1

vf-OS Architecture

Danny Pape [1], Tobias Hinz [1], Oscar Garcia Perales[2], Francisco Fraile[3], José Luis Flores [4] and Oscar J. Rubio [4]

[1] Ascora GmbH, 27777 Ganderkesee, Germany
[2] Information Catalyst Enterprise, Crewe, United Kingdom
[3] Universitat Politecnica de Valencia, 46980 Valencia, Spain
[4] IK4-Ikerlan Technology Research Centre, 20500 Gipuzkoa, Spain
pape@ascora.de

Abstract. This paper discusses the software planning of the Industry 4.0 project vf-OS. Information and requirements of the real world are included in the structural planning of the complex software system. The waterfall model is used, in which several phases of the planning build on each other. Starting with the system architecture, the individual components are defined in vf-OS and the individual connections to each other and to the outside world are specified. The application of various technical solutions is also dealt with in this phase. Based on this, the functional and technical specifications were drawn up, which describe the internal processes in more detail and also define the communication level. In today's industry 4.0 the emphasis is on security, since sensitive data in particular must be protected and the risk of abuse has increased as a result of the connection of industrial processes to the Internet. In order to minimize or avert these dangers, vf-OS has developed a concept for security and privacy that lists the main sources of danger and their solutions.

Global Architecture Definition

vf-OS is using a Service Oriented Architecture (SOA) approach in which the different vf-OS components implement individual functionalities and thus the composition of all these inter-related components form the complete vf-OS architecture to ground the vf-OS ecosystem. In order to apply this approach, all

components implement and publish a REST interface allowing the exchange of data (primarily) with the messaging bus to be implemented within the project. vf-OS will support Event Driven SOA features so that the different components can decide their interaction pattern and react to internal and external events. Following this approach, the components of vf-OS can behave either as services or as event producers and consumers.

The diagram in Figure 1 provides an overview of the high-level architecture of vf-OS and provides a formal split of functional components. This diagram classifies the components based on their role to vf-OS (design, runtime, etc.) and its relationship with external resources. The "Kernel box", situated at top-left of the diagram, indicates that all the components marked with the same color can be deployed within the vf-OS Kernel when vf-OS is installed as an OnPremise solution.

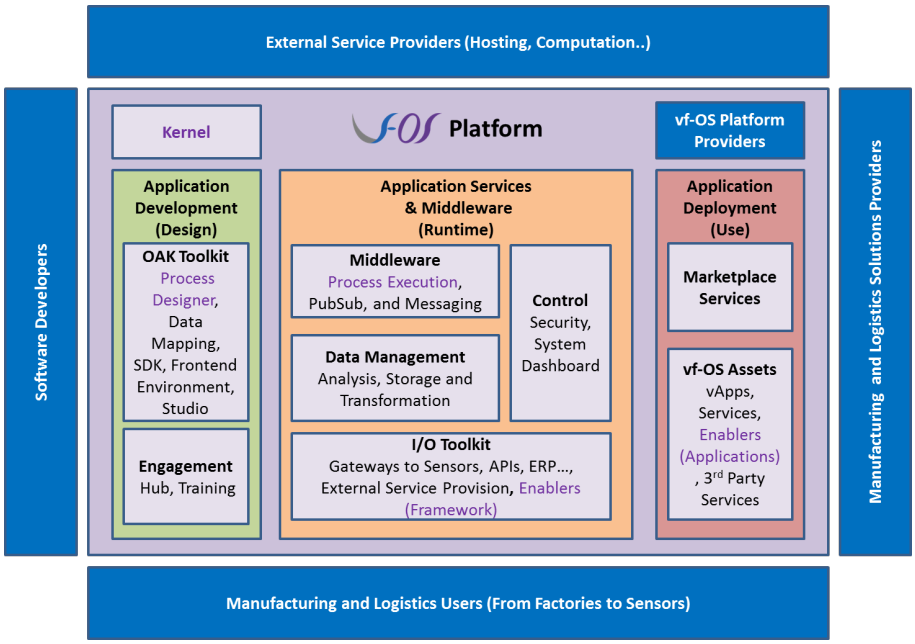

Figure 1. *vf-OS Global Architecture Definition*

Thus, the architectural building blocks are the following:

– Master Block: The vf-OS Platform acts as an executing environment of vf-OS architecture. It acts as a container, which includes all other components.

– Application Development Block aka design time comprises the different vf-OS components that will be used for the development of vf-OS Manufacturing Assets/vApps.

– Application Services & Middleware Block aka Runtime includes the vf-OS components that will be used by the Assets/vApps, when they are requesting vf-OS resources.

– Application-Deployment Block includes all vf-OS components that will be taken into consideration when the vf-OS environment is going to be in use.

vf-OS Assets are the different elements that are executed within, or interacts with, the vf-OS ecosystem. Intrinsically speaking, a vApp (a Manufacturing Smart Application) is the first example of what a vf-OS Asset can be, but it is not the only one. As such, a vf-OS Asset consist of several Solution Building Blocks (SBB). In summary, the Solution Building Blocks are prioritized as follows:
FIWARE Enablers > Specific Enablers > Open Source Solutions > Existing Solutions > New Development.

Functional Specification

The functional specification describes how the vf-OS platform works from the perspective of the user. More specifically, for every component, the functional specifications contain a description of the functionality and behavior from the perspective of the user, the sequence of actions taking place during the execution of the component to satisfy its functionality and the mockups of the user interfaces that support the interactions with different vf-OS users.

The behavior and functionality of the components is described using story maps [PAT 14]. Story maps organize user stories for software development in a matrix, following the user workflow from left to right and the user priority from top to bottom. This visualization helps users and developers to build a common understanding on the behavior of the component and the expected functionality of the different software releases. User stories capture the functionality that needs to be developed with enough granularity to identify inconsistencies or overlapping of functionalities between components. The same methodology is applied to the pilot applications. This provides an effective feedback mechanism to from the pilots to the individual components.

The story maps are linked to UML sequence diagrams used to depict the interactions between components and to UI mockups that describe the interaction with users. This way, every activity in the story map sequence has a high level description of the interaction with external components and/or users.

Finally, the models used in the component and the inner interactions of the component are described using UML class diagrams and a detailed component architecture diagram to enrich the high-level description provided in the global architecture definition.

Technical Specification

The technical specification is the primary document detailing the communication amongst vf-OS components. It defines and describes the API endpoints and data models to be transferred in vf-OS. Therefore, it takes into account the knowledge of the Functional Specification and builds the foundation for development efforts.

The target group of the technical specification are developers who need to build software that interacts with vf-OS components. To ease this approach the outfacing interfaces of each component are accessible in a RESTful manner [RIC 08]. An online documentation tool has been used to do this where each vf-OS component describes its endpoints and data models. This approach keeps the documentation flexible and update-able for later use during the development or if there need to be changes in a later phase.

To provide all information needed by the developer each API endpoint is separated into three parts:

– Description: API endpoints are introduced with a short description to show the purpose of the related endpoint.

– Request: contains the endpoint URI and the necessary parameters (information and data) to be transmitted to the endpoint component for a successful response

– Response: describes both cases of responses (success and error). In case of success, it describes the expected data and the corresponding standardized HTTP status code (e.g. HTTP/1.1 200 OK), in case of an error a description is returned including a HTTP status code (e.g. HTTP/1.1 404 Not Found).

Holistic Security and Privacy Concept

The main aim of the vf-OS infrastructure is to provide a set of information services enabling industrial processes. From the security point of view, protecting industrial processes is very challenging, due to the high economical incentives for hackers to interfere with them (e.g. stop, corrupt, spy). This has resulted in a rising wave of cyber attacks (e.g. denial of service, ransomware, espionage) targeting industrial services. Therefore, the vf-OS infrastructure must be adequately protected before it is deployed in production. This is not a trivial task, since vf-OS is composed of a

variety of inter-related components (see Figure 1) and services that may expose large attack surfaces.

Hence, guaranteeing continuous and appropriate availability, integrity and confidentiality (AIC) levels along the whole vf-OS infrastructure calls for a solid, holistic security and privacy model. This model elicits know-how from reference software development life cycles, applicable regulations (General Data Protection Regulations), key industrial IT standards (mainly ISA-62443) and recommendations from top-class security institutions (e.g. NSA, NIST). The objectives pursued are to minimize attack surfaces, maintain security risks at acceptable levels and keep security threats at bay, while minimizing the negative impact of security measures on the functionality and performance of vf-OS services.

– Mainly inspired by SELinux , NSA's Open Source Security Enhanced Linux, the vf-OS holistic security model integrates five simple, ruling principles:

– The security component shall act as a security proxy, intercepting, authenticating and authorizing all actions on the platform in order to guarantee the protection of the vf-OS resources.

– Following the example of Marketplaces such as Google Play or iOS Store, every external software development shall be provided as an Asset so that they can be examined and controlled in a uniform manner.

– Most security processes (credentials exchange, tokenisation, cryptographic verifications) are automated and transparent, but there may be humans in the loop. Particularly, the process of vApp installation will typically involve granting vApp access permissions (to resources) by the administrator(s) of the vf-OS Platform instance. These access permission policies are managed by using a central console.

– There is sandboxing by default. Any access (to a resource) that it is not explicitly permitted, because it has been authorised, is prohibited.

– Authorisation policies follow a mixed system based on Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) called RABAC model. This will be addressed by means of XACML, and can be extended to support ABAC as well.

The holistic security model envisioned above translates into a vf-OS security component, whose architecture is defined by the following components:

– A Primary Security Component: Comprised of a Security Command Center that concentrates the majority of security processes (authentication, authorization, and continuous security monitoring), an Installer Broker Service, to download and install vf-OS asset, an Identity Service to provide authentication, and an

authorization Service to provide global security policies. This component is the main target for cyber-attacks and represents the largest attack surface.

– Marketplace Identity Manager, located in the vf-OS Marketplace, it is intended as the dependable source of applications, externals packages, components etc.

– Marketplace PKI, located in the vf-OS Marketplace, it carries out the creation, delivery, and renewal of digital certificates for vf-OS in order to guarantee the authenticity and authorship of the vf-OS Assets.

– OAK Studio/Toolkit. It includes a set of tools for assessing the security of any development following a Software Development Life Cycle.

– Engagement Hub. It is responsible for providing required security information to the OAK Studio. This service is focused on providing check lists, threats lists, etc.

Conclusions

This paper describes the overall architecture and the specifications of vf-OS which are needed for the subsequent programming work. It ensures that the selected vf-OS technologies fit the requirements and that interfaces for all vf-OS software components are defined. Important security and privacy issues are promoted to their own focused task as they are cross-cutting concerns which largely need to be fulfilled by all vf-OS software components but of course they will be integrated with the overall specification stack. With the following completed core phases of software planning, the programming work of the individual components can begin.

– Architecture to define relations and data streams between vf-OS components and excluding APIs.

– Functional specification to define and prioritize functionalities depending on the requirements analysis

– Technical specification to define the access rules of each component.

– Holistic Privacy and Security Concept to prevent the system for misuse.

References

[JAL 12] Jalote, P., An integrated approach to software engineering, Springer Science & Business Media, 2012.

[PAT 14] Patton, J., Economy, P., User story mapping: discover the whole story, build the right product. "O'Reilly Media, Inc.", 2014.

[RIC 08] Richardson, L., Ruby, S., RESTful web services. "O'Reilly Media, Inc.", 2008.