The GOBIA Method: Towards Goal-Oriented Business Intelligence Architectures

David Fekete[1] and Gottfried Vossen[1,2]

[1] ERCIS, Leonardo-Campus 3, 48149 Münster, Germany,
firstname.lastname@ercis.de
[2] University of Waikato Management School, Private Bag 3105, Hamilton 3240, New Zealand, vossen@waikato.ac.nz

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, 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.

1 Introduction

Big Data has generated widespread interest among both academia and practitioners [9]. 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 it was with data in established traditional technologies. Big Data challenges often exceed an organization's capability to process and analyze data in a timely manner for decision making [9]. On the other hand, traditional Data Warehouse (DWH) architectures are an established concept for Business Intelligence based on a common reference architecture (e.g., [8]). Nevertheless, with the plethora of novel Big Data products, the question arises which impact these have on analytic architectures and which form a reference architecture for both Big Data and DWH could have. Especially Apache Hadoop distributions such as MapR[1] offer so many products that building a customized architecture is rendered an increasingly complex task. 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

 In: R. Bergmann, S. Görg, G. Müller (Eds.): Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Trier, Germany, 7.-9. October 2015, published at http://ceur-ws.org

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

design science approach [6]. To this end, a theoretical background on the foundations of the solution artifacts is given in Sec. 2. The various solution artifacts are described in Sec. 3. Finally, the work is summarized and next research steps are outlined in Sec. 4.

2 Fundamentals

The following fundamentals explain the basic concepts regarding architectures and Business Intelligence required for the proposed solution and outline the problem statement to be addressed. Definitions of and further reading on the basic terms Data Warehouse and Big Data can be found in [3,4,8] and in [10,9], resp.

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 [8,4], 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., [10, p. 23], [5, p. 18]), but cannot be generalized into a reference architecture. Other attempts include more general (reference) architectures (e.g., [7, p. 62], [5, p. 12], [9]), 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,

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

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 mentioned challenge of actually allocating these heterogeneous products to layers or services in a specific scenario remains and needs to be addressed (cf. Fig 1). 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. see 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 scenario or application. Goal orientation and layered architecture should hence be part of the solution artifacts to be designed, which will be elaborated upon in the following.

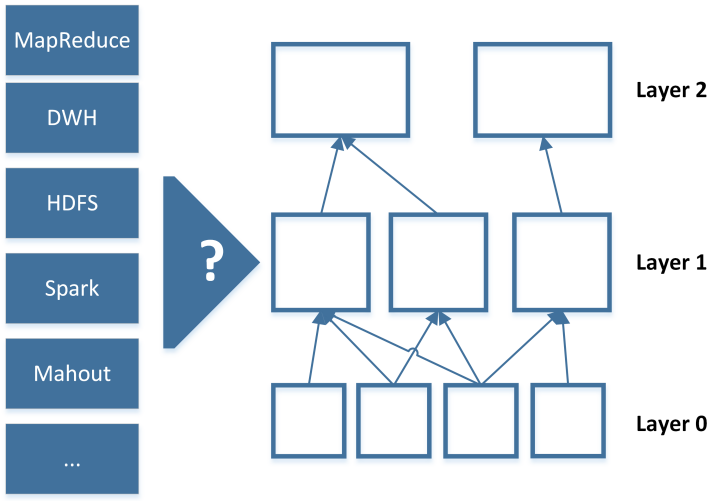

Fig. 1. A layered architecture with a service hierarchy (right) with the to be addressed gap of allocating heterogeneous products and technologies to it.

3 Goal-Oriented Business Intelligence Architectures (GOBIA)

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.

GOBIA.REF aims to address the architectural gap outlined above and is intended as a layered, technology-independent BI reference architecture. It is accompanied by a development process (GOBIA.DEV) that 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. The resulting architecture is a high-level conceptual model resembling a service hierarchy, which is not yet focused on technical details, but aims to alleviate the challenge of implementing the architecture (i.e., assigning specific products to the defined roles and functions).

3.1 BI reference architecture

The proposed BI reference architecture (see Fig. 2 on the right) as a layered architecture generalizes DWH and Big Data in the 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 sources of 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 "Mapping data" or "Transportation routes". The "Data Preparation and Preprocessing" 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 mapping data" or to "Complete disease data". Instead of having a DWH and/or Big Data tools in the core analytics layer, this layer contains BI functionality in general, 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". Data marts, as in a DWH, can fulfill the role to provide subsets of data to the BI-specific applications. As the layers are conceptual, a decision whether to materialize any of these subsets is not made at this point.

BI-specific applications consume the BI functionalities delivered through the data mart layer to deliver applications to a client or end-user, much as in many 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.

3.2 Development process

The proposed development process of the customized architecture, GOBIA.REF (see Fig. 2), is designed so that actual goals and requirements on a target BI system are derived from a more coarse-grained strategy, which is assumed to be already defined. The latter, indicated as (0) in Fig. 2, allows to derive application domain(s) and scenario(s) (use cases) from it in step (1). The underlying domain should define the playing field laid out by the strategy (e.g., finance, health care...). The scenarios, set in the domains, define the requirements and goals of the customized architecture (2), and business-relevant information such as costs, expected value, or revenue. A defined goal could, for example, be to "Analyze customer behavior to map his characteristics to products that the he might find interesting". At this point, the BI-specific applications required at the top-most layer are determined.

This is followed by a co-alignment step (3). The main outcomes are BI functionalities to be placed in the architecture, as well as necessary data preparation or preprocessing tasks, and data properties of suitable data. For this, requirements and goals are aligned together with BI functionality and data properties. The result should be that, eventually, suitable BI functionalities realize the goals and adhere to the requirements set before and that these BI functionalities and data preparation tasks fit the data properties. If, e.g., a goal was to differentiate groups of customers, BI functionality for a suitable clustering method must be defined.

Data properties are characterized by using the "V's" [10,9], 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 and goals are adjusted as well in the course of an iterative definition process. For example, if data properties for an initial set of requirements and goals are characterized and the data is of higher quality than expected , goals could be refined to explicitly exploit this data. This refinement, then, could lead to a further adjustment of BI functionality or data preprocessing tasks.

Input to the requirements and goals in step (2) is a BI reference architecture (GOBIA.REF). Also, there can be a direct strategic impact on it, e.g., a decision not to have any data marts in the final architecture. Moreover, domain-specific template reference architectures could be possible like, e.g., a set of typical finance-algorithms as BI functionality templates.

Finally, in step (4), the customized 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. This high-level conceptual output can be used further in the implementation of the target BI system.

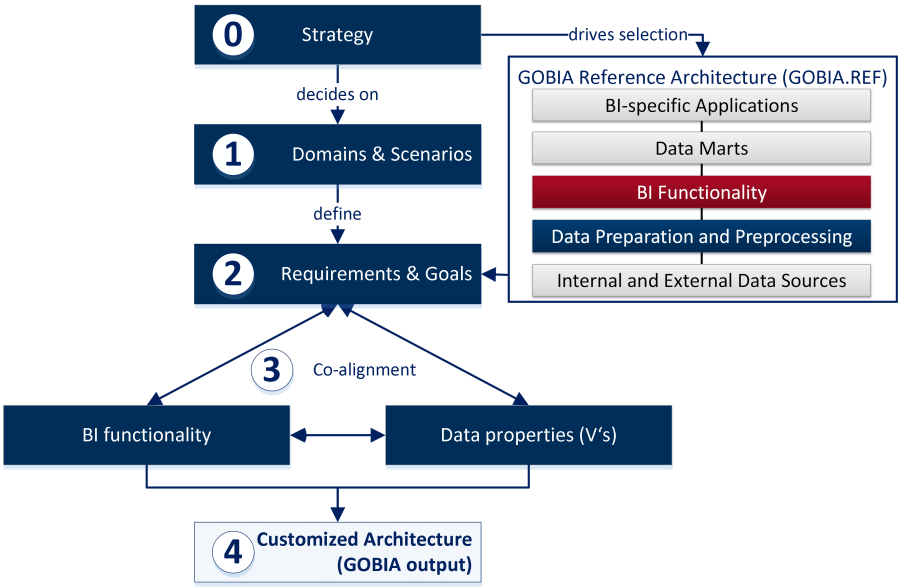

Fig. 2. Customized BI architecture development process proposal for the GOBIA method (GOBIA.DEV) and the BI reference architecture proposal (GOBIA.REF) on the right.

3.3 Sample case

For illustration purposes, a sample use case is briefly discussed and its outcome presented (cf. Fig. 3). This fictitious case is tailored towards a global organization concerned with the health of people, e.g. the World Health Organization (WHO). Firstly, the GOBIA.DEV process is executed to determine the goals and requirements and to present the functionality at the different architectural layers. Then, GOBIA.REF is used to assemble this into a layered and hierarchical form that can be used as a blueprint for a subsequent implementation.

The set strategy (0) could be the objective of the WHO, which "is the attainment by all people of the highest possible level of health" [4]. Naturally, healthcare is set as application domain (1). In this scenario, global disease management should be the specific application that allows for disease monitoring and pattern recognition to facilitate appropriate mitigation or containment procedures and that supports the set objective of the WHO.

The goal (2) of the BI system should be recognizing disease patterns around the globe to allow description and comparison of current disease spread patterns to eventually allow for enhanced monitoring. To achieve this, several data sources are required. These could be confirmed disease cases ("disease occurrence data"), incoming reports of potential, unconfirmed diseases ("emergent disease information") and knowledge about diseases like symptoms ("disease data"). To create a map and to predict endangered areas in the future location data — ranging from basic maps to more refined data as health infrastructure and environmental data, which could influence disease spreadings and potential dangers — is needed as well as, e.g., public and private transportation routes such as flight routes or roads. Furthermore, appropriate algorithms are required to map health dangers of certain levels to appropriate mitigation or containments procedures.

The results of the subsequent co-alignment lead to a service hierarchy or layered architecture as shown in Fig. 3. To conduct co-alignment, the properties of the required data sources should be assessed. For instance, because of the extensive mapping data, potential data volumes could be regarded as "high". Besides technical properties (volume, variety, velocity), qualitative properties as "value" can be assessed. In this case, value could be rather low for unconfirmed, emergent disease reports due to the uncertainty and poor initial data quality and be potentially high for actually confirmed disease cases from, e.g., hospitals. With this, the required BI-specific functionality can be formulated that supports the goals of the application as well as the necessary tasks that, e.g., deal with the data properties (such as the cleansing of unconfirmed disease cases). These tasks could include a classification of emergent disease information (i.e., if its actually a confirmed case or an irrelevant report). By clustering the cleansed and completed diseases occurrences, these can be integrated into the preprocessed mapping data. Lastly, disease patterns could be recognized and eventually plotted on an interactive map or assembled into regional health reports.

[4] `http://apps.who.int/gb/DGNP/pdf_files/constitution-en.pdf`

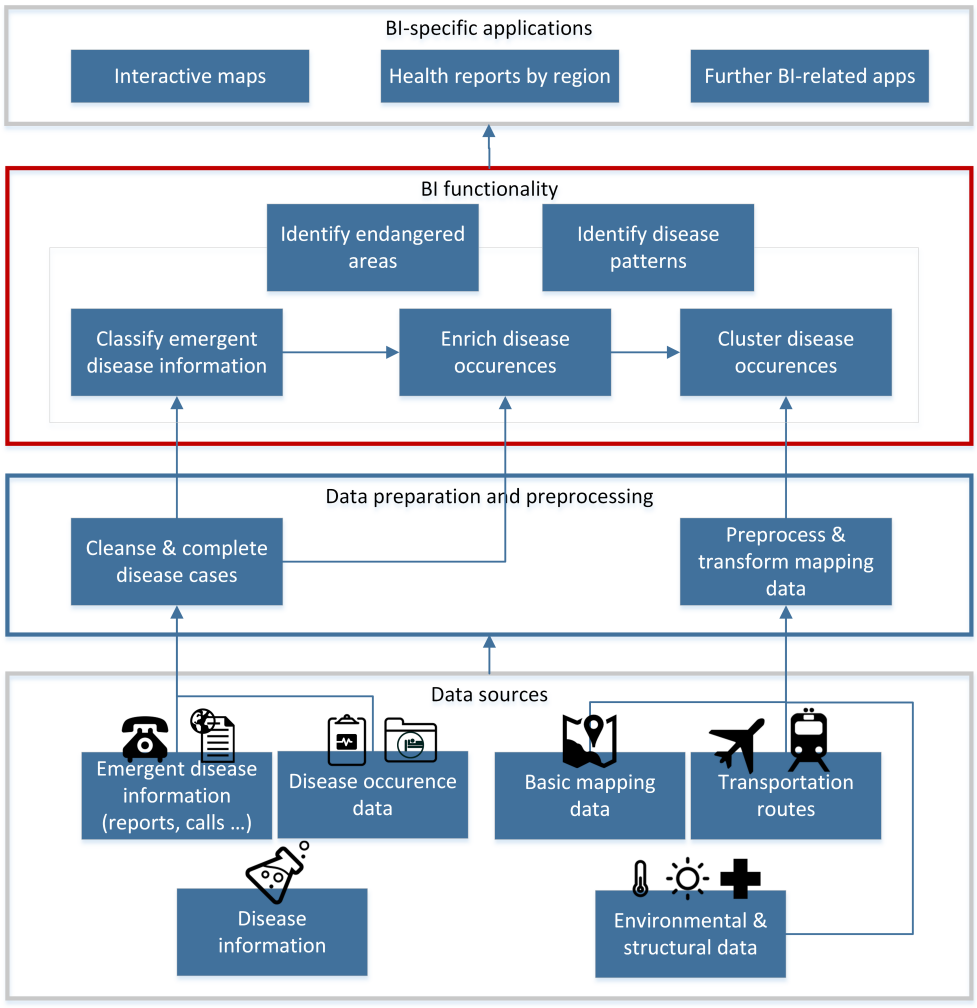

Fig. 3. GOBIA output example: Customized layered, hierarchical architecture for a fictitious public health use case.

4 Summary and Future Work

This work has tried to outline gaps in "universal" reference architectures that did arise as a result of moving into the age of Big Data. A proposal for a BI reference architecture based on a basic concept in Computer Science was made. A development process has been proposed to support a goal-oriented creation of a customized BI architecture, yielding a possible prerequisite for choosing suitable analytic tools.

Future work should address various parts of the proposed method. Firstly, the proposal is to be refined. For example, the semantics in the development process needs to be elaborated upon, and inputs and outputs be specified in more detail.

Secondly, the steps following an execution of the development process are to be elaborated, since the high-level conceptual model output cannot be directly operationalized. The challenge to select technological artifacts (e.g., from a Hadoop distribution) and connect these to realize the concept is not resolved yet. Here, best practices or generalizations of architecture setups could be derived in order to address this challenge. It should also be elaborated how findings from these can be generalized to templates to enhance the method itself. For instance, best practices could be used to derive domain-specific reference architectures or predefined building blocks for the co-alignment step in GOBIA.DEV (e.g., common data processing tasks that address certain data properties or specific BI functionalities).

Thirdly, 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. 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.

References

1. Abeck, S., Lockemann, P.C., Schiller, J., Seitz, J.: Verteilte Informationssysteme: Integration von Datenübertragungstechnik und Datenbanktechnik. dpunkt, Heidelberg (2003)
2. Bauer, A., Günzel, H.: Data Warehouse Systeme. dpunkt, Heidelberg, 3rd edn. (2009)
3. Inmon, W.: Building the Data Warehouse. John Wiley & Sons Inc., New York, New York, USA, 2nd edn. (1996)
4. Lehner, W.: Datenbanktechnologie für Data-Warehouse-Systeme. d.punkt Verlag, Heidelberg (2003)
5. Oracle: Oracle Information Architecture: An Architect's Guide to Big Data (2012), `http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf`

6. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 24(3), 45–77 (Dec 2007), http://www.tandfonline.com/doi/full/10.2753/MIS0742-1222240302
7. Thiele, M., Lehner, W., Habich, D.: 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). pp. 57–68. Wolfgang Lehner, Gunther Piller, Mainz (2011)
8. Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme. Oldenbourg, München, 5th edn. (2008)
9. Vossen, G.: Big data as the new enabler in business and other intelligence. Vietnam Journal of Computer Science 1(1), 3–14 (Feb 2014)
10. Zikopoulos, P., Eaton, C., DeRoos, D., Deutsch, T., Lapis, G.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill, New York, USA, 1st edn. (2012)