# Systems Integration within Cyber-Physical Systems

José Azinheira
joseazinheira@live.com.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2018

**Abstract**

With the popularization of cyber-physical systems (CPS), the necessity to develop solutions regarding the architectures of these systems arises. This dissertation addresses the complexity of CPS, relating the difficulties of designing and implementing them in real-life applications. These systems are more complex when compared to traditional systems, due to the combination of both computation and physical processing, thus requiring the comprehension of several distinct disciplines, such as cybernetics, mechatronics and process science. Since the concept of these systems is relatively new, its definition is still unclear, as are the main foundations for the respective architecture design. By analyzing several key concepts and studying already existing architectures, we propose a possible solution to address this problem. First we will propose a definition for CPS, and then present a general-purpose architecture using Feature-Oriented Software Development (FOSD) techniques. Finally, we will present a scenario for architectural application.

**Keywords:** Cyber-Physical Systems, Service-Oriented Architecture, Embedded Systems, Feature-Oriented Software Development, Internet of Things

## 1. Introduction

The term *cyber-physical system* (CPS) was first used around 2008 by Helen Gill, to define the combined effort of computation along with physical processes [15]. Essentially, a CPS is a system comprised by both software computation and physical processing over a network, which work together as a mechanism whose performance excels when compared to regular systems.

These systems can be used for diverse purposes which span across several areas, such as automotive [5], avionics [3] , medicine [3] and industry [3], which also demonstrate CPS interdisciplinarity [4, 26]. There is a wide belief that CPS will play an important part in technological advance, as its use expands into more areas [3, 27, 17].

The main motivation for this research is the fact that CPS are a relatively new topic, meaning there are oportunities for research and new ideas. We identified three main research obstacles: lack of specific information regarding CPS, for example, in security; lack of consensus regarding the definition of a CPS; and several architectures with distinct characteristics make it difficult to idealize a general-purpose CPS architecture.

### 1.1. Objectives

To better understand our objectives, we first researched general CPS information and definitions, as well as architectures that allowed us to identify key similarities and differences. We then identified interesting areas of research that connect to CPS. Finally, we determined the key features in a CPS, with focus on features that may not have given proper research.

Having achieved these research milestones, we defined our objectives:

- We propose a definition for CPS based on the studies made.
- We develop a CPS architecture using Feature-Oriented Software Development (FOSD) techniques.
- We apply the developed architecture to two real-life scenarios.

By acomplishing these objectives, we believe that it will contribute to the development of CPS.

## 2. Architectural Survey

The following architectural survey is relevant to the first phase of the FOSD methodology: the Domain Analysis.

### 2.1. **Shop Floor**

Liu and Jiang present an architecture whose objective is to reduce downtime, provide efficient decision-making and reduce operational costs, achieving the goal of inteligent manufacturing [17].

Since CPS development is in its early stages, the authors stress that there is a need for the development of a universal architecture, not only for specific applications. Some challenges, like implementing the connection between the cyber and physical counterparts, and also the lack of readiness of most manufacturing systems to handle big-data, contribute to this delay in implementation [17].

To address these problems, the authors suggest a three-layered architecture: physical connection, middleware and computation layers [17].

- **Physical connection layer**: groups the architectures sensorial capabilities, starting with embedding relevant components with sensors, RFID devices and measurement devices, which capture data in real-time.
- **Middleware layer**: responsible for transmitting the captured data to the central server for processing, serving as a connection between the physical and computation layers.
- **Computation layer**: responsible for giving context to the processed information, using different data analysis methods.

The architecture also considers networking as a communication channel to transmit data among the systems' several resources [17].

In order to better implement this architecture, the authors suggest three fundamental technologies: interconnection and interoperability among different devices, since CPS are often constituted of heterogeneous components; abstraction of a CPS elements for better understanding; and definition of data interface [17].

### 2.2. **Industry 4.0**

Lee, Bagheri and Kao defend that since CPS are a new technology, it is important to define a clear structure and methodology for their implementation in industrial applications, along with guidelines for a unified framework [16].

According to the authors, CPS architectures traditionally consist of two main components: advanced connectivity, responsible for real-time physical data acquisition and information feedback from the cyber-space; and intelligent data management, analytics and computational capability, responsible for constructing the cyber-space. This results in an abstract concept that does not allow for general architecture implementation [16]. The 5C architecture (Connection, Conversion, Cyber, Cognition, Configuration) attemps to provide an answer to the lack of specificity in some architectures [16]:

- **Smart Connection Level**: responsible for the sensorial component of the CPS, capturing data from the environment.
- **Data-to-Information Conversion Level**: responsible for the inference component of the CPS, converting raw data into meaningful information.
- **Cyber Level**: responsible for being the data repository of the whole architecture, having information being delivered to it from every other machined connected to it in its respective network.
- **Cognition Level**: responsible for decision-making in the system, acting based on knowledge extracted from system monitoring.
- **Configuration Level**: responsible for configuration of machines, supervising control to allow machines to self-configure and self-adapt.

### 2.3. **Prototype architecture**

Tan, Goddard and Pérez propose an architecture whose objective is to provide a framework that unites human and machine computational models [24]. The authors then proceed to explain that in a traditional embedded system architecture, sensors and actuators are tightly-coupled with the control unit, which affects modularity [24]. This poses a problem, as the system is less adaptive and more centralized.

With this, the authors proceed to present their architecture [24]:

- **Global reference time**: a global reference time is provided by the network and accepted by the components.
- **Event/information driven system**: *events* are separated from *information*, where events are data captured from the sensors (sensor events) or actions performed by the actuators (actuator events), and information is the output processed by the control units.
- **Quantified confidence**: a unified event/information model should be used, with the following properties: global reference time, life-span; confidence; digital signature; trustworthiness; dependability; criticalness.
- **Publish/subscriber model**: each CPS control unit only receives the events/information it has subscribed to, based on the role it plays in the system, and publishes events/information when necessary.

- **Semantic control laws**: represent the core of the CPS control units, where they limit the system behaviors according to user defined conditions and scenarios, which allows control over the output.

- **New networking techniques**: the network, in addition to providing the global reference time, also provides event routing and data management.

### 2.4. PowerCyber architecture

Hahn, Ashok, Sridhar and Govindarasu present a security testbed for CPS, rather than an system architecture. This architecture is a response to the limited availability of real-life CPS that allow for security testing, a difficulty that we struggled with during research. This is important to discover possible vulnerabilities as well as validate a system [9].

This architecture divides itself into three layers: cyber, physical and communication [9].

- **Cyber Layer**: computational part of the system and the human-in-the-loop and closed-loop mechanisms used to measure the grid's reliability and performance. Uses control centers for SCADA (Supervisory Control and Data Acquisition) based operations and substations for auxiliary purposes.

- **Physical Layer**: uses simulation platforms such as RTDS (Real Time Digital Simulator) and DIgSILENT PowerFactory for real-time and non-real time power usage, respectivelly.

- **Communication Layer**: communication is made using LAN (Local-Area Network) and WAN (Wide-Area Network) environments. ISEAGE (Internet-Scale Event and Attack Generation Environment) is also used to simulate cyber-attacks.

The authors also suggest the development of cyber-physical metrics in order to improve system security and resilience, where in the physical counterpart we can use metrics such as power flow and stability for measure [9]. This is important because metrics for CPS have not been defined yet, as they are very hard to define due to the abstraction of systems.

### 2.5. Service-Based Architecture

Motivated by the challenge of designing and the multidisciplinarity of CPS, La and Kim present a SOA (Service-Oriented Architecture) solution. Using services, it is possible to use loose-coupling to deal with dynamic composition, improving a systems' modularity [14]. The authors then define three key assumptions on CPS: physical devices being connected to the control system over a network; functionality not being tightly-coupled to hardware elements; and real-time, on-demand processing [14].

The authors then proceed to present their architecture composed of three tiers (layers): Environmental Tier, Control Tier and Service Tier [14].

- **Environmental Tier**: sensors and actuators, where information is gathered and then transfered to applications in the Control Tier, and presented to the user by the actuators.

- **Control Tier**: responsible for decision-making and data-analysis, as well as monitoring physical devices and services, determining what is the appropriate service for an action.

- **Service Tier**: service storage, possesses control systems that decide which services will be deployed, and then forwards the final output to the user.

### 2.6. Modular architecture

Ahmed, Kim and Kim propose a service-based architecture that allows for rapid and low cost deployment in systems, as well as interoperability between systems due to the reusability of services [3].

The authors divide their architecture into five modules (layers): Sensing, Data Management, Next Generation Internet, Service Aware and Application modules [3].

- **Sensing Module**: responsible for data collection using sensors and providing that data to the DMM. Supported by networks which connects the sensors that enable real-time control.

- **Data Management Module (DMM)**: responsible for data processing such as normalization, noise reduction and data storage. This data is then forwarded to the SAM using NGI.

- **Next Generation Internet (NGI)**: responsible for wireless communication in the system. Its main feature is that it enables for applications to select the packet transmission path, instead of being forced to a single path.

- **Service Aware Modules (SAM)**: responsible for decision-making in the system and task analysis and scheduling, sending data to the respective services.

- **Application Module**: responsible for deploying services while saving data in a secure database. The system makes use of both cloud storage and local storage for added security.

The authors then divide the security of CPS into three phases: awareness, which considers the security and accuracy of collected data; transport, which considers data transmission safety; and physical, which considers the security of physical devices, such as servers and workstations [3].

### 2.7. Context-Aware Vehicular Cyber-Physical Systems with Cloud Support

Wan, Zhang, Zhao, Yang and Lloret suggest an architecture for vehicular networks in order to answer to the increasing demand of adapting these networks into cloud-assisted, context-aware vehicular CPS. The goals of this architecture are to improve road safety and traffic efficiency while having ambiental preocupations [27].

These context-aware services consist of services like live traffic updates or direct video-feeds for a certain route chosen by a driver.

The authors divide their architecture into three computational layers: Vehicle, Location and Cloud [27].

- **Vehicle Computational Layer**: considers equipment and devices in a vehicle that can be used as sensors to infer information.
- **Location Computational Layer**: considers devices deployed in the environment, such as roads, that exchange information with the sensors in the vehicles. Vehicles that are out of range of these devices can connect to the network through nearby vehicles.
- **Cloud Computational Layer**: considers applications and services owned by diverse entities that exchange information with each other in a single, large cloud.

The authors also consider security to be an important aspect, as the user's information should be protected by services such as access control, encryption and authentication [27].

## 3. Architecture comparison

We will now present compare the architectures based on their characteristics, as depicted in Table 1. These characteristics include functionalities or operation methodology that the authors mentioned but may have not specified how they function.

### 3.1. Data collection

All architectures, except the PowerCyber testbed architecture, consider the data collection component, as it is the most important of a cyber-physical system: to collect raw data through its sensors and present it to the final-user using actuators.

The Shop Floor architecture considers sensors of vibration, pressure and temperature for data collection, but does not consider how this data is presented by the actuators.

The 5C architecture considers the typical sensor usage for data collection, but also ERP (Enterprise Resource Management), as well as other tools.

The Prototype architecture does not make any specific reference on how data is collected and presented. Interestingly, Tan et al. consider the sensors to be part of the cyber layer of the architecture, instead of the physical layer, like most of the other authors and ourselves do.

The Service Based architecture does not specify what the sensors and actuators are and how they perform, but how the applications that run them should route the information to the respective services.

The Modular architecture does not specify the type of sensors that may be used to collect data. On the other hand, Ahmed et al. suggest that the actuators may be cars, lamps or watering pumps.

The Context Aware Vehicular architecture considers sensors that are present in a vehicle. The actuators can be live feed from vehicles or real-time information about traffic.

We can draw the following conclusions: devices that act as sensors or actuators can be almost anything. Other data management tools, such as ERP, can also be used for data collection. When considering sensors in a system, it should be given attention to several aspects: positioning, in order to improve communications with other nodes in the system; resource usage, in order to reduce operational costs; real-time operation, in order for information to be the current. Actuators are not as specified as their sensor counterparts.

### 3.2. Network

The Shop Floor, PowerCyber and Service Based architectures do not specify how networking is implemented.

The 5C architecture considers the MTConnect standard, which is a data and information exchange protocol used in manufacturing operations.

The Prototype architecture considers the implementation of a set of backup servers, dubbed Secured Network Knowledge Database Servers, which only accepts expired data. It also considers using a global-reference time for event-ordering.

The Modular architecture considers both wired and wireless networking. It also defends that network protocols should be able to adapt in order to individual application necessities in order to assure quality-of-service.

The Context Aware Vehicular architecture considers wireless networking for information exchange between its systems and the environment it is connected to.

We can draw the following conclusions: most architectures favor wireless networking instead of Ethernet, mainly due to convenience, such as portability; protocols should be taken in consideration when designing a systems' networking infrastructure.

Table 1: Surveyed architecture comparison.

| | Data collection/presentation | Networking | Decision-making | Security |
|---|---|---|---|---|
| Shop Floor | Yes | Yes | Yes | No |
| 5C | Yes | Yes | Yes | No |
| Prototype | Yes | Yes | Yes | No |
| PowerCyber | No | Yes | No | Yes |
| Service Based | Yes | Yes | Yes | No |
| Modular | Yes | Yes | Yes | Yes |
| Context Aware | Yes | Yes | Yes | No |

### 3.3. **Decision-making**

The PowerCyber architecture does not consider decision-making and the 5C architecture does not specify how it is implemented.

The Shop Floor architecture considers big data for analysis, but identifies the excess, various types and low quality of data to be processed to be a challenge. To tackle this, data processinmg is divided into two categories: stream computing and batch computing. Stream computing processes real-time data while batch computing processes batches of already existing data.

The Prototype architecture considers semantic control laws as control systems, defined by the user.

The Service Based architecture dedicates a whole tier to decision-making: the control tier. In this tier there is a local registry with the systems' services, where a service monitor updates this registry whenever there a change is made.

The Modular architecture dedicates a whole module to decision-making: the Service Aware Module. This module redirects received data to the respective services.

The Context Aware Vehicular architecture does not specify how the decision-making is processed, only the factors that may affect it.

We can draw the following conclusions: there are several alternatives when considering decision-making, such as dedicating a whole layer of the architecture for that purpose and what tools to use in them.

### 3.4. **Security**

The Shop Floor and 5C architectures do not consider security.

The Prototype architecture considers security as an important aspect of CPS, as well as acknowledging that security is a challenge, but does not specify how it is implemented.

The PowerCyber architecture provides a testbed to test CPS against cyber attacks. While this does not fully relate to our architecture, it was important to have a perspective on how attacks may be conducted on these systems.

The Service Based architecture considers security as an important aspect of CPS but does not specify how it is implemented.

The Modular architecture considers that the security of each CPS should be done according to its nature. Ahmed et al. also divide security into three phases: awareness, which is essentially an assurance of integrity; transport, which is also an assurance of integrity; and physical, which considers safety procedures in physical devices. This last phase is very important, as it is equally important to protect the physical devices, such as sensors, actuators and desktops, when comparing to information in the system.

The Context Aware Vehicular architecture considers information privacy through access control, encryption and authentication, but does not specify how to implement it.

We can draw the following conclusions: while security is acknowledged by some authors, it is not one of the main concerns in CPS architecture design. Because of this, we will try and provide some contribution to security measures when proposing our architecture.

## 4. Architecture Proposal

We will now present the results of our research.

### 4.1. **Definition consensus**

An early and common problem when researching were the multiple definitions for the concept of CPS, which has lead to a lack of consensus. Despite all definitions converging in the same direction, they also have differences that end up creating confusion, which can lead to difficulties when trying to understand what should be part of a CPS. As such, we will propose a definition of our own based on all of our studies.

We define CPS as **distributed systems**, comprised of both **computational** and **physical** counterparts which operate in synchronism, **loosely coupled** between themselves, and whose operation is regulated by **decision-making** mechanisms and connected to their surrounding environment by a **network**.

This definition attempts to encompass all characteristics that we deem necessary in a CPS: computational and physical aspects, decision-making

and networking. We also consider loose-coupling of components and functionalities, as it allows for easier replacing, modularity and cost reduction. We also classify CPS as distributed systems, because they consist of a networking of some autonomous machines that share resources [25]. Despite security being very important, it does not fit into the definition because we cannot assume all CPSs to have security mechanisms, as evidenced in our architecture survey.

### 4.2. **Requirement definition**

First, we present the requirements for a CPS, pertaining to the second phase of the FOSD: domain analysis (depicted in Fig. 1). These requirements are divided into two categories: functional, mandatory for system functioning, and non-functional, that while useful are not necessary for functionality.

In the following figure, on the left are the functional requirements, while on the right are non-functional requirements.

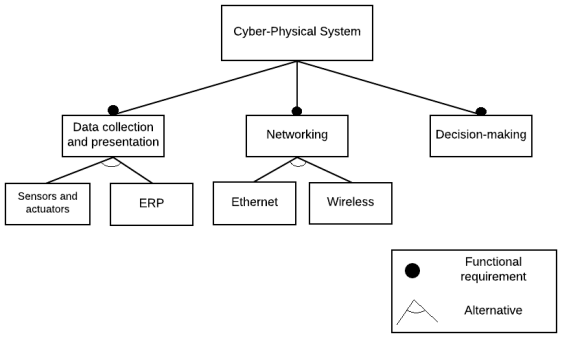


Figure 1: Feature diagram for the second phase of FOSD.

**Functional requirements**:

- **Data collection/presentation**: how the system gathers and presents data. Considers physical devices responsible for data collection and presentation.
- **Communication**: how the system's components communicate with each other and with the external world. Can be wired or wireless networking and considers communication protocols.
- **Decision-making**: how the system processes the information infered by the sensors. Considers tools for data optimization, normalization and noise-reduction.

**Non-functional requirements**:

- **Security**: assures the security of the system and the information in it against attacks and environmental hazards.
- **Redundancy**: assures that the system's components have a safeguard component in case of any unexpected problem that affects functioning. This should be emphasised to the critical components.
- **Interfaces**: most systems possess an user-interface, and as such their design should be given care as to develop an easy to use interface.
- **Scalability**: the system should be ready to receive resource upgrades.
- **Availability**: the system should be available at all times, specially in critical appliances such as medicine or security. This can be improved by also improving security and redundancy.

### 4.3. **Architecture proposal**

We will now present a model for a CPS architecture, which corresponds to the second phase of the FOSD methodology: the domain design implementation. We followed the studies of Jansen and Bosch [13], Ahmed et al. [3], Derler, Lee and Vincentelli [6] and Gunes, Peter, Givargis and Vahid [8] as guidelines for our design process.

There is not a lot of emphasis given to security in many architectures, so what we propose is an architecture that combines all of the functional requirements, as well as non-functional requirements, while still considering security.

Our architecture is service-based, using the SOA software design methodology. Due to the heterogeneous nature of the several components of a CPS, using services allows for a seamless integration of their functionalities, as well as assuring loose-coupling of these same services, providing easier replacing should it be needed, as well as reusability for several systems, allowing rapid deployment and lower costs for interoperable and scalable sytems [3].

A service is a software functionality that can be deployed using Web Services. A service has two main properties: it is self-contained, so its functionality is separated from other services and encapsulated [20], meaning changes in one service will not impact others (providing loose-coupling); and produces an output based on an input, where users do not need to understand the service's functioning in order to use it. A service also has three main components: an interface, where the user operates the service (Web Service); a contract, that defines

how the service provider and service requester interact [21]; and its implementation, where the service's code is defined.

A service is then used to expose the functionality of a component or application via WebService [19], which allows for different components, which would otherwise be incompatible, to exchange information via a open standard protocol: SOAP (Simple Object Access Protocol).

We can also consider the use of microservices instead of regular services. While SOA tipically divides its services into four categories (Business, Enterprise, Application and Infrastructure), microservice architectures only rely on Functional and Infrastructure services. This makes its implementation easier and cheaper when compared to SOA, but there are also several disadvantages: microservice-based architectures limits the number of protocols used, thus limiting interoperability when using heterogeneous components and protocols; decoupling is also non-existant in microservices, which limits interoperability [21]. Despite these limitations, microservices are appropriate when considering smaller scale systems that perform a limited set of functions.

Inspired by the studies of Tan et al. [24], Hu, Xie, Kuang and Zhao [10] and La et al. [14], we will divide our architecture into several layers. In the case of [14], the authors divide their architecture into three layers: Environmental Tier, Control Tier and Service Tier. In our opinion, separating the service capabilities from the Control Tier and dedicating a single layer to the services is unnecessary, adding more complexity to the system. Our solution is to create a single layer that encompasses all cyber capabilities of the system, including decision-making and services, and another layer for the physical capabilities of the system [1].

With this, our architecture, depicted in Section 4.3, will be divided into three layers: Physical Layer, Computational Layer and Security Layer.

#### 4.3.1 Environment

The Environment is where information is present and collected via sensors, and also where it is presented via actuators. Information can be collected and presented directly or undirectly: via human requested operation or automated operation, respectively.

The Environment is also where the Internet is and connects to the system.

#### 4.3.2 Physical Layer

The Physical Layer contains the physical devices that either capture or present data.

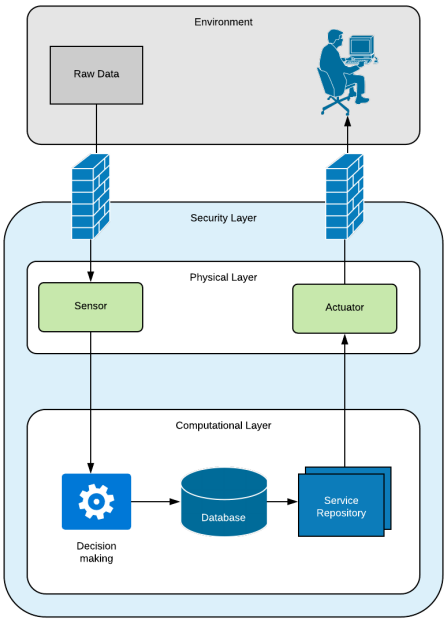


Figure 2: Model for the service-based architecture.

**Sensors**: captures unprocessed data from the Environment.

**Actuators**: displays processed data to the user, affecting the Environment.

The selection of these devices should have several aspects in consideration: resource usage should not hinder the systems' performance [23]; positioning should be done in order to reduce information propagation times.

#### 4.3.3 Computational Layer

The Computational Layer contains components that are involved in computational tasks that manage information and services.

**Data storage**: dedicated to storage data for further use. This storage can be done with local storage or using cloud storage via a service-provider.

**Decision-making**: decision-making tools to process the data infered by the sensors as well as the systems' corrective measures for resilience control. Since CPS are part of the IoT (Internet of Things), big-data is usually involved in these systems, so its adequate to adapt big-data analysis tools.

**Services**: contains the definitions of the system's services. Definition is done using Web Services and communication through the respective protocols, such as SOAP and REST (Representational State Transfer).

**Networking**: how the system's components are connected with each other and how the system contacts the external world. This networking can be done using Ethernet or with a WAN.

#### 4.3.4 Security Layer

The Security Layer contains mechanisms that ensure the systems' safety when exposed to internal and external attacks. Due to the heterogeinity nature of CPS, it is necessary to consider a considerable larger number of threats when compared to traditional systems: both cyber and physical parts are prone to attacks or environmental hazards [11].

Before CPS, some systems would be isolated from the Internet, unable to connect with it, so outside threats would not be considered. This lead system designers and security experts to rely in *security by obscurity*, which "hopes" that the possible attackers do not possess knowledge of the system in order to attack it. Nowadays, with connectivity being mandatory in CPS, this is simply not acceptable as a security standard [12].

We can use a VPN (Virtual Private Network) for both wired and wireless networking, where in the first we use a EVPN (Ethernet Virtual Private Network) and in the latter a wireless VPN. VPNs allow us to set up a network whose access is much more restricted, therefore much harder to be susceptible to attacks as it enforces encryption [22]. For our architecture, we only consider VPN services provided by companies which ensure the best service.

We will now go over how VPNs protect integrity, confidentiality and availability:

- **Integrity**: consists in the information arriving from the origin to its destination exactly the same (unmodified) [18].
- **Confidentiality**: consists in the information being transmitted being only seen and received by its supposed desination and never by anyone else [18].
- **Availability**: consists in the information being available at all times for authorized users only [18].

Like any other system, using a VPN is not a failproof solution. VPNs also have the disadvantage of reduced connection speed in some occasions (latency) in situations where, for example, the VPN service provider is located relatively far from the service requester [22]. Even so, we consider using a VPN in a CPS to be very advantageous, as the advantages far outweigh the disadvantages.

#### 4.3.5 Publish-subscribe messaging

In order to properly align the systems' messaging service with the loose-coupling nature of SOA, the publish-subscribe communication paradigm is an adequate option. The publish-subscribe paradigm, often dubbed *pub-sub*, consists of nodes (subscribers) that register their interest in events through a subscription, and whenever a publisher receives any event, they are notified of it [7]. With this, we can attribute a publisher node to each sensor and a service responsible for the infered data, which subscribes to the sensor. Whenever a sensor has to be removed from the system, the node is removed for the messaging list without affecting other nodes.

The analogy for application in our system would be: a sensor (publisher) sends a reading (topic), while a control mechanism (subscriber) responsible for monitoring a certain reading will pull its respective reading.

### 4.4. Theoretical Example - System of Systems

This example consists of an application of our architecture to a system of systems, in a parts warehouse, which also includes a stock-control subsystem. The architecture is depicted in Fig. 3.

The problem is that the owner wants to maximize automation in order to improve efficiency. The solution we propose is implementing a service-based CPS architecture that allows for better stock control and invoicing.

Regarding the **Physical Layer**:

- **Sensors**: a thermometer for climate control; a lighting sensor for lighting control; a motion-detector in the warehouse's entrance for an alarm.
- **Actuators**: air-conditioning system; LED lights; alarm system; desktops or laptops with interfaces.

Regarding the **Computational Layer**:

- **Data processing**: an ERP such as PRIMAVERA [2] for the overall stock-control, billing and invoicing.
- **Data storage**: an appropriate storage option considering the warehouse size: for a smaller warehouse, local-storage with external and internal hard-drives: for medium or larger businesses, cloud-storage provided by a professsional provider would be the best option for scalability concerns.
- **Services**: services for air-conditioning and lighting control, already considering their automated operation. The ERP could be coupled with Web Services for stock-control, billing and invoicing.
- **Networking**: a WAN set-up for communication. RFID tags in the shelves.

Regarding the **Security Layer**:

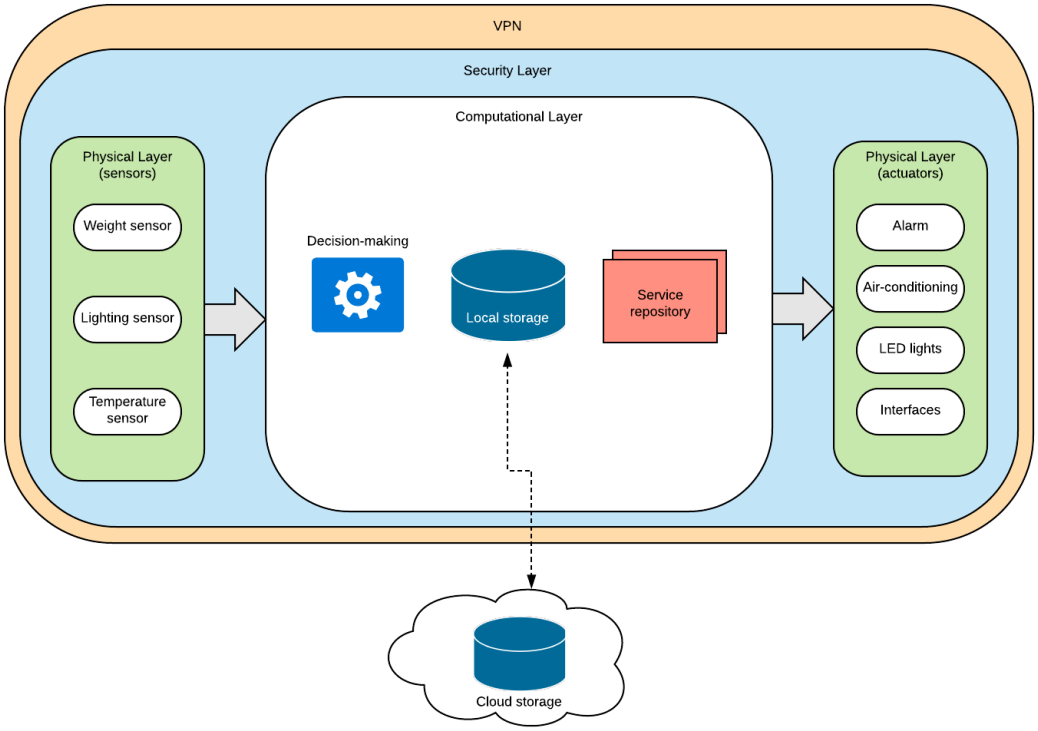


Figure 3: Parts warehouse CPS architecture.

The company would have a VPN set-up for communications, provided by a professional company. This would assure information encryption.

## 5. Conclusions and Future Work

The major goal of this dissertation was to develop a service-based architecture for CPS, based on an architectural survey and related concepts, which was accomplished with success. The other goal was to propose a definition for CPS, which was also accomplished with success. The initial goal to develop a practical implementation of this architecture was not achieved.

With this dissertation we consider that we gave a contribution to CPS by suggesting an alternative for architectures, while also giving visibility to an area that is unknown to the general public.

Its development had several positive aspects: knowledge regarding an area that was not previously known to us and that is becoming increasingly more relevant; developing the architecture required several studies in the area of architecture design that were challenging but useful for project developing; explaining to people the what a CPS is was also a positive experience.

There were also some negative aspects: the pratical implementation was not possible to conclude due to a lack of exiquibility. The difficulty in finding quality information took a toll on the time for the dissertation's completion, exceeding the research time than what was expected.

Regarding future work and despite the goals being accomplished, we feel like there is still room for improvements, such as: a practical implementation, since the objective of developing a pratical implementation was not achieved, this would be the starting point for future work; FOSD conclusion, as due to the lack of a pratical solution, only the first and second phases of the FOSD paradigm have been concluded in this dissertation, so its conclusion would be interesting; and Performance measuring, as due to the heterogeinety of the components present in CPS, metrics for evaluation have not yet been clarified, so developments in this area would be very benificial.

## References

[1] Microservices vs soa: What's the difference? `http://www.bmc.com/blogs/microservices-vs-soa-whats-difference/`. Retrieved in: 2018-04-19.

[2] Primavera bss, software de gestão, faturação, erp e pos. `https://pt.primaverabss.com/pt/`. Retrieved in: 04-04-2018.

[3] S. H. Ahmed, G. Kim, and D. Kim. Cyber Physical System: Architecture, applications

and research challenges. *IFIP Wireless Days*, pages 0–4, 2013.

[4] R. Alur. *Principles of Cyber-Physical Systems*. 2015.

[5] S. Chakraborty, M. A. Al Faruque, W. Chang, D. Goswami, M. Wolf, and Q. Zhu. Automotive Cyber–Physical Systems: A Tutorial Introduction. *IEEE Design & Test*, 33(4):92–108, 2016.

[6] P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1), 2012.

[7] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[8] V. Gunes, S. Peter, T. Givargis, and F. Vahid. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet and Information Systems*, 8(12):4242–4268, 2014.

[9] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *IEEE Transactions on Smart Grid*, 4(2):847–855, 2013.

[10] L. Hu, N. Xie, Z. Kuang, and K. Zhao. Review of cyber-physical system architecture. *Proceedings - 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, ISORCW 2012*, pages 25–30, 2012.

[11] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-Physical Systems Security – A Survey. 2017.

[12] D. Information, A. Centers, D. Technical, C. Security, I. Systems, and T. Information. *Journal of Cyber Security and Information Systems*, volume 5. 2017.

[13] A. J. A. Jansen and J. B. J. Bosch. Software Architecture as a Set of Architectural Design Decisions. *WICSA 2005 - 5th Working IEEE/IFIP Conference on Software Architecture*, 2005:109–120, 2005.

[14] H. J. La and S. D. Kim. A Service-Based Approach to Designing Cyber Physical Systems. *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 895–900, 2010.

[15] E. A. Lee and S. a. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. 2011.

[16] J. Lee, B. Bagheri, and H. A. Kao. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23, 2015.

[17] C. Liu and P. Jiang. A Cyber-physical System Architecture in Shop Floor for Intelligent Manufacturing. *Procedia CIRP*, 56:372–377, 2016.

[18] S. Maconachy and W. Ragsdale. A Model for Information Assurance: An Integrated Approach. *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, US Military Academy, West Point, NY*, pages 5–6, 2001.

[19] M. P. Papazoglou. Service -oriented computing: Concepts, characteristics and directions. *Proceedings - 4th International Conference on Web Information Systems Engineering, WISE 2003*, pages 3–12, 2003.

[20] R. Perrey and M. Lycett. Service-oriented architecture. *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.*, pages 116–119, 2003.

[21] M. Richards. *Microservices vs. Service-Oriented Architecture*. 2016.

[22] S. Sridhar, A. Hahn, and M. Govindarasu. Cyber-physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2012.

[23] A. F. Taha, N. Gatsis, T. Summers, and S. Nugroho. Actuator selection for cyber-physical systems. *Proceedings of the American Control Conference*, pages 5300–5305, 2017.

[24] Y. Tan, S. Goddard, and L. C. Pérez. A prototype architecture for cyber-physical systems. *ACM SIGBED Review*, 5(1):1–2, 2008.

[25] A. S. Tanenbaum and M. Van Steen. *Distributed Systems: Principles and Paradigms, 2/E*. 2007.

[26] W. F. V. D. Vegte and R. W. Vroom. Considering cognitive aspects in designing cyber-physical systems : an emerging need for transdisciplinarity physical systems from a transdisciplinary perspective. (i):41–52, 2013.

[27] J. Wan, D. Zhang, S. Zhao, L. Yang, and J. Lloret. Context-aware vehicular cyber-physical systems with cloud support: Architecture, challenges, and solutions. *IEEE Communications Magazine*, 52(8):106–113, 2014.