KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 5 (2019) 141–145

www.elsevier.com/locate/icte

A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management

George Kousiouris[a,*], Stylianos Tsarsitalidis[a], Evangelos Psomakelis[a], Stavros Koloniaris[a], Cleopatra Bardaki[b], Konstantinos Tserpes[a], Mara Nikolaidou[a], Dimosthenis Anagnostopoulos[a]

[a] *Department of Informatics And Telematics, Harokopio University of Athens, Omirou 9 Str., Athens, Greece*
[b] *ELTRUN E-Business Research Center, Athens University of Economics and Business, Evelpidon 47 and Lefkados 33 Str., Athens, Greece*

Received 5 April 2019; accepted 9 April 2019
Available online 17 April 2019

Abstract

With the usage of technologies like the Internet of Things and Cloud, spanning across the spectrum of manufacturing, production and distribution, typical supply chain management (SCM) systems rely on a multitude of services for support. The aim of this paper is to present an integrated system microservice architecture and implementation, consuming and semantically annotating data feeds from online systems, while performing post-processing tasks such as supply chain monitoring, goods location analysis and estimated delays. Suitable application code programmed through intuitive workflow-programming based structures is also injected while the approach is validated through a set of deployment scenarios.

Keywords: Internet of Things; Supply chain management; Cloud computing; Semantics; Microservices

1. Introduction

With the advent of the Internet of Things (IoT), new opportunities and capabilities emerge with relation to the real time monitoring, management and optimization of goods distribution and supply chains. Gartner includes Artificial Intelligence (AI), IoT and predictive/prescriptive analytics among the most important supply chain technology trends for 2018 [1]. New capabilities include Cloud based services, software as a service product management systems that may enforce monitoring of Things (instances of these products) through smart sensors embedded on the products, able to be scanned and provide information such as temperature, humidity, and location [2], while enabling the definition of conditional rules for alerting the stakeholders. Combinations of technologies including smart sensing, communications and centralized, cloud based analysis have been designed for tackling issues related to distribution as well as monitoring of the state of sensitive goods in industrial and agricultural use cases [3,4]. Data centricity is also a key feature [5]. Through these solutions, companies may reduce the lack of sufficient information on what is occurring in their complex supply chains and thus optimize risk assessment and sustainable supply chain management strategy [6].

However, from a practical point of view these combinations are very challenging due to the diversity of the involved IT systems and the technical inability to merge and reason on the provided data in a semantically meaningful manner. While globalization has aided in increasing the competition between vendors and should ensure better supply conditions for companies, the increased complexity, adaptation needs and risk associated with managing a global supply chain may negate these benefits. Agility achieved by process and information integration is among the largest contributors in optimizing supply chain management [7], as well as incorporation of new technologies and adaptation to the factory needs and process [2].

The aim of this paper is to present a microservice based system architecture and implementation (namely the AffectUs framework) (Section 3) that targets at easily integrating and deploying (through a single click deployment process)

* Corresponding author.
E-mail address: gkousiou@hua.gr (G. Kousiouris).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2019.04.002

various critical elements of a supply chain management system in an integrated fashion (Section 4). System elements include external IoT management platforms for baseline data feeds, semantic services (enriched with specialized ontologies, decoupled however from the end user) aiming at annotating incoming data and Artificial intelligence mechanisms for detecting abnormal conditions in the annotated feeds. Section 5 presents measurements performed on its operation while Section 6 concludes the paper.

2. Related work

Related work for defining system architectures for complex Supply Chain Management (SCM) systems appears in [8], especially for incorporating IoT principles and supply chain tracking. The rationale of using field data to populate data stores on which information can be gathered and processed is similar in the context of the supply chain and product tracking, while also recognizing the need for relevant alerts to the involved stakeholders with relation to the specific products. However in this case the identification of notified users is performed only through the static registration to a common publish/subscribe system and not dynamically, based on identified dependencies through the use of semantics as is the case of AffectUs. In [9], a reference architecture is presented for food supply chain management. Key innovative aspects include event intelligence, formulated via manually inserted rules and predictive analytics for more supply chain and object automation. The approach is dictating also a distributed nature, taking under consideration the Electronic Product Code Information Services (EPCIS) for information sharing between participants of the supply chain, indicating that such information sharing is critical. The AffectUs approach follows the aforementioned set of architectural and functional recommendations, including an abstract way of managing systems automation. In [10], a Labview prototype of a system for modeling similar stages of the supply chain to the ones defined in this paper is introduced, especially for the food domain. The value of using enhanced Big Data related architectural solutions in the context of Supply Chain Management (SCM) is argued in [11], which is also in accordance to the structure of the work in this paper, including NoSQL datastores and a way of parallel computation (in our case through the MongoDB queries), as well as visualization components and ETL processes for linking supply chain data (in our case performed through the semantic annotations and structures).

3. System model and design

The system structure appears in Fig. 1. It comprises three main blocks, the Node-RED middleware, implementing the UI and coordination/adaptation block, and the back end components, implementing subsystems such as the Semantic service and the Data Management/AI subsystem. There is also a vertical link between all system elements through the use of RabbitMQ as a messaging layer, where non-REST based interactions are needed (such as asynchronous notifications to listening users for violations, data feeds, etc.).

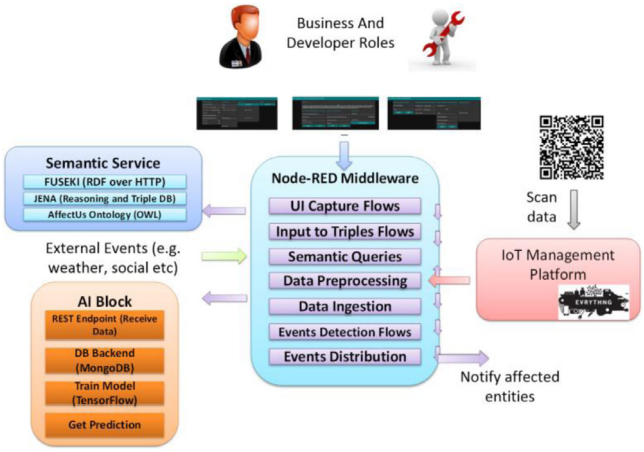

Fig. 1. Overall system architecture and blocks.

3.1. Node-RED layer

Node-RED is a flow programming tool that enables the creation of a variety of features, including web services, UIs, adaptation and middleware logic. In the context of AffectUs it is used for the following purposes:

- As a UI implementation layer that incorporates graphical interfaces through which the various actors can declare usage of products, dependencies as well as instantiate the specific elements of their supply chain (such as locations and types of stages). They also insert practical information needed, such as API keys for linking to external IoT platforms.
- As a plugin mechanism to existing IoT management platforms (such as EVRYTHNG [12]) in order to retrieve products (through client methods for their respective APIs) as well as notification feeds from these products based on scan data from smart sensors in a streaming fashion.
- A middleware and integration layer through which the respective modifications and adaptations from one data source to another may be performed (e.g. transformation of UI declarations to semantic triples for feeding in the respective service) as well as implementation of application specific logic (e.g. rules based on which anomalies will be detected and propagated) in a user friendly flow programming based model. Furthermore, it uses the Semantic service in order to enrich incoming data feeds with supply chain related annotations and forward these data in the Data Management block for usage in the AI process.

3.2. Semantic service block

The second major block is the Semantic service. This incorporates the AffectUs ontology that correlates concepts from the supply chain such as stages, types of locations

and relationships among actors. Through the application of these links the incoming data feeds from the products can be mapped to semantic concepts of the supply chain and dependencies from other actors or objects can be determined. More information on the structure of the AffectUs Ontology can be found in [13]. The semantic framework is built on three fronts: The Knowledge Base which uses the OWL ontology for inference and reasoning, the SPARQL 1.1 entailed queries and the integration with programmatic rules on the Node-RED side as well as additional custom rules for the Jena reasoner in the Knowledge Base (KB). The architecture is service-oriented, therefore the KB is actually a Semantic Web Service. For this reason the Apache Jena Fuseki Server is used, which serves RDF data over HTTP, using SPARQL to communicate knowledge in RDF form.

3.3. Artificial intelligence block

The third major block consists of an Artificial Intelligence (based on Tensorflow) and Data Storage (based on MongoDB) service, that aims to infer about key metrics of each stage (such as average rest times at a given location) and detect anomalies from the examination of annotated historical data. To this end it stores forwarded annotated data feeds regarding the entry and departure of products in the respective chain stages (product ID, thing ID, chain stage type, day, month and time). Following, one can utilize the enriched input data in order to extract prediction models based on historical data for transition timings in each supply chain stage. This model (based on a DNNRegressor deep neural network) needs to create a prediction for the expected duration that a thing (product instance or batch) should stay in a specific stage of the chain. Then and based on the real time data, one can detect deviations from the normal behavior and therefore trigger early notifications to the involved entities. Prediction parameters may also be enriched with data coming from external notifications (such as Large Crowd Concentration events detected from social networks [14], traffic and weather information, etc.). The AffectUs AI framework is based on Tensorflows, using Python 3.

4. System implementation

All elements of the system have been encapsulated as Docker images in order to enable their easy management, update and deployment processes. Docker deployment can be based either on individual containers or grouped together in a combination of elements forming the overall service. The latter is more beneficial since it enables a one click deployment process following the existence of a relevant Docker compose file. This file describes numerous aspects, among which virtual networks in which all services join, virtual IPs (for automatic discovery between services), ports of the services, sequence with which they are started (if dependencies exist), startup commands and installations, number of desired instances, etc. Furthermore, Docker Swarm was used as a cluster management system for load balancing between the nodes for service deployment.

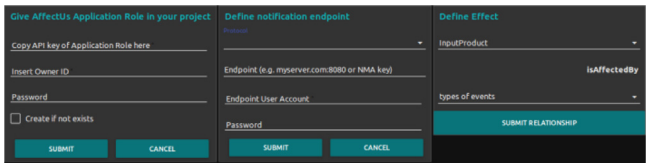

Fig. 2. UI for declaring products from EVRYTHNG.

Implementation of the User Interfaces for linking with external platforms such as EVRYTHNG and selecting products to be incorporated is performed in Node-RED. One of the key aspects of the framework is its ability to transform these user friendly declarations in the necessary triples for inclusion in the Ontological KB. Therefore, the logic behind the UI (Fig. 2) is to retrieve this information from the user, then link to the platform to retrieve product characteristics and finally create internally all the necessary triples that declare the various features. Through this approach it completely decouples all the semantic complexity from the end user. An indicative flow in Node-RED for achieving this appears in Fig. 3. To create the semantic communication between all needed flows (such as the UI captured and generated ones) and the Semantic web service for inclusion of the triples, a connector subflow is created which enables querying and processing the KB at will. It needs to be noted that there is no Node-RED node readily available to accomplish that. SPARQL nodes that currently exist can only make static SELECT queries. To make the query formulation dynamic we pass a query template and the dynamic parameters. The formulation as well as the evaluation of these queries is done directly from JSON, with the help of the SPARQL.js library. The query is then performed over HTTP and the response is processed.

5. System deployment and testing

In order to deploy the system, a single command is needed to launch the aforementioned Docker compose file. Following, containerized service elements are launched and made available in the various available nodes of the Docker Swarm (Fig. 5). Initially, one aspect that needs to be investigated is the time needed to spin up the overall service cluster in Docker. This is directly related either to aspects such as service continuity and/or systems management for the packaged solution. In order to measure this time delay, one needs to take into account not only the container start time, which is documented via the container state (“Started” status) and uptime statistics, but also the time needed to launch the applications inside the container and until the respective endpoints are indeed responsive. For this reason, the “Healthcheck” feature of Docker is used, in which one can directly dictate the way Docker may validate the fact that the application inside the container is running (e.g. via an HTTP GET method at the respective endpoint, via a command executed inside the container, etc.).

For monitoring of the status of the chain, relevant UIs have also been created (Fig. 4).

The initialization measurement includes the timestamp at launch and a continuous sampling up to the time all service

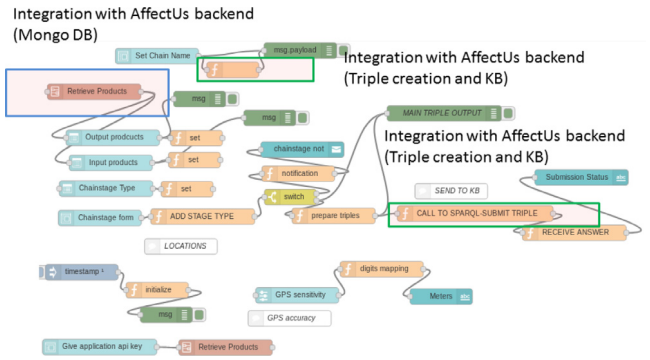

Fig. 3. UI Node-RED flow for incorporating programming logic to transform inserted information to semantic triples.

Fig. 4. Indicative UI for product delays across the chain.

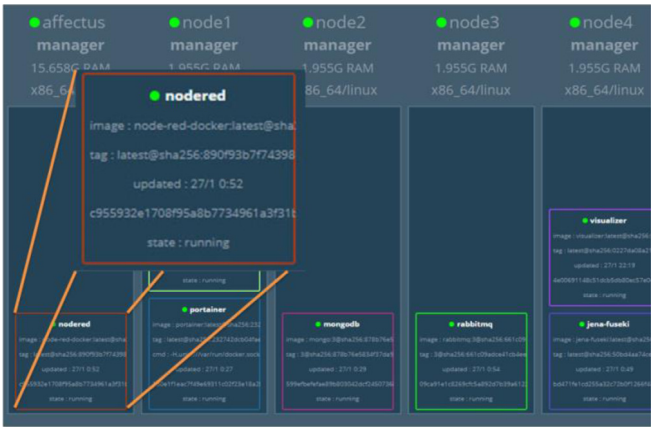

Fig. 5. Monitoring of the Swarm deployed services.

elements are considered healthy (i.e. endpoints reachable). The overall process is repeated 100 times in order to extract more accurate results (PDFs presented in Fig. 6). From the aforementioned results one can conclude that the overall spin-up time of 41 s is dependent on the RabbitMQ element, since in all cases this was the latest to start, given that its distribution does not overlap with any other case. Another conclusion is that the most stable ones are Python, MongoDB, and Jena containers, as indicated by the taller and narrower PDFs of Fig. 6. On the other hand, Node-RED shows two concentrations around 18 and 21 s.

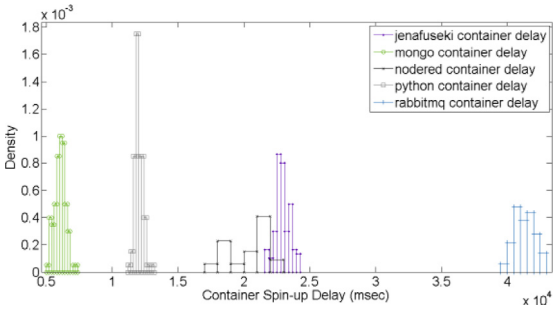

Fig. 6. Containers spin-up time distributions.

6. Conclusions

As a conclusion, combining a multitude of technologies spanning across different domains such as semantics, AI, integration approaches and IoT management platforms can prove both challenging and beneficial in order to increase the level of situational awareness around the status of a supply chain and its dependencies. The usage of Node-RED has enabled the integration between diverse and complex systems, enabling information adaptation and workflow creation in order to implement the sequences of actions needed. Finally, dockerization of the service enabled the quick and seamless deployment of all the elements and their automated connection, discovery and configuration, while the extended focus on enabling inputs to be performed through UIs may enable the easier usage and needed declarations of the framework, fully decoupling the end user from the underlying semantic definitions.

Acknowledgments

The work presented in this paper has received funding under the AffectUs extension (Greece), a TagItSmart Open Call 1 Extension project co-funded by the European Union (EU) Horizon 2020 program under Grant number 688061.

Conflicts of interest

The authors declare that there is no conflict of interest in this paper.

References

[1] Gartner Top 8 supply chain technology trends 2018, available at: https://www.gartner.com/smarterwithgartner/gartner-top-8-supply-chain-technology-trends-for-2018/.

[2] T. Cucinotta, et al., A real-time service-oriented architecture for industrial automation, IEEE Trans. Ind. Inf. 5 (3) (2009) 267–277.

[3] Z.D. Patel, A review on service oriented architectures for internet of things (iot), in: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, 2018, pp. 466–470.

[4] A. Pal, K. Kant, Iot-based sensing and communications infrastructure for the fresh food supply chain, Computer 51 (2) (2018) 76–80.

[5] P. Alho, M. Jouni, Real-time service-oriented architectures: A data-centric implementation for distributed and heterogeneous robotic system, in: IESS, Springer, Berlin, Heidelberg, 2013, pp. 262–271.

[6] C. Busse, J. Meinlschmidt, K. Förstl, Managing information processing needs in global supply chains: a prerequisite to sustainable supply chain management, Journal of Supply Chain Management, http://dx.doi.org/10.1111/jscm.12129.

[7] Kuo-Jui Wu, et al., Achieving competitive advantage through supply chain agility under uncertainty: A novel multi-criteria decision-making structure, IJPE (ISSN: 0925-5273) 190 (2017) 96–107.

[8] Z.D.R. Gnimpieba, A. Nait-Sidi-Moh, D. Durand, J. Fortin, Using internet of things technologies for a collaborative supply chain: Application to tracking of pallets and containers, Procedia Comput. Sci. (ISSN: 1877-0509) 56 (2015) 550–557.

[9] C.N. Verdouw, J. Wolfert, A.J.M. Beulens, A. Rialland, Virtualization of food supply chains with the internet of things, J. Food Eng. (ISSN: 0260-8774) 176 (2016) 128–136, http://dx.doi.org/10.1016/j.jfoodeng.2015.11.009.

[10] Riccardo Accorsi, Marco Bortolini, Giulia Baruffaldi, Francesco Pilati, Emilio Ferrari, Internet-of-things paradigm in food supply chains control and management, Procedia Manuf. (ISSN: 2351-9789) 11 (2017) 889–895, http://dx.doi.org/10.1016/j.promfg.2017.07.192.

[11] S. Biswas, J. Sen, A proposed architecture for big data driven supply chain analytics, IUP J. Supply Chain Manag. 13 (3) (2016) 7–33.

[12] EVRYTHNG IoT management platform, available at: http://evrythng.com/.

[13] AffectUs D1.2 Integrated Component Prototype, 2018, Available at : https://tagitsmart.eu/AffectUs/D1.2.pdf.

[14] G. Kousiouris, et al., An integrated information lifecycle management framework for exploiting social network data to identify dynamic large crowd concentration events in smart cities applications, FGCS (ISSN: 0167-739X) 78, Part 2 (2018) 516–530.