Working Papers

ERCIS — European Research Center for Information Systems
Editors: J. Becker, K. Backhaus, M. Dugas, B. Hellingrath,
T. Hoeren, S. Klein, H. Kuchen, U. Müller-Funk, H. Trautmann, G. Vossen

Working Paper No. 27

Technology Selection for Big Data and Analytical Applications

Denis Lehmann, David Fekete, Gottfried Vossen

ISSN 1614-7448

cite as: Denis Lehmann, David Fekete, Gottfried Vossen: Technology Selection for Big Data and Analytical Applications. In: Working Papers, European Research Center for Information Systems No. 27. Eds.: Becker, J. et al. Münster 2016.

Contents

List of Figures

List of Tables

Working Paper Sketch

Type

Research Report

Title

Technology Selection for Big Data and Analytical Applications.

Authors

Denis Lehmann, David Fekete, Gottfried Vossen

contact via denis.lehmann@gmx.net, {david.fekete, gottfried.vossen}@ercis.de

Abstract

The term Big Data has become pervasive in recent years as smart phones, televisions, washing machines, refrigerators, smart meters, diverse sensors, eyeglasses and even clothes connect to the Internet. However, their generated data is worthless without information retrieval through data analytics. As Big Data is too big for a single person to investigate, appropriate technologies are being used. Unfortunately, there is not one solution but a large variety of different tools, each of them with other functionalities, properties and characteristics. Especially small and mid-sized companies have a hard time to keep track as this requires time, skills, money, and specific knowledge which result in high entrance barriers for Big Data utilization. This papers aims to reduce these barriers by explaining and structuring different classes of technologies and basic criteria for proper technology selection. It proposes a framework that guides especially small and mid-sized companies through a suitable selection process that can serve as a basis for further advances.

Keywords

Big Data, Analytics, Technology Selection, Architecture, Reference Architecture, Selection Framework

1 Introduction

The Big Data Era which started a couple of years ago has meanwhile seen an abundance of tools for processing and managing data and its applications, be it for searching, stream processing, or sentiment and text analysis, to mention just a few. Most of these software tools are open-source and can hence be employed by anybody who feels capable of arranging them into an appropriate solution architecture for the problem at hand. However, the sheer mass of tools often makes it difficult to come up with a reasonable selection, and beyond that with an organization or arrangement of the tools that can serve the given application well. This paper presents an approach to technology selection for big data and analytical applications that can considerably ease the task of navigating the "'jungle"' of tools that are available.

Data has become the most important asset for companies [56]. It is the new oil [73] that lubricates business processes and helps companies evolve towards data-driven decision making [30]. Being in line with labor, natural resources and capital, Big Data has become the next important production factor [30] [91]. At its essence, it is all about predictions and simulations [65]. Facebook predicts friends, Amazon predicts purchases, government agencies predict crimes as well as terrorist attacks, and Netflix predicts movies. Big Data analytics even enables to forecast people's behavior and emotional moods [30], as some predictions aim at customer personalization, satisfaction [62], and even online dating [14].

This vast amount of data requires new technologies and mechanisms for storage, processing, management, and analysis. It is commonly accepted that Big Data is too large, fast, and diverse for traditional Relational Database Management Systems (RDBMSs) [39]. Hence, new technologies are required that include a wide range of novel database systems, file systems, programming paradigms and languages, and machine learning tools, among other components [77]. According to DEMCHENKO, DE LAAT, and MEMBREY [35], there is no comprehensive analysis of such emerging Big Data technologies in the literature yet [43]. Instead, most discussions are happening in blogs between contributors and early adopters of open source solutions.

As a consequence, Big Data concepts and tools and their implications for technology selection or system architectures are still poorly understood [54], and traditional Business Intelligence (BI) tools for Online Analytical Processing (OLAP), such as RDBMS, are still being used for structured data and have gained capabilities to deal with larger volumes of data. [41] has identified the need for a structured technology selection approach in the context of the complexity of this tool landscape. The proposed *Goal-oriented Business Intelligence Architecture (GOBIA)* method emphasizes the selection of technologies a key to transform business needs into customized analytics architectures. However, a specific process has not been proposed yet [41]. MARR proposes a framework for organizational change towards Big Data, driven by strategy, but does not focus on specific technologies [63]. On the other hand, companies are increasingly confused with hundreds of different available tools and unsure about how to build an analytics architecture for their needs. In fact, building a suitable infrastructure comes with significant integration challenges, as each technology has its own functionality, performance, and scalability strengths and weaknesses [56].

This paper aims to develop artifacts that can aid in a structured technology selection process for customized analytics architectures in the Big Data era and is based on [61]. Specifically, it develops a guideline for technology selection and a regulatory framework that structures current technologies into distinct classes for a better overview. Overall, it explains essential selection criteria and technology differentiating dimensions. The resulting framework can also be used to complement existing approaches such as the aforementioned GOBIA method.

The remainder of this paper is structured as follows. First, the layered reference framework as a means to structure technology is outlined in Section 2. Section 3 introduces the technology selection framework and describes its process-based approach. Section 4 illustrates technology

selection using an application scenario. Finally, the paper concludes with Section 5.

2 Layered Reference Framework

This section introduces a layered reference framework that can be used to ease the classification and assessment of new technology. It maps technologies to different service layers and serves as a guide for selecting suitable technology mixes for given use cases.[1] It is the foundation of the technology selection framework to be presented in Section 3. As such, it inherits Big Data technologies at different service layers for data generation, acquisition, storage, processing, and analytics [35].

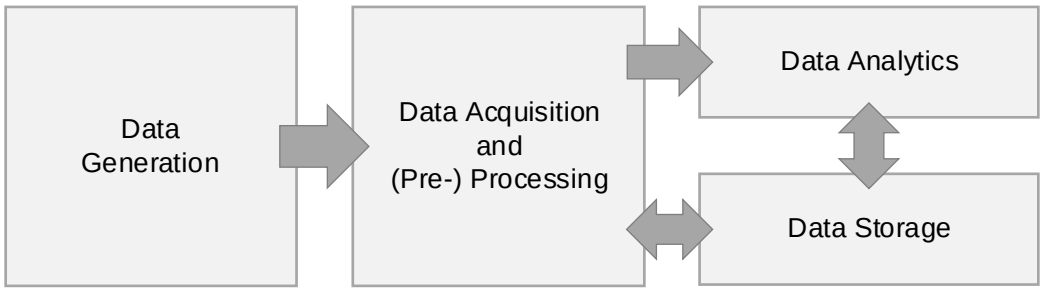

Figure 1: Adaptive Big Data Value Chain (based on [30], [49], and [23]).

A common way to visualize the process of value generation is known as the Big Data value chain (see Figure 1). It consists of four sequential phases [30] [49]: data generation, data acquisition, data storage, and data analytics. The first four layers of the reference framework correspond to the process steps of this Big Data value chain, while the top-level one accounts for its primary purpose to deliver valuable results.

The resulting layered reference framework is illustrated in Figure 2. Layer elements are ordered with increasing volume, variety, and velocity from right to left. While traditional BI technologies are indicated in blue, components associated with advanced analytics are colored red. However, the transition between BI and advanced analytics is smooth, as components sometimes belong to both groups, depending on the use case.

While advanced analytics requires input of data scientists [8], traditional BI technologies are usually set up by data analysts without profound mathematical knowledge [88]. Thus, the former usually requires good programming skills and knowledge on analytical tools using API, REPL, and CLI while the latter can often be employed using GUI or GWFU. This corresponds to the easy of use structuring from left to right.

The layered reference framework does not visualize single technologies, but classifies them by their type into different structural elements such as *Distributed File Systems* and *OLAP tools*. As there are lots of tools and projects arranged in each of these elements, there is not a single solution for a given use case [56, p. 41].

[1] The usage of a layered architecture with a service hierarchy is suggested by FEKETE and VOSSEN [41] in their research on the GOBIA method.

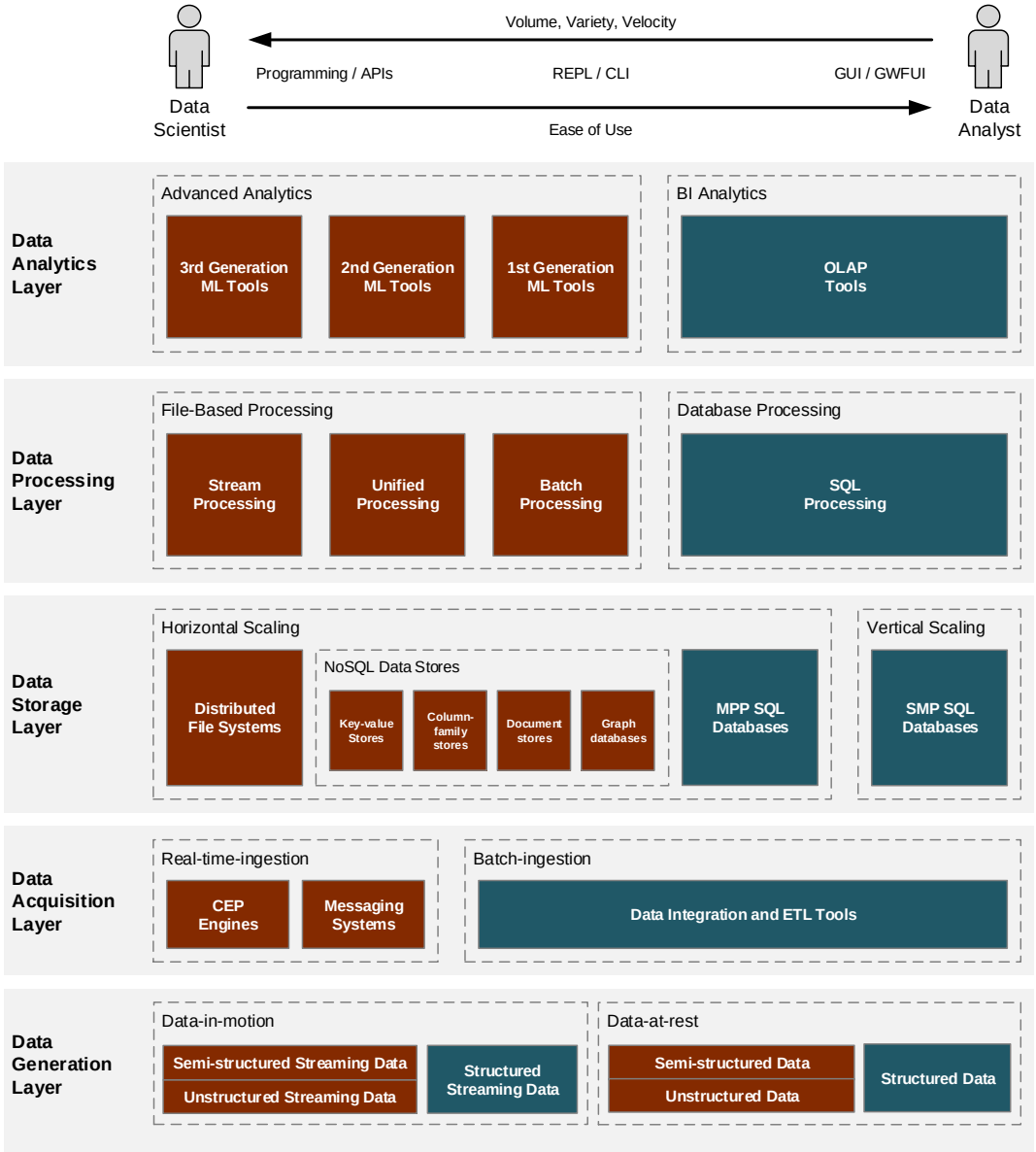

Figure 2: The Layered Reference Framework.

2.1 Data Generation Layer

The data generation layer deals with different types of data sources. The main differentiating dimensions are variety and velocity. While velocity differentiates between data-in-motion and data-at-rest [42], variety determines between structured, semi-structured, and unstructured data.

Table 1: Layered Reference Framework – Data Generation Layer.

Layer Element	**Examples**
Structured Data	Tabular, transactional, inventory, and financial data
Semi-structured Data	XML files, JSON documents, e-mails
Unstructured Data	Text, images, videos, and log files
Structured Streaming Data	High-frequency transactional and financial data
Semi-structured Streaming Data	Sensor and event data, Twitter streams
Unstructured Streaming Data	Log files for security, audio, video, and live surveillance

Data-in-motion summarizes all data that is constantly generated at low and high velocities, also known under the umbrella term streaming data. It describes events that need to be analyzed as they happen. Examples include social media streams (e.g., Twitter APIs such as Firehose[9], Facebook[2] or Xing[3]), sensor data, and log files for security access, as well as multimedia streams from music and video platforms and surveillance cameras. Other examples include high-frequency financial or transactional structured data streams. The counterpart of data-in-motion is data-at-rest [42]. This term summarizes historically generated data at fixed locations with no velocity. It includes all data that needs to be stored prior to analysis.

The distinction between data-in-motion and data-at-rest influences technology selection. Business use cases usually put requirements on response times and latency of analysis results. For instance, an earthquake or tsunami warning system is required to provide warnings in real-time, not on the next business day [27]. Consequently, the velocity of data generation and its required analysis latency have a reasonable impact on the selection of suitable technology.

Notably, more than 95% of all data is unstructured or semi-structured and thus requires additional preprocessing [44]. This work also uses the term *multi-structured data* as a generalization of semi-structured and unstructured data. All of these data can be data-in-motion (streaming data) or data-at-rest, depending on the use case at hand. The share of multi-structured data is constantly growing as everyday contents such as video, images, documents, log files, and e-mails contribute to these groups [17]. The resulting data is diverse as it includes unstructured text, logs, scientific data, pictures, voice and video records as well as sometimes metadata [56]. However, currently, structured input data has still a major role in analytical tasks, even with Big Data (e.g., cf. [43]).

2.2 Data Acquisition Layer

The data acquisition layer deals with technologies for an ingestion of data into Big Data infrastructures [56]. The main differentiating dimension is velocity. It distinguishes between batch and real-time ingestion. Real-time ingestion is sub-divided into messaging systems and Complex Event Processing (CEP) engines, while batch ingestion includes traditional Extract-Transform-Load (ETL) data integration tools. Sample technologies for the different layer elements are given

[2] See `https://developers.facebook.com/docs/graph-api` for further information.
[3] See `https://dev.xing.com/docs/resources`

in Table 2.

Table 2: Layered Reference Framework – Data Acquisition Layer.

Layer Element	Exemplary Technologies
Data Integration Tools	Apache Sqoop (http://sqoop.apache.org/)
	Microsoft SQL Server Integration Services
	Pentaho Data Integration
	Talend Open Studio for Big Data
Messaging Systems	Apache Kafka (http://kafka.apache.org/)
CEP Engines	Apache Flume (http://flume.apache.org/)
	Apache Storm (http://storm.apache.org/)

Batch ingestion has been done for decades in traditional Business Intelligence and Analytics (BI&A) environments (cf. [54]), is very well researched (cf. [37]) and is widely understood. Usually, data flows like ETL, Extract-Load-Transform (ELT), or Extract-Transform-Load-Transform (ETLT) are specified (cf. [56, 33]). Which of these order variations to use is determined by the use case and its data characteristics [42]. Most traditional tools such as Microsoft SQL Server Integration Services (SSIS) and Pentaho Data Integration (PDI) allow integration of both, structured and multi-structured content, between traditional file systems and RDBMSs. Connections to new, distributed types of Big Data storages such as Hadoop Distributed File System (HDFS) [4] and HBase [2] can be established using new technologies such as Apache Sqoop [7].

Real-time ingestion of data-in-motion differs severly from batch-processing and pushes processing and analytics down to the acquisition layer such that the data is essentially processed before it is stored [42]. This is done because it is not reasonable to store all incoming events, due to the velocity of up to millions of events per second and the associated large data volume [26].

Supporting technologies for real-time ingestion include CEP engines that search streams of data for predefined events and compute results on the fly as they arrive.[5] Such systems allow essential operations such as aggregation, union, joins, and filtering on input streams to perform predefined analysis, automatic decisions and actions in real-time. By filtering events prior to ingestion, only the information needed is assessed, analyzed, and eventually stored [42] [33]. Typical use cases are early warning systems [19], fraud detection (e.g., large withdrawal from bank accounts), mouse clicks on website, security systems, and the assessment of new tweets. In general, this is used when the system must decide immediately whether to disregard an event or perform an action as the situation does not allow to wait for human interaction [42].

In between CEP engines and traditional batch-oriented ETL tools are messaging systems. They do not provide functionality for processing of data streams but rather serve as a messaging queue between systems to ensure that no message gets lost. Such tools are oftentimes used to enqueue events and messages from external sources before they are processed by a CEP engine. They furthermore allow communication using a publish-subscribe paradigm between loosely coupled parts of a system [38].

2.3 Data Storage Layer

The data storage layer deals with technologies for persistent data storage in Big Data infrastructures. The main differentiating dimensions are volume and variety. Variety distinguishes between

[4] See [36, 1]

[5] This can be compared with an ETL pipeline that has near-zero latency [33].

different types of storages, namely distributed file systems, Not-Only SQL (NoSQL) data stores, and RDBMSs. These are ordered with increasing data structure flexibility from right to left within the layered reference framework. While structured data is well supported by RDBMSs, multi-structured data requires NoSQL or distributed file systems. NoSQL data stores are particularly sub-divided into key-value, document, graph-based and column family stores. The expected overall data volume determines if horizontal or vertical scaling systems are required [79]. In case of horizontal scaling (see [79, 60]) for multi-structured data, the maximum supported data volume is used to order NoSQL and distributed file systems with increasing capabilities from right to left. Exemplary technologies for different layer elements are given in Table 3. The ones in brackets are not explicitly included in the selection framework introduced later, but will be introduced in future versions (cf. Section 3).

Table 3: Layered Reference Framework – Data Storage Layer.

Layer Element	**Exemplary Technologies**
SMP RDBMS	Microsoft SQL Server, (MySQL)
MPP RDBMS	Greenplum, (Vertica, Teradata)
NoSQL Key-value Store	Riak
NoSQL Document Store	MongoDB
NoSQL Column-family Stores	HBase
NoSQL Graph Databases	Neo4J
Distributed File Systems	HDFS

RDBMSs can be categorized as Symmetric Multi Processing (SMP) RDBMSs and Massively Parallel Processing (MPP) RDBMSs [42] [49]. SMP RDBMSs make use of vertical scaling, while MPP RDBMS scale horizontally (cf. [76]).

MPP RDBMS are best suited for large Data Warehouse (DWH) applications and in-database analytics, in particular for Big Data environments, while they still exploit the commonly known and well understood relational data model [42] [49]. This is, among others, due to horizontal scaling which increases performance and throughput [79] through inter-node parallelism [22]. Also, they can be combined with traditional OLAP tools.[6]. However, MPP databases typically require their own special purpose hardware [42, p. 16] and need specialized linkage [22] which result in higher costs. Examples for MPP databases are Teradata, Netezza, Greenplum, Vertica and SAP Hana [49] [28]. MPP RDBMS are designed for structured data, not multi-structured data [30, 49]. Nevertheless, MPP RDBMSs are still relevant for Big Data, as long as the workload focusses on structured data.

For multi-structured data, other techniques like NoSQL data stores and distributed file systems are more promising. The latter usually allow any kind of workloads stored within files [30]. This makes them most suitable for exploratory analysis, which can be used to extract structure from multi-structured data, that can be stored and analyzed using other technologies such as MPP RDBMSs [42]. Distributed file systems allow multiple clients to access files and directories provided on several hosts sharing a computer network [58]. A prominent example for such a system is the HDFS. Key features are automatic data distribution, high availability, fault tolerance, and high throughput access [16]. It allows to dynamically scale up and down while the system automatically re-distributes the data [49]. Compared to MPP RDBMSs, HDFS storage is cheap, requires no licensing costs, and runs on commodity hardware.

In between MPP RDBMSs and distributed file systems are NoSQL data stores. They represent a new category of database systems that includes four different types: key-value, document,

[6] Microsoft SQL Server Analysis Services (SSAS) can for instance directly connect to Teradata. See `https://msdn.microsoft.com/en-us/library/ms175608.aspx` for further information.

and column-family stores as well as graph databases [78, p. 122] [73]. Each of them is specialized for specific purposes and workloads. Therefore, NoSQL gave rise to the polyglot persistence approach, where different data stores are used depending on situation and workload [75]. Features of NoSQL include low latency, low-cost commodity nodes, and the ability to deal with multi-structured data [57]. On the one hand, they allow to easily increase performance linearly with number of nodes. With this, front-end applications can frequently and interactively query the database with low latency [90]. Yet they lack standards and are reported to have bad analytical performance [57].

High performance real-time support for read and write operations can be achieved by using in-memory storage functionality. The key idea is to eliminate slower storages on lower levels of the storage hierarchy [60]. In-memory databases load their entire data into memory on startup and use it as their primary storage to achieve permanent higher velocity and lower latency on read operations [60]. Due to their enhanced speed, they enable processing of higher data volumes in shorter time such that they are most suitable for data-in-motion scenarios (e.g., streaming data from sensors). In combination with horizontal partitioning, their performance increases almost linearly to the number of nodes. Overall, databases with in-memory capabilities are highly relevant in the context of Big Data as they directly address the volume and velocity dimensions of the original 3 Vs (Volume, Variety, and Velocity) [87].

A survey by KING and MAGOULAS with data analysts and scientists from 2014 [55] reveals that SQL is used by 42% of the respondents while HDFS is only used by 23%. Similarly, a Jaspersoft survey shows, that most popular storage systems within enterprises are RDBMS (56%), MongoDB (23%), MPP RDBMSs (14%), and HDFS (12%) [77]. Conclusively, RDBMSs have not been replaced by other tools. They are still the cornerstone of data analytics, even in the Big Data era.

2.4 Data Processing Layer

This layer includes technologies that are responsible for the execution of data operations such as read, write, and delete, where the main differentiating dimensions are velocity and variety. Variety determines between database and file-based processing. While structured data can be processed using database processing of RDBMSs, multi-structured data is usually stored as files and processed within distributed file systems or NoSQL stores. File-based approaches are particularly sub-divided into batch, unified, and stream processing, depending on the velocity requirement for first results in descending order. Associated processing technologies are abbreviated as Batch Processing Engines (BPEs), Unified Processing Engines (UPEs), and Stream Processing Engines (SPEs) respectively. As the data generation speed must fit the data processing speed for some applications [49], they must be carefully chosen with regard to the use case at hand. Exemplary technologies for different layer elements are given in Table 4.

Table 4: Layered Reference Framework – Data Processing Layer.

Layer Element	**Exemplary Technologies**
SQL Processing	RDBMSs
Batch Processing	MapReduce
Unified Processing	Spark
Stream Processing	Storm

A distributed processing engine can be seen as an infrastructure rather than a tool. It is an enabling technology which can be used or build upon, for instance by analytical tools, which employ large scale machine learning algorithms. Big Data necessitates the use of distributed

technologies [19]. New distributed processing technologies constantly emerge [31].

Database processing utilizes functionalities of underlying databases to perform operations over data within their repositories [36]. Costly data movement is not necessary. Functionalities includes typical SQL operations such as joins or aggregations (e.g., *Sum*) and groupings [36, p. 356]. Some databases also support enhanced functionalities such as regular expressions [36] or user-defined functions (UDF) [36].

When combined with MPP RDBMSs, database processing is considered even faster and more efficient than file-based in-memory processing with large datasets [36]. It is therefore a reasonable choice for the deployment of machine learning algorithms. In contrast, file-based processing cannot be done with off-the-shelf software [56]. As the data is rarely structured and diverse, it requires custom coding to derive structure and meaningful insights, as in the approaches described next.

Batch processing is used in situations where the entire data is stored prior to analysis [49]. BPEs are capable to handle large amounts of data-at-rest. Algorithms divide it into chunks and process each of them individually on its own machine to generate intermediate results which are eventually aggregated to a final result. Such execution jobs are predefined by programmers, given to the system, and executed over a longer period of time. They cannot be adjusted while execution is in progress. MapReduce [34] is a representative for BPEs.

Stream processing handles high frequency data-in-motion and is used in situations where immediate results are required [31]. Although it is considered challenging to build a real-time streaming architecture [16], organizations frequently move towards collecting and processing real-time data [77]. Apache Storm [5] is a representative for SPEs.

Unified processing aims to combine the advantages of batch and streaming into a single system that enables to process both, large amounts of data-at-rest and data-in-motion. UPEs provide a single programming model for all purposes and use micro-batches to simulate stream processing. Such systems do not provide real-time but near-real-time. While the former seeks to guarantee results within application-specific time constraints, the latter does not. Unified processing furthermore aims to provide users with interactive query capabilities and fast answers, even for large amounts of data-at-rest [16]. Thus, engines in this category employ in-memory storage to better support low latency queries and iterative workloads such as machine learning [59]. This is also denoted as iterative-batch processing [59]. A well-known representative for UPEs is Apache Spark [4].

2.5 Data Analytics Layer

The data analytics layer comprises technologies responsible for the value generating process of the adaptive Big Data value chain introduced earlier. Such technologies uncover hidden patterns and unknown correlations to improve decision making [49] and are a means for implementing Big Data use cases. Data analytics is differentiated by two dimensions: the type of data analytics and the generation of machine learning. The former distinguishes (cf. [86] [83]) technology by their support for descriptive (cf. [67, 24, 83]), predictive (cf. [86] [83]), and prescriptive (cf. [85, 83]) methods, which are eventually condensed to BI and advanced analytics. BI analytics focusses on descriptive analytics (e.g., OLAP), while advanced analytics focusses on predictive and prescriptive analytics [10] [47]. Advanced predictive or prescriptive analyses ´typically employ machine learning (cf. [86] [62] [36]). Machine learning methods, among others, include [21] classification (cf. [40, 50]), regression (cf. [66]), topic modelling (cf. [29] [36]), time series analysis (cf. [36]), cluster analysis (cf. [36], [32, 40]), association rules (cf. [66] [36]), collaborative filtering (cf. [84, 14, 50]), and dimensional reduction (cf. [74, 89]). Advanced analytics can be further described by a maturity model proposed by AGNEESWARAN [13], that distinguishes analytical tools

into three generations of machine learning as follows:

1st Generation Machine Learning (1GML) requires the data workload to fit into memory of a single machine. Such tools are restricted to vertical scaling (cf. Section 2.3), which is a drawback when considering Big Data. Tools in this group were usually developed before Hadoop and are referred to as *traditional analytical tools*. Usually, vendors try to enhance or re-engineer their product in a way that allows the usage of Big Data. Mostly, connectors are added that allow read and write operations to HDFS while the analysis is still performed within the tool. Hence data is exported from storage, analyzed, and later re-imported.[7]

2nd Generation Machine Learning (2GML) enhances 1GML with capabilities for distributed processing across Hadoop clusters. In contrast to 1GML, data remains at its location while the code execution is divided and processed on each required data node in parallel.[8] Tools in this class are denoted as *over Hadoop* [13]. Many algorithms do not translate easily into MapReduce [59]. While non-iterative algorithms can be translated into efficiently performing series of MapReduce operations, iterative algorithms such as machine learning cannot. Thus, the expected performance for such workloads is poor.

3rd Generation Machine Learning (3GML) enhances 2GML with capabilities to efficiently perform distributed processing of iterative algorithms. This class is referred to as *beyond Hadoop*. Associated tools such as Spark use more advanced distributed processing methods or in-database execution to cope with some of the disadvantages that come with MapReduce.

Sample technologies for different layer elements and machine learning generations are given in Table 5[9]. Usually, tools evolve over time due to re-engineering efforts by vendors. For instance,

Table 5: Layered Reference Framework – Data Analytics Layer.

Layer Element	Exemplary Technologies
OLAP Tools	Microsoft SSAS, Pentaho Mondrian
1GML	R, RapidMiner, KNIME, SAS, WEKA
2GML	Mahout (MapReduce)
3GML	Mahout (Spark / H_2O / Flink), MLlib, H_2O ML, Flink-ML SAMOA, MADlib

Mahout just recently evolved from 2GML to 3GML as it now supports processing on Spark, Flink and H_20 along with MapReduce. As these engines support efficient execution of iterative machine learning algorithms, Mahout is classified into two layer elements.

The distinction between BI and advanced analytics is supported by a study of KING and MAGOULAS [55]. According to them, traditional data analysts use commercial tools such as Excel, Microsoft SQL Server, and Tableau for explanatory BI for descriptive analytics. On the other hand data scientists (cf. [88]) utilize open source tools like R, Apache Hadoop, and scalable machine learning such as Apache Mahout (see also [3]).

BI analytics is about dicing, slicing, drill-up, drill-down, and drill-through operations over cleaned historical data using a predefined multidimensional model [36] [26]. This can be done using server-based OLAP Engines such as Microsoft SSAS and Pentaho Mondrian[10]. For small amounts, simple off-the-shelf software like Excel can be sufficient.

[7] This is referred to as *data-to-code*.
[8] This is referred to as *code-to-data*.
[9] All tools are classified without extensions. Extensions could allow to tools be classified in a higher tier, e.g., Revolution R, which enables distributed execution over Hadoop clusters [51]
[10] See `http://community.pentaho.com/projects/mondrian/`.

Big Data analytical solution can be differentiated offline and online analytics [12] [30] as well as combined approaches (cf. [79]). Online analytics is used for real-time environments that require low latency for results, especially with data-in-motion. Offline analytics usually employs batched processing for ingestion, transformation, and analytics.

While latency (cf. [59]) is the most important factor for online-analytics, throughput is essential for offline-analytics [25]. Latency highly depends on the technologies for processing and storage on the corresponding layers of the layered reference framework. While online-analytical systems usually operate on SMP, MPP, and NoSQL databases using in-database, stream, or unified processing, offline-analytical tools usually employ distributed file systems in combination with batched processing [42].

A survey among data analysts and data scientists from 2014 [55] reveals that in-database analytics with SQL is used by 71% of the respondents, while the next high ranked tool, R, is only used by 43%. Only 7% of the respondents use Mahout. NoSQL and Hadoop may have solved the storage problem for large amounts of raw data, but still seem unable to sufficiently fulfill needs of business users with regard to data analytics.

3 The S.T.A.D.T. Selection Framework

This section introduces the S.T.A.D.T. Selection Framework (SSF), which aims to guide technology selection in the Big Data era. It seeks to find a set of valid solutions for given Big Data use cases. SSF is based on the layered reference framework presented in Section 2 and consists of two parts: a business and a selection process. Figure 3 provides an overview of the framework.

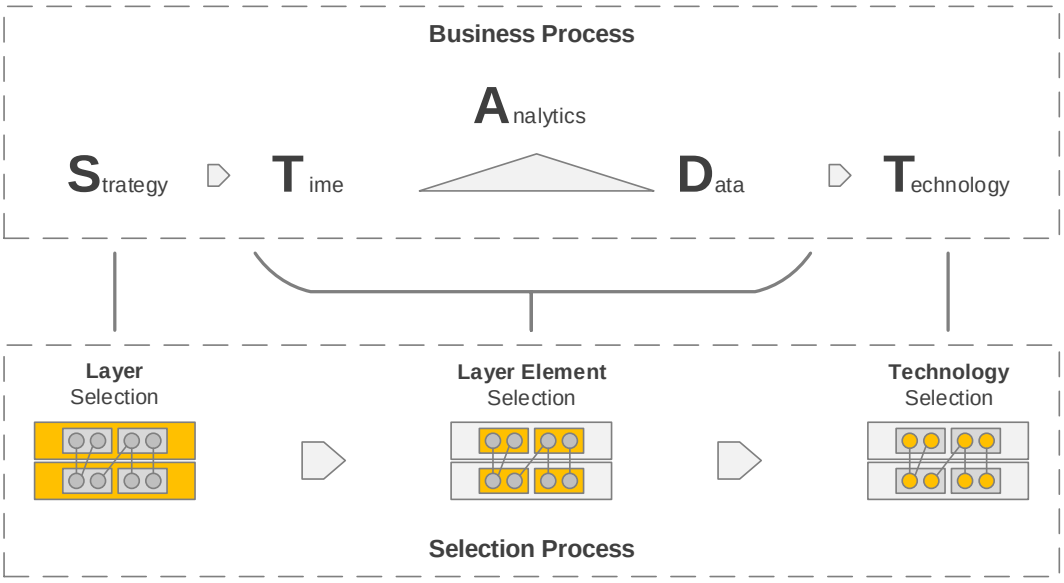

Figure 3: The S.T.A.D.T. Selection Framework

The business process is partly based on Marr's *SMART Model* [63], which can be used as a guideline on how to evolve towards a Big Data driven smart business. However, SSF as presented

here is fundamentally different, except for the general idea of the first two process steps of *strategy* and *measures* (here: *data*). SSF aims at selection of technology, not at business transformation, and hence reinterprets and renames the process steps by MARR to reflect this change (*S*trategy, *T*ime, *A*nalytics, *D*ata, and *T*echnology). In this, it is similar to the *GOBIA* method of [41], which also combines a reference architecture with a development process. Notably, the process of technology selection could be extracted from the SSF and be seamlessly embedded as final step in the GOBIA method development process (*GOBIA.DEV*, cf. [41]).

The business process of SSF serves as a roadmap for companies who want to select technology for their Big Data use case at hand. It starts with the overall strategy, i.e., business objectives to be achieved [63]. Depending on the strategy, measures of input data, suitable analytics and required response times are derived and used to select suitable classes of storage systems, analytical tools, and processing engines respectively. Finally, a suitable technology mix is selected that corresponds to the input use case.

All steps of the SSF's business process have implications on technology selection. They filter the layered reference framework and thereby narrow the search space for valid solutions. First, the overall strategy is used to select relevant layers. Secondly, *data* measures, *analytical* requirements, and response *times* determine relevant layer elements. Finally, the remaining technologies are filtered by their interdependencies (e.g., compatibilities), individual properties as well as user preferences to derive the final solution space.

There is no single decision tree that determines the right technology mix with respect to all conceivable circumstances [75]. Thus, SSF aims to find the set of best suited technologies in each selection step. It does not seek a complete list of possible technology sets for a use case. As the great potential for Big Data arises when different technologies are used in concert [43], it attempts to recommend at least one tool on every required layer for further investigation.

The remainder of this section follows the structure of the SSF business process. It starts with strategy (cf. Section 3.1), defines requirements on (response) times (cf. Section 3.2), decides on analytics (cf. Section 3.3), then continues with data (measures) (cf. Section 3.4), and finishes with selection of suitable technologies (cf. Section 3.5). Each process step is elaborated with tangible executable actions and their resulting implications on technology selection. The complete SSF process is illustrated in 5 and 6, in the form of flow charts. It subsets are elaborated in the following.

3.1 Strategy

This section deals with Big Data strategies and their transformation into executable tactical plans. It describes different building blocks and associates each with required layers and steps of the SSF's business process. While the development of a specific Big Data strategy is out of scope, this section still provides a brief strategy guideline as well as a description of organizational requirements and impacts.

Overall, strategy is essential and drives the selection of technology [41]. Big Data initiatives need to be aligned with the overall business strategy [42]. Prior to analysis of Big Data, it's essential to derive relevant and business related questions that need to be answered [53] (see also [63] [35] [42]).

Once a strategy has been settled and a business relevant question been derived, it can be translated into an executable tactical plan. Initial building blocks are *storage*, *processing*, and *analytics*, because they represent categories for typical Big Data use cases respectively Big Data products used in these use cases. These building blocks can be arranged in any sequence of arbitrary length to solve a business relevant question. Each block starts a new iteration of the SSF pro-

cess and covers a unique functionality. Storage for instance acquires and stores data from any source. It makes sure that the data is stored in an appropriate data store that fits the data at hand. Processing transforms data from one state to another within the data source it resides, e.g., from multi-structured data to structured data. Finally, analytics performs machine learning algorithms to create additional value. Figure 4 provides an sample tactical plan.

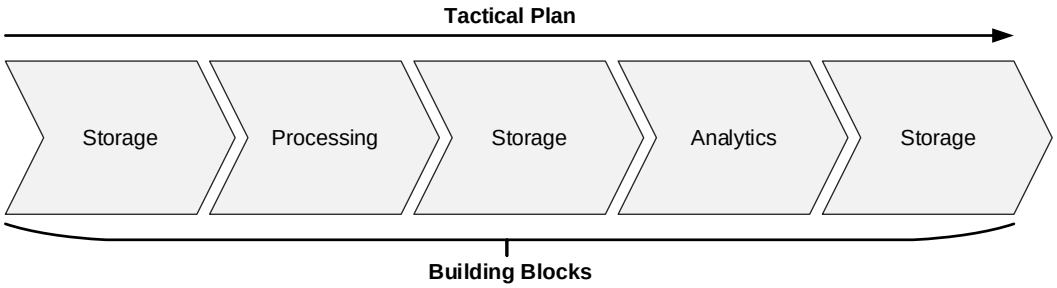

Figure 4: Building Blocks for Tactical Plans: Storage, Processing and Analytics.

First, a storage building block acquires for instance multi-structured data from an external source and stores it in a suitable storage system within the infrastructure, e.g., HDFS. Secondly, a processing building block transforms the data into a structured format, while it remains within HDFS. The third iteration takes the resulting processed data from HDFS as source and stores it in the most suitable storage system of the infrastructure, e.g., into a RDBMS. The subsequent analytics building block performs machine learning algorithms on the data stored in the RDBMS. Such blocks may also employ a distributed processing engine to fulfill their task. Finally, the storage building block seeks the best suited system to store the analytical outcome.

Each type of building block seeks technologies at different layers of the layered reference framework (cf. Section 2). The assignment of building blocks to layers is given in Table 6. Storage for

Table 6: Building Blocks – Layer Assignments.

Layer	Storage	Processing	Analytics
Data Analytics Layer	✖	✖	✔
Data Processing Layer	✖	✔	✔
Data Storage Layer	✔	(✔)	(✔)
Data Acquisition Layer	✔	✖	✖

instance seeks compatible technologies on two layers: the data acquisition layer and the data storage layer. Analytics searches for compatible technologies on the data processing and the data analytics layer while considering a specific storage system as input source. This is indicated by using parentheses. Processing can be described analogously. Note that the data generation layer is not listed in Table 6 as it does not contain technologies but data characteristics, which are used for filtering layer elements in Section 3.4.

Different types of building blocks also require other SSF process steps. Their mappings are given in Table 7. For each building block, the associated steps need to be executed in their corresponding top-down order to receive a suitable technology mix. This is automatically taken care of by the process flow charts in figures 5 and 6. Storage building blocks for instance rely

Table 7: Building Blocks – Process Step Assignments.

SSF Process Step	Storage	Processing	Analytics
Measures	✔	✖	✖
Analytics	✖	✖	✔
Response	✖	✔	✔
Technology	✔	✔	✔

solely on the data and technology steps, while analytics building blocks require the latter three steps of analytics, time, and technology. Required steps for processing building blocks can be derived analogously.

The decomposition of a use case into sequences of storage, processing, and analytics has at least two advantages. First, it narrows the search space for each block which makes especially large and extensive Big Data use cases more tangible. Secondly, the decomposition only requires to understand the purpose of each building block and can be carried out by business staff without extensive IT expertise.

However, decomposition may lead to an over-optimizing of solutions as building blocks are handled in isolation. The result may be many "'locally optimal"' pieces of technology, which each require specially trained staff and integration. Trade-offs have to be made to select few, yet manageable ones. But this consideration is out of scope for this work and not yet covered by the SSF.

3.2 Time

This section handles the selection of best-suited layer elements with regard to processing in distributed environments. Hence it is only needed in cases where the underlying data is stored in distributed storage systems [73]. In such cases, the selection depends on the assessment of required response times to be derived from the use case. If the data is not stored in distributed storage systems, then distributed processing is also not required. In such cases, the whole processing layer is deselected and not used in the final technology selection step (cf. Section 3.5). The process is illustrated in Figure 6 below and elaborated upon in subsequent paragraphs.

In case of distributed data, users need to specify their requirements for latency (cf. Section 2.5). Essentially, they need to determine if the latency of a result is a fundamental measure for their use case at hand. If so, the use case needs to be assessed to determine if specific time constraints are prescribed that must be guaranteed. In cases where real-time results are needed (i.e., where short response times must be guaranteed), SSF selects stream processing as the most suitable layer element. In cases where near-real-time results are sufficient and small random time gaps (e.g., a few seconds) between data arrivals and processing results are acceptable, SSF selects SQL [79] and unified processing. The latter uses micro-batches to simulate streaming (cf. Section 2.5). This comes with more latency but also with less complexity compared to stream processing. Unified processing furthermore unifies the programming model for batch and streaming which makes it a more universal tool. As such, it should be preferred over stream processing where possible [59].

If low latency results are not fundamental for a given use case, it is not recommended to use SPEs due to their complexity [59]. In such situations, batch or iterative-batch processing are more suitable (cf. Section 2.4). Such engines come with higher latency but allow high throughput [64]. The choice between the two depends on the need for iterations. Ad-hoc queries and most

machine learning algorithms are iterative in nature [79]. Thus, SSF selects unified and SQL processing in case of their presence. In all other cases, the usage of batch processing is sufficient, such that the corresponding layer element is selected.

3.3 Analytics

This section prepares the selection of suitable machine learning tools. It aims to select best suited layer elements on the corresponding layer of the layered reference framework. The selection depends on three factors: the required type of analytics, the expected data volume and the required machine learning methods (cf. Section 2.5). The process is illustrated in Figure 6 and discussed in the following paragraphs.

The first decision determines between BI and advanced analytics (cf. Section 2.5). The former represents descriptive methods while the latter emphasizes predictive and prescriptive analytics. In case of descriptive analytics, traditional BI technologies such as OLAP tools are naturally supportive and thus selected. In case of predictive or prescriptive analytics, the required machine learning methods need to be derived to select appropriate tools in the later technology selection step of the SSF [79]. For instance, if a use case aims to provide recommendations, then it usually employs collaborative filtering. Clustering can be used if a use case needs to find similar entities, e.g., groups of customers.

The expected data volume determines the minimum required generation of machine learning for a given task (cf. Section 2.5). While 1GML tools are sufficient for data workloads that can be analyzed on a single machine, 2GML or 3GML are required in situations that determine horizontal scaling (cf. Section 2.5). The latter two need distributed processing engines while 1GML does not. Such tools process data in local memory and just connect to arbitrary storage systems for read / write operations. If a task can be analyzed on a single machine, then that's the recommended solution. 1GML tools are easier to handle, more mature, and more extensive in their machine learning capabilities than horizontally scaling tools [59]. So, 2GML and 3GML technologies are only recommended in situations that require distributed processing due to high volumes. The actual choice between the two is implicitly further refined in the *time*-step of the SSF by selection of processing types (cf. Section 3.2).

There is a variety of different tools for advanced analytics available on the market. Due to their large numbers, it's not reasonable to handle them in this work simultaneously. Instead, a representative subset is selected and evaluated. KDNUGGETS [11] considers itself as one of the top web resources for analytical software and conducts a poll about their usage every year. The results for 2015 are based on $2,800$ votes by users of the data mining community who have chosen from a record of 93 different predefined tools [71]. With some adjustments, these results can serve as the foundation for tool selection in the thesis at hand. First, formal languages like SQL, Python, Perl, Pig, and Lisp are removed from the list. Secondly, all 1GML tools other than the top 3 with regard to usage are removed. The same holds for Big Data processing engines and analytical tools without capabilities for advanced analytics (i.e., predictive or prescriptive methods) (cf. Section 2). Furthermore, spreadsheet tools with a focus on office users like Excel are excluded. Finally, the list is extended by promising findings during literature research and interviews for this work. Examples for such include MADlib, Flink ML and SAMOA. Additionally, Microsoft SSAS is included as a representative for OLAP engines. Table 8 provides the resulting list of analytical tools and their classification for machine learning generations.

The gap between 1GML and 2GML / 3GML tools with regard to their usage suggests that most analytical use cases are still solved with traditional tools, even in the Big Data era.

Table 8: Analytical Tools – Classification and Usage in 2015 (Source: based on [71])

Rank	Usage	Analytical Tool	ML Generation
1	46.9%	R	1GML
2	31.5%	RapidMiner	1GML
3	20.0%	KNIME	1GML
4	9.7%	Microsoft SSAS	OLAP
5	3.3%	MLlib	3GML
6	2.8%	Mahout	2GML / 3GML
7	2.0%	H_2O ML	3GML
8	-	Flink ML	3GML
9	-	SAMOA	3GML
10	-	MADlib	3GML

3.4 Data

This section deals with measurements of data characteristics, which are used to select layer elements on the data acquisition and the data storage layer. The overall goal is to find layer elements that are best suited for the data at hand [43]. For this, a proper understanding of data characteristics is key to success [45].

A starting point are the well-known 3 Vs of Big Data [36]: volume, variety, and velocity. While velocity distinguishes between data-in-motion and data-at-rest [79], variety distinguishes between structured and multi-structured data [36] (see also Section 2.1). Furthermore, the volume dimension determines how much scalability is needed. It distinguishes between horizontal and vertical scaling (cf. Section 2.3) [59]. As the desired infrastructure must be scalable for the future, all decisions on data characteristics have to support the current and the future dataset [30]. Thus, not the current state needs to be measured, but the expected one.

The assessment of the 3 Vs follows a three-step process, as illustrated inside Figure 5. First, the velocity dimension is inspected. It determines between data-in-motion and data-at-rest. Both require fundamentally different technologies and methods for data acquisition (cf. Section 2.2). While data-in-motion leads to the selection of CEP engines and messaging systems [64], data-at-rest selects the layer element for traditional data integration tools. The respective flow chart part in Figure 5 highlights all process steps for selections with orange color.

Secondly, the volume dimension needs to be inspected. It determines whether a Big Data platform is required or whether the data can be processed on a single machine [79]. Big Data technologies should not be used if there is no need to do so [16] [75]. It is a magnitude easier to solve problems with traditional SQL based systems or by using script-based processing of multi-structured data on the local file system of a single machine [59]. These tools are less complex [75], more mature, widely understood, and broadly available. In a nutshell, if the data volume allows storage and processing on a single machine, then that's the recommended solution. In this case, SSF selects RDBMSs and recommends to use local non-distributed file systems in combination with scripts for data transformation.

In cases where the overall expected volume exceeds a single machine's capacity with regard to storage, CPU, or memory [16], the variety dimension needs to be inspected to select a best suited storage system [53]. While structured data is well-suited for MPP RDBMSs, multi-structured data requires NoSQL stores or distributed file systems. The selection for multi-structured data can be further refined by assessing the expected number and size of files [75]. For small numbers of large files, it is suggested to use distributed file systems. For large numbers of small files, the recommendation is to use NoSQL stores. MARZ [64] explains that Hadoop can be a magnitude

slower for processing of many small files compared to few big files, although both scenarios have the same overall volume. Reasons for this include high latencies for individual record lookup in HDFS [22]. The framework therefore suggests to select distributed file systems for large files and NoSQL stores for large amounts of small files in accordance with the mentioned authors. However, there are newer distributed file systems with in-memory capabilities for random and fast data access such as Alluxio[11]. For such systems, the distinction for number and size of files is less important. If they win recognition, they possibly form a new class of storage systems in the layered reference framework for further distinction. However, this is not yet included in its current version.

The choices for layer elements are derived from interviews [61] and from a comprehensive literature review. BEGOLI and HOREY [18] for instance provide some principles for good Big Data architectures. The authors especially give advice on the influence of data variety on technology selection. They suggest to use Hadoop for unstructured data, MPP RDBMSs for structured data, and NoSQL stores for semi-structured data. Similarly, FERGUSON [42] suggests to align data characteristics with storage and recommends to use MPP RDBMSs for complex analysis of structured data and Hadoop for multi-structured data, especially for storage and processing tasks of archive data. He also discusses the differences between data-at-rest and data-in-motion and their relation to CEP engines, stream and batch processing. CHAN [22] contributes to the discussion and argues about the impact of velocity on technology selection. The author introduces an integrated conceptual architecture for stream and batch processing. Finally, MARZ [64] suggests the Lambda Architecture, which unifies processing of data-at-rest and data-in-motion on a conceptual level.

3.5 Technology

This section handles the final step of the SSF business process which eventually selects a suitable technology mix. The selection follows a three-step process as illustrated in the lower part of Figure 6. First, suitable machine learning tools are selected in cases where analytics is required. Secondly, the storage system that holds the input data is selected if the current SSF iteration handles an analytics or processing building block. Finally, interdependencies are inspected to find compatible technology mixes between required layers of the layered reference framework. The results can be further refined by investigation of technology-specific characteristics. Each process step is described in the following paragraphs.

If the current SSF iteration handles a building block for analytics, suitable analytical tools must be selected. Recall the assessment for machine learning methods performed in the analytics-step (cf. Section 3.3). A suitable tool must support the identified required methods. For proper selection, Table 9 and Table 10 provide mappings between analytical tools and supported machine learning methods. The SSF process requires all technologies that enable the required methods of the use case to be selected for the later compatibility check.

Note that all assessed 1GML tools support any of the machine learning methods. As most Big Data analytical tools offer less functionality compared to solutions that operate in-memory on a single machine, Big Data technologies are less promising for small data [59], which is another indication that they should only be used when certainly needed (cf. Section 3.3).

The mappings in Tables 9 and 10 are based on the work by LANDSET et al. [59] and RICHTER et al. [74] who assess analytical tools with regard to machine learning support. This work enriches their findings with additional tools and methods. It furthermore refines their results with information collected from the individual websites and documentations of the tools.

[11] See `http://alluxio.org/`.

Table 9: Supported Machine Learning Methods for 1 GML and OLAP Tools (based on [52]).

ML Method	RapidMiner	KNIME	R	Microsoft SSAS
Regression	✔	✔	✔	✔
Time Series	✔	✔	✔	✔
Classification	✔	✔	✔	✔
Topic Modeling	✔	✔	✔	✖
Cluster Analysis	✔	✔	✔	✔
Association Rules	✔	✔	✔	✔
Collaborative Filtering	✔	✔	✔	✖
Dimensional Reduction	✔	✔	✔	✖
ML Generation	1GML	1GML	1GML	OLAP

Table 10: Supported Machine Learning Methods for 2/3 GML Tools (based on [59] and [74]).

ML Method	Mahout (MR)	Mahout (Spark)	Mahout (H_2O/Fl)	H_2O ML	Flink ML	MLlib	MADlib	SAMOA
Regression	✖	✖	✖	✔	✔	✔	✔	✔
Time Series	✖	✖	✖	✔	✖	✖	✔	✖
Classification	✔	✔	✖	✔	✔	✔	✔	✔
Topic Modeling	✔	✔	✖	✖	✖	✔	✔	✖
Cluster Analysis	✔	✖	✖	✔	✖	✔	✔	✔
Association Rules	✔	✖	✖	✖	✖	✔	✔	✔
Collaborative Filtering	✔	✔	✖	✖	✔	✔	✖	✖
Dimensional Reduction	✔	✔	✔	✔	✖	✔	✔	✖
ML Generation	2GML	3GML	3GML	3GML	3GML	3GML	3GML	3GML

For simplicity, SSF only uses machine learning methods for mappings. However, each of these methods may include several different specific algorithms that are suitable to fulfill the task. For instance, *classification* can be performed with decision trees, linear and logistic regression, Naïve Bayes, Support Vector Machines (SVMs), gradient boosted trees, random forests, adaptive model rules, and generalized linear models [59]. The framework indicates a tool's support for a machine learning method if one of the enabling algorithms is included. A more comprehensive list of available machine learning algorithms as well as their coverage by processing engines is given by the formerly mentioned authors [59] [74]. If needed, SSF can easily be extended with specific algorithms. However, this is out of scope for the work at hand.

The next process step requires to select the input storage system where the data is located. This is mandatory for processing and optional for analytical building blocks. While the former always performs on data within the local infrastructure, analytical tasks can also be executed on a data stream without prior storage. This is also explained with the adaptive Big Data value chain in Section 2. If the data to be analyzed is located within the local infrastructure, a specific storage system needs to be selected, thus given as input. In case the data is not stored prior to analysis, the storage layer can be omitted for the subsequent compatibility check.

4 An Application Scenario

This section examines an application scenario for SSF and thereby demonstrates the technology selection, which is based on continuous paths through the layered reference model and technology capability mappings. First, the application scenario is introduced. It features a retailer with an existing traditional data warehouse that has been created based on traditional requirements.

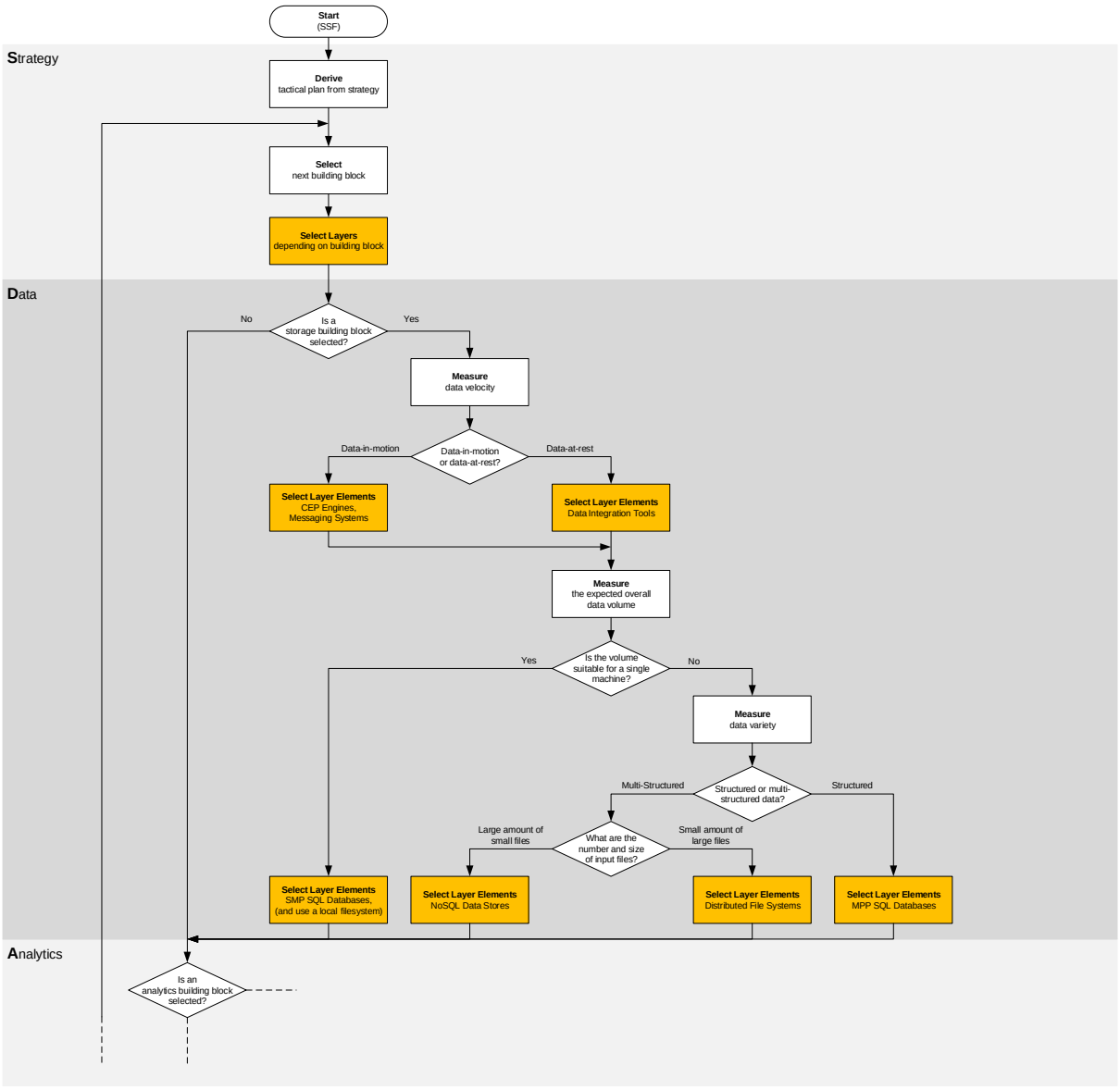

Figure 5: Complete SSF Process – Part 1.

These are used to infuse the SSF process to find a suitable technology mix. This section shows which technological choices SSF suggests in the context of current technologies, and if and to which extend they deviate from the existing choices. Finally, the application scenario is revisited with a new requirement to determine required changes to the underlying technologies to remain compliant with requirements.

4.1 ShopMart Scenario Characteristics

The usage of a traditional data warehouse with traditional requirements is illustrated using fictitious German retailer *ShopMart*. Although the scenario and its assumptions are fictitious, they represent common elements in warehouse architectures and related requirements (e.g., report-

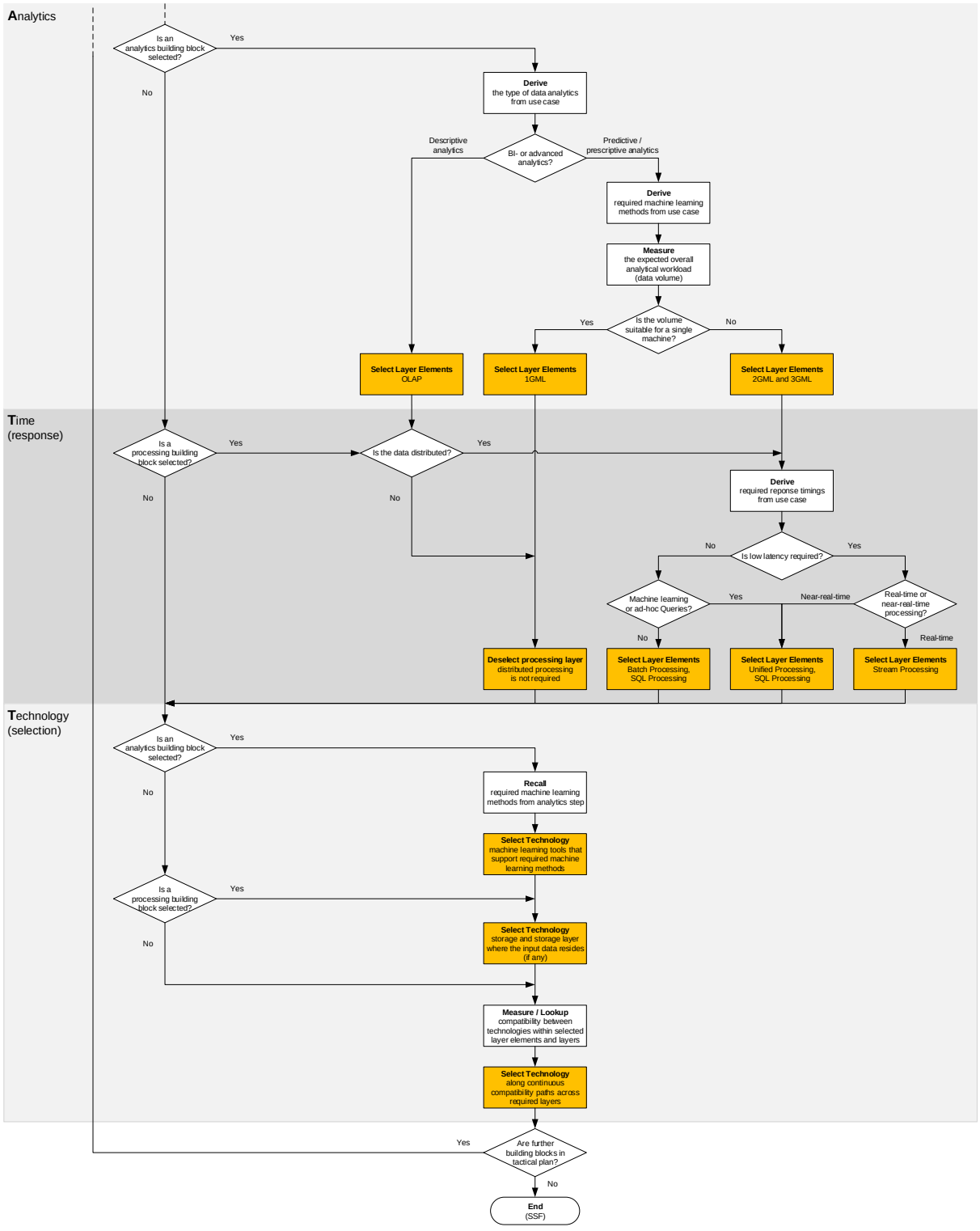

Figure 6: Complete SSF Process – Part 2.

ing or OLAP), which have evolved over time in both research and practice. Thus, the application scenario presented could be applicable to other traditional setups that rely on similar technologies.

The long term goal of *ShopMart* is to become the most profitable retailer in the low price segment in Germany with the highest profit margin. The product selection offered by *ShopMart* appeals to a broad customer base (i.e., not too expensive). To achieve these long term goals, strict cost control mechanisms are employed. This strategy is implemented in its data warehouse with two analytical tools that are represented as tactical plans in SSF. We outline *ShopMart's* goals and requirements next; subsequently, the current warehouse implementation is described. With this, the necessary information for the SSF process can be derived (rather abstract tactical plans and, based on these, data, time, analytics process part information).

1. *ShopMart* points out profit and cost as KPI for each subsidiary, each product, and the combination of the aforementioned. These are used for daily and quarterly reports. To this end, *ShopMart* has an ERP system which collects all transactions (e.g., a customer buying a product) from the subsidiaries. The cash registers push their data either in real-time or asynchronously to the ERP system. From there, the data warehouse receives the data via ETL processes, which perform data cleaning and transformation procedures to generate materialized views that prepare the data for report generation.

2. *ShopMart* monitors and analyses current and historical prices of its various suppliers to select the most cost-efficient supplier for short-term and long-term contracts. The response time requirements are stated as ”as fast as possible” so that new orders can be placed exactly when the time is right. The available warehouse technology allows for a response of one day (daily ETL with analytics in the warehouse) when *ShopMart* built it. To this end, *ShopMart* has various systems in place to capture current prices. For instance, wholesaler B2B online shops are scraped regularly to acquire prices for products purchased via wholesales. The captured data is loaded via an ETL process and the placed in the data warehouse for enhanced analytics. *ShopMart* currently employs time-series analysis to forecast price trends for its products. The results are saved in materialized views, which are refreshed daily, and supplied to a tool that can access these data via SQL.

These requirements are used to derive two more abstract tactical plans as proposed by SSF (see Figure 7 and Figure 8). These do not refer to specific technologies, only to the requirements at hand. That way, the technology selection can be done with SSF, after it is introduced in the following section.

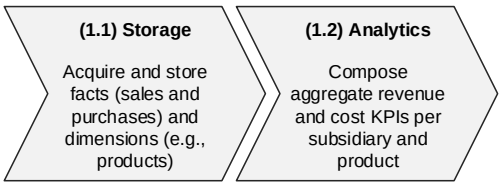

Figure 7: *ShopMart* Tactical Plan for Profit and Cost KPI Goal (1).

4.2 Technology Selection Approach

Once relevant layers, layer elements and perhaps input data sources have been determined with the SSF process, a suitable technology mix can be selected (cf. Figure 6). The selection process checks for compatibilities between candidate technologies within selected layer elements and searches for continuous compatibility paths from the topmost to the lowest selected layer of the

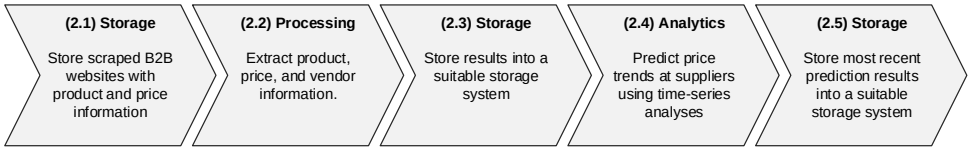

Figure 8: *ShopMart* Tactical Plan for Price Forecasting (2).

layered reference framework. Every continuous path is a valid solution. Figure 9 provides a scenario that represents an analytical building block with persistent input storage. In this example,

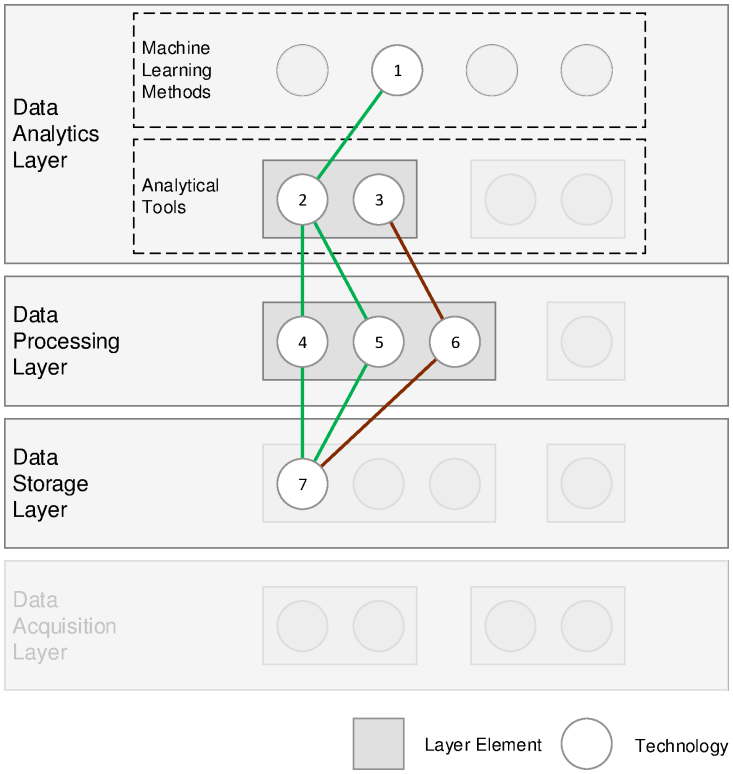

Figure 9: Technology Selection – Search for Continuous Paths.

previous process steps have already selected best suited layer elements. Unselected layers and layer elements are faded out and not considered for the final result. The sample use case requires machine learning method $\{1\}$ and data storage $\{7\}$, which have been provided as input in the corresponding steps of the SSF process. With this preselection, valid solutions include the sets $\{1, 2, 4, 7\}$ and $\{1, 2, 5, 7\}$ as both represent a continuous path from the topmost to the lowest selected layer. The candidate solution $\{1, 3, 6, 7\}$ is interrupted, as the analytical tool named $\{3\}$ does to support the required machine learning method. Thus it is not a valid solution.

This concept of technology selection requires compatibility mappings between technologies at adjacent layers. One example for such is given in Table 11. It provides mappings for analytical tools and distributed processing engines. SAMOA for instance can be employed in combination

with Storm or Flink, while MLlib only supports Spark.

Table 11: Technology Selection – Example for Compatibility Mappings (based on [59] and [74]).

Processing Engine	Mahout (MR)	Mahout (Spark)	Mahout (H_2O/Fl)	H_2O ML	Flink ML	MLlib	MADlib	SAMOA
Spark	✖	✔	✖	✔	✖	✔	✖	✖
MapReduce	✔	✖	✖	✖	✖	✖	✖	✖
Storm	✖	✖	✖	✖	✖	✖	✖	✔
H_2O	✖	✖	✔	✔	✖	✖	✖	✖
Flink	✖	✖	✔	✖	✔	✖	✖	✔
SQL Processing	✖	✖	✖	✖	✖	✖	✔	✖

The general idea for mappings is based on LANDSET et al. [59] who also provide a graph-based compatibility mapping between processing engines, machine learning methods, and analytical tools. This work extends their idea to other layers such as storage and data acquisition to provide a more comprehensive mapping which can be used for diverse and more customized technology selections.

Valid sets of technologies can be further refined with user preferences and technology specific individual properties. Storage systems can for instance be filtered with regard to their preference for consistency, availability and partition tolerance as proposed by the CAP theorem [20, 46]. In case of distributed systems, partition tolerance is mandatory [73]. Thus, users can decide between consistency and availability for their use case at hand and filter results accordingly.

4.3 ShopMart Technology Selection

Applying the SSF technology selection approach to the *ShopMart* scenario at hand yields the following results.

Tactical plan for profit and cost KPI goal (1)

(1.1) Storage. Storage building blocks work with storage and acquisition layers (cf. Section 3). The only input storage here is an operational ERP system out of scope of the analytical system. To decide for layer elements, data velocity, overall volume, and variety need to be clarified upon. *ShopMart* uses a traditional ERP solution (SAP ERP), which uses a structured data format. Data Integration Tools are a suitable data acquisition choice, as the data is at rest there. For *ShopMart* the size of an ERP currently fits inside a single server machine, therefore an *SMP SQL database* is selected for storage.

As for Oracle and SAP ERP products, for instance, accessing their relational SQL databases to extract data is considered possible, albeit challenging (cf. [6]). Furthermore, specialized APIs and connectors can be used to access ERP systems like SAP ERP (e.g., Oracle Business Warehouse offers a connector for SAP [68]). Some ETL tools also offer SAP connectors (e.g., Pentaho Data Integration [70]).

(1.2) Analytics. For this case, an analytics building block is selected. To select it, one must decide between BI and advanced analytics. Standard reporting with KPIs is a typical BI analytics task. Therefore OLAP is selected. Besides dedicated OLAP engines, some data warehouses can be SMP or MPP SQL databases, which could also offer the required functionality (e.g., with SQL:2003).

For *ShopMart* revenue and costs grouped by various dimensions are most important. Both OLAP engines and DWH and RDBMS with respective SQL support can provide this functionality.

For instance, for an SAP ERP system a Oracle Data Warehouse with an SAP connector could be one viable solution that covers both building blocks. Alternatives include other DWH like SAP BusinessWarehouse, which also offer a connection to a SAP ERP. These connectors can act as data acquisition tools. However, it is also possible to use a dedicated ETL tool with SAP support, if more control is necessary.

Overall, it is possible to use the same SQL database for this and the previous building block. The exemplary choice here is a Oracle Database to be used as Data Warehouse with OLAP support, which can represent both storage and analytics requirements.

***ShopMart* tactical plan for price forecasting (2)**

(2.1) Storage. To decide on layer elements for this block, again data velocity, volume, and variety need to be determined. However, certain assumptions also need to be made. Although the requirement is to scrape data ”as fast as possible”, the input data is classified as data-at-rest. One reason is that *ShopMart* actively requests the data and constant polling is inefficient. Also, human staff places purchase orders throughout the day so that a real-time data supply would not lead to increased business value at this point. To estimate the volume and whether a distributed system is needed, the average data volume for all wholesaler B2B websites for several updates a day is estimated. The total size of all pages to be retrieved for one update are the total number of unique products through all subsidiaries, each multiplied by the number of wholesalers that have the respective product on their website. In the worst case, one page needs to be retrieved for each product at each offering wholesaler. To simplify, *ShopMart* is assumed to have a common product portfolio in all branches. The largest average number of articles in a retailer in Germany are approximately 50000 products as of 2013 (cf. [80]). The total number of wholesalers in Germany as of 2014 is approximately 160,000 (cf. [81]), from which ca. 75,000 could be potentially used for retailers (cf. [82]). Depending on the size of each price request (e.g., a regular HTML page is 60 KB in average, cf. [15]) and if requests can be bundled, the size could exceed typical sizes of a single machine. For instance, if all products are requested from 5 % of these retailers in one request of 60 KB each, 220 MB of space would be required. In the worst case, if all products were requested separately from all retailers in one request of 60 KB each, 210 TB of space would be required. This has to be multiplied by the desired update frequency each day, although older raw data can be deleted after it has been further processed. Because of this and to gain flexibility for future growth, a distributed system should be selected. Due to the files being potentially semi-structured and being rather small in size, NoSQL data stores are selected as storage solution. In this case, *Riak* is chosen as key-value store. In a key value store, HTML page data can be stored under a single key to be further processed without introducing HDFS inefficiencies with many small files.

(2.2) Processing Suitable for the underlying processing blocks are Batch Processing and SQL Processing. Low latency is not required for several intra-day updates and extracting information does not require machine learning or ad-hoc queries. The goal is to extract the relevant price and product as well as supplier information from the sources files and to transform these into a more structured format. As the input source is a NoSQL data store, Batch Processing is a suitable candidate for this task. MapReduce in Apache Hadoop is one suitable technology to achieve this and it is compatible with the previous storage choice.

(2.3) Storage This storage building has the goal to store the results from the information extraction in the previous building block. As MapReduce has the potential to reduce information size and to aggregate similar results already (i.e., not too many small files), HDFS could be employed as distributed storage.

(2.4) Analytics The needed time series-analysis is a case of advanced analytics. As the data needed distributed processing before and historical data is retained, distributed processing is set as requirement again, 2GML or 3GML tools are selected for this building block. Of these, for instance, MADlib and H_2O ML support time-series analysis. However, only H_2O ML on H_2O

supports a distributed approach and also HDFS (cf. [48]). Thus, H_2O with its ML library are chosen.

(2.5) Storage As only the most recent analysis data, which is already condensed, should be stored, an SMP SQL database is selected for this task. Due to the nature of needed response time in the process before, also this data is classified as data-at-rest. As H_2O only works with HDFS or local file systems, a data integration must be performed to permanently store the result data. This could be done with a HDFS connector, where a database can use SQL processing to access the result files on HDFS, e.g. in an Oracle database [69].

Comparing the results with the existing architecture
Comparing the choices made with SSF and the implemented system at *ShopMart*, both commonalities and differences can be identified. For the KPI reporting tactical plan, SSF recommends an RDBMS respectively a DWH, which is exactly what *ShopMart* has already built. For these requirements, the choice for traditional SQL technology remains. However, for the second tactical plan and time-series forecasting, the choices differ. It is evident that *ShopMart* has employed the existing data warehouse out of necessity, because suitable alternatives were not available in the past. The selected technologies with the SSF can potentially better fulfil the posed requirements. For instance, an updated forecast could be available several times a day instead of once a day only. Also, the data intake can be scaled more effectively with the proposed technology than a traditional RDBMS. However, besides a better fit to the requirements and data characteristics, other trade-offs are not considered by SSF, although they could be relevant for *ShopMart* or any other company. A smaller fit to the requirements could be worthwhile when the better solution is relatively more expensive. For instance, costs are saved for material and immaterial (i.e., hardware and software), as well as human resources, when the same technology stack is employed. Also, the solution is less complex. For the SSF recommendations, a more heterogeneous architecture and more diverse employee skill set is needed. Moreover, more technologies must be integrated with one another.

4.4 Changing Requirements

To point out how the selected technologies change, a new requirement is added and the SSF process is invoked with it. The new requirement is that *ShopMart* wants to find out how their customers sentiment and attitude towards them has evolved over time. With this information, *ShopMart* intends to verify if strategic decisions negatively or positively influenced their customers' attitude towards them. For instance, overly aggressive cost-cuttings could lead to a negative sentiment over a perceived loss in quality. To measure this, *ShopMart* plans to analyze posts on its Facebook wall and messages sent by users to their Facebook account. Posts and direct messages need to be retrieved by the Facebook API and stored. After this, a sentiment analysis needs to be carried out on this data (see Figure 10).

(3.1) Storage For this storage building block, acquisition and storage layer elements are selected. Data from Facebook can be requested via its Graph API, which returns JSON responses (semi-structured)[12]. While the Facebook pages of *ShopMarts* are regularly visited, actively retrieving a snapshot constitutes data-at-rest, thus Data Integration tools are selected for acquisition. While there are many Facebook messages and posts for *ShopMart* their overall data volume can be expected to fit on a single machine [13]. Therefore, SMP SQL databases are selected for the storage layer element. A specific one could be, in line with the previous recommendations, an Oracle Database [14].

[12] see also `https://developers.facebook.com/docs/graph-api`
[13] See also this estimation of page posts per month on a Facebook page [72]
[14] Notably, Oracle natively supports JSON content in its database - `https://docs.oracle.com/database/121/ADXDB/json.htm#ADXDB6247`

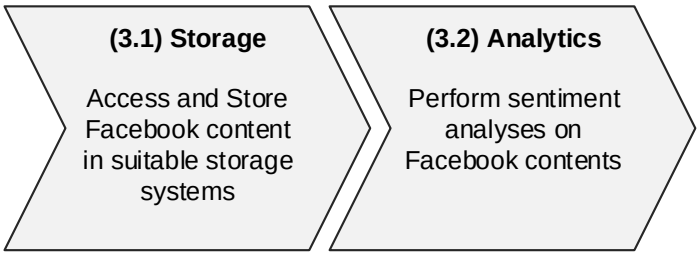

Figure 10: New tactical plan for *ShopMart*.

(3.2) Analytics For this analytics block, a 1GML tool is needed. The data is not distributed, but text analytics is required for a sentiment analyses. RapidMiner supports SQL as input source for analyses (in addition to others as HDFS) and offers support for text mining. Due to this, RapidMiner is selected as analysis tool.

The technology selection for this new tactical plan demonstrates that even new use cases can be enabled by rather traditional technologies. For instance, the Oracle database can be re-used for this tactical plan and no new novel technology is required for storage. However, RapidMiner is a new tool that needs to be properly integrated into *ShopMarts* landscape. While it does not belong to the seemingly modern 2GML or 3GML tool, its capabilities suffice to conduct the needed sentiment analysis.

5 Conclusions

This paper has considered the problem of making an appropriate technology selection for a given big data application, and has introduced a corresponding framework, denoted S.T.A.D.T. Selection Framework (SSF). As its foundation, a layered reference framework was described that categorizes technologies into groups of similar types with common characteristics and functionalities. All technology classifications, selection rules, and mapping tables are meant to guide both researchers and business users, who want to select technologies for their use cases at hand or who want to use SSF as basis for further research.

As the field of Big Data is currently advancing and evolving rapidly, it makes sense to simultaneously advance frameworks, methods, and tools for technology selection. To this end, SSF is a first step and can be extended and adapted as time passes by and new technologies emerge. Furthermore, it can be advanced with regard to additional needs.

SSF and the layered reference framework can be extended in both width and depth. One possibility is the addition of specific machine learning algorithms and new corresponding mapping tables. The layered reference framework could be completed by additional layers, such as a topmost data utilization layer that holds technologies and applications for end-user deliverables (e.g., by distinguishing between explanatory, exploratory and automation tools). Both contributions can also be used to complement and enhance the approaches they were motivated by. For instance, both layered reference model and, especially, SSF could be used to extend and refine the GOBIA method [41]. It could allow to have a comprehensive and coherent tool that guides companies fully from strategy to a customized tool mix in a customized analytics architecture. It could allow to revisit

previous choices and to validate or refresh them as the *ShopMart* example has demonstrated.

Moreover, compatibility maps and feature maps can be subject to further research, e.g., which granularity in describing features is most purposeful. In addition this, weights can be introduced to the process and these maps to allow for multi-objective based decisions. If these were given, mathematical methods for choosing an optimal technology mix for a given use case could be applied (e.g., by maximizing an objective or utility function based on this). As demonstrated in the application scenario case, only choosing the best tools in isolation and based on functionalities alone, may lead to new challenges, such as increasing complexity or costs.

SSF can be integrated into an automated tool (e.g., a web-application) that supports users with technology selection by using the deliverables of the thesis at hand. This could also be combined with weights to gain a (semi-)automated support system. Finally, the question remains how exactly the resulting technologies should be combined into a Big Data scalable infrastructure. While there are concepts like the Lambda Architecture, there is still no cookbook or commonly accepted best practice on how to exactly proceed. As this is needed to encourage especially small and mid-sized companies for a comprehensive coverage of Big Data utilization, it is certainly a promising field for future research.

References

[1] Apache Hadoop. Last accessed: 2016-04-02. URL: http://hadoop.apache.org/.

[2] Apache HBase. Last accessed: 2016-04-06. URL: https://hbase.apache.org/.

[3] Apache Mahout. Last accessed: 2016-04-02. URL: http://mahout.apache.org/.

[4] Apache Spark. Last accessed: 2016-04-03. URL: http://spark.apache.org/.

[5] Apache Storm. Last accessed: 2016-04-04. URL: http://storm.apache.org.

[6] Quora - Can I access SAP, Oracle and most of the ERP by SQL? Last accessed: 2016-09-14. URL: https://www.quora.com/Can-I-access-SAP-Oracle-and-most-of-the-ERP-by-SQL.

[7] Sqoop User Guide (v1.4.6). Last accessed: 2016-04-04. URL: https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html.

[8] The RDS Blog - What Is Advanced Analytics, Anyway. Last accessed: 2016-03-25. URL: http://www.recoverydecisionscience.com/what-is-advanced-analytics-anyway/.

[9] Twitter Firehouse vs. Twitter API: What's the difference and why should you care? Last accessed: 2016-03-10. URL: http://www.brightplanet.com/2013/06/twitter-firehose-vs-twitter-api-whats-the-difference-and-why-should-you-care/.

[10] What is Advanced Analytics? Last accessed: 2016-03-18. URL: http://www-01.ibm.com/software/data/infosphere/what-is-advanced-analytics/.

[11] KDnuggets: Data Mining Community's Top Resource, 2014. Last accessed: 2016-05-11. URL: http://www.kdnuggets.com.

[12] Big Data: Examples and Guidelines for the Enterprise Decision Maker, 2015. Last accessed: 2016-03-10. URL: http://s3.amazonaws.com/info-mongodb-com/10gen_Big_Data_White_Paper.pdf.

[13] Vijay Srinivas Agneeswaran. Why Look Beyond Hadoop Map-Reduce. Last accessed: 2016-04-18. URL: http://www.ftpress.com/articles/article.aspx?p=2215090&seqNum=2.

[14] Xavier Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013. URL: http://dl.acm.org/citation.cfm?id=2481250.

[15] HTTP Archive. HTTP Archive - Interesting Stats. Last accessed: 2016-09-16. URL: http://httparchive.org/interesting.php?a=All&l=Sep%201%202016&s=Top1000.

[16] Bahaaldine Azarmi. *Scalable Big Data Architecture: A practitioners guide to choosing relevant Big Data architecture*. Apress, 1st edition, 2015.

[17] Kapil Bakshi. Considerations for Big Data: Architecture and Approach. *IEEE Aerospace Conference Proceedings*, pages 1–7, 2012. arXiv:0402594v3, doi:10.1109/AERO.2012.6187357.

[18] Edmon Begoli and James Horey. Design Principles for Effective Knowledge Discovery from Big Data. *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, pages 215–218, 2012. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6337722, doi:10.1109/WICSA-ECSA.212.32.

[19] Albert Bifet. Mining Big Data in Real Time. *Informatica*, 37(1):15–20, 2013. doi:10.1.1.368.1416.

[20] EA Brewer. Towards robust distributed systems, 2000. URL: http://openstorage.gunadarma.ac.id/{~}mwiryana/Kuliah/Database/PODC-keynote.pdf.

[21] Jason Brownlee. A Tour of Machine Learning Algorithms, 2013. Last accessed: 2016-03-22. URL: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/.

[22] Joseph O. Chan. An Architecture for Big Data Analytics. 13(2):1–14, 2013.

[23] Samantha Chan. Streams Quick Start Guide - Streams Application Pattern. Last accessed: 2016-03-14. URL: https://developer.ibm.com/streamsdev/docs/streams-quick-start-guide/.

[24] Mayurika Chattergee. Let's Make Data Talk. *TechTalk*, 7(3):4–9, 2014.

[25] Surajit Chaudhuri, Umeshwar Dayal, and Venkatesh Ganti. Database Technology for Decision Support Systems. *IEEE Computer Society*, 34(12):48–55, 2001. doi:10.1109/2.970575.

[26] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. An Overview of Business Intelligence Technology. *Communications of the ACM*, 54(8):88–98, 2011. doi:10.1145/1978542.1978562.

[27] Dunren Che, Mejdl Safran, and Zhiyong Peng. From Big Data to Big Data Mining: Challenges, Issues, and Opportunities. pages 1–15, 2013.

[28] Fei Chen and Meichun Hsu. A Performance Comparison of Parallel DBMSs and MapReduce on Large-Scale Text Analytics. *Proceedings of the 16th International Conference on Extending Database Technology - EDBT '13*, 2013. URL: http://dl.acm.org/citation.cfm?doid=2452376.2452448, doi:10.1145/2452376.2452448.

[29] Hsinchun Chen, Roger H. L. Chiang, and Veda C. Storey. Business Intelligence and Analytics: From Big Data To Big Impact. *Mis Quarterly*, 36(4):1165–1188, 2012. doi:10.1145/2463676.2463712.

[30] Min Chen, Shiwen Mao, and Yunhao Liu. Big Data: A Survey. *Mobile Networks and Applications*, 19(2):171–209, 2014. doi:10.1007/s11036-013-0489-0.

[31] Shangyi Chen, Wei Li, Min Li, Xiaofei Zhang, and Yue Min. Latest Progress and Infrastructure Innovations of Big Data Technology. *Proceedings - 2014 International Conference on Cloud Computing and Big Data, CCBD 2014*, pages 8–15, 2014. doi:10.1109/CCBD.2014.25.

[32] Aakanksha Chopra and Suman Madan. Big Data: A Trouble or A Real Solution? 12(2):221–229, 2015.

[33] Umeshwar Dayal, Malu Castellanos, Alkis Simitsis, and Kevin Wilkinson. Data Integration Flows for Business Intelligence. *Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology - EDBT '09*, pages 1–11, 2009. URL: http://portal.acm.org/citation.cfm?doid=1516360.1516362, doi:10.1145/1516360.1516362.

[34] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107, jan 2008. URL: http://portal.acm.org/citation.cfm?doid=1327452.1327492, doi:10.1145/1327452.1327492.

[35] Yuri Demchenko, Cees De Laat, and Peter Membrey. Defining Architecture Components of the Big Data Ecosystem. *2014 International Conference on Collaboration Technologies and Systems, CTS 2014*, pages 104–112, 2014. doi:10.1109/CTS.2014.6867550.

[36] David Dietrich, Barry Heller, and Beibei Yang. *Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. John Wiley & Sons, Inc., 2015.

[37] Anhai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[38] Patrick TH. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The Many Faces of Publish / Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[39] Wei Fan and Albert Bifet. Mining Big Data: Current Status, and Forecast to the Future. *ACM SIGKDD Explorations Newsletter*, 14(2):1–5, 2013. doi:10.1145/2481244.2481246.

[40] Usama Fayyad, G Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI magazine*, 17(3):37–54, 1996. URL: http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1230.

[41] David Fekete and Gottfried Vossen. The GOBIA Method: Towards Goal-Oriented Business Intelligence Architectures. *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, FGDB*, 1458:409–418, 2015.

[42] Mike Ferguson. Architecting A Big Data Platform for Analytics. *Intelligent Business Strategies*, pages 1–36, 2012.

[43] Jonas Freiknecht. *Big Data in der Praxis*. Hanser Verlag, München, 2014.

[44] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144, 2014. URL: http://dx.doi.org/10.1016/j.ijinfomgt.2014.10.007, doi:10.1016/j.ijinfomgt.2014.10.007.

[45] G. R. Gangadharan and Sundaravalli N. Swami. Business Intelligence Systems: Design and Implementation Strategies. *26th International Conference on Information Technology Interfaces ITI*, 1:139–144, 2004. doi:10.1109/ITI.2004.241406.

[46] Seth Gilbert and Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News*, 33(2):51–59, 2002.

[47] Mike Gualtieri and Noel Yuhanna. Big Data Hadoop Distributions: Five Top Vendors Have Significantly Improved Their Offerings. Technical report, Forrester Research, 2016. URL: http://cloudera.com/content/dam/www/static/documents/analyst-reports/forrester-wave-big-data-hadoop-distributions.pdf.

[48] H2O. Importing Data - H20. Last accessed: 2016-09-16. URL: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-munging/importing-data.html.

[49] Han Hu, Yonggang Wen, Tat-Seng Chua, and Xuelong Li. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*, 2:652–687, 2014.

[50] Grant Ingersoll. Introducing Apache Mahout, 2009.

[51] Bill Jacobs. Using Hadoop with R: It Depends. Last accessed: 2016-04-21. URL: http://blog.revolutionanalytics.com/2015/06/using-hadoop-with-r-it-depends.html.

[52] A. Jovi, K. Brki, and N. Bogunovi. An overview of free software tools for general data mining. In *37th International Convention MIPRO 2014*, 2014.

[53] Ravi Kalakota. Big Data Analytics Use Cases, 2015. Last accessed: 2016-02-23. URL: http://practicalanalytics.co/2015/05/25/big-data-analytics-use-cases/.

[54] Chris Kimble and Giannis Milolidakis. Big Data and Business Intelligence: Debunking the Myths. *Global Business and Organizational Excellence*, 35(1):23–34, 2015. URL: citeulike-article-id:13798943http://dx.doi.org/10.1002/joe.21642, doi:doi:10.1002/joe.21642.

[55] John King and Roger Magoulas. 2013 Data Science Salary Survey: Tools, Trends, What Pays (and What Doesn't) for Data Professionals. *O'Reilly Strata*, pages 1–16, 2014.

[56] Krish Krishnan. *Data Warehousing in the Age of Big Data*. Morgan Kaufmann, 2013. URL: http://www.sciencedirect.com/science/article/pii/B9780124058910000052, doi:10.1016/B978-0-12-405891-0.00005-2.

[57] Rakesh Kumar, Neha Gupta, Shilpi Charu, and Sunil Kumar Jangir. Manage Big Data through NewSQL. *National Conference on Innovation in Wireless Communication and Networking Technology*, (August 2015), 2014. doi:10.13140/2.1.3965.3768.

[58] Ihor Kuz, Felix Rauch, Manuel M. T. Chakravarty, and Gernot Heiser. Distributed File Systems. 2015.

[59] Sara Landset, Taghi M. Khoshgoftaar, Aaron N. Richter, and Tawfiq Hasanin. A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *Journal of Big Data*, 2(1):24, 2015. URL: http://www.journalofbigdata.com/content/2/1/24, doi:10.1186/s40537-015-0032-1.

[60] Jens Lechtenbörger, Vanessa Jie Ling, and Gottfried Vossen. Hauptspeicherdatenbanken: Denkgeschwindigkeit auch für KMU? *Arbeitsberichte des Institut für Wirtschaftsinformatik, WWU Münster*, No. 136, 2015. URL: http://www.wi1.uni-muenster.de/pi/iai/publikationen/IMDB.pdf.

[61] Denis Lehmann. Technology Selection for BI Architectures in the Big Data Era. 2016.

[62] Jimmy Lin and Alek Kolcz. Large-Scale Machine Learning at Twitter. *ACM*, pages 793–804, 2012.

[63] Bernard Marr. *Big Data: Using Smart Big Data Analytics and Metrics to make better Decisions and improve Performance*. Wiley, 1st edition, 2015.

[64] Nathan Marz and James Warren. *Big Data - Principles and best practices of scalable realtime data systems*. 2015.

[65] Viktor Mayer-Schönberger and Kenneth Cukier. *Big Data: A Revolution That Will Transform How We Live, Work and Think*. John Murray, 2013.

[66] Ralf Mikut and Markus Reischl. Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):431–443, September 2011. URL: http://doi.wiley.com/10.1002/widm.24, doi:10.1002/widm.24.

[67] Cecilia Nembou. Contemporary Discussions in Mathematics - The Challenges of Big Data. In *Contemporary Discussions in Mathematics*, chapter 13, pages 155–169. 2013.

[68] Oracle. Extracting Data from SAP Applications. Last accessed: 2016-09-14. URL: http://docs.oracle.com/cd/E11882_01/owb.112/e10582/sap_integrate.htm#WBDOD30500.

[69] Oracle. Oracle SQL Connector for Hadoop Distributed File System. Last accessed: 2016-09-16. URL: https://docs.oracle.com/cd/E37231_01/doc.20/e36961/sqlch.htm#BDCUG126.

[70] Pentaho. Connecting with SAP Systems. Last accessed: 2016-09-14. URL: http://wiki.pentaho.com/display/EAI/Connecting+with+SAP+Systems.

[71] Gregory Piatetsky. R leads RapidMiner, Python catches up, Big Data tools grow, Spark ignites. Last accessed: 2016-04-20. URL: http://www.kdnuggets.com/2015/05/poll-r-rapidminer-python-big-data-spark.html.

[72] Postplanner. Is Your Facebook Page Better than Average? Here's the Data You Need to Find Out. Last accessed: 2016-09-16. URL: https://www.postplanner.com/facebook-data-on-fan-page-performance/.

[73] Eric Redmond and Jim R. Wilson. *Seven Databases in Seven Weeks*. The Pragmatic Programmers, 2nd edition, 2012.

[74] Aaron N. Richter, Taghi M. Khoshgoftaar, Sara Landset, and Tawfiq Hasanin. A Multi-Dimensional Comparison of Toolkits for Machine Learning with Big Data. *IEEE 16th International Conference on Information Reuse and Integration*, pages 1–8, 2015. doi:10.1109/IRI.2015.12.

[75] Pramod J. Sadalage and Martin Fowler. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison Wesley, 2012.

[76] Sahaj Saini. Transition from SMP to MPP, the why and the how. Last accessed: 2016-03-29. URL: https://blogs.technet.microsoft.com/dataplatforminsider/2014/07/30/transitioning-from-smp-to-mpp-the-why-and-the-how/.

[77] Olov Schelén, Ahmed Elragel, and Moutaz Haddara. A Roadmap for Big-Data Research and Education. Technical report, 2015.

[78] Paul Simon. *Too Big to Ignore*. John Wiley, 2013.

[79] Dilpreet Singh and Chandan K. Reddy. A survey on platforms for big data analytics. *Journal of Big Data*, 2(1):8, 2014. URL: http://www.journalofbigdata.com/content/2/1/8, doi:10.1186/s40537-014-0008-6.

[80] Statista. Anzahl der Artikel im Lebensmitteleinzelhandel in Deutschland nach Betriebsformen im Jahr 2013 — Statistik. Last accessed: 2016-09-16. URL: http://de.statista.com/statistik/daten/studie/309556/umfrage/artikel-im-lebensmitteleinzelhandel-in-deutschland-nach-betriebsformen/.

[81] Statista. Anzahl der Unternehmen im Großhandel in Deutschland in den Jahren 2002 bis 2014 — Statistik. Last accessed: 2016-09-16. URL: http://de.statista.com/statistik/daten/studie/274464/umfrage/unternehmen-im-grosshandel-in-deutschland/.

[82] Statista. Anzahl der Unternehmen im Großhandel in Deutschland nach Segmenten in den Jahren 2013 und 2014 — Statistik. Last accessed: 2016-09-16. URL: http://de.statista.com/statistik/daten/studie/201186/umfrage/grosshandel-zahl-der-unternehmen-in-deutschland-im-jahr-2009-nach-wirtschaftszweigen/.

[83] Zhaohao Sun, Huasheng Zou, and Kenneth Strang. Big Data Analytics as a Service for Business Intelligence. *IFIP International Federation for Information Processing*, 9373:200–211, 2015. URL: http://link.springer.com/10.1007/978-3-319-25013-7, doi:10.1007/978-3-319-25013-7.

[84] Bartlomiej Twardowski and Dominik Ryzko. Multi-agent Architecture for Real-Time Big Data Processing. *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pages 333–337, 2014. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6928203, doi:10.1109/WI-IAT.2014.185.

[85] Mark van Rijenam. The Future of Big Data: Prescriptive Analytics Changes the Game. Last accessed: 2016-04-09. URL: http://data-informed.com/future-big-data-prescriptive-analytics-changes-game/.

[86] Mark van Rijmenam. *Think Bigger: Developing a Successful Big Data Strategy for Your Business*. AMACOM, 2014.

[87] Gottfried Vossen. Big data as the new enabler in business and other intelligence. *Vietnam Journal of Computer Science*, 1:3–14, nov 2013. URL: http://link.springer.com/10.1007/s40595-013-0001-6, doi:10.1007/s40595-013-0001-6.

[88] Jason T. Widjaja. What is the difference between a data scientist and a business intelligence analyst? Last accessed: 2016-03-27. URL: https://www.quora.com/What-is-the-difference-between-a-data-scientist-and-a-business-intelligence-analyst.

[89] Svante Wold, Kim Esbensen, and Paul Geladi. Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[90] Brian Wylie, Daniel Dunlavy, Warren Davis Iv, and Jeff Baumes. Using NoSQL Databases for Streaming Network Analysis. *IEEE Symposium on Large Data Analysis and Visualization*, pages 121–124, 2012. URL: http://www.cse.ohio-state.edu/{~}raghu/teaching/CSE5544/Visweek2012/ldav/papers/wylie.pdf.

[91] Gerd Zeitler. Factors of Production. Last accessed: 2016-02-11. URL: https://gerdzeitler.wordpress.com/factors-of-production/.

Working Papers, ERCIS

Nr. 1 Becker, J.; Backhaus, K.; Grob, H. L.; Hoeren, T.; Klein, S.; Kuchen, H.; Müller-Funk, U.; Thonemann, U. W.; Vossen, G.; European Research Center for Information Systems (ERCIS). Gründungsveranstaltung Münster, 12. Oktober 2004.

Nr. 2 Teubner, R. A.: The IT21 Checkup for IT Fitness: Experiences and Empirical Evidence from 4 Years of Evaluation Practice. 2005.

Nr. 3 Teubner, R. A.; Mocker, M.: Strategic Information Planning – Insights from an Action Research Project in the Financial Services Industry. 2005.

Nr. 4 Gottfried Vossen, Stephan Hagemann: From Version 1.0 to Version 2.0: A Brief History Of the Web. 2007.

Nr. 5 Hagemann, S.; Letz, C.; Vossen, G.: Web Service Discovery – Reality Check 2.0. 2007.

Nr. 6 Teubner, R.; Mocker, M.: A Literature Overview on Strategic Information Management. 2007.

Nr. 7 Ciechanowicz, P.; Poldner, M.; Kuchen, H.: The Mnster Skeleton Library Muesli – A Comprehensive Overview. 2009.

Nr. 8 Hagemann, S.; Vossen, G.: Web-Wide Application Customization: The Case of Mashups. 2010.

Nr. 9 Majchrzak, T.; Jakubiec, A.; Lablans, M.; kert, F.: Evaluating Mobile Ambient Assisted Living Devices and Web 2.0 Technology for a Better Social Integration. 2010.

Nr. 10 Majchrzak, T.; Kuchen, H: Muggl: The Muenster Generator of Glass-box Test Cases. 2011.

Nr. 11 Becker, J.; Beverungen, D.; Delfmann, P.; Rckers, M.: Network e-Volution. 2011.

Nr. 12 Teubner, A.; Pellengahr, A.; Mocker, M.: The IT Strategy Divide: Professional Practice and Academic Debate. 2012.

Nr. 13 Niehaves, B.; Kffer, S.; Ortbach, K.; Katschewitz, S.: Towards an IT consumerization theory: A theory and practice review. 2012

Nr. 14 Stahl, F., Schomm, F., & Vossen, G.: Marketplaces for Data: An initial Survey. 2012.

Nr. 15 Becker, J.; Matzner, M. (Eds.).: Promoting Business Process Management Excellence in Russia. 2012.

Nr. 16 Teubner, R.; Pellengahr, A.: State of and Perspectives for IS Strategy Research. 2013.

Nr. 17 Teubner, A.; Klein, S.: The Mnster Information Management Framework (MIMF). 2014.

Nr. 18 Stahl, F.; Schomm, F.; Vossen, G.: The Data Marketplace Survey Revisited. 2014.

Nr. 19 Dillon, S.; Vossen, G.: SaaS Cloud Computing in Small and Medium Enterprises: A Comparison between Germany and New Zealand. 2015.

Nr. 20 Stahl, F.; Godde, A.; Hagedorn, B.; Kpcke, B.; Rehberger, M.; Vossen, G.: Implementing the WiPo Architecture. 2014.

Nr. 21 Pflanzl, N.; Bergener, K.; Stein, A.; Vossen, G.: Information Systems Freshmen Teaching: Case Experience from Day One (Pre-Version of the publication in the International Journal of Information and Operations Management Education (IJIOME)). 2014.

Nr. 22 Teubner, A.; Diederich, S.: Managerial Challenges in IT Programmes: Evidence from Multiple Case Study Research. 2015.

Nr. 23 Vomfell, L.; Stahl, F.; Schomm, F.; Vossen, G.: A Classification Framework for Data Marketplaces. 2015.

Nr. 24 Stahl, F.; Schomm, F.; Vomfell, L.; Vossen, G.: Marketplaces for Digital Data: Quo Vadis?. 2015.

Nr. 25 Caballero, R; von Hof, V.; Montenegro, M.; Kuchen, H.: A Program Transformation for Converting Java Assertions into Control-flow Statements. 2015.

Nr. 26 Foegen, K.; von Hof, V.; Kuchen, H.: Attributed Grammars for Detecting Spring Configuration Errors. 2015.

Nr. 27 Lehmann, D.; Fekete, D.; Vossen, G.: Technology Selection for Big Data and Analytical Applications. 2016.