Defining Software Architectures for Big Data Enabled Operator Support Systems

Benjamin Klöpper, Marcel Dix, Lukas Schorer*[1], Ann Ampofo*
ABB Corporate Research Center Germany
{benjamin.kloepper, marcel.dix}@de.abb.com

Martin Atzmueller
Research Center for Information System Design
KDE Group, University of Kassel, Germany
atzmueller@cs.uni-kassel.de

David Arnu, Ralf Klinkenberg
RapidMiner GmbH, Germany
{darnu, rklinkenberg}@rapidminer.com

[1] Lukas Schorer from TU Ilmenau and Ann Ampofo from Heilbronn University have been research interns at the ABB Corporate Research Center

***Abstract*-Big Data technologies enable new possibilities to analyze historical data generated by process plants. One possible application is the development of new types of operator support systems (OSS), which could help plant operators during operations in identifying and dealing with critical situations. The project FEE has the objective to develop such support functions based on Big Data analytics of historical plant data. In this contribution we describe our approach to define software architectures for Big Data enabled OSS in industrial plants.**

I. INTRODUCTION

A typical process plant like a paper mill, a hot-rolling mill or a petro-chemical plant generates a large amount of documents and data throughout its entire life-cycle: I/O and tag lists, piping and instrumentation diagrams (P&ID), control logic, alarm configurations (during planning and commissioning), measurement values, alarm and event logs, shift books, laboratory results (during operation), maintenance notification, repair and inspection reports (during maintenance). Already today, analytics are applied to improve the operations of such a plant; however, usually only data from single data sources is considered. One common example of such an analysis is loop performance monitoring: Evaluating the control quality of single control loops based on measurements, set points, and controller output [1]. However, the limitation to a single data source hampers the process of learning from historical data.

An integration of all different types of data within process plants will bring up the notorious Big Data attributes: high volume, high velocity, and high variety [2]. For instance, a refinery can generate more than 300 GB measured values per year, produced by more than 60,000 sensors with sampling rates between 1 and 60 seconds. Furthermore, data is typically heterogeneous, existing in structured (sensor readings, database tables), semi-structured (alarm and event logs) and unstructured formats (shift books, operation manuals). Often, data has been often stored for 10 years or even longer.

The objective of the public-funded project FEE [3] is the analysis of the large and heterogeneous data volumes stored in chemical productions plants in a Big Data analytics platform with the aim to support plant operators. Big Data technologies will enable operator support systems, warning the operator early about unexpected and uncommon situations and support them in an ad-hoc analysis as well as in the development of intervention strategies. The main goal is a transition from reactive to a more proactive operation of process plants. Some of the central support functions that will be developed in the FEE project include early warnings, ad-hoc analysis with what-if scenarios, and an interactive search on vast amounts of historical data.

Chemical production plants imply specific requirements regarding the system to be developed. First of all, such plants are *safety-critical systems*, and consequently the requirements regarding the dependability and understandability of control systems are highly important to operators. Furthermore, the plants and the corresponding process control systems are *real-time systems* with deterministic deadlines. Even if no hard deadlines will be needed for data analytics, the applicability of the analytics results will depend on their timely or early availability. Overall, the development of data-driven assistance systems has to account for both the specific requirements of future users, as well as requirements arising from the technical environment. Existing reference models of data analytics or software development do not seem to be well suited for these tasks. In consequence, even during the early phases of the project FEE, the project team felt a need for the development of tools and methods tailored for the industrial domain.

This is a preprint of an article accepted for publication in Proc. INDIN 2016 IEEE International Conference on Industrial Informatics.

II. Related Work

Fan and Bifet [4] introduce and discuss different aspects of the current status as well as different challenges in mining Big Data with respect to extracting useful information from large datasets and streams. Similarly, Casado and Younas [5] outline trends and emerging technologies in that field. They outline a lifecycle for Big Data processing classifying different tools and methods in terms of the respective phases.

Begoli and Horey [6] discuss best practices for knowledge discovery in Big Data. They recommend to provide flexibility in the choice of analytical methods and tools, support of different storage engines based on data characteristics and analytics requirements, and end-user access to data by a light-weight architecture and communication protocols like a REST API.

Marz and Warren are more specific by introducing the Lambda Architecture as reference architecture for Big Data Systems [7]. The main concept of the Lambda Architecture is to keep an immutable collection of the original data and to use separate tools for specific tasks. This is in contrast to traditional multi-purpose databases that try to perform incremental calculations on the original data. Because the original data is not changed, this layout offers a high fault tolerance, because there exists always a clean backup. But calculating queries on the complete data set can be too slow and therefore so-called batch views preprocess the data. The batch views are usually realized in classic Big Data environment backed by Map/Reduce [8] like Hadoop, which offers parallelization and high reliability. The views will be stored in a serving layer, which will handle query request. Updating the batch views might take some time so that queries to the serving layer might be already outdated by new data. Therefore a speed layer is added, that should handle real-time analysis of arriving data until it is processed by the batch layer.

The original purpose of the Lambda Architecture was for large-scale web service based systems. However, since the model is quite general it has also been applied in other contexts, e.g., for building software platforms for the Internet of Things [9]. Thus, the basic ideas can be applied to OSS as well. Building appropriate data mining models over the complete data set might require some time and computational effort, so it is only natural to move it to the Batch Layer. Furthermore, applying the built models on the live data can be performed in memory for small batches of data (similar to the speed layer) or even for a continuous stream of data.

III. Project Life-cycle for Big Data Enabled Operator Support Systems

The FEE project takes a problem and an end-user-oriented approach towards the topic of Big Data enabled OSS. Workshops and interviews with industrial end-users and data analysts from the FEE-project development team[2] were an essential part of the work and have become building blocks of our suggested architecture process, which is described in Section V.

An important finding from our workshops is that we are able to describe the development of plant specific OSS functionality in a rather generic way. Figure 1 summarizes this description and shows the different phases that we encounter (and anticipate) in our efforts of developing Big Data based OSS.

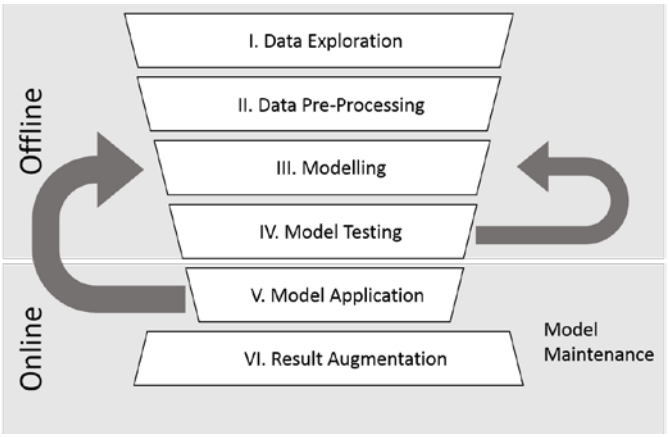

Fig. 1. Phases of a Big Data OSS Project

The project life-cycle adapts many valuable insights from the established CRISP-DM reference process [10] and adapts it to the specific needs of OSS. In particular, it separates the system life-cycle into two major phases: the offline and the online phase. The offline phase largely matches the steps in the CRISP-DM model, but takes on a stronger focus on how the analyst interacts with the data. The online phase basically replaces the CRISP-DM deployment phase and deals with the integration of the data-driven models generated during the offline phase into the operation of the process plant.

The conic shape of the diagram illustrates that the process works like a sieve on the rich and heterogeneous data available in process plants: during execution of the process, the data analyst starts from all the data available and narrows it down to the data relevant for the given problem and available for model application during the operation of the plant. The next subsections describe the phases in more detail.

A. Data Exploration

In this phase the data analyst reviews the available data (formats, data quality); guided by the problem definition he or she browses through the data including structured and unstructured data, analyzing different formats and specific data types.

[2] The FEE (Frühzeitige Erkennung und Entscheidungsunterstützung für kritische Situationen im Produktionsumfeld) project [2] consists of the development partners (ABB AG, the KDE Group and the MRT Group of the University of Kassel, RapidMiner GmbH and the PLT Group of the University of Dresden), as well as several industrial application partners.

B. Data Pre-Processing

In this phase data pre-processing methods [11] (e.g. for structured data: outlier detection and removal, noise reduction, replacement of missing values, feature generation; for unstructured data: term extraction, entity and concept resolution) are extended from small data samples to all relevant historical data as identified in the previous steps.

C. Modelling

In this phase, actual data mining and model building algorithms (e.g. classification (KNN, Decision Trees, etc.), subgroup discovery [12], or rule induction (FPGrowth)) are applied to the subset of historical data in order to derive a model suitable for the problem solution.

D. Model Testing

In this phase, the generated models are tested on (yet unused) historical data and evaluated. The performance on that test data set are measured by common evaluation metrics.

E. Model Application

In this phase, the model is connected to the live data and produces operator support outputs.

F. Result Augmentation

During the model application, the output of the model may be enriched by additional data, in order to create a better contextual understanding for the operator.

IV. The FEE Architecture Process

This section describes the process developed and followed in the FEE project, for defining the architecture for our different application scenarios.

A. Scenario Workshop and Operator Interviews

The goal of the problem analysis is to obtain a basic understanding of the user situation, and to lead to the right questions to be asked during the data analysis.

To identify application cases, a so-called scenario canvas has been developed that supported us in communicating with plant experts and enabled the systematic identification and documentation of any relevant information needed for further development. In the upper area of the canvas, the key information for the plant situation is captured (especially giving this situation a scenario name). In the left area, data is captured that is available to plant operators as information about the state of the plant (online data). The middle area of the canvas captures operator actions needed for reacting to or avoiding this situation. Below, any consequences are listed which become relevant when the situation occurs, or which would become relevant if the situation is not handled properly. This allows an approximation of the risk and extent of damage.

Overall, the canvas has a chronological order, from prior-situation, via situation occurrence and handing, to consequences. The right area serves capturing of context information, which are only indirectly related to the situation described and are therefore illustrated separately. Here, additional information sources and supportive tools are listed, which are available in the given risk situation and could be useful for avoiding or handling it, respectively. Hence, the canvas covers relevant information for describing a potential prediction problem, such as whether the situation could be avoided by the plant operator, which data seems promising for predicting this situation, or whether the consequences of a situation would justify the effort for developing a Big Data-based OSS function.

The workshop enabled the development team to get an understanding for the plant context and to identify an initial set of possible problems to tackle. In the next step, the findings of the workshop were confirmed by combined operator interviews and observations. Besides a confirmation of the already identified cases, new scenarios can be identified in this step. More importantly, the interviews and observations form the basis for deriving the specific user requirements and input for the design of user interfaces.

Finally, the identified scenarios are captured in a systematic but lean fashioned way so that the most relevant scenarios can be selected to proceed with. The description captures the following information:

- *As-is-situation*: Which users are involved, which tasks are they working on and how do they do it, what problems and obstacles do they encounter during the execution of the tasks?
- *Should-be-situation*: What OSS functionality can support in the tasks, what could be the desired new way of executing the tasks and are there special requirements that can be already observed?

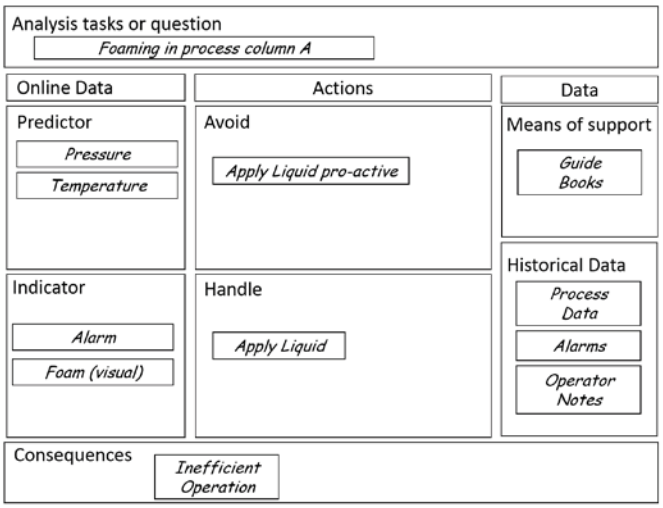

Fig. 2. Scenario Canvas

B. Requirements Workshops

Starting from the results of the scenario workshop and the operator interviews the requirements are refined in a subsequent workshop. The requirements workshops participants are user experience experts (involved in the previous workshops), data analysts and software developer or architects and if available end-users. The workshop consists

of four steps: review of the selected application scenarios, collection of detailed user stories in the scenarios, creation of paper prototypes and definition of the analytics workflow and collection of non-functional requirements.

C. *Identifying Big Forces and Technology Choices*

Designing a Big Data systems concerns the reasoning about trade-offs, such as the accessible or processable volumes of data, the resulting and acceptable latency, and the consistency of data. Specifically, we consider the following Big Data forces or data handling requirements:

- *Analytics Latency*: What is the level of latency acceptable for execution of user tasks (queries, execution of algorithms, or application of models)?
- *Consistency*: How important is considering all possible writes (especially the newest ones) in this step?
- *Volume accessible*: How large are the volumes of this data type that can be extracted or queried during this phase? All historic data, a pre-defined data set or data from within a certain and limited time-window?
- *Volume processable*: How large are the volumes of this data type on which actual computations have to be executed?
- *Update Latency*: How long does it take, until new data is considered and processed - if deterministic or probabilistic depends on the consistency requirement?

Choosing the right trade-off for an OSS implementation depends on two aspects: the life-cycle phase of the analytics project and the characteristics of the data that shall be accessed or processed. Figure 3 illustrates our approach to drive this decision in the context of a software architecture and to identify the right technologies.

Our starting point is the exploration of the mentioned Big Data forces for each project phase and typical data types encountered in a process plant. This will specify the detailed requirements for handling the data types in the corresponding life-cycle phase. Based on the identified requirements or Big Data forces it is possible to choose an appropriate compute and storage paradigm, addressing the Big Data forces in different ways. Compute and storage paradigms will link to different sets of specific technologies or software products and thus drive and support technology choices.

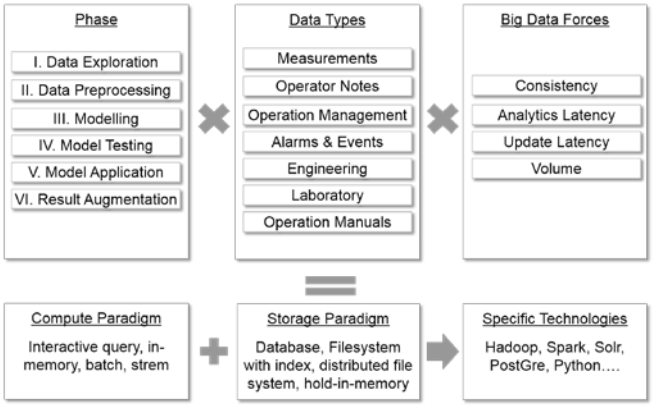

Fig. 3. Aspects in determining technology choices

Concerning the different project phases, the following compute paradigms are considered to be relevant: Interactive Query, In-Memory Processing, Batch Processing, Stream Processing. Furthermore, the following storage paradigms below are considered to be relevant: File system (FS), File System with index (FSi), Distributed File System (DFS), Database, RAM.

V. Example Technology Choices

In this section, we describe a high level system architecture along with possible technology choices for an *event prediction OSS* functionality, a generalization of the application scenario depicted in Figure 2. It represents a simplified version of the Lambda Architecture, tailored for prediction scenarios, featuring a batch and speed layer.

While systematically exploring the Big Data forces for each combination of data type and project phase the observation was made, that the project phase dominates the Big Data characteristics. Table 1 summarizes the observation based on the strongest requirement found in each phase and also gives the exception.

TABLE I
PROJECT PHASES AND BIG DATA FORCES

	Big Data Forces				
Phase	Analytics Latency	Update Latency	Consistency	Volume Accessible	Volume Processable
I.	Low	High	Secondary	Depending on data type	Depending on data type
II.	High	High	Secondary		
III.	High	High	Irrelevant[3]		
IV.	High	High	Irrelevant[3]		
V.	Low	Low[1]	Important		
VI.	Low	Low[2]	important[4]		

[1] For Operator Instructions and Maintenance Notifications: medium latency acceptable

[2] For Process Data and Operator notes eventual consistency acceptable

[3] Only working on well-defined model or test data set, no updates required

[4] For Process Data, Operator notes, Operator Instructions, Maintenance Notifications, medium latency is acceptable. For operator manuals high latency is acceptable

Following this observation, it was also possible to find compute and storage paradigms satisfying the requirement for all data types in each project phase. Table 2 summarizes the selected paradigms. In the data exploration phase (I.) the data analyst builds his understanding for the data, conducts first experiments and selects the data relevant for the next steps. This has to be supported by interactive queries for exploration and analytics, by in-memory processing, and the suitable storage options, which are databases or a file system with an index to support the queries. The pre-processing step (II.) processes larger amounts of data in a pre-configured standardized fashion. Batch computing supported by distributed file system is the choice to support this step. The model building (III.) works on a medium to large training data set in a batch fashion. Depending on the training data set

size, storage on a DFS or in RAM can be chosen. The model test (IV.) evaluates the model by using single samples from the test data set as input. The small amount of data can be processed in memory and possible hold in memory or some other data source. The model application (V.) has to work on the data streams produced by the process plant, perform pre-processing operations and calculate the appropriate input for applying the model. This happens best in a stream processing fashion and in the memory of each of the streaming nodes. The result augmentation (VI.) finally will pull appropriate data from various data sources to improve the operators understanding of the situation. Again, interactive query and the suitable storage paradigms haven been chosen.

TABLE II
PROJECT PHASES AND STORAGE AND COMPUTE PARADIGMS

PHASE	COMPUTE	STORE
I.	INTERACTIVE QUERY, IN-MEMORY ANALYTICS	DATABASE, FS
II.	BATCH	DFS
III.	BATCH	DFS, RAM
IV.	IN-MEMORY	FS, DATABASE, RAM
V.	STREAM	RAM
VI.	INTERACTIVE QUERY	DATABASE, FS WITH INDEX

Figure 4 shows a high-level component architecture for the prediction case. It is split into an offline phase where historical data is processed and an online phase where live data streams are analyzed. In addition to the batch processing system the offline processing component also includes a data exploration sub-system with a search engine for unstructured data and a database for structured data. With that, comprehensive exploratory approaches [13], process diagnostics, and interactive search can be supported for data analytics in the context of process plants.

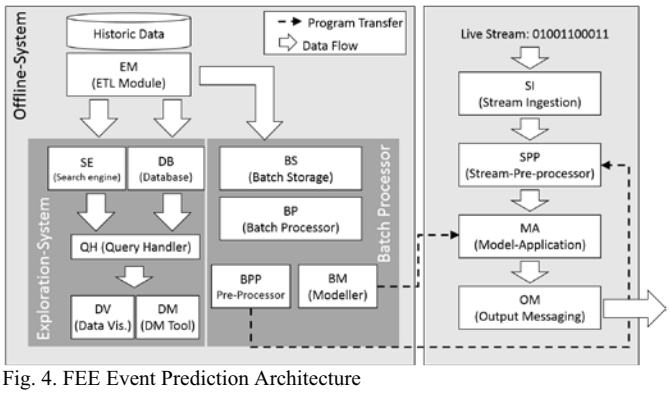

Fig. 4. FEE Event Prediction Architecture

VI. SUMMARY AND OUTLOOK

This contribution addressed the use of Big Data analytics and machine learning approaches for the realization of more predictive OSS in industrial plants. Based on current experiences in the FEE project we have described our approach how to define software architectures for such systems. As an example, we described a high-level system architecture with according technology choices for an event prediction *OSS* functionality. In the further course of the project, assistant functions will be developed as proof-of-concepts, where the architecture approach described herein shall be applied and further refined, as well as suited software technologies and tools to be evaluated or developed.

ACKNOWLEDGMENTS

The underlying project of this contribution was sponsored by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF), under the sponsorship reference number 01IS14006. The authors are responsible for the contents of this contribution.

REFERENCES

[1] McMillan, Gregory K. (2014) Tuning and control loop performance. Momentum Press
[2] McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D.J. Barton, D. (2012) Big Data. The management Revolution. Harvard Bus Rev 90(10), 61-67
[3] FEE Project. (2014). Project website http://www.fee-projekt.de Last accessed on 2016-02-02.
[4] Fan, W. and Bifet, A. (2013) Mining big data: current status, and forecast to the future. *SIGKDD Explor. Newsl.* 14, 2 (April 2013), 1-5
[5] Casado, R., and Younas, M. (2015), Emerging trends and technologies in big data processing. Concurrency Computat.: Pract. Exper., 27, 2078–2091
[6] Begoli, E. and Horey, J. (2012): Design principles for effective knowledge discovery from Big Data. Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, IEEE, pp. 215-217
[7] Marz, N. and Warren, J. 2015. Big Data: Principles and Best Practices of Scalable Realtime Data Systems (1st ed.). Manning Publications Co., Greenwich, CT, USA
[8] Dean, J. and Ghemawat, S. (2008) MapReduce: simplified data processing on large clusters. *Commun. ACM* 51(1), 107-113
[9] Atzmueller, M., Becker, M., Kibanov, M., Scholz, C., Doerfel, S., Hotho, A., Macek, B.-E., Mitzlaff, F., Mueller, J. and Stumme, G. (2014) Ubicon and its Applications for Ubiquitous Social Computing. New Review of Hypermedia and Multimedia, (20)1, 53—77
[10] Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. Journal of data warehousing, 5(4), 13-22.
[11] Atzmueller, M., Schmidt and A., Hollender, M. (2016) Data Preparation for Big Data Analytics: Methods & Experiences. Enterprise Big Data Engineering, Analytics, and Management, IGI Global, Hershey, PA, USA
[12] Atzmueller, M. (2015) Subgroup Discovery - Advanced Review. WIREs: Data Mining and Knowledge Discovery, (5)1:35—49
[13] Atzmueller, M. (2016) Advances in exploratory pattern analytics on ubiquitous data and social media. In: Stefan Michaelis, Nico Piatkowski, and Marco Stolpe (Eds.) Solving Large Scale Learning Tasks: Challenges and Algorithms. Springer, LNAI 9580