# Conceptualizing a Framework for Cyber-Physical Systems of Systems Development and Deployment

Jessica Díaz, Jennifer Pérez, Jorge Pérez, Juan Garbajosa

## ABSTRACT

Cyber-physical systems (CPS) refer to the next generation of embedded ICT systems that are interconnected, collaborative and that provide users and businesses with a wide range of *smart* applications and services. Software in CPS applications ranges from small systems to large systems, aka. Systems of Systems (SoS), such as smart grids and cities. CPSoS require managing massive amounts of data, being aware of their emerging behavior, and scaling out to progressively evolve and add new systems. Cloud computing supports processing and storing massive amounts of data, hosting and delivering services, and configuring self-provisioned resources. Therefore, cloud computing is the natural candidate to solve CPSoS needs. However, the diversity of platforms and the low-level cloud programming models make difficult to find a common solution for the development and deployment of CPSoS. This paper presents the architectural foundations of a cloud-centric framework for automating the development and deployment of CPSoS service applications to converge towards a *common open service platform* for CPSoS applications. This framework relies on the well-known qualities of the *microservices* architecture style, the *autonomic computing* paradigm, and the *model-driven software development* approach. Its implementation and validation is on-going at two European and national projects.

## CCS Concepts

**•Software and its engineering→Software organization and properties→Software system structures →Distributed systems organizing principles →Cloud Computing**

## Keywords

Cyber-Physical Systems; Cloud Computing; Software Architecture; Microservices; Model-driven development

## 1. INTRODUCTION

Society demands more intelligent, more energy-efficient and more comfortable homes, hospitals, offices, factories and cities. Cyber-Physical Systems (CPS) refer to ICT systems (sensing, actuating, computing, communication, etc.) embedded or software integrated in physical objects, interconnected (including through the Internet) and providing citizens and businesses with a wide range of smart applications and services: healthcare, smart home, smart energy/water grids, smart logistics, and smart cities [8][19]. Therefore CPS can grow from small systems, such as smart home or vehicles, to the above-mentioned large systems. These systems are known as Systems of Systems (SoS), since they are composed by other heterogeneous systems geographically extended that pursue a unified goal and leverage an emergent behavior [16][21]. These large CPS are known as Cyber-Physical Systems of Systems (CPSoS).

CPSoS are composed of devices, most times smart devices, with embedded sensors that continuously acquire information from the physical environment, which is stored as historical data and/or is processed often in real time to make decisions and act on the physical world through actuators. CPS intend to be *aware* of the physical environment enabling effective and fast *feedback control loops* between sensing and actuation, possibly with cognitive and learning capabilities [19]. CPS are ever more interconnected through the Internet of Things (IoT), which encompasses the extension of the Internet into the physical realm, resulting in a global network that interconnects thousands or even millions of "things" [23]. This fact has generated an important economical and societal impact [13][26][29], and CPSoS are continually increasing the number of resources. Therefore, CPSoS are required to be *scalable* to remain effective when there is a significant increase in the number of resources or/and users [5].

Cloud computing offers self-provisioning *infrastructure as a service* which makes it the natural candidate for supporting the scalability and data processing, storing and analyzing needs of CPSoS [2][3][7]. In addition, most cloud providers support dynamic allocation of the resources required by an application to match performance requirements and satisfy service-level agreements. In this way, CPSoS can benefit from the almost unlimited resources of clouds not only for storing data, but also for hosting and delivering their services—e.g. sensing, analysis and actuation tasks—as Software as a Service (SaaS) applications.

However, the diversity of platforms and the low-level cloud programming models and interfaces make the development and deployment of CPSoS applications difficult. Research in constructing specific frameworks—integrated set of software artefacts (e.g. components) that collaborate to provide a reusable architecture for a family of related applications [28]—for rapid creation of these applications and their deployment on cloud infrastructures could become a driver for mass market users [13]. In fact, the survey conducted by Botta et al. [2] identifies the need to converge towards a *common open service platform* for providing end-users with APIs to develop intelligent sensing and actuating applications deployed in the cloud, rather than having to deploy these infrastructures by themselves *"—which proved to be a time-consuming and tedious task that dramatically slows innovation"*. Current research is focused on cloud-centric frameworks which provide support for device monitoring, storage resources, analytics tools, visualization platforms and client delivery [13] and their

integration with new architectural styles, paradigms and approaches. In this paper, we focus on *microservices* [20], *autonomic computing* [14], and *model-driven development* (MDD) [1], which have been individually applied to CPS/IoT (see respectively [9], [4] and [7]) and demonstrated qualities, such as, scalability and modifiability, self-managing, and automation, among others. This work takes a step forward by integrating these approaches on a cloud-centric framework for CPSoS.

This paper presents the architectural foundations of a framework for automating the development of CPSoS SaaS applications and their deployment on cloud infrastructures based on the microservice architectural style, autonomic computing, and MDD. The CPSoS Framework defines the services that allow to build CPSoS-based applications. Implicitly, the framework defines the architecture of a global CPSoS solution that mainly consists of a set of services that allow to implement typical autonomic loops [14]: *data services* to read information and store it; *analysis services* of data; and *planning* and *execution services* to react in the physic environment. In addition, most of the involved stakeholders in CPS development are taken into account, such as data suppliers, analysis tool developers, and utilities/retailers. The SaaS applications can be deployed to the cloud to be accessible by other developers and final users. As a result, the CPSoS Framework converges towards the construction of a common open service platform to develop third-party CloudCPS-based applications.

Some of the foundations of the **CPSoS Framework** are the partial result of previous work on two ITEA2 projects called IMPONET[1] and NEMO&CODED[2], the research initiated in a national project called MESC[3], and currently starting to be developed in a large Horizon2020 project called CPSELabs[4]. The novelty compared to our previous work is the conceptualization of a framework itself as platform to develop CPSoS applications as well as the evolution of previous work in MDD and autonomic computing [31] to support microservices and cloud-based applications.

This paper is structured as follows: Section II describes the background and Section III related work. Section IV presents the conceptualization of the CPSoS Framework. Finally, conclusions and further work are described in Section V.

# 2. BACKGROUND

## 2.1 Cyber-Physical Systems

Cyber-physical systems (CPS) are systems where real-time or *cuasi* real-time computing elements and physical systems interact tightly. "The most challenging class of cyber-physical systems are *cyber-physical systems of systems* which are characterized by being spatially distributed, having distributed control, supervision and management with partial autonomy of the subsystems, are dynamically reconfigured on different time-scales and can show emerging behaviors"[5].

Future CPSoS applications are expected to be more transformative than the IT revolution of the past three decades [28]. In the last few years there has been an active research and industrial community on defining and facing with the CPS's challenges to ensure competitiveness in this emerging field. In Europe, an example of this is the FP7's CyPhERS[6] that aimed to define the strategic research and innovation agenda about CPS. In its last report, this project stated, among others, the following challenges: **Ch. 'Service vs. Product':** *While cyber-physical systems require substantial investments in equipment and infrastructure, at the same time they facilitate the establishment of business models focusing on the provision of a service rather than manufacturing of a product.* **Ch. 'Multi-Domain Modeling':** *there is no established body of knowledge on how to adequately model all the relevant aspects of cyber-physical systems – especially with respect to useful combinations of those aspects and the required level of abstraction.* **Ch. 'Autonomy':** *Cyber-physical systems are self-controlling or even self-adapting and self-optimizing systems, leading to increasing levels of highly automated or even autonomous behavior in the components of those systems, as well as in their collaboration* [6]. CyPhERS also makes some recommendations based on the need of building reference architecture for CPS and Open Cyber-Physical Systems Platforms (OCPSP) [12], i.e. common cross-domain engineering platforms to develop CPS which support reusability of components, repositories, and tools ("*open platforms* for every body and *open interfaces* for everything") [12]. Later, European R&I programme Horizon 2020 included topics for reducing development time and maintenance costs of such systems. Specifically, the proposal of a *European research and innovation agenda on cyber-physical systems of systems 2016-2025* emphasizes the following challenges: engineering support for the design-operation continuum of cyber-physical systems of systems; and cognitive cyber-physical systems of systems.

## 2.2 Autonomic Computing

To satisfy the self-management requirements of CPSoS to being more autonomous (context-aware) applications, we rely on the fundaments of *Autonomic Computing* (AC) [14][17]. Horn [14] defines autonomic systems as software systems that mostly operate without human or external involvement according to a set of rules or policies; in other words, the systems are self-managed. IBM proposed the MAPE-K control loop for supporting autonomic computing. According to the MAPE-K loop, resources to be managed are composed of a set of sensors that provide information about the current state of the resources. The model implements the following: the monitoring of the information (Monitor); the analysis to detect symptoms that need corrective action (Analyze); the planning of the action required to change the current state of the resource according to a set of goals or policies (Plan); and the execution of the plan through a set of effectors (Execute). These actions are operated over a knowledge base. The MAPE-K loop model offers the advantage of isolating the main concerns that any autonomic process has to provide, and thus, also offers the main concerns that CPSoS applications' architecture should include.

## 2.3 Cloud Computing

Cloud computing emerges as the natural candidate to support the scalability required by CPSoS [2][3][7]. The European technology platform dedicated to software, services, and data, NESSI (NEtworked Software and Services Initiative), also identifies cloud

[1] Intelligent Monitoring of Power NETworks http://innovationenergy.org/imponet/

[2] NEtworked MOnitoring & COntrol, Diagnostic for Electrical Distribution http://innovationenergy.org/nemocoded/

[3] Platform for Monitoring and assessing the Efficiency of distribution systems in Smart Cities, http://exit.udg.edu/mesc/

[4] European Union-funded initiative supporting European businesses http://www.cpse-labs.eu/

[5] IoT/CPS Expert Group. H2020 PICASSO Project http://www.picasso-project.eu/iotcps-expert-group/

[6] Cyber-Physical European Roadmap and Strategy. Available on http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=109303

as an opportunity for CPS [24].Cloud Computing is a new model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, platforms, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing providers, such as Microsoft, Amazon, and Google, offer a variety of computing services built on top of their own infrastructure, which are managed in dedicated globally distributed data centers that offer high availability, resilience, and scalability. Cloud perceives of all tasks accomplished as a "*service*" rendered to users: Infrastructure as a Service (**IaaS**); Platform as a Service (**PaaS**); and Software as a Service (**SaaS**).

## 2.4 Microservices

Many companies are finding that making their applications highly available, scalable, modifiable, and agile is still challenging. Microservices is an emerging architectural style (aka. paradigm) for the development of distributed systems that intends to deal with these issues. "*A microservices application is decomposed into independent components called microservices, that work in concert to deliver the application's overall functionality*" [27]. This is known as *componentization via services* [20]. The principle of the microservices architecture is akin to the Unix principle: *Do one thing and do it well* [20]. Each microservice has well-defined contracts (typically RESTful) for other microservices to communicate and share data with it. As microservices can be deployed independently of one another and are loosely coupled, they can scale independently and are easily replaceable and upgradeable which supports the rapid/agile and reliable evolution of an application.

## 2.5 Model-driven Development

MDD is a software development approach in which models can be managed and transformed to facilitate and automate tasks involved in the development and evolution of software systems by employing high-level abstractions. The primary technical advantages claimed by MDD proponents are improvements in productivity, portability, maintainability, and interoperability [1]. Increasingly, MDD has been widely adopted in the industry as reported in [15][30]. Current quantitative analysis shows that the main advantage of MDD is code generation [25] and documenting a good software architecture [32]. This work takes advantage from both of them.

In MDD, models must be described by a well-defined modeling language. Languages that are specific to a domain are referred to as *domain-specific languages* (DSL) [18]. In last few years, there is an increasing interest in applying MDD and DSLs to IoT and, in general, CPS.

# 3. RELATED WORK

Nowadays, the research community is working on integrating cloud and autonomic computing, MDD, and microservices with CPS/IoT. A good example are the works of Cavalcante et al. [3] and Botta et al. [2] that perform a study mapping and a survey, respectively, about integrating Cloud and IoT. These authors review some projects which deserve to be highlighted: IoTCloud[7] and OpenIoT[8]. These projects focus on managing sensors and their messages, filtering these messages, and processing events for interested applications, to providing utility-based IoT services. In the same way, but in specific context of smart cities, the work [11] proposes new cloud service models. Specifically, this work proposes a *City Platform as a Service*, which expands the traditional PaaS layer to foster the development of applications by providing a set of specialized middleware services—e.g. data storing, processing, and analysis. Finally, the work of Gubbi et al. [13] also presents a cloud-centric framework which is near to integrate a MAPE-K loop by providing support for *"(1) reading data streams either from sensors directly or fetch the data from databases, (2) easy expression of data analysis logic as functions/operators that process data streams, and (3) if any events of interest are detected, outcomes should be passed to output streams, which are connected to a visualization program"*.

In the integration of MDD into the development of IoT applications, it is possible to highlight the work by de Farias et al. [7]. This work presents a cloud-based IDE (Integrated Development Environment) which relies on MDD, particularly MDA [22] for developing IoT applications which are compiled and/or simulated in the cloud to be deployed then into the devices. A cloud-based IDE allows developers to have the same environment to generate *sensor code images*, thus avoiding incompatibility regarding compiler versions or differences in the IDEs. Although it is not directly related to CPS/IoT, it deserves to be mentioned the MODAClouds EU project and related key results (e.g. MODACloudML [10]) to model and automate the provisioning and deployment of cloud-provider independent applications.

Finally, the work of Familiar [9] postulates microservices as a solution to develop IoT applications in the cloud Microsoft provider called Azure. However, none of these works have researched the advantages of integrating cloud and autonomic computing, MDD, and microservices with CPS.

# 4. CPSoS FRAMEWORK

In view of the promising qualities of cloud, microservices, autonomic computing, and MDD, this paper presents the conceptualization of a framework that integrates all of them. The framework, named CPSSos Framework, aims to improve the development and deployment of CPSoS-based applications thorugh the incorporation of the abstraction and automation of MDD, the self-management of autonomic computing, the self-provisioning and scalability of cloud computing, and the scalability, modifiability and agility of microservices, among others qualities. But the question is, how to do it? This work presents the conceptual overview of this framework from an architectural view perspective. The framework has been defined based on previous work and the current experience results obtained from the projects CPSELabs and MESC. This conceptualization can be used as a guidance for the construction of cloud-centric frameworks that pursue to automate the development of CPSoS SaaS applications and their deployment on Cloud infrastructures. By extension, this could be a seed of a future common open service platform to develop third-party Cloud CPSoS-based applications.

## 4.1 Conceptual overview

The CPSoS Framework is designed over a cloud platform to take advantage from the almost unlimited resources of clouds for hosting and delivering services. The framework defines: (i) the services and their coordination for achieving system-level objectives, i.e. the SaaS that it should provide, (ii) the stakeholders

[7] https://sites.google.com/site/opensourceiotcloud/ (2014)
[8] http://www.openiot.eu/ (2014)

that are involved in the framework, and (iii) the software architecture that a global CPSoS solution implements (see Figure 1). The CPSoS Framework defines the main services that CPSoS applications should include. These services are provided through a set of SaaS, since the framework is deployed on the cloud. These SaaS are the following (see Figure 1): *data services* to read measurements coming from sensors and store them in databases of public and/or private clouds; *analysis services* to identify symptoms, problems or anomalies in the measurements stored by the previous service; *planning services* to schedule a plan to react to the symptoms, problems or anomalies identified by the previous service; and finally *execution services* to run plans. It is necessary to emphasize that this service conceptualization is compliant with the MAPE-K Loop.

Through the specification of these services, the framework provides a unified management of most stakeholders who <<develop>> or <<use>> CPSoS services in the cloud (see Figure 1). The stakeholders are the following: *data providers* that develop data services based on their knowledge on IoT and wireless sensor networks as well as cloud storage; *analysis tool developers* that analyze data to extract new knowledge mainly based on their expertise on technologies such as complex event processing (CEP), big data, and machine learning, among others; finally, *operators, utilities and retailers* that use their knowledge in a particular domain to define the goals that a CPSoS has to reach and proceed with actuation plans to react to the symptoms, problems or anomalies that occur in the system, and thereby to obtain the required information to implement a successful MAPE-K Loop (see Section 2.2). As a result, the framework incorporates the notion of MAPE-K loops in such a way that the goals defined by the CPSoS are to be constantly maintained through these control loops.

The stakeholders model these services, according to their expertise, through a MDD process (see Figure 1). This MDD process uses CPSoS domain-specific models that facilitate the specification of the MAPE-K that the CPSoS SaaS applications must follow according to this framework. The MDD process also alleviates the uncertainty of SoS, which undergoes the common situation of adding changes in the plans and goals of such systems, or extending them. This improvement is due to the automation of the code generation and deployment from these CPSoS domain-specific models. In order to support the modelling and automatic code generation of CPSoS domain-specific models, it is required the specification of a metamodel (see Label 1 in Figure 1) to define the abstract syntax of the modeling language, in such a way that the stakeholders can use these language to define models (see Task A in Figure 1). To take advantage of model-to-code transformations, the specification of code generation patterns are necessary (see Label 2 in Figure 1), in such a way that the stakeholders can apply these transformations (see Task B in Figure 1) to the previously defined models to generate code. Finally, this code could be ready for deployment and execution (see Task C in Figure 1).

The metamodel and model-to-code transformations (generation patterns) are part of the framework for the usage of the stakeholders as many times as they need. The CPSoS Framework incorporates to the cloud our previous work about using MDD in CPS [31] with the corresponding migrations of these transformations. This previous work [31] defined a metamodel for CPS and their code generation patterns. It is called MindCPS (doMaINmoDel for CPS) DSL. The metamodel MindCPS is part of the results of two ITEA2 projects called IMPONET[1] and NEMO&CODED[2]. The metamodel provides the modelling primitives to model data services, analysis services, planning services and execution services, and transform these models to code. In this way, one is able to specify all of a subset of the services of a MAPE-K control loop for a CPSoS and generate the backbone code. In this work, we take a step forward migrating it to the cloud, in such a way that the generated code constitute the SaaS to be deployed in the cloud by following the microservice style (see Section 4.2). These SaaS are transversal to the variety of CPSoS domains, and thus, can be used by any of them. The MindCPS supports the specification and definition of the main concepts of a CPSoS and its autonomous behavior—i.e. sensors, measurements, events, problems, actions and plans.
Figure 2 shows a fragment of a MindCPS that models some of the services of a CPS called "Arboleda Demonstrator". It was deployed in a building located on the UPM South Campus, Madrid, Spain. The Arboleda demonstrator was equipped with various artefacts; including sensors, gateways, and actuators. The sensors included power, water, humidity and temperature meters and a presence detector. The actuators included a HVAC system controller and a photovoltaic generator PLC connected to a solar panel.

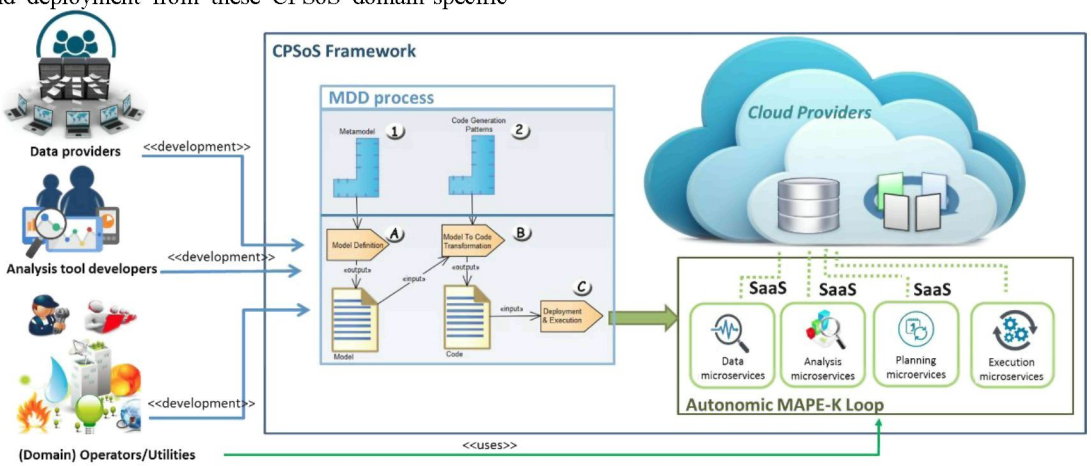


**Figure 1. Cloud-centric model-driven framework for CPS SaaS overview**

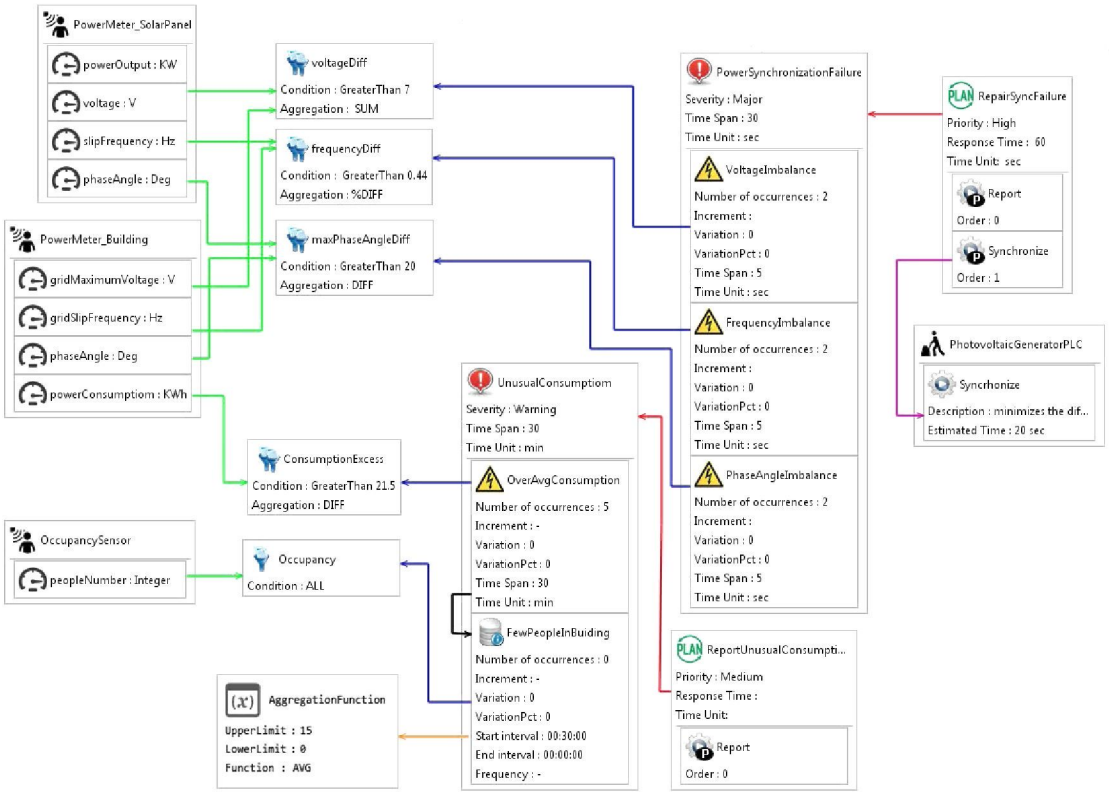


**Figure 2. A fragment of the MindCPS model of the CPS deployed at the Arboleda demonstrator.**

A set of SaaS applications modeled in Figure 2 are described as follows: The*PowerMeter_Building* measures and records *power consumption* and the *PowerMeter_SolarPanel* measures *power output* and *phase angle*, among others. The resulting measures are filtered to detect symptoms and problems related to unusual consumption levels in the building, as well as synchronization failures[9] (see *Unusual Consumption* and *Power Grid Synchronization Failure*). For example, the problem of *Power Grid Synchronization Failure* is detected when the limits for synchronization are exceeded (e.g. maximum voltage difference is 7% (see *voltageDiff*). This problem is solved through a plan that consists of an action that synchronizes—i.e. minimizes the difference in voltage between the corresponding phases of the solar panel output and grid supply—through a PLC connected to the solar panel (see the plan *Repair Sync Failure*).

Finally, the CPSoS Framework provides a software architecture which CPSoS solutions must be compliant with. This compliance leverage the qualities that the integration of cloud, autonomic computing and microservices promises.

## 4.2 Software Architecture

The initial conceptual definition of the software architecture was based on a previous reference architecture for CPS based on three-tier monolithic applications [31] (see Figure 3). This architecture implements MAPE-K control loops in such a way that one component contains all the logic of a control loop that implements a self-managing requirement. However, after some pilot projects, we identified some problems.

Monolithic applications puts all its functionality into a single process [20]. Thus, monolithic architectures may hamper scalability and flexibility of applications as *each tier is still its own monolith, implementing diverse functions that are combined into a single package deployed onto hardware pre-scaled for peak loads* [27]. On the one hand, it is required to face the scalability need of CPSoS. On the other hand, this architecture forced have cross-functional teams to develop CPS applications that implemented services for monitoring, analyzing, planning and executing (see Figure 3). Since diverse functions are necessary to develop a CPSoS applications, a cross-functional team with inter-disciplinary knowledge is required—such as, knowledge about devices for monitoring, about big data and machine learning for data analysis, etc. To deal with scalability, the migration to cloud of the CPS architecture was required. The self-provisioning and dynamic allocation of resources, that is offered by most cloud providers, deal with the required scalability of CPSoS to support complex and high-demanding MAPE-K control loops.

However, we realized that agility in development and deployment, different needs of scalability of certain components, their modifiability, or even reliability could be improved by using microservices. The software architecture that this framework proposes for CPSoS SaaS applications, introduces microservices by decomposing the MAPE-K control loop into independent components called microservices (see service definition in Section 4.1). Figure 4 shows the decomposition of MAPE-K control loop into microservices for data monitoring, real-time or historical analysis, planning, and execution. These microservices work in concert to deliver the application's overall functionality through well-defined RESTful API contracts. This makes it easier to scale

[9] Abnormalities of voltage and frequency between the corresponding phases of a solar panel output and grid supply

the specific services demanding more computing resources, such as data analysis, and to version and update independently each service of each other. This loose coupling makes it possible that an application evolves in a reliable manner, and additionally, makes it easier to keep well-defined the team boundaries, (development can be separated according to business capabilities, aligned with stakeholder's definition in Section 4.1), and to define a multidisciplinary team as shown in Figure 4. In this way, the software architecture of the framework is based on autonomic computing, cloud and microservices.

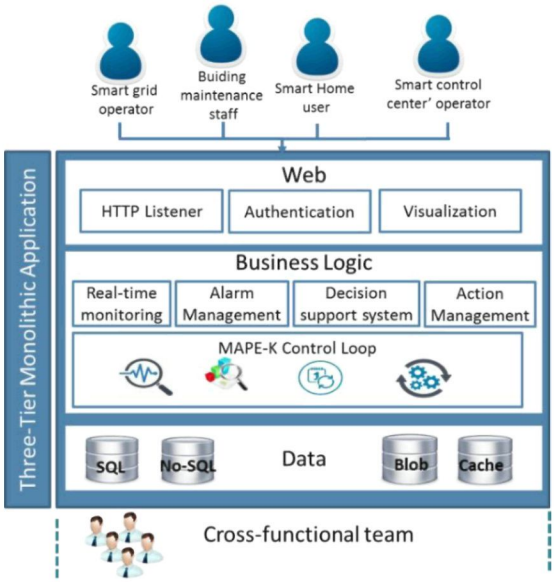


**Figure 3. Three-Tier Monolithic Architecture for CPS Apps**

Additionally, microservices enable the separation of the application from the underlying infrastructure on which it runs, as they declare their resource requirements to a "cluster manager". This cluster manager schedules microservices onto machines assigned to the cluster in order to maximize the cluster's overall resource utilization [27]. Microservices of the CPSoS Framework are automatic packaged and deployed in a container[10]. Figure 5 illustrates this definition in which each microservice container fits with a virtual machine (VM1-VM7). Hence, different VMs allocate web, business logic, and data, but also, business logic is divided according to the different stakeholders who are involved. In this way, it is easier to evolve a specific service of the MAPE-K or isolate a fault of a particular microservice. Additionally, each microservice manage its own database or instance to ensure visibility only in those data that are required. Finally, VM5 and VM6 are an example of the flexibility and independence that microservices provide with regard to monolithic architectures, since the variability of each service can be implemented through different microservices and scale out independently (see Fig. 5).

## 5. CONCLUSIONS AND FURTHER WORK

This paper presents the architectural foundations that serve as a guidance for the development and deployment of CPSoS applications dealing with scalability, flexibility, adaptation, agility and self-management needs. The CPSoS Framework has been conceptualized with the purpose of being one the seeds of a future common open service platform to develop third-party Cloud CPSoS-based applications. The framework has been founded from the solid bases of previous experience and work on CPS carried out in past European research projects, and whose implementation is on-working in current H2020 and Spanish research projects.

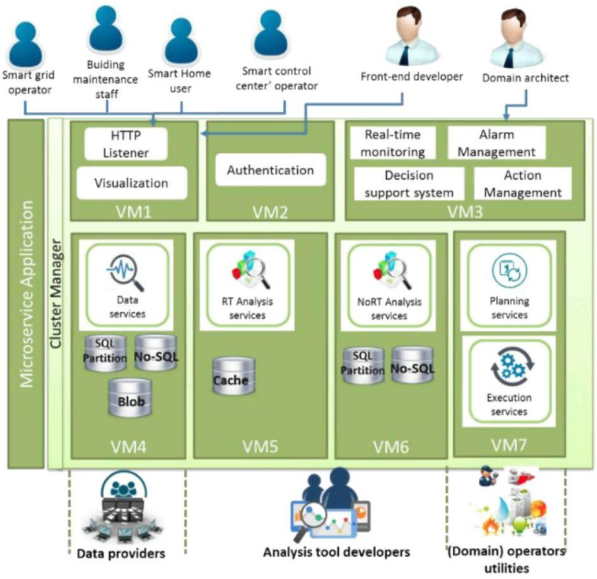


**Figure 4. Microservice Architecture for CPS Apps**

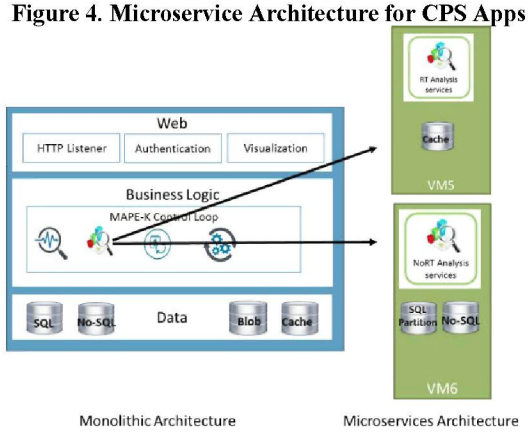


**Figure 5. Monolithic vs. Microservices Architecture**

The framework is defined in terms of stakeholders, SaaS, an MDD process and a software architecture. The design of the framework makes a step forward in the field by describing how to integrate MDD, autonomic computing, cloud and microservices to guarantee the required qualities of CPSoS SaaS applications. The MAPE-K control loop is implemented through independent microservices that can be self-provisioned for scaling out over a cloud platform. In this way, it is possible to introduce the adaptability and self-management of autonomic computing at fine-level granularity, gaining in decoupling and independence of changes. This independence is also guaranteed since changes are introduced via a well-defined MDD process for CPSoS by generating code at the microservice level from domain CPS models. These foundations are presented in this work as a starting point for future research and validation to evaluate (i) each one of the promising qualities of the architectural styles and approaches we integrated into the CPSoS Framework, and (ii) possible challenges such real-time response and security—typical of any distributed system—which could be

[10] *Containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries. This guarantees that the software will always run the same, regardless of its environment.* https://www.docker.com/what-docker

addressed through hybrid cloud and fog computing approaches. In this regard, we are planning to conduct several case studies with different pilots at the university campus.

## 6. ACKNOWLEDGMENTS

This work is supported by the Spanish fund MESC (DPI2013-47450-C2-2-R) and CPSELabs H2020 644400.

## 7. REFERENCES

[1] Beydeda, S. M. and V. Gruhn. 2005. Book *Model-Driven Software Development*, Springer

[2] Botta, A., De Donato, W., Persico, V., &Pescapé, A. 2016. Integration of Cloud computing and Internet of Things: A survey. *Future Generation Computer Systems*, 56, 684–700. DOI=http://dx.doi.org/10.1016/j.future.2015.09.021

[3] Cavalcante, E. et al. 2016. On the interplay of Internet of Things and Cloud Computing: A systematic mapping study. *Computer Communications*, Volumes 89–90, 2016, pp. 17-33, DOI=http://dx.doi.org/10.1016/j.comcom.2016.03.012

[4] Chun, I. 2010. Autonomic computing technologies for cyber-physical systems. In *the 12th Int.Conf. on Advanced Communication Technology,* Vol. 2, pp. 1009–1014, IEEE.

[5] Coulouris, G. et al.. 2012. Book: *Distributed Systems: Concepts and Design*, 5ª ed., pp. 19. Addison-Wesley

[6] CyPhers – Report Research Agenda and Recommendations for Action http://cyphers.eu/sites/default/files/d6.1+2-report.pdf

[7] de Farias, C. et al. 2016. COMFIT: A development environment for the Internet of Things. *Future Generation Computer Systems*. Available online 5 July 2016, DOI=http://dx.doi.org/10.1016/j.future.2016.06.031

[8] European Commission. 2013. Cyber-Physical Systems: Uplifting Europe's Innovation Capacity. Available: ec.europa.eu/newsroom/dae/document.cfm?doc_id=3948

[9] Familiar B. 2015. IoT and Microservices. In Book *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions*. Apress.

[10] Ferry, N., et al. 2014. CloudMF: Applying MDE to Tame the Complexity of Managing Multi-cloud Applications. In *IEEE/ACM 7th Int. Conference on Utility and Cloud Computing* pp. 269–277 http://doi.org/10.1109/UCC.2014.36

[11] Galache, J.A. et al. 2014. ClouT: Leveraging Cloud Computing Techniques for Improving Management of Massive IoT Data. In *IEEE 7th Int. Conference on Service-Oriented Computing and Applications*, pp. 324-327 http://doi.org/10.1109/SOCA.2014.47

[12] García, J.A. and Rodríguez, J.J. 2014. Open CPS platforms. In *Proceedings of the CPS 20 Years from Now—2nd Experts Workshop Cyphers*, pp. 22–26, 2014.

[13] Gubbi, J., Buyya, R., Marusic, S., &Palaniswami, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660 http://doi.org/10.1016/j.future.2013.01.010

[14] Horn, P. 2001.*Autonomic Computing: IBM's perspective on the state of information technology*. IBM Corp, Oct 2001. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[15] Hutchinson, J., et al. 2011. Empirical assessment of MDE in industry. In *Proc. of the 33rd Int. Conf. on Software engineering - ICSE '11* pp. 471-480. ACM Press. http://doi.org/10.1145/1985793.1985858

[16] Jamshidi, M. 2008. System of systems engineering - New challenges for the 21st century. In *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, no. 5, pp. 4-19. DOI=http://dx.doi.org/10.1109/MAES.2008.4523909

[17] J. O. Kephart and D. M. Chess, The vision of autonomic computing, *Computer*, vol. 36, no. 1, pp. 4–50, 2003.

[18] Kosar, T., Bohra, S., Mernik, M. 2016. Domain-Specific Languages: A Systematic Mapping Study, *Information and Software Technology*, Vol. 71, March 2016, pp. 77-91, DOI=http://dx.doi.org/10.1016/j.infsof.2015.11.001

[19] Lee, E.A. 2008. Cyber Physical Systems: Design Challenges. *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, pp. 363-36 DOI=http://doi.org/10.1109/ISORC.2008.25

[20] Lewis, J., Fowler, M. 2014. Microservices [Online]. Available: http://martinfowler.com/articles/microservices.html

[21] Maier, M. W. 1998. Architecting Principles for Systems-of-Systems. *Systems Engineering*, vol. 1, no. 4, pp. 267-284

[22] Miller, J., and Mukerji, J. (Eds.). 2003. MDA Guide Version 1.0. 1. Object. Management Group, Needham.

[23] Miorandi, et al. 2012. Internet of things: Vision, applications and research challenges, *Ad Hoc Networks*, 10(7), pp. 1497-1516. DOI=http://dx.doi.org/10.1016/j.adhoc.2012.02.016

[24] NESSI White Paper. 2015. Cyber Physical Systems Opportunities and Challenges for Software, Services, Cloud and Data. Available http://www.nessi-europe.eu/Files/Private/NESSI_CPS_White_Paper_issue_1.pdf

[25] Papotti, P. E., et al. 2013. A Quantitative Analysis of Model-Driven Code Generation through Software Experimentation. In *Proc. of CAiSE*. Springer Berlin Heidelberg pp. 321–337 DOI=http://doi.org/10.1007/978-3-642-38709-8_21

[26] Rajkumar, R., et al. 2010.Cyber-physical systems: The next computing revolution. In *47th ACM/IEEE Design Automation Conference*, pp. 731-736. DOI=http://doi.org/10.1145/1837274.1837461

[27] Russinovich, M. Microservices: An application revolution powered by the cloud. [Online] Available: https://azure.microsoft.com/es-es/blog/microservices-an-application-revolution-powered-by-the-cloud/

[28] Schmidt, D. C., et al. 2004. Leveraging Application Framework. *ACM Queue*, 2 (5 (July/August), pp. 66–75 DOI=http://dx.doi.org/10.1145/1016998.1017005

[29] Sztipanovits, J., et al. 2013. Foundations for Innovation: Strategic R&D Opportunities for the 21th Century Cyber-Physical Systems, NIST. Available: http://www.nist.gov/el/upload/12-Cyber-Physical-Systems020113_final.pdf

[30] Tolvanen, J.-P., & Kelly Steven. 2016. Model-Driven Development Challenges and Solutions Experiences with Domain-Specific Modelling in Industry. In *Proc. of the 4th Int. Conf. MODELSWARD* pp. 711–719.

[31] Vidal, C., Fernandez-Sanchez, C., Díaz, J. and Pérez, J. 2015.A Model-Driven Engineering Process for Autonomic Sensor-Actuator Networks. *Int. Journal of Distributed Sensor Networks*. Vol 2015, Hindawi Publishing Corporation

[32] Whittle, J., Hutchinson, J., &Rouncefield, M. 2013. The State of Practice in Model-Driven Engineering. *IEEE Software*, *31*(3), 79–85. http://doi.org/10.1109/MS.2013.65