Enablers Framework: an approach to develop applications using FIWARE

Pedro Corista[1], Joao Giao[1], Joao Sarraipa[1], Raquel Almeida[2], Oscar Garcia, Néjib Moalla[4]

[1] CTS, UNINOVA, DEE, FCT, Universidade Nova de Lisboa, Caparica, Portugal
{pc, jgs, jfss}@uninova.pt
[2] KnowledgeBiz, Almada,Portugal
raquel.melo@knowledgebiz.pt
[3] Information Catalyst SL,Valencia, Spain
oscar.garcia@informationcatalyst.com
[4] Universite Lumiere Lyon II, France
Nejib.Moalla@univ-lyon2.fr

Abstract. Industry is moving towards smart systems and automation, which leads modern factories to demand a set of tools capable of sustaining their needs. Generic and specifics sets of enablers, are emerging as core industry 4.0 components, as they provide the software services that factories of the future require. Although enablers can be at the FoF foundation, a system able to integrate them together in an interoperable way does not exist. To integrate the several enablers with their matching applications, it is necessary to create an adaptable system able to overcome this gap. This article proposes the Enablers Framework, a component of vf-OS project that acts as a bridge between applications and enablers, granting programmers an easy access to the services they need. This framework will thus open the possibility to several business opportunities to arise, since the integration of such versatile components is an alternative to the heterogeneous components used in the industrial panorama.
Keywords: Enablers, Enablers Framework, FoF, vf-OS

1 Introduction

The World is facing the fourth industrial revolution based on ICT, specifically architectures and services, as key innovation drivers for manufacturing companies. Traditional factories will increasingly be transformed into smart digital manufacturing environments but currently, the full potential of ICT in manufacturing is far from being fully exploited. Factories are complex systems of systems and there is a need to develop a platform on which future manufacturing applications can be built. Examples of platforms exist in some industrial sectors but there is a lack of cross cutting platforms based on open standards for creating an ecosystem for cooperative innovation. The approaches that have been used in industry rely on point to point solutions that compete, by providing new specific features or fixing specific issues. Using these solutions result on having a highly heterogeneous industrial IT panorama, where each

factory may have similar solutions with different architectures and programming languages. Having such heterogeneous solutions create problems on adapting current solutions from an existing factory to another, as a specific solution is developed to a specific end. There is the need for a homogeneous solution that allows to support the needs from manufacturing companies and fill current approaches' gaps, vf-OS (virtual factory Operating System) aims to be that solution. The goal of the vf-OS project is to develop an Open Operating System for Virtual Factories, responsible for managing factory related computer hardware and software resources and providing common services for factory computational programs. This system will be the component in a real factory system where all factory application programs will run. To fulfill this objective, the system must have a component which follows kernel's principles and logic. A kernel is a component that acts as a resource manager by taking care of the various programs that are running concurrently in the system [1]. As seen in [2] one of Kernel's functions is to interface all the applications, granting the bridge between them and the resources they need. Following this concept, vf-OS has a component called Enablers Framework (EF) that interfaces the applications (vApps) and the services they require. Such a component allows to integrate the necessary tools, without the user needing to experience the associated installation complexity. This paper proposes a definition for the EF. The first section highlights the EF's purpose and its role as a wrapper, it also explains and details its main components. The second section is related to the business opportunities that EF provides, considering software developers, manufacturing users and ICT providers.

2 Enablers Framework

EF is the component that acts as a bridge between service providers and service consumers. There already exist approaches to work with FIWARE, that allow to integrate enablers with applications, as seen in [3], EF aims to provide support for enabler integration and also installing and managing instances of an enabler. To understand the overall functionality of EF's, it was necessary to define what are the service providers for the applications that consume them (vApps).

The service providers were chosen from the set of solutions adopted from the FIWARE project [4] and are classified as generic enablers (GE) or specific enablers (SE). GE can be found at [5] and provide functionalities that ease the connection to the Internet of Things (IoT) technologies and devices, process data and media in real-time at large scale, perform Big Data analytics or incorporate advanced features for the interaction with users. SE can be found at [6] and provide functionalities, which are more specific to the manufacturing domains such as manufacturing assets virtualization, collaborative manufacturing asset management, 3D scanning and virtualization, social data analysis, etc. Using the functionalities of the several GE and SE, a programmer can have a high range of tools that can act as core software components, while designing the vApps. How EF integrates the existing enablers is explained in the next sections.

2.1 Enablers Framework as a bridge

The EF component, provides a solution for integration, exposition of and uniform access to functionalities of the different enablers in a single service-based component. The enablers, specifically GE and SE, can expose heterogeneous sets of service interfaces that could be used by vApp developers posing a need to understand and implement them. EF component acts as a wrapper engine for the different enablers and the vApps consume its services. Such an approach presents high interoperability advantages, as any enabler can be accessed through the platform, and integrated in any developed application. The developer doesn't have to worry about the enablers installation process in the target machine, as the EF oversees its whole process. This way he can focus on the development process, saving time and resources. The first step to use an enabler is to register it through EF's interface. From that moment on, the user can create an instance of the enabler and use it. The instantiation process allows users to choose whether the enabler instance is public or private.

The instance creation and installation are processed by creating a Docker container from an enabler image, as seen in [7]. To run an enabler on a personal computer or public server it is necessary to have access to its virtual instance. The Docker solution allows to instantiate enablers, following an approach similar to the one done in [8] for microservices, presenting an alternative to actual Virtual Machines (VM). Docker is a platform that lets the user run an application in an isolated environment meaning that it is possible to run multiple containers at the same time on the same host. Using such approach, it allows any enabler to be packaged into isolated containers with necessary code, runtime environment, system tools, system libraries and settings bundled all together. This strategy can lead towards the realization of efficient, lightweight, self-contained software packages and guarantees that software always runs the same way, regardless of where it is deployed. A Docker based approach secures the enablers instantiation, granting that all enablers can be installed, thus contributing to the integration process.

2.2 Components functionalities

After an instance is created, a user can access the enabler he needs through the EF. The access to each enabler is granted by following an approach analogous to proxy communication. Whenever an application needs a service it communicates with the framework, that contacts the corresponding enabler, performing the necessary request and delivering its result to the vApp. The communication between the applications and the framework can be made using Messaging and Pub/Sub components, all through the Process Execution. Between the applications and EF exists a Process Executor that orchestrates what the application will need. Communication between enablers and the framework is performed using IoT protocols such as REST and NGSI. To perform the enablers integration, EF uses two main modules: Request Handler (RH) and Enabler Registry Lookup Services (ERLS) (see Figure 1).

The RH abstracts and implements all necessary functionalities for translating requests to/from vApps and enablers that are served by the EF. The internal components

of this module parse the requests and create a suitable communication channel for accessing the functionalities of the enablers and build necessary response messages, based on the result of the function invocation. This module uses the technical details of the enablers from the ERLS and relies on suitable proxies to invoke the method requested by vApp. ERLS abstracts all the data models corresponding to registrations details and provides functionalities for performing Create, Read, Upgrade and Delete (CRUD) operations on them. This module is also responsible for providing service details of the enabler to the RH.

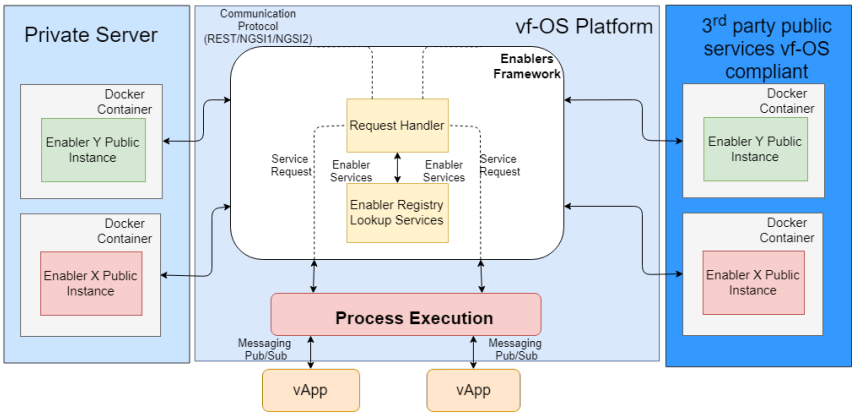

Fig. 1. Enablers integration using Enablers Framework

The two presented modules interactions grant the enablers integration. After receiving a service request from an application, the RH contacts ERLS to verify if such enabler is registered in the framework. Upon verifying it, ERLS informs RH about the target enabler's services, this information is then used to exchange information between the framework and the enabler. The framework informs then the vApp on the service's response or delivers an error if the target enabler or service are unreachable.

2.3 Implementation scenario

This section provides a scenario that explains how the vApps can consume the enabler's services they need. The implementation scenario uses the EF's components presented in section 2.2, along with the Process Enabler (PE) (figure 2). The PE is a tool used to orchestrate the different tasks that a vApp will carry out. When developing a vApp (Design Time), the Process Designer (the designer side of the Process Enabler) will be used to define the different sub-tasks (e.g. call an analytics algorithm) and/or sub-processes (e.g. persist data in the storage) that will be necessary for executing the vApp, and assign the necessary EF libraries to achieve integration. This process model, in the form of a BPMN, will be then incorporated to the vApp source code so vf-OS is able to execute the different process steps. Thus, the Process Designer will contact the vf- OS Marketplace, vf- Store, to access the definition and the services of the different assets that the vApp developer wants to make use of. These

assets could indeed be an enabler which offers certain services and then, this enabler will be accessed through the EF at runtime. At runtime, when the vApp is being executed, the vf-OS Platform will call the Process Execution (PEX) PE. This way, any vApp will be executed following the same pattern. Then, the PEX will open the BPMN definition of the vApp and orchestrate the calls to the different assets specified in this definition. When it is time to call the services offered by an enabler, regardless if its GE, SE or vf-OS Enabler, the PEX will call the EF that will be on charge of relaying the call to the enabler specified. Then, when the enabler replies, the EF will take its response, pack it and send it to the PEX so the response packet can continue its route in the BPMN flow of the vApp.

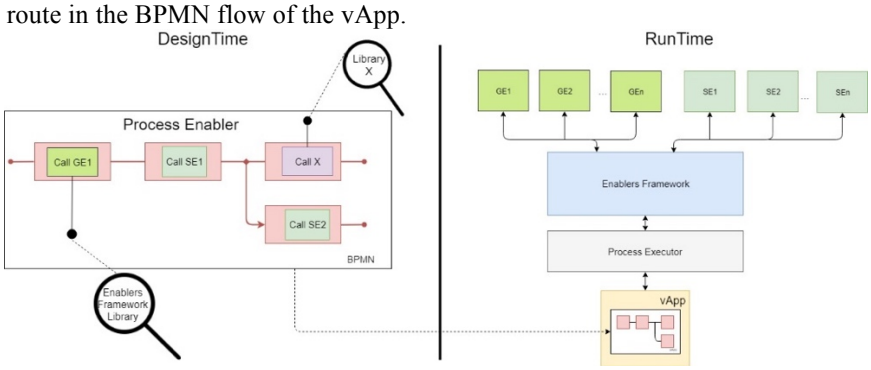

Fig. 2. Application scenario

3 Enablers Framework business opportunities

vf-OS aims to become a business multi-side platform enabling value by creating interactions between external producers and consumers. This approach provides an open, participative infrastructure for these interactions and sets governance conditions for them. Its main goal is to make producers' and consumers' information available to each other, in a timely, accurate and transparent way, diminishing their information asymmetry, and making the digital exchange of products and services a better interaction experience for both. The EF and its underlying enablers are a significant part of the added value of vf-OS, which business value relates to: 1) End user enablers that could be directly utilised in the development of vApps (design time), or solely to facilitate an easy plug and play of other resources (CPS, APIs etc.); 2) Scalable Services through enablers that can provide services, which vApps or other components can consume; 3) The vf-OS Ecosystem where the EF as the bridge between service providers and consumers facilitates their integration with the vf-OS platform and its associated services.

Being developed under FIWARE guidelines Enablers must follow an open source approach, but this doesn't exactly mean that the enabler services execution through the EF should be taken for free. On top of this free approach, enablers developers could sell support contracts on top of enablers deployment for continuous maintenance, specially customization. Thus, it is expected that all these transactions occur

through vf-OS Store, being the platform the main sponsor of the marketing and selling activities of all its vApps and enablers specific services.

4 Conclusions and future work

Modern industry demands new tools capable of providing resources to its needs. Generic and specific enablers provide the services that factories of the future require. This article presented a framework, part of the vf-OS project, able to integrate enablers with the applications, acting as a bridge between them. By relying on a Docker-based approach the framework provides several instances of the desired enabler, without the user having to pass through its installation process. The enabler can then be accessed through the framework, that provides the means to access the enabler's services. The orchestration of the services can be achieved through Process Enabler, that assigns the necessary tasks which the application will carry out. The PE will then generate a BPMN which will be integrated in the vApp to execute the defined tasks. The integration of the enablers implies new business opportunities as it opens new business opportunities for its customer segments directly. Additionally, enablers can be used directly in shop floors for easy establishment of IoT connections, or indirectly by enrichment of the vApps functionalities.

5 Acknowledgements

The research leading to these results has received funding from the European Union H2020 Program under grant agreement No. 723710 "Virtual Factory Open Operating System" (vf-OS).

References

[1] W. Mauerer, *Linux ® Kernel Architecture*. 2008.

[2] Justin Garrison, "What is the Linux Kernel and What Does It Do?," 2010. [Online]. Available: http://www.howtogeek.com/howto/31632/what-is-the-linux-kernel-and-what-does-it-do/. [Accessed: 19-Oct-2016].

[3] S. Di Cola, C. Tran, K. Lau, A. Celesti, and M. F. B, "A Heterogeneous Approach for Developing Applications with FIWARE GEs," *Eur. Conf. Serv. cloud Comput.*, vol. 2, pp. 65–79, 2015.

[4] FIWARE, "FIWARE Project." [Online]. Available: https://forge.fiware.org/projects/fiware/.

[5] FIWARE, "Generic Enablers." [Online]. Available: https://catalogue.fiware.org/enablers.

[6] FITMAN, "FITMAN Manufactoring Enablers," 2013. [Online]. Available: http://catalogue.fitman.atosresearch.eu/catalogue.fitman.atosresearch.eu/enablers.html.

[7] D. Merkel, "Docker: lightweight Linux containers for consistent development and deployment," *Linux J*, p. 239, 2014.

[8] J. Stubbs, “Distributed Systems of Microservices Using Docker and Serfnode,” 2015.