Theta Architecture: Preserving the Quality of Analytics in Data-Driven Systems

Vasileios Theodorou[1](✉), Ilias Gerostathopoulos[2], Sasan Amini[2], Riccardo Scandariato[3], Christian Prehofer[4], and Miroslaw Staron[3]

[1] Intracom SA Telecom Solutions, Athens, Greece
theovas@intracom-telecom.com
[2] Technische Universität München, Munich, Germany
Gerostat@in.tum.de, sasan.amini@tum.de
[3] University of Gothenburg, Gothenburg, Sweden
{riccardo.scandariato,Miroslaw.Staron}@cse.gu.se
[4] Fortiss GmbH, Munich, Germany
prehofer@fortiss.org

Abstract. With the recent advances in Big Data storage and processing, there is a real potential of data-driven software systems, i.e., systems that employ analysis of large amounts of data to inform their runtime decisions. However, for these decisions to be trustworthy and dependable, one needs to deal with the well-known challenges on the data analysis domain: data scarcity, low-quality of data available for analysis, low veracity of data and subsequent analysis results, data privacy constraints that hinder the analysis. A promising solution is to introduce flexibility in the data analytics part of the system enabling optimization at runtime of the algorithms and data streams based on the combination of veracity, privacy and scarcity in order to preserve the target level of quality of the data-driven decisions. In this paper, we investigate this solution by providing an adaptive reference architecture and illustrate its applicability with an example from the traffic management domain.

Keywords: Big Data · Reference architecture · Data veracity

1 Introduction

Modern systems collect raw data from the users and their personal devices (like health trackers), raw data from the environment and its smart objects (smart thermostats, home automation devices), as well as higher-level data coming from information providers like social platforms, open-data sites (e.g., OpenStreetMap), and other silos of information. Beyond functionality, the success of such systems is tied to the availability of the information that is processed as well as its quality. Functionality is often centered around the analysis of data to extract useful information (e.g. make user-specific recommendations, adapt to user habits to make an application more ergonomic, etc.).

M. Kirikova et al. (Eds.): ADBIS 2017, CCIS 767, pp. 186–198, 2017.
DOI: 10.1007/978-3-319-67162-8_19

We believe that we are moving from traditional software systems, which are functionality- and software-centric, towards systems that are data-centric and where the functionality is driven by the availability of data and the decisions drawn from them. This is true not only for systems that perform offline analysis (e.g. business intelligence systems), but also, and even more so, for systems that employ real-time analysis of data to inform their runtime decisions (personalized advertising, traffic control).

In this paper, we focus on systems that make decisions at runtime based on Big Data analytics. For these decisions to be trustworthy and dependable, one needs to deal though with the well-known challenges on the data analysis domain: data scarcity, low-quality of data available for analysis, low veracity of data and subsequent analysis results, data privacy constraints that hinder the analysis. Indeed, we argue that unless a data-centric system deals with the above issues effectively at runtime, it will not matter whether it can process terabytes or petabytes at very high rates (which, in itself, is a noteworthy achievement of current Big Data systems)—as the resulting decision recommendation cannot be trusted.

Promising solutions already exist in the data analysis domain, where several data cleaning and data preparation methods have been proposed [7]. While these are certainly relevant and important, this paper takes a different (yet complementary) approach and proposes to introduce flexibility to change data streams in runtime. This will allow the system architects to depart from the current, more rigid model where the data analytics schema is planned ahead of deployment (which data sources to use, which information to mine, which algorithms to use) and kept fixed afterwards, at run time. The main drawbacks of such systems are that (i) the quality of the decisions will decay over time in case the quality of the analyzed data drops, and (ii) the system cannot opportunistically adapt or even improve in light of new operating conditions (e.g., when a new, richer, better data source becomes available).

In this work we introduce a novel reference architecture for adaptive data-driven systems (called Theta architecture) that can change used data sources and data analysis algorithms at runtime to preserve the target quality of its outcome. To achieve our ambitious goal we have set up an agenda consisting of three items: (i) identify the need for adaptive Big Data analytics in the context of data-driven systems via concrete examples, (ii) propose a high level architecture for adaptive Big Data analytics in data-driven systems, (iii) evaluate our architecture via applying it in concrete systems identified in (i). In this paper, we present our first steps towards fulfilling (i) and (ii) from above. In particular, we identify and describe an example of a data-driven system where adaptive Big Data analytics are of high value. The example comes from the vehicular traffic management domain and is presented in Sect. 2. We present the Theta architecture in Sect. 3 and illustrate its use on the running example. Finally, after presenting an overview of the related work in Sect. 4, we provide the concluding remarks in Sect. 5.

2 Real-Time Freeway Traffic Control System

In this section, we present an example of a data-driven system to showcase that adapting the data analytics processes at runtime is beneficial. This is used for both motivating and exemplifying our approach.

The main objective of traffic management on freeways is to increase the freeway capacity, i.e. to maximize the vehicular flow per time unit:

$$flow\,(veh/h) = speed\,(km/h) * density\,(veh/km)$$

In order to increase the capacity of a freeway, existing control measures involve a combination of dynamically changing the speed limit, opening or closing freeway lanes and recommending alternative routes through in-car navigation devices and Variable Message Signs (VMS).

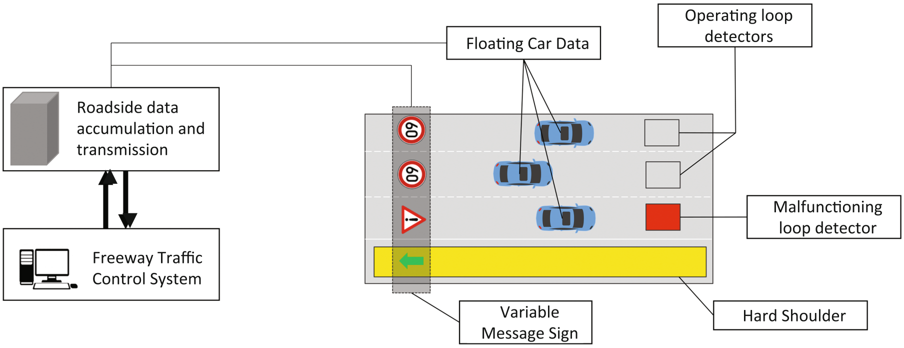
Floating Car Data
Operating loop detectors
Roadside data accumulation and transmission
Freeway Traffic Control System
Malfunctioning loop detector
Variable Message Sign
Hard Shoulder
60
60

Fig. 1. Data sources in real-time freeway control system.

To know when to make the decision of applying the control measures, a freeway traffic control system (FTCS) can rely on data coming from different sources (Fig. 1). Data sources include inductive loop detectors installed on the pavement, surveillance cameras, cars transmitting their position, speed, etc. (referred to as *floating car data* in the Intelligent Transport System domain), Doppler radars, and ultrasonic and passive infrared sensing devices. In addition, environmental detectors are commonly used for freeway traffic control such as scatter measurements—measuring visibility range—and precipitation detectors—measuring thickness of water film on the road. Data from these different sources are transmitted to a central traffic control center where they are analyzed (this typically involves also visualizing the data).

We envision a fully automated real-time FTCS. In this case, a human operator simply configures the FTCS upon startup, e.g. by setting the thresholds on calculated flow for opening and closing the hard shoulder. The FTCS performs the calculation of traffic flow in predefined time windows (e.g. every 30 s) and, based on these calculations, makes autonomous decisions to apply (a combination of) the available control measures. Applying the opening of a hard shoulder, for instance, results in changing the signs on the VMS gantries and disseminating the information about the opening/closing of a lane or a different speed limit to the in-vehicle navigation devices.

A challenge in order to achieve a fully automated FTCS is to be able to deal with non-veracious (untrue) data that may creep in the analysis and ultimately influence the control logic, which resides in the control center. Data can have low veracity (i.e., little correspondence to reality) due to several reasons, e.g., sensor inaccuracy, sensor faults, sensor tampering, communication errors, to name a few. For example, dirt could block the view of a camera or water spray from the passing vehicles may influence the visibility range at the location of the scatter measurements. A loop detector may start malfunctioning due to pavement cracking or moving, inadequate sealant application, electronics unit failure, breakdown of wire insulation, electrical surges, etc.

Ideally, FTCS should be able to detect that it is using non-veracious data and adapt its analytics logic by choosing different data sources or a different analytics algorithm (e.g., one that makes weaker assumptions on the veracity of incoming data).

3 Theta Architecture

We have already pointed out the need for introducing flexibility in the data analytics as part of our running example in Sect. 2. We believe this need extends to other systems that need to make correct and reliable real-time decisions based on large numbers of collected data. In this section, we propose a high level architecture to support this.

Our proposed architecture, termed *Theta* architecture, is depicted in Fig. 2. It consists of three main sub-systems: Business subsystem, Analytics subsystem and Adaptation subsystem. We describe each one in turn and exemplify them by applying them to our running example (Fig. 3).

Business Subsystem. This part represents any software-intensive cyber-physical system that needs to be controlled at runtime based on analysis of data collected from its execution, optionally enriched with external data.

Interfaces. *Business* communicates with the other two subsystems through two interfaces, the *Data & metadata* and the *Adaptation* interface. The first is used by the *Analytics* to collect data in a pull (e.g., polling a RESTfull API [26]) or a push (e.g., publishing Kafka [15] messages) fashion. Together with the actual data values (e.g., speed, precipitation, etc.), data points may also contain metadata such as reported accuracy of sensors, ownership, etc.

The second interface is used by the *Adaptation* for requesting the *Business* to adapt. It is the task of the system designer to define how adaptation requests from *Adaptation* are translated into actual changes in *Business*. We note that a change can affect the cyber part of the system (e.g., setting a parameter of a route planner software to a new value) or the physical one (e.g., switching a traffic light).

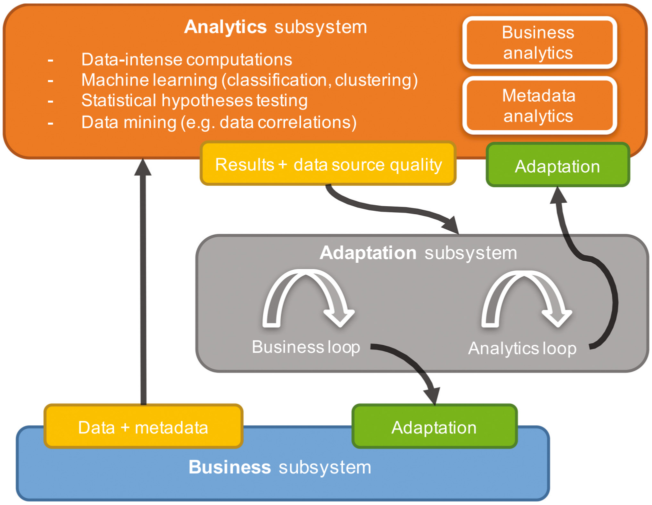
Analytics subsystem
- Data-intense computations
- Machine learning (classification, clustering)
- Statistical hypotheses testing
- Data mining (e.g. data correlations)
Business analytics
Metadata analytics
Results + data source quality
Adaptation
Adaptation subsystem
Business loop
Analytics loop
Data + metadata
Adaptation
Business subsystem

Fig. 2. Theta architecture.

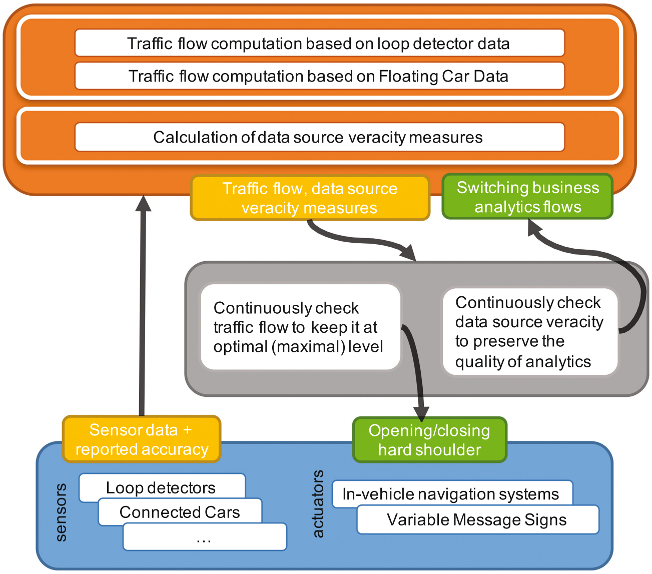
Traffic flow computation based on loop detector data
Traffic flow computation based on Floating Car Data
Calculation of data source veracity measures
Traffic flow, data source veracity measures
Switching business analytics flows
Continuously check traffic flow to keep it at optimal (maximal) level
Continuously check data source veracity to preserve the quality of analytics
Sensor data + reported accuracy
Opening/closing hard shoulder
sensors
Loop detectors
Connected Cars
...
actuators
In-vehicle navigation systems
Variable Message Signs

Fig. 3. Application of Theta architecture to the running example.

Example. In our running example (Fig. 3), *Business* is the traffic control system deployed in the freeway, consisting of a number of different types of sensors, e.g., inductive loop detectors, cars, cameras, radars, etc. Each sensor provides different data according to its type. For instance, a loop detector provides data on the number of vehicles that passed in a time interval, together with its average occupancy in the interval. A car provides its position and speed, while a camera provides either live feed or how many cars have been detected at a time interval. *Business* includes also two actuators, in-vehicle navigation systems and variable message signs, which are used to implement the "open/close hard shoulder" directive from *Adaptation*.

Analytics Subsystem. It is responsible for generating actionable information by large-scale data analysis. It should be able to provide different types of data analysis, from simple but data-intense computations to statistical hypothesis testing and machine learning and data mining.

In terms of the analytics processes that are run in *Analytics*, we distinguish between processes that support the data-driven functionality of the business system (business analytics) and processes that support the reasoning of the quality of input sources (metadata analytics). Metadata analytics not only use and summarize the metadata on provided accuracies of sensed data (e.g., from confidence interval values accompanying GPS measurements or weather data-based quality estimations on camera feeds), but also generate inferred metadata about the data sources based on the actual data values. Flexible formalisms can be employed for maintaining metadata about quality measurements, such as the Data Quality Vocabulary (DQV[1]).

Technically, *Analytics* can be implemented as a Big Data processing system following one of the existing lambda or kappa architectures or forks of them and the corresponding Big Data stacks (e.g., HDFS and Hive/Pig, Kafka and Spark/Flink).

Interfaces. *Analytics* exposes two interfaces for its interaction with the other two subsystems, the *Results & data source quality* and the *Adaptation* interface. The first interface is used by *Adaptation* in order to access the results of the business analytics processes, as well as to retrieve quality information about the available and utilized data sources. An interesting approach would be for the *Adaptation* to receive provided and inferred metadata about the data sources in an event-driven fashion, e.g., with notifications being triggered when veracity of some source is estimated to fall behind a predefined threshold.

The *Adaptation* interface is used by *Adaptation* for imposing adaptations on *Analytics*. The adaptations we currently consider take the form of switching the running business analytics processes and/or their input data sources. We also envision more fine-grained adaptations such as parameter configuration of the analytics algorithms embodied in the running processes.

Example. In our running example, *Analytics* comprises (a minimum of) three analytics processes. Two of them are business analytics; they compute the traffic

[1] https://www.w3.org/TR/vocab-dqv/.

flow based on different data sources and computations algorithms. The computation may involve also near-future prediction of the traffic flow (e.g., based on regression using historical data). A third one is metadata analytics; it provides an estimation of the veracity of data sources such as loop detectors or cars. In case of loop detectors, the veracity calculation process can involve performing some well-known plausibility checks about loop detector data from the traffic domain [13].

Adaptation Subsystem. It is the subsystem responsible for decision making. This is performed in two independent adaptation loops, the business and the analytics loop. The first is responsible for planning and executing adaptations on *Business* based on the business analytics results of *Analytics*. The second is responsible for planning and executing adaptations on *Analytics* based on the metadata analytics results of *Analytics*. To this end, data profiling [2] and metadata extraction [27] play a crucial role. A notable adaptation in the latter loop is the substitution of running business analytics processes with others that have either the same input data or different ones (a sensible option when the veracity of previously used input data gets high).

By making these two loops explicit in Theta architecture, we achieve separation between the concern of performing reliable and robust business analytics—by adapting *Analytics*—and using them to inform business decisions—by adapting *Business*).

Interfaces. *Adaptation* communicates with the other two subsystems with three interfaces (not depicted in Figs. 2 and 3). Via the first one, it receives the results of both the business and metadata analytics. The second and third ones are used to propagate the adaptation directives to both *Analytics* and *Business*.

Example. In our running example, the two loops in Adaptation take the following form. The business loop continuously checks the computed values of traffic flow (current values or near-future predictions) and issues the adaptation requests of opening or closing the hard shoulder in order to maximize the flow. In this case, decision making is based on well-known, empirically-validated rules on when to open/close a hard shoulder in a freeway based on the flow [11].

The analytics loop continuously checks the data source veracity measures calculated in *Analytics*. If the measures for the active business analytics processes indicate low veracity of input data, the analytics loop looks for alternative processes that can calculate the same result with different inputs to substitute the running processes. For example, the analytics loop switches the process that calculates the traffic flow based on loop detector data to an equivalent one that uses Floating Car Data, when a number of loop detectors start malfunctioning.

4 Overview of Related Work

Our approach builds upon existing works in Big Data analytics (relevant for the Analytics subsystem), self-adaptive systems (relevant for the Adaptation subsystem), and data veracity definition and management (relevant for the Metadata Analytics inside the Analytics subsystem and the Analytics loop inside the Adaptation subsystem), which we overview below.

4.1 Big Data Analytics

Analytics on Big Data refer to data analysis approaches able to scale to large amounts of data (e.g. petabytes or exabytes) which are typically unstructured or semi-structured [30], i.e. they do not follow a particular schema. Big Data are considered to encompass the "5-V" properties: volume, velocity, variety, veracity, and value [21]. Here, the value is of particular importance, since Big Data analytics is ideally concerned with deriving insightful information, or even actionable results, from raw data by applying techniques from statistics, machine learning, and data mining. Big Data analytics has been used in a diverse set of applications, from analyzing user clicks on websites to analyzing the results of high-energy physics experiments. It has also been used in the analysis of software artifacts such as source code, execution traces, and runtime logs (software analytics [23]).

On the tooling side, Hadoop [1] is an open-source ecosystem that has become the de facto standard. A distinction is typically made between analyzing historical data in batch mode (full-data analytics) and analyzing data as they come in stream mode ("tuple-at-at-time" analytics) [22]. Important tools in the first category are Hadoops MapReduce, Hive, and Pig; Flink and Kafka Streams are representative of the second category. Spark is a hybrid system that combines batch and stream processing. Stream processing in Spark relies on analysis of micro-batches, rather than on analyzing one tuple at a time. Spark comes also with extensive support in machine learning algorithms (its MLlib library).

When architecting a Big Data analytics system, two main approaches exist, namely lambda and kappa architecture. Lambda architecture [22] combines both batch and stream analytics in two layers called batch and speed layer, respectively. The results of the two layers are combined at a third layer, the serving layer (a NoSQL database). Essentially, stream processing complements batch processing by analyzing the data that comes in the system since the start of batch processing (which typically has high latency). An alternative to lambda is kappa architecture, where stream processing is used for both the batch and stream processing. This builds on the idea that stream processing, when applied for large windows that can fit in all the historical data, essentially corresponds to batch processing. We note here that both lambda and kappa architectures focus on solving performance issues (e.g. by balancing throughput and latency), and not on quality issues of data and data analysis results—which is our focus.

4.2 Self-adaptive Systems

Self-adaptation refers to the ability of a system to change its structure and/or behavior at runtime in response to external stimuli and changes in internal state. Self-adaptation is typically achieved in three fundamental ways: (i) by relying on a detailed application model, e.g., Markov Decision Processes [12], and employing simulations or other means of state-space traversal to infer the best response of the system, (ii) by identifying control parameters and employing feedback-based control techniques from control theory [8], and (iii) by reconfiguring architecture models, typically with the help of Event-Condition-Action rules [10].

Self-adaptation techniques typically follow the MAPE-K loop [14], which divides self-adaptation into four phases: *Monitoring* activities, *Analyzing* runtime metrics, *Planning* strategies, and *Execute* planall based on a shared *Knowledge* base. Self-adaptation strategies are expressed as actions involving particular architecture reconfigurations; they are applicable under certain conditions in the presence of certain events or situations [4]. Actions can be associated with the satisfaction of one or more system goals, typically quantified via fitness or utility functions [28].

Although adapting a system based on analysis of data collected from its execution is a well-researched idea, there is a vacuum of approaches that use Big Data analytics in self-adaptation. In our recent work, we have proposed an approach to do so [29], since we believe that as the amount of data collected from a running system and its environment increases, Big Data analytics become relevant for self-adaptation.

In our approach, we employ two self-adaptation loops: one that controls the data-driven system itself (by employing Big Data techniques in its Analysis phase), and one that controls the analytics subsystem (by switching data sources and analysis algorithms).

4.3 Data Veracity

Historically, the notion of veracity is derived from the area of sociology and its major popularity lies in the area of criminology—the ability to detect whether a witness is veracious or not [17,20]. In that particular context, the term veracity is used both in relation to actors (e.g. witnesses) and their statements [3]. The latter refers to judging the truthfulness of a statement and is in scope for our purposes as well.

In our context, we consider the definition of veracity as quoted by Krotofil [16] who defines the veracity as *the property that an assertion truthfully reects the aspect it makes a statement about*. We can see a direct relation to the field of criminology and also see the challenges related to automated assessment of the veracity in the context of software systems.

For instance, the veracity of the data can be violated by: (i) non-adequate measurement of a physical property by a sensor because of the inappropriate design of the sensor (ii) non-adequate measurement caused by a faulty sensor during the operation (iii) non-adequate measurement caused by an obstructed sensor (iv) faulty data caused by a malicious agents tempering with the sensor data.

Lukainova and Rubin recognize data veracity as the sum of a number of quality attributes such as correctness, accuracy, free from biases and free from errors [19]. Based on this work and our previous work [31], we can see that veracity can be modelled as a composition of elements. To be able to perform such modelling, we need to first decide upon which elements are relevant (e.g. correctness, free or errors) and how they relate to each other.

In our approach, we first need to model and assess the veracity of the data that is being used in Big Data analytics In our running example, a rudimentary way to do is to apply existing plausibility checks on the sensor readings (such checks are e.g. well-known for inductive loop detector readings). We then propose to switch between different data sources and corresponding data analysis algorithms (i.e. Big Data jobs) at runtime when the veracity of the used data drops below a threshold. In other words, we intend to use simple threshold-based adaptation rules to adapt the analytics subsystem in order to increase the quality of its results.

4.4 Data Source Evaluation

Data source selection has received interest since the advent of the Internet. Florescu et al. [9] use probabilistic knowledge on a mediation schema that quantitatively determines the probability of information being covered by specific data source and provide corresponding data source ranking in cases of overlapping data sources (i.e., sources containing same documents). Naumann et al. [25] additionally put into play data source quality and propose a methodology that weights different data sources with regards to their information quality, considering quantified quality criteria (e.g., relevance to specific query) and cost and thus formulating linear programs. Mihaila et al. [24] introduce the use of metadata to maintain in XML format content and data quality information about data sources and by relaxing accuracy requirements, they propose a methodology for efficient source selection and ranking.

More recently, Dong et al. [6] have dealt with the problem of selecting the subset of data sources that maximize quality gain and minimize cost. In their work, they showcase the peculiar behavior of information gain as a result of utilizing multiple data sources of varying information coverage and accuracy. This work poses a pragmatic view on data source selection and can provide a stepping stone for conducting analysis on integrating data sources in presence of errors and false values.

Examining the quality of web sources, Dong et al. [5] use an information extraction system to employ aside from exogenous signals, inference about probability of correctness of facts which they define as accuracy of a web source. They introduce a novel methodology for assessing source and extracted data correctness, which can pave the way for veracity inference in case of multiple available data sources. Finally, data source quality assessment approaches have been proposed [18,32] that can deal with high variety and variability of available sources.

In our approach, we combine explicit metadata derived from the Business subsystem with statistical methods, to determine the veracity of data sources and to detect anomalies. This analysis provides feedback for decisions on data source selection and switching, which aim at maximizing veracity while minimizing cost.

5 Discussion

In this section, we conclude by providing a discussion that reflects on our proposed architecture:

1. Essentially, if we disregard *Analytics* from Theta architecture, the resulting architecture degenerates into the classic MAPE-K loop, comprised of a controlled subsystem (*Business*) and a controller (*Adaptation*).
2. Both the business and the analytics loops in *Adaptation* can be arbitrarily complex. In our first attempts based on Apache Kafka and Python for traffic management of a simulated freeway, we have successfully considered only simple adaptation logics (e.g. based on a short number of rules); however, Theta architecture imposes no restrictions to the complexity of the adaptation logics.
3. Cluster-based approaches at the Analytics subsystem are only necessary if data size is large enough to render single-node approaches impractical.
4. We note here that although data veracity is the primary concern in our running example, our architecture can be used for adapting between data sources based on other concerns such as data privacy and confidentiality.

References

1. Apache Hadoop (2017). http://hadoop.apache.org/
2. Abedjan, Z., Golab, L., Naumann, F.: Data profiling: a tutorial. In: Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD 2017, pp. 1747–1751 (2017)
3. Carey, P.W., Mehler, J., Bever, T.G.: Judging the veracity of ambiguous sentences. J. Verbal Learn. Verb. Behav. **9**(2), 243–254 (1970)
4. Cheng, S.W., Garlan, D., Schmerl, B.: Stitch: a language for architecture-based self-adaptation. J. Syst. Softw. **85**(12), 1–38 (2012)
5. Dong, X.L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., Zhang, W.: Knowledge-based trust: estimating the trustworthiness of web sources. Proc. VLDB Endow. **8**(9), 938–949 (2015)
6. Dong, X.L., Saha, B., Srivastava, D.: Less is more: selecting sources wisely for integration. In: Proceedings of the 39th International Conference on Very Large Data Bases, PVLDB 2013, pp. 37–48. VLDB Endowment (2013)
7. Dustdar, S., Pichler, R., Savenkov, V., Truong, H.L.: Quality-aware service-oriented data integration: requirements, state of the art and open challenges. SIGMOD Rec. **41**(1), 11–19 (2012)
8. Filieri, A., et al.: Software engineering meets control theory. In: Proceedings of SEAMS 2015, pp. 71–82. IEEE, May 2015
9. Florescu, D., Koller, D., Levy, A.Y.: Using probabilistic information in data integration. In: Proceedings of the 23rd International Conference on Very Large Data Bases, VLDB 1997, Athens, Greece, pp. 216–225, 25–29 August 1997
10. Garlan, D., Cheng, S.W., Huang, A.C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. Computer **37**(10), 46–54 (2004)

11. Geistefeldt, J.: Operational experience with temporary hard shoulder running in Germany. Transp. Res. Rec. J. Transp. Res. Board **2278**(6), 67–73 (2012)
12. Ghezzi, C., Pinto, L.S., Spoletini, P., Tamburrelli, G.: Managing non-functional uncertainty via model-driven adaptivity. In: Proceedings of ICSE 2013, pp. 33–42. IEEE (2013)
13. Gladbach, B.: Bundesanstalt fr Straenwesen: Merkblatt fr die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen (MARZ). Technical report, Ausgabe 1999 (1999)
14. Kephart, J., Chess, D.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
15. Kreps, J., Narkhede, N., Rao, J., et al: Kafka: a distributed messaging system for log processing. In: Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB 2011), pp. 1–7 (2011)
16. Krotofil, M., Larsen, J., Gollmann, D.: The process matters. In: Proceedings of the 10th ACM Symposium on Information Computer and Communications Security. Association for Computing Machinery (ACM) (2015)
17. Levine, T.R., Park, H.S., McCornack, S.A.: Accuracy in detecting truths and lies: documenting the "veracity effect". Commun. Monogr. **66**(2), 125–144 (1999)
18. Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., Han, J.: Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 1187–1198. ACM (2014)
19. Lukoianova, T., Rubin, V.L.: Veracity roadmap: is Big Data objective, truthful and credible? (2014)
20. Mann, S., Vrij, A.: Police officers' judgements of veracity tenseness, cognitive load and attempted behavioural control in real-life police interviews. Psychol. Crime Law **12**(3), 307–319 (2006)
21. Marr, B.: Big Data: the 5 vs. everyone must know. https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know
22. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Realtime Data Systems, 1st edn. Manning Publications Co., Greenwich (2015)
23. Menzies, T., Zimmermann, T.: Software analytics: so what? IEEE Softw. **30**(4), 31–37 (2013)
24. Mihaila, G.A., Raschid, L., Vidal, M.: Using quality of data metadata for source selection and ranking. In: Proceedings of the Third International Workshop on the Web and Databases, pp. 93–98 (2000)
25. Naumann, F., Freytag, J.C., Spiliopoulou, M.: Quality driven source selection using data envelope analysis. In: Third Conference on Information Quality (IQ 1998), pp. 137–152 (1998)
26. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. "Big" web services: making the right architectural decision. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, pp. 805–814. ACM, New York (2008)
27. Quix, C., Hai, R., Vatov, I.: Metadata extraction and management in data lakes with GEMMS. CSIMQ **9**, 67–83 (2016)
28. Salehie, M., Tahvildari, L.: Self-adaptive software: landscape and research challenges. ACM Trans. Auton. Adapti. Syst. **4**(2), 1–40 (2009)

29. Schmid, S., Gerostathopoulos, I., Prehofer, C., Bures, T.: Self-adaptation based on big data analytics: a model problem and tool. In: Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2017), pp. 102–108. IEEE Press, Piscataway (2017). https://doi.org/10.1109/SEAMS.2017.20
30. Srinivasa, S., Bhatnagar, V. (eds.): BDA 2012. LNCS, vol. 7678. Springer, Heidelberg (2012)
31. Staron, M., Scandariato, R.: Data veracity in intelligent transportation systems: the slippery road warning scenario. In: 2016 IEEE Intelligent Vehicles Symposium (IV), pp. 821–826. IEEE (2016)
32. Zhang, Y., Wang, H., Gao, H., Li, J.: Efficient accuracy evaluation for multi-modal sensed data. J. Comb. Optim. **32**(4), 1068–1088 (2016)