The GOBIA Method: Fusing Data Warehouses and Big Data in a Goal-Oriented BI Architecture

David Fekete
European Research Center for Information Systems (ERCIS)
Leonardo-Campus 3
Münster, Germany
david.fekete@ercis.de

ABSTRACT

Traditional Data Warehouse (DWH) architectures are challenged by numerous novel Big Data products. These tools are typically presented as alternatives or extensions for one or more of the layers of a typical DWH reference architecture. Still, there is no established joint reference architecture for both DWH and Big Data that is inherently aligned with business goals as implied by Business Intelligence (BI) projects. In this paper, the current iteration of a work-in-progress approach towards such custom BI architectures, the GOBIA method, is presented to address this gap, combining a BI reference architecture and a development process. A use case example is presented to illustrate the proposed method.

Keywords

Big Data, Business Intelligence, Conceptual Architecture, Development Method, Business Intelligence Architecture

1. INTRODUCTION

Big Data has generated widespread interest among both academia and practitioners [14]. Several new products (such as Apache Hadoop) and approaches have emerged that allow to store or process Big Data, which was not feasible or efficient before. Big Data is larger, more diverse, and speedier than data in established traditional technologies. As a consequence, Big Data challenges often exceed an organization's capability to process and analyze data in a timely manner for decision making [14]. On the other hand, traditional Data Warehouse (DWH) architectures are an established concept for Business Intelligence, based on a common reference architecture (e.g., [13]). Nevertheless, with the plethora of novel Big Data products, the question arises which impact these have on analytic architectures and how the investments into warehouse technology, that companies have made over the years, can be preserved. In this paper, an initial answer to these questions is attempted by outlining a reference model fusing Big Data and DWH.

28^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 24.05.2016 - 27.05.2016, Nörten-Hardenberg, Germany.

Apache Hadoop distributions such as MapR[1] offer so many component products that building a customized architecture suited for the specific purposes of an application is rendered an increasingly complex task. Overall, the solution space available for Big Data and Business Intelligence endeavors is far more diverse than in previous times (e.g., cf. [11]). Thus, additional clarity on the process of deriving a customized architecture from a reference architecture is required as well.

The goal of this work is to design artifacts that address these questions following a design science approach [8]. The artifacts designed here are an enhancement of a initial proposal for the issue at hand (cf. [4]). To this end, a theoretical background on the foundations of the solution artifacts is given in Sec. 2. The various solution artifacts are described and illustrated by an example case in Sec. 3. Finally, the work is summarized and next research steps are outlined in Sec. 4.

2. FUNDAMENTALS

In this section, the basic concepts regarding architectures and Business Intelligence required for the proposed solution are explained and the problem statement to be addressed is outlined. Definitions of and further reading on the basic terms Data Warehouse and Big Data can be found in [5, 6, 13] and in [15, 14].

The term *Business Intelligence* (BI) is used to describe a holistic enterprise-wide approach for decision support that integrates analytics systems (e.g., a DWH), but also strategy, processes, applications, and technologies in addition to data [2, p. 13]. Besides that, BI is also said to postulate the generation of knowledge about business status and further opportunities [2, p. 13]. More importantly, a crucial aspect of BI is its alignment to its business area of application [2, p. 14]. This implies, that BI and also its parts (including an analytics system) should be aligned to the respective business in order to support decision making for business operations.

While a traditional DWH architecture has well-defined layers such as the staging area (Extract-Transform-Load, ETL) or data marts [13, 6], several examples for Big Data attached to DWH architectures (e.g., with Big Data tools used for ETL) can be found. Typically, these represent specific setups (e.g., [15, p. 23], [7, p. 40]), but cannot be generalized into a reference architecture. Other attempts include more general (reference) architectures (e.g., [12, p.

[1] https://www.mapr.com/products/mapr-distribution-including-apache-hadoop

62], [7, p. 17], [14]), yet the question remains of how to allocate (which) products to specific roles in an architecture, especially with several alternatives to traditional DWH architectures and products available. This is exacerbated by the fact that some of these new product offerings can be used for multiple purposes inside such an architecture. For example, MapReduce as a generic tool can be used for data preprocessing (e.g., performing large-scale cleansing operations) as well as for an actual analysis (e.g., basic word count statistics or sentiment analyses).

However, no BI reference architecture has been established yet that is inherently technology-independent, i.e., usable for both DWH and Big Data, and addresses the business-alignment of BI. Such *goal-orientation* aids the selection of customized architectures, since specific goals can be considered in the process.

As several combinations of technologies and products can be placed in an analytics architecture nowadays, the potential complexity of architectures is increased. For instance, certain Apache Hadoop distributions (e.g., by MapR[2] or Hortonworks[3]) present all of their offered product options in a single package, where no process to a customized architecture is outlined and the necessary architectural choices are left to the implementer. For instance, if a weather prediction BI application should be implemented using these Hadoop distributions the fitting products have to be chosen. While these choices could possibly be made with certain effort, e.g., Apache Storm for streaming weather data processing and MapReduce for batch analytics, the process of arriving at these decisions cannot be supported best solely by considering a (simple) classical layered view as with the DWH reference architecture before. Previously, this view was sufficient as typical products were located mainly in the DWH sphere, but to match todays complexity and heterogeneity from an architectural point of view, the classical layered view needs to be further refined.

Reference architectures used in computer applications typically exhibit a layering of services [1]. The various layers interact through well-defined interfaces, and their structure commonly follows an abstraction process. Indeed, the top layer comprises the most coarse (high-level) services, which are refined at the next lower layer, and this is often repeated until a layer of most basic functions is reached. In other words, in a system representing a service hierarchy, higher-level services are realized by lower-level services.

An example for a service hierarchy is a high-level telecommunication service provided to an end-user that can be comprised of several lower-level services in the back-end. In a data analytics scenario, high-level analytical services could be placed in a core analytics layer (e.g., "Cluster customer groups" or "Sentiment analysis of product-related tweets") and be consumed by BI applications on top, possibly supplied to by a middle-ware (e.g., data marts). These services are provided for by data preprocessing services (such as "Cleanse customer data" or "Filter tweets") at a lower layer and are ultimately based on several data sources (e.g., "Twitter" or "ERP"). Each of these services can be allocated, respectively be backed, by a novel or traditional product. However, the challenge of actually allocating these heterogeneous products to layers or services in a specific use case remains and needs to be addressed. We do so using a service hierarchy within a layered architecture that serves as a guide towards a final implementation of a customized architecture, since it allows for a clear structuring of complex architectures in a modern heterogeneous product landscape.

Abeck et al. describe a layered architecture as a foundation for (software) reference architectures, as software systems development would be based on layering [1]. Employing a layered architecture for a BI reference architecture could use these properties during customization and place adequate BI-related services at the appropriate architectural layer, which adhere to the intended level of abstraction. When BI is seen in this way, a general reference architecture can individually be customized and hence aligned to the goals and requirements of a specific business use case. Goal orientation and layered architecture should therefore be part of the solution artifacts to be designed, which will be elaborated upon in the following.

3. GOAL-ORIENTED BUSINESS INTELLIGENCE ARCHITECTURES

The proposed approach is termed the "Goal-oriented Business Intelligence Architectures" (GOBIA) method and consists of a BI reference architecture (GOBIA.REF) and development process (GOBIA.DEV). In the following, both artifacts are briefly presented.

The main idea of the method is to offer a more abstract and layered BI reference architecture (GOBIA.REF) in combination with a process that aids in transforming business goals into a customized BI architecture (GOBIA.DEV). It is not the primary intent of the method to assemble all BI technologies from an overall BI solution space into an all-encompassing reference architecture. Instead, guidance through a vast technological landscape and a large variety of possible use cases should be offered. This should allow to cope with present complexity and heterogeneity in architectural possibilities and, potentially, future ones as well.

GOBIA.REF aims to address the architectural gap outlined above and is intended as a layered, technology-independent BI reference architecture. The accompanying development process GOBIA.DEV aids in its customization, so that the outcome is aligned to the goals and requirements of a specific scenario or application. This inherently supports the principle of BI to be business-aligned. This is achieved with a high-level conceptual architecture as intermediate step. It resembles a service hierarchy, which is not yet focused on technical details, but aims to alleviate the challenge of implementing the architecture in a final step (i.e., assigning specific products to the defined roles and functions).

An initial proposal for GOBIA (cf. [4]) is enhanced by redefining and detailing both GOBIA.REF and GOBIA.DEV. In this iteration, an overall process for GOBIA.DEV is now explicated and GOBIA.DEV itself remodeled with a process modeling language (Petri nets) to render the process more precise. Also, the wording of elements is adjusted in GOBIA.REF (e.g., BI services instead of applications; BI Functionality Marts instead of Data Marts). The aim is to make both parts of GOBIA more concise and commonly exhaustive in total.

3.1 BI reference architecture

The proposed BI reference architecture (see Fig. 2 on the

[2]http://doc.mapr.com/display/MapR/Architecture+Guide
[3]http://hortonworks.com/building-an-enterprise-data-architecture/

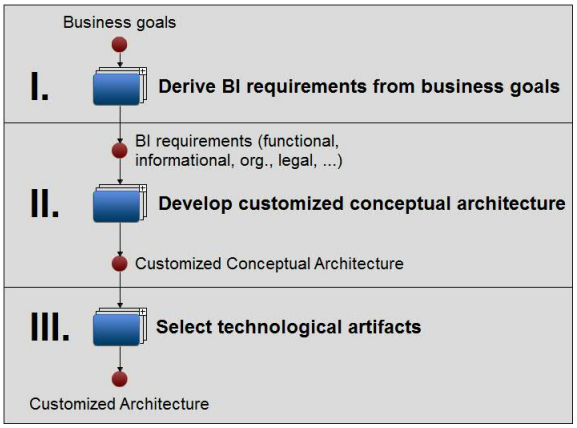

Figure 1: Overview of the 3-phased customized BI architecture development process for the GOBIA method (GOBIA.DEV).

bottom left) as a layered architecture generalizes DWH and Big Data in the core analytics layer as "BI functionality" as common denominator. The customized architecture is built based on this reference architecture and should be seen as a service hierarchy.

Data for the architecture reside at the bottom of the reference architecture. These can be located internally or externally (e.g., in a cloud). While this is comparable to other architectures, no restrictions are imposed on data formats or delivery and persistence modes. For instance, data source blocks could simply be conceptual data entities such as "Inventory data", which are realized by one or more actual (real) data sources. The "Data Preparation" above it fulfills a similar purpose as the staging area in a DWH, but the tasks should be more coarse-grained and mostly omit technical details. For instance, a task in this layer could be to "Transform and normalize prices" or to "Extract purchase history". Instead of having a DWH or Big Data tools in the analytics layer, it contains BI functionality, which is technology-independent and focused on the results of BI. For instance, BI functionality could include high-level functionalities such as "Classify customer into types" or "Identify sales patterns". BI functionality marts, similar to data marts in a DWH, can fulfill the role to provide flexible subsets of data to BI services. As the layers are conceptual, a decision whether to materialize functionality marts is not made at this point.

BI services consume the BI functionalities delivered through the BI functionality mart layer to deliver applications to a client or end-user, as common in other architectures. The difference, however, is that GOBIA.REF aims to clarify on the actually needed BI functionality so that the choice of selecting suitable technological artifacts afterwards becomes less complex. BI services are later represented by actual BI application software providing these BI services to the respective target user groups.

3.2 Development process

The development process for the customized architecture, GOBIA.DEV (see Fig. 1), is designed so that business goals ultimately lead to the customized architecture. GOBIA.DEV follows a three phase approach, starting with business goals, which are assumed to be given. These are used to derive specific BI requirements used in the second phase to build a customized conceptual architecture. This conceptual architecture, technology-independent and functionality-focused, is the basis to perform the actual technology artifact selection and assembly of the customized architecture.

3.2.1 BI requirements definition

In the first phase, specific BI requirements are derived using more coarse-grained business goals. Firstly, however, business goals lead to more detailed and specific business use cases. The underlying domain (e.g., retail, finance...) is also given by these use cases. Business use cases as a popular instrument in organizations are used to determine the actual BI requirements on the customized architecture. This includes functional and information requirements, e.g., business-relevant information such as costs, expected value, or revenue to be delivered. Moreover, requirements could be of legal (e.g., for data protection) or organizational respectively project-related (e.g., use of specific tools, budget) nature. A functional requirement could, for example, be to "Analyze customer behavior to map his characteristics to products that he might find interesting". Often, these already hint which kind of information is required to be delivered. Existing requirements engineering approaches can be employed for this purpose (e.g., cf. [10], [3]) as GOBIA.DEV does not intent to impose a specific approach as long as actionable BI requirements for the next phase are defined in the process.

3.2.2 Conceptual architecture development

The second phase handles the development of a conceptual BI architecture (cf. Fig. 2). It focuses primarily on technology-independent BI functionality and abstract BI data entities. The development is based on a so-called co-alignment. The main outcomes are BI functionalities to be placed in the architecture, as well as necessary data preparation tasks, and required data (including defined data properties). All outcomes are finally assembled into a layered conceptual architecture. Based on the BI requirements, a custom GOBIA.REF is selected if need be. With this, domain-specific template reference architectures could be possible like, e.g., a set of typical finance-algorithms as BI functionality templates. Also, there can be a strategic impact on it, e.g., to not use any BI functionality marts.

For this, requirements are aligned with BI functionality and data properties. The result should be that, eventually, suitable BI functionalities adhere to the requirements set before and that these BI functionalities and data preparation tasks fit the data at hand. If, e.g., a requirements was to differentiate groups of customers, BI functionality for a suitable clustering method must be defined.

Data is characterized by using the "V's" [15, 14], which are typically used in Big Data context, but should be applied to any data in this method. For instance, if the quality of a data source is poor (e.g., low validity or high vagueness), but the set goal is to work on higher quality data, proper data cleansing or enrichment tasks have to be conducted.

Notably, co-alignment can also mean that requirements are re-adjusted iteratively. For example, if the actual data sources are of higher quality than necessitated by the requirements, the latter could be refined to explicitly exploit this data. That refinement, then, could lead to a further

adjustment of BI functionality or data preparation tasks.

In addition to this, another co-alignment is used to synchronize required data for the BI functionality and actually deliverable data (sources) in an organization. This step can take place right at the beginning or after the aforementioned co-alignment. If it takes place in the beginning, BI functionalities can be immediately synchronized with BI data entities that reflect what is currently deliverable in an organization. If this is done after BI functionality was co-aligned with required data entities, the modeling of needed functionality and data can be aligned to the actual requirements first, before deliverable data is aligned against it. The respective modeling parties should be able to choose which approach fits the architecture development needs or modes of operation in an organization. For instance, if it is challenging to retrieve which data sources are actually available, the development of required conceptual elements (functionalities and data entities) could begin without delay.

Then the customized conceptual architecture is assembled by assigning the outcomes of the co-alignment (e.g., BI functionalities) to the respective layers and by building a service hierarchy. Such service-hierarchy is built by, on the one hand, defining lower-level functions and data sources that comprise the higher-level conceptual elements (if not done yet). On the other hand, BI functionality marts and BI services must be defined so that the BI functionality can be delivered in accordance with the requirements.

3.2.3 Technology artifact selection

This high-level conceptual output is an intermediary result to select an appropriate set of technologies. The conceptual architecture should alleviate the technology selection by providing necessary decision-related information in a structured way. The technology selection phase is currently under research. After technologies have been selected and composed to an actual BI architecture, the implementation of it can start and GOBIA.DEV has reached its end.

3.3 Sample case: Retail market

To illustrate the first part of GOBIA.DEV a sample use case based on a IBM game retailer case is employed [9]. It contains to-be-achieved business goals and certain details of the case, which lead to BI requirements.

The GOBIA.DEV process is followed until an initial conceptual architecture is developed and deliverable data sources are aligned to it. Out of scope for this illustration are the derivation of BI functionality marts and BI services as well as the technology selection phase.

3.3.1 Business goals, use case, BI requirements

BestMart, a video game retailer, "wants to become the salesleader for video games this season" [9]. This is to be achieved by gaining competitive advantage through novel analytic techniques. The plan is to enhance existing enterprise data with "other relevant information to create predictive models of trends" [9].

These trends should help in predicting the "hot gaming item" [9] (i.e., a top-selling video game) for the respective sales season. Also, the quantity of video games in stock is to be optimized at the retailers and the central online shop.

The requirements can be extracted as-is from the source material as following [9]:

1. Predict areas where demand would be greatest.
2. Synchronize prices hourly with demand, inventory and the competition.
3. Pinpoint the customers who will likely buy [the top selling video] game by segmenting customers according to expected buying behavior. Know what other items customers would likely purchase with this game.
4. Contacting them as they wish to be reached, when they are in the right location, engaging them with personalized real-time offers.

These requirements are comprised of functional and informational requirements. Other possible requirements (e.g. legal or organizational) are not detailed in this sample case.

3.3.2 Initial conceptual elements

The creation and alignment of conceptual elements can follow after BI requirements have been defined. Here, these can be transformed almost directly into BI functionality by mainly adjusting the wording (e.g., "Predict hot demand areas" or "Determine new price suggestions by product").

Next, required data entities can be created, such as the conceptual ones "Competitor Prices" and "Online and Retail Sales Data" for a hourly price adjustment. Sales data allows to estimate the product demand as input to the adjustment BI functionality. There is a certain degree of a freedom when defining conceptual entities. For example, online and retail data could be separate if this were to render the architecture more comprehensible. Distinct data entities could, e.g., help to partition data when delivering it to the functionality later during implementation.

Competitor prices may need to be transformed (e.g., due to varying packing quantities) or normalized (e.g., due to different currency) before using them. Thus, a data preparation step is created as as result of an alignment between functionality and data entity. The full initial conceptual architecture is depicted in Fig. 3.

3.3.3 Alignment to deliverable data sources

The following step is to ensure that the data entities are aligned to actually deliverable data sources. A possible misalignment and its resolution can be demonstrated with the entity "Competitor prices". Inspection of actual data sources might reveal that the prices are located on the competitors websites and must be extracted first. Thus, the data entity is renamed to "Competitor websites with price information" and a data preparation step for the extraction is added ("Extract product prices from websites") before the transformation and normalization step.

3.3.4 Refining the conceptual model

After this alignment, lower-level elements (e.g., BI functions for BI functionality) can be determined to add further detail to the conceptual model. For instance, to "determine new price suggestions by products", the current product demand needs to be estimated and the own prices need to be compared to the competition. These two BI functions deliver the necessary information to decide upon new prices on a hourly basis. Where needed, more sub-layers of BI functions could be created.

3.3.5 Next steps

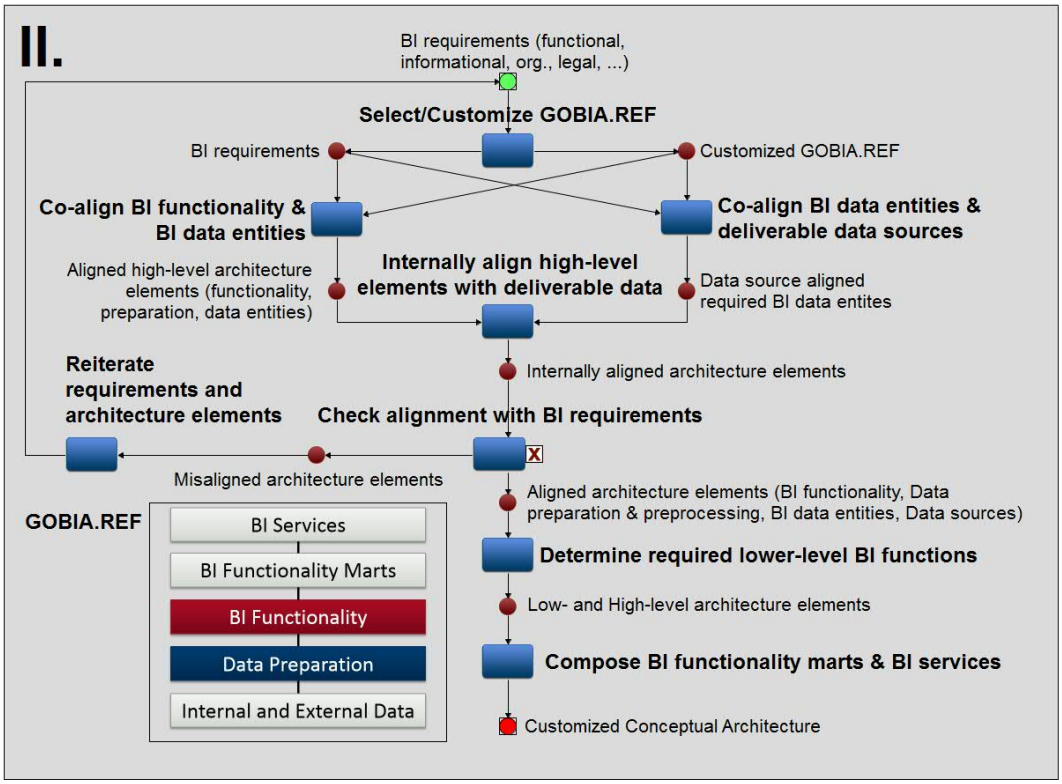

Figure 2: GOBIA.DEV phase II: Customized conceptual architecture development (as simplified Petri net) using the BI reference architecture proposal (GOBIA.REF) on the bottom left.

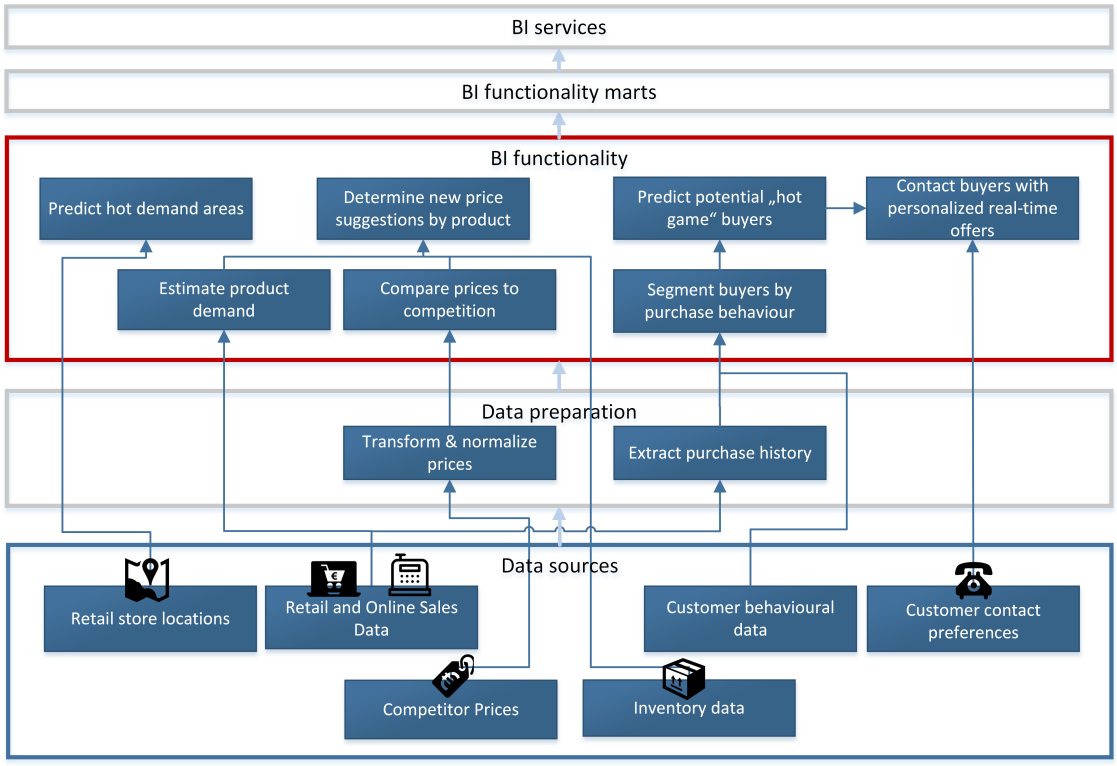

Figure 3: GOBIA retail example: Conceptual architecture before alignment with deliverable data.

The immediate next steps would be the completion of the conceptual BI architecture by determining BI functionality marts and BI services.

The following third phase would detail the technology selection, which is also a future research item (cf. Sec. 4).

4. SUMMARY AND FUTURE WORK

This work has attempted to show how new developments that arose in the Big Data trend necessitate a more "universal" reference architecture for BI. For this end, a BI architecture based on a layered architectures as a basic concept in Computer Science was proposed. To support this generic BI reference architecture and to manage the technology complexity, a proposal for an accompanying development process was made, which is to support a goal-oriented development of a customized BI architecture. It does so by yielding a conceptual architecture as intermediate step for selecting an appropriate mix of analytic technologies. A use case sample was then employed to illustrate the proposed method.

Future research should address the two following areas. Firstly, the technology selection phase needs to be detailed, because the conceptual architecture cannot be implemented directly. The challenges to select appropriate technological artifacts (e.g., from a Hadoop distribution) orchestrate them to one overall system is not yet solved. One main issue is to find a way to bridge the conceptual BI architecture and a fitting subset of the BI "solution space" (i.e., all technological artifacts that could be used). To tackle this, properties or characteristics of the artifacts and the conceptual architecture could be determined to find commonalities, which could aid in the selection process. Also, it is to determined how BI requirements can support this. In this context, it will be important to determine which characteristics are rather strictly defined terms (e.g., suitable data formats that do not necessarily need to be matched manually) or rather loosely defined (e.g., textual descriptions of skills required). For instance, existing architecture setups or use cases could be analyzed to derive best practices or generalizations. These findings would also be useful to discover elements for templates such as common BI functionalities or preparation tasks for specific data situations. Moreover, these may lead to refinements of the GOBIA method itself.

Secondly, both reference architecture and development process should be evaluated empirically to test if they fit their intended usage. Such an evaluation should build, for example, a customized architecture based on a Hadoop framework (e.g., MapR) to verify whether the process is indeed less complex when using the GOBIA method. Preferably, this would be conducted in a real-world environment (e.g., by actually using the GOBIA method to define a conceptual architecture in a real BI project) to gain better practical insights. However, employing the method in several completed projects or use cases could give further insights about the performance of the method as well (e.g., as done in the illustration in this work, but with all three phases in full). Here, the several GOBIA outputs created could be compared to the outcome of the use cases and projects. Also, such evaluation should include a comparison to other existing approaches (e.g., for reference architectures) to better assess to which extent GOBIA.REF and GOBIA.DEV can utilize the proposed advantages in practice.

5. REFERENCES

[1] S. Abeck, P. C. Lockemann, J. Schiller, and J. Seitz. *Verteilte Informationssysteme: Integration von Datenübertragungstechnik und Datenbanktechnik*. dpunkt, Heidelberg, 2003.

[2] A. Bauer and H. Günzel. *Data Warehouse Systeme*. dpunkt, Heidelberg, 3rd edition, 2009.

[3] R. M. Bruckner, B. List, and J. Schiefer. Developing Requirements For Data Warehouse Systems With Use Cases. *Seventh Americas Conference on Information Systems*, pages 329–335, 2001.

[4] D. Fekete and G. Vossen. The GOBIA Method: Towards Goal-Oriented Business Intelligence Architectures. In R. Bergmann, S. Görg, and G. Müller, editors, *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB, Trier, Germany, October 7-9, 2015.*, pages 409–418, Trier, Germany, 2015. CEUR-WS.org.

[5] W. Inmon. *Building the Data Warehouse*. John Wiley & Sons Inc., New York, New York, USA, 2nd edition, 1996.

[6] W. Lehner. *Datenbanktechnologie für Data-Warehouse-Systeme*. d.punkt Verlag, Heidelberg, 2003.

[7] Oracle. Oracle Enterprise Architecture: An Enterprise Architect's Guide to Big Data, 2016. URL: `http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf` [last accessed: 2016-03-31].

[8] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77, dec 2007.

[9] D. Pittmann. Big Data in Retail - Examples in Action, 2013. URL: `http://www.ibmbigdatahub.com/presentation/big-data-retail-examples-action` [last accessed: 2016-03-10].

[10] J. Schiefer, R. M. Bruckner, and B. List. A Holistic Approach For Managing Requirements Of Data Warehouse Systems. *Eight Americas Conference on Information Systems*, pages 77–87, 2002.

[11] The Big Data Group LLC. Big Data Landscape, 2016. URL: `http://www.bigdatalandscape.com` [last accessed: 2016-03-29].

[12] M. Thiele, W. Lehner, and D. Habich. Data-Warehousing 3.0 Die Rolle von Data-Warehouse- Systemen auf Basis von In-Memory-Technologie. In *Innovative Unternehmensanwendungen mit In-Memory Data Management (IMDM)*, pages 57–68, Mainz, 2011. Wolfgang Lehner, Gunther Piller.

[13] G. Vossen. *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. Oldenbourg, München, 5th edition, 2008.

[14] G. Vossen. Big data as the new enabler in business and other intelligence. *Vietnam Journal of Computer Science*, 1(1):3–14, feb 2014.

[15] P. Zikopoulos, C. Eaton, D. DeRoos, T. Deutsch, and G. Lapis. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill, New York, USA, 1st edition, 2012.