How To RAMI 4.0: Towards An Agent-based Information Management Architecture

Andreas Kirmse, Vadim Kraus,
Tristan Langer, Andre´ Pomp
Institute of Information Management
in Mechanical Engineering
RWTH Aachen University
Dennewartstr. 27, 52068 Aachen
{andreas.kirmse, vadim.kraus,
tristan.langer, andre.pomp}@ima.rwth-aachen.de

Prof. Dr.-Ing. Tobias Meisen
Chair of Technologies and Management
of Digital Transformation
University of Wuppertal
Rainer-Gruenter-Str. 21, 42119 Wuppertal
meisen@uni-wuppertal.de

***Abstract*—With the latest advances in digitalization and Industry 4.0, the manufacturing industry is collecting more and more production data. However, with the increasing interconnection of machines, not only the volume but also the variety of data is being expanded. The data life cycles of collection, processing, combining, analyzing and feeding new findings back into sources are becoming increasingly challenging for data scientists to complete. Reference architectures such as the RAMI 4.0 provide conceptual guidelines to address these problems. In this paper, we focus on the implementation of an agent-based architecture that is in line with RAMI 4.0. This architecture implements the guidelines provided by RAMI 4.0 by applying modern approaches from the areas of data lake based data acquisition, semantic description, look up and processing as well as information utilization.**

***Index Terms*—Data acquisition, Data integration, Information management, Multi-agent systems, Big Data applications, Industry applications, RAMI 4.0**

I. Introduction

With the emerging of Industry 4.0, more and more manufacturing systems are being upgraded to enable intelligent manufacturing. Sensors are installed to map the state of physical systems into the digital world and machines are interconnected to derive optimization potential for production from the data flow [31]. Initial attempts of bridging the gap between the physical and digital domain, include the implantation of custom solutions for collecting, storing and analyzing data. For instance, companies started to centralize data collection from manufacturing systems by creating data warehouses to have a single structured data source for business analysis. Data warehouses use a schema-on-write approach that requires every data source to be processed and organized into one or multiple predefined data schemes. Therefore, using data warehouses as a versatile general purpose storage system requires complex processing and structural adaptation before new data sources are available. However, in industrial production, which is driven by the fear of losing hidden insights from shop floor data, all data sources are currently being identified as potentially useful and, therefore, are preemptively recorded [8]. Due to the increasing number of machines, the amount and also variety of data increases immensely. Consequently, using data warehouses became increasingly costly and the use of data lakes has emerged as an suitable storage and staging layer. Data lakes, in contrast to warehouses, store all data without modification, which provides cheaper storage and integration of new data sources. However, analyzing the data in a data lake to derive hidden insights is more complicated than with data warehouses, because a data analyst must first find and understand the stored data. Hence, companies started again to implement and introduce tools that facilitate data analytics for Industry 4.0 data lake architectures.

These trends show that the manufacturing industry is trying to achieve the goals of Industry 4.0 by implementing solutions for the different aspects of collecting, storing, integrating, discovering and analyzing data, leaving uncertainty about the individual implementation as well as their interoperation. To overcome this issue, different Industry 4.0 reference architectures were proposed, providing guidelines on how to implement overall solutions for bridging the physical and digital domain. For instance, these architectures need to support the collection of appropriate data sets and provide feedback on data sets to the associated data provider in order to adapt to the requirements of the analysis use case [14]. One of these architectures is the Reference Architectural Model Industry 4.0 (RAMI 4.0) [30]. However, since RAMI 4.0 is only an abstract framework, there is a need for more concrete architectures that show how to implement data lakes for the data analytics use case within Industry 4.0 applications.

In this paper, we introduce an approach for the implementation of a RAMI 4.0 conform architecture dealing with connecting industrial data sources to a data lake storage system and using the data for analytic use cases. We consider semantic enrichment by adding semantic models during the integration phase from ingestion agents and user input. The usage of data considers semantic extraction by corresponding agents, providing semantic and structural transformation tasks. The combined use of the agents is utilized in an extended data analytic cycle providing applications with a close integration to business processes.

II. RAMI 4.0 and Implementation Challenges

In this section, we explain the RAMI 4.0 reference architecture based on its layered structure and derive implementation challenges for each layer.

RAMI 4.0 refers to the service oriented Reference Architectural Model Industry 4.0, which defines hierarchy levels for organizing the digitalization of industrial components [30]. In its three-dimensional representation it is based on the OSI software layer model [7], the automation pyramid [11] as hierarchy levels and the value stream of a product.

Figure 1 shows the layered visualization of the OSI-based software model together with a depiction of the components in the automation pyramid scheme. The bottom **asset** layer describes physical components like the actual product, equipment or machinery in the real world. The **integration** layer bridges real and digital world and contains virtual representations of the assets. The next layer of **communication** interconnects diverse systems and digital representation on network protocol level over switches, in order to enable data access. In the **information layer**, description and identification are added to relevant data. The **functional layer** describes and enables control of the asset itself. The last layer on top, is corresponding to the organizational and operational processes of the **business**, similar to an ERP system.

The levels are extended with the product, which is the uniquely tailored good for the customer and thereby characterizes one goal of Industry 4.0 with lot size one. Additionally, the connected world is included on the upper end of the automation pyramid. On top of it, the RAMI 4.0 introduces the concept of administrative shells that encapsulate an asset into the digital world.

In the third dimension of the life cycle of a product and value stream definition the product is in the focus. Here the development and production stages of a product are considered together with the required maintenance usage.

There are several challenges hindering the RAMI 4.0 conform implementation of a data driven analytic use case. Major hindrances are related to the *accessibility*, *discoverability* and dataset *interoperability* of data [22].

The data *accessibility* challenge relates to physical restrictions when accessing industrial data sources. These data sources are generally distributed along the vertical hierarchy of the automation pyramid. On the lowest levels, data is produced by machines at the highest granularity. In an industrial setting the flawless operation of these machines on the shop-floor is critical. Therefore, access to these devices is usually strictly restricted. However, from an analytic point of view, these fine-grained data often provide the most detailed insights into the process. Which then again leads to the problem that all amount of data produced by devices need to be stored and transmitted. Therefore, limited network capacities impose a restriction to the access of data. This challenge is located on the layers of asset and integration, where the digitalization in RAMI 4.0 of the real device occurs, but also on the communication layer in order to enable the interconnection.

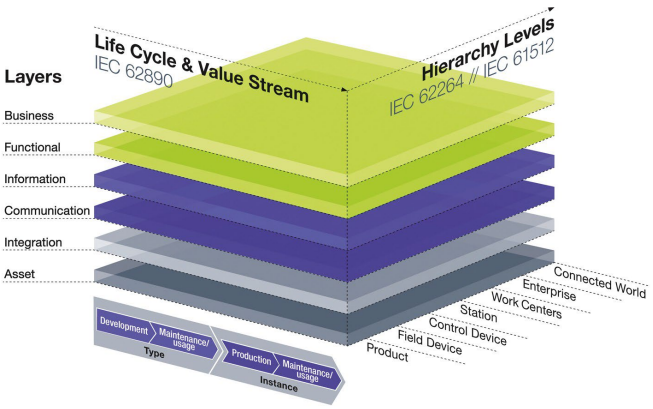

Fig. 1. The RAMI 4.0 reference architecture picture [30], showing the three dimensions of layers, life cycle & value stream as well as hierarchy levels.

Independence of the solution for the accessibility challenge the *discoverability* or findability of data sources is another challenge. Even if all sources are accessible, it is still unclear what kind of information they contain. This means that it is a challenge for a data scientist to find data related to the use case if it is stored in a data lake alongside the raw data of all production data.Analytic use cases are often driven by the cognitive abilities of data scientists. They have the necessary expertise to understand the process to be optimized and have a general understanding of relevant criteria, but it is often unclear from a technical point of view where to find suitable use case related data. This identified problem mainly resides at the information layer in RAMI 4.0, which only defines the relevant data to be present, but gives no clues about how to act about this. After being able to access data sources and being able to describe and understand them, the final challenge of data analytics is related to the data set incompatibilities. For example, there could be two machines that perform the same production process but generate data that cannot be transformed in a consistent unified format. Consequently, the underlying goal of all approaches is to achieve interoperability between different data sources. Interoperability can be be summarized as the possibility to utilize resources from heterogeneous resources in unison, despite their technical, syntactical, semantic or organizational differences [9], [29]. Looking again at the RAMI 4.0 layers this is directly correlated to the functional layer.

In summary there are three major challenges when designing an architecture for Industry 4.0 scenarios that build on data lakes. The first challenge is to enable technical interoperability to achieve *accessibility* even to fine granular machine data. The second challenge is to implement a method that supports *discoverability* of use case related data by data scientists and the last challenge is to deal with *data set interoperability* to allow data scientists to query data in a consistent semantic and syntax.

III. State of the Art

In the past, a number of high-level architectures have been proposed to meet the aforementioned challenges [14]. One such high-level architecture is the so called 'big data pipeline' published in [3]. The authors describe a serial process of multiple phases which are necessary steps to enable the analysis of data.The pipelines give readers a basic overview on how to generally extract value from big data, but does not describe how to link the various phases in the pipeline, nor how to implement the contents of the individual phases.

Similar to RAMI 4.0, the "Industrial Data Space" introduces a five-layer structure [26]. It focuses mainly on the description of different roles within one "data ecosystem". Each role having certain functions, depending on the layer. E.g., authorization of data usage is a task for the data owner in the functional layer. This reference architecture establishes roles and assigns responsibilities in the data space on a higher level, but does not deal with findability.

Furthermore, Mohsen et al. [16] present a critical review of the different reference architectures for smart manufacturing including RAMI 4.0, IIRA, IBM Industry 4.0 and NIST Smart manufacturing. Their expert based interview approach comes to the conclusion that there is a lack of micro-service definitions in the current architecture designs and that none define a proper ready-to-use implementation. However, they also state that due to the similarities all references describe an interchangeable idea, leading to our decision on using the RAMI 4.0 as representative in this paper.

Lastly, the Internet of Things (IoT) is the extension and definition of bringing the Internet to everything and embedding it into sensors and devices all around [12]. Therefore, even the specialized Industrial Internet of Things (IIoT) is conceptually similar, but there are technical and organizational barriers, making the industrial application of IoT approaches, unfeasible.

As all of these references and guidelines do not yield a direct usable nor implemented solution that tackles all the aforementioned three challenges. Hence, in the following we take a closer look at specific solutions that are state-of-the-art with regards to the three identified challenges of data acquisition, findability and data usage.

Data Acquisition : Theorin et al. [28] proclaims the Line Information System Architecture (LISA) that uses the idea of an Enterprise Service Bus (ESB) to reduce point-to-point connections in a traditional client/server approach by making use of service mediation techniques. They claim to have made the service oriented architecture principle of ESB together with an event-driven bus system industrially applicable and scalable based on ActiveMQ. LISA uses an own message format to in-cooperate source systems and thereby solve the homogenization aspect.

Kirmse et al. [13] describe an approach with a lightweight architectural framework and integration chain capable of abstracting the specifics of individual data source systems including legacy devices in the manufacturing domain. It deals with decoupling of different network zones as well as security levels by enforcing a message queue based technique all realized in open source technologies.

Bonci et al. [5] show a database-centric approach based on cyber-physical production systems. The idea to use RDBMS along with the SQL query language is quiet established; their novel approach however focuses on lightweight database synchronization through distributed replication on every CPS device. Furthermore, they add the swarmlet concept by facilitating the publish/subscribe paradigm for IoT devices and add a plug-in structure, which extends the central database to a service-oriented architecture similar to an Enterprise Service Bus. The general principal followed in architectures which enable the combined use of distributed entities in the IoT context can be summarized as an indirection architecture. An indirection approach consists of a mediator that tries to enable the bridging of interoperability. Such a mediator is often also called middleware. Razzaque et al. [23] present a survey about different concepts and implementations of such middleware systems in the IoT context. As as conclusion they state, that semantic and syntactic interoperability is the most lacking in current systems, with syntactic interoperability being the most challenging. Furthermore, agent systems are introduced as part of the IoT ecosystem, where the concept of proactive handling and communicating is depicted.

Data Discoverability: The challenge of finding the right data is often tackled by systems dealing with meta data management. Meta data, generally defined as data about data, can be any additional data and knowledge providing further insight [15]. Various approaches exists to deal with meta data. The simplest form is a data catalog containing an item, e.g., a data source and a set of tags, i.e., keywords [25]. The discoverability is facilitated by providing a search interface that allows to search for catalog entries based on the tags. Therein lies the problem with this approach, if an entry is not tagged extensively enough, i.e., a tag is missing, the entry cannot be found. Furthermore, word relations, such as synonym, hypernym (generalization) and hyponyms (specification) are not considered. Thus an extension to the tag system is the use of a semantic modelling approach, which especially includes these word relations and thereby allows much deeper understanding and discoverability even when not using the exact term. . Pomp et al. [21] present an integration approach using a user-centric dynamic modelling approach, ESKAPE. The approach is focused on establishing a common semantic understanding, i.e., a knowledge graph, while preserving the individual user semantics. Also in [22] the impact of such an approach on the reduction of the time-to-analytics was discussed. Strassner et al. [27] present a concept similar to ESKAPE focusing on semantic interoperability. They concentrate on the IoT and thus base their methods on a greenfield mentality. In result, such concepts often are not usable for industrial applications and also provide no room for a transitory state where new "smarter" devices and legacy hardware can operate side-by-side.

Data Processing: The use of semantically adaptable applications is researched extensively in the context of web applications, summarized under the terms of semantic web and linked open data. Approaches in this context are centered around the use of ontologies to describe web services using the Web Ontology Language (OWL) and performing reasoning on them using SPARQL [4]. Barder [2] presents a concept for self-governing tasks, i.e., detecting changes to their inputs and switching to more adequate data sources. Phillip et al. [19] describe a decentralized combination of services for a multi-step analytic task, where semantic interchangeability for substeps is known beforehand. The applicability beyond web services is currently unclear, i.e., if and how these approaches would cope in a setting where the processing tasks are not limited to web technologies (HTTP, REST, JSON).

Application: Finally, we take a look at related work that deals with the user's view of big data analysis scenarios in order to deduce which factors are relevant for the application layer. Elgendy and Elragal [6] developed a big data, analytics and decision framework that guides analysts through four layers to support decision making in big data applications. The *intelligence* layer deals with data acquisition, discovery and preparation. The second layer is called *design* and consists of model planning and data analytics. It presents analytic tools to the user to perform analysis tasks and generate insights from the data set. The third layer, *choice*, integrates business tasks into the framework e.g. visualization and reporting. The top layer is called *implementation* and deals with process monitoring.

Sacha et al. [24] contribute a conceptual visual analytics and machine learning pipeline. They define several steps that cover typical analysis interaction options and also provide examples of how to support the interactions. Therefore, defining pipelines seem to be a convenient tool to perform all steps of a data analytics cycle.

Kinjo et al. [10] developed a cloud-based next-generation sequencing big data analysis platform. The system consists of four subsystems: Job Management System, Data Management System, Pipeline Management and Genome Explorer. To use the system, there is a graphical user interface that allows users to create new projects. In a project, the user can upload new data files and configure analysis pipelines based on a data file. Configured analysis pipelines can then be started or rerun, for example if the data was updated. The analysis progress can be tracked by the user and in the end a HTML report will be created.

IV. Agent-based Big Data Analysis Architecture

We discussed challenges for data lake based Industry 4.0 architectures in Section II and showed some existing solutions for those challenges as well as reference architectures in Section III. We introduce a new data lake based architecture that combines some of the existing solutions and extends them to fit to the RAMI 4.0 reference architecture to meet Industry 4.0 standards. The architecture consists of four layers (see Figure 2) that allow data providers to offer data in a consistent way and make it available to data scientists. Furthermore, we consider a standardized application layer to integrate the data analysis into business activities.

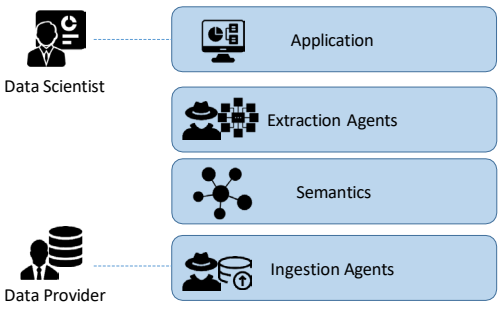

Fig. 2. Components and roles of the proposed layered agent-based big data analysis architecture.

The lowest layer is served by data providers. Data providers provide data through one ingestion agent per data set. An ingestion agent itself contains information about the provided data that enable the semantic layer to reference the data based on its semantic context. The semantic layer stores and manages all information of the referenced data sets. It processes data queries from the top layers executed by data scientists and suggest matching data sets for a specific use case. To merge those data sets into a consistent format and structure, the third layer implement extraction agents. Extraction agents provide a single endpoint to access a collection of data sets in a format that is defined by the data scientist. Data scientists interact with the application layer at the top of the architecture. We define a standard analysis cycle that utilizes data management from the other layers and embeds the analysis into business activities.

A. Semantic Data Ingestion

The semantic data ingestion process consists of modular source specific agents that are responsible for the data collection. Here diverse data source system are connected via a specific agent to the semantic integration system.

In context of RAMI 4.0 hierarchy layers the agent implements the 'administrative shell' of the assets in order to extract data and communicate them to the semantic ingestion system to get into information that are usable for analytic applications. Each agent is responsible for encapsulating protocols and machinery specific connections. They rely on a common framework that enforces same behavior and are independent by using modules, specific for each protocol. The agents behave alike to the semantic data platform and provide same functionality of acquiring data, regardless of the data's dynamic.

A data source can be a continuous stream as well as a finite batch set that also can be updated at definitive time intervals. In the following, we give two examples. In the first case, we look at a traditional relational database (RDBMS) as batch source, while for the second case we look at a machine connected via OPC UA that provides a continuous stream of changing values. Each agent collects meta information about the data

source system, in order to combine a unique fingerprint:*source system information, data syntax* This fingerprint is used to uniquely identify the data source stream, in order to effectively re-recognize same input sources.

Each agent is responsible for the data transfer and therefore has to incorporate a buffer or queue system, when required. Especially, in circumstances where the source system has no persistence of data, it is the responsibility of the specific agent to assure data completeness. In case of source systems that store and hold data for a longer period of time on their own, this mechanism is optional. Databases are a typical example of source systems that maintain a longer ”history” of data that can still be retrieved at a later time (to some extend). But also systems like OPC UA, provide a history server which can yield such a functionality, so that the agent can rely on these functions to retrieve data even when the target system is not reachable or capable of persisting incoming data streams.

In order for the agent to connect to the data source, it requires physical access to the machine at hand. The agent therefore resides either directly and physically at the machinery or has access to it, meaning a firewall exemption rule.

Agent configuration parameters necessary for connecting into a source system and describing it with meta information: *connection information, authentication, location information, Time-To-Live(TTL), owner, trigger, whitelist*

The data source connection can either be a hostname or the IP address with the respective port number for a TCP/IP connection, but also a Uniform Resource Identifier (URI) which denotes the protocol.

The location information denotes the originating geo location where the agent is collecting the information from and thus where the data is generated. This spatial information is not only limited to the location in the production process, but hints at the geographic location in order to help with synchronization issues when having to deal with different time zones.

Time-to-Live (TTL) is a suggested liveness time of the data, regarding how long they are considered valid, but also in concern to archival periods, which are required by some regulations e.g. for warranty or security reasons.

The owner information not only denotes a human contact point that is responsible for the source system, but also acts a the liable entity for the data ownership.

The trigger information regards the way of data retrieval from the data source system. The most generic differentiation is between push and pull principal. In a stream scenario the source system pushes new data directly to the agent. This might even achieved with the help of a pub-sub mechanism, so that the agent subscribes directly at the source system for changes. In case of OPC UA there is the differentiation between interval and change trigger for this subscription based approach. The interval is a time based continuous polling-like data acquisition which is initiated by the server itself. The change trigger defines a threshold value by which the specified value has to change in order to be send out to the agent. In RDBMS systems, a polling mechanism to detect data changes has to be facilitated. Change Data Capture techniques exist in order to detect changes in a RDBMS. A simpler mechanism can use identification fields, such as primary key, if only new entries are added into a table or field that denote their last change, e.g. an updating timestamp on modification.

Last configuration option is a whitelist of available and wanted data points depending on the source system; for RDBMS systems this is the schema and/or table, for OPC UA the node names (or sub-tree), for message broker systems such as MQTT, Kafka or AMQP the topic names.

Providing a new data source: The following denotes the steps required to connect a new data source, after the initial configuration of the ingestion agent. Upton startup the agent establishes a connection to the data source. The agent examines the system, determines its fingerprint and checks if it is already known. If the system was not known, based on the whitelist the agent extracts a data sample and defines a data syntax description model. Here each data field is provided with the name, the actual data type as defined in the source system and some example values. This information is transmitted to semantics layer as part of a registration. In this layer the data provider is now able to semantically model the data source, defining the concepts and their relations used in the data source. During this time the agent is in a pending state.

After the semantic model is created by the data provider, which is determined by a status API, the agent begins sending of actual single data points. Here a common unified data exchange format, that enables syntactic description is required. One example could be based on Apache Avro [1], which provides this capability. Future scenario includes that each agent has a unique signature identifier, so that their authorization and authentication can be acknowledged over general identification with public key infrastructure or there-like. Thereby, it enables the scenario of a simple plug-and-produce factory setup.

For reduced complexity and current approach, changing and existing data sources are considered as new data sources.

B. Data Extraction

In the previous section, we focused on data access and acquisition. This step included the use of ingestion agents to get a syntactic model of a data source and the definition of a semantic model. The semantic model contains a mapping of the data points to concepts of a knowledge graph, describing their meaning [21]. Therefore, at this point data is stored semantically enriched and is available to be consumed or semantically extracted. We define semantic data extraction in our approach, therefore, by:

- **Semantic Look-up** Finding data sources.
- **Semantic Transformation & Processing:** Transforming between semantically related concepts and processing of data points based on their semantic properties.
- **Semantic presentation** Presenting the result in an application specific semantic and format.

Semantic Look up From a usage perspective, the look up of required data sources is always the first step. In our approach this this search includes a *semantic look up* of data sources

based on concepts used in the semantic model of a data source. Additionally, the search considers semantic similarity. For example, the knowledge graph constructed via ESKAPE contains synonyms and related concepts based on external information sources [18], [20].

Semantic transformation defines the usage of data points based on their semantics. One typical application in this setting is the transformation between related concepts based on their measurement unit. For example, consider data sets containing distance measurements. One data set contains measurements in the English unit mile, while another contains measurements in the SI unit meters. A typical scenario for a semantic transformation would be: A user makes a request for *Distances Measured in Kilometer* the knowledge graph contains the knowledge that *Distance* can be *Measured in Miles, Kilometer, Meter, ….* Based on this knowledge and the request a semantic homogenization of the two data sets will be performed automatically (without the knowledge of the user), i.e., convert miles and meters to kilometers. This use case can also be seen as an extension of the look up functionality in terms of semantic similarity. Searching for *distance* would yield both data sets. The extension would now be the additional information about the transformation ability of units across different measurement systems.

Semantic reasoning & processing goes a step further, instead of concerning with singular concepts and their immediate semantic neighborhood, the combination of different concepts and a potentially reasoning between them is required for semantic processing. Consider following example: The previous data sets contain next to the distance measurements another measurement, a duration. Also for this example, the duration was measured in different units of time, e.g., seconds and minutes. A scenario for semantic processing would be: A user makes a request for *Velocity measured in km/h*. None of the data sets contains this concept explicitly. A semantic reasoning must be performed to identify that *Velocity* can be determined by combining *Distance* and *Duration*. An application based on this premise would not care about the specifics of the actual data set, e.g. *Miles* vs *Kilometers*, as they can be transformed into the required semantics, as explained before. This reasoning will be performed by comparing the semantic models of different data sets in terms of alignment with local (i.e. similarity of the individual models) and global similarity (i.e. similarity when considering additional concepts and relations of the knowledge graph).

Semantic presentation The final component of the extraction is related to the presentation of the data, i.e., the output format and protocol. Making use of a template engine, e.g., mustache [17] would allow user specific output formats. Applications might require the data to be presented in a specific protocol. For example, the result might be needed as a HTTP/REST-based service producing content in a JSON format, while a process control logic might require a specific PLC protocol and format.

The whole process, i.e., selecting and retrieving the data, which was collected by the ingestion agents, performing a

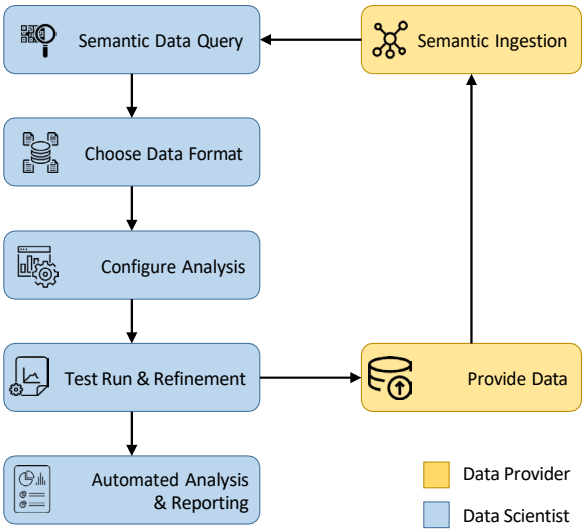

Fig. 3. The usage cycle in the application layer.

semantic homogenization and processing can be implemented by a flexible micro service approach. Each of the steps are defined as processing task, implementing a processing programming interface, defining a semantic function. Such a function takes a set of input semantic models and produces another semantic model. Each of such tasks needs to be stored in a processing task repository. Based on the user request our approach will look up required transformations implicitly or present an unfulfilled gap in the processing chain. Since we follow a bottom up approach, i.e., starting with a clean slate, most of these transformations will be unknown at the initial setup of the system. The fulfillment of such gaps can be incited by a per-use reimbursement. The task repository might contain deployable code or references to external resources, which provide the execution capabilities. For achievable requests a complete pipeline will be defined. The system will create an extraction agent, which will take the pipeline, orchestrate it and maintain its operation. Scenarios of cloud, edge or mist-computing approaches are feasible here, i.e., individual steps of the pipeline might run in different locations, for example, because of computing or latency requirements. The extraction agent also acts as a continuous communication channel for the application layer, providing data, notifying about changing conditions in the sources systems, adapting to these changes or to new requirements of an analytic use case or application.

C. Application Layer

The tools and methods to search, select and analyze data as well as create reports are implemented in the application layer. We define an extend data analytics cycle that should be supported by tools implemented in the application layer to utilize the underlying layers to optimally support the most common tasks of a data scientist. The extended cycle is based on the analytics cycle presented in Section I and related work

presented in Section III and allows close integration with presented agents and the overlying business activities.

Figure 3 gives an overview of the usage cycle in the application layer. The first step is the *semantic use case specific querying of data*. In this step, data scientists can view all suggested data sets that fit their use case. Corresponding queries are processed by the semantic management of the data records. Data scientists will then evaluate the list of data sets found, select matching sets, and determine their preferred target format and structure to configure the extraction agent for the selected data set.

Then, the *exploratory analysis* begins. The application layer provides machine learning algorithm that are used in the respective domain and allows their configuration (e.g. DBScan for clustering with configurable parameters *epsilon* and *min samples*). The actual algorithm needs to be implemented in a big data processing engine to handle large data sets. The extracting agent will provide an endpoint for the engine to query all extracted data.

The application layer also needs to provide tools for visualizing the data and the results of the configured analyses to perform test runs and refine automatic analyses. If an analysis provides valuable insights, the analysis and the associated visualization can be saved for a report. The report configuration can be implemented by presenting slide layouts, images from automated analyses and text field to add comments. In addition, problems with the data sets or requests for supplementary data records can be passed to data providers. The extracting agent then distributes the requests to the suitable data providers via the semantic layer and data providers can react by providing updated data sets through semantic ingestion.

In the final step, the data scientist can set up the automatic generation of further reports for the aggregated analyses. This allows, for example, to observe effects of initiated process improvements or quality parameters to be continuously monitored. There is also a need for an overview of active automated analyses and the status of associated analysis jobs that are processed in the big data processing engine.

V. Discussion

In Section II we presented several challenges, in this chapter we are going to discuss how the presented approach can help to solve these problems. Additionally, we will show that our approach is compatible with current industrial reference guidelines. As noted in Section III, these references are effectively interchangeable, we chose the RAMI 4.0 [30] due to its importance in the context of Industry 4.0. Therefore, the concepts of this paper and the discussion apply to any of the presented references. Specifically, we will inspect the architectural dimension of the RAMI 4.0. The architectural dimension consists of six layers, starting with the physical world represented by the *Asset* layer, the layers on top of it are concerned with the digital aspects. Our architecture operates from the second layer of the RAMI 4.0 onwards, therefore, is located in the digital world.

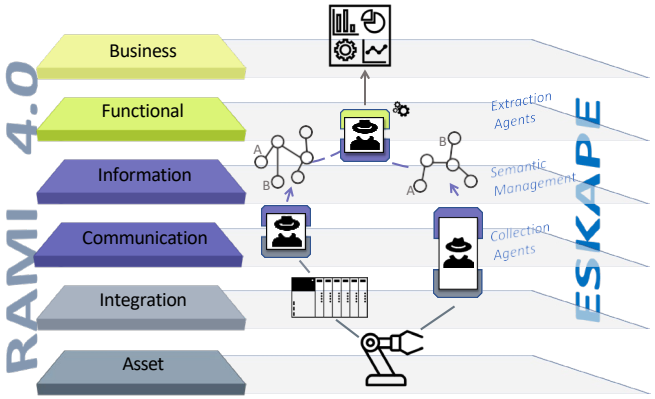

Fig. 4. Comparison of the presented architecture in relation to the RAMI 4.0 reference architecture, showing how agents are encapsulated by shells, mediating based on semantic information modelling.

Accessibility: The first challenge we described was related to accessibility. This challenge starts to manifest in the *Asset* layer, which contains physical devices, for example a manufacturing robot. In our architecture we rely on the existence, e.g., a digital interface. These interfaces are, however, located on the next layer, *Integration*, building an immediate bridge between physical and digital world. For example, a PLC provides a digital interface to the robot. Conceptually we do not limit our ingestion agents to operate on this level, i.e, be part of the digitization of an analog process. For example, consider physical paper reports, our ingestion agent could be implemented as a device that scans these reports and, therefore, provides a digital interface to the physical entity. Most of our ingestion agents are, however, located on the next layer, *Communication*. The *Communication* layer of the RAMI 4.0 describes the problem of technical interoperability, e.g., different communication media and protocols. The major benefit of our ingestion agents is related to their adaptability when dealing with different protocols and interfaces.

Discoverability: The next problem manifests in the layers above the previous ones, i.e. the *Information* layer. The RAMI 4.0 is presenting a concept called 'administrative shell'. This shell is conceptually supposed to act as a mediator or translator between layers and their respective entities. In our architecture, a technical shell and a semantic shell effectively surround our ingestion agents. For example, in Figure 4 the lower two agents are encapsulated by a lower shell responsible for data access, while the upper shell provides the semantics needed to describe the accessed data on the *Information* layer. Our approach, specifically, therefore, provides the possibility to semantically describe assets and their data in order to find, manage and use them based on their semantic.

Data set Interoperability: The next challenge is related to usage of data, it starts to manifest most prominently from the *Functional* layer on wards. In our architecture, we envision the extraction agents to act as the bridge between the *Information*

layer and the *Functional* layer. The information shell contains semantic look up and semantic transformation capabilities providing consistent semantics, while the functional shell allows for application specific (functional) access to assets. Implications for the 'Business' layer are the result of adaptive semantic applications described in Section IV-C. In summary, our approach is an implementation that is compatible with the second to last layer of the RAMI 4.0 reference architecture.

VI. Conclusion

In this paper we presented how the challenges of data *accessibility*, *discoverability* and *data set interoperability* manifest when implementing a concept like the Reference Architectural Model Industry 4.0 (RAMI 4.0). We described a realization of the concept based on an implementation using an agent-based approach. The approach is centered around a data lake system, considering the heterogeneity of industrial data sources and assets, by providing mechanisms to semantically enrich these sources, describing their contents and formats during the ingestion stage. For the utilization of the data, we described how the data could be accessed and processed semantically by self-adapting extraction agents, providing flexible interfaces for data driven applications.

Lastly, we discussed how the presented approach is considering the challenges arising for data driven applications in regard to the layers defined by the RAMI 4.0.

Future work involves the realization of the conceptual approaches of processing data semantically and adaptively. We plan to evaluate our implementation within real-world scenarios.

Acknowledgment

The authors would like to thank the German Research Foundation DFG for the kind support within the Cluster of Excellence Internet of Production (IoP). Project-ID: 390621612

References

[1] Apache Software Foundation. Apache Avro, 2009. https://avro.apache.org - last accessed: 2019-04-18.

[2] Sebastian R Bader. Automating the dynamic interactions of self-governed components in distributed architectures. In *European Semantic Web Conference*, pages 173–183. Springer, 2017.

[3] Elisa Bertino, Philip Bernstein, Divyakant Agrawal, Susan Davidson, Umeshwas Dayal, Michael Franklin, Johannes Gehrke, Laura Haas, Alon Halevy, Jiawei Han, and Hosagrahar Visvesvaraya Jadadish. Challenges and opportunities with big data. *Whitepaper*, 2011.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global, 2011.

[5] Andrea Bonci, Massimiliano Pirani, and Sauro Longhi. A database-centric approach for the modeling, simulation and control of cyber-physical systems in the factory of the future. *IFAC-PapersOnLine*, 49(12):249–254, 2016.

[6] Nada Elgendy and Ahmed Elragal. Big data analytics in support of the decision making process. *Procedia Computer Science*, 100, 2016.

[7] International Organization for Standardization. ISO 7498. *Information processing systems – Open Systems Interconnection – Basic Reference Model*, 1983.

[8] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137 – 144, 2015.

[9] IERC. IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps. *EUROPEAN RESEARCH CLUSTER ON THE INTERNET OF THINGS*, 2015.

[10] Junichi Imoto, Norikazu Kitamura, Sonoko Kinjo, Kazuho Ikeo, Norikazu Monma, Sadahiko Misu, Kazutoshi Yoshitake, and Takashi Gojobori. Maser: one-stop platform for NGS big data from analysis to visualization. *Database*, 2018, 04 2018.

[11] ISO/TC 184/SC 5 Interoperability, integration, and architectures for enterprise systems and automation applications. IEC 62264-1:2013: Enterprise-control system integration – Part 1: Models and terminology. *ISO*, 2013.

[12] Sabina Jeschke, Christian Brecher, Tobias Meisen, Denis O¨ zdemir, and Tim Eschert. Industrial internet of things and cyber manufacturing systems. In *Industrial Internet of Things*, pages 3–19. Springer, 2017.

[13] Andreas Kirmse, Vadim Kraus, Max Hoffmann, and Tobias Meisen. An architecture for efficient integration and harmonization of heterogeneous, distributed data sources enabling big data analytics. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS,*, pages 175–182. INSTICC, SciTePress, 2018.

[14] Andreas Kirmse, Felix Kuschicke, and Max Hoffmann. Industrial big data: From data to information to actions. In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS*. INSTICC, SciTePress, 2019.

[15] David Marco. *Building and Managing the Meta Data Repository: A Full Lifecycle Guide*. Wiley & Sons, 2000.

[16] Mohsen Moghaddam, Marissa N. Cadavid, C. Robert Kenley, and Abhijit V. Deshmukh. Reference architectures for smart manufacturing: A critical review. *Journal of Manufacturing Systems*, 49, 2018.

[17] Mustache. Mustache.github.io, 2019. https://mustache.github.io -last accessed: 2019-04-18.

[18] Alexander Paulus, Andre´ Pomp, Lucian Poth, Johannes Lipp, and Tobias Meisen. Gathering and combining semantic concepts from multiple knowledge bases. In *ICEIS (1)*, pages 69–80, 2018.

[19] Patrick Philipp, Achim Rettinger, and Maria Maleshkova. On automating decentralized multi-step service combination. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 736–743. IEEE, 2017.

[20] Andre´ Pomp, Johannes Lipp, and Tobias Meisen. You are missing a concept! enhancing ontology-based data access with evolving ontologies. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 98–105. IEEE, 2019.

[21] Andre´ Pomp, Alexander Paulus, Sabina Jeschke, and Tobias Meisen. Eskape: Information platform for enabling semantic data processing. In *ICEIS (2)*, pages 644–655, 2017.

[22] Andre´ Pomp, Alexander Paulus, Andreas Kirmse, Vadim Kraus, and Tobias Meisen. Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures. *Technologies*, 6(3), 2018.

[23] Mohammad Abdur Razzaque, Marija Milojevic-Jevric, Andrei Palade, and Siobha´n Cla. Middleware for internet of things: A survey. *IEEE Internet of Things Journal*, 3(1):70–95, 2016.

[24] Dominik Sacha, Michael Sedlmair, Leishi Zhang, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268:164–175, 2017.

[25] Gurmeet Singh, Shishir Bharathi, Ann Chervenak, Ewa Deelman, Carl Kesselman, Mary Manohar, Sonal Patil, and Laura Pearlman. A metadata catalog service for data intensive applications. In *SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, pages 33–33. IEEE, 2003.

[26] Fraunhofer Society. Reference architecture model for the industrial data space. *Whitepaper*, 2017.

[27] John Strassner and Wael William Diab. A semantic interoperability architecture for Internet of Things data sharing and computing. *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, 2017.

[28] Alfred Theorin, Kristofer Bengtsson, Julien Provost, Michael Lieder, Charlotta Johnsson, Thomas Lundholm, and Bengt Lennartson. An event-driven manufacturing information system architecture. *IFAC-PapersOnLine*, 48(3):547–554, 2015.

[29] Hans Van Der Veer and Anthony Wiles. Achieving Technical Interoperability. *European Telecommunications Standards Institute*, 2008.

[30] VDI/VDE Society Measurement and Automatic Control (GMA). Reference architecture model industrie 4.0 (RAMI4.0). *Whitepaper*, 2015.

[31] Ray Y. Zhong, Xun Xu, Eberhard Klotz, and Stephen T. Newman. Intelligent manufacturing in the context of industry 4.0: A review. *Engineering*, 3(5):616 – 630, 2017.