large software development is not a new problem. In a *service-oriented architecture* (SOA), the software is separated into functions that provide services to other applications [10]. Every service in SOA is an independent, reusable, and discrete service that is decoupled from the other services and can be therefore developed and scaled independently [11]. A service is not tied to any technology but is a more abstract concept allowing each service to be built with the best technology available by a dedicated team. Even though the SOA has evolved and new even more distributed architectures have been introduced, the key to scaling remains in the distribution.

SOA integrates modular, distributed, and independent services [11]. The emphasis is on the communication and cooperation taking place usually over the network. An application programmable interface (API) is a central concept in SOA, as an API defines the communication interfaces. As with any distributed system, the definitions of used standards and technologies enable this communication to take place in the most convenient manner.

2.5   Microservice architecture

Micro service architecture (MSA) [9] is an implementation technique that follows the ideals of SOA, but further emphasises the decentralized, distributed, and independent services. Much like the principles in SOA, modern MSAs are built using RESTful APIs that formalise the interfaces for communication between independent services. Such independent services provide scalability, flexibility, and reliability for agile development of business-centric systems [5]. Furthermore, self-contained systems with loose coupling enable continuous deployment according to customer's agile needs.

In MSA, each service is working on its dedicated business function and therefore should focus on that function solely [9]. This is somewhat opposite to forming a company-wide canonical data model, requiring heavy upfront investment and complex integration patterns. Instead, the focus in MSA is on the data and ontology of one business function alone. This approach is based on one of the core patterns of domain-driven design, called bounded context[3]: Large models are separated into explicit, internally consistent bounded contexts usually around a specific business function. This bounded context creates a place for ubiquitous language within its boundaries simplifying the data models.

The modularity of services adhering to MSA enables developing services independently of each other, thus supporting continuous software engineering [4] to take place. Modularity also allows each service to be technologically independent of each other, and hence the teams building the services can choose the best tools for the job they are performing. Each service can easily adapt to changes in demand by scaling human resources and technical resources, decoupled from other services. Legacy systems can be re-engineered incrementally by breaking parts of the legacy system functionality into individual services [6]. Finally, services can be monitored easily, in varying detail, depending on the needs and critically of the service in question.

Even though MSA has been around for a while in software development and used successfully as a way to scale applications, the architectural model of MSA has not been implemented in the field of data management. The data platform architecture is often designed to be a centralised system owned by a single team from IT.

## 3   Case context and challenges

We use a large Nordic manufacturing company as the case context in this study. The company operates in global markets. The company has several organizational units in different business domains and numerous partner companies in its business ecosystem. The operations are not limited to manufacturing but different post-sales services are also important.

Originally, the data management strategy of the company consisted of a central data warehouse as presented at top of Figure 1. The data warehouse had been used for years and was running in a local data centre. An ETL (extract, transform, and load) tool was used to integrate data from multiple source systems into the data warehouse as well as re-model the data into dimensional data marts for reporting purposes. The central integration team provided a data bus, which was used wherever possible to support defining integration patterns.

The applications landscape based on data in the data warehouse was vast, including several hundreds of operational systems. The application landscape was also very heterogeneous in terms of age and technology. The development and operation models for the applications differed including in-house build applications, licensed off-the-self applications, and software-as-service applications from external vendors. Some of the applications were owned and operated by the company's partners. Applications were used in different business domains, linked to different business processes, and some used along with partners.

This legacy architecture had started to introduce several challenges for the business. With an old and heterogeneous application landscape, data from many of the operational source systems were not accessible for analytical or reporting use nor to the data warehouse. This was due to security reasons, performance issues, or networking limitations. As the data warehouse was in many cases the only way to access the data stored in the operational systems, the data warehouse was used for operational purposes as well analytical ones.

This central- and multi-role of the data warehouse kept increasing the number and complexity of requirements for the data warehouse as new data sets were requested constantly by different business domains for different purposes. Years of data warehouse development, when multiple operational systems were integrated into the central system, made the data model of the data warehouse very complex. Also the loading logic, as well as business logic, were hard to maintain. Bringing new data into the data warehouse had become cumbersome and error-prone delaying or even blocking the development of applications based on the data warehouse as well as breaking the existing reporting based on it.
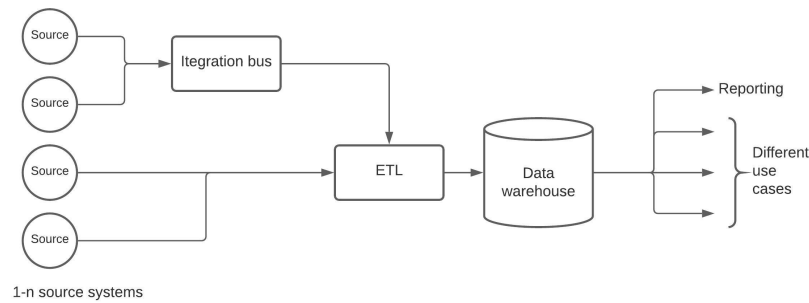
Above: A high-level architecture of the original centralized data management based on data warehouse.

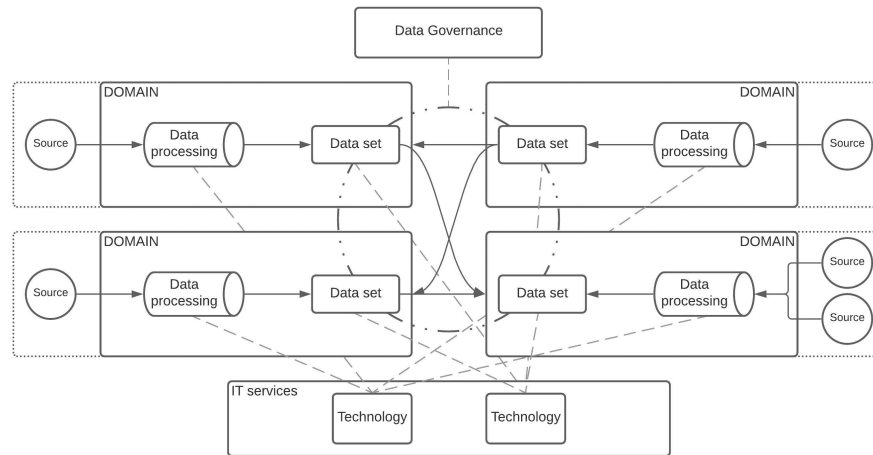Below: High-level architecture of our decentralized data platform.

Fig. 1. Top: A high-level architecture of the existing centralized data management based on the data warehouse. Bottom: High-level architecture of our decentralized data platform.

Data management in charge of the data warehouse was limited in resources and had a hard time finding competent people to work with the ever-increasing complex context of the data warehouse. With the increasing amount of data loaded and modelled in the data warehouse, from multiple different business domains, caused a centralized team to become a bottleneck. Simply scaling around these central services and the central team of the data warehouse was not truly an option anymore.

4   Drivers of modernization

The company had a high-level, strategic objective: take better advantage of data in its business operations that placed extra pressure and demand for the central data warehouse. The company was constantly looking into digitization and better usage of data. To guide this development, among the several principles were to be API and data-driven.

To make the data trapped in legacy applications available for wider analysis, the company started to modernize the legacy applications by building modern API facades on top of them about four years ago. The APIs simplified and even made possible the usage of the data from the legacy applications. In many cases, business domains, however, needed more data than what APIs provided. In particular, data from a longer time period was needed for analytics purposes in order to make better decisions based on the data. The data was used in many ways, often by data scientists who were able to create machine learning models, train artificial intelligence features in applications, and solve business-related analytical questions with data.

The departments of the company in different business domains were independent, empowered, and technologically savvy and the departments had accountability for their work. The departments were large enough to have their own developers, business process owners, solution owners, and other competencies related to the business domain they were working on.

The company had decided to make a technology transition from the in-house maintained infrastructure to a cloud-based infrastructure. The main cloud provider was chosen already two years ago, which created a very good foundation for a new data platform. To fully leverage the cloud, the data management team needed to reconsider what core functionalities it would provide to the company.

The cloud-transition decision was followed by a project to modernise the data warehouse infrastructure by moving the data into cloud-based solutions to better match the demand for scaling with increasing volume, variety, and velocity of data. To further support more flexible data usage, the design of a cloud-native data lake was introduced to create more direct access to the data for analytical purposes, as well as to support the data warehouse.

Since the organization had already elements of distributed application architecture by the means of MSA as a result of cloud transformation, it was proposed that the data management would be fitted into this same architectural model as well. The principle adopted from MSA was that by distributing the data management to the different departments much alike micro services distribute functionality, the central data management could focus more on core entities, connecting different departments, data governance, and providing required core services. In this data distribution, the departments would be in charge of the data assets they consume and produce as well as applications they built. This way the domain expertise would be placed naturally closer to the data assets they were dealing with. Departments would communicate with each other through curated data assets that were requested by the other departments.

A careful analysis of the technology offering being used at the time and adaptation of cloud services made it clear that also distributing the data platform to domains was technologically possible. Cloud offering supported the distribution and it was according to architectural principles.

5   New data architecture

A domain-oriented distributed data platform architecture (Figure 1, bottom) was formed to overcome the scaling problems that the centralized data management was facing earlier with the data warehouse and lake. Using the same patterns that are fundamental to MSA allowed the data platform architecture to scale better. The domain oriented distributed data platform was implemented by distributing the data platform to different business domains and defining standardised interfaces for each domain to allow interaction between them. Each business domain was made responsible for the data assets they were in a relationship with as well as generating curated historised data assets for analytic usage. The business domains also had the freedom to develop data assets in a fashion most suitable for them as long as they adhered to defined core principles defined by the central data management team.

The decentralised approach allowed IT to move from building applications and data analytics into providing infrastructure, guidelines, standards, and services for the domain teams. The domain teams were provided with a cloud environment where they were able to readily build their applications and analytics, as well as create standardised data assets for general usage by leveraging services provided by IT. Distributing the data development was in line with the company's core principles. In many cases, the business domains had independent tech-savvy developers that made the transition of development to domains smooth.

Some of the central services were developed in parallel to the business domain data-intensive projects. A data lake was implemented in a distributed manner, where each domain team hosted their own analytical data in their own data store and, thus, formed distributed data storage. The team responsible for the data lake focused only on providing central data cataloguing capabilities, metadata models, data standards as well as "getting started"-guides. A data warehouse renewal project was initiated to clean up the complexity of the existing data warehouse. The new data warehouse was to source the data from the new distributed data lake in order to form integrated data models for reports that were used company-wide. Consequently, the data warehouse was transformed from the central data repository to a single consumer of distributed data assets. Master data management was also used to form some data sets that were seen as key assets for the entire company. These data assets were shared through the data catalogue similarly to domain-oriented data assets. These traditional data management tools, data warehouse, data lake, and master data were distributed and seen as components of the distributed system.

The computing infrastructure was managed by the cloud providers, as they offered an excellent technological baseline for the distributed data platform architecture. Guidelines and standards were defined to make the data sets in the business domains discoverable as well as interoperable forming a base of the data governance practices. Central services included business domain agnostic services out of which among the most important was data catalogue. All the different data sets from business domains were registered to the central data catalogue with a very detailed data on-boarding process.

Business domains that had very little software development skills, struggled to get on board with the distributed data platform architecture. To help such teams getting started, the central data management team focused on bringing the basic setup for the teams easily available as well as generic infrastructure and code templates. The central data management team was able to guide different teams to use specific technologies by offering a set of development tools fast and easily but not limiting teams' liberties with the offerings. Templates were built so that they encouraged security and good development practices, giving a baseline on top of which the teams were able to build own extensions. Templates were generalized from different teams building their solutions. Later, a specialised team was dedicated to helping kick-start different business domains analytical data development to match the defined interfaces.

6   Experiences

The distributed data platform architecture described above removed many of the bottlenecks that traditional data management projects had faced from the central data warehouse and data lake. There were clear indications that having a cross-functional development team working in the business domain along with business stakeholders, allowed the teams to focus more clearly on business needs being able to respond more effectively and agile fashion. The business domains were no longer coupled with the central data warehouse or data lake and, therefore, the business domains were not heavily dependent on these centralized services and their schedules. With the introduction of the distributed data platform architecture idea, business domains used their own development teams to work on the business cases most relevant to them in the prioritised order.

The autonomous domain-oriented teams were able to work parallel to one another as the data platform architecture distributed the data. The domain teams were able to ingest raw data easily as they already had the source system and data knowledge in their control as well as business-subject matter experts in the team. Many of the domain teams worked in simple data assets composed out of source systems most relevant to them. With the data usually combined within the business domain, data models were kept simple and easy to manage.

Business domain data sets along with associated metadata were published in the central data catalogue service and in this way made known to the company internally. As the metadata contained key information about the data and some contact information, this sparked new interactions between the domains. The

role of the data catalogues was key to making the distributed data platform work and to govern the publishing process. Data catalogue with well-defined metadata created the baseline for trust in the system and simplified the usage of the data. Data catalogue allowed different teams to connect to different data assets in an agile manner.

In many cases, the data was mainly needed by the business domain owning the data and there were only few teams that requested data from other business domains. In such cases, only the metadata was placed into the central data catalogue. The actual data was only released when requested. This just-in-time data releasing led to a more efficient and better-prioritised way of working.

Distributing the realization of data platform parts within the common data platform architecture to the business domains allowed the IT department to focus on enabling the business domain teams to build their data capabilities instead of building for them in IT. IT and data management were able to focus more on defining coherent, understandable, and usable unified data interfaces. Having the opportunity to review data assets that were to be published, allowed the data management team to maintain governance and quality of the system. As many of the teams were deep in digitalization, having their own development teams, the transition towards the distributed data platform was natural.

The distributed system required cross-functional teams to work on the issues. It should be noted that the company had adopted an API-driven mindset, which in turn had pushed the organisational structure towards the autonomous business domain-oriented cross-functional teams. This can be seen to have a large effect on the success of distributing the data management since data management knowledge was simply yet another factor that the teams needed to master. There were few domains that did not have development teams and had to set a development team up in order to work in a distributed manner. In many of these cases, the central IT was easily able to provide them with the infrastructure, templates, training, and even pioneering developers to get started.

7   Conclusions

Many larger companies are struggling to find a way to scale for the increasing demand for data in a centralized manner. Even though scaling by distribution has been seen as an effective way of working in many fields, there seems to be no dominant design at the moment. Rather, various proposals have been made, but from the industrial rather than the academic perspective.

In this paper, we have presented an industrial case on distributed data platform architecture. The principal contribution of the study is that centralized data platforms stop scaling at a certain point when organization domains are large, heterogeneous, and non-flexible by nature.

In the study, it was found out that distributed approach to data management provides features that enable complex enterprises' data platforms to scale (Table 1). This approach is based on two widely known and acknowledges principles – MSA and Domain-driven design. These principles adopted to data management

Table 1. Comparison between centralized and distributed data platform architecture.

Dimension	Centralized	Distributed
Competency location	Centralized single team	Cross functional domain teams
Technological choices	Single stack	Freedom of choice
Data Modeling	Top-to-Bottom	Bottom-up
Data governance	Centered around the team working on the model	Definition and monitoring of interface
Business & IT Alignment	IT focused	Business focused
Data ownership	IT owns the data and data lineage	Business owns the data and data lineage
Data application development	IT owns applications and the development	Business owns the applications and development

and combined with agile autonomous cross-functional teams make rapid concurrent data development possible. On one hand, MSA enables data management to move from centralized solutions towards distributed information ecosystems. On the other hand, domain-driven design specifies distribution to domains and bounded contexts. Together, these two methodologies enable data platform development in clusters within each business domain without the limitations and restrictions of centralized data platform teams. Such a distributed approach to data management facilitates data quality and interoperability by orchestrating data and information management over domain-oriented teams. This enables data modelling without a canonical model and enables business domains to work independently.

Acknowledgement

References

1. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: new analysis practices for big data. Proceedings of the VLDB Endowment **2**(2), 1481–1492 (2009)
2. Dehghani, Z.: How to move beyond a monolithic data lake to a distributed data mesh (2019), Martin Fowler's blog. https://martinfowler.com/articles/data-monolith-to-mesh.html
3. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional (2003)
4. Fitzgerald, B., Stol, K.J.: Continuous software engineering: A roadmap and agenda. Journal of Systems and Software **123**, 176–189 (2017)
5. Hasselbring, W., Steinacker, G.: Microservice architectures for scalability, agility and reliability in e-commerce. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). pp. 243–246. IEEE (2017)
6. Kalske, M., Mäkitalo, N., Mikkonen, T.: Challenges when moving from monolith to microservice architecture. In: International Conference on Web Engineering. pp. 32–47. Springer (2017)

7. Kimball, R., Ross, M.: The Data Warehouse Toolkit. Wiley Computer Publishing (2002)
8. Miloslavskaya, N., Tolstoy, A.: Big data, fast data and data lake concepts. Procedia Computer Science **88**, 300–305 (2016)
9. Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M.: Microservice architecture: aligning principles, practices, and culture. " O'Reilly Media, Inc." (2016)
10. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer **40**(11), 38–45 (2007)
11. Perrey, R., Lycett, M.: Service-oriented architecture. In: 2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings. pp. 116–119. IEEE (2003)
12. Stein, B., Morrison, A.: The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast: Rethinking integration **1**(1-9), 18 (2014)
13. ThoughtWorks: Data mesh (2020), https://www.thoughtworks.com/radar/techniques/data-mesh