JARVIS, A Hardware/Software Framework for Resilient Industry 4.0 Systems

Jacopo Parri, Fulvio Patara, Samuele Sampietro, and Enrico Vicario(✉)

Department of Information Engineering, University of Florence, Florence, Italy
{jacopo.parri,fulvio.patara,samuele.sampietro,enrico.vicario}@unifi.it

Abstract. JARVIS is a Research & Development project, jointly developed by industrial SME partners and by the University of Florence, aimed at development of a hardware/software framework supporting integration among physical IoT devices, data analytic software agents, and human operators involved in operation and maintenance of resilient Industry 4.0 systems. At the heart of the JARVIS architecture, a suite of software digital twins deployed in a Java EE environment supports runtime monitoring and control of the hierarchy of hardware configuration items of the system, capturing their composition and representing their failure modes through a reflection architectural pattern enabling agile adaptation to the evolution of configurations. Besides, analytic modules can be deployed as micro-services leveraging both the knowledge base provided by digital twins and the data flowing from the ingestion layer. This enables agile development of advanced monitoring and control services supporting maintainability and resilience. We describe the JARVIS architecture, outlining responsibilities and collaborations among its modules, and we provide details on the structure of representation of digital twins, showing how this is exploited in a data analytic agent providing an executable representation of fault trees associated with failure modes of configuration items.

Keywords: I4.0 System of Systems · Digital twins · Fault tree

1 Introduction

In the agenda of Industry 4.0 (I4.0), Information Technology and Operational Technology are expected to provide facilities for conduction and maintenance of cyber-physical systems, developing on various pillars, including industrial IoT, big data and analytics, horizontal and vertical integration, cloud computing [23]. This gives rise to a class of software controlled distributed systems, for which resilience comprises a core requirement [1,19,24] shaping software engineering processes and architectural solutions.

R. Calinescu and F. Di Giandomenico (Eds.): SERENE 2019, LNCS 11732, pp. 85–93, 2019.
https://doi.org/10.1007/978-3-030-30856-8_6

JARVIS (Just-in-time ARtificial intelligence for the eValuation of Industrial Signals) is a project co-funded by the Tuscany regional government (Italy) in the POR FESR 2014–2020 program, developed by the industrial SME partners LASCAUX, SISMIC SISTEMI, JAEWA, and BEENOMIO, with the scientific support of the labs of Software Technologies, Artificial Intelligence, and Global Optimization of the University of Florence.

JARVIS aims at developing a hardware/software framework for integration, operation, and maintenance of Industry 4.0 systems, leveraging a software architecture that facilitates interaction among physical IoT devices, enterprise scale software agents, data analytics, and human operators, so as to support planning and scheduling of predictive maintenance and assets analysis, both offline and at runtime. On the one hand, a suite of software digital twins [25] deployed in the domain logic of a Java EE environment mirrors the hierarchical structure of physical devices in an IoT layer, enabling runtime monitoring and control of system hardware configuration items. On the other hand, a variety of data analytics and software agents drives agile development of advanced monitoring and control services for maintainability and resilience.

The project develops a framework open to reuse in the general context of Industry 4.0 systems, and validates its applicability through a concrete instance in a real operative scenario, addressing the case of a gate system for speed control and access regulation to limited traffic zones (ZTL), produced and manufactured by SISMIC SISTEMI, and installed in several Italian municipalities.

In this paper, we report a general description of the JARVIS project, outlining its major requirements (Sect. 2) and describing the architecture as a System of Systems (Sect. 3), and we then provide details, focusing on the structure of representation of digital twins and showing how this is exploited in a micro-service providing an executable representation of fault trees associated with failure modes of configuration items (Sect. 4). Conclusions are drawn in Sect. 5.

2 System Requirements Specification

JARVIS is an architecture-driven project, developed along a V-model process [9], documented according to the MIL-STD-498 [20].

Main system requirements and their consequent structural choices include: *(i)* the system must be able to ingest Big Data from a plethora of IoT devices, which led to the adoption of an IoT broker; *(ii)* the system must manage the persistence of raw and semi-structured data into a high capacity data-store, which led to the adoption of a schema-less NoSQL column-oriented DBMS [12]; *(iii)* the system must promote inversion of responsibility, by allowing actions and end-users notifications be triggered by edge and unmanned components on occurrence of faulty conditions; *(iv)* the system must provide an executable software representation of physical field devices with monitoring capabilities, which led to the design of a digital twins domain logic; *(v)* the system must exhibit elasticity in scaling up/down the computational power of single modules, which led to a micro-service oriented architecture [6]; *(vi)* the system must

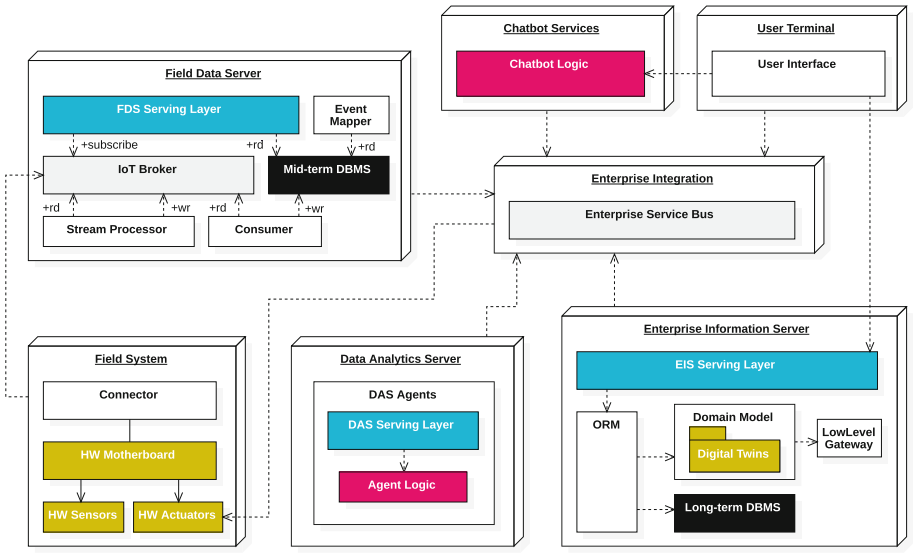

Fig. 1. UML deployment diagram of JARVIS architecture as a System of Systems.

be able to integrate applications, horizontally and vertically, along the production chain and among domains of authority, clients, customers, and tertiary manufacturers, which led to the adoption of Enterprise Application Integration (EAI) principles [15]; *(vii)* the system must integrate a swarm of data analytics and agents, supporting operations management and just-in-time maintenance processes, developed independently by different parties through paradigms of polyglot programming and polyglot persistence; *(viii)* the system must support push communications, providing an alternative multi-platform user interface, which led to the adoption of an ecosystem of chatbots [5].

In the specific focus of this paper, requirements *(iv)* and *(vii)* play a major role in the discussion.

3 System/Subsystem Design Description

JARVIS is developed around a System of Systems architecture (see Fig. 1), designed so as to promote high-levels of data ingestion, fault-tolerance, portability, and adaptability. Roles and responsibilities of subsystems are here explained in the general perspective of the project.

Field System (FS) acquires and generates IoT data flows, playing the role of perception layer [14] of an IoT architectural stack. An FS instance is a physical device composed of hardware components (e.g. motherboard, sensors and actuators) and software controllers (e.g. embedded firmware). In the JARVIS specific prototype, each FS represents a ZTL gate.

Field Data Server (FDS) stores raw data coming from the FS in a mid-term database, also applying analytic processes to filter, fix, and synthesise data. The IoT broker component, which acts as an asynchronous Message Oriented Middleware [3] based on the Publish-Subscribe EIP [13], performs ingestion of the IoT data streams.

Enterprise Information Server (EIS) maintains status information about monitored FSs, adopting the abstraction of digital twins, in order to maintain a long-term consistent knowledge base of field devices, interpreting and refining the mid-term FDS raw-data into a high level semantics.

Data Analytics Server (DAS) is composed by a plurality of agents executing dynamic context interpretation and processing, enabling descriptive, predictive, and prescriptive analysis (e.g. failure prediction and diagnoses), through artificial intelligence, machine learning mechanisms, and stochastic model techniques.

User Terminal (UT) interprets the role of decoupled presentation layer for the end-users. In the JARVIS specific prototype, this manages municipal authorities, municipal police officers, help desk operators and maintenance technicians.

Chatbot Services (CS) implements the internal logic of real-time messaging assistants, so as to expose an alternative UI which allows both push and pull duplex communications among human operators and physical devices, enabling inversion of responsibility mechanisms (i.e. machine-to-human and machine-to-machine interactions).

Enterprise Integration (EI) is responsible of subsystems interoperability, orchestrates services, and handles dynamic dependencies, authorizing and securing accesses to field devices. The core component is represented by the Enterprise Service Bus (ESB) [2], which guarantees high decoupling and push communications, also implementing some major micro-services patterns [16] to enhance availability and reliability, notably including: Circuit Breakers to limit fault propagations, Service Discovery and Gateways to route messages.

Overall, all these subsystems give rise to a so-called Lambda architecture [18], where the EIS implements the *batch layer*, the FDS serves as the *speed layer*, and the FDS, EIS and DAS jointly represent the *serving layer*. In the specific focus of this paper, EIS and DAS are the subsystems which cover the requirements *(iv)* and *(vii)*, respectively.

4 Digital Twins as Knowledge Base

The combination of EIS and DAS comprises a Knowledge Base supporting monitoring and control of resilience: the EIS provides a digital representation of structure and components of managed physical devices; besides the DAS hosts a variety of micro-services supporting operation and maintenance processes.

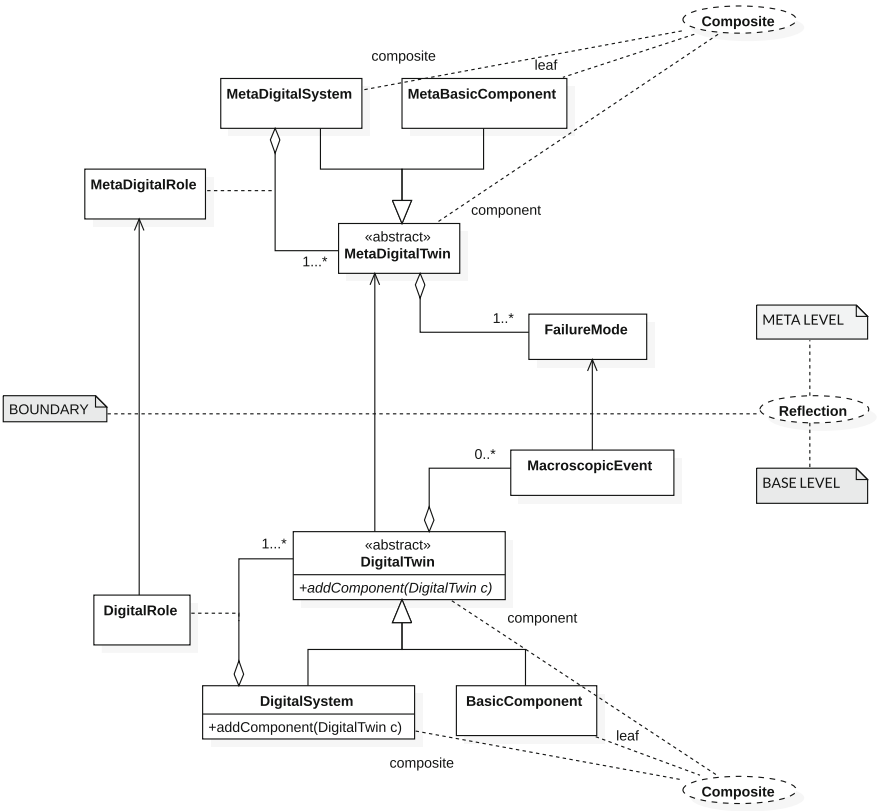

Fig. 2. UML class diagram of the domain model of EIS, showing the reflective and composite structure of digital twins. The association class *DigitalRole* (and its counterpart in the meta level) has been introduced to support same-typed and reusable components among different instances of *DigitalSystem* (and *MetaDigitalSystem*).

4.1 EIS Subsystem

The EIS subsystem is based on a domain logic, explicitly oriented toward reliability requirement, and populated by digital twins instances, whose focus is on capturing significant macroscopic events and failure modes, exhibited by whole physical systems or devices. Digital twin abstraction enables a two-way interaction on physical counterparts providing an interface to collect and query telemetries as well as to control remote actuators (e.g. reset command).

The domain model, depicted in Fig. 2, combines two software design patterns. The Reflection [22] pattern provides a mechanism to modify dynamically, at runtime, the structure and behaviour of modeled digital twins, by splitting the domain logic in two parts: the meta level captures the types of devices and their interconnections; the base level identifies concrete instances of physical components and their interfaces in the actual configuration of the system. Besides, the

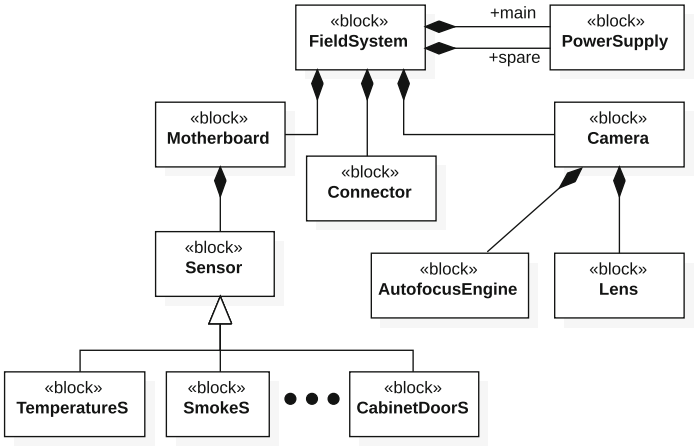

Fig. 3. SysML bdd of the field system of the ZTL gate comprising the specific prototype of the JARVIS project.

Composite [11] pattern is used to represent the hierarchical compositions of FS instances, in both the meta and base levels of the Reflection pattern.

The patterns combination enables the model to evolve so as to cope with different configurations of a product line and reliably adapt to changes in operation conditions. In particular, this permits to modify the compositional structure of some FS digital replica, allowing to plug new FS instances at runtime into the system, avoiding service unavailability due to EIS reboot.

The resulting software architecture promotes an engineering process where a specification of the structure of the system can be translated into an executable representation made of software digital twins.

Figure 3 illustrates the concept with reference to the configuration of the FS of a ZTL gate, here modeled as a SysML [10] Block Definition Diagram (bdd). Each block element of the bdd results into a *DigitalTwin* instance at runtime: basic and composed blocks are implemented as objects of type *BasicComponent* and *DigitalSystem*, respectively; *DigitalTwin* components of a *DigitalSystem* and their *DigitalRole*s can be derived from compositions and association role names, respectively; in so doing, roles permit to give identity to multiple instances of subsystems of the same type, as occurring in redundant configurations (e.g. the power supply in the example).

4.2 DAS Subsystem

DAS hosts a swarm of micro-services consuming information provided by the EIS Knowledge Base to support operation and maintenance processes through a variety of context-dependent techniques.

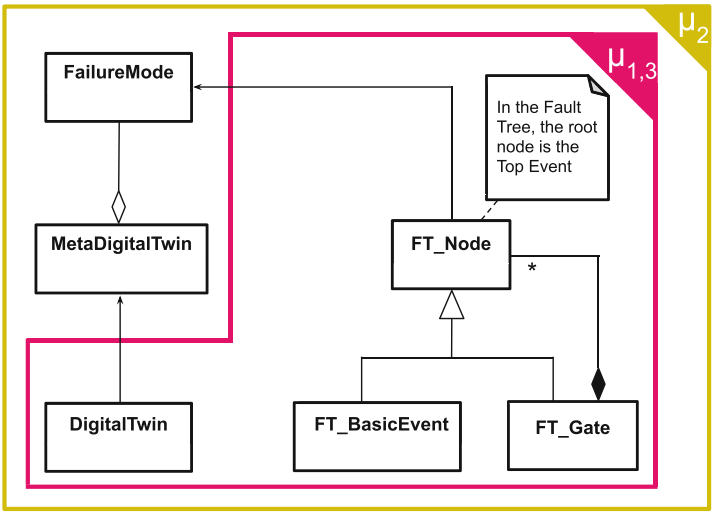

Fig. 4. UML class diagram of the *FT-agent* domain logic as μ_i bounded contexts.

We illustrate the concept with reference to an agent, termed *FT-agent*, which implements Fault Tree Analysis (FTA) [7] so as to enable diagnoses and predictions over system failures [21].

The *FT-agent* is partitioned in 3 micro-services: μ_1 performs analyses over a Fault Tree (FT), combining the outputs provided by the other two micro-services (in the specific project prototype, the task is achieved exploiting the modeling and analysis capabilities offered by the SIRIO Java library included in the ORIS Tool [17], a toolbox for quantitative evaluation of stochastic models); μ_2 exposes a collection of FTs capturing different failure modes of the FS, designed and managed by domain experts and maintenance technicians; μ_3 computes minimal cut-sets and importance measures (e.g. Birnbaum, Fussel-Vesely) over a FT.

Figure 4 represents the domain model of the *FT-agent*, distributed among the three micro-services following the Bounded Context pattern [8]. In the representation, the role of top, basic, or intermediate events in the FT is implemented by an object of type *FailureMode*.

Also in this case, the software architecture promotes an engineering process where models for reliability can be translated into an executable software representation. Specifically, identification of failure modes and their associations with digital twins can be conveniently guided by the artifacts of Failure Mode and Effects Analysis (FMEA) [4]. Additional concepts captured in Failure Mode, Effects, and Criticality Analysis (FMECA), may provide information about criticalities and probabilities, opening the way to the construction of executable quantitative models as data analytics.

5 Conclusions and Future Works

JARVIS is a hardware/software framework supporting operation and maintenance of Industry 4.0 systems. The framework is designed to integrate a System of Systems, allocating roles and responsibilities to components in a dynamic IoT scenario, where multiple operational devices need to be monitored at runtime to enable just-in-time maintenance.

Digital twins have been designed and adopted for representing conceptual composite structures of physical components, offering facilitation to monitor, manage and interact with operating instances, whose telemetries are ingested as IoT data streams and interpreted by analytic agents in order to detect and predict critical failures at runtime. Specifically, detected failures are collected into a high-level events register, held in the digital twins domain logic, and notified to maintenance technicians; instead, predicted failures enable self-healing mechanisms or extraordinary maintenance activities.

The JARVIS architecture promotes an engineering process for resilient systems from two orthogonal perspectives: on the one hand, a specification of the knowledge base of the system is mapped into an executable domain model made of software digital twins; on the other hand, reliability artifacts are translated into runnable failure models. These representations open the way to a variety of runtime monitoring and control services supporting maintainability and resilience.

The project will be completed by mid 2020 with experimentation in a concrete operation scenario, addressing the case of smart city gates for speed control and access regulation to limited traffic zones.

References

1. Abreu, D.P., Velasquez, K., Curado, M., Monteiro, E.: A resilient internet of things architecture for smart cities. Ann. Telecommun. **72**(1–2), 19–30 (2017)
2. Chappell, D.A.: Enterprise Service Bus. O'Reilly Media Inc., Sebastopol (2004)
3. Curry, E.: Message-oriented middleware. In: Middleware for Communications, pp. 1–28 (2004)
4. Department of Defense: MIL-STD-1629A - Procedures for performing a failure mode, effects and criticality analysis. Military Standard, Washington, DC (1980)
5. Di Prospero, A., Norouzi, N., Fokaefs, M., Litoiu, M.: Chatbots as assistants: an architectural framework. In: Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, pp. 76–86. IBM Corp. (2017)
6. Dragoni, N., et al.: Microservices: yesterday, today, and tomorrow. Present and Ulterior Software Engineering, pp. 195–216. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67425-4_12
7. Ericson, C.A.: Fault tree analysis. Syst. Saf. Conf. Orlando, Florida **1**, 1–9 (1999)
8. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, Boston (2004)

9. Forsberg, K., Mooz, H.: The relationship of system engineering to the project cycle. In: INCOSE International Symposium, vol. 1, pp. 57–65. Wiley Online Library (1991)
10. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann, Burlington (2014)
11. Gamma, E.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education India, New Delhi (1995)
12. Han, J., Haihong, E., Le, G., Du, J.: Survey on NoSQL database. In: 2011 6th International Conference on Pervasive Computing and Applications, pp. 363–366. IEEE (2011)
13. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, Boston (2004)
14. Khan, R., Khan, S.U., Zaheer, R., Khan, S.: Future internet: the internet of things architecture, possible applications and key challenges. In: 2012 10th International Conference on Frontiers of Information Technology, pp. 257–260. IEEE (2012)
15. Linthicum, D.S.: Enterprise Application Integration. Addison-Wesley Professional, Boston (2000)
16. Montesi, F., Weber, J.: Circuit breakers, discovery, and API gateways in microservices. arXiv preprint arXiv:1609.05830 (2016)
17. Paolieri, M., Biagi, M., Carnevali, L., Vicario, E.: The ORIS tool: quantitative evaluation of non-markovian systems. In: IEEE Transactions on Software Engineering (2019)
18. Parri, J., Sampietro, S., Vicario, E.: Deploying digital twins in a lambda architecture for industry 4.0. ERCIM News **115**, 30–31 (2018)
19. Pradhan, S., Dubey, A., Gokhale, A.: Designing a resilient deployment and reconfiguration infrastructure for remotely managed cyber-physical systems. In: Crnkovic, I., Troubitsyna, E. (eds.) SERENE 2016. LNCS, vol. 9823, pp. 88–104. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45892-2_7
20. Radatz, J., Olson, M., Campbell, S.: Mil-std-498. Crosstalk J. Defense Softw. Eng. **8**(2), 2–5 (1995)
21. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. ACM Comput. Surv. (CSUR) **42**(3), 10 (2010)
22. Schmidt, D.C., Stal, M., Rohnert, H., Buschmann, F.: Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects, vol. 2. Wiley, Hoboken (2013)
23. Shrouf, F., Ordieres, J., Miragliotta, G.: Smart factories in industry 4.0: a review of the concept and of energy management approached in production based on the internet of things paradigm. In: 2014 IEEE International Conference on Industrial Engineering and Engineering Management, pp. 697–701. IEEE (2014)
24. Suciu, G., Vulpe, A., Halunga, S., Fratu, O., Todoran, G., Suciu, V.: Smart cities built on resilient cloud computing and secure internet of things. In: 2013 19th international conference on control systems and computer science. pp. 513–518. IEEE (2013)
25. Weippl, E.R., Sanderse, B.: Digital twins - introduction to the special theme. ERCIM News **2018**(115), 6–7 (2018)