A Reference Architecture for Big Data Solutions

Introducing a model to perform predictive analytics using big data technology

Bas Geerdink
Faculty of Science and Engineering
Utrecht University of Applied Sciences
Utrecht, the Netherlands
bas@gdink.com

Abstract—With big data technology and predictive analytics techniques, organizations can now register, combine, process and analyze data to answer questions that were unsolvable a few years ago. This paper introduces a solution reference that gives guidance to organizations that want to innovate using big data technology and predictive analytics techniques for improving their performance. The reference architecture is the result of an iteration of Hevner’s framework for designing information systems artifacts.

I. INTRODUCTION

Traditional business intelligence (BI) has been around for a long time and there is still a big market for IT systems with large data warehouses and reporting solutions. However, traditional BI cannot cope with the demands of organizations to store and use ever more data, process data faster, and make better predictions. ‘New’ data sources such as social media networks, on-line news, open data from the “internet of things”, log files, email, video, sound, images and file shares offer huge opportunities for data analysis, which is simple too complex and demanding for traditional BI [1]. For these kind of requests, big data technology can be used.

The combined force of big data technology, predictive analytics, and open data offers a wealth of possibilities for organizations that want to make predictions about the future. There are plenty of free and open-source big data products and frameworks. In addition, several commercial vendors offer big data products or as-a-service platforms. Organizations will have to choose components for their big data solutions, and find ways to approach the big data projects. A big data solution reference architecture will facilitate and guide architects of these organizations.

A. Big Data

The amount of data in organizations is growing rapidly. Data production will be 44 times greater in 2020 than it was in 2009 and there will be a 650% growth in enterprise data in the next five years [2]. In the near future, many machines and other devices will get an IP address and connect to the web in the ‘internet of things’, providing even more data to be accessed [3]. Big data is the term that is used for the field of analysis of large datasets. The origin of the term ‘big data’ goes back as far as the 1990s [4]. The term became widespread with an article in The Economist in 2010 [5]. However, big data is not just about size; after all, what is ‘big’ is relative and changes across the years. Other aspects of big data are the speed of data (e.g. streaming media) and the different types and formats of the data (e.g. non-relational, semi-structured, or unstructured content). Therefore, the definition of big data according to Gartner is “high volume, velocity and/or variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision-making, and process automation” (Gartner, 2012). Doug Laney introduced this “3V” definition already in 2001 [6]. IBS and IBM provide another definition: “Big data is a term associated with new types of workloads that cannot be easily supported in traditional environments”, which indicates the switch from traditional BI to big data and the relative size of the term [1].

Big data technology is for analyzing very large collections of data sets on parallel-distributed commodity hardware. A breakthrough came in 2004, when Dean and Ghemawat introduced the MapReduce programming paradigm [7]. The world of big data technology has concentrated around a number of free and open-source software (FOSS) components. A very important framework is the Apache Hadoop ecosystem, which offers an implementation of the MapReduce algorithm and HDFS, a distributed file system. The big data technology group includes more than MapReduce, for example the underlying file system (e.g. GFS [8]), and a relatively new type of lightweight, non-relational database that are often part of a big data solution: NoSQL databases. The purpose of these databases is to store unstructured and semi-structured data such as files, documents, email, and social media. Examples of these databases are BigTable [9] and HBase.

After MapReduce, several new technologies appeared that create even more possibilities for organizations. An important technique is the streaming of high-speed data. Platforms such as S4 [10], Twitter Storm, and Akka are capable of processing enormous amounts of data in (near) real-time, by making use of clever algorithms and the architecture principles that were already used for MapReduce: massive parallelism on commodity hardware.

There are a number of commercial products available that provide enterprise solutions based on Hadoop, for example Cloudera, Karmasphere, MapR, HortonWorks, and IBM InfoSphere BigInsights [11]. Other examples of commercial big data products are Amazon Web Services Elastic MapReduce, Infochimps Enterprise Cloud, EMC GreenPlum, Microsoft Windows Azure, and Google BigQuery [12]. Some system integrators such as Cap Gemini, Accenture, CSC, HP, and Dell offer big data products and services to their clients.

B. Predictive Analytics

Predictive analytics is a collective term for techniques with the aim of predicting the future based on static or historical data. By making use of clever algorithms, and statistical models, people working with predictive analytics techniques try to find trends in the data and then project these trends to say meaningful things about the upcoming events. The results of predictive analytics always contain uncertainties. Techniques from the fields of statistics and machine learning can be used or combined; a predictive analysis engine or forecasting program can contain regression models and/or neural networks, for example in
time series forecasting [13]. Examples of concrete prediction methods are autoregressive integrated moving average (ARIMA) and machine learning. The concept of a model is crucial in predictive analytics; the model determines the prediction based on the data. This model is constantly adjusted, tuned, optimized, and trained depending on the environment and altering insights of the users.

There are several free and open-source tools that can be used for predictive analytics, including R, KNIME, Orange, and Weka. In addition, enterprise software vendors such as Angoss, Alteryx, KXEN, Salford Systems, StatSoft, SAP, SAS, IBM, Tibco, and Oracle provide solutions that help with analyzing data and predicting the future.

A special case of predictive analytics and data mining is data exploration and discovery. Other names for this research field are knowledge extraction and knowledge discovery. With big data, it is possible to analyze and combine very many data from very many different sources. Specialized software can identify relationships or clusters in those combinations of data sets, which are invisible to the human eye [14]. The main difference between data exploration and discovery with other areas of predictive analytics is that the order of data sources does not matter. In ‘normal’ predictive analytics, there usually is a time sequence or transaction sequence, where in data exploration and discovery the data is just ‘there’, in a random or unimportant order. In addition, data exploration and discovery calls for a data-driven approach, whereas business questions or use cases drive other methods of predictive analytics. In data exploration and discovery, there is no a priori hypothesis for the results of the analysis.

There are several techniques and methods available from the fields of mathematics and artificial intelligence that have a relation with data exploration and discovery, for example association rule learning, spatial indices, affinity analysis, pattern recognition, and certain machine learning algorithms [15]. Some commercial vendors (e.g. SAS and IBM) offer solutions specifically for data exploration and discovery [16]. K-means, decision trees, deep learning (multi-layered neural networks) and random forests (weighted multiple decision trees based on randomly selected sets of variables) are the most successful prediction algorithms.

C. Reference Architectures

A reference architecture is an abstraction of ‘real’ architectures. There are various forms of reference architectures: enterprise reference architectures, solution reference architectures, information systems reference architectures, etc. A solution reference architecture is a skeleton for a solution, where the elements are templates or outlines for components. According to Muller, architects can use a reference architecture as guidance to create a concrete architecture for their organization, business context and technology [17]. A solution reference architecture contains hardware and components, patterns and best practices, principles, and presents itself in a visually appealing way [18].

Angelov et al. defined a useful framework for the analysis and design of software reference architectures [19]. The framework contains classifications of reference architectures, for different context of use of a reference architecture.

This paper describes the Big Data Solution Reference Architecture, which is a technology-independent solution reference architecture. The model will contain conceptual components, with a list of options as possible implementations. These options are free and open-source projects, as well as products and solutions of commercial big data vendors. However, the concrete implementations of the technology components is not important as the reference architecture will be high-level and abstract. The technology itself does not matter, the business value it brings does. The model is primarily aimed at giving guidance big data architects when creating big data solutions.

II. RESEARCH METHOD

The Big Data Solution Reference Architecture was created using Hevner’s Information Systems Research Framework [20]. It is perfectly suited to structure the design of the Big Data Solution Reference Architecture, since that is an information system artifact based on business needs and existing literature (the knowledge base).

The research method contains two other models: Angelov’s framework for designing reference architectures [19], and Kazman’s Software Architecture Analysis Method (SAAM) [21]. The latter two are interpretations of the elements Develop/Build and Justify/Evaluate of Hevner’s model.

III. LITERATURE REVIEW

As a first step, the existing literature was searched for big data architectures. Both scientific and non-scientific sources were used to get an overview of work that has been done on architectures considering big data, open data, and predictive analytics.

An evaluation of the literature identified the usable elements for the Big Data Solution Reference Architecture. In total, 27 literature sources were investigated. Most articles and websites mention several aspects of big data architecture. It is not just about software; a solution architect should also concern himself with business processes, infrastructure, patterns, principles, and best practices. Big data architectures in literature points contain the following elements: hardware and software components, architecture principles, and best practices.

Five important observations were made from the literature review. First, the literature clearly defines the core of a big data architecture. Nearly all sources contain the following components: a parallel batch-processing engine (e.g. Hadoop MapReduce), a distributed file system (e.g. HDFS), and a NoSQL database (e.g. HBase).

Second, there is obviously more than MapReduce: data sources, data mining processes, coordination and configuration engines, databases, monitoring, etc. In addition, traditional BI systems and software components still seem to have a place in a big data architecture. All these components play a role in the literature, some more than others. Several other components are typical for a big data architecture, simply because they surface often in the literature. The following components have a place in the majority of the literature that describes a big data architecture:

- A querying engine;
- A predictive analytics engine;
- A statistical analysis or machine learning engine;
- A data importing / collecting / ETL engine;
- A real-time / stream / complex event-processing engine.

Third, several architecture principles exist in the articles and websites on big data. Loose coupling, cloud computing, and scalability are popular principles in literature. There are several principles about whom the literature sources disagree. For example, IBM, SAS, and Kimball very strongly believe in the principle of “Close-to-source data processing”, which implicates that data should be analyzed as early as possible to reduce storage costs and processing time. On the contrary, MicroStrategy believes in retrieving and storing as many data as possible and performing analytics at a relatively late stage.

Fourth, for best practices there is only one item that truly stands out: the “data pipeline approach”. This best practice indicates that a big data architecture is like a pipeline through which data flows. Several literature sources point to another best practice that is in contrast with the pipeline approach, namely the “data exploration and discovery” method. This best practice is actually a type of big data analytics where the data is not retrieved or imported, but remains at its source and is approachable directly for analytical purposes.

Fifth, there seems to be more consensus about the hardware and software component than about the principles and best practices. This indicates that people agree about big data technology, but have yet to reach a common understanding about the approaches and patterns in big data architecture.
The components, architecture principles, and best practices found in literature were put forward in the expert interviews to confirm their place in the final model. In this way, the literature review in Hevner’s framework served to create a provisional model of the final Big Data Solution Reference Architecture.

IV. DEVELOPMENT OF THE REFERENCE ARCHITECTURE

Angelov’s framework guided the development of the reference architecture. Angelov et al. created a model for creation and classification of reference architectures, wherein answers to questions are guiding the type of a reference architecture. The model consists of dimensions, split up in sub-dimensions. Each sub-dimension has a code (e.g. D2) and is linked to one question [19]. There are three dimensions (G/Goal, C/Context, and D/Design) and eight sub-dimensions and questions (G1/Why, C1/Where, C2/Who, C3/When, D1/What, D2/Detail, D3/Concreteness, and D4/How). Angelov et al. describe a process to answer these questions, and thus determine the characteristics of a reference architecture before actually designing it. The following paragraphs describe the steps in this process, and the obtained results.

A. Define “Why”, “Where” and “When”

Angelov’s model requires a clear statement on the following aspects of the reference, before commencing the design of the model:

- The goal of the reference architecture (“Why”);
- The application context of the reference architecture (“Where”);
- The timing aspects of the reference architecture (“When”).

The goal of the Big Data Solution Reference Architecture is to guide architects who want to create a solution architecture that is capable of working with big data. Angelov et al. defined two possible values for the Goal sub-dimension G1: standardization and facilitation. The Big Data Solution Reference Architecture clearly aims at providing guidelines and inspiration for the design of solutions. The main ambition is not to standardize concrete architectures or to improve interoperability of existing components/systems. Thus, the goal of the Big Data Solution Reference Architecture is facilitation.

The context of the reference architecture is organizations who want to predict the future using large datasets of enterprise data combined with open data sources. The reference architecture is industry-independent but targets organizations of considerable size that have the resources (time, money, and people) available to perform a big data project under architectural guidance. Typically, an organization using the reference architecture has at least 100 employees and an IT department of at least 10 employees. The intended recipient of the Big Data Solution Reference Architecture is a lead architect who is able to make decisions about the concrete solution architecture, architecture principles, and resources. Since the Big Data Solution Reference Architecture must be industry-independent, the Context sub-dimension C1 gets the value “multiple organizations”.

The reference architecture is time-independent. However, it is likely that the abstract hardware and software components that are included in the reference architecture will be outdated in a few years’ time. Therefore, the owner must maintain and update the reference architecture on a regular basis. The C3 sub-dimension has two possible values: preliminary and classical. A typical preliminary reference architecture is designed when no concrete components or other parts of the reference architecture exist in practice. This is not the case for the Big Data Solution Reference Architecture; there are several known big data solutions working in practice [22] [21] [23]. Rather, the Big Data Solution Reference Architecture takes the practical experience of a group of experts and uses that to give a “best practice reference architecture”. Thus, the Context sub-dimension C3 gets the value “classical”.

B. Classify the reference architecture

Next, the architecture type was classified using these “Why”, “Where” and “When” answers. This gives the reference architecture a place amongst other reference architectures, in one of the five types defined by Angelov et al.

According to the “why”, “where”, and “when” statements above, the reference architecture is of “type 3”. Reference architectures of type 3 are facilitating, classical, designed for multiple organizations and created by an independent organization.

C. Invite stakeholders (“Who”)

To gather more data for the creation of a reference architecture, the research conducted a number of interviews with experts in big data, open data, and/or predictive analytics. The interview data formed the basis for the reference architecture. Qualitative data analysis techniques facilitated in acquiring the building blocks of the reference architecture.

The Context sub-dimension C2 contains the list of stakeholders that were involved in the design of the Big Data Solution Reference Architecture. There are two groups involved: requirements Designers (D) and providers (R). Following these guidelines, a group of five stakeholders was identified. The stakeholders were experts in the fields of big data and predictive analytics from various organizations and disciplines. The interviews were structured; each interview followed a fixed schedule of questions while leaving room for side steps and digression [24]. In addition, the researcher used elements of the provisional model and the acquired insights from the literature during the interviews.

D. Define “What” and “How”

The activity in this step was to define the following characteristics of the final reference architecture, as part of the Design dimension:

- The elements of the model;
- The level of detail of the model and its elements;
- The concreteness of the described elements;
- The representation of the elements, e.g. visually or text.

To generate the model (the reference architecture) from the data, the researcher conducted several iterations grounded theory. Grounded theory is “the theory that was derived from data, systematically gathered and analyzed through the research process. In this method, data collection, analysis, and eventual theory stand in close relationship to one another.” [25] A central process in grounded theory is coding, a practice where the researcher processes the transcripts of interviews (or other sources such as diagrams, field notes, etc.) by labelling text and categorizing the labels (codes). When working in iterations, with each iteration the code base diminishes in size as the understanding of the researcher grows and codes group or combine.

1) D1: What is described?

According to Angelov’s model, type 3 reference architectures should consist of components, interfaces, and policies/guidelines. Adhering to this model, the codes were categorized in the following categories:

- Components & interfaces;
- Architectural patterns;
- Architecture principles;
- Architectural best practices.

These categories match the provisional model, with the exception of the architectural patterns. After reviewing the literature, interviewing the stakeholders and analyzing the transcripts with grounded theory and using the provisional model, the coded transcripts pointed out that this category is necessary for the reference architecture.

Components are business processes, software applications or frameworks, and hardware. Interfaces are the functional relationships, technical connections, data flows, compositions, and aggregations between these components.

Architectural patterns are proven solutions to recurring enterprise architecture problems. They offer architects abstracted methods and
techniques to work with, which have been applied in similar problems by other experts in the field [26]. In this regards they are similar to application architecture patterns in software engineering, which have been more widely used in practices [27], [28]. Garlan and Shaw introduced some examples of architectural patterns and called them “Common Architectural Styles”. Examples of their patterns are Pipes and Filters (also known as the Data Flow pattern), Data Abstraction and Object-Oriented Organization, and Layered Systems [29].

Architecture principles are “fundamental approaches, beliefs, or means for achieving a goal” that give guidance to architects [30]. Architecture principles are very important parts of any solution or enterprise architecture. Principles can be normative or scientific. A normative principle is “a declarative statement that normatively prescribes a property of something”, whereas a scientific principle is “a law or fact of nature underlying the working of an artifact” [31]. The Big Data Solution Reference Architecture will contain normative principles that give guidance to architects who are designing big data solutions. In a sense, the normative architecture principles work as constraints to the organization; they give a certain amount of freedom to work with, but specify absolute boundaries for the solution.

Finally, architectural best practices describe other aspects that are important when creating a big data architecture. These best practices give guidance in the processes in which architects surely are involved: management, planning, estimating, budgeting, cooperation with internal and external suppliers, and so forth.

2) D2: How detailed is it described?

Type 3 reference architectures prescribe semi-detailed components and policies/guidelines, and aggregated or semi-detailed interfaces. That suits well with the Big Data Solution Reference Architecture since it is supposed to be an industry-independent, generic reference architecture. Angelov et al. suggest to measure the level of detail by counting the number of elements (e.g. components, guidelines) or the number of aggregation levels (e.g. layers in an enterprise architecture). The Big Data Solution Reference Architecture, with semi-detailed components, interfaces and policies/guidelines, should not contain numerous elements at more than two aggregation levels.

3) D3: How concrete is it described?

Reference architectures of type 3 should have abstract or semi-concrete elements. This implies that the Big Data Solution Reference Architecture will describe its components, interfaces, and policies/guidelines in a non-specific, abstract way. The components that surfaced from the literature and interviews will become abstract concepts rather than concrete products or frameworks in the reference architecture. This will keep the reference architecture high-level, and keep the reference architecture simple because the number of components will be small. The abstraction will be done in the iterative coding cycles of the transcribed interview data. For example, if an expert mentions ‘MongoDB’ or ‘Cassandra’, both are coded as ‘NoSQL database’. In the reference architecture, the abstract concept of a ‘NoSQL database’ is then added to the list of components.

4) D4: How is it represented?

According to Angelov’s model, type 3 reference architectures have semi-formal element specifications. The semi-formal representation requires well-defined notations of the elements of the reference architecture. The different parts of the Big Data Solution Reference Architecture are presented in different ways. The hardware and software components are presented visually, in a diagram on one page. Additional text will explain the components in detail, and their interfaces. The choice for the visual representation, aided by text, was made because that is the standard in existing literature and because this representation will give an overview of the reference architecture in one notion. The other elements of the reference architecture, e.g. the architecture principles and best practices, will be represented as text, tables, or lists, since no good visual representation is possible. The Big Data Solution Reference Architecture uses the following notations:

- ArchiMate 2.0 [32] for the components and interfaces;
- The Pattern Language of Avgeriou and Zdun [33] for the architectural patterns;
- TOGAF 9.1 [34] for the architecture principles;
- No specific format for best practices.

E. Summary

Angelov’s classification model for reference architectures contains the very dimensions, questions and dimensions that determine the type of a reference architecture. The following table gives an overview of the characteristics for the Big Data Solution Reference Architecture.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Sub-Dimension</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>G1</td>
<td>Why</td>
<td>Facilitation</td>
</tr>
<tr>
<td>Context</td>
<td>C1</td>
<td>Where</td>
<td>Multiple organizations</td>
</tr>
<tr>
<td>Context</td>
<td>C2</td>
<td>Who</td>
<td>Independent organization (D), Software organizations (R), User organizations (R)</td>
</tr>
<tr>
<td>Context</td>
<td>C3</td>
<td>When</td>
<td>Classical</td>
</tr>
<tr>
<td>Design</td>
<td>D1</td>
<td>What</td>
<td>Components, interfaces, policies/guidelines</td>
</tr>
<tr>
<td>Design</td>
<td>D2</td>
<td>Detail</td>
<td>Semi-detailed components and policies/guidelines, Aggregated or semi-detailed interfaces</td>
</tr>
<tr>
<td>Design</td>
<td>D3</td>
<td>Concreteness</td>
<td>Abstract or semi-concrete elements</td>
</tr>
<tr>
<td>Design</td>
<td>D4</td>
<td>How</td>
<td>Semi-formal element specifications</td>
</tr>
</tbody>
</table>

V. THE BIG DATA SOLUTION REFERENCE ARCHITECTURE

This paragraph contains a summary of the Big Data Solution Reference Architecture that was created after investigating the literature, interviewing the experts, using grounded theory to perform quantitative data analysis, and determining the representations of the various elements. The reference architecture is a guideline, not a prescription. Each element in the model is optional in the solution architecture that is ultimately created. The reference architecture consists of categories, according to the elements that were defined in the “what” question (dimension D1) of Angelov’s model. This paper only highlights the elements of the reference architecture; the full text, which contains an extensive explanation of the usage of the model, is available online at http://gdkm.com/Data/Sites/1/big-data-solution-reference-architecture.pdf.

A. Components & Interfaces

This category contains all components (business, software, and hardware) that are part of the reference architecture, as well as the interfaces between them.

Using the knowledge gained in the literature and interviews, a visual representation in ArchiMate was made of the components that make up the big data reference architecture. Figure 1 on page 2 displays the diagram. The components of the reference architecture are divided into the three common ArchiMate layers Business, Application, and Technology. The model focusses on “what” rather than “how”. The conceptual elements of the reference architecture are all structures. The structural character of the model was chosen on purpose, to give architects a clear guidance while still presenting abstract components. The focus on “what” allows for an easy translation of the abstract
reference architecture to real physical components in the big data solutions that are implementations of the model. The size of the component blocks in the Application Layer give an indication of the “importance”, the complexity, and the required amount of computer resources (memory, disk space, number of frameworks, etc.) of the component. For example, the Processing Engine is displayed larger than the Management Engine since it has more tasks, demands more resources and is considered one of the core components of the big data solution.

The visual representation of the components and interfaces is semi-detailed and contains semi-concrete elements, following the guidelines of Angelov et al. for a “type 3” reference. The level of detail can be measured from the number of layers and the number of components. In the Big Data Solution Reference Architecture, both are reasonably low: three layers and relatively small number of components. The level of concreteness is apparent from the abstractness of the components; no concrete components are listed but rather the templates or concepts. For example, instead of including “HDFS”, the concept of a Distributed File System is used. However, the supporting text of the reference architecture (available on request) contains several options and examples for the components, to give architects a feel for the possibilities.

A concept that sprouts from the literature and the interviews is to see big data analytics as a data pipeline. In contrast to other solution architectures where data “moves around” between services, databases, applications, objects, and so forth, big data is about getting a large dataset “streaming” through a set of tools and frameworks to get insight and derive meaning, and ultimately take action. In our research, the sources are enterprise data and open data, and the insight is a statement of predictive analytics. Thus, one of the first choices was to represent the reference architecture as a data pipeline. In the diagram, the data "flows" at the Application layer and the Technology layer; the application components communicate and pass data to each other figuratively, while the actual data flow happens in the infrastructure. As mentioned above, the reference architecture aims at both batch processing and (near) real-time data processing solutions.

B. Architectural Patterns

Architectural patterns “offer well-established solutions to architectural problems help to document the architectural design decisions, facilitate communication between stakeholders through a common vocabulary, and describe the quality attributes of a software system as forces” [33]. These architectural patterns are important to the working of the reference architecture; they define the way the reference architecture is constructed and the way the architects should work with the reference architecture. The Big Data Solution Reference Architecture contains two patterns: “Pipes and Filters” and “Layers”.

PATTERN 1: PIPES AND FILTERS

Following the data analysis, the first architectural pattern included in the reference architecture is the “Pipes and Filters” pattern. This pattern deals with how streams of data are successively processed or transformed by components.

The definition of the Pipes and Filters pattern is as follows: “Consider […] the case where a complex task can be sub-divided into a number of smaller tasks, which can be defined as a series of independent computations. Additionally the application processes streams of data, i.e. it transforms input data streams into output data streams. This functionality should not be realized by one monolithic component because this component would be overly complex, and it would hinder modifiability and reusability. Furthermore, different clients require different variations of the computations, for instance, the results should be presented in different ways or different kinds of input data should be provided. To reach this goal, it must be possible to flexibly compose individual sub-tasks according to the client’s demands. In a PIPES AND FILTERS architecture a complex task is divided into several sequential subtasks. Each of these sub-tasks is implemented by a separate, independent component, a filter, which handles only this task. Filters have a number of inputs and a number of outputs and they are connected flexibly using pipes but they are never aware of the identity of adjacent filters. Each pipe realizes a stream of data between two components. Each filter consumes and delivers data incrementally, which maximizes the throughput of each individual filter, since filters can potentially work in parallel. Pipes act as data buffers between adjacent filters. The use of PIPES AND FILTERS is advisable when little contextual information needs to be maintained between the filter components and filters retain no state between invocations. PIPES AND FILTERS can be flexibly composed. However, sharing data between these components is expensive or inflexible. There are performance overheads for transferring data in pipes and data transformations, and error handling is rather difficult.” [33]

Using the Pipes and Filters pattern implies that the architecture of a big data solution must be built around a series of tasks. In the Big Data Solution Reference Architecture, all layers contain an example of the division into tasks. The best example is the Application Layer, which consists of the components Importing Engine, Processing Engine, Analytics Engine, and Visualization Engine. Each component is independent and modular, and can be thought of as a filter. Data flows
or streams in a pipe between these components, represented by the “Flow” ArchiMate relation.

The Pipes and Filters pattern matches best with the common form of predictive analytics, where data is presented, imported and processed in a sequence. In case of data exploration and discovery (or knowledge discovery), there is less of a data flow. In that case, the Importing Engine is probably not used, or only has the function of a throughput engine that simply transfers the data without doing anything with it. The Processing Engine will contain data exploration and/or stream processing engines, and the Processing Engine takes the form of a high-performance machine learning, mathematical analytics, or pattern recognition framework.

**PATTERN 2: LAYERS**

Another architectural pattern that can be identified from the interviews is “Layers”, as part of the “Layered View”. Codes “Cloud” and “Application layer is leading” both indicate a layering in the architecture. The Layers pattern is closely connected to the architecture principle “Loose coupling”.

The definition of the Layers pattern is: “Consider a system in which high-level components depend on low-level components to perform their functionality, which further depend on even lower-level components and so on. Decoupling the components in a vertical manner is crucial in order to support modifiability, portability, and reusability. On the other hand, components also require some horizontal structuring that is orthogonal to their vertical subdivision. To achieve these goals, the system is structured into layers so that each layer provides a set of services to the layer above and uses the services of the layer below. Within each layer, all constituent components work at the same level of abstraction and can interact through connectors. Between two adjacent layers, a clearly defined interface is provided. In the pure form of the pattern, layers should not be by-passed: higher-level layers access lower-level layers only through the layer beneath.”

[33]

The Layers pattern is implemented in the Big Data Solution Reference Architecture by representing the components of the architecture in the layers Business Layer, Application Layer, and Technology Layer. This division follows TOGAF and ArchiMate and is a standard partition of solution architectures.

**C. Architecture Principles**

The Big Data Solution Reference Architecture contains two principles: “Loose coupling” and “Interoperability”. The principles were derived from analysis of the literature and grounded theory using the expert interviews. The two principles are presented in the following sub paragraphs, with the notation prescribed by TOGAF 9.1.

**PRINCIPLE 1: LOOSE COUPLING**

<table>
<thead>
<tr>
<th>Name</th>
<th>Loose coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement</td>
<td>Create a solution with loosely coupled building blocks, e.g. message-exchanging software components instead of integrated frameworks</td>
</tr>
<tr>
<td>Rationale</td>
<td>By loosely coupling the components, the modularity, reusability and modifiability of the solution increases. Big data is a fast-moving field, where components are developed, improved, and retired frequently. To be able to cope with the changing requirements and components, the big data solution has to be flexible. If a building block has to be replaced, upgraded, removed, or added, other building blocks should be impacted as little as possible. By loosely coupling the components, these kind of actions are relatively easy.</td>
</tr>
<tr>
<td>Implications</td>
<td>Components of the solution such as software packages, frameworks, databases should be selected based on their ability to be decoupled from the solution. That means components should have clear service contracts, data interfaces, and/or APIs that preferably rely on messaging.</td>
</tr>
</tbody>
</table>

**PRINCIPLE 2: INTEROPERABILITY**

<table>
<thead>
<tr>
<th>Name</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement</td>
<td>Software and hardware should conform to defined standards that promote interoperability for data, applications, and technology.</td>
</tr>
<tr>
<td>Rationale</td>
<td>Standards help ensure consistency, thus improving the ability to manage systems and improve user satisfaction, and protect existing IT investments, thus maximizing return on investment and reducing costs. Standards for interoperability additionally help ensure support from multiple vendors for their products, and facilitate supply chain integration.</td>
</tr>
<tr>
<td>Implications</td>
<td>Interoperability standards and industry standards will be followed unless there is a compelling business reason to implement a non-standard solution. A process for setting standards, reviewing and revising them periodically, and granting exceptions must be established. The existing IT platforms must be identified and documented.</td>
</tr>
</tbody>
</table>

**D. Architectural Best Practices**

The reference architecture contains two best practices “Use free and open-source software” and “Agile development”.

**BEST PRACTICE 1: USE FREE AND OPEN-SOURCE SOFTWARE**

Most architectures that are addressed in the interviews and literature are based on free and open-source software (FOSS) components. When asked specifically, the interviewed architects agree to the notion that free and open-source software forms the core of big data. There are commercial (proprietary) products and services available that are based on the FOSS stack; for example, several large IT companies are trying to solutionify Hadoop. However, as pointed out by the interviewed stakeholders, the free and open-source community is leading in innovation when it comes to big data software components. This notion is even supported by commercial organizations such as IBM [35]. The FOSS products are simply better than the commercial ones in terms of usability, modifiability, performance, reliability, and costs. Vendor lock-in is avoided, and the architecture principles “Loose coupling” and “Interoperability” can be applied more easily with FOSS.

Free and open-source (FOSS) in this reference model is software that is both free and open-source, classified according to the definition of the Free Software Foundation. This definition states that software can used, copied, changed, and distributed by anyone [36]. The FOSS definition is tighter than the Open Source definition, which only states that the software should be free of charge and the source code should be publicly available and modifiable [37]. The Open Source definition is only applicable to practical applications, not to the social and political aspects [38]. In contrast, FOSS is about the liberty of software, not about the price. The unfortunate event is that the word “free” in English speech has two meanings, unlike for example French where there is libre and gratuit. This point is made clear in Richard Stallman’s famous article in which he states: “think of “free speech,” not “free beer.”” [39]. Examples of FOSS licenses are the GNU (Lesser) Public License, the Apache Licenses, the Microsoft Public License, the Mozilla Public Licenses, and the Intel Open Source License [40].

Pretending FOSS components over proprietary software can have some impact on organization, especially if this best practice is not implemented yet. With FOSS, organizations cannot rely on support contracts and have to build up knowledge of the components in-house.

**BEST PRACTICE 2: AGILE DEVELOPMENT**

The expert interviews indicate a strong preference for agile methodologies when it comes to creating big data solutions. Therefore, architects and project managers should be advised to create software
and hardware iteratively, and release small changes to an existing working solution. Examples of methodologies that have proven to be successful in an “agile” way are Scrum [41], Kanban [42], Lean [43], and XP [44]. All these methods have in common that they stress high-quality working solutions by having small teams working collaboratively in short iterations, focusing on the delivery of useful artifacts. In a certain sense, the Rational Unified Process (RUP) can also be considered “agile” when applied in the correct way, although this methodology is usually not thought of as a purely iterative but rather as a mixture of traditional “waterfall” and modern “agile” approaches [45]. Each organization should take their pick for a method, as long as the principles in the Agile Manifesto [46] are applied strictly. For some organizations that are engaging a big data project, an agile way of working will already be in place since agile is becoming the de-facto standard in software development [47]. If that is not the case, the introduction of agile development will introduce some difficulties as a switch from traditional methodologies can be a cultural challenge [48].

E. Summary
The Big Data Solution Reference Architecture is a model for creating solutions that make predictions about the future using open data sources and structured, semi-structured, and unstructured enterprise data. The reference architecture is usable for architects in “green field” situations or in projects with an existing technology base. The reference architecture is generic in purpose: any commercial or public organization can use it to apply in a typical big data use case. The Big Data Solution Reference Architecture presents the following key points to architects:

- Create a big data solution that is derived from the components & interfaces diagram;
- Think of the big data solution as a pipeline, with components that act as filters;
- Divide the solution in layers;
- Make sure components are loosely coupled;
- Use open standards to enable interoperability;
- Use free and open-source software components wherever possible;
- Develop agile.

VI. USAGE OF THE BIG DATA SOLUTION REFERENCE ARCHITECTURE

The reference architecture can be used to create solutions for use cases of predictive analytics using big data technology and open data sources. The following table contains a small subset of all use cases, meant to give an appetite of the possibilities; there are plenty example in the given industries and sectors such as telecom, energy, education, and others. Each organization should find its own use case and purpose for the reference architecture.

<table>
<thead>
<tr>
<th>Industry</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defense</td>
<td>A ministry of defense wants to build a system to collect and analyze signals, to notice national security threats (SIGINT). The system predicts the chance that a certain data source or communication contains hostile information that could harm citizens.</td>
</tr>
<tr>
<td>Financial Services</td>
<td>A national authority for financial markets wants to improve fraud detection of credit card data. By using a sophisticated prediction engine, individual transactions can be marked as likely to be fraud based on the behavior of clients.</td>
</tr>
<tr>
<td>Financial Services</td>
<td>A large bank wants to offer a service to clients that predicts account balance. When a client log in to his or her personal online banking website, a forecast of the balance on the bank accounts in possession is shown on the screen. The forecast is based on historical earnings and spending of the individual as well as the group the person is in, based on social categorization. Forecasts can also include other data sources such as the search behavior of the person on the internet; if the client visited second-hand car sales pages it's likely he or she will buy a car in the near future.</td>
</tr>
<tr>
<td>Government</td>
<td>A local government organization with its own customer support helpdesk want to predict the load of calls, and thereby the staffing needs of the helpdesk</td>
</tr>
<tr>
<td>Government</td>
<td>A national law enforcement agency wants to predict crime threats by analyzing sensory data, social media, web traffic and email records.</td>
</tr>
<tr>
<td>Healthcare</td>
<td>A hospital wants to reduce re-hospitalization figures, and improve &quot;patient flow&quot; to increase the quality of care and reduce costs. A prediction is made for each patient that determines the risk of recurring illness once he or she is discharged from the hospital. The prediction is based on historical data, patient profile, and the latest illness research reports.</td>
</tr>
<tr>
<td>Healthcare</td>
<td>A national health organization wants to predict outbreaks of diseases as soon as possible, to distribute medication, and take other pre-emptive actions. Sources for the predictions are hospital data, social media, illness records of companies, online news feeds, and others.</td>
</tr>
<tr>
<td>Insurance</td>
<td>An insurance company wants to forecast the amount of deaths and other indicators of life insurance payments, to adjust policies and manage costs. The data used for these predictions are customer profiles (including income, location, age, and sex), historical data, and sources that contain indications of disease outbreaks such as news feeds and social media.</td>
</tr>
<tr>
<td>Retail</td>
<td>A manufacturer of mobile phones want to forecast the amount of sales for the upcoming period, based on historical data, market trends, and sentiment of (potential) customers.</td>
</tr>
<tr>
<td>Retail</td>
<td>An e-commerce website wants to promote cross-selling of products by presenting related products to potential customers. The prediction of products that are likely to be interesting to the customer is performed by an analytics engine that takes various sources as input: other clients' buying behavior, web traffic of the customer from cookies, and price differentiation of products on sale.</td>
</tr>
<tr>
<td>Oil and gas</td>
<td>An oil refinery wants to predict machine failures to optimize costs and downtime. The machines produce sensor data that can be used for analytics, as well as working schedules and sales forecasts.</td>
</tr>
<tr>
<td>Transportation</td>
<td>A railway company wants to lower the costs of maintenance on trains by improving the predicted replacements of train parts, based on sensor data of the trains.</td>
</tr>
<tr>
<td>Transportation</td>
<td>A national government wants to predict road traffic flows and congestions. These predictions can be used to optimize digital road signs and send better routes to in-car satnav systems. The predictions are based on actual traffic data, historical data, Twitter feeds, public holidays, and other sources.</td>
</tr>
</tbody>
</table>

VII. EVALUATION

Ten big data experts out of a sample of 50 completed a questionnaire that investigated the quality aspects of the Big Data Solution Reference Architecture. The results of the questionnaire indicate that the reference architecture meets all of the quality criteria that were defined in the research design: maintainability, modularity, reusability, performance, and scalability. Moreover, the general impressions of the model are reasonably positive. There are high scores on the likeliness that architects and other big data experts are going to use elements of the model. Altogether, this indicates that the model can be qualified as a ‘reasonably good’ reference architecture. It will have its use in the architecture community, and will probably be adopted once published. However, there is some criticism on the reference architecture as well. Most importantly, respondents question the usability of the model. The respondents give relatively low ratings to the performance, scalability,
completeness, level of detail, and the concreteness of the model. The reason for doubt on the usability is primarily due to the level of abstraction; the model is considered too general, especially the architectural patterns, architecture principles, and architectural best practices.

VIII. CONCLUSION

This paper described a research project for a reference architecture of big data solutions. The reference architecture was designed using qualitative data analysis and grounded theory, and evaluated using a questionnaire that investigated several quality criteria.

A. Observations

By presenting a questionnaire to a group of big data experts, the resulting reference architecture was justified and evaluated. The quality of the model was ramified into five criteria: maintainability, modularity, reusability, performance, and scalability. A group of ten people completed the questionnaire, was out of a sample of 50. That is too little to make hard classifications about the quality of the model. However, the questionnaire was only distributed to people working with big data, and all respondents were more or less knowledgeable and experienced in that area. This makes the results of the questionnaire fairly accurate, and gives certain weight to the outcome. Nonetheless, the results should not be treated as scientific proof but as indicative evidence.

The results of the questionnaire indicate that big data architects will likely use the Big Data Solution Reference Architecture in their work. The following table displays the average scores on the quality criteria, on a scale of 1 to 5.

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>2.95</td>
</tr>
<tr>
<td>Modularity</td>
<td>3.10</td>
</tr>
<tr>
<td>Reusability</td>
<td>3.00</td>
</tr>
<tr>
<td>Performance</td>
<td>2.70</td>
</tr>
<tr>
<td>Scalability</td>
<td>2.70</td>
</tr>
</tbody>
</table>

The overall average quality score is 2.99 on a scale of 1 to 5. This answers the main research question; the created model is a ‘reasonably good’ reference architecture for a solution with big data technology to perform predictive analytics of open data sources combined with structured, semi-structured, and unstructured data sources.

B. Contribution

Since the Big Data Solution Reference Architecture is a ‘good’ reference architecture for its purpose, the question rises what this implies.

First, the model is unique in its kind. It is the first (and currently only) reference architecture for big data, predictive analytics, and open data that is somewhat supported by scientific evidence. This makes the model the best reference architecture for architects working in this area; all other reference architectures are either commercial in intend or have been created by individuals or organizations without evidence of the components. Most authors even fail to explain the reasoning behind their model.

Another aspect that makes the Big Data Solution Reference Architecture unique is it completeness. Most similar reference architectures only consist of a diagram of software components. In contrast, the Big Data Solution Reference Architecture contains components & interfaces, architectural patterns, architecture principles, and architectural best practices. All of these elements are described in detail and are backed up by literature research and a qualitative data analysis (grounded theory) of expert interviews.

The results of the questionnaire indicate that it is likely that big data experts will use the Big Data Solution Reference Architecture in their daily work. That statement in itself is a strong accomplishment; an important goal of the model is to have a place in real projects by big data architects. Finally, by performing extensive literature research and interviewing subject matter experts, this thesis has made an important contribution to the fields of big data, solution architectures, reference architectures, BI, and predictive analytics. By documenting the findings, the knowledge about these subjects has been enlarged and deepened.

To summarize, the Big Data Solution Reference Architecture is a new and unique model that delivers a strong contribution to the community of architects and other people who are working with big data technology, open data, and predictive analytics.

C. Future Research

An obvious future step is to create a solution architecture with guidance of the reference architecture. By conducting one or more case studies with the model, its practical use and quality can be investigated. An example of a case study is to create a solution architecture for a bank, with the goal of combining open data sources (e.g. weather data) with enterprise data (e.g. bank account balances) to produce a forecast (e.g. buying behavior of citizens in a shopping area). Another example is to create an architecture in a government organization that helps to predict the amount of violence in the upcoming weekend by combining holiday dates, the football calendar, traffic data, weather data, and others.

The Big Data Solution Reference Architecture was only evaluated with ten respondents to the questionnaire. A future research project could repeat the evaluation with a larger sample, to gain more accurate insight into the quality of the reference architecture.

Another iteration of Hevner’s Design Science framework could improve the model. As suggested, the type of reference architecture according to Angelov’s framework could be adjusted to increase the levels of concreteness and detail of the reference architecture. Another option is to maintain a type 3 architecture, and design the model so that the concreteness and detail are maximized within the boundaries of Angelov’s model. The third option is to step away from Angelov’s framework and use another model to design an improved version of the reference architecture.

Finally, a future research project could investigate the way in which organizations move from traditional BI systems to modern big data analytics. It would be useful to see if the Big Data Solution Reference Architecture can play a role in migration projects, e.g. by mapping the existing BI software components to new systems in the target architecture.

IX. ACKNOWLEDGEMENTS

I would like to express my gratitude to all of the people who participated in the project. My lectors Raymond Slot and Norman Manley gave excellent guidance and spend a lot of time reviewing my material. Teachers Kobus Smit and Bas van Gils have helped me tremendously by providing useful feedback.

X. REFERENCES


