

Open Modular Components in the Industry using vf-OS components

Joao Giao¹, Joao Sarraipa¹ and Ricardo Jardim-Gonçalves¹,

¹ CTS, UNINOVA, Dep.^o de Eng.^a Electrotécnica, Faculdade de Ciências e Tecnologia,
FCT, Universidade Nova de Lisboa,
Caparica, Portugal
{jgs, jfss, rg}@uninova.pt

Abstract. The increasing complexity in industrial and information technology in the last decades forced the manufacturing process to adapt to these new evolution trends. Factories are using the advancement of the Internet of Things (IoT) to have multiple platforms with sensorial information into their factory processes, to improve and reduce the costs of their products and increase their profit. To address these complexes and useful computations, the vf-OS (Virtual Open Operating System) aims to facilitate the development of applications using their individual components as well as FIWARE enablers.

Keywords: Industry, IoT, Sensor, vf-OS

1 Introduction

An increasing number of heterogeneous smart objects communicate and interconnect through the internet. However, to use the full potential of these smart objects, it is necessary to transcend the interoperability issues between them. In the past years, there has been efforts to overcome this issue at system and network level. At system level, there can be a lack of system's interoperability, which makes necessary rewriting the software code for every new smart component that is inserted into the old system. At the network level, the smart components cannot communicate with each other because they have different interfaces and API protocols to interconnect with the surrounding environment [1].

Nowadays, it has been emerged a transversal discipline called CPS (Cyber Physical Systems), which integrates various intelligent components that has helped software architects and designers to deal with complex and difficult systems. CPS are combinations of computations and physical processes that enable the control of physical processes with computing and communication infrastructures [2] [3]. CPS allows sensors and actuators to be controlled by a control unit, usually composed by more than one microcontroller, where physical processes affect computations and in the other way around as well. The design of these systems requires from experts a complete knowledge about the computers dynamics (e.g. software or networks) and the physical processes to deal with big and complex data flows. In order to ease the development and deployment of new services that use such kind of data processes, smaller operating systems have been developed to help in the building of applications

that use IoT devices, like the vf-OS (Virtual Factory Open Operating System). The vf-OS system has been developed by the European Commission co-funded project called vf-OS [4]. vf-OS provides the necessary software tools to connect a factory and their internal devices by managing a factory related computer hardware and software resources, providing mutual services for factory computational programs.

Therefore, the main research question that emerges is:

Can modular systems contribute to the development and deployment of CPS in manufacturing systems?

One additional motivation is to use open-source software to speed up the software development and minimize the costs. As such, this paper proposes an architecture that explores the potential usage of open source software components applied in the industry, using some of the vf-OS system components already available. Accordingly to the Open Source Initiative [5], an open source software is a software that can be used without costs, changed and shared with anyone. Additionally, IoT components together with open standard protocols, can both assure interoperability and hence present a unified and a stable software infrastructure.

2 Related literature

In smart factories, condition monitoring systems (CMS) represent the business logic that monitor the state variables and create diagnostics of current and future factory' errors. However, most factories have several tasks depending on the associated production machine or process, which results in varying the requirements for cyber-physical computing architectures. Notwithstanding, in [6] the authors proposed a general processing steps for CMS: (1) State detection - measuring and store machine parameters that represent the current status of the factory. For this stage, sensors are used for monitoring and quality assurance; (2) State comparison – comparison of the current data with predetermined nominal data. After data collection from, eg., sensors from loads, temperatures, machine parameters and environment conditions, these values are compared with the manufacturer machine supervisor. This state relates with CPS, by having multiple types of sensors for perceiving the factory' environment and efficient mechanisms to process information and send events when is necessary; (3) Diagnosis – Specific error diagnostics. Based on the previous states, it is possible to detect potential errors or anomalies.

2.1 CPS Architectures

A CPS model contains models of software and physical processes, intelligent computation platforms and networks. It requires a multi-disciplinary development, not only on the distinct computational and physical components, but also on their integration and interaction. Accordingly to [7], CPS research has been focused on identifying needs, challenges and industrial sectors opportunities, encouraging multidisciplinary collaborative research between academia and industry. The main goal between this synergy is the development of new engineering techniques and methods to build high-confidence systems, intelligent, secure and integrated at all scales.

In a near future, a factory will have millions of sensors and actuators controlled by thousands of CPUs. To have this sensorial adaptation, the factories' system needs to flexibly adapt to an ever-changing external environment. System's architecture dictates how the passive or active elements can be organized and behave, accordingly to the external environment and the internal conditions defined by controllers [8].

2.2 Modularity

Modularity allows specialized functionalities that can be achieved through a plug-and-play fashion, opening opportunities for innovation and specialized software that allows developing heterogeneous and complex systems. To have this seamless integration of different components, the interfaces of the components must be standardized to facilitate the interoperability between them [7]. Adaptability is one advantage of these modular components. Since factory's machines need to be enhanced over time to add and improve their functionalities, their hardware/software must also adjust to these changes, which can be easily made through changing/adding new components on the process workflow.

Yelamarthi et Al. in [9] present that most of the IoT architectures are not prepared to be used in various domains. To overcome this customization problem, the authors proposed a modular IoT architecture that can be configured for different domains, such as smart home, city, industry for environment control, optimizing energy usage and automating applications, agriculture for soil and temperature monitoring, healthcare for monitoring patient physiological parameters or remote motion tracking.

2.3 Warehouse Management Systems

Accordingly to C.K.M. Lee et al [10], the major challenge in modern warehouse management is the inventory accuracy, space utilisation, process management and optimize when raw materials are ordered. To overcome these challenges, it is necessary an agile supply chain, which can be done with CPS network, that will connect users, objects and physical processes in a warehouse operation. This improvement requires the integration of technological and administrative innovations, which includes the proper selection of CPS technologies, ambient intelligence, timely information flow and agility.

3 vf-OS

IoT platforms are designed on the aspect of collecting data and using it in edge computing platforms for cloud computing. To be used in real-life scenario, the existing software need to be able to simulate and process parts of the production processes, which can be done through vf-OS. vf-OS system will provide a set of manufacturing applications and tools for manufacturing users and software developers, respectively. Thus, they provide a set of services for factory computational programs, which can be used for managing factory related computer hardware and software resources as well

as to provide services for factory computational programs. As an open operating system, it has components that allow industrial users to download and upload the applications (vApps) developed with vf-OS. The vApps, as illustrated by Fig. 1, can use the vf-OS components, as a software layer extended over multiple factories, to acquire sensorial information and use the actuators over them.

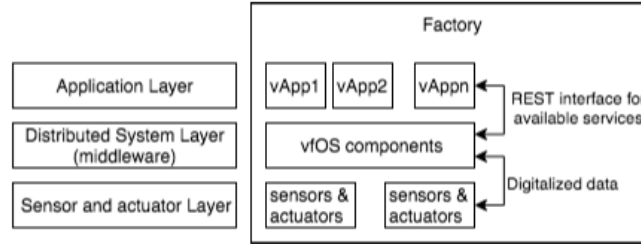


Fig. 1. High-level layers interaction and information flow in a proposed modular architecture for an industrial scenario.

First, the physical entities need to be sensed by a transducer and the electrical property will be converted into a numeral value, done by the sensorial and actuator layer. Then, through the distributed system layer, this raw data will be acquired from different remote locations in the factory' environment to be filtered and send to a common storage to be processed. This same layer will also be responsible to perform the necessary algorithms and operations that are necessary to give new information and functionalities to the factories, which is done afterwards by the application layer.

3.1 vf-OS modular components

vf-OS project uses a Service Oriented Architecture (SOA) approach in which all components perform individual functionalities and can connect with each other to create more complex services. This interoperability can be achieved taking into consideration that all components implement and publish a REST interface allowing a common data interface. This has the advantage of being easy to test and modify the service' workflow, by adding/removing components as needed. As presented in [11], the vf-OS building blocks were already used in an industrial implementation scenario to orchestrate a vApp through the vf-OS component "Process Designer", a process model in form of a BPMN (Business Process Model and Notation). Some components able to be used in such BPMN component are presented in the following:

- Enablers Framework (EF) - Acts as a bridge between service provider and service consumer, providing support for enablers' integration, installation and management of their instances. The main added value of this component is to ensure an unique interface and common access for all registered enablers, acting as a wrapper engine for the different enablers and the vApps. For the European commission, these enablers, developed from FIWARE, can be used in multiple domains, e.g, connect to IoT devices, process data and media in real-time, perform Big Data analysis [12], ensuring portability, interoperability and openness of services across Europe, by contributing to the Digitisation of Europe Industry and European Cloud Initiative [13].

EF module represents an advantage for the existing enablers due to the fact that it facilitates the enablers' usage through unique and easy REST API;

- Enablers - Open-source software that exposes heterogeneous services. They can be classified as FIWARE Generic Enablers, Manufacturing enablers and vf-OS enablers. The FIWARE enablers were developed under the FIWARE foundation on the context of the Future Internet Private Public Partnership (FI-PPP) that was part of the FP7 (Seventh Framework Programme) of the European Commission [14]. Their goal was to provide services in different domains such as IoT, data context management, Cloud Hosting, security platforms, etc [15]. Manufacturing enablers are enablers that were developed with the purpose of providing functionalities for manufacturing industries, such as the developed enablers under the FITMAN (Future Internet Technologies for Manufacturing Industries) project [16]. vf-OS enablers were developed to complement/improve the specific applications developed to be used by the vf-OS users;

- Messaging-Publish/Subscribe (PubSub) - Data infrastructure middleware responsible for the data communication between different components. This component has an open source message broker, RabbitMQ [17], that creates a distributed communication layer for messaging handling with Advanced Message Queuing Protocol (AMQP)[18] standard protocol. It also allows to configure the security policies in order to define which modules have access to specific message queues;

- Datastorage - Scalable data storage system that is capable of handling real-time sensor data and events through a common API (REST). It encapsulates a relational database (for structured information), time series database (for sensorial data) and document oriented database (for unstructured information);

- Drivers - This component provides a collection of libraries that allows to collect data from, and send commands to, physical devices such as industrial automation devices (RFID readers, PLCs, smart sensors, etc). This software module can connect to sensors and actuators through physical interfaces, communication protocols or intermediary systems such as field bus controllers, providing a flexible and modular approach to interact with devices [15].

3.2 Proposed Architecture and Prototypical Implementation of an Industrial Scenario

In this paper, a generic architecture to integrate different modules in order to be used by a manufacturing factory, is presented. The developed work presented in this paper is in the context of the vf-OS project, where the PhD student, author of this paper, developed the module “Enablers Framework” and the libraries to use the “Messaging and PubSub” services. These modules are central components that enables integration and interoperability required in system modules. Further information about these components is described in next section.

The goal is to allow two different factories to communicate with each other in order to coordinate their production and verify if the factories have stored enough amount of spare parts for future production, as can be seen by Fig. 2.

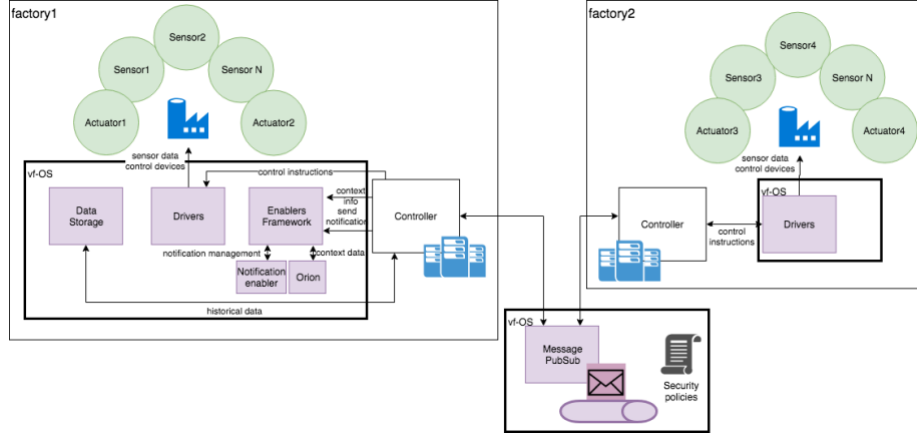


Fig. 2. General overview of the proposed architecture between two different factories and a communication channel for inter-factory resources' planning.

The required modules for the proposed architecture are the following: (1) data storage, a software component responsible to save the historical data of the factory's sensors. A predictive analytics can be used through historical and current data to predict future outcomes [19]; (2) drivers, to interact with physical industrial devices to integrate sensors' data and send commands to them; (3) Enablers Framework, Orion & Notification Enablers. Orion is an enabler that is used for a data/context scenario to update context information, queries and subscriptions. This context-aware component allows to model and gain access to context information connected to sensor networks, users and/or existing industrial applications allowing, for e.g., to receive notifications when some pre-defined condition is accomplished. The Notification enabler is a vf-OS enabler that provides the components of the system an easy to use notification system, to alert when the stock is off limits for each spare-part of the factory. To facilitate the usage of enablers, the EF is used by interpreting the enablers' REST and NGSI [20] protocols into an easy to use and understand REST interface; (4) controller, this module will control the production of the factory, accordingly to the available resources and the output of the other factory; (5) messaging-pub/sub, that will be responsible for the communication between the factory's modules and inter-factory communication. By defining a security policy, it is possible to secure the communication channels, granting the access to the authorised components.

4 Industrial scenario

The presented industrial scenario is related with manufacturing, automation and logistics, which uses an application named "vfStockPolicies" (Pilot 1, application 3) [21]. The main goal of this vApp is to manage spare-parts that each industrial company needs, monitor their stocks and alert when the stock is lower than a minimum defined for each spare-part. Contrarily to other IoT applications described in [9], the goal of this application is not to have a low-power system, but rather having a reliable and

wearable devices attached to the existing factory devices. Since this application requires users from different factories, it is crucial that a RBAC (Role based access control) is used. Currently, the messaging-pub/sub component already has integrated an external service which allows the configuration of the authentication (process of ascertaining the user identification through a password) and authorisation (specification of rules that determine who is allowed to use the messaging channels). After the authentication and authorisation permissions, the factory developer must configure the data access of the sensors/PLCs to perform the device integration with the industrial vApp. Taking into consideration that the Driver's module already contains a complete working copy of the IO component binaries as well as many common functionalities for input and output of the sensors and actuators, respectively, the factory developer only needs to develop the software algorithms [22]. This variant has the advantage of the software being transferable and it doesn't require rewrite of the whole code. Another configuration to be made is to define the context rules on Orion enabler and Notification Enabler, which will be used to evaluate a specific case and later validate if there are materials to create a product inside both factories. The communication channels also need to be configured inside the message and PubSub component for the message channel between factories and between the factory's internal components.

The controller will use the ORION context broker to reason the previously formalized rules for the context and validate if both factories have the required spare parts to create the final product. If the stock of the spare parts is lower than a certain threshold, the Notification Enabler will be used to alert the maintenance manager of the company about this situation in order to acquire new materials.

The future works concentrate on the total experimentation of the proposed architecture in the real industrial environment under the last phase of the vf-OS project, represented on

Fig. 3. This application will be executed in different industrial systems in Spain in order to manage the spare parts of two different factories that use materials from each factory to create a new final product.

Another crucial aspect to take into consideration about the implementation of this architecture in a real factory is the security. Factories have sensitive data that must be protected for unauthorized use, so it is mandatory to correctly define a user management to regulate user permissions, especially if there is a communication between different factories.

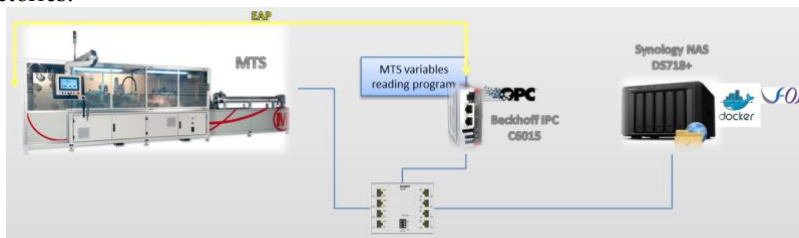


Fig. 3. vfStockPolicies - Pilot 1 - Application 3 from vf-OS project use case.

5 Conclusions

This paper defines an architecture aimed at the software design using open-source modular components to develop an application to be used in the industrial field. It allows researchers to speed and quickly develop, without monetary costs, their IoT applications, without having to understand how each component works and communicate with each other. This modular component approach provides a flexible way of building industrial applications which decouples the complexity of constructing complex CPS into simpler subsystems. Thus, the proposed architecture works as a building CPS catalyst tool. It facilitates the development of intelligent systems that actuate accordingly to its applied environment. In this case, it was used the proposed architecture with an application to remotely determine which spare parts are available on a different factory and schedule the production accordingly to such lack of materials. Furthermore, the application can be used to warn maintenance managers with sufficient time to prevent stock breakout and the consequent repair delay in forthcoming maintenance operations.

Due to the use of such modular components the presented work contributes directly at the system level, where multiple components with different structures interoperate with each other to perform a workflow of services and is able to be customized for a diverse range of applications. On the other hand, at the network level, the solution proposed just uses a communication standard protocol to facilitate the communication between different systems.

References

- [1] P. Rosenkranz, M. Wählich, E. Baccelli, and L. Ortmann, "A Distributed Test System Architecture for Open-source IoT Software," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems - IoT-Sys '15*, 2015, pp. 43–48.
- [2] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling Cyber-Physical Systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, Jan. 2012.
- [3] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, 2014, pp. 1–4.
- [4] "vf-OS Wiki," *Vf-OS, vf-OS consortium*. [Online]. Available: <https://www.vf-os.eu>. [Accessed: 10-Jan-2019].
- [5] "Open Source Definition," *Open Source Initiative*. [Online]. Available: <https://opensource.org/>. [Accessed: 10-Jan-2019].
- [6] H. Fleischmann, J. Kohl, and J. Franke, "A Modular Architecture for the Design of Condition Monitoring Processes," *Procedia CIRP*, vol. 57, pp. 410–415, 2016.
- [7] R. Baheti and H. Gill, "Cyber-physical Systems, from : The Impact of Control Technology, available at www.ieeeccs.org," 2011.
- [8] P. Hehenberger, B. Vogel-heuser, D. Bradley, B. Eynard, T. Tomiyama, and S. Achiche, "Design , modelling , simulation and integration of cyber physical

- systems: Methods and applications,” *Comput. Ind.*, vol. 82, pp. 273–289, 2016.
- [9] K. Yelamarthi, M. S. Aman, and A. Abdelgawad, “An application-driven modular IoT architecture,” *Wirel. Commun. Mob. Comput.*, vol. 2017, 2017.
 - [10] C. K. M. Lee, Y. Lv, K. K. H. Ng, W. Ho, and K. L. Choy, “Design and application of Internet of things-based warehouse management system for smart logistics,” *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2753–2768, Apr. 2018.
 - [11] P. Corista, J. Gao, J. Sarraipa, R. Almeida, O. G. Perales, and M. Nejib, “Enablers Framework: an approach to develop applications using FIWARE,” in *Enterprise Interoperability*, 2018.
 - [12] “FIWARE Foundation: from research to those digital services you will love,” *European Commission*, 2016. [Online]. Available: <https://ec.europa.eu/digital-single-market/en/news/fiware-foundation-research-those-digital-services-you-will-love>. [Accessed: 01-Feb-2019].
 - [13] “The Future Internet platform FIWARE,” *European Commission*, 2018. [Online]. Available: <https://ec.europa.eu/digital-single-market/en/future-internet-public-private-partnership>. [Accessed: 01-Feb-2019].
 - [14] P. Corista, D. Ferreira, J. Gao, J. Sarraipa, and R. J. Goncalves, “An IoT Agriculture System Using FIWARE,” in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 2018, pp. 1–6.
 - [15] “D2.1: Global Architecture Definition - version 1.2.2,” in *Vf-OS, vf-OS consortium*, 2017.
 - [16] “Fitman project,” *Fiware for Industry*. [Online]. Available: http://www.fiware4industry.com/?page_id=979. [Accessed: 10-Jan-2019].
 - [17] Pivotal, “RabbitMQ.” [Online]. Available: <https://www.rabbitmq.com/>. [Accessed: 10-Jan-2019].
 - [18] “AMQP - Advanced Message Queuing Protocol.” [Online]. Available: <https://www.amqp.org/>. [Accessed: 10-Jan-2019].
 - [19] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” *Int. J. Inf. Manage.*, vol. 35, no. 2, pp. 137–144, 2015.
 - [20] “NGSI-9/NGSI-10 information model,” *Fiware*. [Online]. Available: https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/NGSI-9/NGSI-10_information_model. [Accessed: 10-Jan-2019].
 - [21] “D1.3 Providers Scenarios Characterisation - version 1.0.1,” in *Vf-OS, vf-OS consortium*, 2017.
 - [22] “D4.1.2a: WP4 Virtual Factory I/O Umbrella Deliverable - Vs: 1.0.8,” in *Vf-OS, vf-OS consortium*, 2017.