Eclipse SAM IoT 2020
Security | AI | Modelling

1st Eclipse Research International Conference on Security, Artificial Intelligence and Modelling for the next generation Internet of Things

PROCEEDINGS

17–18 September, 2020

EDITORS

Enrico Ferrera
Philippe Krief

CO-ORGANIZED BY

Eclipse Foundation, Germany
LINKS Foundation, Italy

SUPPORTED BY

BRAIN-IoT EU H2020 Project
IoT European Security and Privacy Projects

SAM IoT 2020

Proceedings of the
1st Eclipse Research International Conference on Security, Artificial Intelligence and Modelling for the next generation Internet of Things

Virtual Conference | September 17-18, 2020

CO-ORGANIZED BY

Eclipse Foundation, Germany
LINKS Foundation, Italy

SUPPORTED BY

BRAIN-IoT, EU H2020 Project, Grant agreement 780089

Edited by Enrico Ferrera and Philippe Krief

Submitted by Enrico Ferrera

Published on ceur-ws.org

ISSN 1613-0073

https://events.eclipse.org/2020/sam-iot/

samiot2020-chairs@edas.info

BRIEF CONTENTS

ORGANIZING COMMITTEES

GENERAL CHAIRS

Enrico Ferrera, LINKS Foundation, Italy
Philippe Krief, Eclipse Foundation, Germany

PROGRAM COMMITTEE CHAIR

Rosaria Rossini, LINKS Foundation, Italy

LAYOUT DESIGNER

Ilaria Bosi, LINKS Foundation, Italy

PUBLICITY

Susan Iwai, Eclipse Foundation, Germany

INVITED SPEAKERS

Henrik Plate
SAP Security Research
Germany

Paul-Emmanuel Brun
Airbus Cybersecurity
France

PROGRAM COMMITTEE

FOREWARD

The adoption of the Internet of Things (IoT) is drastically increasing in every application domain, contributing to the rapid digitalization of contemporary society. Current IoT scenarios are characterized by constantly increasing demands in terms of non-functional requirements, from low latency to high reliability, dependability, and dynamic resources allocation. This paradigm shift, also considered as the next evolutionary phase of IoT, is expected to create numerous opportunities for the technology market supporting applications in multiple areas, i.e. Smart Factories, Smart Cities, Critical Infrastructures, Cooperative Service Robotics, etc. To cope with these demanding requirements, a multitude of novel technologies - such as Edge Computing, Artificial Intelligence and Analytics, Digital Twin, as well as Security, Privacy and Trust schemes – are being investigated in order to be adopted in current IoT architectures standards, identifying efficient integration schemes with proper design patterns. Hence, designing and managing the next generation of IoT-based systems is set to become even more complex.

This book contains the proceedings of the 1st Eclipse International Conference on Security, Artificial Intelligence and Modeling for the next-generation Internet of Things (SAM IoT 2020).
SAM IoT 2020 is the first scientific conference organized by Eclipse Foundation with the aim of promoting the building of a richer public domain culture within the research community, with special attention to applied research. SAM IoT 2020 has organized a call for papers to collect the latest research results in Europe and all around the world, with a specific focus on the open issues related to Security, Artificial Intelligence and Modelling in the next-generation of Internet of things applications. SAM IoT is also supported by the EU-funded H2020 BRAIN-IoT project. As a project focused on the definition and implementation of novel architectures and methodologies for supporting the developers and operators of modern IoT applications to deal with the increasing complexity and dynamicity of IoT systems in Smart city, Industry and Robotics domains, BRAIN-IoT is among the pioneer projects on the Next-Generation IoT paradigm. For this reason, LINKS Foundation, as coordinator of the BRAIN-IoT project, participates in the organization of SAM IoT 2020 to promote the discussion around the Next-Generation IoT research topics, bringing together participants from research and industry.

Submissions, with authors from 16 different countries spread across Europe, Asia and the United States have been received. To evaluate each submission, a blind paper review was performed by the Technical Program Committee, whose members are highly qualified researchers in SAM IoT topic areas. Each paper was reviewed by at least three reviewers. Based on those reviews, papers that adequately balanced quality, originality and relevance to the conference themes were selected. Based on the classifications provided, 11 papers have been selected.
The conference also featured 2 keynote lectures delivered by experts, namely Henrik Plate (SAP Security Research) and Paul-Emmanuel Brun (AIRBUS CyberSecurity). These talks contributed to increasing the overall quality of the conference and to provide a deeper understanding of the conference fields of interest.
The proceedings of SAM IoT 2020 will be submitted for publication to CEUR Workshop Proceedings (CEUR-WS.org), which is a free open-access publication service at Sun SITE Central Europe operated under the umbrella of RWTH Aachen University. CEUR-WS.org is a recognized ISSN publication series, ISSN 1613-0073.
We believe the proceedings published demonstrate new and innovative solutions, and highlight challenging technical problems in each of the SAM IoT fields.

To recognize the best submission, an award based on the paper review scores, as assessed by the Technical Program Committee was conferred at the closing session of the conference.
As a final point, we would like to express our thanks, first of all, to the authors of the technical papers, whose work and dedication made it possible to put together a program that we believe is very exciting and of high technical quality. Next, we would like to thank all the members of the program committee, who helped us with their expertise and time.

We would also like to thank the invited speakers for their invaluable contributions and for sharing their vision in their talks. Finally, we acknowledge the professional support of the SAM IoT 2020 team for all organizational processes, especially given the need to introduce online streaming, breakouts management, direct messaging facilitation and other web-based activities in order to make it possible for SAM IoT 2020 authors to present their work and share ideas with colleagues in spite of the logistic difficulties caused by the current pandemic situation.

Enrico Ferrera
LINKS Foundation, Italy

Philippe Krief
Eclipse Foundation, Germany

CONTENTS

Cobbles and Potholes – On the Bumpy Road to Secure Software Supply Chains

Henrik Plate
SAP Security Research, Germany

Abstract — Open source software is ubiquitous – all across the stack, in the cloud and on-premise, on all devices, in commercial and non-commercial offerings. This success, the dependency of the software industry on open source, combined with recent data breaches and attacks, puts security into the spotlight. This talk will provide an overview - for sure opinionated, hopefully controversial – about the state of affairs and current trends regarding the security of software supply chains, both from consumer and producer perspective.

Brief Biography — Henrik Plate is a senior researcher at SAP Security Research. He received his MSc in Computer Science and Business Administration in 1999 from the University of Mannheim. His current research focusses on the security of software supply chains, esp. the use of open source components with known vulnerabilities and supply chain attacks. He is a co-author of Eclipse Steady [5], which supports the detection, assessment, and mitigation of vulnerable open source dependencies in Java and Python applications.

Securing Low Power Device Communication in Critical Infrastructure Management

Paul-Emmanuel Brun
Airbus Cybersecurity,France

Abstract — An overview of a secure IoT data transmission ecosystem will be completed with a concrete example of a water management use case from the Brain-IoT project. In this use case, to ensure high trust in the device's data, we integrated an end-to-end security layer that is compatible with battery-less devices with high constraints in terms of energy, power computation and bandwidth.

Brief Biography — Passionate about cybersecurity, IoT and identity management, Paul-Emmanuel Brun, is expert in IoT system security, and leading the innovation activities within AIRBUS CyberSecurity France. As a former security and identity management engineer, he was involved in several European initiatives and contributes to several projects for the French MoD, from secure architecture definition to integration of cybersecurity solution. After applying several patents linked to the cybersecurity of IT and IoT systems, he is now focus on innovation and IoT system security.

Risk Assessment in IoT Case Study: Collaborative Robots System

Salim Chehida, Abdelhakim Baouya
University of Grenoble Alpes, CNRS, VERIMAG F-38000
Grenoble, France
{name.surname}@univ-grenoble-alpes.fr

Paul-Emmanuel Brun, Guillemette Massot
Airbus CyberSecurity SAS
Elancourt, France
{name.surname}@airbus.com

Miquel Cantero
Robotnik Automation S.L.L
Valencia, Spain
mcantero@robotnik.es

***Abstract*—Security is one of the crucial challenges in the design and development of IoT applications. This paper presents an approach that focuses on existing security standards to evaluate and analyse the potential risks faced by IoT systems. It begins by identifying system assets and their associated vulnerabilities and threats. A list of security objectives and technical requirements are then defined to mitigate the risks and build a secure and safe system. We use our approach to assess risks in the robotic system for supporting the movement of loads in a warehouse.**

***Index Terms*—Security Risk Assessment, IoT, Threats, Security Requirements.**

I. Introduction

Internet of Things (IoT) is a promising technology that offers significant improvements to various domains such as health, commerce, construction, buildings management, energy, and transport. It reduces management costs, automates the monitoring of infrastructures and equipment, saves energy, and more. An IoT system consists of a network of smart devices that collaborate with users to accomplish intelligent services. It generally groups a large number of devices that interact using multiple communication technologies and protocols.

In the last decade, IoT systems are increasingly susceptible to various security issues, such as malicious access to services and network attacks. These problems have caused considerable damage and affected the secrecy, integrity, and availability of information. There are several surveys, such as [1]–[4], that discuss vulnerabilities that can be exploited by attackers to damage IoT systems. Taking into account these risks and their possible consequences constitute one of the principal challenges for the designer and developer of these systems.

Security Risk Assessment (SRA) is the process that aims to identify the most critical threats and provide the required measures to avoid these threats. It aims to mitigate the risks and build a secure system while covering its vulnerabilities. Several SRA methodologies [5]–[9] have been proposed to evaluate risks and enforce a common level of security. However, these methods are generic, and they do not consider the complexity and the dynamic of IoT systems.

In this work, we present a new approach that considers existing methodologies and standards for risk assessment in IoT systems. It starts by identifying the assets that should be protected and evaluating the threats they face. Then, a list of security objectives and requirements are defined to defend the system against potential threats. We apply our approach to the collaborative robots system. Our approach is different from all the generic approaches mentioned above and presented in Section II. It is dedicated to IoT systems and takes into account the relevant domain model and standards, as well as the need for evolution of these systems.

This paper is organized as follows: Section II presents the main approaches and standards for security assessment. We give an overview of our risk assessment approach in section III, then we describe its different stages and apply them to our case study in sections IV to VI. Finally, we give our conclusions in Section VII.

II. State of the art

We first present the main security standards, then the existing methods for risk assessment.

A. Security Standards

Security standards guide an organization in best security practices in order to enforce a common level of security by ensuring availability, integrity, and confidentiality requirements. Many countries and organizations have established standards for risk assessment and analysis. In this section, we briefly present the relevant common and IoT security standards.

(a) *Common Standards*

- ISO/IEC 27002 [10]: International standard that gives general guidance on the commonly accepted goals of information security management. It describes general principles structured around 36 security objectives and 133 controls.

- AS/NZS 4360 [11]: The joint Australian/New Zealand risk management standard that provides a generic framework for identifying, analysing, evaluating, treating, monitoring, and communicating risk.
- ISO/IEC 27005 [12]: International standard that provides guidelines for managing information security risks in an organization. The standard describes the risk management process, which includes context establishment, risk assessment, risk treatment, risk acceptance, risk communication, and risk monitoring and review.
- BS7799 (ISO17799) [13]: British Standard (Code of Practice for Information Security Management), evolved into ISO17799 (The Information Security Standard). It gives a basis guide for risk assessment and information security management.
- NIST SP 800-30 [14]: Special Publications Risk Management Guide for Information Technology Systems standard that provides practitioners with practical guidance for carrying out each of the three steps in the risk assessment process (i.e., prepare for the assessment, conduct the assessment, and maintain the assessment). It also discusses how organizational risk management processes complement and inform each other.
- NIST SP 800-82 [15]: This standard guides on improving security in Industrial Control Systems (ICS), including Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC).
- IEEE 1686 [16]: Standard for Intelligent Electronic Devices Cyber Security Capabilities' that defines functions and features to be provided in Intelligent Electronic Devices (IEDs). The document addresses access, operation, configuration, firmware revision, and data retrieval of an IED.

(b) *IoT Security Standards*
The authors in [17] analyse the existing regional and international standards for IoT security and indicate their limitations. Among international standards:

- ITU-T standards[1] :
  - Y.2060 provides reference models of IoT and shows generic security capabilities on every layer.
  - Y.2063 covers the authorization of heterogeneous devices of WoT.
  - Y.2066 defines common requirements of IoT and also security and privacy protection requirements related to all the IoT actors.
  - Y.2067 covers gateway security mechanisms including authentication, data encryption, privacy protection, etc.
  - Y.2068 defines concepts of functional framework and capabilities of IoT, including service provision security, security integration, security audit, etc.
  - Y.2075 specifies the security capabilities of EHM (e-health monitoring) with IoT.
  - Y.4112/Y.2077 specifies the concept, purpose, and components of plug and play (PnP) capability of the IoT, including security-related requirements.
  - Y.4553 specifies the requirements of the smartphone as a sink node for IoT applications, including authentication and data protection capabilities.
  - Y.4702 provides common requirements and capabilities of device management (DM) in IoT, including security management capabilities such as security event detection and reporting, device security assurance, and device security control.
- ISO/IEC standards: ISO/IEC 30128 [18] covers IoT security related to sensor network application interface.

Among regional standards, ETSI (standards organization in the telecommunication industry in Europe) recently provided "ETSI TS103645" [19] (Cyber Security for Consumer Internet of Thing) standard that gives security practices for consumer devices connected to the Internet.

According to [17], most of the IoT security standards presented above are just specification-level standards and a few of them are involved in availability and non-repudiation.

[1]https://www.itu.int/en/ITU-T/Pages/default.aspx

B. Risk Assessment Methods

EBIOS [9] is used for the assessment and treatment of risks associated with an Information System (IS). Its steps are: definition of the context, identification and estimation of the security needs and eventual sources of threats, identification and analysis of threat scenarios, and finally specification of security objectives and measures to be implemented for risk treatment. The goal of the EBIOS method is to create a common ground for security discussion between various stakeholders in order to support management-level decision-making. One of the main strengths of the EBIOS approach is its modularity; its knowledge bases can be tuned to comply with local standards and best practices, and to include external repositories of attack methods, entities or vulnerabilities [20].

CRAMM [7] (CCTA Risk Analysis and Management Method) is a qualitative risk assessment methodology that consists of the following steps: collection of data and definition of objectives, identification and evaluation of system assets, threat and vulnerability assessment, and finally determining countermeasures.

AURUM [5] (Automated Risk and Utility Management) supports the NIST SP 800-30 standard [14]. It consists of the following steps: identification of risks and their impacts, implementation of adequate countermeasures, and evaluation of the impact of countermeasures.

CORAS [6] allows risk assessment, documentation of intermediate results, and presentation of conclusions. The main steps of the methodology are: definition of security goals, description of threats, risk estimation by giving likelihood values for identified unwanted incidents, and risk treatment.

MEHARI [8] (MEthod for Harmonized Analysis of RIsk) aims to provide a risk management model compliant to ISO-27005 [12]. The steps of MEHARI are: establishment of the organization context, identification and classification of assets, identification and analysis of risks, and finally quantification and management of risks. MEHARI allows the analysis of the security stakes and the preliminary classification of the IS entities according to three basic security criteria (confidentiality, integrity, and availability).

OCTAVE [21] (Operationally Critical Threat, Asset, and Vulnerability Evaluation) method allows to define a risk-based strategic assessment and planning technique for system security. It is based on process broken into three phases : development of initial security strategies, identification of infrastructure vulnerabilities, and development of final security strategy and plans.

IT-Grundschutz [22] provides methods, processes, procedures, and measures to establish a system for information security management. It describes a two-tier risk assessment: one is designed for reaching a *standard* level of security, while a second *supplementary risk analysis* can be undertaken by companies that desire an approach customized to their specific needs or sector or that have special security requirements. IT-Grundschutz also provides lists of relevant threats and required countermeasures that can be adapted to the needs of an organization.

III. An outline of our Methodology

Starting from standards and methods presented in the previous section, we define the risk assessment methodology depicted in Figure 1.

Our method consists of four steps:

1) The first step identifies the assets based on the IoT domain model.
2) The second step specifies threats on the assets based on common threats database proposed by the risk assessment methods presented in Section II. In this work, we consider EBIOS database [9], which is compatible with all relevant ISO standards and provides a complete list of possible threats (42 threats) relative to information systems. EBIOS threats database is widely used in risk assessment. Some works like [23] have used it for risk analysis of IoT systems.
3) In the third step, security objectives are derived from the threats. In this step, we extract relevant objectives (13 objectives) for IoT systems from ISO-27002 [10] that provides a set of generic security objectives supported by a set of controls that are an important part of information security management.
4) In the last step, security requirements are built in order to implement the security objectives and provide countermeasures of the identified threats.

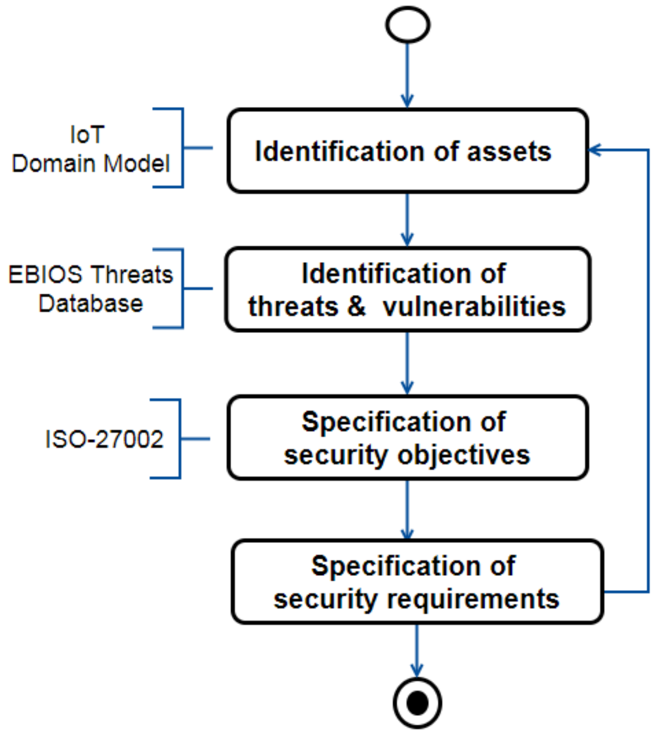

IoT Domain Model
Identification of assets
EBIOS Threats Database
Identification of threats & vulnerabilities
ISO-27002
Specification of security objectives
Specification of security requirements

Fig. 1. IoT Risk Assessment Methodology.

Our approach is iterative, and security requirements can be revised after the system assets have been refined. The results of each step should be checked with the customer.

In this work, we apply our method to the service robotics system. As shown in Figure 2, our system consists of a fleet of robots installed in a warehouse to support the movement of different loads.

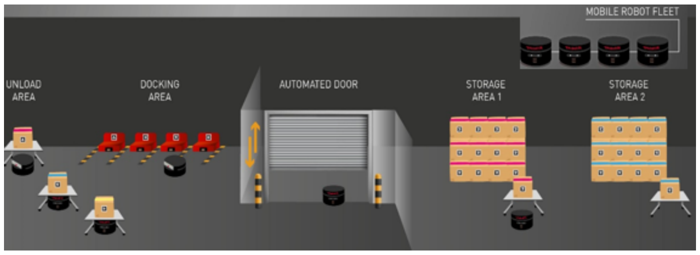


Fig. 2. Service Robotics System.

The flow of these loads does not require any operator to command the fleet. Robots are expected to empty continuously an "unload area" where different loads are put together. At some point, the system needs to identify the different items and then asks a specific robot to pick it and place it in a specific storage area following some predefined rules. It is also foreseen that in order to perform such activity, the system will need to actuate IoT devices, for example, an automated door in the middle of the robot's path to "storage areas".

IV. Identification of assets

ISO-27001 [24] defines an asset as "*any tangible or intangible thing or characteristic that has value to an organization*". In our approach, we refer to IoT domain model proposed by [25] to facilitate the identification of the system assets. In this model, the main concepts are: *thing*, *device*, *user* and *resource*.

As shown in Figure 3, *Thing* is the combination of PE (Physical Entity) together with its digital representation VE (Virtual Entity).

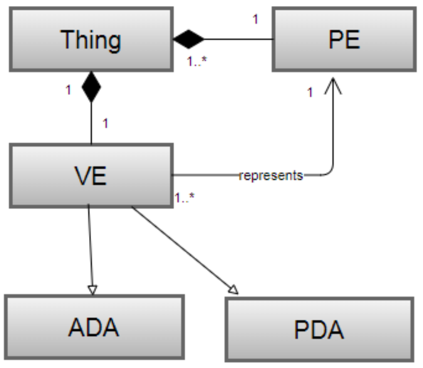


Fig. 3. IoT Things.

VE can be of both types:

- Passive Digital Artefact (PDA): a digital representation of PE stored in a database or similar form.
- Active Digital Artefact (ADA): any type of active code or software program usually be some sort of software agent or embedded application.

Device is a hardware with computing and network capabilities that allows to monitor or interact with PE. As shown in Figure 4, device can be:

- Sensor : allows to monitor PE.
- Actuator : allows to act on PE.
- Tag : allows to identify PE and can be read by sensors.

User represents who interacts with PE physically or through software interfaces. Users can either be humans or ADA.

Resource is software components that can provide information about PE, allow the execution of actuation tasks, or analyse data provided by multiple sensors. Resources may be hosted on a Device, or they could be hosted anywhere in the network.

Table I presents examples of 16 assets identified in our case study. The system includes different types of devices, such as sensors (e.g., A3, A4, A5) and actuators (e.g., A13, A14, A15).

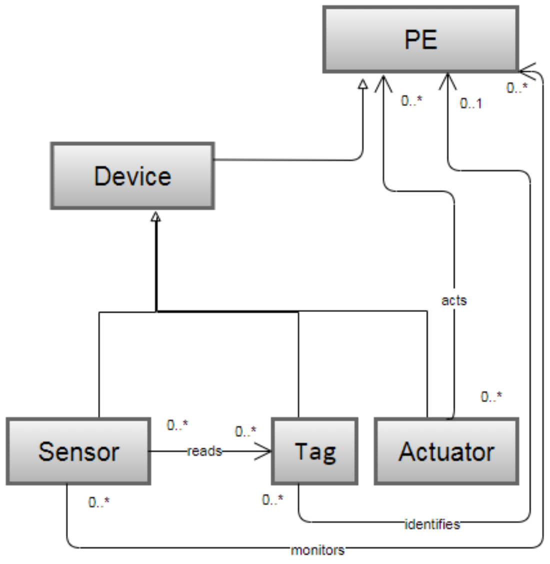


Fig. 4. IoT Devices.

Asset ID	Asset Description
A1	Mobile Robot: Embedded Computer
A2	Mobile Robot: Motion Control (motor driver)
A3	Mobile Robot: Sensor 1, RGBD Camera
A4	Mobile Robot: Sensor 2, Lidar
A5	Mobile Robot: Sensor 3, Odometry
A6	Mobile Robot: Lift Mechanism
A7	Mobile Robot: Battery (LiFePo)
A8	Mobile Robot: Network (Card)
A9	System: User Computer
A10	System: Network (Router and infrastructure)
A11	System: Mission Command (Outwards)
A12	System: Robot State (Inwards)
A13	Door PLC
A14	PLC WiFi Gateway
A15	PLC: Opening order (Inwards)
A16	Operator HMI

TABLE I
Robots System Assets.

V. Identification of threats and vulnerabilities

ISO-27001 [24] defines a threat as "*a potential cause of an unwanted incident, which may result in harm to a system or organization*" and considers vulnerability as "*weakness that is related to the organizations' assets, which sometimes could cause an unexpected incident*".

As mentioned in Section III, our method considers a list of generic threats from EBIOS database. In Table II taken from the EBIOS knowledge bases, threats are classified into eight main categories:

- Physical damage: T-1010 to T-1050.
- Natural events : T-2010 to T-2050.
- Loss of essential services : T-3010 to T-3030.
- Disturbance due to radiation : T-4010 to T-4030.
- Compromise of information : T-5010 to T-5110.
- Technical failures : T-6010 to T-6050.
- Unauthorized actions : T-7010 to T-7050.

ID	Threats Description	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
T-1010	Fire	X	X	X	X	X	X	X	X	X	X			X	X		
T-1020	Water damage	X	X	X	X	X	X	X	X	X	X			X	X		
T-1030	Pollution	X	X	X	X	X	X	X	X	X	X			X	X		
T-1040	Major Accident	X	X	X	X	X	X	X	X	X	X			X	X		
T-1050	Destruction of equipment or media	X	X	X	X	X	X	X	X	X	X			X	X		
T-2010	Climatic Phenomenon	X	X	X	X	X	X	X	X	X	X			X	X		
T-2020	Seismic Phenomenon	X	X	X	X	X	X	X	X	X	X			X	X		
T-2030	Volcanic Phenomenon	X	X	X	X	X	X	X	X	X	X			X	X		
T-2040	Meteorological Phenomenon	X	X	X	X	X	X	X	X	X	X			X	X		
T-2050	Flood	X	X	X	X	X	X	X	X	X	X			X	X		
T-3010	Failure of air-conditioning	X	X							X				X			
T-3020	Loss of power supply	X	X	X	X	X	X	X	X	X	X			X	X		
T-3030	Failure of telecommunication equipment	X							X		X	X	X		X	X	X
T-4010	Electromagnetic radiation								X		X	X	X		X	X	X
T-4020	thermal radiation	X	X	X	X	X	X	X	X	X	X			X	X		
T-4030	Electromagnetic pulses	X	X	X	X	X	X	X	X	X	X			X	X		
T-5010	Interception of compromising interference signals										X	X	X		X	X	X
T-5020	remote spying			X													X
T-5030	eavesdropping	X							X		X	X	X		X	X	
T-5040	Theft of media or documents													X	X		
T-5050	Theft of Equipment	X	X	X	X	X	X	X	X	X	X			X	X		
T-5060	Retrieval or recycled or discarded media																X
T-5070	disclosure																X
T-5080	data from untrustworthy sources	X											X				X
T-5090	Tampering with hardware	X	X	X	X	X	X	X	X	X	X			X	X		
T-5100	Tampering with software	X	X							X		X		X	X	X	X
T-5110	Position detection				X	X											
T-6010	Equipment failure	X	X	X	X	X	X	X	X	X	X			X	X		
T-6020	Equipment malfunction	X	X	X	X	X	X	X	X	X	X			X	X		
T-6030	Saturation of the information system	X														X	X
T-6040	Software malfunction	X								X							X
T-6050	Breach of information system maintainability	X								X				X			X
T-7010	Unauthorised use or equipment	X								X							X
T-7020	Fraudulent copying of software	X								X				X			X
T-7030	use of counterfeit or copied software	X								X				X			X
T-7040	corruption of data	X							X	X	X			X	X		X
T-7050	Illegal processing of data	X		X					X	X	X			X	X		X
T-8010	Error in use	X							X	X	X			X	X		X
T-8020	Abuse of rights	X							X	X	X			X	X		X
T-8030	Forging of rights	X							X	X	X			X	X		X
T-8040	Denial of actions	X								X				X			X
T-8050	Breach of personnel availability	X								X				X			X

TABLE II
THREAT-ASSET MATRIX.

- Compromise of functions :T-8010 to T-8050.

The threat factors can be divided into two categories:

- *Environment factors* such as earthquakes or floods, cannot be avoided. The risk manager should always consider environment threats according to their operating environment, even if it is difficult to consider them.
- *Human factors*, which are more of our concern because they are vagrant regarding different people and different situations, and it is more difficult to predict human behavior than regular natural disasters. We distinguish persons who belong to the organization like different users of the system and persons from outside the organization such as recipient, provider, and competitor.

In Table II, we show the threats associated to each asset presented in Table I.

VI. Specification of security objectives and requirements

In this step, we based on ISO-27002 [10] generic list to specify security objectives needed to protect the system assets against the identified threats. We also map each security objective with the threat list. Table III gives an example of security objectives that cover the most potential threats presented in the previous step.

After the specification of security objectives, we define security requirements. In Table IV, each security objective from Table III leads to the implementation of one or more technical requirements.

ID	Security Objective	Security Objective Description	Threats
O1010	Protection Against Malicious Code	Prevent and detect the allocation of any malicious code, as well as connections of any unprivileged user to the robot network	T-50xx
O1020	Backup	The data from the initial robot setup and the robot firmware require regular backup	T-10XX T-20XX
O1030	Network Security Management	Protect the information and communication in network from a client to robot. Sending REST Command once authenticated in the same network can modify the operations	T-5030 T-5090 T-7010 T-7020 T-7040
O1040	Exchange of information	Secure the interaction between the platform and robot system	T-5070 T-5080
O1050	Monitoring	Logs and robot system state shall be secured to prevent a bad usage (i.e. a door opened)	T-5030 T-5040 T-60xx T-70xx T-80xx
O2010	User Access Management	Authentication and authorization of the robot and any user or system accessing the robot	T-7010 T-7020 T-7040 T-8020 T-8030
O2020	Network Access Control	Prevent unauthorized use of robot network services	T-6030 T-70xx
O2030	Operating System Access Control	Rely on the access control mechanism offered by Ubuntu	T-8020 T-8030 T-8040
O3010	Correct processing in applications	Check any command received by the robot and the processing status of the robot. No robot shall accept commands out of reach by itself	T-60xx
O3020	Cryptographic controls	Protect the sensible information in the robot network and also the authentication operations of the users or systems accessing the robot	T-8020 T-8030
O3030	Security of system files	Rely on the security mechanisms and limitation rules offered by Ubuntu to protect the system files	T-8020
O3040	Security in Development and support process	Control of information flow and integrity in robot systems	T-6040 T-6050 T-8040 T-8050
O3050	Technical vulnerability management	Detect and deal with the technical vulnerabilities to reduce the risks such as physical interfacing of robots.	T-6020 T-6040

TABLE III
Security Objectives of Service Robotics System

Objective ID	Requirement ID	Requirements Description
O-1010	R-1010-0010	REST API must detect malformed commands
	R-1010-0020	Access to the REST API must be authenticated
	R-1010-0030	Robot firewall should block all the connection except SSH
	R-1010-0040	SSH connection should be restricted to unprivileged users
O-1020	R-1020-0010	Robot firmware should be stored in a non-erasable memory
O-1030	R-1030-0010	Network access must require authentication
	R-1030-0020	Network communication from a client with a robot must be authenticated and encrypted
O-1040	R-1040-0010	Communication from platform to robot must be authenticated and encrypted (e.g: using protocol like TLS1.2 minimum)
O-1050	R-1050-0010	Access to log information must be limited to authorized person only
O-2010	R-2010-0010	System account management (right, password, creation, deletion, ...) should be done in a central application (to avoid account / password duplication and error in duplicated right management system)
	R-2010-0020	User (or technical account) password should be at least 12 characters, with at least one upper case, lower case, number and special character)
O-2020	R-2020-0010	Network equipment should implement network access control (e.g: 802.1.X)
O-2030	R-2030-0010	Sudo account should be blocked
	R-2030-0020	Sudoers rules should be set up according to the system privileged action to perform
O-3010	R-3010-0010	Commands received by the robot should be parsed and checked using whitelist approach
	R-3010-0020	The robot should monitor its processing status (to avoid overprocessing)
O-3020	R-3020-0010	Authentication operation should be performed using cryptographic signature (at least SHA256 combined with RSA or ECC algorithms)
	R-3020-0020	Operating system integrity should be guarantee using cryptographic proof (signature) securely stored (e.g: TPM)
O-3030	R-3030-0010	File systems access must be limited to authenticated and allowed users (or technical account)
	R-3030-0020	File systems should be encrypted
O-3040	R-3040-0010	Source code and binaries should be signed to ensure their integrity
	R-3040-0020	Binaries compilation should be done using hardening arguments (memory randomization, . . .)
O-3050	R-3050-0010	Software vulnerability should be managed
	R-3050-0020	Outdated packaged should be upgradable

TABLE IV
SECURITY REQUIREMENTS OF SERVICE ROBOTICS SYSTEM

VII. CONCLUSION

In this paper, we have tackled the highly vast subject of IoT systems security while concentrating on risk assessment. The proposed approach provides several advantages, including:

- It considers IoT domain model to identify all system assets.
- It follows relevant security standards to define security requirements.
- It is an iterative approach and responds to the need for evolution of IoT systems.

We have applied this methodology to a robotic system that supports the movement of loads in the warehouse. We started by identifying the critical assets and the potential threats that might compromise them. Then, we defined the technical requirements considering the identified threats and a list of security objectives extracted from a common database. All the steps of our approach was understandable and easy to follow by the case study owners and several threats related to the target infrastructure not previously considered were discovered in this study.

In the analysis performed in this paper, we have taken into account all system assets and a complete list of possible threats taken from the standards, which allows us to identify all potential risks and the requirements needed to mitigate those risks.

After the specification of security requirements, appropriate countermeasures can be deployed to protect the system against the identified risks. There are also approaches such as [26] that helps security experts to determinate impactful and adequate countermeasures considering organization defense budget.

In future work, we plan to apply our method to other systems. We also plan to support our approach with a tool that automates the various analysis activities.

Acknowledgment

The research leading to these results has received funding from the European Union through the BRAIN-IoT project H2020-EU.2.1.1. Grant agreement ID: 780089.

References

[1] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, Jan. 2015.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[3] J. Sengupta, S. Ruj, and S. Das Bit, "A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT," *Journal of Network and Computer Applications*, vol. 149, p. 102481, Jan. 2020.

[4] P. I. Radoglou Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, threats and solutions," *Internet of Things*, vol. 5, pp. 41–70, Mar. 2019.

[5] A. Ekelhart, S. Fenz, and T. Neubauer, "Aurum: A framework for information security risk management," in *2009 42nd Hawaii International Conference on System Sciences*, 2009, pp. 1–10.

[6] F. den Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen, "Model-based security analysis in seven steps — a guided tour to the CORAS method," *BT Technology Journal*, vol. 25, no. 1, pp. 101–117, Jan. 2007. [Online]. Available: http://link.springer.com/10.1007/s10550-007-0013-9

[7] Z. Yazar, "A qualitative risk analysis and management tool–CRAMM," *SANS InfoSec Reading Room White Paper*, vol. 11, pp. 12–32, 2002.

[8] "MEHARI: MEthod for Harmonized Analysis of RIsk," 2010. [Online]. Available: https://en.wikipedia.org/wiki/MEHARI

[9] The National Cybersecurity Agency of France (ANSSI), *EBIOS 2010 - Expression of Needs and Identifiation of Security objectives.*, 2010. [Online]. Available: https://www.ssi.gouv.fr/guide/ebios-2010-expression-des-besoins-et-identification-des-objectifs-de-securite/

[10] ISO/IEC 27002:2013. (2013) Information technology — Security techniques — Code of practice for information security controls. [Online]. Available: https://www.iso.org/standard/54533.html

[11] AS/NZS 4360-2004. (2004) Risk management. [Online]. Available: https://www.standards.org.au/standards-catalogue/sa-snz/publicsafety/ob-007/as-slash-nzs–4360-2004

[12] ISO/IEC 27005:2011. (2011) Information technology — Security techniques — Information security risk management. [Online]. Available: https://www.iso.org/standard/56742.html

[13] ISO/IEC 17799:2005. (2005) Information technology — Security techniques — Code of practice for information security management. [Online]. Available: https://www.iso.org/standard/39612.html

[14] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems," *Nist special publication*, vol. 800, no. 30, pp. 800–30, 2002.

[15] K. Stouffer, J. Falco, and K. Scarfone, "Nist special publication 800-82, guide to industrial control systems (ics) security," *NIST Special Publication*, pp. 800–882, 01 2011.

[16] IEEE 1686. (2013) IEEE Standard for Intelligent Electronic Devices Cyber Security Capabilities. [Online]. Available: https://standards.ieee.org/standard/1686-2013.html

[17] I. Hwang and Y. Kim, "Analysis of Security Standardization for the Internet of Things," in *2017 International Conference on Platform Technology and Service (PlatCon)*, 2017, pp. 1–6.

[18] ISO/IEC 30128:2014. (2014) Information technology — Sensor networks — Generic Sensor Network Application Interface . [Online]. Available: https://www.iso.org/standard/53248.html

[19] ETSI TS 103 645. (2019) Cyber Security for Consumer Internet of Things .

[20] European Network and Information Security Agency, *Inventory of risk management/ risk assessment methods*, 2013. [Online]. Available: https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods

[21] C. J. Alberts, S. G. Behrens, R. D. Pethia, and W. R. Wilson, "Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Framework, Version 1.0," 6 1999.

[22] Federal Office for Information Security . (2005) IT Grundschutz. [Online]. Available: http://www.bsi.de/gshb/

[23] B. F. Zahra and B. Abdelhamid, "Risk analysis in Internet of things using EBIOS," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–7.

[24] ISO/IEC 27001:2013. (2013) Information technology — Security techniques — Information security management systems — Requirements. [Online]. Available: https://www.iso.org/standard/54534.html

[25] S. Haller, A. Serbanati, M. Bauer, and F. Carrez, "A Domain Model for the Internet of Things," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 411–417.

[26] S. Chehida, A. Baouya, M. Bozga, and S. Bensalem, "Exploration of impactful countermeasures on iot attacks," in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, 2020.

Integrated Solution for Industrial IoT Data Security – The CHARIOT Solution

Konstantinos Loupos, Alexandros Papageorgiou, Thomas Krousarlis, Antonis Mygiakis
Inlecom Innovation,
Athens, Greece
{name.surname}@inlecomsystems.com

Sofiane Zemouri, Magdalena Kacmajor
IBM Ireland Ltd,
Ballsbridge, Ireland
sofiane.zemouri1@ibm.com,
magdalena.kacmajor@ie.ibm.com

Harris Avgoustidis, Vassileios Kalompatsos
TELCOSERV,
Agios Stefanos, Greece
{h.avg, vkal}@telcoserv.gr

Christos Skoufis, Stelios Christofi, Vasos Hadjioannou
EBOS Technologies Ltd,
Nicosia, Cyprus
{christoss, stelios, vasosh}@ebos.com.cy

Andrea Battaglia, Andrea Chiappetta, Jacopo Cavallo
ASPISEC Srl,
Rome, Italy
{a.battaglia, a.chiappetta, j.cavallo}@aspisec.com

Basile Starynkevitch, Franck Vedrine
CEA, LIST,
Gif-sur-Yvette, France
{name.surname}@cea.fr

Konstantinos Zavitsas
VLTN GCV,
Antwerpen, Belgium
kzavitsas@gmail.com

George Theofilis
CLMS Hellas,
Athens, Greece
g.theofilis@clmsuk.com

Abstract— **The CHARIOT H2020 (IoT) project (Cognitive Heterogeneous Architecture for Industrial IoT), integrates a state-of-the-art inclusive solution for the security, safety and privacy assurance of data in industrial networks. The solution is based on an integrated approach for IoT devices lifecycle management (based on blockchain and public key infrastructure technologies), IoT firmware development and deployment (source and binary level vulnerability analyses), data analytics (privacy by design, sensitive data detection, dynamic network configurations etc.) and a set of user interfaces for management and control of the network, devices and the CHARIOT platform. CHARIOT is funded by the H2020 programme under the IoT topic, has a 3-year duration and concludes its activities by the end of 2020.**

Keywords— IoT, industrial data, security, privacy, safety

I. Introduction

The CHARIOT project is focusing its activities on an integrated solution towards recent risks and challenges of the industrial IoT domain. These include a wide span of cyber technological concerns and attacks that include: i) eavesdropping, interception and hijacking (man in the middle, protocol hijacking, network reconnaissance etc.), ii) Nefarious activities, abuse (malware, denial of service, software manipulation, targeted attacks, personal data abuse and brute force attacks), iii) unintentional damages (configuration changes, third party damages, erroneous usage etc.), iv) network failures and malfunctions (failure of sensor/device, software vulnerabilities, failure/malfunction of control systems) and v) legal (contractual requirements, violation of rules). The paper contribution is summarized to IoT Devices' Lifecycle management, IoT Firmware Development and Deployment, Intelligent IoT Data Analytics and IPSE and Platform and User Interface as components of the CHARIOT solution.

II. Industrial IoT Security Orientation

A. *Industrial Requirements Overview*

The requirements related to the CHARIOT project offerings are strongly related to recent challenges in modern IoT networks and mostly target sensing and monitoring systems in various industrial themes including smart buildings, airports and trains. All investigated scenarios require data exchanges in a safe, secure and private approach resulting into overall needs of trusting the actual sensors and information they convey in a complex network, guaranteeing thus the network devices accuracy and non-intrusion. These challenges have driven the CHARIOT solutions in placing the actual network devices as the 'root of trust' in these IoT networks [1] [2] [3].

CHARIOT central revolution and innovation over the current state of the art is oriented in placing the actual devices of an IoT network as the root of trust through its cohesive approach towards Privacy, Security and Safety (PSS) of industrial IoT Systems. This is achieved through a combination of Public Key Infrastructure (PKI) technologies coupled with pre-programmed private keys deployed to IoT devices with corresponding private keys in Blockchain for affirming/approving valid transactions, a blockchain ledger affirming various levels of operational/functional changes in the network (devices authorization, provisioning, status changes etc. as an audit log), a supervision engine combining supervision, analytics and predictive modelling over IoT data and a firmware development, validation and update approach (based on online and offline code/binary analyses) securing end-to-end code development and execution on the devices.

CHARIOT provides a series of unique and innovative management features for Industrial IoT and connected devices including providing devices' software and firmware level security and sensor visibility through a dashboard for, configuration, software updates management etc. By automating key sensor management functions using blockchain, PKI and automated workflows, CHARIOT provides a solution to coping with the fast pace growth of emerging IoT technologies whose pace of evolution is faster pace than skilled staffing and available resources while at the same time places the IoT devices as the root of trust (central innovation point in CHARIOT). In other words, CHARIOT automates key sensor management functions to improve their cost effectiveness. In this direction, CHARIOT, addresses the whole lifecycle of IoT devices and networks supporting various verticals.

B. Building Management Requirements and Challenges

In building management view, CHARIOT has investigated the IBM Technology campus (partner in CHARIOT) including thousands of sensors and actuators of varying types, functionalities and levels of sophistications deployed across six main buildings. These endpoints constantly monitor and report back to different systems such as safety and workplace management systems. The endpoints range from state-of-the-art fire detection sensors down to inexpensive heat sensors placed in computer racks in internal lab rooms by operations staff. These systems perform monitoring and control functions in an isolated manner. Each system is an IoT silo that has visibility over a limited area and has actionability to perform a constrained set of functionalities only. In addition, these heterogeneous systems contain different user interfaces, which makes it difficult for administrators to get used to and use them to their full potential. This makes the enforcement of campus wide safety and security policies extremely difficult to realise. In fact, in the best of cases, these systems only allow for basic analysis of aggregated and historical data collected through some datapoints spread across multiple silos on the campus. Visualization and reporting of intrusions, out of boundary behaviour as well as end to end devices lifetime monitoring (software upgrades etc.) are of primal importance and need.

C. Airport Environment Requirements and Challenges

In airport situations, as analyzed from the Athens International Airport (partner in CHARIOT), the primal importance of the operators is focusing on evacuation cases, passengers' comfort and maintaining smooth conditions in both cases. For this, monitoring/sensing systems are spread in various places of the airport infrastructure and continuously monitor the infrastructure sensor measurement to ensure in bounds behaviour. However, tampering (software or hardware) of these devices remains practically impossible (or very difficult), airport operators remain seriously alert in keeping up with modern IoT cyber security solutions and standards to avoid this. For this, recent cyber security implementations ensuring the data safety, security and privacy are of outmost importance in view of trusting the sensor data itself.

D. Train/Rail Environment Requirements and Challenges

Cooperation with TRENITALIA (as also a partner in CHARIOT), has revealed a different dimension also related to data security and privacy that relates to data collection for safety and predictive maintenance operations as well as efficiency management. This is seen usually in train (wagon) scenarios where collected data are analysed in modern systems to perform continuous monitoring of traffic flows, prevention, early detection, diagnosis and mitigation of the data breaching effect controlling the IoT sensors data package that are delivered to Dynamic Maintenance Management Systems. In this case, train operators need a system that checks the IoT communications and collects status reports informing the operator of potential security violations detected.

III. Overall CHARIOT Technical Orientation

In view of detailed analyses of the above requirements, CHARIOT is developing an innovative Privacy, Security and Safety (PSS) platform for IoT Systems, that places devices and hardware at the root of trust, in turn contributing to high security and integrity of industrial IoT.

The solution consists of a CHARIOT platform that integrates the various components and services of the solution integrated into a cohesive and dynamic approach. The main components consisting the CHARIOT solution include three run-time engines: i) privacy engine ii) security engine and iii) safety engine, each responsible for different layer of IoT data management and security. Machine Learning (ML) technologies are running in both the safety and privacy engines to ensure that data are inside the predictive boundaries and follow normal (and acceptable) operational behaviors inside the networks.

The solution also integrates recent research results on software level guarantees, including source code analysis (development time) and binary code analysis (execution time). These are strongly interconnected (via metadata interchanges into the security engine) to provide an end-to-end IoT devices lifecycle management and security at the firmware level.

A strong component of the solution includes a blockchain layer combining Public Key Infrastructure (PKI) technologies to affirm firmware or devices modifications storing the related information in a Distributed Ledger approach. This is used for both the devices' network registration (and commissioning) and also for the firmware updates (guarantees of IoT device firmware) from source code development up to the firmware update at the device. Operational and management dashboards serve as the User Interface (UI) for the platform and system operators including IoT sensors/devices commissioning, network setup, management and control as well as zones' definition and topology considerations.

As described above, a reference architecture integrates all above modules and technologies into a modern IoT solution span inside the cloud and fog layer of services. A high-level system description is included in the diagram below:

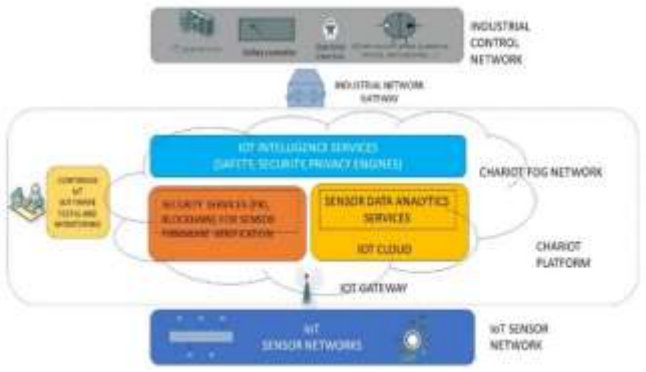

INDUSTRIAL CONTROL NETWORK
INDUSTRIAL NETWORK GATEWAY
IOT INTELLIGENCE SERVICES (SAFETY, SECURITY, PRIVACY ENGINES)
CHARIOT FOG NETWORK
SECURITY SERVICES (PKI, BLOCKCHAIN FOR SENSOR FIRMWARE VERIFICATION
SENSOR DATA ANALYTICS SERVICES
IOT CLOUD
CHARIOT PLATFORM
IOT GATEWAY
IIOT SENSOR NETWORKS
IoT SENSOR NETWORK

Fig. 1. High Level CHARIOT System Design

More details for the operation and capabilities of the developed modules are described in the following sections in this publication.

The table below summarizes the technical orientation of CHARIOT over modern IoT threats and the particular components of CHARIOT

IoT Threat	CHARIOT Solution
▪ **Man-in-the-middle attack** ▪ **IoT protocol high jacking** ▪ **Network reconnaissance**	▪ Ruggedized communication protocol and encrypted communications between devices and controllers/gateways supported by blockchain ▪ Provisioning of all sensors in an IoT network through blockchain registration/affirmation ▪ Blockchain-based PKI for sensor and gateway authentication ▪ Four-eye-principle based sensor provisioning in the IoT network ▪ Dashboard-based solutions for sensor configuration, management and alerting
▪ **Malware** ▪ **Denial of service** ▪ **Software/hardware/ info manipulation** ▪ **Targeted attacks** ▪ **Abuse of personal data** ▪ **Brute force**	▪ Firmware static analysis avoiding software vulnerabilities (etc.) at source code and existence of backdoors, software scope alteration etc. ▪ Firmware binary checking against injected code at execution level avoiding Ransomware, viruses, Trojan horses and spyware ▪ Firmware hashing and meta data storage inside the binary (and blockchain) for increased software update assertion ▪ Orchestrating mechanism for sensor data ingestion, management, storage, normalization and external API ▪ Registration of sensor status and alerts in blockchain affirming transactions and events ▪ Private data automated flagging and reporting ▪ Safety engine managing topology, sensors deployment, commissioning and provisioning ▪ Data encryption policies based on blockchain technologies to avoid privacy breaches in IoT ▪ Dashboard-based solutions for sensor configuration, management and alerting
▪ **Unintentional configuration changes** ▪ **Damages by third parties** ▪ **Erroneous usage by administration**	▪ Orchestrating mechanism for sensor data ingestion, management, storage, normalization and external connectivity API ▪ Machine learning anomaly detection based on user-defined models and neural networks ▪ IoTL (language) for dynamic network configuration, access control rules and network topology definition ▪ Dashboard-based solutions for sensor configuration, management and alerting
▪ **Failure of sensor or device** ▪ **Software vulnerabilities exploitation** ▪ **Failure/malfunction of control system**	▪ Machine learning anomaly detection based on user-defined models and neural networks ▪ Predictive analytics to highlight out-of-bounds behaviors and assess combined interdependent risks
▪ **Contractual requirements** ▪ **Violation of rules**	▪ Machine learning anomaly detection based on user-defined models and neural networks ▪ Predictive analytics to highlight out-of-bounds behaviors and assess combined interdependent risks
▪ **Sabotage / Vandalism**	▪ Out of CHARIOT scope for CHARIOT however support for malfunctioning devices is provided

IV. The CHARIOT IoT Engines

CHARIOT integrates three (3) IoT data management layers responsible for performing operations on the data to verify and affirm their privacy, security and safety inside the IoT network. The components have been designed by taking into consideration the operation and scalability requirements of the three living labs participating in CHARIOT (rail, airport, smart buildings) into the IPSE (Integrated Privacy and Safety Engine). Safety here refers to Machine learning anomaly detection based on user-defined models and neural networks. The IPSE can be scaled out by distributing the runtime across multiple nodes if needed. A CHARIOT simulation tool will also be used internally to test the platform and overall system scalability and elasticity through exhaustive testing using large series of data that may not be available in the CHARIOT LLs but still pose a significant challenge in IIoT systems and networks. These are described below:

A. *Privacy Engine*

The CHARIOT Privacy Engine employs and integrates modern security protocols and technologies (e.g. Blockchain) to provide the foundation layer for the trusted interchange of information between the different network actors (sensors, nodes, devices, gateways, controllers etc.). The Privacy engine utilizes the IoT topology described with the IoTL language to ensure that only data from well-known sensors are accepted into the system. The IoTL language itself was extended with new concepts that can fully describe access control rules and allow access to sensor data only to specific systems, users, roles, etc. These new concepts also add semantics relevant to privacy, such as explicitly flagging a sensor as a sensitive data sensor, that can later be used e.g. to obfuscate or anonymize some or all properties of the data [4]. When a system needs to receive sensor data it must register its public key with CHARIOT's Blockchain-based PKI. The Privacy engine uses the PKI to get the public keys of the system that is allowed to receive sensor data and uses it to encrypt the data before sending them. This way only the owner of the private key can decrypt and access the raw data [4].

This component considers recent privacy issues in IoT systems including data being collected by individual sensors that should enter the system if only the sensor is known and registered in the topology and also if the data is from a known sensor, data encryption must be applied using a public key stored in a blockchain PKI. This module uses advanced cryptography in achieving protection towards confidential information stored in network and secure transmission over one network to another network. Cryptography is applied on the sensor data, immediately after, sensor data are verified over their receival from a (topology) well known sensor. CHARIOT has designed the encryption PKI engine so it can support multiple encryption algorithms and has initially adopted the RSA Cryptography algorithm for the first version of the Engine. The integrated blockchain layer provides valuable security features such as certificate revocation, elimination of central points-of-failure and a reliable transaction record that are otherwise unattainable by traditional PKI systems. Additionally, blockchain is applied as a public append-only log, naturally provides the certificate transparency (CT) property proposed by Google [5].

The CHARIOT Privacy engine ensures data privacy through encrypting data at the source, specifically at the southbound dispatcher through a PKI supported by CHARIOT blockchain infrastructure. Using CHARIOT Blockchain solution for handling PKI provides secure encryption for the multiple data streams handled by CHARIOT. Alert flags are raised in every case of sensitive data transfer through the fog-node; thus, the Network Administrator is informed in order to report accordingly.

To build the Privacy Engine, open source solutions and Python scripts have been used to develop this application. For encryption an RSA algorithm was used to complete the engine. The solution was packed as a docker container and it is available at GitLab Private Registry.

B. *Security Engine*

The CHARIOT security engine is responsible for the integrity and trust of the devices (sensors, gateways, controllers etc.) of the IoT network. This protects the devices (and network) against modern IoT attacks such as: i) reverse-engineer of the entire firmware (extract the file system and understand how the entire device works, knowing the possible use of known-to-be-vulnerable out-of-date API/libraries or unknown exploitable vulnerabilities), ii) insert a firmware backdoor (making the device covertly connected to a malicious Command & Control server), iii) change the device behaviour (altering its performance), iv) find hard-coded private symmetric-cryptography keys/passwords/user-names or private certificates (used to encrypt communications between the device and other systems and eavesdrop these communications) and v) roll-back the firmware to a previous legitimate version with known vulnerabilities he/she wants to exploit (verify if the pushed firmware is authentic, so it can easily survive most of the in-place controls, as usually, they tend to check just the firmware source and/or the firmware integrity) [6].

The CHARIOT security engine verifies the reliability of new issued firmware(s) during the tricky and demanding update phase using features detection and heuristic approach. The firmware verification analyses the firmware's binary that will be flashed on the end-device (sensor or gateway). The firmware analysis is performed during the firmware update process, and its purpose is to highlight any vulnerabilities inside the firmware code that could potentially lead to cyber-attacks. A created hash (during the firmware development stage) of the firmware is stored in the blockchain after the validation of the Security Engine. The hashing of the binary file is performed by the CHARIOT platform along with the keypair and the registration of the hashing to the blockchain. When a potential security issue has been found inside the reversed binary code of the firmware, the Engine reports a security violation to the management for the subsequent actions and analysis.

The heuristic method treats the system as different sub-systems so that the sub-system's solution must spread widely at the solution space. This approach is more appropriate since we have to deal with types of firmwares that are often very different from each other (in architectures/CPUs/ characteristics). Heuristic method brings several benefits, giving us flexibility in analysis, in fact we can combine different features as well as news instructions and features could be added as new functions with new parameters for analysis. This allows an analysis addressed by considering different aspects of the characteristics of the firmware, the change of its behavior and possible vulnerabilities that could be exploited to tamper the firmware, leading to a more complete and reliable analysis.

The utility is designed to collect data by binaries, perform statistical analysis, compare two firmware images and checking for vulnerabilities and formal contracts. The analysis is performed on the assembler instructions level. Based on the analysis results, a report is generated which contains information on the differences between the two images and if a vulnerability has been detected. An advanced attack pattern recognition helps to detect unusual hardware behavior and compares anomalies with an internal set of instruction that can lead to recognize an unknow attacks and exploitations [6].

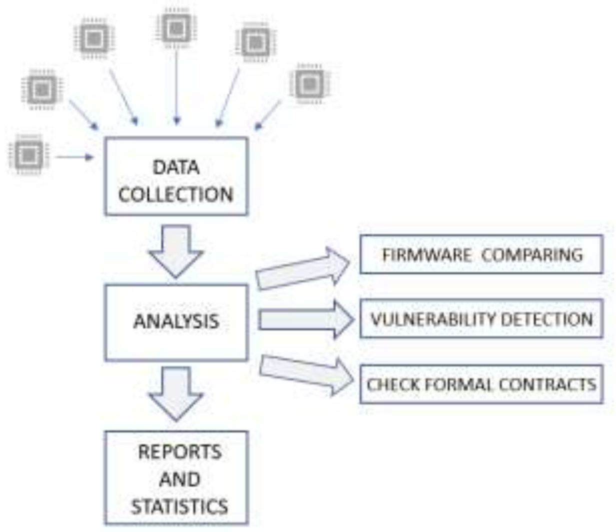


Fig. 2. CHARIOT Security Engine Model Implementation [6]

The CHARIOT security engine vulnerability detection layer provides the following vulnerability classes check: i) buffer overflow, ii) format string and iii) artbitrary memory access and reports its findings during the firmware update process to the platform and in-turn to the User Interface, accepting or declining/stopping the firmware update process.

C. *Safety Engine*

The CHARIOT Safety Engine analyses the IoT topology and signal metadata relative to the relevant safety profiles and applies closed-loop machine-learning techniques to detect safety violations and alert conditions. This comprises a later capability on the cognitive engine that will leverage the Cyber-Physical topological representation of the system-of-systems combined with the security and safety polices.

Anomaly detection aids finding patterns in data that do not conform to expected behavior [7]. Under IoT terms, anomalies are considered as any abnormal data stream pattern whose root cause may have safety security implications. These may be a faulty sensor, a safety hazard or a security issue. By identifying these issues and providing a central alerting mechanism, CHARIOT will help operators in reducing response time and identify root causes in cases of issues.

The CHARIOT security engine uses rule-based policies with simple arithmetic comparisons to enforce policies on data streams. An innovative IoTL (IoT scripting language -IOT Language) supports alerting the industrial gateway if a safety policy violation is observed within the IoT State. Furthermore, the security engine is using machine learning based anomaly detection.

In addition to a low-level Swagger API, IBM has developed a high-level UI for interfacing with the IoTL to facilitate interactions with the service topology as well as static and dynamic policies enforcement. The IoT Manager UI is implemented using the React and Leaflet libraries and features a Quake-style terminal for inputting IoTL commands [8].

V. PREDICTIVE MACHINE LEARNING MODELLING

IoT data are in general characterized by volume, velocity and variety-lack of structure/heterogeneity. The frequent lack of structure in IoT data makes it difficult to analyze such data with traditional analytics and business intelligence tools. Additionally, IoT data that capture physical processes such as temperature, motion, or sound can be noisy. Finally, the quality of IoT data can vary, i.e. datasets can have significant gaps, and contain corrupted readings. Lastly, meta-data/context may be essential to understand IoT data, as such data are often meaningful in some context. IoT data typically contain patterns that include seasonal fluctuations and trends. Such patterns must be detected amongst noise, random fluctuations and other non-important findings. IoT analytics systems can filter, transform, and enrich the IoT data before storing it, usually in a time-series data store for analysis. Insights from the IoT analytics are then used to better understand the system measured by the IoT sensors and to make better decisions.

Anomaly detection refers to the problem of finding patterns in IoT data that do not conform to some norm [9]. These non-conforming patterns are often referred to as anomalies, (and also as outliers, exceptions, aberrations, etc.) in different contexts. Anomaly detection has wide applicability in a variety of IoT applications such as for security protection and fault detection in industrial systems. One major application of anomaly detection, of relevant to CHARIOT is fault detection in mechanical units. The anomaly detection techniques in this domain use IoT to monitor the performance of industrial components such as motors, turbines, and other mechanical components to detect when maintenance of the system will be required (‘predictive maintenance’).

CHARIOT is using several different methodologies for the anomaly detection layer including: i) One Class Support Vector Machine (OSVM) - trained using both positive and negative examples, however studies have shown there are many valid reasons for using only positive examples, ii) Elliptical Envelope (EE) - based on the Minimum Covariance Determinant (MCD) estimator the first affine equivariant and highly robust estimators of multivariate location and scatter and iii) Isolation Forest (IF) - efficient unsupported machine learning algorithm for anomaly detection focusing on identifying the few different points of the dataset, rather than the normal data, and uses the isolation mechanism that detects anomalies purely based on the concept of isolation without employing and distance or density measure, which is fundamentally different from previously described methods [11].

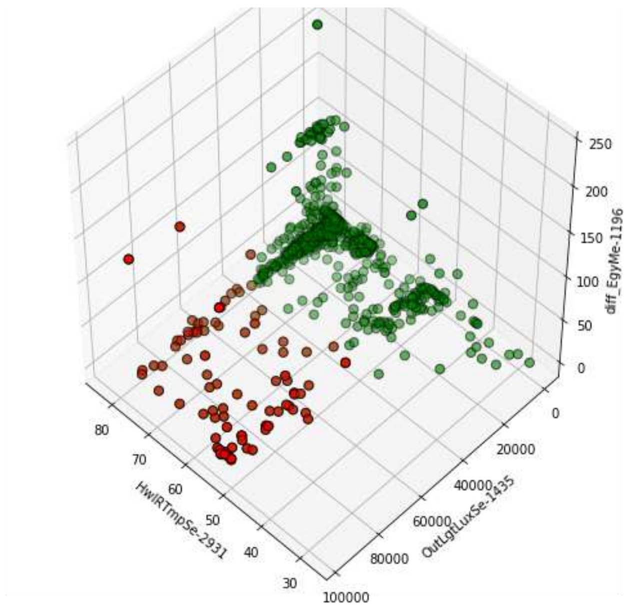


Fig. 3. Example of Anomaly Detection Modelling

VI. Software Life-Cycle Management

The CHARIOT software analysis and lifecycle management includes a software source code verification analysis level (Bismon) that is strongly linked to the CHARIOT security engine (and the firmware update process). This, includes the source code analysis, creation of metadata and hashing of source code inside the binary file that are analysed during the firmware update process (via the security engine and together with the binary level warnings) to either accept or decline the software update process.

CHARIOT focuses mainly on a system of systems (e.g. networks of systems and systems of networks) approach, so [10] "aims to address how safety-critical-systems should be securely and appropriately managed and integrated with a fog network made up of heterogeneous IoT devices and gateways.". Within CHARIOT, static analysis methods support its Open IoT Cloud Platform through its IoT Privacy, Security and Safety Supervision Engine. Some industrial CHARIOT partners, while being IoT network and hardware experts, acknowledge that their favourite IDE (provided by their main IoT hardware vendor) is running some GCC under the hoods during the build of their firmware. Nevertheless, these partners do not use static source code analysis tools.

The CHARIOT approach to static source analysis leverages on an existing recent GCC cross-compiler [11] so focuses on GCC-compiled languages [12]. Hence, the IoT software developer following the CHARIOT methodology would just add some additional flags to existing gcc or g++ cross-compilation commands, and needs simply to change slightly his/her build automation scripts (e.g. add a few lines to his Makefile). Such a gentle approach (see figure 1) has the advantage of not disturbing much the usual developer workflow and habit, and addresses also the junior IoT software developer. The compilation and linking processes are communicating -via some additional GCC plugins (cf. GCC Community [6] §24) doing inter-process communication- with our persistent monitor, tentatively called bismon. It is preferable (see Free Software Foundation) to use free software GCC plugins (or free software generators for them) when compiling proprietary firmware with the help of these plugins; otherwise, there might be some licensing issues on the obtained proprietary binary firmware blob, if it was compiled with the help of some hypothetical proprietary GCC plugin.

CHARIOT static analysis tools will leverage on the mainstream GCC compiler (generally used as a cross-compiler for IoT firmware development). Current versions of GCC are capable of quite surprising optimizations (internally based upon some sophisticated static analysis techniques and advanced heuristics). But to provide such clever optimizations, the GCC compiler has to be quite a large software, of more than 5.28 million lines of source code (in gcc-8.2.0, measured by sloccount). This figure is an under-estimation, since GCC contains a dozen of domain specific languages and their transpilers to generated C++ code, which are not well recognized or measured by sloccount.

Since a single Bismon process is used by a small team of IoT developers, it provides some web interface: each IoT developer will interact with the persistent monitor through his/her web browser. In addition, a static analysis expert (which could perhaps be the very senior IoT developer of the team) will configure the static analysis (also through a web interface) [13].

VII. Supporting Blockchain and PKI Technologies

The blockchain component of CHARIOT (based on a hyperledger Fabric implementation) is used at different engines and layers to affirm data, devices and network information. In this, the information stored in blockchain include sensor IDs, network states and firmware validation hashings. These are used by the privacy, security and safety engine as described before.

Blockchain-based PKI approach makes MITM attacks virtually impossible as when group of authorities publishes or revokes the public key of an identity on the blockchain, the information will be distributed across all nodes, so tampering the public-key will be (theoretically) out of the question. Traditional PKI resolves MITM risks by embedding Root CA certificates into browser installations, thus artificially expanding CA entrance barriers and increasing the time necessary for Root CA certificate revocation.

There are several advantages of using this PKI-based blockchain implementation including: i) The validation of a certificate is simple and fast with no form of CA certificate chain, ii) Blockchain-based PKI solves a longstanding problem of traditional PKIs by not requiring the use of a service that issues certificate revocation lists (CRLs) thanks to blockchain synchronization between network's nodes where any modification to the state of a certificate will be instantaneously notified to the all nodes and iii) Blockchain-based PKI provides flexible protection against the man-in-the-middle (MITM) attacks. Traditionally, MITM is considered as a major security risk implying attacker to hijack a browser's connection for a given website by presenting a valid certificate (i.e., forged public key) for that domain. For users and web browsers it is difficult to identify the replacement of certificate when the related CA has been hacked by the attacker [7] [8].

VIII. Operational and Device Management Dashboards

User interfacing is considered as an important layer where two distinct interfaces (dashboards) are being developed (Device Management Dashboard: handling blockchain devices registration, firmware updates, engine management and IoTL interfacing and Operational Dashboard: providing Engines' health and performance monitoring as well as alerts' and sensor data visualization).

The device management dashboard is utilizing the latest state-of-the-art web technologies to deliver rich content information to the LL users and achieve cross-browser and multi-device compatibility. Further to that, the dashboard is designed as a user friendly and fully responsive web solution, based on the CHARIOT industrial needs, providing an easy access to the necessary information. Blockchain security and accessing controls are applied to secure the access to specific information and data by different users. Moreover, Dashboards focus not only to standard monitoring actions and providing a visibility on an industrial IoT topology, sensor values and alerts but also to secured (utilizing blockchain technology) managerial activities. Those activities such as authenticating and registering (or unregistering) a sensor in the IoT topology and updating the firmware (of a sensor or a gateway) can be performed by the security engineers and management. It is important to mentioned that during the "firmware update" there is a chain of actions and integration with a number of CHARIOT components.

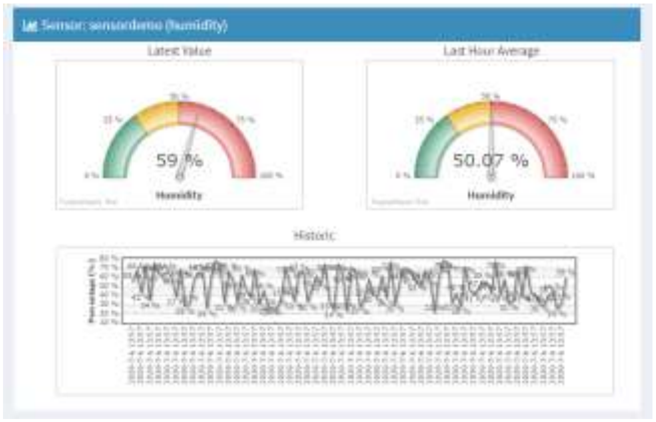


Fig. 4. Example of Data Management Dashboard

The CHARIOT Operational Dashboard is providing Engines' health and performance monitoring as well as alerts' and sensor data visualization. CHARIOT has identified the need of a more sophisticated method for platform performance monitoring as designed following the micro-services software architecture paradigm. After research on the industry-standard of micro-service platform monitoring techniques, CHARIOT has decided to adopt CNCF best practices and deploy Jaeger. With Jaeger, we can trace every action trail at the CHARIOT platform. The analysis of the collected traces helps the developer to identify bottlenecks to improve system performance and find the cause of platform malfunction. In addition to this, we implement service to monitor health of every micro-services by sending a "magic-package" to it and then wait for its response, in the end the system administrator has a dashboard to view all the collected information [14].

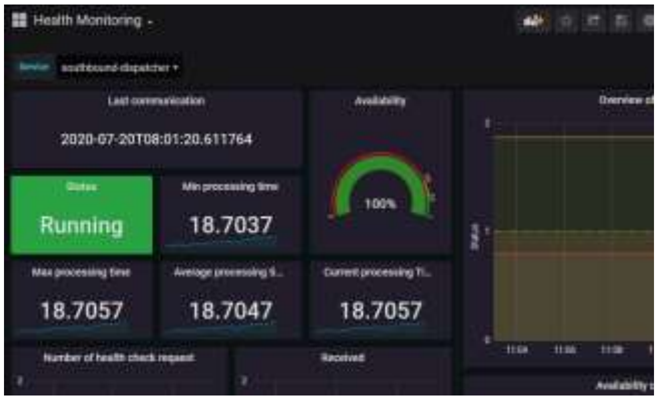


Fig. 5. Example of Operational Dashboard

IX. CHARIOT Industrial Validation

CHARIOT is by design driven by industrial IoT requirements following actual needs and paradigms of three sectors: rail, airports and smart buildings. These three industrial cases' analysis has derived exhaustive sets of requirements, industrial scenarios and validation KPIs on which, CHARIOT, has based its technical implementations.

CHARIOT will be validated in the above three (3) industrial cases based on representative security related scenarios highlighting the value and integrated approach of CHARIOT in solving modern IoT security issues and challenges.

CHARIOT is currently through its deployment and validation phase, having deployed its whole platform in the three infrastructures and having performed its first round of technical recommendations from the end-users. In the next five months, and up to the end of 2020, CHARIOT is expected to finish its activities with the final feedback of recommendations and adaptations to the three industrial setups.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program (No 780075). The authors acknowledge the research outcomes of this publication belonging to the CHARIOT consortium.

References

[1] K. Loupos - INTEGRATED SOLUTION FOR PRIVACY AND SECURITY OF IOT DEVICES IN CRITICAL INFRASTRUCTURES, Critical Infrastructure Protection and Resilience Europe (CIPRE 2020), 6-8 October 2020, Bucharest, Romania.

[2] K. Loupos, A. Papageorgiou, A. Mygiakis, B. Caglayan, B. Karakostas, T. Krousarlis, F. Vedrine, C. Skoufis, S. Christofi, G. Theofilis, H. Avgoustidis, G. Boulougouris, A. Battaglia, M. Villiani - COGNITIVE PLATFORM FOR INDUSTRIAL IOT SYSTEM SECURITY, SAFETY AND PRIVACY, Embedded World 2020 Conference and Exhibition, 25 - 27 February 2020, Nuremberg, Germany.

[3] Adel S. Elmaghraby, Michael M. Losavio, "Cyber security challenges in Smart Cities: Safety, security and privacy", Journal of Advanced Research Volume 5, Issue 4, pp 491–497, 07/ 2014.

[4] CHARIOT – D3.2 – IoT Privacy Engine based on PKI and Blockchain technologies, CHARIOT 2019.

[5] L. Axon and M. Goldsmith, “PB-PKI: A privacy-aware blockchain based PKI,” in Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017., 2017, pp. 311–318.

[6] CHARIOT - D3.8 – IoT Security Engine based on vulnerability checks, CHARIOT 2020.

[7] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR) 41.3 (2009): 15.

[8] CHARIOT – D3.9 - IoT Safety Supervision Engine (ISSE) (final prototype) v1.0_FINAL, CHARIOT, 2020.

[9] Chandola, Varun, Arindam Banerjee, Vipin Kumar. Anomaly detection: a survey. ACM Computing Surveys, September 2009.

[10] Taken in October 2018 from https://www.chariotproject.eu/About, §Technical Approach.

[11] The actual version and the concrete configuation of GCC are important; we want to stick -when reasonably possible- to the latest GCC releases, e.g. to GCC 8 in autumn 2018. In the usual case, that GCC is a cross-compiler. In the rare case where the IoT system runs on an x86-64 device under Linux, that GCC is not a cross-, but a straight compiler.

[12] The 2019 Gnu Compiler Collection is able to compile code written in C, C++, Objective-C, Fortran, Ada, Go, and/or D.

[13] CHARIOT – D1.5 - Specialized Static Analysis tools for more secure and safer IoT software development (ver.2).

[14] CHARIOT – D6.9 – CHARIOT Rescoping Guideline

Guidelines for Privacy and Security in IoT

Pasquale Annicchino
Archimede Solutions
Geneva, Switzerland
pannicchino@archimede.ch

Simone Seminara, Francesco Capparelli
Istituto Italiano per la Privacy e la Valorizzazione dei Dati
Rome, Italy
{s.seminara, f.capparelli}@ istitutoprivacy.eu

***Abstract*— Norms and standards define the ecosystem in which IoT solutions are developed and deployed. It is often difficult for people without a legal training or an understanding of standardization dynamics to fully grasp the state of the art in this very relevant field. This contribution aims at highlighting the most relevant tools available and explaining their relevance.**

***Keywords*— Data protection; privacy; security; Internet of Things; guide-lines.**

I. Introduction: mapping the landscape

A. Relevance of the exercise

The mapping of international security and data protection by design guidelines is of paramount relevance in the identification of best practices in the context of IoT. With regard to the implementation and demonstration of appropriate technical and organizational measures as referred in Articles 24(1)-(3), 25, and 32(1)-(3) of the General Data Protection Regulation (GDPR) [1], the literature on data protection is extensive, ranging from regulations to privacy-enhancing technologies and rules that are general. Without any objective of completeness, which would be outside the scope of this contribution, we briefly introduce below some of the best-known approaches to data management and data protection from a technical perspective (among those freely available online) which might be useful also for researchers with no previous legal training.

First of all, the European Union Agency for Cybersecurity (ENISA) in the eminent document *Privacy and Data Protection by Design* [2] declares eight general strategies for implementing the principle of "privacy by design" as defined in the GDPR: minimise, hide, separate, aggregate, inform, control, enforce and demonstrate.

Another important approach to formulate general principles for the protection of personal data and cybersecurity is the one developed by the Information & Privacy Commissioner of the State of Ontario, Canada. This work [3] proposes seven general principles: Proactive not Reactive (Preventative not Remedial); Privacy as the Default Setting; Privacy Embedded into Design; Full Functionality (Positive-Sum, not Zero-Sum); End-to-End Security (Full Lifecycle Protection); Visibility and Transparency (Keep it Open); Respect for User Privacy (Keep it User-Centric).

It should also be noted that there are several non-legal frameworks resulting from the application of international cybersecurity standards and, therefore, the present document is useful to provide a mapping of the actual international standards, guidelines and best practices regarding IoT.

B. Regulations

Almost all the articles of the GDPR provide the European interpretation of the concept of personal data protection, specifying several rights for citizens with regard to the processing of their personal data. Rights such as access and limitation are well detailed in the Regulation, which therefore gives control over the data primarily to the individual to whom the data are related. To complement this, there are three articles referring to cybersecurity, without which data protection would inevitably be compromised. Article 32 outlines the security measures, while Articles 33 and 34 the notification obligations in case of data breach.

In relation to the focus on the IoT systems in this document, however, it should be noted that the GDPR is not entirely explicit on how an IoT device should protect data. The manufacturers are therefore obliged to supply products that comply with the Regulation and to ensure that the companies that will (acquire and then) use them can operate in accordance with the Regulation. Finally, Article 25 outlines provisions on data protection by design and by default, i.e. already by design and by default, taking over the concepts outlined in Articles 5 (on "data minimization") and 32 (on security measures, mentioning in particular "pseudonymization"). However, it is completely implicit what characteristics an application must have in order to be considered GDPR-compliant.

The processing of personal data within the IoT framework often sees the interaction between the system and its operator, the latter being authorised to the specific processing possible through the use of the given IoT device. In particular, the authorisation to the processing – as mandated by the GDPR – details the areas of the processing itself, i.e. what and how the authorised person is allowed to process personal data. There is then a so-called *ceremony* between device and operator, i.e. a protocol distributed and enacted between machines and human beings. Sometimes such a protocol may involve several persons or even none: the GDPR defines the latter case as *automated processing*.

The articles of the GDPR can be interpreted as a set of requirements, aimed at achieving the general objective of personal data protection, for the participants in the ceremony/protocol mentioned above.

This work has been partially supported by NGIoT [52], a Coordination and Support Action funded by European Union's Horizon 2020 research and innovation programme (H2020-EU.2.1.1.) under grant agreement ID 825082.

II. Relevant guidelines

In this paragraph we detail general guidelines, reviews and mappings which can be applied to the world of the Internet of Things as a whole. Each subsection details one of the organisations involved in such publications.

A. OWASP

The Open Web Application Security Project (OWASP) [4] is a nonprofit foundation that works to improve the security of software through its community-led open source software projects. One of its flagship projects is the *OWASP Top 10* [5], a standard awareness document for developers and web application security; it represents a broad consensus about the most critical security risks to web applications.
Here are two of its publications about IoT.

1) OWASP IoT Top 10, 2018 (previous version in 2014)

Along the lines of the widely known Top 10 for web apps, the *OWASP IoT Top 10* [6] focuses on things to avoid when building, deploying or managing IoT systems. The list is:

1) *Weak, Guessable, or Hardcoded Passwords.* Use of easily brute forced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorised access to deployed systems.
2) *Insecure Network Services.* Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorised remote control.
3) *Insecure Ecosystem Interfaces.* Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.
4) *Lack of Secure Update Mechanism.* Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.
5) *Use of Insecure or Outdated Components.* Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.
6) *Insufficient Privacy Protection.* User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.
7) *Insecure Data Transfer and Storage.* Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.
8) *Lack of Device Management.* Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.
9) *Insecure Default Settings.* Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.
10) *Lack of Physical Hardening.* Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

2) OWASP IoT Top 10 2018 Mapping Project

This project [7] provides mappings of the OWASP IoT Top 10 2018 [6] to industry publications and sister projects, such as:

- OWASP IoT Top 10, previous version (2014) [8];
- GSMA IoT Security Assessment Checklist [9] (see also § 2.3.2);
- Department for Digital, Culture, Media & Sport (UK Government), *Code of Practice for Consumer IoT Security* [10] (see § 2.2.1);
- ENISA Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures, 20 November 2017 [11] (see § 2.4.1);
- CTIA Cyber-security Certification Test Plan for IoT Devices [12][13] (see § 2.5);
- CSA IoT Security Controls Framework [14][15] (see § 2.5);
- ETSI Technical Specification (TS) 103 645 V1.1.1 (2019-02), *CYBER; Cyber Security for Consumer Internet of Things* [16].

Since this Mapping Project, ETSI published an updated version of the above standard [17] and its European counterpart [18].
From this starting point, in the following subsections we will explore these publications.

B. UK Government, Department for Digital, Culture, Media & Sport

The Department for Digital, Culture, Media & Sport (DCMS) [19] helps to drive growth, enrich lives and promote Britain abroad. Among other activities, the DCMS commissioned the *PETRAS IoT Research Hub* [20], a consortium of universities and research institutions that work together to explore critical issues in privacy, ethics, trust, reliability, acceptability and security of the IoT to conduct two literature reviews: on industry recommendations for government to improve IoT security; on the current international developments around IoT security. The two aims to these reviews, jointly published in [21], were to identify the key themes emerging from the literature and to identify international consensus around core Security by Design principles for the IoT.

1) Code of Practice for Consumer IoT Security, 14 October 2018

The DCMS, in conjunction with the UK National Cyber Security Centre (NCSC) [22] and following engagement with industry, consumer associations and academia, has developed this Code of Practice [10] (see § 2.1.2) to support all parties involved in the development, manufacturing and retail of consumer IoT with a set of guidelines to ensure that products are secure by design and to make it easier for people to stay secure in a digital world. The Code of Practice brings together, in thirteen outcome-focused guidelines, what is widely considered good practice in IoT security. The Code was first published in draft in March 2018 as part of the *Secure by Design* collection of reports [23].

An indication is given for each guideline as to which stakeholder is primarily responsible for implementation. Stakeholders are defined as Device Manufacturers, IoT Service Providers, Mobile Application Developers and Retailers. The thirteen guidelines are:

1) *No default passwords.* All IoT device passwords shall be unique and not resettable to any universal factory default value.
2) *Implement a vulnerability disclosure policy.* All companies that provide internet-connected devices and services shall provide a public point of contact as part of a vulnerability disclosure policy in order that security researchers and others are able to report issues. Disclosed vulnerabilities should be acted on in a timely manner.
3) *Keep software updated.* Software components in internet-connected devices should be securely updateable. Updates shall be timely and should not impact on the functioning of the device. An end-of-life policy shall be published for end-point devices which explicitly states the minimum length of time for which a device will receive software updates and the reasons for the length of the support period. The need for each update should be made clear to consumers and an update should be easy to implement. For constrained devices that cannot physically be updated, the product should be isolatable and replaceable.
4) *Securely store credentials and security-sensitive data.* Any credentials shall be stored securely within services and on devices. Hard-coded credentials in device software are not acceptable.
5) *Communicate securely.* Security-sensitive data, including any remote management and control, should be encrypted in transit, appropriate to the properties of the technology and usage. All keys should be managed securely.
6) *Minimise exposed attack surfaces.* All devices and services should operate on the 'principle of least privilege'; unused ports should be closed, hardware should not unnecessarily expose access, services should not be available if they are not used and code should be minimised to the functionality necessary for the service to operate. Software should run with appropriate privileges, taking account of both security and functionality.
7) *Ensure software integrity.* Software on IoT devices should be verified using secure boot mechanisms. If an unauthorised change is detected, the device should alert the consumer/administrator to an issue and should not connect to wider networks than those necessary to perform the alerting function.
8) *Ensure that personal data is protected.* Where devices and/or services process personal data, they shall do so in accordance with applicable data protection law, such as the GDPR. Device manufacturers and IoT service providers shall provide consumers with clear and transparent information about how their data is being used, by whom, and for what purposes, for each device and service. This also applies to any third parties that may be involved (including advertisers). Where personal data is processed on the basis of consumers' consent, this shall be validly and lawfully obtained, with those consumers being given the opportunity to withdraw it at any time.
9) *Make systems resilient to outages.* Resilience should be built in to IoT devices and services where required by their usage or by other relying systems, taking into account the possibility of outages of data networks and power. As far as reasonably possible, IoT services should remain operating and locally functional in the case of a loss of network and should recover cleanly in the case of restoration of a loss of power. Devices should be able to return to a network in a sensible state and in an orderly fashion, rather than in a massive scale reconnect.
10) *Monitor system telemetry data.* If telemetry data is collected from IoT devices and services, such as usage and measurement data, it should be monitored for security anomalies.
11) *Make it easy for consumers to delete personal data.* Devices and services should be configured such that personal data can easily be removed from them when there is a transfer of ownership, when the consumer wishes to delete it and/or when the consumer wishes to dispose of the device. Consumers should be given clear instructions on how to delete their personal data.
12) *Make installation and maintenance of devices easy.* Installation and maintenance of IoT devices should employ minimal steps and should follow security best practice on usability. Consumers should also be provided with guidance on how to securely set up their device.
13) *Validate input data.* Data input via user interfaces and transferred via application programming interfaces (APIs) or between networks in services and devices shall be validated.

2) Mapping of IoT Security Recommendations, Guidance and Standards to the UK's Code of Practice for Consumer IoT Security, 14 October 2018

This document [24], and the open data files and graphs provided in its companion website [25], maps the Code of Practice for Consumer IoT Security against published standards, recommendations and guidance on IoT security and privacy from around the world. Around 100 documents were reviewed from nearly 50 organizations. Whilst not exhaustive, it represents one of the largest collections of guidance available to date in this area.

The purpose of the mapping is to serve as a reference and tool for users of the Code of Practice. Manufacturers and other organisations are already implementing a range of standards, recommendations and guidance and will seek to understand the relationship between the Code of Practice and existing material from industry and other interested parties.

C. GSMA

The GSM Association (GSMA) [26] represents the interests of mobile operators worldwide, uniting more than 750 operators with almost 400 companies in the broader mobile ecosystem, including handset and device makers, software companies, equipment providers and internet companies, as well as organisations in adjacent industry sectors.

1) GSMA IoT Security Guidelines, version 2.2, 29 February 2020

The goal of the Internet of Things Security Guidelines document set [27][28][29][30] is to provide the implementer of an IoT technology or service with a set of design guidelines for building a secure product. The set of guideline documents promotes a methodology for developing secure IoT Services to ensure security best practices are implemented throughout the life cycle of the service. The documents provide recommendations on how to mitigate common security threats and weaknesses within IoT Services.

2) GSMA IoT Security Assessment

The GSMA IoT Security Assessment [31][32] (see § 2.1.2) provides a flexible framework that addresses the diversity of the IoT market, enabling companies to build secure IoT devices and solutions as laid out in the *GSMA IoT Security Guidelines* (see § 2.3.1), a comprehensive set of best practices promoting the secure end-to-end design, development and deployment of IoT solutions.

D. ENISA

The European Union Agency for Cybersecurity [33] has been working to make Europe cyber secure since 2004. The Agency works closely together with Members States and other stakeholders to deliver advice and solutions as well as improving their cybersecurity capabilities. It also supports the development of a cooperative response to large-scale cross-border cybersecurity incidents or crises and since 2019, it has been drawing up cybersecurity certification schemes.

1) ENISA Good practices for IoT and Smart Infrastructures Tool

This website [34] intends to provide an aggregated view of the ENISA Good Practices for IoT and Smart Infrastructure [35] that have been published the last years. This link comprises the above-mentioned *Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures* [11] (see § 2.1.2) and then other publications about cars, hospitals, airports, public transport and Industry 4.0.

E. Other sources and references

In this subsection we mention other miscellaneous sources about privacy and security in IoT.

CTIA [36] represents the U.S. wireless communications industry and companies throughout the mobile ecosystem and has organised a certification programme for the cybersecurity of IoT devices [12, 13] (see § 2.1.2).

The Cloud Security Alliance (CSA) [37] is an organization dedicated to defining and raising awareness of best practices to help ensure a secure cloud computing environment, including the Internet of Things with specific security controls [14, 15] (see § 2.1.2).

The Internet of Things Security Foundation (IoTSF) [38] is a collaborative, nonprofit, international response to the complex challenges posed by cybersecurity in the expansive hyper-connected IoT world. Among its publications, listed in [39], we can cite [40][41].

The World Wide Web Consortium (W3C) [42] is an international community that develops open standards to ensure the long-term growth of the Web. It is led by Tim Berners-Lee, the inventor of the Web. Its *Web of Things* (WoT) section [43] seeks to counter the fragmentation of the IoT through standard complementing building blocks (e.g. metadata and APIs) that enable easy integration across IoT platforms and application domains; to date, two W3C Recommendations have been published about WoT [44][45].

Of course, international standards developing organisations (SDOs) – whose members are governmental bodies, agencies or committees, one per member economy – have published IoT-related standards. We can cite the ITU-T Y.4000 series from the International Telecommunication Union (ITU) [46][47] and a few of those jointly published by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) [48][49][50].

Lastly, we show a glimpse of other relevant international standards, under development or just finished. This list is taken from the outcome of the 2020-04-10 webinar *Integrating privacy in the IoT ecosystem* [51], organised by the Horizon 2020 project *Next Generation Internet of Things* (NGIoT) [52], with the participation of Antonio Kung:

- ISO/IEC TR 20547-1, *Information technology — Big data reference architecture — Part 1: Framework and application process*, first edition published August 2020
- ISO/IEC TR 20547-2:2018, *Information technology — Big data reference architecture — Part 2: Use cases and derived requirements*, first edition published January 2018
- ISO/IEC 20547-3:2020, *Information technology — Big data reference architecture — Part 3: Reference architecture*, first edition published March 2020
- ISO/IEC 20547-4, *Information technology — Big data reference architecture — Part 4: Security and privacy*, first edition published September 2020
- ISO/IEC TR 20547-5:2018, *Information technology — Big data reference architecture — Part 5: Standards roadmap*, first edition published February 2018
- ISO/IEC CD 23751, *Information technology — Cloud computing and distributed platforms — Data sharing agreement (DSA) framework*
- ISO/IEC CD 27400.2, C*ybersecurity – IoT security and privacy – Guidelines* (formerly known as ISO/IEC CD 27030, *Information technology — Security techniques — Guidelines for security and privacy in Internet of Things (IoT)*)
- ISO/IEC CD TS 27101, *Information technology — Security techniques — Cybersecurity — Framework development guidelines*

- ISO/IEC CD 27556, *Information technology — User-centric framework for the handling of personally identifiable information (PII) based on privacy preferences*
- ISO/IEC WD 27557, *Organizational privacy risk management*
- ISO/IEC WD TS 27560, *Privacy technologies — Consent record information structure*
- ISO/IEC AWI 30149, *Internet of things (IoT) — Trustworthiness framework*
- ISO/AWI 31700, *Consumer protection — Privacy by design for consumer goods and services*

III. Conclusion

Guidelines are important tools for different stakeholders involved in the deployment of IoT solutions. They offer key basic points and requirements to enhance the trust of end-users and facilitate deployment. The documents highlighted in this contribution show that an important amount of work has been already done by several organisations and deserves to be taken into account.

Following the recommendations provided by the mapped international standards, therefore, allows to respect the principles of the GDPR: for example, the international standards referred to the development phase are useful to respect the "privacy by design" principle set in Article 25 GDPR; the standards on the security of personal data processing are functional to the respect of Article 32 GDPR.

As a side note, the application of the principles of the GDPR is not sufficient in cases where such processing of personal data should concern Law Enforcement Agencies (LEAs). According to the provisions of Article 29 of Directive (EU) 2016/680 [53], in fact, the data controller may use an accountability mechanism in the evaluation and adoption of technical-organisational measures. In any case, the aforementioned measures must be suitable to guarantee an adequate level of security in order to avoid the risk of personal data violation.

In general, it is useful to use all international standards as guidelines and to deduce the best practices necessary to achieve a level of security that can generate trust in end-users and simultaneously achieve compliance with the main regulations. All the mapping efforts across different security controls and publications show that the amount of redundancies is very high: we can then state that a consensus, a "common sense" has emerged in the field of IoT cybersecurity and privacy. Moreover, from a broader perspective, we can say that IoT security measures overlap consistently with cybersecurity frameworks and standards already in place for "traditional computing": consider, for instance, ISO/IEC 27001:2013 [54] and the *Common Criteria for Information Technology Security Evaluation* [55][56][57][58][59][60].

It is paramount at legislation level to properly address the need to go beyond what the GDPR and the NISD (Network and Information Security Directive) [61] today represent. With the progress of technology, is obvious that lawmakers have the duty to follow rapidly the new challenges that arise from the evolution in the societal and economic global landscape. In this sense, the integration of IoT in homes, cities and industries gives the legislators the opportunity (or necessity?) to build a new legal framework to comply with ethical requirements, to better protect freedoms and rights of citizens, at an increasingly supranational and intergovernmental level. A "GDPR of Things" is therefore urgent, with an expanded scope from previous laws, in order to establish stricter rules and norms for information security and personal data protection in World that moves fast towards "ubiquitous computing" (IoT, 5G, wearables, etc.).

In parallel with new legislative frameworks, it would be preferable a consolidation of standards and best practices carried forward by SDOs and the private sector in an open and interoperable way, before the proliferation of "walled gardens" that may compromise freedoms and rights of citizens worldwide.

References

[1] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of per-sonal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance); current consoli-dated version (2016-05-04) available at https://eur-lex.europa.eu/eli/reg/2016/679/2016-05-04

[2] European Union Agency for Cybersecurity (ENISA), Privacy and Data Protection by Design – from policy to en-gineering, 12 January 2015; PDF available at https://www.enisa.europa.eu/publications/privacy-and-data-protection-by-design/

[3] Information & Privacy Commissioner (Ontario, Cana-da), Ann Cavoukian, The 7 Foundational Principles. Im-plementation and Mapping of Fair Information Practices; PDF available at https://www.ipc.on.ca/wp-content/uploads/Resources/pbd-implement-7found-principles.pdf

[4] OWASP Foundation, Inc.; https://owasp.org/

[5] OWASP Top 10; https://owasp.org/www-project-top-ten/

[6] OWASP IoT Top 10, 2018; https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10

[7] OWASP IoT Top 10 2018 Mapping Project; https://scriptingxss.gitbook.io/owasp-iot-top-10-mapping-project/ and https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=OWASP_IoT_Top_10_2018_Mapping_Project

[8] OWASP IoT Top 10, 2018; https://wiki.owasp.org/index.php/Top_10_IoT_Vulnerabilities_(2014)

[9] https://www.gsma.com/security/resources/clp-17-gsma-iot-security-assessment-checklist-v3-0/

[10] https://www.gov.uk/government/publications/code-of-practice-for-consumer-iot-security/

[11] https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot/

[12] CTIA, IoT Cybersecurity Certification Program Manage-ment Document, version 1.1, May 2019; PDF available at https://api.ctia.org/wp-con-tent/uploads/2019/05/ctia_IoT_cybersecurity_pmd_ver-1_1.pdf

[13] CTIA, IoT Cybersecurity Certification FAQ, version 1.0, 28 March 2019; PDF available at https://api.ctia.org/wp-content/uploads/2019/03/CTIA-Certification-FAQ-Ver-1.0-28-March-2019.pdf

[14] CSA, IoT Security Controls Framework, 5 March 2019; https://cloudsecurityalliance.org/artifacts/iot-security-controls-framework/

[15] CSA, Guide to the IoT Security Controls Framework, 5 March 2019; https://cloudsecurityalliance.org/artifacts/guide-to-the-iot-security-controls-framework/

[16] https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf

[17] https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/02.01.02_60/ts_103645v020102p.pdf

[18] ETSI European Standard (EN) 303 645 V2.1.1 (2020-06); https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf

[19] https://www.gov.uk/government/organisations/department-for-digital-culture-media-sport/

[20] https://petras-iot.org/

[21] PETRAS IoT Hub, Summary literature review of industry recommendations and international developments on IoT se-curity, 7 March 2018; https://www.gov.uk/government/publications/summary-literature-review-on-iot-security/

[22] https://www.ncsc.gov.uk/

[23] https://www.gov.uk/government/collections/secure-by-design/

[24] https://www.gov.uk/government/publications/mapping-of-iot-security-recommendations-guidance-and-standards/

[25] Copper Horse Ltd. on behalf of DCMS, Mapping Securi-ty & Privacy in the Internet of Things; https://iotsecuritymapping.uk/

[26] GSM Association; https://www.gsma.com/

[27] GSMA, IoT Security Guidelines Overview Document; https://www.gsma.com/iot/iot-security-guidelines-overview-document/

[28] GSMA, IoT Security Guidelines for IoT Service Ecosystem; https://www.gsma.com/iot/iot-security-guidelines-for-iot-service-ecosystem/

[29] GSMA, IoT Security Guidelines Endpoint Ecosystem; https://www.gsma.com/iot/iot-security-guidelines-for-endpoint-ecosystem/

[30] GSMA, IoT Security Guidelines for Network Operators; https://www.gsma.com/iot/iot-security-guidelines-for-network-operators/

[31] GSMA, IoT Security Assessment Checklist, version 3.0, 30 September 2018; .zip file available at https://www.gsma.com/iot/iot-security-assessment/

[32] GSMA, IoT Security Assessment Process, version 2.0, 30 September 2018; .zip file available at https://www.gsma.com/iot/iot-security-assessment/

[33] https://www.enisa.europa.eu/

[34] https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot/good-practices-for-iot-and-smart-infrastructures-tool/

[35] https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/

[36] https://www.ctia.org/

[37] https://cloudsecurityalliance.org/

[38] https://www.iotsecurityfoundation.org/

[39] https://www.iotsecurityfoundation.org/best-practice-guidelines/

[40] IoTSF, IoT Security Compliance Framework, Release 2.1, May 2020; .zip file available at https://www.iotsecurityfoundation.org/wp-content/uploads/2020/05/IoTSF-IoT-Security-Compliance-Framework-Questionnaire-Release-2.1.zip

[41] IoTSF, Mapping the IoT Security Foundation's Compliance Framework to ETSI TS 103 645 Standard, February 2019; PDF available at https://www.iotsecurityfoundation.org/wp-content/uploads/2019/02/Mapping-the-IoTSF%E2%80%99s-Compliance-Framework-to-ETSI-TS-103-645-Standard.pdf

[42] https://www.w3.org/

[43] https://www.w3.org/WoT/

[44] W3C, Web of Things (WoT) Architecture, W3C Recom-mendation, 9 April 2020; https://www.w3.org/TR/wot-architecture/

[45] W3C, Web of Things (WoT) Thing Description, W3C Rec-ommendation, 9 April 2020 (link errors corrected 23 June 2020); https://www.w3.org/TR/wot-thing-description/

[46] ITU-T Recommendation Y.4000/Y.2060 (approved in 2012-06-15); SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS; Next Generation Networks – Frameworks and func-tional architecture models; Overview of the Internet of things (former ITU-T Y.2060 renumbered as ITU-T Y.4000 on 2016-02-05 without further modification and without being republished); https://www.itu.int/rec/T-REC-Y.4000

[47] https://www.itu.int/ITU-T/recommendations/index.aspx?ser=Y

[48] ISO/IEC 21823-1:2019, Internet of things (IoT) — Interop-erability for IoT systems — Part 1: Framework, February 2019; https://www.iso.org/standard/71885.html

[49] ISO/IEC 21823-2:2020, Internet of things (IoT) — Interop-erability for IoT systems — Part 2: Transport interoperabil-ity, April 2020; https://www.iso.org/standard/80986.html

[50] ISO/IEC 30141:2018, Internet of Things (loT) — Reference Architecture, first edition published August 2018 (sec-ond edition pending); https://www.iso.org/standard/65695.html

[51] https://www.ngiot.eu/event/ngiot-webinar-integrating-privacy-in-the-iot-ecosystem/

[52] https://cordis.europa.eu/project/id/825082

[53] Directive (EU) 2016/680 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of per-sonal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penal-ties, and on the free movement of such data, and re-pealing Council Framework Decision 2008/977/JHA; current consolidated version (2016-05-04) available at https://eur-lex.europa.eu/eli/dir/2016/680/2016-05-04

[54] ISO/IEC 27001:2013, Information technology — Security techniques — Information security management systems — Requirements; https://www.iso.org/standard/54534.html

[55] ISO/IEC 15408-1:2009, Information technology — Securi-ty techniques — Evaluation criteria for IT security — Part 1: Introduction and general model, December 2009 (cor-rected version January 2014); https://www.iso.org/standard/50341.html

[56] ISO/IEC 15408-2:2008, Information technology — Securi-ty techniques — Evaluation criteria for IT security — Part 2: Security functional components, August 2008 (corrected version May 2011); https://www.iso.org/standard/46414.html

[57] ISO/IEC 15408-3:2008, Information technology — Securi-ty techniques — Evaluation criteria for IT security — Part 3: Security assurance components, August 2008 (corrected version May 2011); https://www.iso.org/standard/46413.html

[58] ISO/IEC DIS 15408-4, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 4: Framework for the specification of evaluation methods and activities, under development; https://www.iso.org/standard/72913.html

[59] ISO/IEC DIS 15408-5, Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 5: Pre-defined packages of security requirements, under development; https://www.iso.org/standard/72917.html

[60] ISO/IEC 18045:2008, Information technology — Security techniques — Methodology for IT security evaluation, Au-gust 2008 (corrected version January 2014); https://www.iso.org/standard/46412.html

[61] Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and in-formation systems across the Union; https://eur-lex.europa.eu/eli/dir/2016/1148/oj

Semantic Models for Network Intrusion Detection

Peter Bednar, Martin Sarnovsky, Pavol Halas
Department of Artificial Inteligence and Cybernetics
Technical University of Kosice
Kosice, Slovakia
{name.surname}@tuke.sk

***Abstract*—The presented paper describes the design and validation of the hierarchical intrusion detection system (IDS), which combines machine learning approach with the knowledge-based methods. As the knowledge model, we have proposed the ontology of network attacks, which allow to us decompose detection and classification of the existing types of attacks or formalize detection rules for the new types. Designed IDS was evaluated on a widely used KDD 99 dataset and compared to similar approaches.**

***Keywords*—ontologies, network security incidents, machine learning**

I. Introduction

With the extensive usage of the information and communication technologies the number and variety of the security attacks grow. This is also reflected in the growing of budget invested by companies or public institutions into the security. In order to cope with the current situation, the new and innovative techniques are applied in order to automatize the security management [1].

Recently, we can observe two main approaches to the security of the ICT: the first approach is data-oriented, and it is based on the application of machine learning techniques to proactively achieve the best possible prediction of the new attacks [2][3][4][5]. The second approach is more user-centric and it is based on the application of knowledge modelling techniques in order to model user behavior and ICT environment [8][9][10].

The presented article tries to combine these two approaches into a single system, where the domain knowledge about the types, effects and severity of the attacks is used to decompose intrusion detection task into the classification subtasks which can be handled more efficiently with less training data. The design of the proposed intrusion detection system is symmetrical in the sense that both approaches (machine learning and knowledge based) are equal and mutually contribute to address the challenges of the detection and prevention of the security threats.

The rest of this paper is organized as follows: in the following chapter we will present hierarchical knowledge model in the form of the ontology which will be used for the decomposition of the detection problem and which will provide additional contextual information. Subsequent part describes implemented machine learning models and how these models are combined with the knowledge in the ontology. Subsequent section then presents the experimental evaluation of the proposed combined approach. In this chapter we at first define quantitative evaluation metrics and then summarize the performance of the system on the standard benchmark dataset from the KDD Cup competition.

II. Hierarchical knowledge-based intrusion detection system

A. *Overall system architecture*

The main objective of the proposed architecture is to hierarchically decompose detection and classification of the intrusions according to the types of the attacks. For the decomposition we have proposed the Network Intrusion Ontology which main part is formalized as the taxonomy of attack types. This ontology allows to capture all knowledge related to the known types of the attacks, including the description of rare cases which are difficult to detect using the machine learning methods.

The main decomposition of the detection and classification process can be divided into the following phases:

1. Coarse attack/normal classification - this phase is implemented using the machine learning algorithm which distinguish normal traffic and attacks. If a network connection is labelled as a normal one, then an alarm is not raised. Otherwise, the suspicious connection is processed by a set of models to determine the class of attack during the phase 2.
2. Attack class and type prediction—this phase is guided by the taxonomy of the attacks from the Network Intrusion Ontology. The system hierarchically processes the taxonomy and selects the appropriate model to classify the instance on a particular level of a class hierarchy. The model can be a machine learning model statistically inferred from the training data, or rule-based model formalized using the classes and relations from the ontology.
3. When a class of attack is predicted, ontology is queried for all relevant sub-types of the attack type and to retrieve the suitable model to predict the particular sub-type. Knowledge model can also be used to extract specific domain-related information as a new attribute, which could be used either to improve the classifier's performance or to provide context, domain-specific information which could complement the predictive model.

The details about the predictive models and their evaluation will be presented in the subsequent chapter.

B. *Network Intrusion Ontology*

The proposed knowledge model captures all essentials concepts required to describe network intrusion systems. We have designed our semantic model according to the methodology proposed by Grüninger and Fox and with some extensions from Methodology.

The designed ontology is formalized using the OWL 2 RL profile, which allows to formalize common constructs such as multiple hierarchies and at the same time provides compatibility with the rule languages for automatic reasoning. As the objective of the knowledge model was to use it in the data analytical tasks, the concepts and properties map directly to the data used in the process. Moreover, ontology was extended with the concepts related to the classification models, to create the relation between the particular classifier and its usability on the specific level of target attribute hierarchy. The main classes of ontology include:

- Connections - This class represents the status of each connection record. It specifies Attack connection or normal traffic. Attack connections are further conceptualized using the Attack hierarchy described below.
- Effects - This class contains subclasses that represent all possible consequences of individual attacks (e.g., slows down server response, execute commands as root, etc.).
- Mechanisms - The subclasses represent all possible causes of individual ontology attacks (poor environment sanitation, misconfiguration, etc.).
- Flags - The subclasses represent normal or error states of individual connections (Established, responder aborted, Connection attempt was rejected, etc.). Each of these subclasses has a 1 equivalent instance.
- Protocols - The class contains subclasses that represent the types of the communication protocols on which the connection is running (TCP, UDP, and ICMP).
- Services - The subclasses represent each type of connection service (http, telnet, etc. ...). Each of these subclasses has a 1 equivalent instance.
- Severities - This class represents the severity of the attack, its subclasses represent the severity level (weak, medium, and high).
- Targets - The subclasses represent possible targets of a given type of attack (user, network).
- Models concept covers the classification models used to predict the given target attribute.

The instances of the specified classes represent the network connections (e.g., connection records from the data set). Trained and serialized classification models are instantiated as the instances of the Model class. The models are represented as the web resources and they could be accessed by their URI property, which points to the location where the model is serialized in the system. The main concepts and relations of the ontology are represented on the Figure 1.

The central part of the proposed semantic model is the taxonomy of Attacks which are summarized in the following figure. The taxonomy was extracted from the types of the attacks described in the KDD 99 datasets. Attacks are divided into the four main groups such as DOS, R2L, U2R and PROBE. The main types of the attacks are further specified on the additional level of the hierarchy.

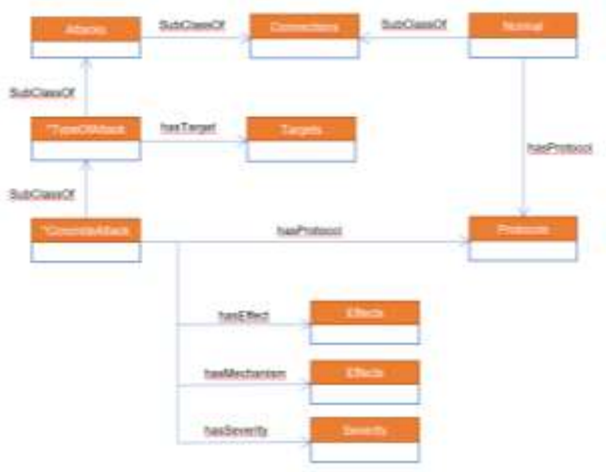


Fig. 1. The main concepts of the proposed sematnic model.

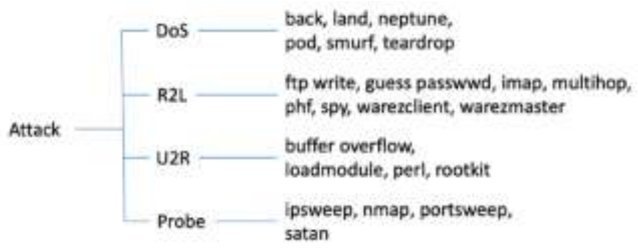

Attack
DoS
back, land, neptune, pod, smurf, teardrop
R2L
ftp write, guess passwwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R
buffer overflow, loadmodule, perl, rootkit
Probe
ipsweep, nmap, portsweep, satan

Fig. 2. The hierarchy of Attacks.

C. *Machine learning models*

To evaluate the proposed approach, we used the KDD Cup 1999 competition dataset, which is a commonly accepted benchmark for the intrusion detection task. The dataset consists of the records from the device logs in a LAN network collected over nine weeks. For the evaluation, we have used 10% sample with the 494,021 records in total. Each record is labeled as the normal communication or it is assigned to the major attack class and specific attack types. There are 22 different attack types which corresponds to the classes in the proposed ontology.

The common problem with the diagnostic tasks such as intrusion detection systems is that the target attribute (i.e. in our case type of the attack) is highly unbalanced with the majority of normal communication. Table I presents the taxonomy of attack types together with the number of cases in the dataset. Some attack classes such as Probe are more balanced but generally for each attack class we can find some minor types with only the few training examples. The lack of cases is problematic not only for the training of statistical models but also for the evaluation. On the other side, rare cases can be still very critical and can in overall a big impact on the security of the system.

TABLE I. ATTACK TYPES AND NUMBER OF SAMPLES

Attack	Attack class	# of samples
back	DoS	2203
land		21
neptune		107,201
pod		264
smurf		280,790
teardrop		979
satan	Probe	1589
ipsweap		1247
nmap		231
portsweep		1040
guess_passwd	R2L	53
ftp_write		8
imap		12
phf		4
multihop		7
warezmaster		20
warezclient		1020
spy		2
buffer_overflow	U2R	30
loadmodule		9
perl		3
rootkit		10
normal	Normal	97,227

The records for each connection are described by set of features, which are represented in the ontology as the data attributes. The features can be divided into the basic features, content features and traffic features. Overall there are 32 features. The first group describes the type of the communication protocol, duration of the connection, service on the destination network node and other standard attributes describing the TCP connection. Content features are attributes that can be linked to the domain specific knowledge depending on the applications and environment in which communication occurs. The last group of features (traffic) describe the communication attributes captured during the 2 seconds time window, e.g. the number of hosts communicating with the target host etc. For the data preprocessing, we have selected only the most relevant features for the classification which were identified in the work of [4]. The final list of features includes: service, src_bytes, dst_bytes, logged_in, num_file_creations, srv_diff_host_rate, dst_host_count, dst_host_diff_srv_rate, dst_host_srv_diff_host_rate, srv_count, serror_rate, rerror_rate,

Since the data of diagnostic tasks are commonly highly unbalanced towards the normal cases, the proposed approach is based on the decomposition of the diagnostic classification task into the hierarchy of classifiers. At the top level of the class hierarchy, an *attack detection model* is used for the prediction to distinguish between the attack connections and normal traffic. The classifier on this level was trained on the whole dataset and target attribute was transformed to the binary indicator. The main goal of this top-level classifier is to reliably separate normal connections from the attack ones.

If the top-level model detects an attack connection, the cases are further classified by the ensemble models into the one of the four types of the attack on the second level of the taxonomy (DoS, R2L, U2R, Probe). In this level, we use ensemble classifier with voting scheme trained on all attack instances (i.e. without the normal communication cases). We found that the proposed ensemble model is more efficient in the case of unbalanced target classes. The standard machine learning models proposed in the previous works were able to gain good accuracy, achieved mostly on the dominant class (in our case on KDD 99 dataset, on the most common DoS attack). However, the simple models struggled to predict minor classes such as U2R, which can be even more serious from the point of view of network security. For example, when training a decision tree model, the model has very good performance for the DoS and R2L classes but missed a significant amount of the Probe attacks and was not able to detect the U2R class at all.

Proposed weighting schema is based on the idea of complementing classifiers which is based on the performance of a particular model on the particular class. This weighting schema is presented on the Table II. The $w_{i,j}$ terms represent the weight associated with the *i*-th model and *j*-th class.

TABLE II. WEIGHTING SCHEME OF THE ENSEMBLE MODEL

Model	DoS	R2L	U2R	Probe
model 1	$w_{1,1}$	$w_{2,1}$	$w_{3,1}$	$w_{4,1}$
model 2	$w_{1,2}$	$w_{2,2}$	$w_{3,2}$	$w_{4,2}$
model 3	$w_{1,3}$	$w_{2,3}$	$w_{3,3}$	$w_{4,3}$
...	...	...	...	...

After the binary classification and classification of the attack class by ensemble weighted classifier, we have trained particular models to further classify specific type of the attack on the most specific level of the taxonomy. Four different models were trained using only the records of particular attack classes (i.e. models for DoS, R2L, U2R and Probe). The most problematic was minority U2R class, as the dataset contains very few records of that type. The final implemented classification schema is presented on the Figure 3. All models were implemented in the Python environment using the standard pandas and scikit-learn stack. Predictive models were then persistently stored and the models URIs (Uniform Resource Locators) were added as the data properties to the knowledge model.

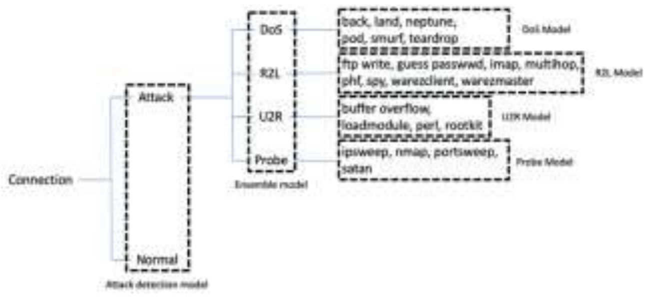


Fig. 3. The implemented hierarchical classification schema.

The main role of the semantic model in the proposed detection system is to navigate through the target class taxonomy and decompose classification problem to the sub-problems implemented by the particular models for the specific type of attack. The system is implemented using the Python language and RDFlib package which provides integration with the ontology using the SPARQL query interface. When predicting the unknown connection, system query the ontology using the SPARQL query and retrieve correspondent model for the particular class of the attacks according to the URL stored in the hasTargetAttribute property. Once the classification of the main type is performed, the system checks in the ontology if there is a classifier able to process the record further and to detect subtype of the attack.

Besides the hierarchical decomposition of the detection process, knowledge model provides also additional context which can be leveraged during the classification and improve detection of the minor classes. We have mainly extended the context with the potential effect of the attack. Additionally, if the models are not reliable enough to predict the concrete attack sub-type, the system can be used to classify attacks at least according to the severity which is retrieved from the knowledge model for the particular main class of the attack. This could serve as a supporting source of information, completing the attach type classification.

III. Evaluation

For the evaluation, we used the most common metrics employed in the classification tasks such as recall and precision. We have also computed confusion matrix for the particular classes of attacks. The confusion matrices were especially informative since they record number of correctly and incorrectly classified examples and also the types of the error. For the binary classification on the top level of the taxonomy hierarchy we used standard evaluation metrics:

- Precision: P = TP / (TP + FP)
- Recall: R = TP / (TP + FN)

where TP, TN, FP, FN are numbers of true positive, true negative, false positive and false negative records (e.g. for true positive number of records when the predicted attack was in fact attack, false positive when the predicted attack was in fact a normal traffic, false negative when the predicted normal traffic was in fact an attack, etc. The entire system was also evaluated with the number of missed attacks and raised false alarms as FAR metric (False Alarm Rate), which corresponds to the false positive records divided by total number of normal traffic records (true negative + false positive).

For the evaluation of the binary classification on the top level of the taxonomy, we used directly precision and recall metrics. In the subsequent stages on the more specific levels of taxonomy we have computed precision and recall for each class and used macro-averaging for overall evaluation. Additionally, we have computed multi-class confusion matrix to further investigate the types of the errors produced by the system.

A. *Training and evaluation*

For the binary classification for the attack detection, we used the decision tree classifier. Dataset includes all records and target attribute was transformed to binary indicator attack/normal traffic. The classifier was trained without the limit for maximum depth with default settings for pruning and gini index as the splitting criterion. We split the dataset randomly to 70/30 training/testing ration. The testing data were also used for overall evaluation of the entire system. Model for the binary classification achieved the accuracy 0.9997. The detailed confusion matrix is presented in the Table III.

TABLE III. Performance of the binary attack classification

	Normal	**Attack**	**Precision**	**Recall**
Normal	29,095	11	0.999	0.999
Attack	35	119,066		

For the training of ensemble classifier, we have selected only the attack records from the training set. As the base classifiers we have used various configuration of the Naive Bayes and Decision Tree models. The experiments proved that the Decision Tree classifier performed well on the Probe, DoS and R2L attacks. On the other hand, for the U2R class model produces many false alarms or (depending on pruning) the model was not able to detect U2R attacks at all. For this reason, we have trained one-vs-all model just to separate U2R class. We have then combined both types of the models into the ensemble classifier. The weights of the base classifiers were computed according to the accuracies of the models on the training data. For the evaluation we have used the same 70/30 dataset split as for the binary classification, but we have further selected only the attack records (since the normal communication is filtered already by the binary classifier). In total, models were trained on 396743 records. The confusion matrix of the ensemble classifier is presented on the Table IV.

TABLE IV. Performance of the ensemble attack classification

	Probe	**U2R**	**DoS**	**R2L**	**Prec.**	**Rec.**
Probe	1279	0	1	0	0.992	0.992
U2R	0	15	0	0	1	0.882
DoS	6	0	117,385	0	0.999	0.999

R2L	4	2	0	331	0.982	1

On the most specific level of the taxonomy, each major attack class has dedicated one model for the further classification of subtypes. The performance of each model was evaluated using the precision and recall macro-averaged for each subtype. The overall performance of the models is summarized in Table V.

TABLE V. PERFORMANCE OF THE SUBTYPE CLASSIFICATION

	Probe	**U2R**	**DoS**	**R2L**
Accuracy	0.991	0.937	0.999	0.989
Precision	0.989	0.927	0.999	0.879
Recall	0.989	0.875	0.999	0.833

The overall system with the hierarchical classification was evaluated using the standard precision, recall F-measure and FAR (False Alarm Rate) metrics. Comparison of the proposed system and models published in previous works [4][6][7][11] is presented in Table VI.

TABLE VI. OVERALL PREFORMANCE OF THE SYSTEM

Classifier	**Acc.**	**Prec.**	**F1**	**FAR**
C4.5	0.969	0.947	0.970	0.005
Random forests	0.964	0.998	0.986	0.025
Forest PA	0.975	0.998	0.998	0.002
Ensemble model	0.976	0.998	0.998	0.001
Our approach	**0.998**	0.998	0.998	0.001

Additionally, we have computed confusion matrix, which summarizes the performance for each attack class. The confusion matrix is presented in the Table VII.

TABLE VII. CONFUSION MATRIX FOR THE OVERALL PREFORMANCE OF THE SYSTEM

	Probe	U2R	DoS	R2L	Normal
Probe	1176	0	5	0	7
U2R	0	15	0	0	5
DoS	4	0	117547	0	1
R2L	3	1	0	346	7
Normal	1	0	3	1	48454

Besides the classification of attack types, we have implemented and also evaluated the classification of the attack severity. To train the severity detector we have used 10 % of KDD 99 dataset with the 70/30 training/testing ratio. The severity classifier was applied complementary to the ensemble model for the detection of the attack class. Overall achieved performance was 0.999 precision and recall with very good accuracy for the high and low severity. The Table VIII presents the confusion matrix for the severity detection in comparison for each class of the attack.

TABLE VIII. CONFUSION MATRIX FOR THE SEVERITY DETECTION

	High	Low	Medium	Prec.	Recall
DoS	117695	0	0	0.999	0.999
Probe	443	0	779		
R2L	0	346	6		
U2R	0	0	20		

Medium severity was biased by our model towards the high severity which has the similar effect like the higher false positive rate. Further details and information about the designed model were published in [9].

IV. CONCLUSION AND FUTURE WORK

In this paper we have proposed an approach based on the combination of knowledge based and machine learning methods for intrusion detection. The proposed knowledge model in the form of the ontology is used for the hierarchical decomposition of the detection process according to the types of the attack. This decomposition allows to overcome the problems with the unbalanced training data which are typical for the diagnostic machine learning tasks. By the leveraging of the domain knowledge, our combined approach also provides an additional context which includes for example the effects and severity of the attacks.

The performance of the proposed IDS is 0.998 in terms of precision as well as recall and 0.001 in terms of FAR metric, which on the standard benchmark dataset outperforms other state-of-the-art methods. Moreover, the proposed method has also potential to partially detect new emerging types of attacks in terms of the contextual information stored in the knowledge model.

In the future work we plan to extend the role of the knowledge model by introducing a rule-based classifier which will be based on the declarative rules and application of automatic reasoning technique and logical programming. We hope that this will allow to further improve accuracy for minor classes with the low number of training examples. Additionally, extended knowledge model will allow to create formalized knowledge base of the existing cases.

ACKNOWLEDGMENT

This work was supported by Slovak Research and Development Agency under the contract No. APVV-16-0213 and by the VEGA project under grant No. 1/0493/16.

References

[1] Park, J. Advances in Future Internet and the Industrial Internet of Things. *Symmetry* 2019, *11*, 244.

[2] Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York, NY, USA, 3-5 December 2016.

[3] Khan, M.A.; Karim, M.d.R.; Kim, Y. A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network. Symmetry 2019, 11, 583.

[4] Zhou, Y.; Cheng, G; Jiang, S.; dai, M. An efficient detection system based on feature selection and ensemble classifier. arXiv 2019, arXiv:190401352

[5] Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J. Comput. Sci. 2018, 25, 152–160.

[6] Sharma, N.; Mukherjee, S. A Novel Multi-Classifier Layered Approach to Improve Minority Attack Detection in IDS. *Procedia Technol.* 2012, *6*, 913–921.

[7] Ahmim, A.; Ghoualmi Zine, N. A new hierarchical intrusion detection system based on a binary tree of classifiers. Inf. Comput. Secur. 2015, 23, 31–57.

[8] Abdoli, F.; Kahani, M. Ontology-based distributed intrusion detection system. In Proceedings of the 2009 14th International CSI Computer Conference, Tehran, Iran, 20–21 October 2009; pp. 65–70.

[9] Sarnovsky, M.; Paralic, J. Hierarchical Intrusion Detection Using Machine Learning and Knowledge Model. *Symmetry* 2020, *12*, 203.

[10] More, S.; Matthews, M.; Joshi, A.; Finin, T. A Knowledge-Based Approach to Intrusion Detection Modeling. In Proceedings of the 2012 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, 24–25 May 2012; pp. 75–81.

[11] Özgür, A.; Erdem, H. A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ Preprints 2016, 4, e1954v1.

Smart Building Energy and Comfort Management Based on Sensor Activity Recognition and Prediction

Francesca Marcello, Virginia Pilloni
Department of Electrical and Electronic Engineering - University of Cagliari - Italy
National Telecommunication Inter University Consortium - Research Unit of Cagliari - Italy
Email: {francesca.marcello,virginia.pilloni}@unica.it

***Abstract*—Thanks to Building Energy and Comfort Managements (BECM) systems, it is possible monitor and control buildings with the aim to ease appliance management and at the same time ensuring efficient use of them from the energetic point of view. To develop such kind of systems, it is necessary to monitor users' habits, learning their preferences and predicting their sequences of performed activities and appliance usage during the day. To this aim, in this paper a system capable of controlling home appliances according to user preferences and trying to reduce energy consumption is proposed. The main objective of the system is to learn users' daily behaviour and to be able to predict their future activities basing on statistical data about the activities they usually perform. The system can then execute a scheduling algorithm of the appliances based on the expected energy consumption and user annoyance related with shifting the appliance starting time from their preferred one.**

Experimental results demonstrate that thanks to the scheduling algorithm energy cost can be reduced of 50.43% and 49.2% depending on different tariffs, just by shifting the use of the appliance to time periods of non-peak hours. Scheduling based on probability evaluation of preferred time of usage of the appliances allows to still obtain evident energy savings even considering the errors on predicted activities.

***Index Terms*—Activity Recognition; Activity Prediction; Energy Management; Comfort Management; Smart Building**

I. Introduction

Smart buildings are characterised by the presence of sensors, actuators and smart devices that give the opportunity to monitor and control, either manually or automatically, key equipment within buildings [1]. This is the concept behind Smart Building Energy and Comfort Management (BECM) systems [2][3]. As a matter of fact, domestic electricity usage accounts for about 40% of the global energy consumption and contributes over 30% of total greenhouse gas emissions [4]. Nevertheless, user comfort is crucial when policies of Demand-Side Management (DSM) are put in place [5]. In such an intelligent scenario, one of the major goals is to provide users with tools that support cost-effective solutions to appliance management, which: i) demand the lowest effort in terms of training and management, dynamically adapting to user requirements, and ii) take into account user habits so that appliance management decisions do not conflict with them, causing a disaffection that may lead the user to turn off the system [6].

Currently, most of the literature considers user comfort as a set of hard constraints on appliance usage, which are a priori set considering general statistics [7][8]. This approach neglects the fact that users are likely not only to have different subjective requirements with respect to the others, but they also dynamically change over time.

In this paper, user preferences and habits about appliance usage are continuously monitored, recognised and predicted, by means of a BECM system based on sensors deployed inside the reference building. The system merges two previous studies about activity recognition [9] and appliance scheduling [6], by including the crucial activity prediction functionality. Indeed, activity prediction enables appliance scheduling by predicting which appliances are likely to be used in the following hours and scheduling them in advance, so that their starting time is shifted to off-peak times when electricity tariffs are lower.

The main contributions provided by this paper can be summarised as follows:

- an activity recognition algorithm used to model user profiles, which was first proposed in [9] and whose accuracy is here improved;
- an activity prediction algorithm is proposed, along with statistics about mutual correlation of activities. Accordingly, appliance usage is predicted for a specified time window (test were run for a time window of 6 hours);
- user profile and activity prediction are incorporated into the user-annoyance-aware energy-cost-saving appliance scheduling algorithm proposed in [6].

To the best of the authors' knowledge, this is the first comprehensive system to use sensor-based activity prediction and occupants' preference inference, and integrate them into a BECM. Accordingly, based on simulations of the system on a real dataset, this paper further analyses how the proposed system affects energy-related costs.

The remainder of the paper is organised as follows. Section II presents past works and the required background. In Section III an overview of the proposed system model is provided. Section IV describes the reference use case considered to test the performance of the system. Finally, in Section V a performance analysis is provided. Conclusions and final remarks are drawn in Section VI.

II. Related Works

Smart technologies can be used in all kinds of different buildings (i.e., residential, office, and retail sectors) to improve the comfort and the safety of people in their home, concerning various topics, from healthcare and providing living assistance, to environmental monitoring and ensuring energy saving. Accordingly, BECM systems have the objective of combining power consumption minimisation while preserving user comfort [10][11]. This issue has been addressed by researchers from many different perspectives. The authors in [10], present a review of control systems for energy management and comfort in buildings, where the quality of the comfort is considered mostly dependent on thermal comfort, indoor air quality and visual comfort, explaining current and conventional controller solutions and their disadvantages. Also in [12][13] two different solutions for building management considering user preference in terms of indoor environment comfort are presented. In [14], an algorithm for thermostatically controlled household loads based on price and consumption forecasts of grid energy is presented. The issue of scheduling appliances according to user preferences was also addressed by [6], where Quality of Experience (QoE) is measured as a function of the interval between the preferred and proposed appliance starting time for switching controlled loads (e.g., washing machines and clothes dryers), and as a function of the interval between the preferred and proposed temperature for thermostatically controlled loads (e.g., conditioning systems and water heaters).

It is evident that user preferences and habits severely affect results of BECM systems. For this reason, in recent years researchers have started to observe users' behaviour, in order to infer their habits and preferences. The monitoring of activities of people in their home can be done by analysing data that can be gathered with different technologies. Different studies proposed solutions based on using cameras and wearable sensors or gathering data provided by phone accelerometer and gyroscope [15][16]. These solutions are not very practical in home scenarios where people are often not inclined to accept those devices. To monitor what activities people are performing in their house, non-intrusive sensors are often preferred: typical devices that are installed in the environment are motion sensors, door sensors or temperature and pressure sensors [17]. The data collected from sensors inside resident houses are analysed using data mining and machine learning techniques to build activity models that are used as the basis of behavioural activity recognition.

With reference to modelling and classification methods, researchers have investigated the recognition of resident activities using a variety of mechanisms, such as Naïve Bayes classifiers, Markov models, and dynamic Bayes networks. In multiple cases, in spite of its simple design and simplified assumptions, Naïve Bayes classifiers often work much better than expected, especially when a specific group of sensors can easily be identified as characteristic of a certain activity [18].

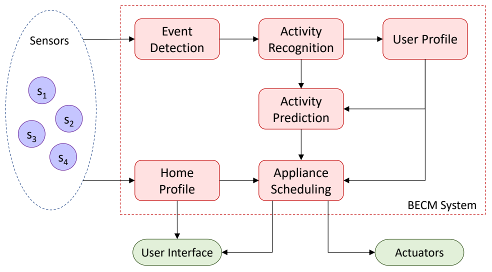


Fig. 1. Overview of the proposed BECM system

III. System Model

In this paper, the considered scenario is that of a BECM system, based on distributed smart home sensor networks. An overview of this system is represented in Figure 1. More specifically, sensors are used to make observations on users and their interactions with the surrounding environment; the combinations of these interactions, which are detected by the *Event Detection* module as events, provide meaningful information on users' activities. As described in more details in [9], after a training period the *Activity Recognition* module can correctly recognise activities with an accuracy of more than 80% on average. Accordingly, a correlation can be observed between detected events and activities, which can be used to infer users' habits. These habits, stored in the *User Profile* module, are used with information about previously recognised activities by the *Activity Prediction* module, with the aim to predict the following activities that are expected to be carried out by the users. The information related to activity prediction, user profile and home profile are then processed by the *Appliance Scheduling* module to find a scheduling for controllable appliances that corresponds to the best trade-off between energy cost reduction and user comfort. Note that the *Home Profile* module stores home-related information collected by sensors and/or through user interfaces, such as electricity tariffs, and which appliances are installed along with their energy consumption characteristics.

In the following, more details will be provided about the core modules of the proposed BECM system, i.e. the Activity Recognition module, the Activity Prediction Module and the Appliance Scheduling module.

A. *The Activity Recognition Module*

The activity recognition approach used in this paper was earlier proposed in [9]. It encompasses two phases: i) *training*, during which the system learns the association between activities and their instances, i.e. sequences of detected events; ii) *running*, which uses the probabilistic model created during the training phase to associate an activity to the detected events.

a) Training phase: for each k-th activity instance $\mathcal{I}_{jk}$ of activity $\mathcal{A}_j$ observed during an observation time window $\mathcal{O}^A$, a feature vector $\mathcal{F}_{jk}(\mathcal{I}_{jk}) = [f_{1jk}, f_{2jk}, \ldots, f_{ijk}, \ldots]$ is computed with the rates of detected event occurrences, that is the number of events related to one specific sensor with respect to the total number of events observed considering all the sensors within $\mathcal{O}^{A}$. Then, for each activity $\mathcal{A}_j$, a model vector $\boldsymbol{m_j} = \text{mean}_k(\boldsymbol{\mathcal{F}}_{jk}) = [\overline{f}_{1jk}, \overline{f}_{2jk}, \dots, \overline{f}_{ijk}]$ is defined such that the rates of event occurrences of its sensors is the average rate for all the observed instances associated with the same activity.

b) Running phase: it relies on the use of a sensor-based windowing implementation [18], according to which sequences of detected events are divided into subsequences using an observation window $\mathcal{O}^{\mathcal{W}}(t)$ starting at time t, which contains a certain number of events equal to its size $\mathcal{W}$. Each subsequence $\boldsymbol{z}$ of events is then associated with a feature vector $\boldsymbol{\mathcal{F}}_{\boldsymbol{z}}^{\boldsymbol{W}}$, computed analogously to $\boldsymbol{m_j}$. Finally, the sequences of detected events are classified based on their probability to belong to a given activity.

For further details the reader is referred to [9].

B. The Activity Prediction Module

The main task of the the *Activity Prediction* module is to provide, for the next *Appliance Scheduling* module, a possible scenario in time t ahead in the future, that explains the probabilities of every activity that can begin in t, calculated thanks to information about all the activities happened and recognised before the current time t_0. Starting from the assumption that activities are linked between one another, so that when an activity $\mathcal{A}_i$ occurred at time t_i it is possible to evaluate the probability of another activity $\mathcal{A}_j$ to be performed by the user in a different time t_j, the module has to evaluate the probability in t for every single activity $\mathcal{A}_j$.

During the training phase, two different kind of probabilities have been evaluated for every activity under consideration:

- $p(\mathcal{A}_i(t_i))$ indicate the prior probability for activity $\mathcal{A}_i$ of starting at time t_i;
- $p((\mathcal{A}_j|\mathcal{A}_i)(k\Delta t))$ indicate the conditional probability of activity $\mathcal{A}_j$ of being carried out since activity $\mathcal{A}_i$ has started $(k\Delta t)$ before.

The period between the current time t_0 and the time t in which the prediction is needed, is split in different k time intervals of duration (Δt).

All the activities that have been recognised before the current time t_0 are stored with their respective starting time t_i, so that it is known how many time intervals outdistance every t_i up to t, and it is possible to calculate the conditional probability of activity $\mathcal{A}_j$ in t considering all the activities $\mathcal{A}_i$ that occurred in t_i. Then, for every $(k\Delta t)$ between t_0 and t, the probability of $\mathcal{A}_j$ to be happening in t is calculated with respect to the fact that $\mathcal{A}_i$ could be happening in $(k\Delta t)$. These two contributions are added together according to the equation below:

$$\begin{gathered} p(\mathcal{A}_j(t)) = \sum_i p((\mathcal{A}_j|\mathcal{A}_i)(\frac{t-t_i}{\Delta t}))+ \\ \sum_i \sum_k p((\mathcal{A}_j|\mathcal{A}_i)(k\Delta t)) \cdot p(\mathcal{A}_i(t_0+k\Delta t)) \end{gathered} \tag{1}$$

with $k \in \{0, (t-t_0)/\Delta t\}$.

Every activity coincides with one of the appliances in the house, so that the probability for each activity in t, calculated as explained, is then translated in the probability of one appliance to be used at time t. Therefore, the output from this module is going to enable the scheduling algorithm to make the validation necessary for the scheduling of controllable appliances and for evaluating energy consumption. The algorithm decides to schedule at time t only those appliances corresponding to activities that have their value of probability higher than a certain threshold.

C. The Appliance Scheduling Module

The appliance scheduling algorithm is based on the smart home energy management system proposed in [6]. This system dynamically shifts tasks of controlled appliances to times when it is more convenient (e.g. off-peak times), after finding a trade off between the overall energy cost and the annoyance experienced by users as a consequence of this shift. Accordingly, appliances are subdivided into three groups:

G1: not controlled loads, i.e., small loads such as lights, and not controlled high loads such as fridges;
G2: switching controlled high loads, such as washing machines and dishwashers;
G3: thermostatically controlled high loads, i.e. appliances that are controlled by a thermostat, such as water heaters.

The energy consumption for an appliance i is defined as $E_i^{cons} = P_i^{cons} \times t_i^{exec}$, where P_i^{cons} is its power consumption and t_i^{exec} is its execution time. While for switching controlled loads the execution time corresponds to a complete working cycle, for thermostatically controlled ones it depends on appliance characteristic parameters and temperature conditions. As described in more details in [6], the execution time of G3 appliances to reach a temperature T_i^{exp} is defined as

$$t_i^{exec}(T_i^{exp}) = -R_i C_i \ln\left(\frac{T_i^{out} - T_i^{exp} + R_i P_i^{heat}}{T_i^{out} - T_i^{in} + R_i P_i^{heat}}\right) \tag{2}$$

where $T_i^{out}(t)$ and $T_i^{in}(t)$ P_i^{heat} are the initial outside and inside temperature respectively, and P_i^{heat}, R_i and C_i are characteristic parameters for the appliance. More specifically, P_i^{heat} is the heat rate (in Watt), R_i is the equivalent thermal resistance (°C/Watt) and C_i is the equivalent heat capacity (Joule/°C). If the appliance is off, $P_i^{heat} = 0$.

The appliance scheduling algorithm then schedules appliances according to their related cost contribution value, which includes both the energy consumption- and user annoyance-related costs. User annoyance is computed according to the results of a survey, completed by 427 people, as reported in [6].

For G2 appliances, the cost to start at time t_i^{ST} and end at time $t_i^{END} = t_i^{ST} + t_i^{exec}$ is defined as

$$C_i^{G2}(t_i^{ST}) = \frac{P_i^{cons}}{\sigma\left(\Delta t_i^{ST}\right)} \cdot \sum_{t=t_i^{ST}}^{t_i^{END}} \Phi(t) \tag{3}$$

where $\Phi(t)$ is the electricity tariff at time t, and $\sigma\left(\Delta t_i^{ST}\right)$ is the relative satisfaction level for a time interval $\Delta t_i^{ST} = t_i^{ST} -$

t_i^{PT}, which is in inverse proportion with the user annoyance of shifting the appliance starting time. If $\sigma\left(\Delta t_i^{ST}\right)=0$, the cost value $C_i^{G2}(t_i^{ST}) \to \infty$.

For G3 appliances, the cost to start at time t_i^{ST} and end at time $t_i^{END} = t_i^{ST} + t_i^{exec}\left(T_i^{exp}\right)$ is defined as

$$C_i^{G3}(t_i^{ST}, t_i^{END}) = \frac{2 \cdot P_i^{cons}}{\sigma\left(\Delta T_i^{ST}\right) + \sigma\left(\Delta T_i^{exp}\right)} \cdot \sum_{t=t_i^{ST}}^{t_i^{END}} \Phi(t) \tag{4}$$

where $\sigma\left(\Delta T_i^{ST}\right)$ and $\sigma\left(\Delta T_i^{exp}\right)$ are the relative satisfaction values for a difference in temperature respectively of $\Delta T_i^{ST} = T_i^{in}(t_i^{ST}) - T_i^{PT}$ between the temperature at the starting time and the preferred temperature, and of $\Delta T_i^{exp} = T_i^{exp} - T_i^{PT}$ between the temperature expected at the ending time and the preferred temperature. Also in this case, if $\sigma\left(\Delta T_i^{exp}\right)=0$, the cost value is $C_i^{G3}(t_i^{ST}, t_i^{END}) \to \infty$.

For further details about this appliance scheduling system, the reader is referred to [6].

IV. Reference Use Case

The algorithm for modelling the activities and then discovering what the resident is doing is implemented and tested using the Aruba real-word dataset from the CASAS smart environment project of the Washington State University [19]. The data were collected from one smart apartment provided with motion sensors, contact sensors in the doors or cabinets and temperature sensors. The events decoded by these sensors are significant for recording elementary actions that people are performing, while the aggregation of these elementary actions defines one activity of interest. To correctly evaluate the correlation between the sets of events and the observed user's activities, without interference from other people, a dataset with only one resident living in the home was considered.

To evaluate the proposed system, in addition to the activities of the Aruba real-word dataset, some other activities have been simulated as performed by the same user inside this home scenario, using the same kind of sensors already installed in the house. The simulated activities are the following three activities not reported in the real dataset: using the washing machine, using the dish washer, taking a shower, which, along with the activity of washing dishes by hand, causes the water heater to turn on. Taking a shower is supposed to be carried out by the user in the bathroom, therefore involving the motion sensors already installed close to this room and assuming that hot water is used, thus causing the water heater to switch on. The use of the dish washer is supposed to be performed in the kitchen, involving the sensors in that area and simulating the presence of a specific cabinet containing the appropriate detergent and with a magnet sensor to understand its opening or closing, so that the activity of loading the dish washer could be recognised concluded only when this cabinet had been closed. The same thing was done for the activity of using the washing machine, by setting up another specific cabinet with its magnetic sensor, and placing it in a room of the house where there are not other linked activities.

TABLE I
Correspondence between activities and home appliances

	Activity	Appliance	Appliance type
1	Housekeeping (HK)	Vacuum Cleaner	G1
2	Meal Preparation (MP)	Microwave Oven	G1
3	Relax (Rel)	TV	G1
4	Wash Dishes (WD)	Water Heater	G3
5	Work	Laptop/Pc	G1
6	Taking Shower (TS)	Water Heater	G3
7	Laundry	Washing Machine	G2
8	Wash Dishes with Dish Washer	Dish Washer	G2
9	Always on	Fridge/Freezer	G1
10	Always on when user is at home/not sleeping	Lighting	G1

The system needs a correspondence between some of the activities and the use of certain household appliances, in order to predict energy consumption based on the probability of the activities to occur. Table I shows the considered activities along with their corresponding appliance owned by the user.

V. Experiments and Results

A. Activity Recognition and Prediction Algorithm

With respect to only the four activities corresponding to controllable appliances, i.e. appliances belonging either to G2 or G3, the recognition algorithm presented in [9] has an accuracy of 100% in recognising the activities of taking the shower and using the washing machine, while for the activity of using the dish washer it has an accuracy of 66.7% and for the activity of wash dishes by hand it gives an accuracy of 69.7%. This result is due to the fact that these two last activities are more difficult to recognize because they involve many of the kitchen sensors, which are also associated with other possible activities. The overall accuracy of the activity recognition algorithm is of 83.2%.

As for the prediction of future activities, the algorithm has an overall accuracy of 67%. The activities more accurately predicted are those with many samples and recurrent starting time, like the activity "Taking a Shower", because statistics about them are quite significant. For other less frequent activities, i.e "Wash Dishes", the prediction is instead less reliable.

B. Scheduling Algorithm

The algorithm has been compared with two different situations with respect to the case where no scheduling is involved. There are then three possible scenarios:

- the first one is the classic situation where appliances are normally used by the resident and the scheduling is never programmed (Without Scheduling Algorithm-WSA);
- the second one is based on a perfect knowledge of the time in which the user wants to use some of the

appliances in the house (Scheduling Based on Perfect Time-SBPT). This case coincides with the possible scenario in which the user instructs the system about the exact moment they want the appliance to start, but it has the disadvantage of requiring continuous interactions between users and system;

- the last one bases its scheduling evaluations on the probability of using any of the appliance at time t, calculated as explained in equation 1 (Scheduling Based on Probability-SBP). This solution allows to avoid interactions between users and the system, considering only the system's previous knowledge about user habits.

The training phase to obtain all the information about user's behaviours and preferences, and that allowed the system to perform the calculations on the probabilities indicated in subsection III-B, took into consideration two months of data about performed activities. Due to the fact that the shortest duration for the activities in exam is around 15 minutes, while the longest activities can elapse for several hours, conditional probabilities between activities was valuated choosing duration of intervals (Δt) equal to 15 minutes. The scheduling algorithm was instead tested on one week of data. For every interval of time t_0 during the testing week, the algorithm schedules appliances that are going to be used every $k\Delta t$ time intervals after t_0, with $k \in \mathbb{N}$, trying to improve energy savings and user's comfort. Simulations were done considering time intervals of 30 minutes and predicting future activities in t up to 9 hours forward in the future, so that every half an hour the scheduling algorithm could re-evaluate its scheduling based on the new information about previously user performed activities and with new calculations of probabilities $p(\mathcal{A}_j(t))$. The obtained results were considered with respect to energy consumption in one week, comparing the case with scheduling in relation to the case of normal use of household appliances, and evaluating if the scheduling could generate some kind of annoyance for the user. The evaluation of the energy costs has been made using two different tariffs listed in Table II, based on some typical Italian tariffs. The annoyance rate is defined as in [6], in relation to a possible shifting of appliance starting time or, with reference to the water heater, to a variation in the water temperature with respect to the user preferred temperature of use. Value 1 of annoyance indicates that there is not any discomfort for the user in the change of time in which the appliance was turned on, while a value of 5 indicates the highest level of annoyance for the user. Annoyance levels are modelled as a normal distribution with 15% deviation.

Table III shows the results about energy saving comparing the two cases with scheduling against the case without scheduling. These results are obtained taking into account the fact that cost savings are coming from a scheduling of switching controlled high loads to hours where the energy has lower prices and considering that there is a reduction in energy consumption due to a better optimisation in the usage of the water heater, which is switched on only at times of interest for the user and not every time the temperature drops below a certain value. Depending on the different tariff considered, energy consumption was calculated to be decreased of 50.4% with tariff 1 and of 49.2% with tariff 2, in contrast to the energy consumption with a classic use of appliances and energy over the week. As expected, greatest savings are obtained when there is a greater pricing difference between the higher cost range and the lower cost range. In particular, thanks to the scheduling, there is an evident better use of the water heater, since this appliance is scheduled and turned on only for the strictly necessary duration of time to obtain the water to be heated enough for when the resident needs to use it. This result can be verified in Fig.3, which represents the energy saving over the week differentiated by three of the appliances of the house: the water heater, the washing machine and the dish washer. Only those three appliances are considered because they are the only ones owned in the house that belong to groups G2 and G3 and that can be scheduled: the other appliances possessed by the user are part of G1 group. From Fig. 3 it is evident how most of the savings come from the scheduling of the water heater, while there is a lower incidence from washing machine and dish washer. This is explained by the fact that the preferred times of using those two appliances are already evaluated as the best compromise between energy consumption and user comfort, especially because in most cases they are very distant in time compared to the periods of non-peak hours. In fact, in Fig.2, where the average annoyance rate is presented, it is possible to observe how for every appliance the annoyance rate is always close to the lowest value of 1. The knowledge of user behaviours has therefore guaranteed the scheduling of the appliance with the best trade-off between energy costs and user preferences.

A slightly different discussion has to be done with reference to the scenario in which the scheduling is evaluated based on the probabilities of activities and, accordingly, on the

TABLE II
ENERGY PRICING

Weekends, holidays and everyday from 19:00 to 8:00	**Everyday from 8:00 to 19:00**
Tariff 1 0.0534 €/kWh	Tariff 1 0.07666 €/kWh
Tariff 2 0.067990 €/kWh	Tariff 2 0.07666 €/kWh

TABLE III
ENERGY CONSUMPTION FOR DIFFERENT SCENARIOS

	WSA	**SBPT**	**SBP**
Energy consumption in kWh/week	65.43	42.43	35.83
Cost Saving with Tariff 1	-	50.4%	64.7%
Cost Saving with Tariff 2	-	49.2%	63.18%

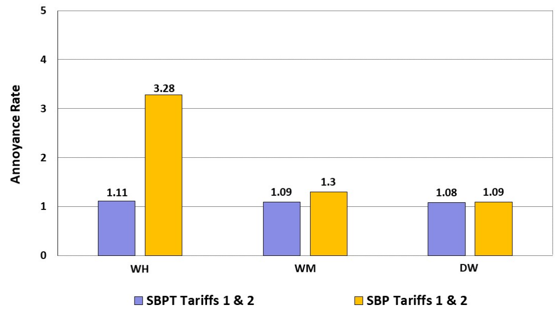


Fig. 2. Average annoyance rate with the proposed system

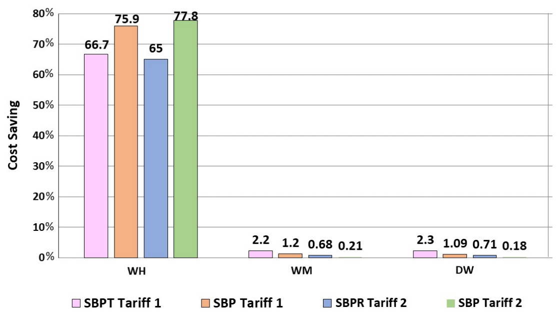


Fig. 3. Energy saving comparison differentiated by appliance

probabilities of using a certain appliance. Even in this case there is an evident reduction in energy consumption during the week, as shown in table III. Most of the saving come again from the wiser use of the water heater thanks to the scheduling only at appropriate time. Looking at Fig.2 it is possible to see, however, how the annoyance rate reaches a higher level. This is due to the fact that the use of the water heater is linked to two different activities, as shown in Table I. While activity 6 is always easily recognised and predicted, and therefore scheduled, activity 4 gives some problems because it is often confused with other activities [9]. Additionally, activity 4 is an activity that the user does not carry out often so there is not much statistical data on it. This last problem is common to the other two activities in exam, and this explains why even for this appliances there are higher level of annoyance rate due to the fact that the prediction module has made an error evaluating the probability of this activity to be performed. The algorithm has otherwise proved that the prevision about future activities can still ensure a good evaluation for the scheduling when the statistical data are reliable.

VI. Conclusion and Future Work

This paper focuses on a solution for energy and comfort management inside buildings, with the purpose of reducing energy waste thanks to a proper control over appliances, while on the same time ensure the well-being of users. To this aim, a BECM system is proposed that integrates a solution for two different problem: the first one concerns the needs for such a system to be able to know users behaviour and preferences and to predict usual activities; the second is about the necessity to manage appliances with respect of that behaviours and preferences and with respect of energy consumption.

The system has been tested in a real scenario, evaluating if the predictions were correct and proposing a coherent scheduling that could guarantee energy savings. The obtaining results show that, as expected, the scheduling of the appliances can guaranteed energy savings, reducing consumption over a week of at least 49.2% in comparison with classic use of energy and appliances. The prediction module permitted a quite accurate scheduling basing on probabilities, even if some of the activities has given some problem due to the fact that the statistic data about them were based of few instances. Furthermore, it was possible to guarantee that the annoyance rate was never too high, thus respecting user comfort.

Future works will investigate the adaptability of the proposed system to different real-case scenario, trying to improve the prediction module considering a larger training phase and more instances of the activities and corresponding use of appliances. Furthermore, it will be evaluated how the presence of Renewable Energy Sources could affect appliance scheduling and improve energy savings.

Acknowledgements

This work was partially funded by the INSIEME MiSE project, HORIZON 2020, PON 2014/2020 POS. 395, and by the LEAPH project, POR FESR Sardegna 2014-2020, Asse 1, Azione 1.1.3.

References

[1] D. Minoli, K. Sohraby, and B. Occhiogrosso, “Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems,” *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 269–283, 2017.

[2] A. I. Dounis and C. Caraiscos, “Advanced control systems engineering for energy and comfort management in a building environment—a review,” *Renewable and Sustainable Energy Reviews*, vol. 13, no. 6-7, pp. 1246–1261, 2009.

[3] M. Shafie-Khah and P. Siano, “A stochastic home energy management system considering satisfaction cost and response fatigue,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 629–638, 2017.

[4] L. Yang, H. Yan, and J. C. Lam, “Thermal comfort and building energy consumption implications–a review,” *Applied energy*, vol. 115, pp. 164–173, 2014.

[5] M. H. Shoreh, P. Siano *et al.*, “A survey of industrial applications of demand response,” *Electric Power Systems Research*, vol. 141, pp. 31–49, 2016.

[6] V. Pilloni, A. Floris *et al.*, “Smart home energy management including renewable sources: A qoe-driven approach,” *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 2006–2018, 2016.

[7] Z. Pooranian, J. Abawajy *et al.*, “Scheduling distributed energy resource operation and daily power consumption for a smart building to optimize economic and environmental parameters,” *Energies*, vol. 11, no. 6, p. 1348, 2018.

[8] M. Rasheed, N. Javaid, A. Ahmad, Z. Khan, U. Qasim, and N. Alrajeh, “An efficient power scheduling scheme for residential load management in smart homes,” *Applied Sciences*, vol. 5, no. 4, pp. 1134–1163, 2015.

[9] F. Marcello, V. Pilloni, and D. Giusto, “Sensor-based early activity recognition inside buildings to support energy and comfort management systems,” *Energies*, vol. 12, no. 13, p. 2631, 2019.

[10] S. Merabti, B. Draoui, and F. Bounaama, “A review of control systems for energy and comfort management in buildings,” in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 478–486.

[11] P. H. Shaikh, N. B. M. Nor *et al.*, “A review on optimized control systems for building energy and comfort management of smart sustainable buildings,” *Renewable and Sustainable Energy Reviews*, vol. 34, pp. 409–429, 2014.

[12] ——, “Intelligent multi-objective optimization for building energy and comfort management,” *Journal of King Saud University-Engineering Sciences*, vol. 30, no. 2, pp. 195–204, 2018.

[13] M. Fayaz and D. Kim, “Energy consumption optimization and user comfort management in residential buildings using a bat algorithm and fuzzy logic,” *Energies*, vol. 11, no. 1, p. 161, 2018.

[14] P. Du and N. Lu, “Appliance commitment for household load scheduling,” in *PES T&D 2012*. IEEE, 2012, pp. 1–1.

[15] G. Bhat, R. Deb *et al.*, “Online human activity recognition using low-power wearable devices,” in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[16] M. M. Hassan, M. Z. Uddin *et al.*, “A robust human activity recognition system using smartphone sensors and deep learning,” *Future Generation Computer Systems*, vol. 81, pp. 307–313, 2018.

[17] Y. Liu, L. Nie, L. Liu, and D. S. Rosenblum, “From action to activity: sensor-based activity recognition,” *Neurocomputing*, vol. 181, pp. 108–115, 2016.

[18] N. C. Krishnan and D. J. Cook, “Activity recognition on streaming sensor data,” *Pervasive and mobile computing*, vol. 10, pp. 138–154, 2014.

[19] D. J. Cook, “Learning setting-generalized activity models for smart spaces,” *IEEE intelligent systems*, vol. 2010, no. 99, p. 1, 2010.

Prescriptive System for Reconfigurable Manufacturing Systems Considering Variable Demand and Production Rates

Catarina Baltazar, João Reis, Gil Gonçalves
SYSTEC, Research Center for Systems and Technologies
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
Email: {up201406435, jpcreis, gil}@fe.up.pt

***Abstract*—The current market is dynamic and, consequently, industries need to be able to meet unpredictable market changes in order to remain competitive. To address the change in paradigm, from mass production to mass customization, manufacturing flexibility is key. Moreover, current digitalization of the industry opens opportunities regarding real-time decision support systems allowing the companies to make strategic decisions, and gain competitive advantage and business value.**

The main contribution of this paper is a proof of concept Prescriptive System with a highly parameterizable simulation environment catered to meet the needs of Reconfigurable Manufacturing Systems allied with an optimization module that takes into consideration productivity, market demand and equipment degradation. With this system, the effects of different throughput rates are monitored which results in better recommendations to mitigate production losses due to maintenance actions while taking into consideration the health status of the remaining assets.

In the proposed solution the simulation module is modeled based on Directed Acyclic Graphs and the optimization module based on Genetic Algorithms.

The results were evaluated against two metrics, variation of pieces referred as differential and availability of the system. Analysis of the results show that productivity in all testing scenarios improves. Also, in some instances, availability slightly increases which shows promising indicators.

***Index Terms*—Reconfigurable Manufacturing Systems, Industry 4.0, Variable Throughput, Genetic Algorithm**

I. Introduction

Nowadays industries face constant changes as the result of unpredictable market trends. The challenge is to be flexible enough in order to respond in a timely manner to clients demand while maintaining a sustainable cost structure to remain competitive in a fierce business environment. For the purpose of attending markets needs, it is necessary to increase the efficiency of manufacturing processes in which machinery plays a fundamental role.

Reconfigurable Manufacturing Systems (RMS) arise to deal with uncertainty and individualized demand [1] by combining advantages of both Dedicated Manufacturing Lines and Flexible Manufacturing Systems [2]. Moreover, during the current industrial revolution, also referred as Industry 4.0, significant interest in the upgrade of Prognostics and Health Management (PHM) frameworks emerge as they allow improvements in reliability and reduction of costs associated with maintenance actions [3]. Advances in the Information and Communication Technologies domain enable the development of more sophisticated PHM tools, especially, based on Deep Learning methods as they simplify the process of feature learning and have superior performance. Deep Learning approaches represent a promising path towards a one-fits-all framework [4]. An effective PHM system should be able to timely predict failures by constantly monitoring health status of the equipment and also isolate and identify the faults [5]. Additionally, it must support decision-making systems to take full strategic advantage of the predictions provided by diagnosis and prognosis techniques [6]. While prognosis is related to failure prediction and tries to answer the questions ”What will happen?” and ”When will it happen?” [7], diagnosis consists in identifying and isolating the faults. Despite the intuitive relationship between predictions and prescriptions, and the undeniable benefits to gain competitive advantage, prescriptive systems' area is the field with less research [8]. These systems intend to recommend one or more courses of action based on predicted future and, therefore, allow to take proactive measures [7].

A thorough review of prescriptive systems is given by [8] where three categories were identified: production scheduling, life cycle optimization, supply chain management and logistics. For example, regarding inventory management, in both [9] and [10], spare parts are ordered based on equipment degradation. In the former, decisions regarding the purchase of spare parts are decided based on the levels of degradation observed during irregular inspections. In the latter, long short-term memory (LSTM) networks are employed to predict failure probability during different time windows. Then, based on the information provided by the prediction model, the appropriate options regarding maintenance and order of spare parts are chosen.

From the three categories identified, in an industrial context, maintenance scheduling is the more predominant one. In [11], a Genetic Algorithm (GA) is employed to optimize maintenance scheduling for manufacturing systems with a fixed structure. In this paper, it is assumed that the information regarding failure probabilities is available. Similarly, in [12], a GA is used to schedule maintenances based on machine degradation. However, in this case, the variables that are optimized are the throughputs of machines and possible maintenance actions instead of discrete time moments. In general, the proposed optimization procedure searches for the best trade-off between maintenance actions and throughput settings. Likewise, in [13] a continuous maintenance system based on real-time monitoring is proposed. The optimization module is also based on GA and assures production targets by searching the best sequence of machine throughputs taking into consideration equipment degradation. In contrast, in this paper, a Predictive Maintenance module is integrated and the GA helps in avoiding unexpected breakdowns based on constant condition monitoring in real-time. Solving scheduling problems is not limited to the application of GA but these algorithms represent the majority of the proposed solutions [14].

Few Prescriptive Systems are applied to RMS. In this context, the mitigation of production losses due to machines downtime can be achieved not only by tuning throughputs of different machines, but also by routing pieces to healthy assets. Accordingly, the main contributions of this paper are an optimization approach that shows good indicators in finding throughput sequences that balance productivity and maintenance actions in a RMS context, as well as a straightforward simulation module based on Directed Acyclic Graphs (DAG) that allows quick layout changes and easy parametrization of the shop-floor namely, scheduling of maintenance shifts, different types of failures and types of equipment.

The remainder of this paper is organized as follows. In section II both simulation module and optimization module are discussed. Then, in section III, the scenarios that are tested in order to validate the solution were presented. Additionally, some preliminary results are discussed. A more in depth analysis of the results presented in the previous section can be found in section IV and finally, in section V conclusions and future work are discussed.

II. Implementation

The proposed Prescriptive System is mainly composed of two modules: simulation module and optimization module. In the following subsections each module is further described and this current section concludes with the interactions between the two.

A. Simulation

The goal is to model manufacturing layouts such as the one presented in Fig. 1 so it allows easy changes in configurations in order to respond to different demands in the future. These configurations possess crossovers and all machines within the same stage execute the same tasks. Consequently, pieces in stage *i* can be transferred to any machine at the stage *i+1*. According to [15], these configurations are defined as Class II RMS.

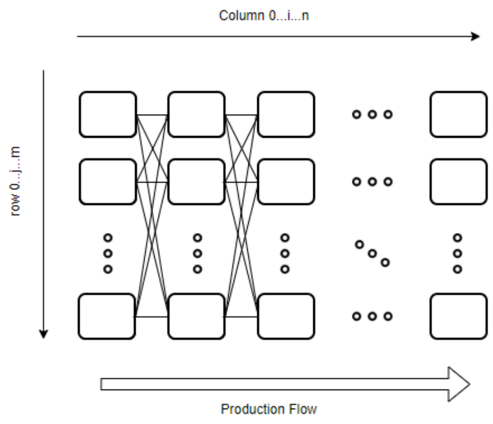


Fig. 1. Generic Manufacturing Layout

Accordingly, DAGs were chosen to model the system. This approach allows the rapid response in changing layouts configuration and the control of pieces flow in the manufacturing system. In order to implement it, the package Networkx, only available in Python, was chosen.

Each node of the graph represents a machine and the edges connections between machines that might be, for instance, conveyor belts. The edges are weighted and represent path priority. The lowest the weight the higher the priority. This approach allows to favour, for example, the shortest path when deciding to which machine should the piece be sent.

The machines are represented by the class Machine and each instance represents a node of the graph. This approach allows high parameterization of the equipment and the parameters can be separated in three main groups:

- Identifying Parameters: relate to the identification of the equipment
  - machine id;
  - type of machine;
  - age;
  - line;
  - stage;
- Operations Parameters: relate to the machine operation
  - available operations;
  - current throughput;
- Reliability-related parameters: relate to degradation of the equipment
  - mean time to repair (MTTR);
  - mean time between failures (MTBF);
  - types of failures.

Concerning to identifying parameters, line and stage correspond to the position of the machine in the layout, Fig. 1, while the remaining parameters in this category are related to specifications of the equipment. In respect of operation parameters, available operations relate to the range of operations that the machine can perform and current throughput identifies the production rate at which the equipment is operating. Lastly, regarding reliability-related parameters, this are of the utmost importance to simulate the degradation of the equipment. In terms of different types of failures, each machine can have associated different ones which will correspond to different MTTR and, as a result, maintenance actions will have different periods of time. Also, MTBF will be used as a mean to predict the failure.

In addition, in this case, the machines are also responsible to control the flow of production in the shop-floor. Each machine has a state machine associated as the one represented in Fig. 2.

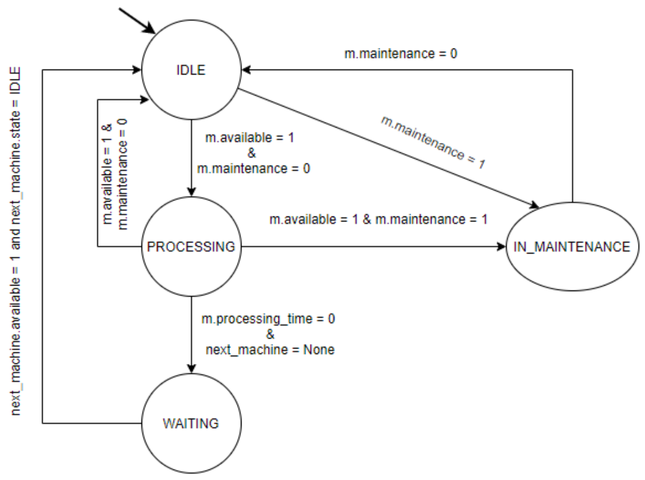


Fig. 2. State Machine associated with each machine

The machine has four states. It starts in its IDLE state and if the machine is not going to start any maintenance, maintenance = 0, and is available, the machine can receive pieces. Once the pieces are received they are processed. When the processing time ends, three things might happen: if the next machine is available the piece is dispatched and then the machine can return to its IDLE state or IN_MAINTENANCE state. Otherwise, it will transition to WAITING state. This transition happens when there are no available machines and the current machine behaves as a buffer until a possible machine becomes available. While in the WAITING state, the machine cannot receive any pieces. In the case that the machine does not have the respective tool, the piece experiences the same cycle, however, processing times are equal to zero. In short, the edges of the graph provides the different connections between machines and each connection is only admissible if green-lighted by the destination machine state.

In addition, not only machines can be parameterized but also other parts of the manufacturing environment. The simulation module developed in this paper takes into consideration, different simulation times, maintenance shifts and different sequences of operations to apply to different raw materials. Simulation times are related to how many seconds each tick (time unit in the simulation environment) worth and how many working weeks are being simulated. Also, it defines how many working days and working hours are considered. In regards to maintenance shifts, if one decides to integrate them in the simulation, the starting times and duration of said shifts can be defined. The only thing, which in some cases might be considered a limitation, is the fact that the maintenance shifts, by default, are periodic. Simply put, in every working day the shift starts at the same time and has the same duration. Additionally, different sequences of operations can be applied to the pieces in order to achieve different final products as long as the needed operations are available in the current machines and as long as the operations can be performed in a sequential manner as represented in Fig. 1. All these features allows the simulation of a wide variety of scenarios not only on time domain but also specification wise.

In this paper, it is assumed that the information regarding probability failures is known, as no predictive model is proposed. Recalling the parameters associated to each machine, namely, reliability-related ones, both MTBF and MTTR are known. In a simplified manner, MTTR refers to the average time to repair certain component and MTBF the forecasted time between failures [16]. Both these terms will allow to simulate degradation of the equipment as well as management of maintenance actions in order to implement the present system. As a result, the prediction of a pending failure will be calculated based on the difference between MTBF and current simulation time. If that difference is below a certain threshold, the failure will be signaled and maintenance scheduling takes place. Both Fig. 3 and Fig. 4 exemplify how the maintenance scheduling is handled. The difference between MTBF and current simulation time corresponds to a certain time window. This time window is the time to failure and is represented by the yellow area. If during that time window a shift takes place, blue area, then the maintenance of the respective equipment will occur when the shift starts (Fig. 4). Otherwise, an emergency maintenance is triggered (Fig. 3).

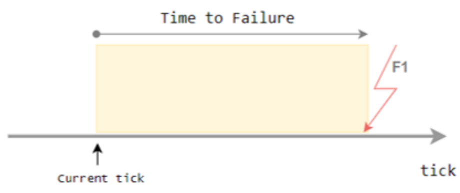


Fig. 3. Pending Failure that will result into an emergency maintenance

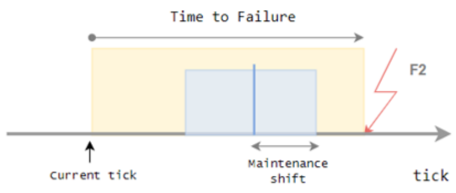


Fig. 4. Pending Failure that will result into a scheduled maintenance

Furthermore, different machines' throughputs have different impacts in degradation of the equipment. As stated in [12], when a machine decelerates it is expected that its degradation slowdown, and vice-versa if a machine increases its throughput. To simulate the degradation effects influenced by the chosen production rates, the MTBF will be inversely proportional to production rate. Similar to [13], five throughputs are available where mode 2 increases production rate two times in regards to baseline production, mode 1 production rate is 1.5 times higher, mode 0 corresponds to baseline throughput, mode -1 production rate decreases in 1.5 times and, lastly, mode -2 where production rate decreases 2 times.

B. Optimization

The optimization module is key to the implementation of the Prescriptive System as it is responsible for the compensation of production losses due to machines' downtime. A standard GA approach was chosen as its employment is well documented and produces near-optimal solutions [17]. GAs can be understood as an abstraction of the theory of evolution by natural selection by Darwin and are suitable to solve multi-objective problems [18]. The genetic variability within a population is simulated through mutation and crossover operators and the selection is done based on the survival of the fittest [19].

The optimization module can be triggered in two instances: when an emergency maintenance takes place, or when a maintenance does not finish during a maintenance shift. As a result, two types of maintenance can be identified:

- **Emergency Maintenance** - a maintenance that occurs outside a maintenance shift;
- **Scheduled Maintenance** - a maintenance that is allocated to a maintenance shift.

Emergency maintenances are more costly not only because of resources allocation, but also their impact in production. Even if a scheduled maintenance continues beyond the shift duration, the losses in production are lower because the downtime during maintenance shift is expected, which does not happen in a context of an emergency maintenance. When formulating the optimization problem, both types of maintenance are taken into consideration with different weights, as their impact is also different on production weekly goals.

The used approach follows very closely the one presented in [13]. The proposed formulation was applied to three parallel machines and can easily be applied to N parallel machines. However, other configurations require some fine-tuning in their weights and the addition of some terms depending on the problem. In summary, the goal is to extend the mentioned formulation to a more broad spectrum of layouts and adapt it to the RMS system considered in this Prescriptive System.

Every week, the production should comply with the customers orders so the GA optimizes a maximum of one week and once the current week ends, the throughputs return to their baseline unless new optimization takes place in that week and the process repeats itself once again. In this regard, each gene of the chromosome will represent the throughput of machine *i* at the day *j* as represented in Fig. 5.

$T_{1,1}$	$T_{1,2}$	...	$T_{1,j}$	$T_{2,1}$	$T_{2,2}$	...	$T_{2,j}$	...	$T_{i,j}$

Fig. 5. Chromosome Structure. Source: [13]

$T_{i,j}$ is an integer between -2 and 2 and corresponds to the machine i operation mode at the day *j*. Thus, the size of the chromosome is variable and equal to $i \times j$.

Companies' main goal is to attend customer's needs while remaining competitive and profitable. Therefore, it is crucial to meet production targets in the most efficient way. Accordingly, the fitness function (1) not only takes into consideration production targets but also machines' degradation.

$$F = min\Bigg[K_p\left(W-P\right)^2 + K_{sm}\sum_{i}^{N}F_{sm_i} + K_{em}\sum_{i}^{N}F_{em_i} + K_{nw}\sum_{i}^{N}F_{nw_i} + K_{ch}\sum_{i}^{N}C_{ch_i} + K_{sd}\sum_{i}^{N}S_i\Bigg] \tag{1}$$

$$\text{subject to:} \quad F_{sm_i}, F_{em_i}, F_{nw_i} = \{0,1,...,N\} \quad \forall_i$$
$$C_{ch_i} = \{0,1,...,d\} \quad \forall_i$$
$$S_i \geq 0 \quad \forall_i$$

The first term is the difference between production weekly target, W, and number of pieces produced, P, by the system, squared. In essence, it evaluates how far the system production is from the target and the square ensures that the algorithm does not favour solutions that exceedingly surpass the target, and the non-negativity of the values. The following three terms are regarding the different maintenances. Each type of maintenance is different and, as a result, also their weight in the fitness function. The second and third term is scheduled maintenance, sm, and emergency maintenance, em, respectively, and their different impacts were already stated. Throughout the formulation of the fitness function, initially there was no distinction between those two maintenances and the results were good so if a more broad approach is desired the maintenance might not be distinguished. However, the prescriptive system proposed has scheduled maintenance shifts integrated and the distinction between the two makes sense since they have different impacts in the production system. The fourth term is also related to maintenance, but it is regarding the first three days of the next week, nw. To increase production the throughput of some machines has to inevitably increase, which accelerates the degradation of those machines. So, this term is to prevent new failures in the beginning of the next week as it will affect the production goals of the next week.

The constant change of throughputs in a real production line is not practical. As a result, the last two terms are introduced to promote homogeneous solutions. The first term of the two, ch, corresponds to the number of changes in relation to the baseline, mode 0, and the second is the standard deviation, S, of the suggested throughputs to machine *i*.

Initially, the weights considered were the same as the ones presented in [13]. After several simulations, it was observed that the convergence of the solutions was not quite as desired. At the boundary of solutions that achieve the weekly targets and solutions with deficits, sometimes close to 2%, but with throughput rates more homogeneous, the latter were given priority (i.e., better fitness values). This behaviour was further proved by the conduction of a sensitivity analysis where the contributions from the different types of maintenance were considered constant and the remaining terms of the Equation (1) variable. As a term of comparison, margins of 1% in relation to production in regards to the desired targets were considered acceptable. So, the weights needed to be refined. Accordingly, based on the previous sensitivity analysis and additional simulations, the finals weights are as follows: $K_{\text{p}} = 10, K_{\text{sm}} = 900, K_{\text{em}} = 1000, K_{\text{nw}} = 300, K_{\text{ch}} = 300$ and $K_{\text{sd}} = 400$.

C. Prescriptive System

The proposed Prescriptive System involves the two modules explained above and an overview can be found in Fig. 6. Once a failure is detected and if the requirements regarding the conditions in which the maintenance will occur are met, the optimization module is triggered. As shown in Fig. 6, represented by blue rectangles, two instances of the simulation module are present: Manufacturing Environment Simulation and Simulation Module. The former corresponds to the simulation of the shop-floor of interest and the latter is an image of the former. However, in this case, its purpose is solely to feed the optimization module with the needed variables to evaluate the candidate solutions: pieces produced and number of maintenances during current week and the following one. These outputs are what allows the calculation of the solution fitness value represented by Equation (1). Additionally, in both these modules, a model to predict failures can be easily integrated. This cycle between optimization module and simulation model stops once the termination criteria is met. In this paper, the optimization stops when the maximum number of generations is exceeded. When the optimization module finishes, the best solution is recommended (white rectangle) and applied to the manufacturing environment simulation if the operator decides to.

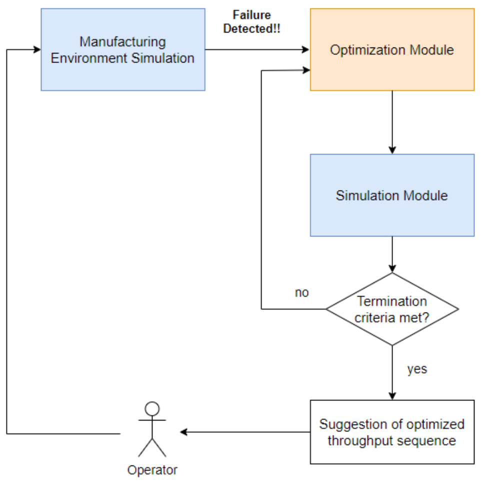

Manufacturing Environment Simulation
Failure Detected!!
Optimization Module
Simulation Module
no
Termination criteria met?
yes
Suggestion of optimized throughput sequence
Operator

Fig. 6. Overview of the proposed Prescriptive System

III. System Validation and Verification

To evaluate the proposed system the testing was divided into two phases. Firstly, a set of tests are applied in order to analyze and validate the results provided by the GA as well as to prove that this system might be easily applied to configurations not fully connected or easily upgraded to handle failure in transport equipment. Secondly, scenarios that are more complex are investigated in order to check scalability. The simulation time in all tests is one working week. Also, there will be two shift changes per working day, where maintenance actions can be performed. One in the beginning of the day and other in the middle. The metrics used to assess the performance of the system are the variation of pieces produced in relation to target, named as differential, and an extension of availability per machine [20] to the whole system defined by the ratio of total real operation time of all machines by the total theoretical operation time of all machines. Taking into consideration Fig. 1, the configurations will be referred as *nxm*, where *n* corresponds to the amount of stages and *m* the amount of production lines. The GA parameters were selected after several runs and set to:

- Population size = 100;
- Maximum generations = 100;
- Mutation Rate = 0.2;
- Crossover Rate = 1.0;
- Crossover Method: Single-point crossover;
- Selection Method: Elitism.

All tests were performed in a personal computer with the specifications: Intel core i5-3750 CPU @ 3.40GHz and 8.00 GB RAM.

A. First Set Scenarios

All tests were performed using a 3x2 configuration. In the first test, one of the machines is down a whole working day and another machine is on the verge of failing in the following week. In the second one, the same machine is down, however there is a second machine that fails in the middle of the week, during half-day. In the third and last test of this set, there are no broken machines but the connections from one of the machines are interrupted which isolates the equipment and, consequently, pieces processed by it have nowhere to flow to. The main goal of all tests is to understand, under different conditions, if the weekly target is achieved and how the algorithm deals with the different maintenance moments. However, the Test 3 is performed not only as a mean to study the previous statements but also as a tool to prove that this system might be applied to layouts different than the one presented in Fig. 1 where all stages are fully-connected. It may be applied, for example, to layouts where the stages have different number of machines. Also, it demonstrates that failures related to transportation equipment can be considered as long as the failure predictions are fed to the algorithm in order to trigger the optimization module.

In Table I, the effects of maintenances and connection interruptions during normal operation without optimization module are represented and summarized. Expected Production is the number of pieces produced by the system if no disturbances in the system occur. Pieces produced are the pieces that system manufactured under the conditions explained previously for each test without the intervention of the Prescriptive System. Also, differential and availability are the metrics previously explained taking into consideration that no optimization took place.

TABLE I
EFFECTS OF FAILURES IN THE SYSTEM WITHOUT OPTIMIZATION MODULE

	Expected Production	Pieces Produced	Differential	Availability
Test 1	796	731	-8,16%	96,3%
Test 2	796	698	-12,31%	94,4%
Test 3	796	607	-23,74%	92,0%

B. First Set Results

Each test was executed three times. In Table II, the averages of these three runs are presented, together with standard deviation, σ, of differentials.

TABLE II
RESULTS OF FIRST TESTING SET WITH OPTIMIZATION MODULE

	Pieces Produced	Differential	σ	Availability	Processing Times
Test 1	796	-0,044%	0,259%	96,3%	4,27h
Test 2	795	-0,084%	0,258%	94,4%	7,9h
Test 3	796	0%	0,000%	92,0%	2,87h

Recalling the conditions the test 1 was under, one of the possible outcomes could be the advancement of the failure that was scheduled to the beginning of the following week. However, this did not happen. In the second test, two optimization moments occurred, one per each failure. This is further supported by the fact that in both cases the availability did not change, which means that the downtime neither increased or decreased. In the third test, it is confirmed that the system can handle other types of situations and/or layouts. In this case, both differential and standard deviation are 0% because in all three runs the weekly target was scrupulously achieved.

C. Second Set Scenarios

Previous tests showed that the system behaves as expected so scenarios that are more complex were tested in order to investigate the scalability of the system. For each configurations tested, two types of situations were considered:

- Type A - weekly production target equal to expected production;
- Type B - weekly production target 1,2 times higher than expected production.

The purpose of type B tests is to explore situations where market demand increases and verify if the manufacturing system can still comply in those situations. Four different configurations were tested and Table III summarizes all the scenarios as well the effects of number of maintenances in the system without the optimization module. It was decided to increase the number of maintenances as the configurations increase in size in order to test similar levels of stress. This increase, in Table III, is referred as ”Number of maintenances”. Expected Production and Pieces Produced, as well as, differential and availability, have the same meaning as the presented in Table I. Each configuration has two different targets as they correspond two each type as stated before.

TABLE III
SCENARIO DEFINITION OF THE SECOND TESTING SET

Config.	Number of maintenances	Expected Production	Pieces Produced (Differential)	Availability	Target	Test name
3x3	1	1194	1113 (-6,78%)	97,5%	1194	Test1a
				97,5%	1433	Test1b
4x4	2	1532	1412 (-7,83%)	97,9%	1532	Test2a
				97,9%	1838	Test2b
7x7	5	1554	1490 (-4,12%)	97,5%	1554	Test3a
				97,5%	1865	Test3b
10x10	8	2030	1954 (-3,14%)	98,3%	2030	Test4a
				98,3%	2436	Test4b

D. Second Set Results

In all tests the target was achieved within 1% margin and, in some cases, the availability slightly increased. Those cases are marked in bold in Tables IV and V. In these instances, the increase in availability was because the algorithm “pushed” some failures to next week as a result of a reduction in the throughputs of the respective machines. In addition, this happened in higher order configurations, which indicates that is likely due to the higher redundancy in these systems.

TABLE IV
RESULTS FOR TESTS TYPE A

	Pieces Produced	Differential	σ	Availability	Processing Times
Test1a	1193	0%	0,181%	97,5%	3,0h
Test2a	1533	0,13%	0,134%	97,9%	8,7h
Test3a	1554	0%	0,273%	**98,0%**	30,9h
Test4a	2024	-0,279%	0,203%	**98,7%**	71,3h

TABLE V
RESULTS FOR TESTS TYPE B

	Pieces Produced	Differential	σ	Availability	Processing Times
Test1b	1434	0,07%	0,057%	97,5%	3,1h
Test2b	1838	0,108%	0,112%	97,9%	9,7h
Test3b	1864	-0,018%	0,241%	**97,8%**	29,5h
Test4b	2438	0,096%	0,102%	**98,5%**	77,3h

Still, in respect to the increase in availability, the comparison between Fig. 7 with Fig. 8 gives an insight of how the algorithm dealt with the different maintenance actions. These figures are related to Run 1 of test4b and its results can be found in Table VI. In Fig. 8, maintenance regarding machines J5 and G7 disappeared from the current week and the throughputs in those machines are, in general, lower than baseline. This is consistent with Equation (1) as maintenances in next week, F_{nw} are less penalizing than current week and the algorithm found a way of decreasing the fitness value by pushing the maintenance to next week without jeopardizing the achievement of the weekly target. In addition, considering once more Fig. 8, maintenance regarding machine G8 was advanced in relation to Fig. 7 however, this advancement translated into a scheduled maintenance instead of an emergency maintenance which is also consistent with the fitness function as emergency maintenances, F_{em}, are more penalizing than scheduled maintenances, F_{sm}.

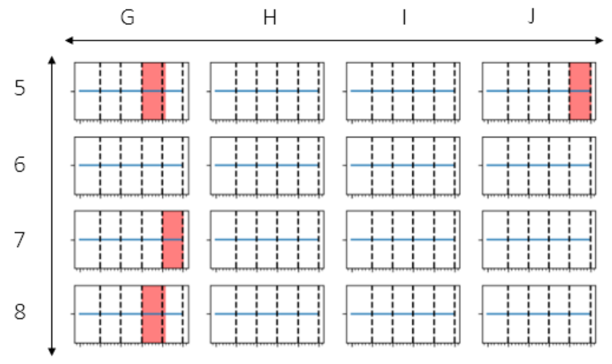


Fig. 7. Part of layout of configuration 10x10. Simulation correspondent to Run1 of test4b where no optimization took place. The red vertical bands represent the time that a machine is under maintenance and the blue horizontal lines are the throughput rates in place during certain day.

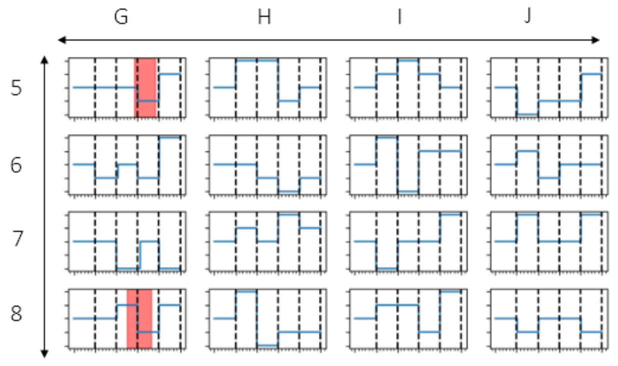


Fig. 8. Part of layout of configuration 10x10. Simulation correspondent to Run1 of test4b where the measures recommended by the Prescriptive System were adopted.The red vertical bands represent the time that a machine is under maintenance and the blue horizontal lines are the throughput rates in place during certain day.

TABLE VI
RESULTS OF RUN1 OF TEST4B

Target	Pieces Produced	Differential	Availability
2436	2441 (+5)	0,205 %	98,8 %

To evaluate how the results vary from configuration to configuration in order to draw some conclusions, the averages of the differential were plotted and the graphs are presented in Fig. 9 and Fig. 10, tests type A and tests type B, respectively.

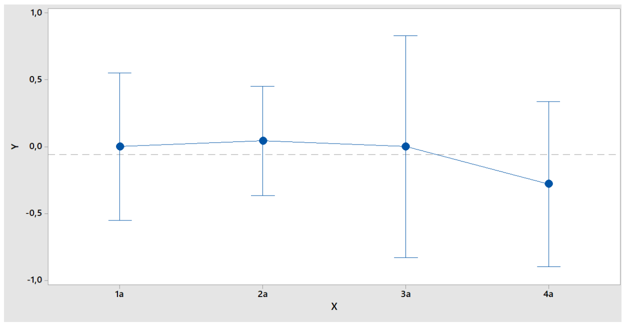


Fig. 9. Differential Averages per Configuration in tests type A

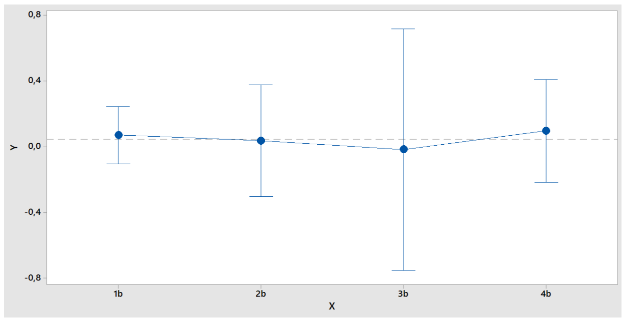


Fig. 10. Differential Averages per Configuration in tests type B

IV. Discussion

The results show large improvements in the pieces differential and, in some instances, a slight increase in availability. Despite the decrease in differential, in some instances, the target value was not fully met, presenting low deficits ($<1\%$), but always by far better than the results without optimization.

The parameters of the GA are problem dependent. In the GA implementation employed in this system, both generations and population size are fixed. However the size of each chromosome is not. Remembering previous sections, the chromosome size is equal to $N \times d$ where N is the total number of machines and d, the days from the point the optimizer was triggered until the end of the week. So, not only between different configurations but also within configurations, the chromosome size varies but the parameters are not recalculated. This could led to believe that the algorithm when applied to bigger configurations would generate worse solutions.

When comparing the averages of each configuration, the desired results are that they gravitate towards zero with low deviations. The solutions seem to follow this behaviour, however, there is a visible increase in deviation from configuration 3 to configuration 4, Fig. 9, in tests of type A but it did not go beyond 1%. In fact, this corresponds to an average deviation of 0,279% as can be observed in Table IV. Therefore, this increase does not seem enough to jeopardize the results regarding the tested configurations, and can be attributed to the search strategy and convergence of the GA. However, more testing should be conducted. Despite the increase in complexity of the system, the GA model was always able to find solutions with 1% margin. As a matter of fact, the biggest differential was a deficit of 0,542% that occurred during Run 1 of Test4a. Additionally, note that the Test 4 refers to a configuration 10×10 meaning that 100 machines are operating which is already a considerable amount of equipment.

V. Conclusion

A Prescriptive System capable of adapting machines' throughput depending on variable demand and taking into consideration pending machine failures was presented. The factory is modelled based on graphs theory which allows a quick response in layout changes and the throughput sequences are managed by a simulation-based GA. The proposed system was evaluated to different layouts and showed consistent results among them – the Differential decreased and had a positive influence in the availability of the system as previously stated.

There is no denying that Prescriptive Systems rely heavily on each company's goals and specifications, which leads to one of the main reasons for the lack of prescriptive systems in the current literature. In this respect, this paper tries to tackle this gap by implementing both manufacturing simulation environment and optimization module in a considerably generic manner. Thus, regarding the optimization module, despite the objectives being already established they were chosen in order to be suitable to any manufacturing industry.

The proposed system was developed with a future integration with a Predictive Maintenance Module in mind and that would be one of the immediate improvements that could be done to this system in order to offer a whole cohesive framework that assists in the process of making decisions based on constant monitorization of the machines health status. Also, further research should be conducted not only by increasing the number of runs per tests but also explore how the system performs with real-data. In addition, other limitation that needs to be addressed are the long processing times needed which are a huge restriction when applied to real-life scenarios. In this case, the exploration of distributed or parallel GA approaches can help to overcame this constraint.

Acknowledgment

This paper is integrated in the project INDTECH 4.0 – New technologies for intelligent manufacturing. Support on behalf of IS for Technological Research and Development (SI à Investigação e Desenvolvimento Tecnológico). POCI-01-0247-FEDER-026653.

References

[1] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: The state of the art," Int. J. Prod. Res., vol. 46, no. 4, pp. 967–992, 2008.

[2] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," Front. Mech. Eng., vol. 13, no. 2, pp. 121–136, 2018.

[3] G. W. Vogl, B. A. Weiss, and M. Helu, "A review of diagnostic and prognostic capabilities and best practices for manufacturing," J. Intell. Manuf., vol. 30, no. 1, pp. 79–95, 2019.

[4] L. Zhang, J. Lin, B. Liu, Z. Zhang, X. Yan, and M. Wei, "A Review on Deep Learning Applications in Prognostics and Health Management," IEEE Access, vol. 7, pp. 162415–162438, 2019.

[5] G. Xu et al., "Data-driven fault diagnostics and prognostics for predictive maintenance: A brief overview," IEEE Int. Conf. Autom. Sci. Eng., vol. 2019-Augus, no. 1, pp. 103–108, 2019.

[6] F. Ansari, R. Glawar, and T. Nemeth, "PriMa: a prescriptive maintenance model for cyber-physical production systems," Int. J. Comput. Integr. Manuf., vol. 32, no. 4–5, pp. 482–503, 2019.

[7] K. Lepenioti, A. Bousdekis, D. Apostolou, and G. Mentzas, "Prescriptive analytics: Literature review and research challenges," Int. J. Inf. Manage., vol. 50, no. April 2019, pp. 57–70, 2020.

[8] A. Diez-Olivan, J. Del Ser, D. Galar, and B. Sierra, "Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0," Inf. Fusion, vol. 50, pp. 92–111, 2019.

[9] F. Zhao, X. Liu, R. Peng, and J. Kang, "Joint optimization of inspection and spare ordering policy with multi-level defect information," Comput. Ind. Eng., vol. 139, no. 3, p. 106205, 2020.

[10] K. T. P. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics," Reliab. Eng. Syst. Saf., vol. 188, pp. 251–262, 2019.

[11] Z. (Max) Yang, D. Djurdjanovic, and J. Ni, "Maintenance scheduling in manufacturing systems based on predicted machine degradation," vol. 19, no. 1, pp. 87–98, 2008.

[12] Z. Yang, D. Djurdjanovic, and J. Ni, "Maintenance scheduling for a manufacturing system of machines with adjustable throughput," IIE Trans. (Institute Ind. Eng., vol. 39, no. 12, pp. 1111–1125, 2007.

[13] L. Antao, J. Reis, and G. Goncalves, "Continuous Maintenance System for Optimal Scheduling Based on Real-Time Machine Monitoring," in IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2018, vol. 2018-Septe, pp. 410–417.

[14] B. Çaliş and S. Bulkan, "A research survey: review of AI solution strategies of job shop scheduling problem," J. Intell. Manuf., vol. 26, no. 5, pp. 961–973, 2015.

[15] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," J. Manuf. Syst., vol. 29, no. 4, pp. 130–141, 2010.

[16] S. Aguiar, R. Pinto, and G. Gonc, "Life-cycle Approach to Extend Equipment Re-use in Flexible Manufacturing," INTELLI 2016 Fifth Int. Conf. Intell. Syst. Appl. (includes InManEnt 2016) Life-cycle, no. November, pp. 148–153, 2016.

[17] C. Renzi, F. Leali, M. Cavazzuti, and A. O. Andrisano, "A review on artificial intelligence applications to the optimal design of dedicated and reconfigurable manufacturing systems," Int. J. Adv. Manuf. Technol., vol. 72, no. 1–4, pp. 403–418, 2014.

[18] S. N. Mirabedini and H. Iranmanesh, "A scheduling model for serial jobs on parallel machines with different preventive maintenance (PM)," Int. J. Adv. Manuf. Technol., vol. 70, no. 9–12, pp. 1579–1589, 2014.

[19] K. Jebari and M. Madiafi, "Selection Methods for Genetic Algorithms," Int. J. Emerg. Sci., vol. 3, no. 4, pp. 333–344, 2013.

[20] S. H. Huang et al., "Manufacturing productivity improvement using effectiveness metrics and simulation analysis," Int. J. Prod. Res., vol. 41, no. 3, pp. 513–527, 2003.

A Cross-Platform Communication Mechanism for ROS-Based Cyber-Physical System

Rui Zhao, Xu Tao, Davide Conzon, Enrico Ferrera
LINKS Foundation
via Pier Carlo Boggio 61,
Turin,
Italy
name.surname@linksfoundation.com

Yenchia Yu
Tongji University
4800 Cao'an Road,
Shanghai,
China
yuyenchia@tongji.edu.cn

***Abstract*—Recently, one of the main research topics in the context of application of Cyber-Physical System (CPS) in the Smart City and Industry 4.0 scenarios is the one related to the use of Robot Operating System (ROS)-based CPS. Specifically, one of the main interest is to allow a ROS-based smart robot communicating with other heterogeneous Internet of Things (IoT) applications in an intelligent environment to efficiently react to the system requirements and environment changes. However, the communication between the IoT systems will face many challenges and increase the cost and risks that lead to the requirement of a cross-platform communication for bridging the ROS-based CPS and other heterogeneous IoT applications.**

This paper introduces ROS Edge Node for the interoperability between Robotics domain and other IoT domains, leveraging the highly modular BRAIN-IoT federation, which allows to decentralize, compose and dynamically federate the heterogeneous IoT platforms using OSGi specification, thanks to its dynamic modularity and wide usage in IoT middlewares. Together with the flexible integration with existing IoT devices/platforms within BRAIN-IoT platform, the event-driven asynchronous communication mechanism realizes cross-platform interaction with ROS-based CPS and solves the major challenges faced. This communication mechanism allows dynamic deployment of new functionalities for enhancing/extending the behaviour of robots according to external events. In addition, some specific behaviours to new "virgin" robots, which might be needed to extend the fleet of robots or replace damaged/low batteries ones can be dynamically deployed at the setup phase. In BRAIN-IoT platform, Edge Node behaves as IoT devices/platform adaptors which integrate the existing IoT devices/platforms. The ROS Edge Node is one type of the Edge Node, which bridges the underlying ROS-based robotics systems and BRAIN-IoT execution environment, thus communicates with various IoT systems connected to the BRAIN-IoT platform. A Service Robotic use case is developed to demonstrate the proposed solution, it shows how the ROS Edge Node enables the fast adaptivity and interoperability between heterogeneous IoT domains in a federated environment.

***Index Terms*—Brain-IoT, Cyber-Physical System (CPS), Service Robotics, IoT Middleware, Cross-platform Communication, OSGi**

I. Introduction

Nowadays, CPS are widely used in various aspects in our society, including but not limiting in areas such as manufacturing, energy, health, transportation and intelligent buildings, causing significant socio-economic impacts. CPS are defined as physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core. To realize such an intelligent system, various technologies need to be involved, including sensor and actuator technology etc. Together with IoT, Cloud computing and Cognitive computing, they become the main technologies of Industry 4.0 [1]. Specially, CPS and IoT are often discussed together, not only because they have many similarities but also because they are the foundations of the intelligent production environment, in another word, the smart factories, which is one of the most important topics of Industry 4.0.

The communication is an important research topic for both of CPS and IoT. As more and more different devices will be integrated in an intelligent production environment, the requirement of communication between different CPSs or between CPS and IoT devices are becoming more and more significant. For example, in a smart factory, different CPSs such as Autonomous Mobile Robots (AMR) [2] need to communicate with each other for cooperation, or an AMR needs to communicate with IoT devices such as automatic doors when it needs to cross different zones in the factory. However, such communication is not that easy to realize since heterogeneity is a basic property for both CPS and IoT devices.

On the one hand, heterogeneity exists because different CPS and IoT devices use different technologies in hardware, software, or communication method due to different needs. On the other hand, it is because the manufacturers of the devices are different. Different manufacturers will design the device according to their own standards. Devices from different manufacturers, even if they are using the same technologies, their protocol will be different. According to the latest statistics [3], there are officially 620 IoT platform companies in the global open market in 2019, in which 50% of the platforms focus on industrial use. From the perspective of communication, devices from these companies could use hundreds of different communication protocols which makes the standardization of communication protocol extremely difficult. In the recent years of research, the idea of using middleware to realize cross-platform communication is proposed.

This paper presents a novel adaptor, ROS Edge Node, which implements an event-driven asynchronous cross-platform communication mechanism for ROS-based CPS. The work is a part of the H2020 research project BRAIN-IoT [4] that is a federation enabling the dynamic deployment, orchestration and monitoring of the distributed IoT applications leveraging the OSGi [5] technology, since OSGi is a series of specifications for Java dynamic modular system, it provides the dynamicity for the life cycle management of software components. One of the main functions is to make components as decoupled as possible, and to allow components to dynamically discover with each other, so that programmers can develop refinable, reusable, and assistive components in accordance with these specifications. Nowadays, OSGi is the widest used technology to support implementations of IoT abstraction layers. For example, Bosch ProSyst and Eurotech are two examples of companies providing OSGi-based IoT gateways; Oracle® Fusion Middleware [6] is developed for developing Java Enterprise Edition (EE) management applications for Oracle WebLogic Server; SNPS [7] is an OSGi-based middleware for Wireless Sensor Networks; The OSGi-based software platform Eclipse SensiNact [8] provides support in technical aspects (e.g. Connectivity, Interoperability, Data processing, and Developer Tool) related to smart city platforms. BRAIN-IoT platform is implemented with OSGi and it aims integrating with generic existing IoT devices or IoT platforms or existing IoT middlewares to allow them communicating with each other. ROS Edge Node is one implementation of the BRAIN-IoT Edge Node mainly focused on the integration of robotics platforms in IoT domain allowing to interoperate with other heterogeneous IoT devices by mapping the ROS services to OSGi services, to maximize the connectivity of robots within IoT systems. ROS Edge Node is a modular software component of the BRAIN-Io platform. It provides four main features: 1) *Interoperability*: it provides the connectivity to IoT platforms connected to BRAIN-IoT solution. 2) *Plug & Play*: leveraging the OSGi specification, the component follows an event-driven approach and it is developed as a software module that can be deployed/undeployed at runtime without interrupting other running services. 3) Automatic Adaptation, it provides a code generator to automatically expose the ad-hoc ROS services provided by different ROS-based CPSs and speeds up the adaptor development process. 4) Standards Compliant, it exploits the Web of Things (WoT) Thing Description (TD) [9] describing the services provided by the ROS-based CPS, making it more portable to the production environment, not restrict to the OSGi implementation.

The rest of this paper is organized as follows. Section II introduces the state of the art of the solutions similar to the one proposed in this paper in multiple domains and the challenges to be addressed typically by these software. It also outlines the relevant technologies used to develop the proposed solution. Section III describes the architecture of the ROS Edge Node, functionalities and development process. Section IV implements a simple robotic application to validate the solution in a distributed environment. Finally, the paper concludes with the Section V to provide a summary of the cross-platform communication mechanism for ROS-based CPS and the future work being undertaken.

II. Background

A. State of the Art

One of the most popular example of CPSs is represented by smart industrial robots. This sector is really fragmented, in terms of hardware and software architectures, But in recent years, more and more robots have begun adopting one common technology, the middleware ROS. According to the annual ROS Metrics Report for 2019 [10], there are near 150 types of documented robots available to the community with ROS drivers, and in recent years, the total number of papers citing “ROS: an opensource Robot Operating System” (Quigley et al.,2009) increase at an annual rate of 20% to 30%. More and more famous robot vendors such as Asea Brown Boveri (ABB) Ltd., Comau Spa. and Kuka AG are starting to support ROS in some models of their robots. ROS is the widest used abstraction layer for robotics. However, even the robots are using ROS middleware, the problems in communication still exists. ROS-based CPS are often used to communicate with other heterogeneous IoT applications to satisfy system requirements and react physical environment changes. There are two existing middlewares for interacting with ROS-based CPS.

The ROS–YARP Framework for Middleware Interoperability [11]: ROS and Yet Another Robot Platform (YARP) [12] are the most popular robotics middlewares. YARP is more used in the domain of humanoid robots and developmental robotics, whereas ROS has higher focus on mobile robots. They have complementary functions and many robotic platforms may benefit from using functions from both. But it’s not an easy task mainly due to fundamental differences in the communication architecture. This approach generates the “bridging gap” code from a configuration file, connecting YARP ports and ROS topics through code-generated YARP bottles. It supports YARP/ROS and viceversa sender/receiver configurations. Reading from/sending to ROS topics needs an additional conversion, which is handled by the existing runtime YARP to ROS converter. The generator abstracts YARP and ROS developers from dealing directly with interoperability issues. However, this work is only focusing the interoperability among robotics middlewares.

ROBIN Middleware for CPS [13]: ROBIN is a middleware funded by European H2020 program providing an effective, bidirectional, reliable and structured data interchange mechanism to address the demand for flexible robotics in contemporary industrial environments and the necessity to integrate robots and automation equipment in an efficient manner. ROBIN, the robotics bridge to industrial automation, aims to allow the interoperability between robotics and automation systems by enabling the communication between ROS and CODESYS [14], which is a softPLC[1], a real-time multi-task control kernel, that can run on embedded devices

[1]http://www.softplc.com/

and that supports a variety of fieldbuses, and other industrial network protocols. CODESYS handles the establishment of external communication with the equipment via a fieldbus and communicates with ROS through the developed robotics bridge. More specifically, since ROS implementation follows the publisher–subscriber messaging pattern, which enables exchanging data between ROS nodes [15], A ROS node is basically a process that performs computation and it is an executable program running inside the robotics application. Many nodes can be implemented in ROS packages. The proposed bridging mechanism allows developers to create or include in existing ROS packages the valuable feature of connecting with an external device via fieldbuses or industrial network protocols promoted by the softPLC bridge. Multiple ROS nodes can access and modify the data shared with the softPLC application in ROS. The information to be propagated to the external devices could be published on a ROS topic, handled by the developed bridge, and then relayed by CODESYS to the proper industrial network protocol or fieldbus.

B. Problems with State of the Art

ROS allows the communication between heterogeneous devices, being deployable on heterogeneous platforms. After ROS is deployed on the device, it can use ROS as communication method to communicate. However, it can only support communication between devices developed based on ROS, it cannot be used to communicate with off-the-shell devices using different technologies. Furthermore, the messages used in ROS are the original data sent by the device, which has not been standardized. The receiver must know the content of the communication in advance, otherwise it will not be able to understand the received data. In State Of The Art (SOTA), one of the limitations of ROS–YARP middleware is the applicability area, which is limited to ROS and YARP only, while CPSs are widely used in various aspects. The robotic bridge provided by ROBIN project allows the communication between ROS and the automation application through the inter-node communication mechanism in ROS, but it requires the developers to directly create or include in existing ROS packages the valuable features for interacting with other external devices via fieldbuses or industrial network protocols. This requires to the developers to be expert in ROS programming. Besides, the direct operation on existing ROS packages may bring the risk of the damaging the basic functionalities. Moreover, in the real production environment, the robotics applications are significantly sophisticated and dynamic, requiring to the bridge to be flexible enough to react to the continuously changing production environment; to allow this the robotic bridge needs to support the update at runtime and the deployment on demand and this feature is not feasible using only native ROS.

Instead, ROS Edge Node bridges ROS and other IoT platforms using OSGi specification, in such way, the robot's behaviours can be developed at the application level using OSGi instead of ROS. And this allows to have all the advanced OSGi capabilities applied in robotics scenario(i.e., start and stop atomic behaviours at runtime, deploy new ones, import, update and upgrade behaviours at runtime every time a new more stable or more secure version is available).

C. Challenges

ROS Edge Node intends to address the gaps mentioned in the previous section in the current State of The Art (SoTA) and the typical challenges faced in researches on middleware for CPS [16], [17]. More specifically, apart from supporting the interoperability between the ROS-based CPS and other heterogeneous IoT applications, it also supports adaptivity, security and privacy protection and autonomous operation, as explained in the following subsections.

Abstraction and Automatic Adaptation: [18] An ideal middleware for an intelligent environment such as the IoT should provide abstractions at various levels such as heterogeneous input and output hardware devices, hardware and software interfaces, data streams, physicality and the development process. And an Adaptive middleware is usually motivated by the need of adapting the middleware to changes in application's requirements, changes of environmental conditions, fixing middleware's bugs or extending/improving the middleware functionality. ROS Edge Node solution proposes an approach to create a software component to abstract the ROS-based CPS for communicating with other OSGi-based IoT middlewares/applications through the distributed BRAIN-IoT EventBus. The adaptor can be generated according to different ROS platform implementations. For the simplicity, a code generator is provided to speed up the development process.

Security and Privacy: Automatic communication of real-life objects represents a huge challenge in terms of trust, security and privacy. The security and privacy component is needed to provide the integrity of the collected data (stream) and to ensure that the user's privacy is not violated. The data can only be able to connect to authenticated/certified IoT devices. The management support of security and privacy has to be considered as a main function of the middleware for the IoT. The ROS Edge Node adaptor will be integrated with the BRAIN-IoT platform, which introduces a holistic end-to-end trust framework and privacy-awareness and control approach to address the challenge [4]. Currently, the integration between ROS Edge Node and the end-to-end framework guarantees only authenticated ROS-based CPSs can integrate with BRAIN-IoT platform and communicate with other authenticated IoT systems.

Autonomous Operation: Many CPS applications are considered complex systems which can be in a huge number of different states at any point of time. It is generally extremely difficult to develop code to handle all these states effectively and in a timely manner. Having middleware that supports autonomous operations such as self-adaptive, self-resilient, and self-protected services can relax implementing and operating these complex CPS applications. The ROS Edge Node addresses this challenge supporting the development of complex IoT solutions for monitoring and controlling physical environments and systems. Specifically, it simplifies the operation of application providing a an event-driven notification method, which allow avoiding to use polling methods to query the robot or mission status, reducing greatly the network traffics.

D. Relevant Technologies

RosJava Open Source Library[2]: RosJava project provides a pure Java implementation of ROS, and it also can interconnect to an existing ROS environment through the Internet Protocol (IP) address. It provides a client library for ROS communications in java that allows Java programmers to quickly interface with ROS topics, services and parameters through the eXtensible Markup Language (XML)-Remote Procedure Call (RPC) [19] protocol. It provides some common Java API allowing to create new ROS nodes, services, topics in native ROS environment, and the corresponding ROS clients. The library can be fully integrated in OSGi software.

JCodeModel Open Source Library[3]: JCodeModel is a Java code generation library. It provides common API to generate Java programs using Java language.

World Wide Web Consortium (W3C) and WoT: In recent years, W3C organization has developed the WoT [9] standard aiming to achieve interoperability problem between IoT platforms and application domains. WoT provides a mechanism for describing IoT interfaces, allowing IoT devices (physical or virtual entity) and services to communicate with each other, independent of their underlying implementation, and can span multiple network protocols. In addition, WoT also provides a standardized way to define and plan IoT behaviors. WoT Architecture specification is centered on the scope of W3C WoT standardization, divided into several building blocks. The four core building blocks provided by W3C WoT are: *Thing Description*, *Binding Template*, *Scripting Application Programming Interface (API)*, *Security and Privacy Guidelines*. More specifically, The central building block is the WoT TD[4], which can describe the metadata of the object and the network-oriented interfaces and it's the entry point of a Thing. TDs are encoded in a JavaScript Object Notation (JSON) format that also allows JSON-based Serialization for Linked Data (JSON-LD) processing, primarily intended to be a way to use Linked Data in Web-based programming environments. The building blocks allow an application client (a Consumer) to interact with Things that expose diverse protocols through the three types of *Interaction Affordances* defined by W3C WoT Interaction Model representing the capabilities of individual Things: i) *Properties (PropertyAffordance class)* can be used for sensing and controlling parameters, such as getting the current value or setting an operation state; ii) *Actions (ActionAffordance class)* model invocation of physical (and hence time-consuming) processes, but can also be used to abstract RPC-like calls of existing platforms. iii) *Events (EventAffordance class)* are used for the push model of communication where notifications, discrete events, or streams of values are sent asynchronously to the receiver. TD can be used for flexible implementation and simulation (if required). WoT will break the barrier of interoperability of various IoT platforms, thereby contributing to the explosive growth of the market. It doesn't aim to define a new platform, but to use the metadata to bridge existing platforms and standards.

In this paper, these technologies will be used in the following aspects. The ROS Edge Node, will be developed as OSGi bundles, which can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot in BRAIN-IoT federation.

RosJava can be considered as the bridge between ROS world and Java world. It provides an efficient way for the ROS Edge Node to establish a communication with ROS-based devices. Different ROS functionalities will be mapped into different OSGi services in the ROS Edge Node. The mapping procedure will be done automatically through JCodeModel library with a TD of the underlying ROS environment. Anyway, the corresponding formatting procedure of events and integration with BRAIN-IoT framework should be done by developers.

III. Architecture

This section details the ROS Edge Node architecture along with its role in the overall BRAIN-IoT platform.

A. Overview in BRAIN-IoT Context

ROS Edge Node will be integrated with BRAIN-IoT Fabric [20] infrastructure service, which is composed with a set of the computing resources (physical/virtual machines) and provides a distributed OSGi execution environment allowing the interaction between the OSGi services deployed on it through events, thanks to the implementation of OSGi Alliance specifications for Remote Services and Remote Service Admin [5]. Each BRAIN-IoT Fibre is an OSGi R7 framework, and different Brain-IoT OSGi bundles deployed on local and remote Fibres can communicate with each other using some specific strongly typed BRAIN-IoT events delivered in the asynchronous BRAIN-IoT EventBus [20] according to the BRAIN-IoT approach. The BRAIN-IoT Nodes are the Service Fabric Fibres with different BRAIN-IoT Services deployed on them, they can be of two types: BRAIN-IoT Processing Nodes and BRAIN-IoT Edge Nodes. The first ones are a sort of Service Fabric Fibres deploying IoT application logic and controlling the CPS behaviours through their adaptors supported by the machine learning algorithms. The latter ones are Fabric Fibres with the installed edge components deployed on the top of them. BRAIN-IoT Fabric allows users to label the BRAIN-IoT nodes, thus to guide where the BRAIN-IoT service, satifying the required capabilities, should be deployed at runtime. The BRAIN-IoT architecture allows the dynamic redeployment of 1) new functionalities for enhancing/extending the behaviour of robots according to external events. 2) specific behaviours to new "virgin" robots, which might be needed to extend the fleet of robots or replace demaged/low batteries ones. Each ROS Edge node can be considered as an access point or an adaptor to ROS-based CPS, to allow

[2]https://github.com/rosjava

[3]https://github.com/phax/jcodemodel

[4]https://w3c.github.io/wot-thing-description/

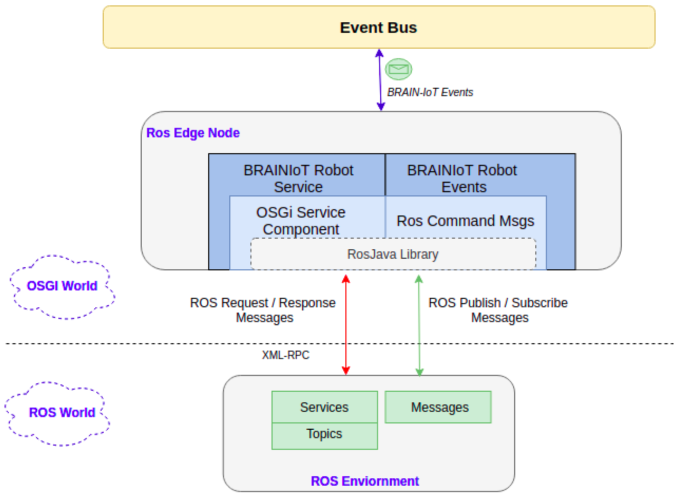


Fig. 1. ROS Edge Node structure

heterogeneous IoT applications running on the processing nodes to control the robots. In the BRAIN-IoT platform, the authors' contribution is to develop the ROS Edge Node as OSGi Declarative Service, so that from the northbound it can receive the interested BRAIN-IoT events from other different IoT platforms in a distributed environment, then construct the data and send to the connected ROS environment. Therefore, any new functionalities for enhancing/extending the behaviour of robots according to external events can be developed using OSGi instead of ROS. From the southbound, the ROS Edge Node is able to retrieve the information from ROS and inject to the BRAIN-IoT Fabric as BRAIN-IoT events, which will be received by other BRAIN-IoT services. The ROS Edge Node aims to achieve the interoperability between heterogeneous IoT applications integrated within BRAIN-IoT platform and the ROS-based CPSs. It exposes the ROS functionalities as OSGi services.

The architecture of ROS Edge Node is shown in Fig.1. For its development and validation, the authors have used the ROS simulation for BRAIN-IoT service robotic use case provided by the Robotnik Automation S.L.L. (see Section IV). The ROS Edge Node has two main requirements: i) to expose all the relevant ROS functionalities as OSGi services (task done by *OSGi Service Component*). The objective is to make the adaptor able to interoperate with the ROS environment leveraging APIs provided by the open source Rosjava project, in this way, the adaptor is able to send/receive the ROS request/response messages from/to the ROS services and publish/subscribe to the ROS topics between the OSGi world and ROS world. ii) To collect and format of the BRAIN-IoT events from different sources based on the Publish-Subscribe pattern[5] (task done by *BRAIN-IoT Robot Service*). The adaptor receives the events from heterogeneous platforms in the distributed BRAIN-IoT Fabric environment and constructs them as Java objects representing ROS messages, then transforms to ROS environment through the exposed OSGi services. In contrast, the adaptor also retrieves the ROS messages from the native services/topics and convert to the BRAIN-IoT events, then deliver them in the distributed EventBus. The connectivity with ROS is configurable through the ROS environment variable *ROS_MASTER_URI* by default whose value is configurable on-the-fly in order to set up which machine as a ROS master when a new ”virgin” robot join the fleet of robots.

[5]https://en.wikipedia.org/wiki/Publish\%E2\%80\%93subscribe_pattern

B. *ROS Edge Node Approach*

The ROS Edge Node adaptor is further explained in this subsection, detailing its approach, implementation steps and the addressed challenges.

1) Abstraction of ROS environment: It aims to address the interoperability challenge when integrating other IoT devices in BRAIN-IoT platofrm. As Fig.1 shows, the basic function of ROS Edge Node is to map the ROS messages to BRAIN-IoT events and vice-versa. The BRAIN-IoT services interact with each other by leveraging the Requirements and Capabilities metadata provided by default by all OSGi Bundles, the events issued by a *source* BRAIN-IoT service contains the sufficient information presenting the Requirements of this service, in the meanwhile, the events also identify the Capabilities of the *sink* BRAIN-IoT services that will consume them.

The ROS Edge Node simply interconnects to the ROS environment thanks to the RosJava library which provides a simple interface enabling the ROS Edge Node to automatically generate a set of Java object classes, which is totally compliant with native ROS messages' structure in the ROS environment. The instances of the classes could be transformed to the ROS environment. However, it's not possible to make the BRAIN-IoT events types the same as the names of the native ROS messages. For a single robot, there could be hundreds of types of native ROS messages in the ROS environment and their data structures are in general very complex, direct mapping between the ROS messages types and BRAIN-IoT events is inefficient and will increase the complexity of the usage of the events for other IoT applications. Also, different robots having same functions may use completely different types of ROS messages as commands. In such case, if directly mapping each ROS message into an event, the difficulty to integrate different robots in a system will be increased. Thus, some common data types including necessary information need to be defined and shared between diverse IoT applications. So it's necessary for ROS Edge Node to format the received events into the specific Java objects that are compliant with ROS messages.

The ROS Edge Node wraps also the native ROS functionalities, exposing them to the OSGi services for heterogeneous IoT applications' access, by creating the corresponding ROS services and publish/subscribe clients in OSGi bundles, through the APIs provided by RosJava project. The exposed OSGi services are a set of java classes containing multiple robot operations mapped to Java methods. The *BRAIN-IoT Robot Service* in Fig.1 is a wrapper of the exposed OSGi services, with the specific Capabilities information represented by the consumed event types. The ROS Edge Node is responsible for communicating with other BRAIN-IoT services using Events. When the robot service receives an event from the EventBus, it will construct a ROS message in Java type and perform the corresponding action by calling the exposed ROS functionalities. Thanks to the BRAIN-IoT solution, the service will be deployed on the BRAIN-IoT Fabric by the event-driven mechanism. It's completely de-coupled from the underlying BRAIN-IoT Fabric runtime.

2) The Autonomous Operation: As mentioned in Section II-C, supporting autonomous operation is one of the challenges for a CPS adaptor. In the ROS Edge Node, the autonomous operation is fully supported by a feedback mechanism: the ROS Edge Node is responsible for continuously querying the execution status of CPS where it is installed and then to issue an response event if the status changes. This allows building services, which leverage the ROS Edge Node, which are "smarter" and reducing the communication workload on the EventBus,(see Fig.6 in Section IV).

3) Automatic Adaptation and Standards Compliant: This is a usual task that needs to be supported by such type of solutions. For ROS-based devices, this challenge is addressed by ROS Edge Node. As mentioned above, the exposed service components are a set of java classes containing multiple methods for robot operations. The operations are done through ROS services/topics clients in Java code via simple API provided by RosJava library. Normally, when developers create the java clients for the used native ROS functionalities, the java clients could be grouped in one or more Java classes for better organization.

Since the services or topics and their related methods in a component class have the same structures, the authors achieve to automatically realize the adaption by creating a Code Generator to automatically generate the OSGi service classes from a predefined configuration file as the input. Since all entities in ROS-based CPS communicate through services and topics, the authors choose to use the W3C WoT TD, which is a general standard for both integrating diverse devices and interoperability of diverse applications to describe the ROS functionalities. In the proposed solution, WoT TD describes the interfaces for OSGi to expose the ROS services and topics. In this way, a standard approach is used to describe the ROS API and to generate the corresponding OSGi services, this allows to have a solution WoT integration-ready and highly reusable. The ROS functionalities are compliant with TD specification: 1) the topics are as *Properties Interaction Affordances* 2) the services are described as *Actions Interaction Affordances*.

For a ROS service, the part of TD file is shown as Fig.2. More specially, there could be a set of ROS services described in the *Actions* element, and for each of them, the information will include *serviceClientName*, *serviceName*, *serviceType*, *serviceRequestType*, *serviceResponseType* and *ClassName*, which are used by the Code Generator to create service clients in OSGi world. The Code Generator uses the *ClassName* property to generate multiple component classes

```
{
   "@type":"Thing",
   "properties":{...},
   "actions":{
      "serviceClientName":{
         ...
            "serviceName":{...},
            "serviceType":{...},
            "serviceRequestType":{...},
            "serviceResponseType":{...},
            "ClassName":{...}
         },
        ...
   }
}
```

Fig. 2. Thing Description of ROS Environment

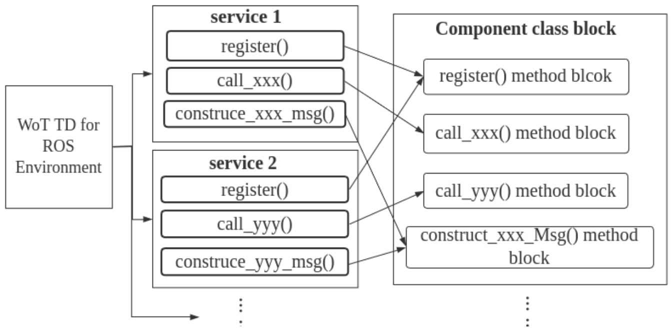


Fig. 3. Expose of ROS Environment to OSGI Services Using WoT TD

for different types of robot functionalities, and each component can contain multiple service clients and the different operations could be done through the methods called by each client. The values of other properties will be used as the arguments of the generated methods. Similarly, the ROS topics will be described in the *Properties* element, including the information about *Role*, *ReferenceName*, *TopicName*, *TopicType*, *MessageType* and *ClassName*. The value of the *Role* property could be *publisher* or *subscriber*, so the corresponding client type will be created with the client name equal to the value of *ReferenceName*.

To automatically expose relevant ROS functionalities and speed up the ROS Edge Node development for the ad-hoc ROS platforms, A code generator is provided to generate the artefacts automatically by taking the TD as a configuration file, to describe the ROS services and topics, as shown in Fig.3, which is demonstrated in Fig.5 in Section IV. This approach greatly reduces the dependence for developers in the process of adapting to different ROS-based CPSs, increasing the development simplicity. Developers can therefore focus more on the development of BRAIN-IoT services rather than on adaptation work.

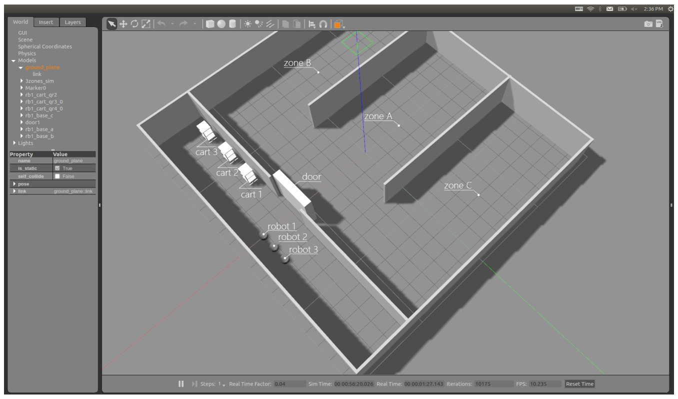
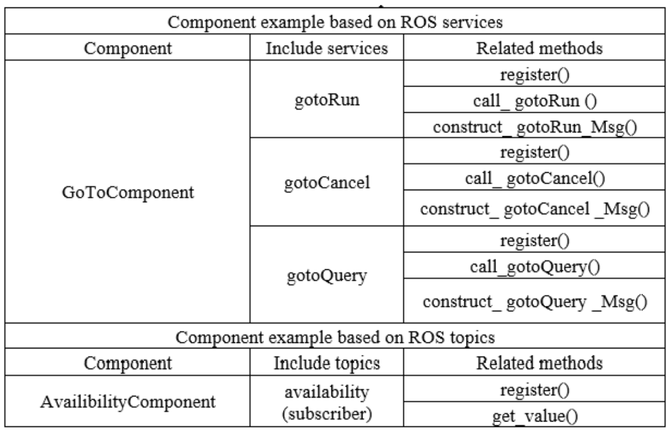


Fig. 4. Warehouse simulation



Component example based on ROS services		
Component	Include services	Related methods
GoToComponent	gotoRun	register()
		call_ gotoRun ()
		construct_ gotoRun_Msg()
	gotoCancel	register()
		call_ gotoCancel()
		construct_ gotoCancel _Msg()
	gotoQuery	register()
		call_gotoQuery()
		construct_ gotoQuery _Msg()
Component example based on ROS topics		
Component	Include topics	Related methods
AvailibilityComponent	availability (subscriber)	register()
		get_value()

Fig. 5. Method Blocks of Automatically Exposed ROS Services from TD file

IV. Use Case Demonstration

In this section, a brief description of ROS simulation used for the Brain-IoT Service Robotic Use Case and the result obtained evaluating the cross-platform communication mechanism using it are presented.

A. Use Case Introduction

Smart warehouse is a popular application of Industry 4.0. The basic elements of a smart warehouse include AMR, heterogeneous IoT devices and cargo carts. To be "smart", these elements need to interact and cooperate with each other. The use case aims to realize one of the basic function of smart warehouse: the cart movement. The Fig.4 shows the warehouse with three zones in 3D perspective in Gazebo simulator: a docking area, a unloading area and a storage area. The last two zones are separated by an automatic door, which is an IoT device. There are three *rb1* base robots (robot1, robot2 and robot3) in the docking area responsible for moving all carts to the storage area. In the unloading area there are 3 carts (cart1, cart2 and cart3) and each has a different QR code attached. The storage area is divided into three sub-zones (zone A, zone B and zone C). The robots need to pick up these carts from the unloading area, pass through the door and place them in the storage area according to the received commands from the robot application.

B. Robot System Design

In order to demonstrate how the adaptor bridges the ROS environment and the OSGi environment, the authors implemented a simple multi-agent robot system based on OSGi to control the simulated robots.

The system contains three system parts: a Robot Behaviour Service, a ROS Edge Node Service and a Door Edge Node Service. There are several tables storing the coordinates of the carts and the storage positions will be used as the shared resources among three robots. In the ROS simulation, the basic task for each robot is that starting from the docking area, it needs to go to the picking area for picking up a cart, then pass through a door to the storage area and finally place the cart, the procedure is controlled by the Robot Behaviour Service. The new functionalities for enhancing/extending the behaviour of robots according to external events can be dynamically upgraded and redeployed in the BRAIN-IT architecture. Based on that, the authors define the events in three types: action, query and cancellation. The events in action type includes *WriteGoTo*, *PickCart* and *PlaceCart*, which present the basic functions of the robot. There are also other events defined for querying the execution status of corresponding actions and for cancelling current actions.

In the use case, there are three services related to the *WriteGoTo* action event, a WoT TD file is defined for the communication with the robotic system, the code generator using the TD as input will automatically generate the class named *GoToComponent* as shown in Fig.5 by using the open source JCodeModel library which is a Java code generation library. Specifically, according to the ROS functionalities described in the TD, there are three ROS service clients (e.g. gotoRun, gotoCancel, gotoQuery) will be created in the generated class. when the ROS Edge Node receives an event, the corresponding *construct_XXX_Msg* method representing the operation of the client will be called to construct a Java object representing the ROS message as a ROS service request to be sent to the native ROS environment through the *call_XXX* method of the service client, where the *XXX* stands for the name of the service client.

The Robot Behaviour Service continuously checks the shared tables to get a task and then it controls the robots through a sequence of BRAIN-IoT events to finish the mission. The events will be received by the ROS Edge Node Service and sent to the connected robot. As an example shown in Fig.6, after the ROS Edge Node receives a *WriteGoTo* event from the Controller Service containing the coordinates where the robot should go, with the autonomous operation of the ROS Edge Node, only twice communications are needed between the Controller Service and the ROS Edge Node, one for sending the action command, while the other one for receiving the action result. When the robots detected a door on the way to the storage area, it reports the situation and the Robot Behaviour Service will instruct the Door Edge Node Service to open the door.

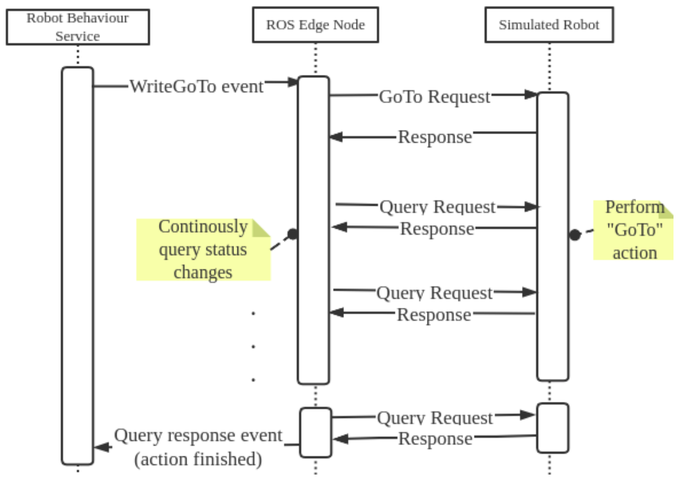


Fig. 6. Communication Procedure of "WriteGoTo" action for ROS Edge Node

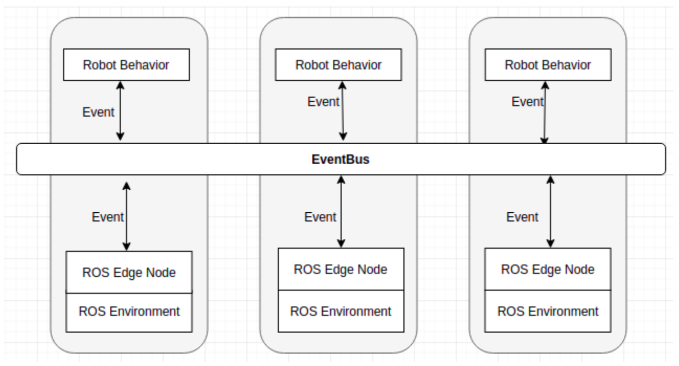


Fig. 7. Deployment of Robot System in the distributed BRAIN-IoT Fabric

In the test, the robot system above is deployed on a distributed Fabric environment containing multiple Raspberry Pis and one Linux server running the ROS simulation as BRAIN-IoT nodes in a same network. The Linux server is labeled in advance when BRAIN-IoT Fabric cluster is created and the ROS Edge Node Service is automatically deployed on it when the label is detected. Since there are three simulated robots, three instances of the ROS Edge Node Service instantiated by OSGi Configuration Admin Service Specification [5] to connect to the robots through IP address, the interested events will be delivered to the ROS Edge Node through the integrated EventBus. In the multi-agent system, each Robot Behaviour Service instance will control one robot to take a task from the shared table, after one task is finished, it will start next iteration. In this case, the deployment of Robot Behaviour Service and the ROS Edge Node Service in the real physical environment can be shown as Fig.7. Meanwhile, The door in the simulation environment is a IoT device controlled by the Door Edge Node Service, which is simulated in the ROS environment.

C. Validation in Simulation Environment

When the multi-agent robot system is activated, each Robot Behavior Service instance will check the shared cart table to find the pending tasks. When a cart to be moved is found in the table, Robot Behavior will change the status of the cart to *moving* and start the moving procedure. Robot behaviors will deliver a sequence of events to the corresponding ROS Edge Node one by one to command the robot to move to cart position, pick up the cart, move the cart to specific position in the storage area and finally drop the cart. When ROS Edge Node receives an event, it extracts the information contained in the event, constructs a ROS message in Java type, and communicates with the robot through the exposed OSGi services. ROS Edge Node is responsible for continuously querying the execution status from the specific ROS service/topic, when the action of an event is finished or something wrong happened, a feedback event will be issued to inform the Robot Behavior to ask for the next action. During the moving procedure, if the door is closed, it will scan the QR code of the automated door, and return a *DoorFound* event to the Robot Behavior. The Robot Behaviour will send a *OpenDoor* event to the Door Edge Node service to open the door.

After finishing the moving procedure, the Robot Behavior Service will change the status of the cart to *moved* in the table and search for the next mission. Finally all carts are moved in ROS simulation.

D. Validation in Physical Environment

To prove the feasibility of the approach proposed in the paper, ROS Edge node has been installed on a real *rb-1 base* mobile robot. Due to the difference of the simulated world and the physical world, the autonomous robotic system is not used in the physical environment, but authors implemented an Orchestrator service, which provides a simple interface for users to manually send a sequence of BRAIN-IoT events including the coordinates to control the actions of the robots in a physical environment. The Orchestrator is implemented as a OSGi Declarative Service and it injects a sort of commands in the Apache Felix Gogo [6], which is a subproject of Apache Felix implementing a command line shell for OSGi, which could be accessed via its web interface in a distributed BRAIN-IoT Fabric environment. When a user enters the command in Gogo shell, the corresponding method in the Orchestrator will issue a specific event to EventBus. The Orchestrator service could be installed on any BRAIN-IoT node. Finally, ROS Edge Node is able to receive the events and the robot will move towards to the target positions.

V. Conclusion and Future Work

The paper has presented the ROS Edge Node, which enables the interoperability between the ROS-based CPS applications and other heterogeneous IoT platforms in a sophisticated IoT software ecosystem, based on the available services in BRAIN-IoT framework. This solution provides several innovative features, to ease such interaction. Firstly, it can be dynamically deployed and flexibly scaled on demand to connect to multiple ROS-based CPSs at runtime whenever a new CPS joins the

[6]https://felix.apache.org/documentation/subprojects/apache-felix-gogo.html

cluster. Secondly, it provides an approach to automatically expose the ROS functionalities as OSGi services for bridging the ROS world and OSGi world. Thirdly, ROS Edge Node maximizes the flexibility of the mechanism by supporting customized autonomous operation to detect the changes of action status and issue feedback events, thus greatly reduces the communication load on EventBus and its dependence on other system components. Finally, the use of WoT TD standardizes the description of the ROS functionalities.

Compared with the existing CPS middleware or IoT middleware, the proposed solution presents several advantages, but some further developments are still ongoing. The integration with the BRAIN-IoT End-to-End Security Framework are still under development. Besides, its functionalities will be enriched more, such as the graphical monitoring of robot status and the warehouse coordinates at runtime by integrating with other BRAIN-IoT monitoring tools. ROS Edge Node will be tested on the real robots with a more complex Robotics use case and the performance will be compared with other existing solutions, in the future work.

ACKNOWLEDGMENT

The work presented here was part of the project "Brain-IoT-model-Based fRamework for dependable sensing and Actuation in iNtelligent decentralized IoT systems" and received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780089.

The simulation environment is provided by Robotnik Automation S.L.L.

REFERENCES

[1] N. Boulila, "Cyber-physical systems and industry 4.0: Properties, structure, communication, and behavior," 04 2019.
[2] S. Barai, M. K. Kundu, and B. Sau, "Path following of autonomous mobile robot with distance measurement using rfid tags," in *2019 IEEE International Symposium on Measurement and Control in Robotics (ISMCR)*, 2019, pp. A3–4–1–A3–4–4.
[3] nud Lasse Lueth, "Iot platform companies landscape 2019/2020: 620 iot platforms globally," https://iot-analytics.com/iot-platform-companies-landscape-2020/, December 2019.
[4] D. Conzon, M. R. A. Rashid, X. Tao, A. Soriano, R. Nicholson, and E. Ferrera, "Brain-iot: Model-based framework for dependable sensing and actuation in intelligent decentralized iot systems," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–8.
[5] O. Alliance, "Osgi core release 7," OSGi Alliance, Tech. Rep., Apr. 2018.
[6] https://docs.oracle.com/middleware/1212/wls/WLPRG/overview.htm#WLPRG107.
[7] G. Di Modica, F. Pantano, and O. Tomarchio, "Snps: An osgi-based middleware for wireless sensor networks," 09 2013, pp. 1–12.
[8] https://projects.eclipse.org/proposals/eclipse-sensinact.
[9] https://www.w3.org/TR/wot-architecture/.
[10] "Community metrics report," http://download.ros.org/downloads/metrics/metrics-report-2019-07.pdf, July 2019.
[11] M. Aragão, P. Moreno, and A. Bernardino, "Middleware interoperability for robotics: A ros–yarp framework," *Frontiers Robotics AI*, vol. 3, p. 64, 2016.
[12] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: Yet another robot platform," *International Journal of Advanced Robotic Systems*, vol. 3, 03 2006.
[13] R. Arrais, P. Ribeiro, H. Domingos, and G. Veiga, "Robin: An open-source middleware for plug'n'produce of cyber-physical systems," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, p. 1729881420910316, 2020. [Online]. Available: https://doi.org/10.1177/1729881420910316
[14] A. Pletsch, "Codesys eases programming for multiple controls hardware," vol. 50, pp. 34–35, 06 2004.
[15] A. Santos, A. Cunha, N. Macedo, R. Arrais, and F. N. dos Santos, "Mining the usage patterns of ros primitives," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3855–3860.
[16] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, and I. Jawhar, "Middleware challenges for cyber-physical systems," *Scalable Computing. Practice and Experience*, vol. 18, no. 4, pp. 331–346, 2017.
[17] M. A. Chaqfeh and N. Mohamed, "Challenges in middleware solutions for the internet of things," pp. 21–26, 2012.
[18] N. Rosa, D. Cavalcanti, G. Campos, and A. Silva, "Adaptive middleware in go - a software architecture-based approach," *J Internet Serv Appl 11, 3 (2020)*, 05 2020.
[19] T. Tomlinson and J. VanDyk, "Xml-rpc," 01 2010.
[20] R.Nicholson, T.Ward, D.Baum, X.Tao, D.Conzon, and E.Ferrera, "Dynamic fog computing platform for event-driven deployment and orchestration of distributed internet of things applications," pp. 239–246, 2019.

REPLICA: A Solution for Next Generation IoT and Digital Twin Based Fault Diagnosis and Predictive Maintenance

Rosaria Rossini, Davide Conzon, Gianluca Prato, Claudio Pastrone
IoT and Pervasive Technology Area
LINKS Foundation
Turin, Italy
{name.surname}@linksfoundation.com

João Reis, Gil Gonçalves
SYSTEC, Research Center for Systems and Technologies
Faculty of Engineering, University of Porto
Porto, Portugal
{jpcreis , gil}@fe.up.pt

***Abstract*—Nowadays competitiveness goes through several aspects: digitalization, productivity and environmental impact. Technology is advancing fast and helping industries to obtain more and more detailed data about their processes and equipment. In fact, the possibility to monitor and control each part of the process is a strong base on which a more intelligent and focused control can be built. Technology advance brings innovation and the possibility to manage the production in terms of ”near future” through AI prediction and decision-making support. Forecasting demands and planning production, optimizing process by reducing costs and improving efficiency without corrupting the quality of the product is a big challenge at the plant level. In this paper, a flexible, scalable architecture for intelligent digital twin realization called REPLICA has been proposed to cope with such problem and help industries to advance and discover possible optimizations. This architecture sits on top of two European projects, namely CPSwarm and RECLAIM, where their contribution focus on distributed simulation and optimization, and Adaptive Sensorial Networks, correspondingly. As a validation process, a hypothetical use case is presented, detailing the key differentiating points and benefits of the proposed architecture.**

***Index Terms*—IoT, Digital Twin, AI, Fault Diagnosis, Predictive Maintenance**

I. Introduction

In the era of Industrial Internet of Things (IIoT) and Industry 4.0, complex electromechanical systems can be equipped with a variety of sensors providing new opportunities for the development of Health Monitoring and Management Systems. These new opportunities target an optimum exploitation of available information in order to maximize the performance of the machinery and optimize the process. Focusing on the increase of production reliability and safety, as well as on the reduction of costs, there is an ever increasing industrial need not only for accurate and on time online diagnostics, but also for a robust and early estimation of the Remaining Useful Life (RUL) of the defected components, within a high confidence interval, independent of the operating conditions.

For this reason, and many others, a new concept of ’interaction’ with the process arose; a concept of controlling and monitoring a replica instead of the real object, a perfect virtual replica that interacts with both humans and machines: a Digital Twin. The concept of “twin” is originally derived from National Aeronautics and Space Administration (NASA)’s Apollo Project when the aircraft’s twin body was a real physical system [1]. Twin models help astronauts and staffs make decisions under emergency situations. Digital twin integrates the life cycle of a machine [2], and achieves a closed loop and optimisation of the machine design, production, operation, and maintenance, etc. In Magargle et al. [3], a multi-physical twin model is built to monitor the status of the brake system through multiple angles. NASA hopes to realise the health management and residual life prediction of the aircraft by building a multi-physical, multi-scale Digital Twin model [4], furthermore serveral roles are envisioned for Digital Twin in the industry 4.0 scenario [5].

The present paper focuses on the proposition of an intelligent digital twin architecture called REclaim oPtimization and simuLatIon Cooperation in digitAl twin (REPLICA), that focuses on two important aspects: 1) plug’n’play of models on demand and 2) Workflow design to orchestrate the models used in the Digital Twin itself. Aspect (1) aims to ease the integration and removal of models into digital twins, whenever a new version of the software is available or it performs any necessary correction in the used models. The goal of the latter aspect is dedicated to the creation of a pipeline that can manage the flow of data among all the available models. These models might be from pure data processing and decision making, from sensor and actuator integration, to third party synchronization with information systems such as Manufacturing Execution Systems (MES) or Enterprise Resource Planning (ERP). Most digital twins, and in particular intelligent digital twin architectures focus on providing the best set of models that one should have to accomplish, e.g. predictive maintenance, from the type of simulation required to machine learning models that should be refined based on newly acquired data. Furthermore, REPLICA can allow a flexible and distributed deployment in such a way that both completely cloud or mixed edge/cloud solutions deployment are accepted, with respect to the need of the specific application.

However, these architectures are normally rigid and do not support changing software models or even easily set up the orchestration of those resources. Fixed data flows are generally hardcoded, meaning that if an implementation needs to be modified, changes in code are required. This is often not recommended since these changes might negatively influence the stability of the system. To this intent, an architecture that addresses these challenges is presented.

The paper is organized as follows: in Section II, the authors introduce a literature review about predictive maintenance and fault diagnosis based on digital twin. Complementary, Section III presents the core technologies of the solution presented in this paper. Section IV describes the architecture of the solution proposed and Section V presents a first prototype implementation followed by Section VI in which a possible application is described. Finally, Section VII concludes the paper by summarizing and discussing the work.

II. Literature Review

With the rapid advancement of Cyber-Physical Production Systems, Artificial Intelligence (AI) and IIoT, Digital Twin (DT) has gained increasing attention due to its capability to adapt and replicate the industry processes. Accordingly to these changes, many different DT architectures have been proposed to realize several use cases in an intelligent and complex production system. Industrial AI [6] brings to the processes self-aware, self-adapt, and self-configure functionalities and facilitates the integration of the DT.

In [7], the authors propose to insert an intelligent DT in the Cyber Layer architecture. The concept has been partially realized with two industrial use cases, namely a modular production system as well as a metal forming industrial process to show its potential and gains over the challenges in Cyber-Physical System (CPS), i.e., synchronization throughout the lifecycle of a cyber-physical production system; development of the DT, which can contain different models; the interaction between DT, both for the purpose of co-simulation and operation data exchange; and the active data acquisition.

In [8], the authors present a methodology for enabling DT using advanced physics-based modelling in predictive maintenance. This methodology for advanced physics-based modeling aims to enable the DT concept in predictive maintenance application and consists of two main points: digital model creation and DT enabling. Then, the user is able to define, create and utilize the digital model of a resource, as well as its DT. The integration of DT and deep learning in CPS environment has been also proposed in [9] for the development and realization of smart manufacturing.

In [10], the authors present solutions for fault diagnosis based on DT. The paper includes an experiment and interesting results obtained with the software proposed. Compared to the solution presented in this paper, this work has a limited flexibility since it is only suitable for fault diagnosis.

In [11], [12] and [13], the authors show how it is possible to build a DT of machines and systems of systems to allow autonomous smart manufacturing, but these works, while interesting, are not specifically presenting a solution for fault diagnosis and predictive maintenance.

The authors of [14] introduce a solution for predictive maintenance of computer numerical controlled machines, based on DT. They demonstrate how the exploitation of a DT for predictive maintenance can provide better results compared to more traditional approaches. Even if this work provides a good example of application of DT for predictive maintenance, it doesn't aim to present a solution that can be leveraged in other scenarios.

In this paper, the authors intend to propose a novel architecture that supports several features missing in the other solutions presented above. Specifically, the proposed solution is not based on a set of fixed components but it can integrate heterogeneous modules, in terms of Internet of Things (IoT) sensors, AI algorithms and simulation tools, easing its customization in different use-cases. Furthermore, thanks to the flexibility guaranteed by the distributed nature of the system, the setting-up of the platform can easily be adapted to the each specific industrial infrastructure, selecting the most suitable mix of edge/cloud deployed components.

Moreover, the proposed architecture can support the creation at runtime of workflows both among the AI modules as well as between the IoT sensors and the models. This drastically reduces the time needed to run and collect results from the AI algorithms. In these terms, a possible process optimization can be quickly evaluated and eventually discarded if not appropriate. Finally, all the entities (sensors, AI modules and simulators) can be substituted following a plug&play approach that ease the adaption of the system to the changes in the physical world.

III. Core technologies

A. RECLAIM platform

Following the industry 4.0 paradigm, the business models of manufacturing companies need to be transformed, resetting their strategies to improve productivity and quality. The current maintenance strategies often require the user to manually analyse data collected to extract useful information from them and, furthermore, periodic human inspection is required to assess the real condition of the assets monitored.

Currently, the lack of continuous operation and health status monitoring tools and predictive maintenance solutions lead to unpredictable situations in industry like sudden machine operation failures. In this case, the current common procedure is to ask the intervention of technicians, which then try to repair and solve the problem. This causes several problems: it is time-consuming; it leads to production delays since the machine is stopped until it is not repaired; it doesn't support resources distribution. The industry 4.0 paradigm goes in the direction to address such problems through different actions: 1) re-manufacturing systems for material and resource efficiency, 2) increased flexibility in changing machine operation purpose,

3) application of big data analytics techniques, and 4) predictive analytics and model-based forecasts and optimization procedures, based on completely data-driven processes.

These four suggestions have been the funding principles of the RE-manufaCturing and Refurbishment LArge Industrial equipMent (RECLAIM) concept definition. The main objective of the project is to increase productivity, extending the lifetime of the machines and reducing the time and cost of machinery refurbishment and/or re-manufacturing. This objective will be achieved designing and developing a set of tools supporting several activities: from the monitor of machines' health status, to the implementation of adequate recovery strategy (e.g., refurbishment, re-manufacturing, upgrade, maintenance, repair, recycle, etc.). To achieve this, the RECLAIM outcomes will include two main components: an Adaptive Sensorial Network used to collect data and a Decision Support Framework (DSF) for optimization based on different criteria. Specifically one of the technologies supporting the DSF is the proposed REPLICA where simulation and optimization is used for fault diagnosis. The Adaptive Sensorial Network is one of the key elements to be used in the proposed architecture and is seen as an entry point for the essential data to be used.

B. CPSwarm Simulation and Optimization Environment

As indicated in [15], the CPSwarm Workbench - the set of tools released by the project for the development of CPS swarms applications - includes also a Simulation and Optimization Environment, used to evaluate the performance of a swarm solution. Such solution is composed mainly by: the Simulation and Optimization Orchestrator (SOO), which oversees the simulation and optimization tasks; a set of Simulation Managers (SMs), which provide common Application Programming Interface (API) to control heterogeneous Simulation Tools (STs); and an Optimization Tool (OT) used to perform the optimization processes. The network-based architecture is depicted in Fig. 1.

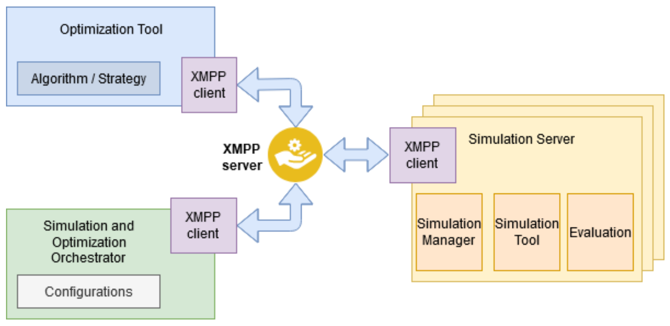


Fig. 1. Network-based Architecture from CPSwarm for Distributed Optimization and Simulation.

Such environment is useful both to simulate the behaviour of a designed swarm solution in a ST, leveraging the ST's Graphical User Interface (GUI) to evaluate its behaviour; and, on the other side, to optimize the controller parameters of algorithm/module, and possible aspects of the problem, i.e., the number of CPSs used, leveraging evolutionary design methodologies. In the latter case, candidate parameter sets are ranked based on a fitness score computed after the controller was executed with those parameters in a predefined environment. Successful parameter sets are then adapted to produce a new generation of candidates to be tested. This is a high time- and resource-consuming process, which requires a high number of simulation runs. To address this, the CPSwarm solution allows to parallelize the execution of these simulations, reducing the times required to complete an optimization. For this objective, the Simulation and Optimization Environment has a network-based architecture, allowing to parallely use a set of STs distributed on different machines [16]. This architecture has been implemented leveraging the eXtensible Messaging and Presence Protocol (XMPP) protocol, already tested executing multiple simulations on Robot Operating System (ROS)-based STs, i.e., Stage, Gazebo and Virtual Robot Experimentation Platform (V-REP). In the last release of the software (available as open-source on github[1]), a set of technologies have been integrated to improve its scalability and easy-to-use, i.e., docker and Kubernetes. Such final release has been tested, showing that it is able to scale till 128 SMs and that the time required to complete one optimization is inversely proportional to the number of STs used. Finally, a proof of concept has demonstrated the ability to deploy the controller with the optimized parameters onto CPSs.

The concept of distributed simulation and optimization is brought to the proposed REPLICA architecture by the CPSwarm results and the whole orchestration process and main building blocks are inspired by this project.

IV. Architecture

This section introduces the REclaim oPtimization and simuLatIon Cooperation in digitAl twin (REPLICA) architecture that has been designed to provide an infrastructure and be used for Digital Twin-based fault diagnostics and predictive maintenance solutions, which can be easily deployed and customize in different Industrial IoT environments.

REPLICA is composed by several modules (shown in Figure 2), mainly subdivided in two blocks: *Backend* and *Frontend*. The first one contains three main components: Artificial Intelligence (AI) Environment that hosts the AI modules, Digital Twin Orchestrator (DTO) that is used to orchestrate the operations done by the REPLICA and the Simulation Environment that is a distributed environment including several heterogeneous simulators deployed into different machines. The latter one instead contains two applications: one devoted to show the results obtained and another one for the configuration of the component. These modules will be described in the remainder of this section.

As explained in Section I, the DT concept concerns the integration of three main components: the data collected by IoT sensors; the realistic models of the real devices and the

[1]https://github.com/cpswarm/SimulationOrchestrator/wiki/Simulation-and-Optimization-Environment

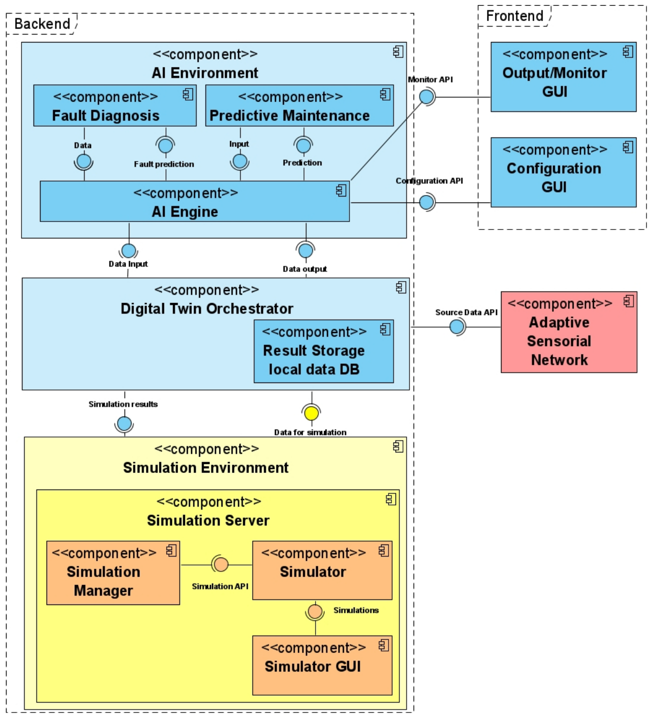


Fig. 2. REPLICA Architecture.

synchronization with those using the data collected; and a set of AI modules connected to these models. REPLICA fully supports this concept, providing the infrastructure to integrate these technologies.

In REPLICA, the Digital Twin Orchestrator (DTO) is the module in charge to manage all the IoT data flow coming from the field: machinery data, historical data and other data from legacy systems already present in the shop-floor. As described in Section III, these data are flowing through a component of the RECLAIM platform, the *Adaptive Sensorial Network*. Furthermore, the DTO is in charge to create the correct flow among the AI modules running in the AI environment and the machine models running in the Simulation Environment. Finally, the DTO oversees the storing and organization of the processed and simulated data, which are saved in a local database.

In REPLICA every machinery of interest has a corresponding realistic model running in one of the simulators integrated in the *Simulation Environment*. Specifically, the *Simulation Environment* is a distributed environment based on the one presented in Section III-B. Similar to what has been presented for the original solution about swarm intelligence, the environment supports a set of heterogeneous simulators distributed in different machines. Each of these simulators is wrapped by a *Simulation Manager*, and the role of this component is to abstract the functionalities provided by the simulators using the standard API exported by the DTO. In this way, the DTO can: 1) control the simulators to run the required simulations; 2) inject the data needed to keep the models synchronized with the real machines; 3) inject in the simulators data produced by the AI algorithms (for example to simulate failures); 4) receive from the simulators the produced results. Finally, the *Simulation Environment* supports for each integrated simulator one advanced Simulator GUI that allows to monitor the simulated device. This GUI is the one integrated in the simulator, which provides a graphical representation (also 3D) of the simulated device.

Finally, in REPLICA the AI algorithms are hosted and executed in the AI Environment. This environment allows to host and run heterogeneous algorithms for fault diagnosis and predictive maintenance. Besides the algorithms, the environment also hosts a module called AI engine. This module has the objective to orchestrate the algorithms creating the needed workflows among them. Furthermore, the AI Engine uses the API provided by the DTO to interconnect the AI modules with the machine models and the data coming from the shop-floor.

The architecture is completed by the two interfaces in the *Frontend*: the OutputMonitor GUI is used to monitor in real time the results produced by the running solutions in a user friendly interface. Instead, the Configuration GUI is leveraged by the users of the system to configure the AI engine for the needed tasks.

The proposed architecture aims to provide the following key features: 1) Allowing the integration of heterogeneous components in terms of sensors data collected from the field, AI algorithms and simulators running accurate machine models in the shop-floor; 2) Supporting the creation at runtime of workflows not only among both AI modules and the IoT sensors, but also among themselves; 3) Supporting the plug&play at runtime of the IoT sensors, the AI modules and machine models, without the need to restart the system; 4) Easing the adaptation of the digital twin to the changes in the physical world; 5) Enabling a flexible and distributed deployment: supporting both completely cloud or mixed edge/cloud solutions, based on the need of the specific application.

A first partial implementation of the proposed architecture and a set of possible future works for the part not yet implemented is presented in the next section.

V. Preliminary implementation and future prospects

This section presents the first prototype of the proposed architecture. The solution is a combination of newly developed components and the evolved version of components already developed in previous European Union (EU) projects. For the new components, this section will introduce only some possible technologies that the authors are evaluating and testing so far to leverage and implement the architecture, while for the existing components a more concrete implementation is presented. More specifically, the software already implemented is one algorithm for predictive maintenance, one for fault diagnosis and a distributed simulation environment already developed in CPSwarm, while the components yet to be implemented are the AI environment, the AI engine, the DTO, the Configuration GUI and the OutputMonitor GUI.

The authors have chosen to base the AI environment on a docker container based solution. Each predictive maintenance and fault diagnosis algorithm will be wrapped in one container. In this way the AI environment will support the integration of AI modules based on different technologies.

Two examples of solutions currently supported in the AI environment are the fault diagnosis and predictive maintenance modules presented in Fig. 3 and 4. The fault diagnosis module is composed by techniques to find abnormal behaviors that deviate from normal process conditions to raise warnings and find root causes for the problem. This algorithm will be fed directly with sensor data (when possible and pertinent) or transformed data from the field in order to be more interpretable. Based on the analysis of data streaming, the algorithm should indicate if a warning should be sent to the key personnel to check the system. This algorithm is the first front-line of analysis from shop-floor components in order to understand machine's health.

Additionally, the predictive maintenance module is composed of 1) a component failure prediction in the future (e.g. 48h and which maintenance action should take place); 2) Optimization module for scheduling future maintenance actions based on the existing scheduling; 3) Simulation module that aims at assessing the impact of changes in the machine and shop-floor [17]. The main idea of this method is to predict what kind of maintenance and when it will be required based on the failing component in the machine. With this, it will be possible to understand what changes need to be done in order to compensate the downtime of the failing machine.

As can be seen from Figure 3, the implementation already follows a block based approach which allows a better flexibility once building the required data workflows among models. For this particular case, the Dynamic INtelligent Architecture for Software MOdular REconfiguration (DINASORE) [18] platform was used, which is a run-time environment developed in python language for the International Electrotechnical Commission (IEC) 61499 standard [19] and integrated with the Eclipse based Framework for Distrubeted Industrial Automation and Control (4DIAC) Integrated Development Environment (IDE) [20]. Moreover, this implementation does not only allow for the orchestration of models, but also the plug&play of such models in a distributed system, where software can be reconfigured on-demand. This supports both completely cloud or mixed edge/cloud systems, depending on the required application and the number of machines available for execution. Finally, since DINASORE is implemented in python, the state of the art implementations of AI can be promptly used.

For the implementation of the AI Engine that interconnects the AI modules, the authors have already evaluated several solutions. One is the possibility to run the modules in a docker environment, making them read and write from text files located in specific folders and then interconnect them through a software that allows to handle the workflow, e.g., Node-red or NiFi. Another evaluated solution is the possibility to use Acumos AI to implement the AI environment; in this case, the deployment of the containers, the interconnection of the components and the workflow will be handled by tools included in the framework. At the moment of writing, the final solution to be used is still under evaluation and the authors are investigating if Acumos AI satisfies all the needed requirements of the AI environment, particularly focusing on the possibility to add and remove AI modules at run-time and the dynamic change of their interconnections to create new workflows.

For the implementation of the DTO and the *Simulation Environment*, the *Simulation and Optimization Environment* solutions provided by CPSwarm will be leveraged and extended. Specifically, REPLICA will incorporate the communication API based on XMPP and the deployment system, based on docker and Kubernetes [21]. The use of these technologies will allow to integrate heterogeneous simulators, to simply deploy and run the simulations needed on distributed machines, add and remove at run-time the simulators running different models. More specifically, in the *Simulation Environment*, considering that the solutions proposed in CPSwarm was integrating only ROS based simulators, new types of simulators, e.g., java based simulators, will be included during the RECLAIM project. For this scope, a specific SM will be developed for the required simulators and the API will be refactored to support also these new simulators. Beside the SM, also docker containers to easily deploy such simulators will be created. Instead, for the implementation of the DTO, the authors have defined that the SOO implemented in CPSwarm will be completely refactored and extended to support the functionalities required by REPLICA. Additionally, only some of the functionalities of the SOO will be leveraged, extending them to support data storage and data analysis features. Finally the DTO will provide a set of API based on some standard technologies, e.g., Message Queue Telemetry Transport (MQTT), which will allow to collect data to be used by the algorithms and to keep the models updated and synchronized with the physical world. Thanks to these API, the DTO will be able to collect data from heterogeneous devices that, in the RECLAIM platform, are integrated through the IoT Gateway (see Section II). Also in this case, it will be possible to add and remove devices at run-time, without the need to restart the system. The new devices can be immediately used by the solution developed, just after they have registered themselves.

Finally, for the implementation of the GUI included in the architecture, the presented modules are in different phases of development. Specifically, for the Configuration GUI, the authors have not yet chosen how to implement it and different solutions will be evaluated, keeping in consideration a thorough integration with the rest of the platform. Instead, for the Output/Monitor GUI, for the monitoring and assessment of the results of the algorithms, a simple implementation based on the work done in CPSwarm is already available. Specifically, this solution is based on Thingsboard, which has been used to develop two different GUI: one for process monitoring and another one for the assessment of results. The first one shows live data in a chart and allows the monitoring of the process;

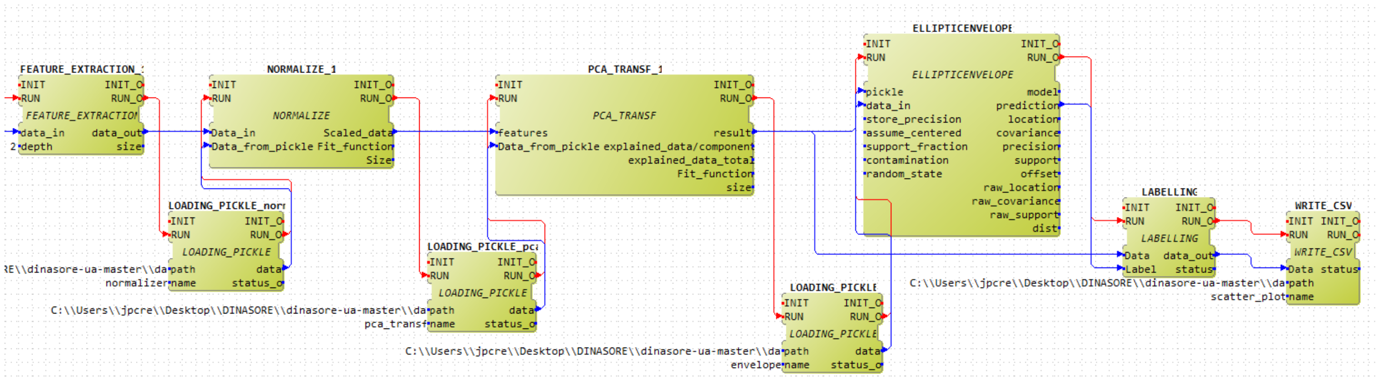


Fig. 3. Fault Diagnosis algorithm.

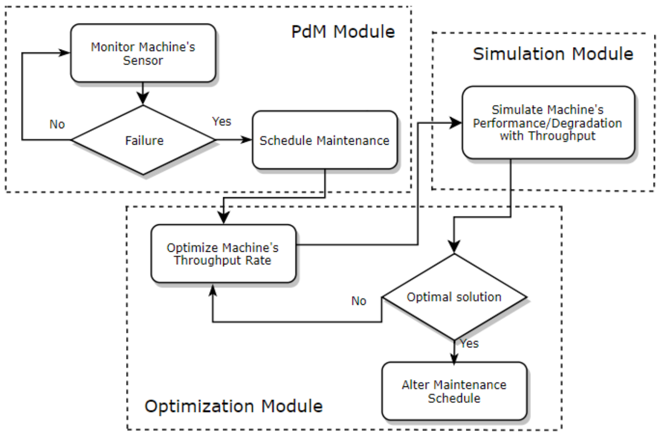


Fig. 4. Predictive Maintenance algorithm [17].

the latter one, instead, shows the data in a table, where it can be sorted by column (one column for each data). These GUIs have been used in this first implementation, but the possibility to enhance or completely replace them with something different, based on the requirements of the solution proposed, will be evaluated in the future.

VI. RECLAIM USE CASE

The aim of this section is to present a possible application of the solution presented in this paper, focusing on the predictive maintenance and refurbishment of a large Woodworking Production Line. The main objective of such use case will be to show the benefits of the adoption of advanced maintenance strategies in a large scale industrial scenario.

The selected scenario presents different challenges: firstly, the need to integrate in a single environment both heterogeneous data collected by installed sensors at the shop-floor and AI modules; together with the realistic models of the machines, enabling and easing the construction of a shop-floor's digital twin. Moreover, the proposed solution will have to optimize the use of a large industrial equipment providing novel machine learning solutions able to monitor the current system status and predict possible failures. In particular, the exploitation of fault diagnosis and predictive maintenance techniques based on the use of digital twin will increase the efficiency of the maintenance activity with respect to the performance obtained with the traditional methods based on a fixed schedule and a simple telemetry analysis.

The flexibility and adaptability of REPLICA can be well demonstrated both during the system setup in the industrial site and in case of replacement of one machine in the Woodworking Production Line.

In the first case, when the platform is going to be deployed, the data collected by the *Adaptive Sensorial Network* and a set of AI algorithms for fault diagnosis and predictive maintenance developed by different analysts have to be integrated. Furthermore, to allow the simulation of different operative scenarios, the realistic models of different machines have to be imported and executed in one simulator.

Using REPLICA, the effort to make all these components provided by heterogeneous vendors working together is significantly reduced, allowing their integration by simply exposing their inputs/outputs through the REPLICA defined interfaces. In particular, for the machine simulation, if the simulator used to execute it is already supported by REPLICA, no further developments are needed; otherwise only the SM for that simulator needs to be developed to allow its integration with the rest of the solution. The same advantage can be considered also for the AI algorithms: if the ones already integrated in the AI environment are suitable for the specific case, no developments are necessary and the platform should be just configured to enable the correct flow of data among different components. Otherwise, to integrate a new algorithms, the only requirement is to implement the inputs/outputs API defined in REPLICA.

Once all the components are connected, the AI modules can be trained using the data coming from the *Adaptive Sensorial Network* (as it is usually done), but also with data produced by the simulated machines. The integration of this secondary source of data not only allows to speed up the training process (more data available means less time to learn) but also to add the possibility of using data that are generally more difficult to collect, such as the one associated with specific failures that, for obvious reasons, are not so common in a real industrial plant.

The advantages of the REPLICA solution can be further demonstrated taking into consideration the scenario of a machines part replacement in the production line which is already monitored by the system. In this case, the administrator of the platform needs to update the components used for the fault diagnosis and predictive maintenance to reflect the new situation on the field. In a traditional system, this is a process that requires a complete shutdown of the system in order to setup and reconfigure it; instead, by using REPLICA the process is fast and mainly automatic. Indeed, for the replacement of the simulation models, this can be done just removing the old simulator and instantiating a new one with the updated model, taking advantage of the integration of the solution with Kubernetes. Once the updated simulator is instantiated, the same process can be applied to the AI algorithms, which can replace the previous ones. All these updates can be done without the need to interrupt the execution of the system. Obviously, if this change requires the integration of some new AI modules or the development of a new SM, these have to be implemented in advance. Also the workflow of the components need to be updated to interconnect the new ones, and can be done simply by using the tools provided by AI engine and DTO. These will automatically update the workflow to reflect the status of the components available in the system and that allow the administrator to easily create the new workflows. Finally as for the previous use-case, REPLICA can also be used to speed up the training time needed to have the new algorithms ready to be used, supporting the use of simulated data, instead of using the actual devices.

VII. Conclusion

The paper has presented REPLICA, a solution that enables the application of innovative fault diagnosis and predictive maintenance techniques based on DT. The software proposed is part of a more complex platform developed by the RECLAIM project that provides a complete software solution for both extending machinery lifetime while also improving productivity and performance.

The paper shows the main key innovations introduced by REPLICA in terms of fault diagnosis and predictive maintenance techniques based on DT. Specifically, one of the basic concepts of REPLICA is to build the solution not as unique suite of technologies, but as a run-time environment with APIs that allow the integration of heterogeneous AI modules and simulators. In this way, the solution can be easily customized to be used in different industrial sites, integrating components provided by different vendors, without requiring the development of new components, but just adapting the existing ones. Furthermore, using REPLICA the user can integrate and replace AI modules and simulation models at run-time, without the necessity to stop the system and reconfigure it. REPLICA provides tools to interconnect among each other the different modules with the data collected from the field. This allows the creation and the modification at run-time of the workflows needed for fault diagnosis and predictive maintenance, adding/removing/replacing entities to reflect the situation of the components available in the system.

In this work, the authors introduced the details of the first implementation of the proposed architecture; since the development currently is only in a preliminary phase, Section V presents the implementations of the modules that were evolved from previous EU projects' outcomes. Instead, for the new components only some initial design choices are presented. In particular, the AI environment and the DTO have been just designed and will be developed in next phases of the project, while the *Simulation Environment* is already available and only some SMs for new simulators will be developed. In the same way, a dashboard for monitoring and results assessment is ready, while the GUI for configuration has still to be implemented.

Finally, the authors have provided in Section VI two use cases based on one realistic industrial scenario, which show the advantages of using the proposed solution to apply fault diagnosis and predictive maintenance techniques based on digital twin.

Acknowledgment

The work presented here was part of the project "RECLAIM- RE-manufaCturing and Refurbishment LArge Industrial equipMent" and received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 869884.

References

[1] R. Rosen, G. [von Wichert], G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567 – 572, 2015, 15th IFAC Symposium onInformation Control Problems inManufacturing.

[2] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, 02 2018.

[3] R. Magargle, L. Johnson, P. Mandloi, P. Davoudabadi, O. Kesarkar, S. Krishnaswamy, J. Batteh, and A. Pitchaikani, "A simulation-based digital twin for model-driven health monitoring and predictive maintenance of an automotive braking system," in *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, 07 2017, pp. 35–46.

[4] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 12 2017.

[5] ——, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 12 2017.

[6] J. Lee, H. Davari, J. Singh, and V. Pandhare, "Industrial artificial intelligence for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 18, pp. 20 – 23, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2213846318301081

[7] B. A. Talkhestani, T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schloegl, and M. Weyrich, "An architecture of an intelligent digital twin in a cyber-physical production system," *at-Automatisierungstechnik*, vol. 67, no. 9, pp. 762–782, 2019.

[8] P. Aivaliotis, K. Georgoulias, Z. Arkouli, and S. Makris, "Methodology for enabling digital twin using advanced physics-based modelling in predictive maintenance," *Procedia CIRP*, vol. 81, pp. 417–422, 2019.

[9] J. Lee, M. Azamfar, J. Singh, and S. Siahpour, "Integration of digital twin and deep learning in cyber-physical systems: towards smart manufacturing," *IET Collaborative Intelligent Manufacturing*, vol. 2, no. 1, pp. 34–36, 2020.

[10] Y. Xu, Y. Sun, X. Liu, and Y. Zheng, "A digital-twin-assisted fault diagnosis using deep transfer learning," *IEEE Access*, vol. PP, pp. 1–1, 01 2019.

[11] K. Ding, F. T. Chan, X. Zhang, G. Zhou, and F. Zhang, "Defining a digital twin-based cyber-physical production system for autonomous manufacturing in smart shop floors," *International Journal of Production Research*, vol. 57, no. 20, pp. 6315–6334, 2019. [Online]. Available: https://doi.org/10.1080/00207543.2019.1566661

[12] G. Zhou, Z. Chao, L. Zi, K. Ding, and C. Wang, "Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing," *International Journal of Production Research*, 04 2019.

[13] M. Borth, J. Verriet, and G. Muller, "Digital twin strategies for sos 4 challenges and 4 architecture setups for digital twins of sos," 05 2019, pp. 164–169.

[14] W. Luo, T. Hu, Y. Ye, C. Zhang, and Y. Wei, "A hybrid predictive maintenance approach for cnc machine tool driven by digital twin," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101974, 10 2020.

[15] M. Jdeed, M. Schranz, A. Bagnato, S. Suleri, G. Prato, D. Conzon, M. Sende, E. Brosse, C. Pastrone, and W. Elmenreich, "The cpswarm technology for designing swarms of cyber-physical systems," in *Proc. Int. Conf. on The Research Project Showcase of Software Technologies: Applications and Foundations (STAF)*, Jul. 2019.

[16] M. Rappaport, D. Conzon, M. Jdeed, M. Schranz, E. Ferrera, and W. Elmenreich, "Distributed simulation for evolutionary design of swarms of cyber-physical systems," in *Proc. Int. Conf. on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE)*. IARIA, Feb. 2018, pp. 60–65, ISBN: 978-1-61208-610-1.

[17] L. Antão, J. Reis, and G. Gonçalves, "Continuous maintenance system for optimal scheduling based on real-time machine monitoring," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 410–417.

[18] Digital and I. I. Lab. Dinasore - dynamic intelligent architecture for software and modular reconfiguration. [Online]. Available: https://digi2-feup.github.io/dinasore/

[19] G. Cengic, O. Ljungkrantz, and K. Akesson, "Formal modeling of function block applications running in iec 61499 execution runtime," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, Sep. 2006, pp. 1269–1276.

[20] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini, and A. Martel, "Framework for distributed industrial automation and control (4diac)," in *2008 6th IEEE International Conference on Industrial Informatics*, July 2008, pp. 283–288.

[21] C. Consortium, "Finalintegration of external simulators, d6.7," Deliverable of the CPSwarm Project, 2020.

DINASORE: A Dynamic Intelligent Reconfiguration Tool for Cyber-Physical Production Systems

Eliseu Pereira, João Reis, Gil Gonçalves

SYSTEC - Research Center for Systems and Tecnologies
Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
Email: {eliseu, jpcreis, gil}@fe.up.pt

***Abstract*—The nowadays industrial digital revolution demands for software driven solutions where reconfiguration is one of the key enablers to achieve smart manufacturing by easy deployment and code reuse. Despite existing several tools and platforms that allow for software reconfiguration at the digital twin / edge level, it is most of the times difficult to make use of state of the art algorithms developed in the most popular programming languages due to software incompatibility. This paper presents a novel framework named *Dynamic INtelligent Architecture for Software MOdular REconfiguration* (DINASORE) that implements the industrial standard IEC 61499 based in Function Blocks (FB) in Python language for Cyber-Physical Production Systems' implementation. It adopts the 4DIAC-IDE as graphical user interface (GUI) to ease the design and deployment of FBs to quickly and on-demand reconfigure target equipment. The proposed framework provides data integration to third party platforms through the use of OPC-UA. The test scenarios demonstrate that the proposed framework 1) is flexible and reliable for different applications and 2) the CPU and memory workload linearly increases for a large amount of FBs.**

***Index Terms*—Cyber-Physical Systems, IEC 61499, Smart Manufacturing, Machine Learning**

I. Introduction

One of the key aspects of the nowadays fourth industrial revolution is the digitization of shop-floor entities like processes, equipment and components to increase their interoperability with users and information systems. In order to achieve digitization, an increased effort of standardization is required to create uniformed interfaces that promote a transparent communication among a set of heterogeneous entities. This standardization is often attained with the concept of digital twin (DT), and is often seen as a wrapper used to integrate any device or process into a network, where information can be easily accessed and shared [1]. In industry 4.0 context, a Cyber-Physical Production System (CPPS) is a distributed system of networked digital twins representing industrial processes, controllers, components, and any sort of information technology (IT) software. These CPPSs should allow for dynamic reconfigurability, software reusability and an external service orchestration [2]. On the one hand, by improving the accessibility via DTs, users can have a better grasp of the holistic shop-floor dynamics through the integration with information systems (vertical integration) such as Manufacturing Execution Systems (MES) or Enterprise Resource Planning (ERP). On the other hand, it is possible to explore new ways of data sharing among shop-floor entities (horizontal integration) promoting a distributed control system for continuous monitoring and process optimization.

With a change in paradigm from closed programmable logic controller (PLC) implementations to industrial PCs that allow a more flexible information sharing and storage, new opportunities taking advantage of this transparency can be explored. From multi-agent systems to artificial intelligence applications, democratizing data storage and sharing is key for the new advances in manufacturing systems, where machine learning is one of the key enablers of industry 4.0. This way, applications like artificial vision to detect production defects, predict when and why a certain component will fail or even explore new energy efficiency solutions are some examples of the well established importance of artificial intelligence in manufacturing.

Albeit not specific for manufacturing applications, there are several platforms developed in order to ease the development of such intelligent systems in a modular fashion, where the main idea is to accelerate the implementation of an end-to-end solution without the need to know the technical details of certain techniques. Some examples of this modular design and execution are Rapidminer Studio [3], Microsoft Azure Machine Learning Studio [4], Cloud AI from Google Cloud [5]. All these platforms have a strong graphical user interface (GUI) based in components that, through drag and drop, a complex machine learning system is possible to be built.

However, the use of such platforms in industrial applications, mainly at the level of control systems, is not straightforward. On the one hand, some of these are Cloud-based solutions, which is still an obstacle for specific industries nowadays due to the industry's policy, restricting the data access only to local network components. On the other hand, these modular designs do not implement any industrial standard, such as IEC 61499 adopted in 2005 and based in the function block (FB) definition that abstracts both software and hardware modules suitable for manufacturing requirements.

Based on this, the present paper proposes a framework called DINASORE for the execution of digital twins in Cyber-Physical Production Systems that is able to support the latest advances in machine learning. DINASORE stands for *Dynamic INtelligent Architecture for Software MOdular REconfiguration* and is compliant with the 4DIAC platform [6] that implements the standard IEC 61499 [7]. The proposed framework is a Python environment for any system (embedded or not) that is able to run Python 3.6, or above, for the execution of function blocks (FBs) developed in Python language. The same way FORTE is used together with 4DIAC for the low level interaction and execution of C/C++ applications in industrial equipment, DINASORE is a similar implementation but for Python FBs, which makes possible the use of important machine learning packages such as TensorFlow, PyTorch, Keras and Theano.

From this perspective, with DINASORE it is possible to develop a distributed control system for industrial applications which is machine learning-enabled. This framework allows to overcome the difficulty to use and develop state of the art algorithms due to the C/C++ (FORTE) requirement. Therefore, all the latest developments in the Python community, from deep learning to optimization, can be used in industrial applications with DINASORE. It is capable of online-reconfiguration of Python FBs, where an easy to understand Python FB template is provided for customized developments. On top of that, and due to industrial requirements, each DINASORE environment has an embedded Open Platform Communications - Unified Architecture (OPC-UA) server with a data model to facilitate the integration with third-party platforms. The OPC-UA data model abstracts the concepts of equipment and device for fully industrial integration with other devices or information systems. In sum, the main contributions and differentiation of this work lie in:

1) A Python implementation of a digital twin that is integrated with 4DIAC and compliant with IEC 61499;
2) A Python template to build FBs compliant with DINASORE;
3) OPC-UA data availability using an OPC-UA server;

The remainder of the present work is organized in five more sections. A literature review is made in Section II, and the DINASORE implementation explained in Section III, from its architecture to the Python template definition to develop new FBs. The following section presents the test case scenarios defined and the main results obtained. Finally, Section V will discuss the results obtained and draw some conclusions and future work for the current implementation.

II. Related Work

One of the most widely used platforms for component-based design and execution of distributed control systems is 4DIAC [6], an Eclispe-based IDE that implements the standard IEC 61499 [7] based in function blocks (FBs). This platform has a vast set of functionalities, from FB design, system design based on a pipeline of FBs, to the deployment of this system in a distributed environment. From CNC applications [8], Distributed Time-Critical Systems (DTCSs) such as aerospace applications [9], to Smart Grids [10], the use of 4DIAC for the implementation of IEC 61499 is becoming well established due to its easiness of system design and execution for distributed environments.

There is already a great motivation in using these approaches, mainly the development of CPPSs using the IEC 61499 standard, as can be seen in literature. A use case that implements such an approach is presented in aluminum cold rolling mill plant demonstrator where the authors design and test an event-driven process based on IEC 61499 standard using OPC-UA [11]. Another similar use case that used the same philosophy is about Oil&Gas production where the authors have implemented a CPPS to deal with the complexity of equipment data on existing production processes, based on IEC 61499 and OPC-UA [12].

In [13] the authors implement a CPPS based in FORTE, and deployed the configuration system into a couple of Raspberry Pi 3 Model B that mimics an industrial process composed of a human-machine interface, a handling system and a conveyor belt. Further, the authors have integrated the OPC-UA into a more complex system [14], with the aforementioned scenario, plus a stack station. The major difference from the proposed framework is the use of FORTE and OPC-UA data model through a Service Interface Function Block (SIFB). In the case of DINASORE there is no SIFB since OPC-UA is embedded in specific FBs and automatically provides data in a OPC-UA server with no effort to the final user.

Regarding the technologies used to support the development of CPPS based on the IEC 61499, there are a set of platforms already available that can be used. Some examples are the Archimedes [15], FBDK [16] and FUBER [17] in Java language (Archimedes also supports C++); FBBeam [18] in Erlang; FORTE and nxtIECRT in C++; ISaGRAF [19] using IEC 61131-3 standard; Icaru-FB [20] and RTFM-RT in C [21]. Demonstrating the relevance of OPC-UA in such approaches, there's a work that presents an implementation of a Service Interface Function Block (SIFB) for OPC-UA communications in 4DIAC [22]. For a more comprehensive understanding of these platforms, there's also a small review about the topic [23].

Some effort has been applied to the integration of UML with the IEC 61499, where a case scenario composed by a distributing and sorting process using FESTO FMS-200 FORTE platform was built resulting in a CPPS [24]. Portability among NxtStudio, FBDK and 4DIAC platforms was also already explored [25] to have a cross-platform implementation of a CPPS, and increase the integration capabilities when building a CPPS.

Regarding all the works previously presented, the DINASORE framework presents an additional, but important, step towards the implementation of the IEC 61499 standard using Python language, and consequently, the use of state of the art machine learning algorithms in CPPSs. Together with the OPC-UA for vertical integration, DINASORE can be seen as a powerful framework that can accelerate and strengthen the next generation of smart industry.

III. Implementation

Similar to FORTE (4DIAC-RTE) portable implementation in C++ of IEC 61499, DINASORE[1] shares the same philosophy but in Python language. With all the latest artificial intelligence advances and current implementations in Python, DINASORE can be seen as a tool that enables advanced machine learning systems to execute as close to shop-floor equipment as possible. The further integration with OPC-UA for sharing a simple and intuitive data model allows for each digital twin DINASORE implementation easy to integrate with most information systems.

A. 4DIAC-IDE

The graphical user interface (GUI) integrated with the DINASORE is the 4DIAC-IDE, which enables drawing and deploying distributed configurations, based in FBs. The 4DIAC-IDE uses the Eclipse Project as a core development framework, providing a GUI based in a desktop application, with the typical Eclipse IDE appearance. The process of development of a new CPPS based in FBs has as main steps: 1) the definition of CPPS network configuration, specifying for each device its IP address and port used for 4DIAC communication, 2) the drawing of the FB pipeline, drag&dropping FBs and linking them through specific connections, modeling the required software architecture, 3) the mapping of specific FB to devices, and 4) the deployment of the FB pipeline to the corresponding DINASORE devices. Therefore, the 4DIAC-IDE GUI has several views directed to the user to implement different steps, e.g. the network configuration, the development and the deployment views. After the development and deployment of the FB pipeline, 4DIAC-IDE enables to monitoring (watch) the whole system for real-time visualization of the current state of each data and event inputs and outputs in the configuration. Additionally, the 4DIAC-IDE allows the interaction with the actual pipeline, triggering events, stopping the configuration, or cleaning the actual DINASORE runtime environment resources.

The standard IEC 61499 defines several rules at the industry level, including the composition and structure of FBs, which is the graphical representation of a set of functionalities. Each FB contains events and variables to communicate with other FBs, where each event allows to trigger the execution of a certain FB, while each variable stores the data (e.g. sensor measurements, algorithm outputs). The 4DIAC-IDE uses FBs as elementary components that connect among themselves, using both events and variables forming a functional pipeline. Considering that, there are three types of FBs defined by IEC 61499, namely:

1) Basic Function Blocks (BFBs): In simplistic terms, they are state machines that according to specific events are able to execute the corresponding algorithms;
2) Composite Function Blocks (CFBs): It is a composition of BFBs making up a network of FBs to model more complex system behaviors;
3) Service Interface Function Blocks (SIFBs): It allows to specify how FBs should interface in terms of both events and data connectors to other FBs that execute (mainly) in different physical platforms (Machine-2-Machine communication).

Additionally, the developer has the autonomy to implement their own FBs and integrate them in both 4DIAC-IDE and DINASORE runtime environment. The main type of FBs adopted in the DINASORE is the Basic Function Block (BFB), characterized by two files, an XML metadata file containing the FB structural information and a Python file implementing the code functionalities.

B. DINASORE

The main goal of DINASORE is to serve a gateway to machine learning into distributed control systems based on IEC 61499. As we believe the future design and deployment of CPPSs will definitely pass through the use of block-based technologies, such as 4DIAC-IDE and IEC 61499 as depicted in the related work, DINASORE can be seen as a key complementary technology to enrich the area of CPPS with artificial intelligence algorithms. Although integrated with 4DIAC-IDE, DINASORE is not designed in its root to execute in embedded systems, complying with real-time constraints as FORTE. The idea behind using Python language is to enable the latest advances in machine learning to integrate at the edge level with existing shop-floor equipment, without the need for cloud based processing. With the embedded implementation of OPC-UA in the DINASORE function blocks (FBs), any external data is automatically provided in a OPC-UA server for further system integration.

As for DINASORE implementation, there's the need to classify the used execution model type framing the technology into the IEC 61499 guidelines. One of the most well known categorization of Execution Control Chart (ECC) execution model was proposed by Ferrarini in [26], where 7 different classes were defined, from A0 to A6, exploring two dimensions, namely scan order and multitasking. The scan order refers to execution models that can have either fixed (predefined order), or dynamically (order calculated on-the-fly) FB execution during the ECC, while the multitasking dimension refers to no controlled ways of multitasking, where FBs execute in a multi-threading fashion; done by time slice allocation to each FB execution (preemptive scheduling) and done by FB slice, where each FB executes at a time (non-preemptive scheduling).

Additionally, there's a small and brief survey of run-time environment (RTE) platforms for IEC 61499 where 4 types of execution models are introduced [23]: 1) Buffered Sequential Execution Model (BSEM) [27]; 2) Cyclic Buffered Execution Model (CBEM) [16]; [26]; 3) Non-Preemptive Multithreaded Resource (NPMTR) [28]; 4) Preemptive Multithreaded Resource (PMTR) [23]. For a more formal definition of the BSEM, CBEM and NPMTR please refer to [29]. Despite the authors in the survey present a table that integrates RTE platforms with Ferrarini model and execution models, there's not

[1]Available Online: github.com/DIGI2-FEUP/dinasore

a one-to-one correspondence of Ferrarini model to execution models. This happens because there are some categories in the Ferrarini model without a corresponding execution model. This way, it is important to first fill this gap in terms of formal definition, and only then use it to frame the DINASORE. Hence, the complete set of Ferrarini categories is presented in the following list, with the corresponding execution models:

- A0: The execution of each FB is calculated on-the-fly depending on the input events of each one. One formalized execution model that meets this category is the BSEM;
- A1: The execution of each FB as a thread-object is made in parallel, like in a multi-threading fashion. We name this execution model as Multithreaded Resource (MTR);
- A2: To each of the executing FBs as a thread-object is given a small time slice of execution, where the allocation of time slice to FB is dynamically done. The most similar execution model is PMTR, briefly defined in [23]. However, due a dynamic scan order, we name this execution model as Buffered Sequential-Preemptive Multithreaded Resource (BS-PMTR);
- A3: Each FB as a thread-object should execute one at a time, and executed dynamically as soon as a notification is created. One formalized execution model that meets this category is the NPMTR. However, due a dynamic scan order, we name this execution model as Buffered Sequential-Non-Preemptive Multithreaded Resource (BS-NPMTR);
- A4: The execution of each FB is predefined beforehand. One formalized execution model that meets this category is CBEM;
- A5: To each of the active (that requires execution due to an event input) FBs as a thread-object is given a small time slice of execution according to a fixed order that follows a list or active FBs. This can be viewed as a PMTR, however, due to fixed scan order, we name this execution model as Cyclic Buffered-Preemptive Multithreaded Resource (CB-PMTR);
- A6: Each FB as a thread-object should execute one at a time according to a fixed order that follows a list or active FBs. This can be viewed as a special case of the NPMTR, however, due to fixed scan order, we name this execution model as Cyclic Buffered-Non-Preemptive Multithreaded Resource (CB-NPMTR);

TABLE I
AGGREGATION OF FERRARINI MODEL [26] WITH EXISTING [23] AND NEW EXECUTION MODELS FOR IEC 61499.

	Multitasking Implementation			
	Not Used	**Not Controlled**	**Time Slice**	**FB Slice**
Dynamic Order	BSEM (A0)	MTR (A1)	BS-PMTR (A2)	BS-NPMTR (A3)
Fixed Order	CBEM (A4)	x	CB-PMTR (A5)	CB-NPMTR (A6)

As for the DINASORE, the closest category is A2, where we have a thread-object per FB with all threads executing using a time slice scheduling and the execution order for all FBs is dynamically calculated by the received event inputs. Since Python language is used together with *threading* package, once we have multiple FBs thread-objects executing at the same time, a time slice scheduling strategy is used. This package uses an implementation called Global Interpreter Lock (GIL) that manages the execution time per thread, being around 5ms. This way, this implementation is not truly multithreaded (A1), but A2 using a similar approach as BS-PMTR.

The DINASORE execution model implementation, Figure 1, uses a producer-consumer pattern, where each FB performs in a different thread, which is both producer and consumer. The data transmitted between FBs addresses both events and variables, where the FB object stores the input events in a queue and the variables in register attributes. Thus, each FB object has an internal queue for the input events, waiting until it receives an input event in the data structure, reading after the variables' actual value, and processing the event's functionality. After completing the functionality execution, the FB object pushes the corresponding output events in the queue of the following connected FBs. The same thing happens to the variables where the FB updates their output variables and consequentially the input variables of the following linked FBs. The actual value of each FB event and variable is available through the monitoring/communication interfaces, i.e., using the OPC-UA server or 4DIAC-IDE *watch* option.

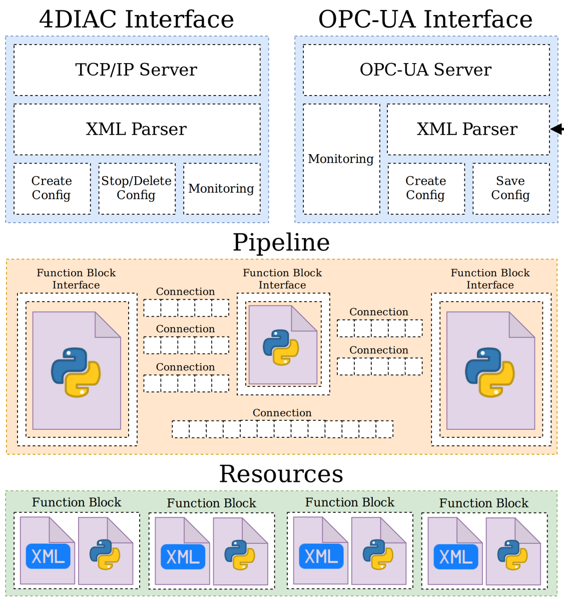


Fig. 1. DINASORE Architecture.

The FB thread-object requires two external files to execute that compose the FB itself. The first file is XML-based and contains the meta-information about the FB, e.g. the FB type, the FB class and the FB structure composed by input/output events and variables, and their details, including the data type and the OPC-UA role (variable, method or none). Besides the DINASORE usage, the metadata file enables the 4DIAC-IDE to render the FB with their events and variables in the GUI. The second file composing the FB is a Python script encoding the FB functionalities, which requires the implementation of a class, assuming the Object-Oriented (OO) paradigm. That class requires the implementation of the *schedule* method, which receives as arguments the event name and respective value, triggering the FB execution, with the current input variable values. Then, according to the event received, the method selects the functionality to execute. After executing the functionality, the *schedule* method returns a list of output events and variables. Both *schedule* method arguments and output variables should follow the order specified in the metadata file.

Concerning the need for external communication between the DINASORE and other applications, e.g. 4DIAC-IDE and third party information systems, there are two different and independent communication interfaces integrated in the DINASORE, 1) the 4DIAC interface, which uses TCP/IP, and 2) the OPC-UA interface, which uses a data model XML file as a reconfiguration file. These two interfaces allow the reconfiguration of the current runtime workflows and the monitoring of each FB. The 4DIAC interface communicates with the 4DIAC-IDE using a TCP/IP connection, where the DINASORE interface executes a TCP server, receiving the commands from the 4DIAC-IDE enabling the creation, stop, and deletion of the configuration workflow and the runtime configuration monitorization. The interaction between 4DIAC-IDE and DINASORE for the configuration creation starts with several messages instantiating every FB (*Create FB*); each message contains the FB type and instance name. Then the 4DIAC-IDE sends the commands to create the connections (events and variables) between FBs (*Create Connection* and *Write Connection*), and finally the IDE requests the start of the FB threads (*Start Configuration*). After starting the workflow, the DINASORE enables the monitoring of each FB variable and event, through the *watch* option (the *Create Watch* message to activate the subscription; the *Read Watch* message to request the variable/event current value; and the *Delete Watch* message to unsubscribe). Additionally, the DINASORE allows from the 4DIAC-IDE to stop and reset (*Stop Configuration*) the configuration workflow, terminating the respective working threads and the remote trigger of an event (*Trigger Event*), performing a FB functionality in the DINASORE.

As an alternative, the DINASORE can use the OPC-UA Data Model as reconfiguration channel. This way, the DINASORE stores the configuration locally in the Data Model XML file, where registers all the used FBs, and their respective connections (events and variables). The FB XML meta-information classifies each FB in different sets, grouping them in 1) devices, 2) equipment, 3) services, 4) endpoints, and 5) start-points:

1) The device abstraction represents sensors, like sensors integrated using Modbus protocol;
2) The equipment representation uses a more complex structure allowing the aggregation of sensors and actuators, using events as relational connections;
3) The service type maps to FBs as a method and when an event is received, its execution is triggered. This is suitable for machine learning algorithms (e.g., Random Forest, SVM) or statistical operations (e.g., moving average, moving standard deviation);
4) The start-point type represents protocols interfaces that receive data, e.g. subscribing a MQTT topic or requesting TCP/IP data;
5) The endpoint type provides data through the defined interface, e.g. publishing in a MQTT topic and replying to a TCP/IP request.

Those representations make the development of new FBs easier by providing templates and specific behaviors to maximize the effectiveness of the Python code written in each FB. The (1) device, (2) equipment, and (4) start-point scripts adopt a loop template, which enables a cyclic execution of the FB. The (3) service and (5) endpoint types use the typical asynchronous approach executing when triggered.

Both OPC-UA and TPC/IP (4DIAC-IDE) interfaces are essential to DINASORE operation, enabling different features and capabilities. The 4DIAC communication interface allows the combination with a graphical user interface (GUI) for the development of workflows based on FBs. The OPC-UA Data Model, besides enabling the workflow reconfiguration, the primary purpose is to monitor the execution of a specific pipeline monitoring using the OPC-UA industrial protocol and the storage of the device configuration in an XML format. The configuration storage transforms the DINASORE in a more reliable and fault-tolerant platform, enabling the memorization of the current state of the workflow, being tolerant of power cuts restarting the runtime environment in the previous state. The Data Model XML file updates every time, when the DINASORE receives commands from the 4DIAC-IDE, for the creation, stop, and removal of configuration workflows.

IV. Test Case Scenarios

The DINASORE evaluation focuses on the validation of the tool in different scenarios. Those scenarios point out the platform advantages and disadvantages, besides establishing the most suitable applications, like sensing, control, or data processing. The first scenario focuses on the use of machine learning (ML) algorithms, in particular classification methods, for the detection of collisions in a real mini-robotic arm based in servo motors. The second example consists of a typical distributed control architecture composed of two components (gripper and robotic arm) that have to synchronize between them to perform the required operation. The third case shows the simulation of a manufacturing production line. Based on the simulation scenario, several experiments with increasing amounts of FBs allow to assess the scalability of the framework in terms of processing and memory usage.

A. *Collision detection based on servo motors analysis*

The main goal of the collision detection implementation is to transform a typical robotic arm into a collaborative one. That scenario uses a robotic arm based in servo motors that provide load metrics able to infer about possible collisions with obstacles. Based on this, the process 1) monitors each servo motor, 2) detects when one servo motor is in overload, and 3) stops the robotic arm if it collides with an obstacle. The network architecture contains two devices, both Raspberry's Pi: 1) responsible for monitoring the servo motors and check the motors overload (Figure 2 rose FBs), and 2) controlling the robotic arm, sending instructions to perform a particular task, and waiting to receive an overload alert to stop its execution (Figure 2 green FB).

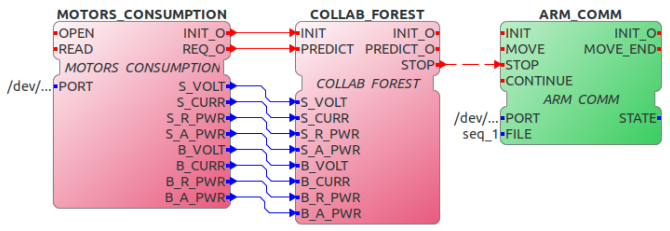


Fig. 2. Collaborative Robot Workflow.

The servo motors sensing component uses as hardware an Arduino Uno board (ATMega328P microcontroller), which collects data of the voltage and current for each motor, using a potential divider to measure the voltage and an amplifier to obtain the current. The Root Mean Squared (RMS) uses the last 250 samples to calculate the voltage and current RMS value, which allows the computation of the real and apparent power. This data processing is performed at the microcontroller level, transmitting the eight features (4 metrics of 2 servo motors) through the serial port to the Raspberry Pi, which uses a FB, implementing the device template, to read and parse the data. After parsing the data and calculating the respective lag features (moving average and standard deviation) for the last 10 samples, the overall features (total of 24 variables) feed a classification method. The model predicts if the robotic arm is performing with regular efforts (output at zero) or in overload/collision situations (output at one). The classification model training uses an offline dataset, collected from the serial port, containing the robotic arm performing different operations with and without collisions. Several classification algorithms, including Support Vector Machine (SVM), Random Forrest (RF), and Artificial Neural Networks (ANN), were validated using the precision, the recall, and the f1-score as performance metrics. The more accurate model was the RF, exported to perform online on the second pipeline FB, that implements the service template. The model output predictions generate a stop event sent, through multicast sockets, to the second Raspberry Pi, which controls the robotic arm. The communication between the two Raspberry Pis, due to the usage of UDP multicast sockets, adopts a paradigm producer-subscriber, where the collision detection component produces stop events when in overload, sent to the robotic arm control component. The robotic arm control component (green FB, using the device template), in a different Raspberry Pi, reads from a text file, the sequence of motor positions to perform, and sends them sequentially through the serial port to the embedded controller. The robotic arm performs, in a cycle, the list of instructions until it receives a stop event from the monitoring component; then, it freezes, waiting to receive a *continue* event from the user, using the 4DIAC-IDE, to restart the operation.

B. *UR5 Collaborative Robotic Arm and Gripper Control*

The synchronized control between a Universal Robots 5 (UR5) robotic arm and a 3D printed gripper requires the usage of a CPPS composed of two Raspberry Pi, similar to the previous scenario. Considering that architecture, one component controls the servo motor that opens and closes the gripper (Figure 3 blue FBs), and the other sends commands to the UR5 robotic arm (Figure 3 purple FBs). The operation performed by both devices consists in 1) catch one object at position A, 2) go with the piece to position B, 3) go back to position A, 4) leave the object in position A, 5) go without the item to position B, 6) go without the object to position A, and restarts the cycle.

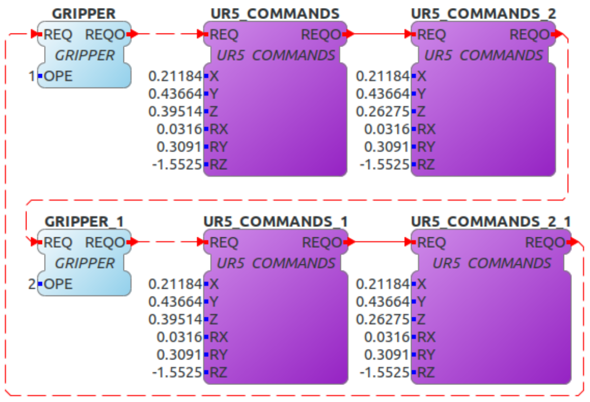


Fig. 3. UR5 Robotic Arm and Gripper Workflow.

The UR5 robotic arm uses an API to send commands through a TCP/IP communication channel with the physical controller, which enables the continuous flow of instructions (X, Y, Z positions and Rx, Ry, Rz rotations) to the UR5 robotic arm. The FB, using the service template, wraps the API function to move the robotic arm to a specific location (X, Y, and Z) with the rotation of the joints (Rx, Ry, and Rz). The gripper developed to pick up objects contains three different parts 1) a Raspberry Pi, 2) a servo motor that opens and closes the gripper, and 3) a 3D printed Polylactic Acid (PLA) casing. The movement of the servo motor opens and closes the 3D printed fingers, according to the instruction sent from the Raspberry Pi interface using the General Purpose Input/Output (GPIO). Thus, a FB, adopting the service template, encapsulates the control of GPIO pins using an input variable to establish the operation to perform (open or close). The communication in the CPPS uses multicast sockets to send events between FBs that are in different devices (different colors), as used in the previous scenario.

C. Manufacturing Applications

The simulation of a manufacturing production line addresses several challenges, like the material tracking on the shop-floor or equipment sensorization. Regarding the flow of materials along the production line, the developed simulation includes each station of the process, with its attributes, like the current manufactured material and the respective transporter, the sensors associated, and the operation time. The simulated representation mainly serves as an advantage for the process industries, which have a sequential set of productive steps, being able to simulate different layouts to optimize productivity.

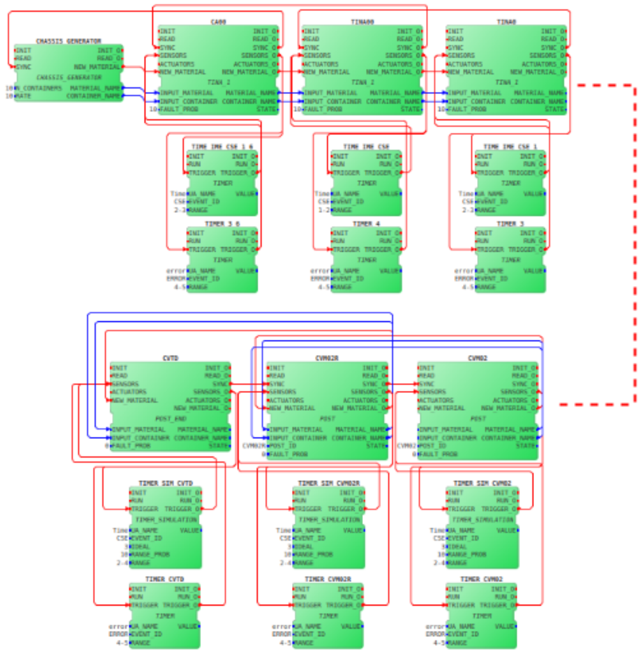


Fig. 4. Manufacturing Scenario.

Figure 4 presents partially the FB pipeline of the simulation implementation, where each simulated station uses three FBs to model its behavior. The main FB, that adopts the equipment template, represents a station and implements a state machine with the following states: 1) *unscheduled*, waiting for new material to produce; 2) *standby*, arrives a material and transporter associated and waits to start the operation; 3) *productive*, working in the operation of the station; and 4) *error*, a stochastic state to simulate an extra time producing the material. The additional two FBs, implemented using the service template, allow to simulate the production time and error time (if any). All the series of three FBs represent the entire production line with its characteristic sequential U shape and bifurcations. Additionally, this approach enables the integration of real components with in this simulation environment, turning the pipeline more accurate.

D. Performance Evaluation

The manufacturing line simulation scenario serves as a basis for performance and workload evaluation for the DINASORE framework. By using a series of three FBs that represent a station it is possible to assess the CPU and memory usage with a varying number of FBs. The *htop* monitoring tool (linux) was used enabling the profiling of each process in the operating system. Typically, the memory usage has a constant value for the same input parameters; however, the CPU usage has high variability, which requires more samples to obtain an accurate estimation. The higher number of collected samples for the same scenario, generates a more substantial variability, computed in the form of standard deviation. Figure 5 considers both metrics, representing the collected samples for the memory and the average and standard deviation for the processing usage.

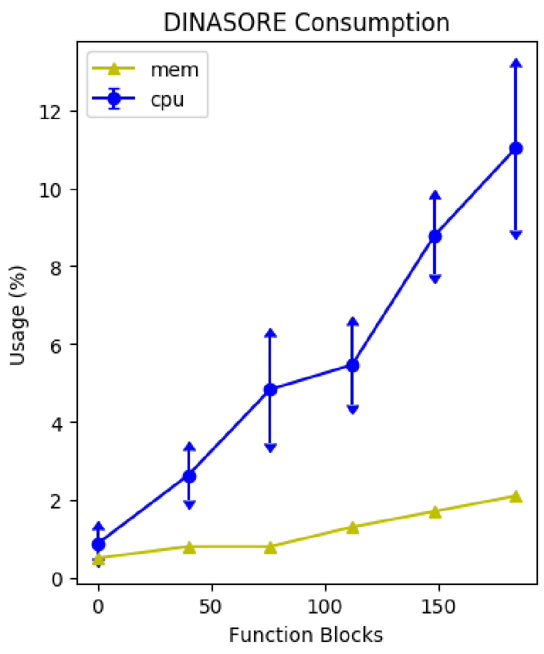


Fig. 5. DINASORE Performance Results.

The main property to evaluate on the DINASORE is how an increasing number of FBs influences the framework usage of computational resources. This kind of test requires a large number of FBs (up to 200) focusing on the DINASORE management of the runtime environment. The hardware specifications of the host machine are 16GB of RAM and an Intel Core i7 processor, with 12 cores and a frequency of 2.20GHz. The analysis of the results indicates a growing trend in both metrics with an increasing number of FBs. That trend follows the natural behavior of computational systems: more complexity causes more resources consumed, with the CPU curve following a rate of approximately 18 FBs per 1% of CPU usage. Those values prove the reliability and scalability of the developed solutions, considering the target hardware, which varies from low computational power devices (e.g., Raspberry

Pi) to high performance machines (e.g., servers). Such experiments' main intention is solely to validate the performance of DINASORE, not considering CPU and memory usage in terms of FB functionality (e.g., training a deep neural network).

V. Conclusion

Looking in perspective, the DINASORE framework enables the deployment of powerful Python algorithms for CPPSs, following industrial standards globally adopted, like the IEC 61499 and OPC-UA protocol. The proposed framework increases the flexibility of the traditionally closed and hard to re-configure industrial systems, being a step forward the high-mix low-volume paradigm. The validation scenarios prove the flexibility and reliability of the DINASORE, giving the necessary freedom to developers for a multitude of different application implementations, like sensor integration, equipment control, data processing, or communication protocols. The performance evaluation demonstrates the scalability in a DINASORE runtime environment with an increasing amount of FBs. Nevertheless, the DINASORE transforms a heavy-weight local application into a distributed solution performing in a clusters of devices. Considering that, the platform provides capabilities to scale up the solutions for a larger amount of devices, isolating the applications into FBs, and communication through popular IoT protocols, like MQTT or OPC-UA.

As for the future work, and assuming that DINASORE enables the use of computation-intensive machine learning solutions, the exploration of speculative computation concept, as implemented in [30] for FB-based systems, will be explored. The central idea is to implement this concept as a background process in DINASORE so the execution can be accelerated transparently. An additional objective is the continuous implementation of new methods using the FB structure to easily experiment with them in different industrial applications due to the FB portability between systems.

Acknowledgment

INDTECH 4.0 - New technologies for intelligent manufacturing. Support on behalf of IS for Technological Research and Development (SI à Investigação e Desenvolvimento Tecnológico). POCI-01-0247-FEDER-026653

References

[1] G. Gonçalves, J. Reis, R. Pinto, M. Alves, and J. Correia, "A step forward on intelligent factories: A smart sensor-oriented approach," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–8.

[2] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of industrie 4.0: an outlook," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, p. 3159805, 2016.

[3] O. Rittho, R. Klinkenberg, S. Fischer, I. Mierswa, and S. Felske, "Yale: Yet another learning environment," in *LLWA 01-Tagungsband der GI-Workshop-Woche Lernen-Lehren-Wissen-Adaptivität*, no. 763. Citeseer, 2001, pp. 84–92.

[4] M. Corporation. (2018) Microsoft azure machine learning studio. [Online]. Available: https://azure.microsoft.com/pt-pt/services/machine-learning-studio/

[5] G. AI. (2017) Google cloud ai. [Online]. Available: https://cloud.google.com/products/ai/

[6] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sunder, A. Valentini, and A. Martel, "Framework for distributed industrial automation and control (4diac)," in *2008 6th IEEE International Conference on Industrial Informatics*, July 2008, pp. 283–288.

[7] G. Cengic, O. Ljungkrantz, and K. Akesson, "Formal modeling of function block applications running in iec 61499 execution runtime," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, Sep. 2006, pp. 1269–1276.

[8] M. Minhat, V. Vyatkin, X. Xu, S. Wong, and Z. Al-Bayaa, "A novel open cnc architecture based on step-nc data model and iec 61499 function blocks," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 560–569, 2009.

[9] C. C. Insaurralde, "Modeling standard for distributed control systems: Iec 61499 from industrial automation to aerospace," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE, 2016, pp. 1–8.

[10] F. Andrén, T. Strasser, and W. Kastner, "Model-driven engineering applied to smart grid automation using iec 61850 and iec 61499," in *2014 Power Systems Computation Conference*. IEEE, 2014, pp. 1–7.

[11] T. Terzimehic, M. Wenger, A. Zoitl, A. Bayha, K. Becker, T. Müller, and H. Schauerte, "Towards an industry 4.0 compliant control software architecture using iec 61499 & opc ua," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2017, pp. 1–4.

[12] M. V. García, E. Irisarri, F. Pérez, M. Marcos, and E. Estévez, "Engineering tool to develop cpps based on iec-61499 and opc ua for oil&gas process," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2017, pp. 1–9.

[13] M. V. García, F. Pérez, I. Calvo, and G. Morán, "Building industrial cps with the iec 61499 standard on low-cost hardware platforms," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–4.

[14] M. V. García, F. Pérez, I. Calvo, and G. Moran, "Developing cpps within iec-61499 based on low cost devices," in *2015 IEEE World Conference on Factory Communication Systems (WFCS)*. IEEE, 2015, pp. 1–4.

[15] K. Thramboulidis and A. Zoupas, "Real-time java in control and automation: a model driven development approach," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1. IEEE, 2005, pp. 8–pp.

[16] V. Vyatkin and J. Chouinard, "On comparisons of the isagraf implementation of iec 61499 with fbdk and other implementations," in *2008 6th IEEE International Conference on Industrial Informatics*. IEEE, 2008, pp. 289–294.

[17] G. Cengic, O. Ljungkrantz, and K. Akesson, "Formal modeling of function block applications running in iec 61499 execution runtime," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*. IEEE, 2006, pp. 1269–1276.

[18] L. Prenzel and J. Provost, "Fbbeam: An erlang-based iec˜ 61499 implementation," in *IEEE International Conference on Industrial Informatics (INDIN'19)*, 2019.

[19] J. H. Christensen, T. Strasser, A. Valentini, V. Vyatkin, A. Zoitl, J. Chouinard, H. Mayer, and A. Kopitar, "The iec 61499 function block standard: Software tools and runtime platforms," *ISA Automation Week*, vol. 2012, 2012.

[20] L. I. Pinto, C. D. Vasconcellos, R. S. U. Rosso, and G. H. Negri, "Icaru-fb: An iec 61499 compliant multiplatform software infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 3, pp. 1074–1083, 2016.

[21] P. Lindgren, M. Lindner, A. Lindner, D. Pereira, and L. M. Pinho, "Rtfm-core: Language and implementation," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2015, pp. 990–995.

[22] S. Kožár and P. Kadera, "Integration of iec 61499 with opc ua," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–7.

[23] L. Prenzel, A. Zoitl, and J. Provost, "Iec 61499 runtime environments: A state of the art comparison," in *17th International Conference on Computer Aided Systems Theory (EUROCAST 2019)*, 2019.

[24] E. X. Castellanos, C. A. Garcia, C. Rosero, C. Sanchez, and M. V. Garcia, "Enabling an automation architecture of cpps based on uml combined with iec-61499," in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2017, pp. 471–476.

[25] A. Hopsu, U. D. Atmojo, and V. Vyatkin, "On portability of iec 61499 compliant structures and systems," in *2019 IEEE 28th International*

Symposium on Industrial Electronics (ISIE). IEEE, 2019, pp. 1306–1311.

[26] L. Ferrarini and C. Veber, "Implementation approaches for the execution model of iec 61499 applications," in *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004*. IEEE, 2004, pp. 612–617.

[27] G. Cengic and K. Akesson, "Definition of the execution model used in the fuber iec 61499 runtime environment.[in:] industrial informatics, 2008. indin 2008," in *6th IEEE International Conference on*, 2008, p. 301.

[28] C. Sunder, A. Zoitl, J. H. Christensen, V. Vyatkin, R. W. Brennan, A. Valentini, L. Ferrarini, T. Strasser, J. L. Martinez-Lastra, and F. Auinger, "Usability and interoperability of iec 61499 based distributed automation systems," in *2006 4th IEEE International Conference on Industrial Informatics*. IEEE, 2006, pp. 31–37.

[29] G. Cengic and K. Akesson, "On formal analysis of iec 61499 applications, part b: Execution semantics," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 2, pp. 145–154, 2010.

[30] D. Drozdov, V. Dubinin, and V. Vyatkin, "Speculative computation in iec 61499 function blocks execution—modeling and simulation," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. IEEE, 2016, pp. 748–755.

Bringing Clouds Down to Earth: Modeling Arrowhead Deployments via Eclipse Vorto

Géza Kulcsár
IncQuery Labs Ltd.
Budapest, Hungary
geza.kulcsar@incquerylabs.com

Sven Erik Jeroschewski, Kevin Olotu, Johannes Kristan
Bosch.IO GmbH
Berlin, Germany
{name.surname}@bosch.io

***Abstract*—The design and development of interconnected industrial production facilities, which integrate aspects of the Internet of Things (IoT) or, more specifically, the Industrial IoT (IIoT), often deals with complex scenarios involving dynamic System of Systems (SoS), resulting in immense development and deployment efforts. The Arrowhead community aims at delivering mechanisms and technologies to cope with such complex scenarios. In particular, the concept of local clouds constitutes a service-oriented architecture (SOA) framework for IIoT. Here, a central challenge is the conceptual modeling of such use-cases. SysML is widely established as a standardized modeling language and framework for large-scale systems engineering and, thus, for Arrowhead local cloud designs. However, SysML and its Arrowhead profile lack a canonical way to support actual platform modeling and device involvement in heavily distributed IIoT scenarios. The Eclipse Vorto project is ideal for filling this gap: it provides a modeling language for IoT devices, a set of modeling tools, and already existing reusable templates of device models. In this paper, we propose an approach to integrating Eclipse Vorto models into Arrowhead SysML models. We illustrate the concept with a realistic yet comprehensible industrial scenario and also present a prototype to emphasize the benefits of our novel integration platform.**

***Index Terms*—System Modeling, SysML, Eclipse Vorto, Eclipse Arrowhead, IoT, IIoT**

I. Introduction

Many IoT and, especially, *industrial* IoT (IIoT) scenarios introduce high complexity in all phases of their life cycle. Reasons for this are, among others, the use of multiple hardware and software platforms or heterogeneous protocols and data formats. With the recent trends of the increasing volume and complexity of such scenarios, it becomes more difficult and more expensive to model, operate, and manage such complex Systems of Systems (SoS) [1]. The *Arrowhead* initiative aims at overcoming these issues using a holistic, comprehensive methodology and mindset. One of the central facets of Arrowhead, also being highly relevant for the present paper, is the application of the concepts of Service-Oriented Architectures (SOA). In turn, Arrowhead introduces *local clouds* for service-providing and service-consuming system resources that can be grouped logically or geographically.

In turn, a novel kind of architectural and design challenges arises in this context of SoS design combined with dynamic, service-oriented orchestration principles, calling for new methods to cope with them. However, established techniques within *model-based systems engineering* (MBSE) [2] serve as a convenient baseline for introducing a new level of dynamicity for system (of system) architectures. MBSE arguably provides a great compromise between domain-specific expectations and rigorous design on the one hand, and a flexible, accessible modeling approach on the other hand. Besides, SysML is an excellent base for formulating and validating the well-formedness of complex systems.

Consequently, the Arrowhead approach to IIoT modeling relies on SysML for specifying local cloud architectures. A major challenge of such a modeling scenario is to integrate the abstract architecture models created in design-time with the actual IoT deployments, more precisely, with their digital twin representations. Naturally, the device-specific and deployment-specific details of these representations are out of scope in abstract SoS (local cloud) models. The very metaphor in the title of the paper, *bringing clouds down to earth*, unites two key points of this conceptual hiatus: (1) while those platform- and hardware-independent local cloud plans lack a connection to their future embodiment, and thus, still have to be brought down to earth, (2) such a device-oriented addition allows the expression of those real communication channels which have to be established between the systems taking part in a given cloud architecture.

As for existing approaches to IoT deployment modeling, VORTOLANG from the Eclipse Vorto[1] project is one of the most relevant and prevalent examples. Other benefits of Eclipse Vorto are that it allows the reuse of existing models through shared repositories and provides plugins for code generation to integrate with other projects and platforms. As a consequence, it seems promising to use Eclipse Vorto to ease the modeling of local clouds by mapping those resources into SysML models. In this paper, we investigate this approach further. In particular, the main contribution of this work, with potential industrial relevance, is an integration concept for coping with the design complexity of industry-scale IoT installations through an integrated modeling approach. As an additional benefit, a baseline arises for a bilateral cross-fertilization: we extend Eclipse Vorto towards functional and architectural design, and systems modeling towards detailed device platform specifications.

Therefore, we first introduce the Eclipse Vorto project and

[1]https://www.eclipse.org/vorto/

its VORTOLANG in Sect. II and give an example use case in Sect. III to which we later apply our modeling approach. Sect. IV introduces the Arrowhead Framework while Sect. V presents the current model-based system engineering process in the Arrowhead Framework with SysML. In Sect. VI, we explain our approach for a mapping between SysML and Eclipse Vorto. This mapping is then illustrated in Sect. VII.

II. ECLIPSE VORTO

Eclipse Vorto [3] is an open-source project for modeling digital twins. The goal of the project is to provide platform-independent and vendor-neutral tools to model digital twins and to make use of the models by supplying plugins to ease the integration in existing IoT ecosystems. The project consists of 4 main components:

- VORTOLANG: a domain specific language (DSL) to describe digital twins
- Repository: a repository to create, manage and distribute models
- Plugins: transform Vorto models into source code, request templates or other representations
- Telemetry Payload Mapping: maps the telemetry data sent by a device using a mapping specification based on a Vorto model

A. VORTOLANG - *The Vorto Language*

VORTOLANG is the domain specific language used to describe digital twins. It consists of four different kinds of elements:

- Information Model (IM): describes a digital twin and its capabilities
- Function Block (FB): describes a set of capabilities that are implemented by the digital twin. Function Blocks can be designed hierarchically by extending other Function Blocks. The individual capabilities are grouped into the following property groups:
  - Status: contains properties of the digital twin that can only be read (read-only)
  - Configuration: contains properties of the digital twin that can be both read and set (read-write)
  - Event: contains events that can be emitted by the digital twin
  - Operation: contains functions that can be invoked on the digital twin
  - Fault: contains fault states that can occur on the digital twin
- Data Type (DT): describe complex data types or enumerations that can be assigned to Function Block properties
- Mapping: describes platform-specific or implementation-specific information that can be added to a generic Information Model or Function Block.

B. *Vorto Repository*

The committers of Eclipse Vorto host an official public instance of the repository[2]. The official repository is an

[2]https://vorto.eclipse.org

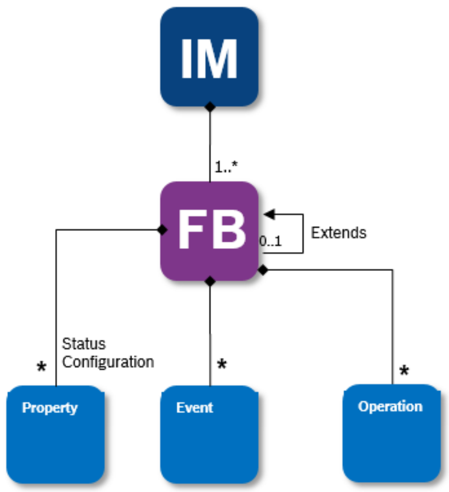
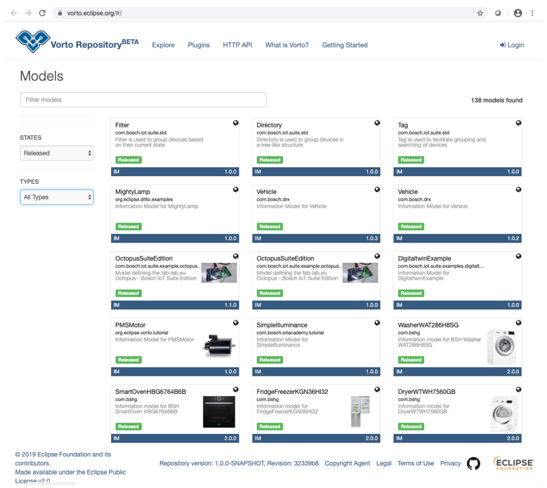


Fig. 1. A simplified model of VORTOLANG

offering for device manufacturers and IoT solution developers to develop and publish re-usable models of their devices / digital twins in a standardized way. However, it is also possible to self-host a Vorto repository (e.g. for on-premise solutions without internet access). The repository offers several features to interact with the Vorto models:

- UI and APIs to interact with the repository and the models
- A web editor to create and edit models
- A review and release process for models
- Different levels of visibility (private / public)
- Import and export functionality of models
- Direct integration with Vorto plugins
- Java client that can interact directly with the APIs of the repository



Models

138 models found

Filter
com.bosch.iot.suite.std
Filter is used to group devices based on their current state
Released
IM 1.0.0

Directory
com.bosch.iot.suite.std
Directory is used to group devices in a tree-like structure
Released
IM 1.0.0

Tag
com.bosch.iot.suite.std
Tag is used to facilitate grouping and searching of devices
Released
IM 1.0.0

MightyLamp
org.eclipse.ditto.examples
Information Model for MightyLamp
Released
IM 1.0.0

Vehicle
com.bosch.drx
Information Model for Vehicle
Released
IM 1.0.3

Vehicle
com.bosch.drx
Information Model for Vehicle
Released
IM 1.0.2

OctopusSuiteEdition
com.bosch.iot.suite.example.octopus..
Model defining the fab-lab.eu Octopus - Bosch IoT Suite Edition
Released
IM 1.1.0

OctopusSuiteEdition
com.bosch.iot.suite.example.octopus..
Model defining the fab-lab.eu Octopus - Bosch IoT Suite Edition
Released
IM 1.0.0

DigitaltwinExample
com.bosch.iot.suite.examples.digitalt...
Information Model for DigitaltwinExample
Released
IM 1.0.0

PMSMotor
org.eclipse.vorto.tutorial
Information Model for PMSMotor
Released
IM 1.0.0

SimpleIlluminance
com.bosch.iotacademy.tutorial
Information Model for SimpleIlluminance
Released
IM 1.0.0

WasherWAT286H8SG
com.bshg
Information model for BSH Washer WAT286H8SG
Released
IM 2.0.0

SmartOvenHBG6764B6B
com.bshg
Information model for BSH SmartOven HBG6764B6B
Released
IM 2.0.0

FridgeFreezerKGN36HI32
com.bshg
Information model for FridgeFreezerKGN36HI32
Released
IM 2.0.0

DryerWTWH7560GB
com.bshg
Information model for DryerWTWH7560GB
Released
IM 2.0.0

© 2019 Eclipse Foundation and its contributors.
Made available under the Eclipse Public License v2.0

Repository version: 1.0.0-SNAPSHOT, Revision: 32339b8 Copyright Agent Legal Terms of Use Privacy

Fig. 2. Screenshot of the landing page of the public Eclipse Vorto repository (https://vorto.eclipse.org/#/, accessed 04.08.2020)

C. Vorto Plugins

Vorto Plugins can be used to process Vorto models to transform them into different formats, representations, source code, and request templates. Currently, there are five officially supported plugins:

- Eclipse Hono Plugin: transforms Vorto models into source code to connect devices to Eclipse Hono via MQTT
- Eclipse Ditto Plugin: transforms Vorto models into Eclipse Ditto Digital Twin representations (JSON or OpenAPI)
- JSON Schema Plugin: transforms Vorto models into a JSON Schema representation
- OpenAPI Plugin: transforms Vorto models into an OpenAPI YML representation
- Bosch IoT Suite Plugin: transforms Vorto models into source code to connect to the Bosch IoT Suite or into a request template to provision devices

In addition to the officially supported plugins, several experimental plugins offer other transformations. Experimental plugins are managed in a separate Github repository.[3] All plugins can be used either as local run applications or as AWS Lambda functions. The public Vorto repository is integrated with the official plugins as AWS Lambda functions and can thus be used directly from the Vorto Repository UI. One can also use the plugin API to develop custom plugins.

D. Vorto Telemetry Payload Mapping

The Vorto Telemetry Payload Mapping engine is a standalone application to map telemetry data that is sent by a device. To use the mapping application, one needs to create a payload mapping configuration, to understand the source format used by the device and the desired normalized target format. The payload mapping engine offers a canonical JSON target format and the Eclipse Ditto JSON format. The normalized payload data can then be used to build applications with normalized APIs based on the Vorto models.

III. Example Models

In the following, we define an artificial example use case to showcase the capabilities of the Eclipse Vorto models. Later in this paper, we use these models for an example mapping of the Vorto meta-model to an Arrowhead SysML Profile. Figure 3 gives an overview of the use case. In the depicted setup, we assume a production facility with several units like conveyors and an assembly robot. A server back end system allows the collection of production data by providing connectivity, a digital twin device abstraction, and storage e.g. through instances of Eclipse Hono[4], Eclipse Ditto[5] and a database. The back-end server also hosts Arrowhead core services to provide the infrastructure for a local cloud with the mentioned machines and software systems. The objective of the use case

[3]https://github.com/eclipse/vorto-examples/tree/master/vorto-generators
[4]https://www.eclipse.org/hono/
[5]https://www.eclipse.org/ditto/

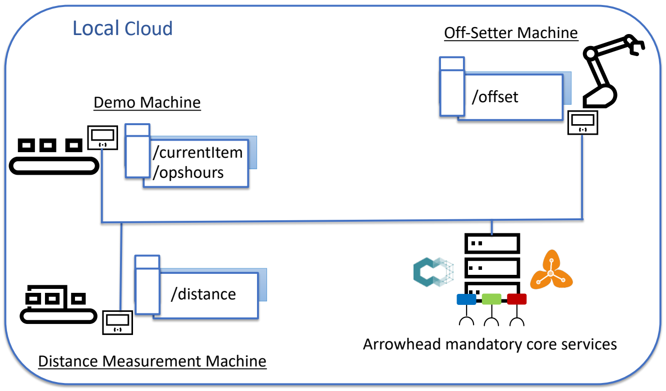


Fig. 3. Demo Setup

is to collect and centrally store data from the various units and to perform basic control operations between the machines.

The local cloud has three machines. One of them is a demo machine for which we created a custom Vorto model. The starting point of the modeling approach is the information model in Listing 1.

Listing 1. Information model for demo machine

```
vortolang 1.0

namespace org.arrowhead.demo
version 1.0.0
displayname "demo-machine"
using org.arrowhead.demo.CurrentItem ; 1.0.0
using org.arrowhead.demo.OpsHours ; 1.0.0

infomodel DemoMachine {

  functionblocks {
    currentItem as CurrentItem
    opsHours as OpsHours
  }
}
```

In our case, this machine tracks its operation hours and which item it currently processes. For both aspects, we defined separate function blocks. Listing 2 shows the function block for the operational hours. Here the assumption is that the machine tracks it operational hours and produces events when it reaches a maintenance window (line 19 to 21) or requires that it gets moved (line 15 to 18) depending on the operation time. We assume that one needs to move the demo machine to avoid it from wearing out when operating in the same position for too long. The operator can further set the duration of the maintenance window and the window until the next movement, which we model with the configuration block in lines 9 to 12.

Listing 2. Function block for operation hours

```
1  vortolang 1.0

3  namespace com.eclipse.arrowhead
4  version 1.0.0
5  displayname "Operation Hours"
6  description "Operating hours"

8  functionblock OpsHours {
9    configuration {
      turnWindow as long
      mandatory largeMaintanceWindows as long
    }

    events {
      moveTimeReached {
        mandatory timestamp as long
        mandatory offset as long
      }
      maintenanceReached {
        mandatory timestamp as long
      }
    }
}
```

Then there is also a function block for the currently processed item depicted in Listing 3. Here the item is identified by an id. Since one can only read but not write this value from the item we modeled this as status.

Listing 3. Function block for the currently processed item

```
vortolang 1.0

namespace com.eclipse.arrowhead
version 1.0.0
displayname "Current Item"

functionblock CurrentItem {
  status {
    mandatory id as string
  }
}
```

Another machine can move the first machine to a given offset. The capability of this off-setter machine can be modeled as an operation in a function block corresponding to Listing 4. We do not show the information model for the second machine because it has strong similarities with the information model of the demo machine.

Listing 4. Model for offset movement

```
vortolang 1.0

namespace com.eclipse.arrowhead
version 1.0.0
displayname "Offset Movement"

functionblock Offset {
  operations {
    moveToOffset(offset as long)
  }
}
```

One had to describe the already mentioned machines in particular models. However, as shown in Figure 6 the modeled Arrowhead local cloud shall also contain a third machine, which measures distances e.g. between objects on a conveyor. For this machine, it is possible to reuse the distance sensor model obtainable from the public Vorto repository [4] and thus integrate the distance measurement machine into the model of the local cloud with minimal additional effort. This integration also highlights the benefit of having a central source for more or less generic models to foster reuse and adoption of existing models to decrease overall engineering overhead.

IV. Eclipse Arrowhead

Eclipse Arrowhead [6] is a newly founded open-source project in incubation at the Eclipse Foundation, which offers methods and tools to bring concepts of service orientation to the Industrial Internet of Things. The *Arrowhead* initiative has a strong focus on fostering interoperability via service and interface descriptions between systems and components [5]. The Arrowhead community originated from a joint European effort of more than 80 industrial and academical partners to bridge the interoperability gaps for applications and tools in IoT-based automated industrial scenarios. Currently, there are multiple projects following up on the promising results, the two most important, large-scale consortial endeavors being Productive 4.0 and Arrowhead Tools. Productive 4.0 explicitly aims at putting Arrowhead concepts into industrial production in the context of Industry 4.0 [6]. In contrast, the goal of the *Arrowhead Tools* project, started in 2019, is to establish a mature software and tooling landscape around the Arrowhead core to foster even broader and more efficient adoption of Arrowhead technology in the industry. It is in the context of Arrowhead Tools that the initially proposed Arrowhead Framework has started its journey in the Eclipse universe.

As for its principles, the whole Arrowhead ecosystem bases on a *service-oriented architecture* (SOA) [7]. However, in contrast to classical SOA, the Arrowhead Framework does not explicitly employ an enterprise service bus (ESB) as a central messaging point. Instead, it uses service-to-service communication as proposed for micro-services instead. The Arrowhead Framework introduces the concept of local clouds, which encapsulate geographically connected processes, such as production facilities. Delsing et al. [5] define five substantial requirements, which local clouds have to meet: (i) Low latency guarantee for real-time use cases; (ii) a high degree of scalability of automation systems; (iii) multi-stakeholder integration and operations agility; (iv) security and safety measures; and (v) ease of application engineering. As long as the listed criteria are met, the Arrowhead local cloud concept does not define the underlying architecture. However, especially latency and security requirements, depending on how important they are for the respective use-case, might require a complete IoT setup involving *edge deployments* [5]. The local clouds contain all necessary components to operate on their own. Generally, each local cloud consists of three kinds of entities:

1) *Devices* are the hardware foundation of each local cloud and are hosts to one or multiple systems. A device is not bound to a specific performance threshold. Hence, small and constrained hardware can be part of the local cloud, as well as more powerful machines.
2) *Systems* are the software artifacts executed on the underlying devices, forming a logical unit of semantically coherent tasks. These systems autonomously register themselves and their provided services at a service registry. Besides service provision, a system is also capable of consuming services of other systems.

[6]https://projects.eclipse.org/projects/iot.arrowhead

3) *Services* are functional representations of systems towards the outside world. They are the primary artifacts in connecting services according to SOA principles: *provided* services get *consumed* by other systems (which might, e.g., depend on specific inputs for their operation). There is no technical specification concerning the choice of protocol or payload format. Since this is part of the interface definition of each service and beyond the scope of the Arrowhead specification. For instance, the system might employ web technology or broker-based communication patterns.

In the following section, we turn ourselves to a founded approach of capturing such local cloud architectures via systems modeling techniques.

V. Modeling a Cloud: Arrowhead Tools and Systems Modeling

The notion of model-based systems engineering (MBSE) plays an important, even crucial role in holistic engineering workflows involving large-scale, complex, dynamic systems [2]. However, MBSE and its primary modeling language, SysML [8] are arguably recognized for capturing monolithic systems with a more fixed (though probably complex) architecture. Recently, there has been a growing interest around the best ways to employ systems modeling in modern, dynamic, even cloud-based scenarios.

As for modeling Arrowhead local clouds, we rely on the aforementioned established systems modeling language and methodology, SysML. SysML is a dialect of the well-known Unified Modeling Language (UML), tailored to meet the specific needs of systems engineering activities. In turn, modeling Arrowhead local clouds requires a custom-tailored approach with a considerable amount of flexibility to adequately capture the diverse set of entities as introduced above. SysML is the canonical language of choice for such endeavors. Also, SysML excels at language extensibility (being a primary concern) and has mature, feature-rich tooling and an active community. The language provides several different diagram types to represent the facets of the system to be modeled, from requirements to static structures to communication protocols. For further general details, there is a large variety of textbooks available — e.g., for practical information of SysML, refer to the comprehensive book of Friedenthal et al. [8].

Recently, there has been a proposal for a concrete, Arrowhead-centered approach for modeling service-oriented applications, i.e., Arrowhead *Systems of Systems* [9], extending and refining the Arrowhead documentation approach proposed earlier [10]. The solution is a standard UML/SysML mechanism to enlarge and tailor the modeling language for a specific domain. In short, a profile is an organized collection of *stereotypes*, i.e., domain-oriented specializations of generic SysML language concepts. We refer the interested reader to [9]; here, we focus on those parts which have direct relevance to the present integration approach:

- The so-called *interface design descriptions* (IDD) can be conceived as the realization blueprints for certain services on certain systems (cf. Sect. IV). In particular, IDDs contain operation signatures representing service functionality. A single IDD can be used as a service "type" for modeling both provider and consumer behavior.
- A central Arrowhead notion is that of *devices*; thus, there is a corresponding stereotype, serving as a mere placeholder (better said, a canonical integration point) in the original version of the profile. The present paper is, in fact, an actualization of such an integration, which results in filling that stereotype with life and details.
- A local cloud configuration is modeled via *deployed entitites*, represented as specialized SysML *part properties*. This part configuration is the place where platform modeling gets realized on a (conceptual) deployment level—the next section demonstrates this through examples.

This profile and modeling approach is referred to as `SoSysML` in the rest of the paper. Figure 4 shows an overview of the `SoSysML` representation of our example local cloud (cf. Sect. III); in particular, the upper row of the diagram consists of *system design descriptions* (SysDDs), representing design templates for the three system kinds involved in the use-case. SysDDs also play a significant role as the hosts for actual interfaces. This is materialized by their *ports* (the small rectangles at the edge of the SysDD boxes). The essential logical structure of `SoSysML` lies here: SysDDs are brought together with IDDs (abstract service and interface descriptions not explicitly depicted here) by *using IDDs as types of ports on SysDDs*. Thus, a SysDD represents an object (a system) while its ports express its behavior (via the typing IDDs). Details on IDDs are out of scope here—for the present paper, it suffices to conceive of them as operation collections.

The bottom row, in turn, contains the SysML representations of those *device kinds*, whose *instances* the actual system *instances* will be allocated to. These device templates come in two fashions: in some cases like the distance sensor in our running example, an already modeled device can be readily used and, thus, directly imported from the Eclipse Vorto repository, while in other cases, the design process might necessitate the modeling of new devices. The next section covers details of this instantiation and allocation.

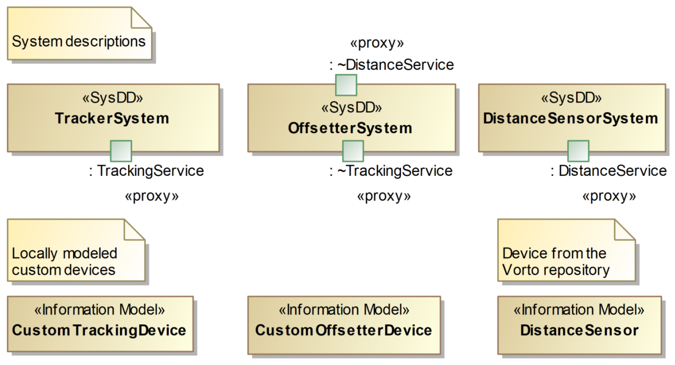


Fig. 4. `SoSysML` Overview: The Smart Assembly Use-Case (Sect. III)

Note that the above figure is an excerpt from an actual model realization as available in MagicDraw, an industrially established and widely used systems modeling tool. Thus, the integration presented as the essential contribution of this paper in the following Section VI also reveals an industrial potential w.r.t. base technologies involved here: it seems that while IoT device/deployment models are addressed in the Eclipse ecosystems, abstract systems modeling and platform-independent design have a home in the NoMagic[7] (MagicDraw, Cameo Systems Modeler, Teamwork Cloud, ...) tool infrastructure. The prototype serving as the practical baseline for the present contribution constitutes an important step towards integrating these two platforms/ecosystems (having some conceptual touching points which we can rely on) and, thus, we find that this "mismatch" is more a benefit than an impediment. Moreover, the underlying implementation of our MagicDraw plugin is based on VIATRA, an established Eclipse model transformation engine.[8] We will continue to build upon this combined ground in the future. As for the future potential of the current approach, we also remark that this approach is very likely to play a leading role in the future of systems modeling as well. Currently, the upcoming new major release of the systems modeling standard, SysML v2, considers the Arrowhead SoS profile mentioned as a candidate for SOA modeling standardization.

VI. Vorto, SysML, Arrowhead: The Integration Approach

As a culmination and summary of the ideas introduced above, we turn ourselves to the primary concept of this paper: the integration of Eclipse Vorto device models with SoSysML, i.e., our Arrowhead-specific SysML-based design approach to device-independent SoS (local cloud) modeling.

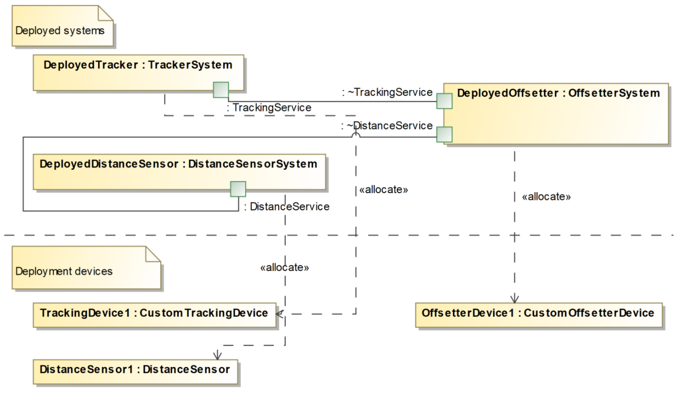
On a conceptual level, Table I summarizes the essence of the integration approach. As it has already been hinted at in the previous section, the main observation behind the integration is a direct correspondence, i.e., a mapping, between concepts from SoSysML and Vortolang. In particular, *Device* in SoSysML serves as a topmost container for any device descriptions, thus corresponding to the topmost Vortolang elements, *Information Models*. The IDDs correspond to the actual functional specifications, the *Function Blocks* of Vortolang. This concise table also indicates that below this level, the two modeling languages become equivalent: their *Operation* concept represents the same abstraction level.

TABLE I
SoSysML to Vortolang concept mapping

SoSysML	**Vortolang**
IDD	Function Block
Device	Information Model
Operation	Operation

The essence of our contribution, i.e., the actual integration of SoSysML local cloud models and Vorto digital twins is

[7]https://www.nomagic.com/
[8]https://www.eclipse.org/viatra/



Deployed systems
DeployedTracker : TrackerSystem
: ~TrackingService
: TrackingService
DeployedOffsetter : OffsetterSystem
: ~DistanceService
DeployedDistanceSensor : DistanceSensorSystem
: DistanceService
«allocate»
«allocate»
Deployment devices
«allocate»
TrackingDevice1 : CustomTrackingDevice
OffsetterDevice1 : CustomOffsetterDevice
DistanceSensor1 : DistanceSensor

Fig. 5. Example Use-Case: SoSysML-Vorto Integration

illustrated through our example in Fig. 5. The upper region above the horizontal dashed line (*Deployed systems*) is the way to organize non-allocated, abstract, platform-independent system instances into a local cloud in SoSysML: each box represents a system instance to be deployed, where the string before the colon is its proper name, while the string after the colon indicates its type, which is, in turn, a SysDD (cf. Fig. 4). This typing provides each system with an interface (port) structure. The structure is turned into an abstract, design-time local cloud representation by the connectors (solid lines) attaching provided interfaces to consumed interfaces (the latter denoted by the same type name depicted with a tilde prefix). Notice the resemblance of this part of the figure to the intuitive scenario depiction in Fig. 3.

The extended, integrated local cloud representation (including the lower region of Fig. 5) now contains Vorto devices as well. These are represented, again, by boxes with typed instances (using the same label convention)—but here, the types come from a set of Vorto device descriptions (represented as *Information Models*, cf. Table I). In turn, these information models might originate from two different sources (cf. also the bottom line of Fig. 4), both considered by our *MagicDraw Vorto Importer* module:

- they either directly come from the Vorto repository itself; to this end, the importer browses the online catalog and the user can automatically create a SysML representation of such information models, if she or he thinks that one of them would fit their design (the case of the DistanceSensor in our example); or
- if the desired functionality is not yet represented by any "stock" solution, the user can choose to write his or her own Vorto specification with any preferred local workflow conforming to Vortolang. For example, they can use the Vorto IDE or the web modeler, and might even reuse already predefined function blocks.

Based on the Vorto models users may also use one of the various code generators and plugins to generate integration code for different back-end systems. We implemented both of the aforementioned model input sources in our implementation of the *MagicDraw Vorto Importer*, a plugin of the established systems modeling tool MagicDraw[9] (or its SysML-equipped distribution Cameo Systems Modeler, the difference being immaterial for our purposes). The prototype features a user-friendly manner, where the built-in *Import* menu is extended by two further options one for browsing the Vorto repository and choosing information models to import, the others to browse locally for user-created Vorto model packages. From this point on, the usage of the imported information models is the same regardless of the source, as detailed above.

VII. Applying the Integration Approach

In the previous section, we presented our integration approach. Now we demonstrate its use in an engineering workflow using our running example from Sect. III. As our metaphor in the title suggests, our integration approach fills `SoSysML` models of Arrowhead local clouds with device details, i.e., relates the cloud model with the bare metal devices (or their abstractions).

To realize the scenario described in Sect. III, the following steps have to be carried out.

a) Model System of Systems Model: Here the initial Model of the SoS scenario, i.e. the Arrowhead local cloud, has to be modeled. In the example case depicted in Fig. 6, one has to create a SysML model for the three Systems and their relationships, as shown in Fig. 4. This step is not unique to our mapping approach but is more a prerequisite.

b) Select Devices in Repository: With the SoSysML model available, the modeler can turn to fill the modeled abstract devices with life. As the Vorto repository already contains a considerable number of different models, the first step should be to browse it and check if an information model of the required device already exists. In our example, this is the case for the distance sensor modeled for machine three.

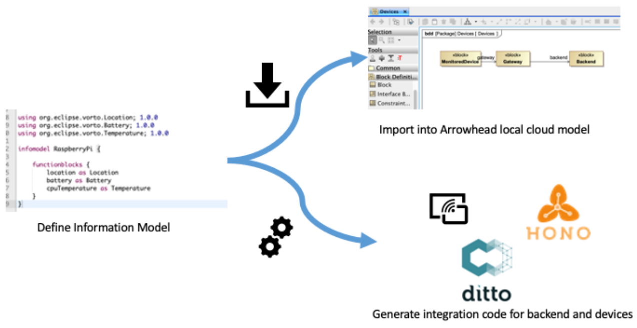
c) Model Device: If a model does not already exist in the repository, the modeler can still use VORTOLANG to model the device capabilities. In our example, this is done with the devices described in Listings 1–4. Those models can be imported into the SoSysML as well. Using VORTOLANG has the advantage that the whole infrastructure provided by the Eclipse Vorto project can still be applied. For example, one can integrate existing function blocks into an information model. It is then possible to publish the models later on in the Vorto repository if they are stable and of broader use.

d) Import Device Models: With the device models at hand, the modeler can now import the Information models into the SoSysML models (upper part of Fig 6). The device models provide the SoSysML model with the concrete capabilities of the devices, which in turn allows for fine-grained modeling of processes and procedures. An example is the conveyor maintenance procedure described in Sec. III.

With that step, the actual integration of Eclipse Vorto and SysML is complete but there are some further steps possible.

e) Generate Connectivity Code: As the device models are available as Vorto information models also the code generators of Eclipse Vorto can be used to generate the connectivity code (lower part of Fig 6). In our example, the machines connect to an Eclipse Hono instance. So we need code for the connectivity with Eclipse Hono. The Vorto generator plugin for Hono directly generates connectivity code stubs in C (Arduino), Java, and Python that can be integrated into the connectivity stack of the machine or an attached gateway.

f) Perform Arrowhead Wiring: As the `SoSysML` model has all the required data, the actual Arrowhead wiring, i.e., registration of the services at the Service Registry and configuration of the Arrowhead Orchestrator can happen semi-automatically utilizing another plugin.[10]


```
using org.eclipse.vorto.Location; 1.0.0
using org.eclipse.vorto.Battery; 1.0.0
using org.eclipse.vorto.Temperature; 1.0.0

infomodel RaspberryPi {

    functionblocks {
        location as Location
        battery as Battery
        cpuTemperature as Temperature
    }
}
```

Fig. 6. Applying Vorto models in an engineering process

VIII. Related Work

The integration approach pursued in the paper touches on various industrial concepts and frameworks of high relevance; however, due to the rising interest around such topics, it is not typical in IIoT and Industry 4.0 to have widely established industrial solutions. We perceive an abundance of often non-public domain-specific architectural solutions. The EU has started to establish an industrial reference architecture framework, RAMI4.0 [11], which represents a much higher abstraction than our present contribution, and a direct comparison is, therefore, out of scope here. We mention some related approaches in the most relevant fields in the following.

First, service-oriented architectures already have a standard modeling language called *SoaML* [12] which is also a dialect of UML (just as SysML). The concepts which SoaML relies on are more orthogonal to the SysML-based design pursued here. However, our flexible setup allows an integration of various further diagram and representation types, even SoaML. SoaML also lacks a device modeling aspect and one therefore would have to refactor the Vorto integration in that case too. For an overview of other SOA standards, refer to [13].

The topic of *platform modeling* and the clear distinction between a platform-independent and a platform-specific model (PIM/PSM) is a fundamental concept in the OMG standard

[9]https://www.nomagic.com/products/magicdraw

[10]https://github.com/IncQueryLabs/arrowhead-tools

Model-Driven Architecture [14]. Arguably, our integration approach embodies this concept, `SoSysML` models being representing the PIM, and their Vorto mapping the PSM part of a comprehensive platform model. Other European industrial digitalization projects have investigated well-founded platform modeling, such as the OpenCPS[11] endeavor.

As for IoT architecture design, there is no industrially accepted general solution, framework, or workflow yet; i.e., every domain builds its specific modeling solution. However, the Eclipse IoT Working Group aims at providing generic technologies to enable end-to-end IoT solutions [15]. As for other European approaches, Krčo et al. [16] had a look into the various IoT endeavors within Europe. They identified several projects and organizations, which either define different IoT reference architectures or provide reference implementations for IoT core systems. Further, they describe a landscape of IoT activities in Europe in which the Arrowhead framework fits quite well and for which the described integration approach, with adjustments, could be of use, too.

As for combining Vorto with SysML, Pfenning and Roth [17] present an approach to generate Vorto Information Models from existing SysML models. The objective is to generate digital twin implementations from existing SysML models of the Product Lifecycle Management (PLM) via model transformation techniques. The authors make use of the existing generator implementations of Eclipse Vorto to obtain digital twin implementations for different IoT-Platforms. Thus, their approach is the inverse of our method described in this paper.

IX. Conclusion and Future Work

In this paper, we have presented a novel integration approach to combine System-of-Systems modeling and a device modeling ecosystem into a comprehensive industrial IoT platform design solution. In particular, building on ongoing work in *Arrowhead Tools*, we extend the SysML-based modeling approach `SoSysML` with a capacity to represent devices within the models in a tightly integrated fashion, resulting in full-fledged, compact platform models. Moreover, we have created and presented an importer tool prototype, not only to demonstrate feasibility, but also to emphasize the integration potential between the NoMagic modeling ecosystem on the one hand, and the Eclipse Vorto ecosystem on the other hand.

An immediate item for future work is to investigate tooling extensions by reaching deeper into both main ecosystems involved here. On the one hand, we might consider the Eclipse-native, but little less SysML centered modeling tool Eclipse Papyrus.[12] On the other hand, the NoMagic modeling ecosystem has built-in model repository features that resemble the shape of the Eclipse Vorto offering; such features could allow for building a repository system for IIoT.

Acknowledgment

The research has received funding from the EU ECSEL JU under the H2020 Framework Programme, JU grant nr. 826452 (Arrowhead Tools project, https://www.arrowhead.eu) and from the partners' national funding authorities.

Project no. 2019-2.1.3-NEMZ_ECSEL-2019-00003 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-2.1.3-NEMZ_ECSEL funding scheme.

Project no. 16ESE0367 has been implemented with the support from the Federal Ministry of Education and Research of Germany.

[11]https://www.opencps.eu/

[12]https://www.eclipse.org/papyrus/

References

[1] M. Jamshidi, *System of systems engineering: innovations for the twenty-first century*. John Wiley & Sons Incorporated, 2009, vol. 58.

[2] P. Micouin, *Model Based Systems Engineering: Fundamentals and Methods*. John Wiley & Sons, 2014.

[3] The Eclipse Vorto project, "Eclipse Vorto," accessed: 2020-07-17. [Online]. Available: https://www.eclipse.org/vorto/

[4] A. Edelmann, "Distance Sensor - Informationmodel," 2019, accessed: 2020-07-17. [Online]. Available: https://vorto.eclipse.org/#/details/org.eclipse.vorto.tutorial:DistanceSensor:1.0.0

[5] J. Delsing, P. Varga, L. Ferreira, M. Albano, P. P. Pereira, J. Eliasson, O. Carlsson, and H. Derhamy, "The arrowhead framework architecture," in *IoT Automation*. CRC Press, 2017.

[6] D. Kozma, P. Varga, and G. Soós, "Supporting digital production, product lifecycle and supply chain management in industry 4.0 by the arrowhead framework–a survey," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 126–131.

[7] M. Bell, *SOA Modeling patterns for service-oriented discovery and analysis*. John Wiley & Sons, 2009.

[8] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[9] G. Kulcsár, K. Kadosa, T. Szvetlin, B. Péceli, A. Horváth, Z. Micskei, and P. Varga, "From models to management and back: Towards a system-of-systems engineering toolchain," in *Proc. of IEEE NOMS Workshop on Management for Industry 4.0*, 2020.

[10] F. Blomstedt, L. L. Ferreira, M. Klisics, C. Chrysoulas, I. M. de Soria, B. Morin, A. Zabasta, J. Eliasson, M. Johansson, and P. Varga, "The Arrowhead approach for SOA application development and documentation," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, 2014, pp. 2631–2637.

[11] T. Bangemann, M. Riedl, M. Thron, and C. Diedrich, "Integration of classical components into industrial cyber–physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 947–959, 2016.

[12] B. Elvesæter, C. Carrez, P. Mohagheghi, A.-J. Berre, S. G. Johnsen, and A. Solberg, "Model-driven service engineering with soaml," in *Service Engineering*. Springer, 2011, pp. 25–54.

[13] H. Kreger and J. Estefan, "Navigating the SOA open standards landscape around architecture," *Joint Paper, The Open Group, OASIS, and OMG*, 2009.

[14] J. Bézivin and O. Gerbé, "Towards a precise definition of the omg/mda framework," in *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*. IEEE, 2001, pp. 273–280.

[15] The Eclipse IoT Working Group, "The three software stacks required for iot architectures," Eclipse Foundation, https://iot.eclipse.org/community/resources/white-papers/pdf/EclipseTech. Rep., 2017.

[16] S. Krčo, B. Pokrić, and F. Carrez, "Designing iot architecture (s): A european perspective," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014, pp. 79–84.

[17] M. Pfenning and A. Roth, "Systemmodellierung für das internet der dinge – transformation von systemmodell in iot-plattform im kontext später produktlebenszyklusphasen," in *Tag des Systems Engineering*, S.-O. Schulze and C. Muggeo, Eds., GfSE. Hanser, 2016, in German.

Model Based Methodology and Framework for Design and Management of Next-Gen IoT Systems

Xu Tao, Davide Conzon, Enrico Ferrera
LINKS Foundation
Turin, Italy
{name.surname}@linksfoundation.com

Shuai Li
CEA, LIST
Paris-Saclay, France
{name.surname}@cea.fr

Juergen Goetz
Siemens AG Corporate Technology
Munich, Bavaria, Germany
juergen.goetz@siemens.com

Laurent Maillet-Contoz,
Emmanuel Michel, Mario Diaz-Nava
STMicroelectronics
Grenoble, France
{firstname.lastname}@st.com

AbdelHakim Baouya, Salim Chehida
Univ. Grenoble Alpes
Grenoble, France
{name.surname}@univ-grenoble-alpes.fr

***Abstract*—Internet of Things (IoT) is a pervasive technology covering many applications areas (Smart Mobility, Smart Industry, Smart Healthcare, Smart Building, etc.). Its success and the technology evolution allow targeting more complex and critical applications such as the management of critical infrastructures and cooperative service robotics, which requires real time operation and a higher level of intelligence in the monitoring-control command for decision-making. Furthermore, these applications type need to be fully validated in advance considering that bugs discovered during real operation could cause significant damages. In order to avoid these drawbacks, IoT developers and system integrators need advanced tools and methodologies. This paper presents a methodology and a set of tools, defined and developed in the context of the BRAIN-IoT European Union (EU) project. The overall framework includes both Open semantic models to enforce interoperable operations and exchange of data and control features; and Model-based development tools to implement Digital Twin solutions to facilitate the prototyping and integration of interoperable and reliable IoT system solutions. After describing the solution developed, this paper also presents concrete use cases based on the two critical systems mentioned above, leveraging the application scenarios used to validate the concepts developed and results obtained by the BRAIN-IoT project.**

***Index Terms*—Model-Based System Engineering, Internet of Things, Digital Twin, Brain-IoT**

I. Introduction

Nowadays, the IoT concept is adopted in new application domains, allowing fast digitalization of contemporary society [1]. The application of IoT in innovative scenarios such as critical infrastructures management, and cooperative service robotics demand to satisfy a set of strict requirements in term of low latency, high reliability, adaptability, heterogeneity and scalability, highly more challenging than the ones required by the traditional (e.g., domotics) environments. To satisfy these requirements, the IoT developers needs to introduce in their solutions next generation Internet of Things (next-gen IoT) technologies, e.g., Edge Computing, Artificial Intelligence (AI), Digital Twin, among others [2], thus leading them to become more complex to design and manage and requiring the introduction of methodologies and tools that ease the users their development and runtime management.

Recently, several approaches have been proposed to ease the development of IoT systems (see section III-A). However, these solutions do not generally support all the functionalities required by next-gen IoT applications and focus only on development, not supporting the other phases of the application life-cycle, e.g., deployment, validation, monitoring and adaptation at runtime. Currently, the market asks for IoT solutions supporting business critical tasks that can be deployed rapidly and with low costs. Such solutions need to allow the design of applications involving several interconnected heterogeneous platforms and smart things and, at the same time, be able to deploy, monitor and evolve the designed complex solution adapting automatically and at runtime to environmental changes.

This paper is organized as follow: Section II presents the motivation and the requirements that has led to the development of the BRAIN-IoT Modeling & Verification Framework presented in this work. In Section III-A, some existing model-based system engineering approaches are discussed. Then, the BRAIN-IoT modeling methodology is introduced in Section IV and the implementation of the methodology is illustrated in Section V. Next, two use cases, in Section VI, are exploited to demonstrate the functionalities provided by the BRAIN-IoT Modeling & Verification Framework. Finally, Section VII concludes the paper.

II. Motivation and Requirements

The design of an IoT system today presents considerable complexity. Several factors contribute to this complexity: first, an understanding of IoT is still too focused on IoT technologies and objects (device types, communication protocols, cloud, database type, etc.). Such a vision loses sight of the true purpose of the system to develop and leads most of the time to unsatisfactory solutions. Then, the wide variety of IoT systems, in terms of their deployment, is a technological source of complexity: IoT systems can be considered with a ”local” deployment such as an automatic lighting system of a house (measurement, storage, data analysis and execution of the application on a single ”device” connected to the owner’s smartphone), or systems with a ”highly distributed” architecture such as a weather forecasting system (sensor networks for data capture capabilities, plus a cloud for storing, analyzing data and running the end-user application). Another aspect concerns the need to integrate legacy systems, i.e., IoT systems not designed from “zero”, the problem is then to create new innovative services on the basis of existing infrastructures. In this case, it is necessary to ”connect” what exists and then develop / integrate supporting components (authentication, monitoring, data distribution, data analysis, etc.). The objective of this work is to develop a framework that supports the designer of complex next-gen IoT applications, easing the use of disruptive technologies, e.g., Digital Twins.

For this scope, the solution proposed needs to allow i) modelling several aspects of an IoT system: the physical layer, the IoT devices, the system layer, the system behaviors and the interactions among the components. ii) Composing IoT services also provided by different IoT platforms, using a model-based design approach and semantically annotated data formats to support interoperability among heterogeneous systems. iii) Formally verifying and validating the models designed with the framework. iv) The automatic generation of code from the models to be deployed on real devices. v) Supporting the co-simulation approach, with the creation of a mixed reality environment, where virtual and real entities can interact with each other. vi) Monitoring the IoT applications at run-time, keeping the models and the physical world synchronized with each other. The next section will provide a state-of-the-art (sota) of the solutions available to design and manage next-gen IoT applications.

III. Background

A. SOTA

A system design process that adheres to the principles and methods related to ”system engineering” allows to understand the design phase of a complex system w.r.t. expected functionality, cost, time, and quality. In the area of critical systems, system engineering is in full development and benefits from domain-specific and often user-specific solutions. Model-based system engineering is a strong trend, and recent projects such as AGeSys [3] and Connexion [4] highlight its importance and have provided an important foundation for the development of tooled solutions. This work places particular emphasis on the importance of having interoperable, open, and scalable solutions.

In the field of IoT, however, the offer has not yet reached this level of maturity, even though stakeholders are investing heavily in the implementation of model-based design solutions for distributed software architectures, such as Ericsson for network software through its development solution based on the Papyrus [5] open source tool; or THALES through the establishment of its Capella [6] engineering chain; or Dassault System with its Delmia [7] solution.

More specifically, in the field of IoT system engineering, the Internet of Things - Architecture (IoT-A) project [8] proposes a methodological framework of design based on the elaboration of a set of models for the specification of the system according to different views. It is important to note that the reference models proposed by IoT-A are independent of modeling languages that could be used to represent them, even though different case studies applying this methodology have been developed using Unified Modelling Language (UML) models.

In the world of standards, the international standardization organization Object Management Group (OMG) has developed the System Modeling Language (SysML) [9] with the aim of treating software systems, CPS or organizational systems alike. The generality of SysML requires that it be specialized and, like any language, it must also be accompanied by a specific methodology for each application domain so that its use is best adapted to a given field of application and reach a level of maximum efficiency. This is for example what has been achieved for the field of critical embedded systems in the AGeSys project, or for the field of nuclear power plants control-command systems in the project Connexion.

Like embedded real-time systems, execution platforms are a prime concern for IoT systems. The problem of platform modeling and application allocation on the execution platforms, in the field of embedded real-time systems, has been addressed by the OMG in the context of the Modeling and Analysis of Real-Time and Embedded systems (MARTE) [9] standard, which complements SysML on this aspect. However, MARTE needs to be taken up for two aspects: the first is related to its richness which makes it a complex language to handle and requires to be supported by a methodological approach targeting the essential elements to the modeling of IoT systems; the second point is that it remains very focused on the embedded concerns, and only addresses very little distributed system aspects, in particular the aspects related to the communications and the interactions between the distributed components. The link between the system / platform model, and the component / communication standards, needs to be refined and even developed for some new cases, e.g., OMG CORBA Component Model (CCM) [10], DDS for Lightweight CCM (DDS4CCM) [11], Service oriented Architecture Modeling Language (SoAML) [12] and more recently Unified Component Model (UCM) [13]).

The authors have identified that no one of the approaches described above is able to satisfy all the requirements listed in Section II. Besides, methodologies proposed for the IoT domain today do not benefit from dedicated model-based development tools. Indeed, either these methodologies target closed systems (e.g. critical embedded systems) or their only goal is to only establish a guideline for the design process. In the latter case, neither languages, nor development techniques, are defined. However, the choice of a modelling language, and its dedicated tools that will exploit the models, is imperative for the development of well-suited integrated development environment for the specific needs of the IoT domain.

B. Progrees Beyond SOTA

In this paper, the authors propose system engineering solution for the IoT domain. The proposed solution combines standard languages with methodologies, while remaining open to domain-specific practices. Furthermore, the approach fosters tools related to the modelling languages and methodologies. The authors of this work believe this solution aims at filling the gaps in the current system engineering offering for IoT systems, while not omitting current developer practices.

The solution provides such main features: i) Modelling methodology: A lightweight methodology accompanies IoT Modelling Language (IoT-ML). Using the language's capacity to describe different levels of abstraction, the methodology proposes to model entities representing system, software, and hardware components. By relating such components among them, and by linking them to their domain-specific representation and tools, the authors of this paper promote a system holistic view of the whole IoT solution. ii) System behavior modeling: It is based on IoT-ML and integrated with World Wide Web Consortium (W3C) Web of Things (WoT) Thing Description (TD), thus the service provided by the IoT devices, communication protocol and its data structure can be modeled as well. iii) IoT physical layer modeling and validation: It models the IoT devices, BRAIN-IoT adopted the digital twin concept to provide the ability for system application validation on the modeled IoT devices before deploying in the production environment. iv) Models@Runtime: One particular aspect that is not well explored in the IoT domain, and critical embedded systems domain altogether, is the human-friendly monitoring of system state at runtime, directly through the system models. A model-based runtime monitoring approach usually helps identify and fix deviations observed at runtime, compared to formal specifications defined in the models. In the BRAIN-IoT solution, not only is runtime formal validation a priority, but the authors also wish to benefit from monitoring to enable behavior explanations friendly to humans. v) Code generation: The code to be deployed is automatically generated from the models.

As mentioned, the core of the system engineering solution proposed for IoT is based on IoT-ML and its Papyrus tooling. However, as a reminder, this work wishes to promote integration of domain practices through linking artefacts or refining models. The following sections describe some domain-specific languages and tools that refine entities in the system model.

IV. BRAIN-IoT modeling methodology

This section will present the modelling methodology proposed BRAIN-IoT. As the primary objective, BRAIN-IoT aims to allow modeling in different abstraction layers. Firstly, it designs the system-level model composed by the involved components' functionalities and interactions based on the system requirements. The system model also eases the linking towards real devices and external services through metadata generation, and human-friendly monitoring of device behaviors. Then, the system-level model will be refined as a formal software-level model whose correctness will be checked by using the statistic modeling checking and formal verification. The obtained correct software model is used by a code generator to generate the software artefacts. Finally, IoT devices can be modeled as the refined physical-level models to validate and test the IoT applications before deploying to the real physical infrastructure. Hence, it allows to have three different abstraction level models, including the system-level architecture models, software-level components models, and physical-level IoT devices, the focus of each abstraction level is as follows: i) **system-level models** focuses on composability of services provided by the IoT devices/platforms and the overall system behaviors. The system model also serves as an aggregator of blocks that are refined in lower level models, i.e., the software and device described further. Finally, the authors of this paper promote exploitation of the system model, not just for design, but also to help deployment, fast prototyping, and human-friendly monitoring of behaviors at runtime. ii) **software-level models** are the formal models of computation with their formal validation capabilities, and their runtime to guarantee execution conformity to formal specifications, are obtained from the system-level models through the syntactical transformation. iii) **IoT physical device models** allow the virtual representation of the edge domain of an IoT system following a Digital Twin approach, i.e., the possibility to combine a virtual world with a physical one to validate the behavior and performances of a complete system or a system of systems in various steps. This is necessary when the system is complex (the number of devices is high, from 100's to 1000's) and/or in the case of critical systems. These models allow the modeling and simulation of the components performing the sensing / actuating, computing and communication functions, which are the fundamental functions of an IoT system. This approach could also facilitate the carry-out of data analysis. However, to design and validate the physical device, the digital twin model must be refined. This paper describes the additional modeling levels required to perform such design. The additional models will allow designing and validating both the hardware components and the embedded software them. The main expected benefits are twofold: first, the elimination of the main bugs impacting the behavior and the performances of the end devices (e.g., power consumption, reach) and the whole system; second, the increase of the system robustness to facilitate and accelerate the complete system deployment in a more secure manner.

V. BRAIN-IoT Modelling & Validation Framework

This section presents the BRAIN-IoT modelling & Validation framework developed to implement the methodology described in Section IV. BRIAN-IoT Modelling & Validation Framework defines a domain-specific Modelling Language describing IoT devices capabilities and system-level behaviors; it also provides toolset supporting the syntax of the modelling

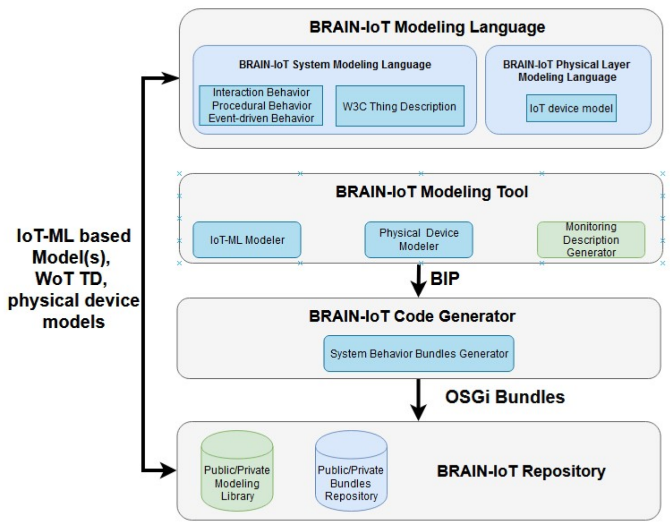

BRAIN-IoT Modeling Language
BRAIN-IoT System Modeling Language
Interaction Behavior
Procedural Behavior
Event-driven Behavior
W3C Thing Description
BRAIN-IoT Physical Layer Modeling Language
IoT device model
BRAIN-IoT Modeling Tool
IoT-ML Modeler
Physical Device Modeler
Monitoring Description Generator
BIP
BRAIN-IoT Code Generator
System Behavior Bundles Generator
OSGi Bundles
Public/Private Modeling Library
Public/Private Bundles Repository
BRAIN-IoT Repository
IoT-ML based Model(s), WoT TD, physical device models

Fig. 1. BRAIN-IoT Modelling & Validation Framework Components

language, allowing model verification, model checking, automatic code generation to provide rapid model-based development approach, and Models@Runtime monitoring features. The components in the BRAIN-IoT modeling & Validation framework are shown as in Fig. 1.

There are four main components in the BRAIN-IoT Modelling & Validation Framework: BRAIN-IoT Modelling Languages and Modelling Tools, BRAIN-IoT Code Generators, and BRAIN-IoT Repository. The detailed introduction for each component are presented in the following subsections.

A. BRAIN-IoT Modelling Language

The BRAIN-IoT Modelling Language is decomposed for three purposes: the system modelling language, the software modelling language, and the physical layer modelling language. Each of these modelling languages suits a different concern according to the abstraction layer. As a reminder, these abstraction layers are the system behavior, the software, and the physical layer. The authors of this paper believe using a domain-specific language, and de-facto common practice, suits the needs and habits of the domain-specific developer. Relationships are manually input to relate elements of each abstraction layer. This allows to have a holistic view of the whole architecture. Each of the following sub-sections describe a particular modelling language for a particular abstraction layer.

1) System Modelling Languages and Tools:

a) System Modelling Language: The IoT-ML is the system modelling language of BRAIN-IoT. It federates the specifications of heterogeneous sub-systems within a global IoT system. The language provides the necessary constructs to design the structural and behavioural system architecture of the system, entities abstracting the software, and entities abstracting the execution platform. At its conceptual core, IoT-ML integrates the concepts present in the IoT-A architecture reference model. Such a model is shown in Fig.2, extracted from the standard.

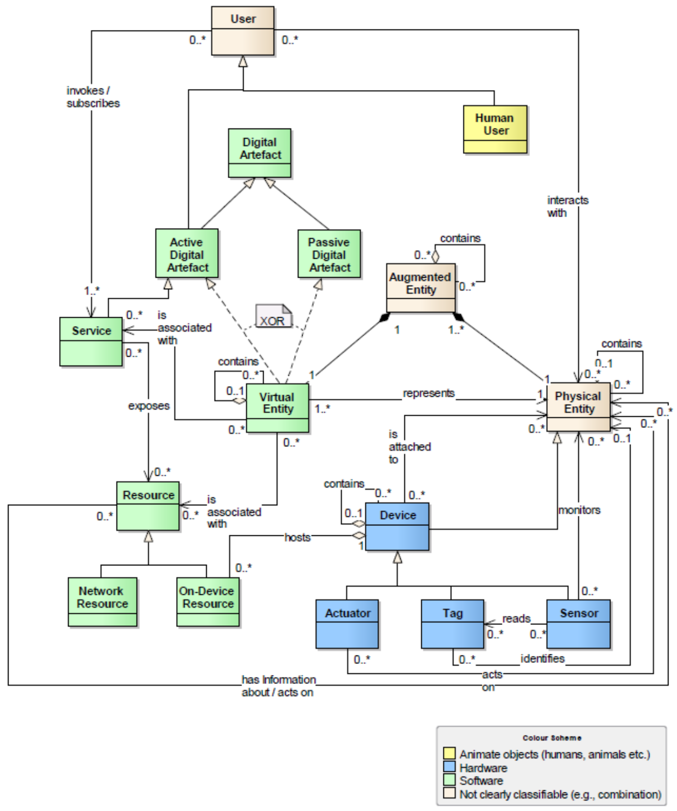


Fig. 2. IoT-A concepts in IoT-ML

IoT-ML is implemented as a UML profile. The UML is a generic modelling language with heavy roots in the object-oriented community. A UML profile is an extension of UML. It is composed of stereotypes that give additional syntax and semantics to the base UML elements that the stereotype extends. IoT-ML aggregates syntax and semantics from standard UML profiles to benefit from the languages they implement. SysML is used to benefit from its ability to describe and trace requirements. MARTE is used not only for the design of real-time embedded systems design, but also because it fosters the construction of models that may be used to make quantitative predictions taking into account IoT characteristics. IoT-ML takes a subset of stereotypes from such standards, and adds new stereotypes of its own representing concepts of IoT-A that are not present in MARTE or SysML. For example Virtual Entity, that can be both software and hardware, is a concept that does not exist in the UML standard profiles.

As a UML-based model, IoT-ML is a graphical modelling language. The language focuses on structural modelling. Such models are represented in UML composite structure diagrams. Components have internal structures that show parts exposing their interfaces through ports that are then connected together.

IoT-ML also focuses on behavioral modelling in the form of UML state-machines. In such models, the behavior represents the component's different states. States can pass from one to the other, based on captured events. Events may be due to operation calls or signalling. Specific behaviors may be executed upon transitioning across states, or entering / exiting / staying in a state.

While IoT-ML can already be used for domain-generic IoT systems modeling, within BRAIN-IoT, the authors of this paper have showcased that the core of IoT-ML (based on MARTE, SysML, extended with IoT-A concepts) is generic and rich enough to build domain-specific extensions for both IoT standards and IoT technologies. Indeed, IoT-ML has been extended with new concepts for the W3C TD standard. For example, the main concept of the W3C TD is the Thing, which can be either software or hardware or both. To showcase IoT-ML for a particular IoT technology, the authors of this work chose to integrate sensiNact [14] concepts into IoT-ML. The sensiNact platform interoperates several different middleware and communication protocols common to the IoT domain, e.g., Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP). Its particularity is that it has a common data model to represent all devices connected to different protocols. It is then possible monitor such devices' variables through common sensiNact API. The common API are also used in behaviors, described with the sensiNact domain-specific textual language, to prototype behaviors actuating the devices according to monitored variables.

Thanks to the profile mechanism of UML, all these domain-specific stereotypes can co-exist with the core IoT-ML stereotypes on the same UML base elements. Otherwise said, a same structural or behavioral element, of an IoT-ML architecture model, can be annotated with information specific to W3C TD or sensiNact. This fosters model re-use, separation of concerns, and model consistency through annotating the same base elements.

The mission of W3C WoT (Web of Things) is to counter the fragmentation in the IoT world through standardized complementing building blocks - e.g., metadata and Application Programming Interfaces (APIs) - based on Web technology. WoT enables easy integration and interoperability across IoT platforms and application domains. Therefore, the goal of WoT is to preserve and complement existing IoT standards and solutions like for the BRAIN-IoT domains Robotics and Critical Water Infrastructure. In this context, the usage of WoT-compliant *Thing Descriptions (TDs)* lays the foundation of interoperable standardized solutions for the various BRAIN-IoT domains and avoids their silo-like separation in order to overcome the problematic diversity of IoT systems.

There are several prominent W3C standard recommendations for WoT based on the W3C WoT architecture[1]; the WoT Architecture specification describes the abstract architecture for the W3C WoT. This abstract architecture is based on a set of requirements that were derived from use cases for multiple application domains. A set of modular building blocks is also identified whose detailed specifications are given in other documents. The architecture document describes how these building blocks are related and work together. Systems based on WoT architecture may cross different domains and integrate several vocabularies and ontologies.

[1]https://w3c.github.io/wot-architecture/

The WoT Thing Description (TD)[2] can be considered as the entry point of a Thing (much like the index.html of a Web site). Its specification is the core enabling technology. Different application layer protocols and media types can be described in a TD .

A TD abstracts the capabilities of individual Things into 3 categories called *Interaction Affordances*: *Properties* for sensing and controlling parameters, *Actions* for invocation of physical (and hence time-consuming) processes, and *Events* for the push model of asynchronous communication. A TD includes information models representing functions, transport protocol description for operating on information models, security information and general metadata about the device.

In summary, a WoT TD comprises the application logic requirements (e.g., values and alerts of a Thing). Devices are required to put a TD either inside them or at locations external to the devices, and to make the TD accessible so that other components can find and access them. As soon as available for a Thing, its TD can be used for flexible implementation and simulation (if required). To support the implementation, *WoT Scripting API* [3] specifies a common programming interface for Thing implementations as well as Consumer applications implemented by different programming languages.

b) System Modelling Tools and Relationship with Runtime: While IoT-ML is expressive enough to encompass IoT design, only with its modelling tools can exploit the full benefits of this formalism. One of the main goals of the IoT-ML modelling tools is to help deployment and connect it to deployed devices at runtime. This is accomplished through model transformation. Since IoT-ML, in BRAIN-IoT, has extensions specific to other IoT standards and technologies, the modelling tool offers transformation tools to go from one formalism to the other. The goal of such transformations is to bridge the gap between the runtime and the system model.

The IoT-ML models are made in the Papyrus modeller tool. The modelling tool is a typical Eclipse Eclipse Modeling Framework (EMF) [15] environment. It offers graphical editors, palettes to populate diagrams, and tree views to visit the hierarchical UML-based models.

An IoT-ML model, with TD stereotypes annotating its base elements, can be transformed to a TD in the JSON-based Serialization for Linked Data (JSON-LD) physical format. As a reminder, the TD files in JSON-LD are embedded on the real devices and polled at runtime. The authors of this paper believe this accelerates deployment of interfaces described in the system model. Furthermore, it bridges the gap between a natural way of describing architecture by the system engineers, and the text-based interface description by the developer. The importance to foster such a collaboration is explained in [16].

Using the same shared architecture model, the authors of this work can also transform the structure of the architecture

[2]https://w3c.github.io/wot-thing-description/
[3]https://w3c.github.io/wot-scripting-api/

into a sensiNact data model. The behaviors in the architecture, represented as state-machines, are transformed to equivalent sensiNact domain-specific language scripts. By then connecting to the sensiNact gateway, the data model and its sensiNact scripts can be run to monitor devices variables, and prototype system-level behaviors w.r.t. the runtime devices that should comply to the system model.

One last feature of the IoT-ML modelling tool is its ability to monitor its state machines. Although the connection between IoT-ML and sensiNact allows us to monitor variables and actuate devices, and therefore validate that the runtime is consistent with the system model, it is not always sufficient to understand the internal behaviors of the devices. For example, actuating a device, and noticing a variable change, may not be sufficient to understand what's happening for the human being. Therefore to provide human-friendly behavior explanations, it is possible to monitor state machines in an IoT-ML model. The state machines are animated and they mirror what's happening in the device state machines. What triggers the animations are messages that are sent to the IoT-ML modelling tool. Such messages are either sent by automatically generated code instrumentation points (i.e., during code generation itself of the state machine), or by any source that builds a string respecting the message format of the state machine monitoring tool.

The system model in IoT-ML, although connectable to existing runtimes, is not sufficient to develop the actual blocks that are to be deployed to form the runtime. Therefore its entities that represent software and hardware, must be refined, respectively, into a software architecture model and a physical architecture model. The next sections describe such models.

2) Physical Layer Modelling Approach: The BRAIN-IoT Physical Layer Modeling Approach allows the refinement of the digital twin model to an architectural model of the physical design including its functional and extra-functional properties. This model can be directly used by the device as well as the Integrated Circuits (ICs) designers as a reference model. This proposal follows a top-down model-based design approach composed of black box and white box models, called virtual twins. They are functionally equivalent to the physical IoT devices and can be used to serve different purposes. One model can be seamlessly replaced by the other, as they all share the same functional specification that represents faithfully the IoT device at its boundaries. They feature the functional and extra-functional properties of the IoT device (behavior, security, energy efficiency, reach, etc.).

a) Black box model: The black box (BB) model is an abstract representation of the IoT device. It is a simple service-oriented model that represents its functionality, regardless of the internal architecture that implements its behavior. The BB model can be considered as the functional reference of the end-device. It can be reused whatever the implementation choices, or even in case of replacement of the physical device by another one, as long as the functional specification and interfaces remain unchanged. It provides the functional contract, the other models or the physical device shall comply to. Executable, it is a non-ambiguous, repeatable and deterministic model of the end-device specifications. It abstracts the internal architecture of the end-device, as well as the embedded software. The BB model takes as input a file containing the data values that would be obtained from a real sensor operating in real conditions.

b) White box model: The second step of the top-down methodology is the creation of a white box (WB) model of the end-device representing the internal architecture of the device.

This architecture is composed of one or several sensors or actuators, a micro-controller, and one or several connectivity elements. Sensors are typically exposing an Inter-integrated-circuit (I2C) interface for digital data (or I/Os for analogue one), to let the microcontroller (MCU) read and write into registers to gather the data from the sensor, while the connectivity Internet Protocols (IPs) can be programmed through a Serial Peripheral interface (SPI) bus or through a Universal Asynchronous Reception and Transmission (UART) connection. The Hardware Abstraction Layer (HAL) of the MCU provides an API to access the hardware resources from the embedded software.

The White-box model represents this typical architecture that is described using the SystemC/TLM IEEE 1666 modeling language [17]. The model of the micro-controller typically includes a model of the embedded processor, such as Quick EMUlator (QEMU) or Instruction Set Simulators, and the models of all the peripheral blocks. This list obviously varies with each micro-controller, but usually includes the timers, the reset / clock / power controllers, the interrupt controller, and the hardware accelerators available for the part number. It also includes I/O models - UART, GPIO, I2C or SPI controllers - that are used to interact with the other elements of the end-device. The MCU is modelled to accurately represent the bus transactions initiated by the processor. The sensor model serves I2C requests issued by the micro-controller, and implements the behavior of the block, to react to the programming sequences. The connectivity model serves SPI or UART requests issued by the micro-controller, and implements the behavior of the block, to react to the programming sequences. In the current developments, the authors have decided to perform the communication as an abstraction of all the communication data path by issuing Hypertext Transfer Protocol (HTTP) requests to the network. When detailed communication protocol information is needed, the connectivity models can be connected to an elaborated communication model including communication medium such as LoRaWAN, serving protocol-specific commands. The WB model conforms to the functional contract of the end-device, as prescribed by the BB model.

c) Benefits of the Physical Layer Modelling Approach: The benefits BRAIN-IoT physical modeling language brings to the IoT domain are 1) Early verification of the embedded software, in charge of: data gathering from sensors; local processing (data formatting, data analysis, power management, payload construction, encryption, etc.); transmission of the encrypted data or metadata using the device connectivity capabilities (Bluetooth, LoRa, SigFox, etc.). 2) A system verification can be achieved in advance without the debug limitations inherent to physical devices, the models offer full inspection and observabilicould yty capabilities for debug and analysis; 3) The complexity of large-scale systems (from tens to thousands of end-devices) can be addressed by instantiating the appropriate number of models in the simulation platform; 4) The system reliability is increased, as the validation strategy of the end-device is strengthened by adding scenarios focusing on device robustness considering its interaction with system environment.

3) Formal Software Modelling Language and BRAIN-IoT Code Generator: The Behaviour, Interaction, Priority (BIP) Modeling language, introduced in [18], is the software modeling language in BRAIN-IoT. It supports the methodology for building systems from atomic components. It uses connectors, to specify possible interactions between components, and priorities, to select amongst possible interactions.

BIP is a highly expressive component-based language that supports the specification of composite, hierarchically structured components starting from the atomic ones. In BIP, the atomic components are finite-state automata having transitions labeled with ports and states that denote control locations where component waits for interactions. Ports are actions that can be associated with data stored in local variables and used for interactions with other components. Connectors relate ports from components by assigning them to a synchronization attribute, which may be either trigger or synchronous. A compound type defines a level of the hierarchy. It contains instances of component and connectors types (i.e., sub-components) with connection definitions and also priorities to schedule the interactions between these components. A compound component offers the same interface as an atom, so, externally, there is no difference between a compound and an atom. Inner ports from sub-components can be exported.

The BIP formalisms allow the rigorous specification and analysis of IoT systems components behavior. Moreover, the component-based approach supported by BIP facilitates portraying behavior with reusability, and maintainability features.

The BRAIN-IoT Code Generator relies on BIP language to describe the system behavior. It accepts as an input a formal specification of system architecture and system behavior using the BIP language, then it translates the system model into a set Java code artifacts. The generated Vanilla code could be simulated independently to the BRAIN-IoT execution platform called Fabric [19], and thus, the user could check the validity of the behavior specified at the BIP level. When the code is simulated and validated by designers, the code is wrapped in an envelope called bundles that fit the Brain-IoT execution platform. BRAIN-IoT Code Generator is an Eclipse plugin that includes two modules: (i) BIP language processor with the full support of BIP grammar and syntax checking, (ii) Java code generator of BIP model.

Using BRAIN-IoT Code Generator, the mapping of a formal BIP language integrates all the structures related to components development engineering such as composability and reusability. Moreover, the target language is independent of any technology; all the existing operating systems embed the processing ability of JAVA language.

The information flow of the components described above within the BRAIN-IoT modeling & validation framework is shown in Fig. 1. It is firstly responsible for designing IoT system application, which is going to be constructed, based on the actions and relations between the available devices - e.g., sensors, actuators, Cyber Physical Systems (CPSs) - and external services, e.g., weather forecast, open data, third-party IoT platforms, databases. The system level application is modelled along with the relevant IoT environment using BRAIN-IoT System Modeling Language (IoT-ML) through BRAIN-IoT modeling tool, representing the system-level behavior models and describing its self-adaptive behavior. Then, the IoT-ML model is refined to BIP model representing functional software-level components model. Finally, the BIP model is converted in source code as system behavior OSGi bundles through BRAIN-IoT Code Generator. The generated bundles are then released and stored in BRAIN-IoT repository, to be deployed and executed in the production environment. Furthermore, it also offers the ability to use development-time models to supervise running execution platform states. This solution enables monitoring the IoT devices' status and system configurations. This is possible, because BRAIN-IoT modeling tool supports the Models@runtime paradigm, allowing to synchronize the system's behavior and the real system. In addition, the generated OSGi bundles can be validated leveraging IoT device models developed with BRAIN-IoT physical layer modeling language to validate the correctness of the system behavior, before deploying in the physical world. In the BRAIN-IoT repository, stores the public/private modeling library that contains the system level models, BIP behavior models, and WoT TD for the IoT device/platform. In the BRAIN-IoT Service Artifacts library, there are System Behavior Artefacts generated from the BRAIN-IoT Code Generator.

VI. BRAIN-IoT Use Case

In this section, the methodology and the modeling framework proposed has been evaluated using BRAIN-IoT use cases [20].

A. Service Robotic

In a warehouse there are two zones: a loading area and a storage area. These zones are divided by an automatic door. This door has a QR code attached on each side which is legible only when the door is closed. In the loading area there are 3 carts each with a QR code attached (different for each case) and three robots responsible for warehouse management are also located. The robots are equipped with vision cameras that allow reading the QR codes they find. The model of the robots is a *RB1* base from Robotnik company. They are capable of detecting and raising carts and also transport them from one point to another in an autonomous way. The warehouse is controlled through an Orchestrator that assigns the carts to the storage areas and orchestrates the robots. On the other hand, the storage area is divided into 3 sub-zones (A, B, C) where

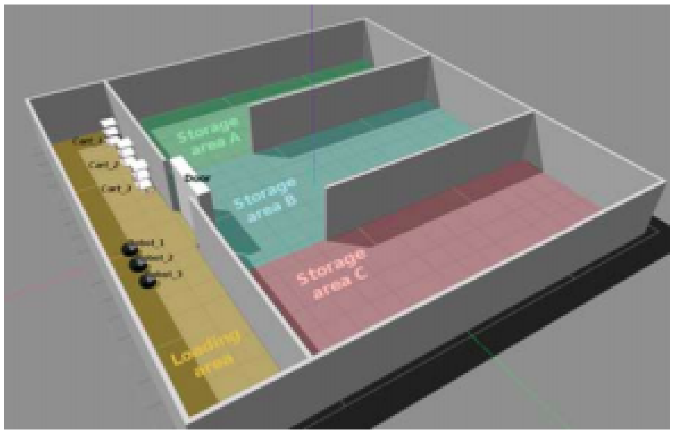


Fig. 3. Simulation environment of the Use Case

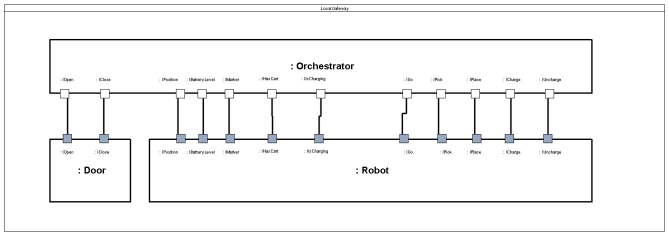


Fig. 4. Example of Composite Structure Diagram in IoT-ML Describing Robot, Door, and Orchestrator

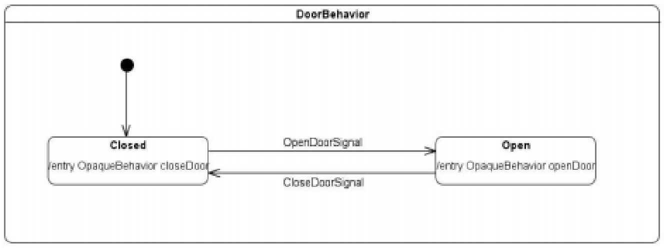

DoorBehavior
Closed
/entry OpaqueBehavior closeDoor
OpenDoorSignal
CloseDoorSignal
Open
/entry OpaqueBehavior openDoor

Fig. 5. Door Behavior

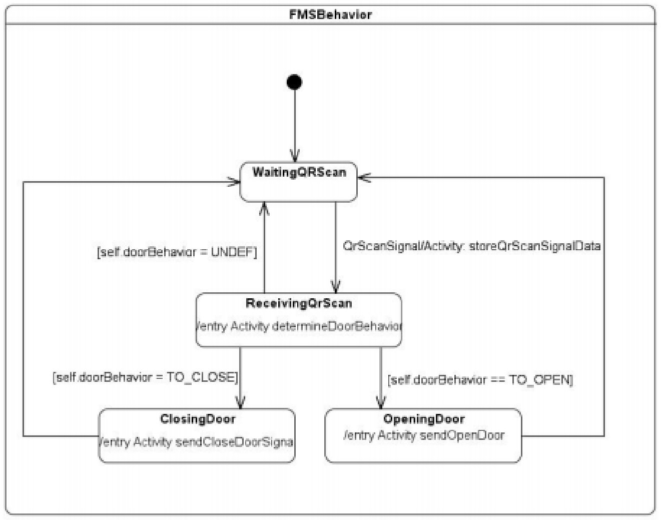

FMSBehavior
WaitingQRScan
[self.doorBehavior = UNDEF]
QrScanSignal/Activity: storeQrScanSignalData
ReceivingQrScan
/entry Activity determineDoorBehavior
[self.doorBehavior = TO_CLOSE]
[self.doorBehavior == TO_OPEN]
ClosingDoor
/entry Activity sendCloseDoorSignal
OpeningDoor
/entry Activity sendOpenDoor

Fig. 6. Fleeet Management System Behavior

carts can be stored. The Fig. 3 shows the scenario of the use case in the simulation environment. The logic of the system is that the robot takes the cart from the loading area and places it in the storage area, on the way, it detects an elevator door. The robot scans the QR code of the door. It then sends the QR code to a Fleet Management System (FMS) with a door open request, FMS opens the door. After the door opened, the robot will send again the QR code to the FMS with a door close request.

Firstly, the authors modeled the system level components and the interaction interfaces using IoT-ML with the BRAIN-IoT as shown in Fig. 4, in which, the FMS is represented by an Orchestrator, with a robot and door connected to an Orchestrator that sends commands to both devices. Moreover, the behavior of the FMS is modeled using BRAIN-IoT modeling language with BRAIN-IoT modeling tool as shown in Fig. 6, and the behavior of the door in Fig. 5

Then the system models are refined as BIP software models and as the inputs of the code generator, the generated artefacts are deployed in the robot. While the robot is running, its status and warehouse coordinates configurations will be monitored through the monitoring tools. The software architecture is as shown in Fig. 7. As a reminder, the EventBus is a service provided in BRAIN-IoT for the communication in a distributed environment and the Robotic Operating System (ROS) Edge Node is an adaptor deployed in the robot for communicating with other applications deployed in the BRAIN-IoT Fabric through the EventBus. Orchestrator Behavior represents the system level behavior and orchestrates the robot and the door. ROS Edge Node provides the adaptor for ROS environment to communicate with other components via eventBus, the current version is extended with Behavior Translator to connect Papyrus using User Datagram Protocol (UDP). Papyrus will provide the visual realtime monitoring of the robot state transitions with the state machine, it is integrated with ROS Edge Node. To demonstrate the Models@Runtime feature, here the authors monitored two aspects: one is the configuration of the warehouse coordinates, another is the runtime status of the robot.

a) Runtime Robot Status Monitoring: The sequence of the workflow is as following: 1) ROS Edge Node receives the command events from Orchestrator. 2) ROS Edge Node sends the command to the robot, meanwhile, it converts the

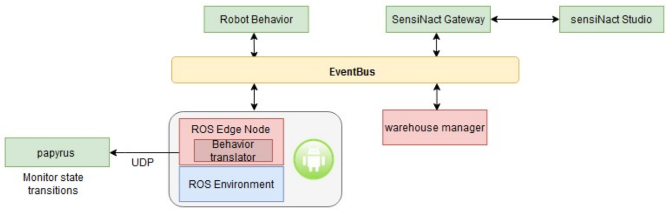


Fig. 7. Service Robotic Monitoring

command to the messages can be received by Papyrus. 3) When Papyrus receives the command transition message, it will dynamically display the state change.

b) Runtime Warehouse coordinates monitoring: Before the application is deployed, the end-user will configure the warehouse through SensiNact studio, these values will be delivered to warehouse manager through SensiNact Gateway and EventBus, then stored in three tables, which are picking points table storing the coordinates in the loading area, storage points table storing the coordinates in the storage area, and cartStorage points table storing the corresponding place point in the storage area of each cart. On the other side, during runtime, whenever a robot changes the attribute value in the table due to the mission of moving a cart, the warehouse manager will send an update event to sensiNact Gateway, then be delivered to sensiNact studio. Hence, the updates will be reflected in the sensiNact studio.

B. Critical Water infrastructure management

The services of water supply are associated to a series of infrastructures that are considered, in accordance with European norms, as critical, and therefore, they are bound to a series of conditions for their development, especially in the technological aspect.

In the water management sector, most of the processes are associated to disperse infrastructures in large and varied geographical sites, with numerous interactions with other elements and services related to the human activities. The sharing of information in a safe and efficient manner is a challenge to optimize the actions in urban surroundings to simplify and improve the citizen's life. The development of models and their implementation in multi-access platforms (internal usage, clients, responsible entities, etc.), constitutes a knowledge and technological challenge for the sector. They require development for the correlation of data, its analysis and the creation of indicators and processes for a better usage of the infrastructure's maximum potential.

In the water management use case, the scenarios aim to leverage prediction models (based on the collected data), to: increase the security of water supplies, optimize the underlying costs, enhance the services for end-users and connect the infrastructure to other urban services. Analyzing gathered data will help to create more accurate indicators for decision-making, and for real-time, smart and adaptive control procedure and generally more efficient and automated business processes. For evaluating scenarios and developments carried on during the project, a mock-up called MEDUSA was built in the facilities. In the Critical Infrastructure of Water Management System, four use cases have been defined to evaluate the main features, concepts and developments of the BRAIN-IoT project. In this paper for space reasons, only the Resilience Use Case is described. The objective of this Use Case is to validate the resilience of the system in case of hydraulic failures. Normally, the hydraulic system works according to a defined consumption curves at the entrance of the water meters sections. These curves represent the real

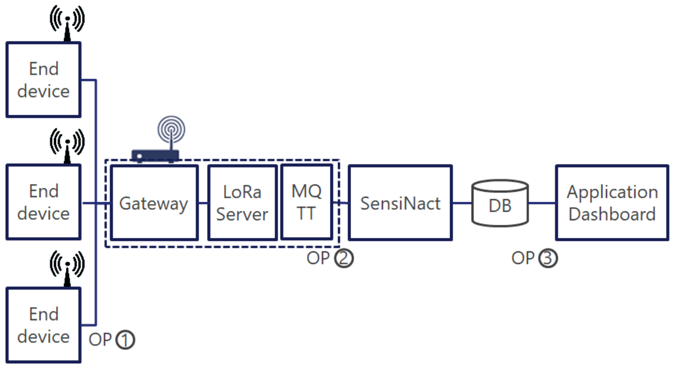


Fig. 8. Model of critical water infrastructure

consumption in various points of Coruña city (Elevado, Cola and Cabecera). The curves are obtained with the opening and closure of the electric valves of every section of water meter, simulating the real consumption of the customers. The resilience of the system is validated in terms of guarantee that the consumption can be ensured in those points. One scenario is to simulate the failure of an electric valve in a pipe and to ensure that BRAIN-IoT platform can recirculate the water for another pipe, controlling the electric valves that allows that, and according to the Detection and Responsive System (DRS) that have been trained for these real failures. Another possible scenario is to simulate the failure of a flow meter and to ensure that BRAIN-IoT platform can recirculate the water for another section with the same goal of ensuring the consumption curves. DRS uses Machine Learning techniques for the detection of failures. The responsive part is based on defined rules, acting as expert system. In the scenario described above, there are three main parts involved: 1) data gathering from the IoT devices 2) An algorithm for detecting the anomalies/failure 3) Control system for reacting the abnormal situation. Point 2 and 3 will be the main components of DRS.

In the critical water infrastructure architecture (see Fig. 8), heterogeneous sensors are placed in various remote locations to collect and transmit data through a LoRaWAN network, obtaining a huge amount of data to be used for training anomalies detection algorithm with the limited number of physical devices. The authors have developed the simulated IoT devices models to generate the data. The system model performs data gathering (from an input table), data encapsulation and encryption, and data transmission to the network through HTTP requests, as an abstraction of the complete LoRaWAN communication data path: gateway + LoRa server / MQTT. Then, the data are processed and stored in the edge database. System security and robustness against vulnerabilities and attacks are guaranteed.

Fig. 9 shows an example of the simulation results, at the system level, obtained by the data transmission, data recovery, processing and display in the EMALCSA application dashboard of a water meter end device. The curves represent, for each device: i) the percentage of valve openness, ii) the water flow measured as output of the valve, iii) the saturation

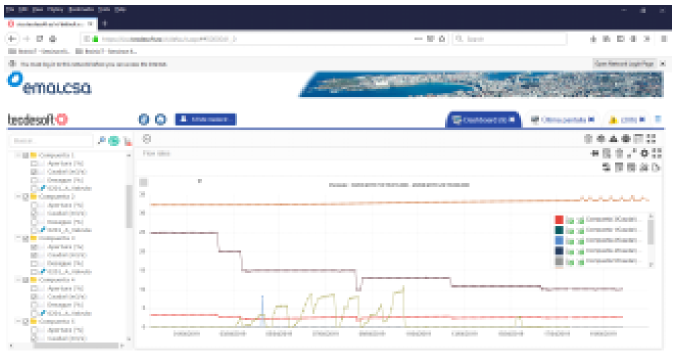


Fig. 9. Plot, in end user dashboard, of the data gathered by a water meter end device model

of the pipe.

The authors also aim implementing the control system in point 3 using the BRAIN-IoT Modeling & Validation Framework, following the proposed modeling methodology to get the intended system control models, then, generate the application artefacts. Its correctness will be validated in the simulated devices, hence the risk of physical critical infrastructure damage could be reduced before deploying to the real environment.

VII. Conclusion and Future Works

This paper has presented the Model Based Methodology and Framework proposed for design and Management of next-gen IoT Systems. This solution provides a Model-Based Engineering (MBE) approach to ease the development of the IoT systems. It offers a system-level model, which captures the system functionalities and behaviors to help refinement of the software-layer modelling; it facilitates the linking towards real devices and external services through meta-data representation in WoT TD; the application code is generated from model for monitoring and controlling the IoT infrastructure; it supports the system application validation leveraging the simulated IoT devices developed with the BRAIN-IoT physical layer modeling language; finally, it allows monitoring the IoT system behaviours and its configurations in a human-friendly graphical manner through the Models@Runtime approach at the execution time.

As future work, the Modelling & Validation framework will be extended to also support AI modelling. In particular the authors will evaluate the feasibility of using system modeling languages to either describe finely machine learning algorithms, or their deployment, in the goal of accelerating development and promoting interoperability between AI (sub)-systems. Furthermore, the Critical Water Infrastructure management use case will be further developed including the control part and the associated actuators models to demonstrate the complete functionalities provided by the modeling framework. Furthermore, the authors would provide a survey to some users and get some evaluations to demonstrate the benefits brought by the solutions proposed in this paper.

Acknowledgment

The work presented here was part of the project "Brain-IoT-model-Based fRamework for dependable sensing and Actuation in iNtelligent decentralized IoT systems" and received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780089. The authors thank Robotnik Automation S.L.L. and EMALCSA for providing the environment, real and simulated to test the solution.

References

[1] G. P. Fettweis, "The tactile internet: Applications and challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.
[2] O. Vermesan and J. Bacquet, *Next generation Internet of Things: Distributed intelligence at the edge and human machine-to-machine cooperation*. River Publishers, 2019.
[3] AGeSys Consortium. AGeSyS Atelier de Genie Systeme. [Online]. Available: https://www.aerospace-valley.com/sites/default/files/encart_html/index.html
[4] Y. Sun, G. Memmi, and S. Vignes, "A model-based testing process for enhancing structural coverage in functional testing," in *Complex Systems Design & Management Asia*, 2016.
[5] S. Dhouib, A. Cuccuru, F. Le Fèvre, S. Li, B. Maggi, I. Paez, A. Rademarcher, N. Rapin, J. Tatibouet, P. Tessier *et al.*, "Papyrus for iot—a modeling solution for iot," *Proceedings l'Internet des Objets (IDO: Nouveaux Défis de l'Internet des Objets: Interaction Homme-Machine et Facteurs Humains. Paris, France*, 2016.
[6] P. Roques, "Mbse with the arcadia method and the capella tool," in *Proceedings of ERTS 2016*, Toulouse, France, 2016.
[7] Z. M. Bzymek, M. Nunez, M. Li, and S. Powers, "Simulation of a machining sequence using delmia/quest software," *Computer-Aided Design and Applications*, vol. 5, no. 1-4, pp. 401–411, 2008.
[8] M. Bauer, M. Boussard, N. Bui, F. Carrez, C. (SIEMENS, J. (ALUBE, C. (SAP, S. Meissner, A. IML, A. Olivereau, M. (SAP, W. Joachim, J. Stefa, and A. Salinas, "Internet of things – architecture iot-a deliverable d1.5 – final architectural reference model for the iot v3.0," 2013.
[9] H. Espinoza, D. Cancila, S. Gérard, and B. Selic, "Using marte and sysml for modeling real-time embedded systems," *Model-Driven Engineering for Distributed Real-Time Systems: MARTE Modeling, Model Transformations and their Usages*, pp. 105–137, 2013.
[10] W. Emmerich and N. Kaveh, "Component technologies: Java beans, com, corba, rmi, ejb and the corba component model," in *Proceedings of the 8th European software engineering conference*, 2001.
[11] Object Management Group, "Dds for lightweight ccm (dds4ccm)," 2009.
[12] B. Elvesæter, C. Carrez, P. Mohagheghi, A.-J. Berre, S. G. Johnsen, and A. Solberg, "Model-driven service engineering with soaml," in *Service Engineering*, 2011, pp. 25–54.
[13] Object Management Group, "Unified component model for distributed, real-time and embedded systems," 2020.
[14] L. Gürgen, C. Munilla, R. Druilhe, E. Gandrille, and J. Nascimento, *sensiNact IoT Platform as a Service*, 08 2016, pp. 127–147.
[15] F. Budinsky, "The eclipse modeling framework," *Doctor Dobbs Journal*, vol. 30, pp. 28–32, 08 2005.
[16] V. C. Pham, S. Li, A. Radermacher, S. Gérard, and C. Mraidha, "Fostering software architect and programmer collaboration," in *Proceedings of the 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2016.
[17] "IEEE 1666-2011; IEEE standard for standard systemc language reference manual," standard, 2011.
[18] A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T.-H. Nguyen, and J. Sifakis, "Rigorous component-based system design using the BIP framework," *IEEE Software*, vol. 28, no. 3, pp. 41–48, May 2011.
[19] R.Nicholson, T.Ward, D.Baum, X.Tao, D.Conzon, and E.Ferrera, "Dynamic fog computing platform for event-driven deployment and orchestration of distributed internet of things applications," pp. 239–246, 2019.
[20] E. Ferrera., X. Tao., D. Conzon., V. S. Pombo., M. Cantero., T. Ward., I. Bosi., and M. Sandretto., "Brain-iot: Paving the way for next-generation internet of things," in *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security - Volume 1: IoTBDS,*, INSTICC. SciTePress, 2020, pp. 470–477.

AUTHOR INDEX