This is the peer reviewed version of the following article:

Plaza AM, Díaz J, Pérez J. Software architectures for health care cyber-physical systems: A systematic literature review. *J Softw Evol Proc*. 2018; 30:e1930. https://doi.org/10.1002/smr.1930

, which has been published in final form at

https://doi.org/https://doi.org/10.1002/smr.1930

This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions. This article may not be enhanced, enriched or otherwise transformed into a derivative work, without express permission from Wiley or by statutory rights under applicable legislation. Copyright notices must not be removed, obscured or modified. The article must be linked to Wiley's version of record on Wiley Online Library and any embedding, framing or otherwise making available the article or pages thereof by third parties from platforms, services and websites other than Wiley Online Library must be prohibited.

It is used to be stored in the Author's institutional repository, i.e. the Archivo Digital UPM (Universidad Politécnica de Madrid) to fulfill the article "Open Science" 37.2 of the 17/2022 Spanish law of Science, Technology and Innovation («BOE» núm. 214 06/09/2022.).

Software Architectures for Healthcare Cyber-Physical Systems: A Systematic Literature Review

Andrea M. Plaza[+,*], Jessica Díaz[+], Jennifer Pérez[+]

[+]Universidad Politécnica de Madrid-CITSEM, Madrid, Spain
[*]Universidad Politécnica Salesiana, Cuenca, Ecuador
aplaza@ups.edu.ec, yesica.diaz@upm.es, jenifer.perez@etsisi.upm.es

Abstract. Cyber-Physical Systems (CPS) refer to the next generation of ICT systems that mainly integrate sensing, computing, and communication to monitor, control, and interact with a physical process to provide citizens and businesses with smart applications and services: healthcare, smart homes, smart cities, and so on. In recent years, healthcare has become one of the most important services due to the continuous increases in its costs. This has motivated extensive research on *healthcare CPS*, and some of that research has focused on describing the software architecture behind these systems. However, there is no secondary study to consolidate the research. This paper aims to identify and compare existing research on software architectures for healthcare CPS in order to determine successful solutions that could guide other architects and practitioners in their healthcare projects. We conducted a systematic literature review (SLR) and compared the selected studies based on a characterization schema. The research synthesis results in a knowledge base of *software architectures for healthcare CPS*, describing their stakeholders, functional and non-functional features, quality attributes architectural views and styles, components, and implementation technologies. This SLR also identifies research gaps, such as the lack of open common platforms, as well as directions for future research.

Keywords. Healthcare; Cyber-Physical Systems; Software architecture; Internet of Things; Systematic Literature Review.

1 Introduction

Healthcare is becoming one the most important concerns for governments as the World Health Organization warns of increasing care costs [1], for example, in Europe, given the ageing population. Cardiovascular or chronic obstructive pulmonary diseases are the main causes of death in the European Union [1]. Nowadays, a wide range of informatics solutions are applied to healthcare—from clinical information systems (including electronic health records, electronic prescribing, and eHealth in general) and administrative and management systems to clinical research informatics and bioinformatics. Among these solutions, *health-monitoring systems* might play a key role in preventing the diseases mentioned above.

Monitoring systems have expanded their capabilities in what are known as *Cyber-Physical Systems* (CPS). CPS integrate sensing, computing, storage, and communication capabilities to control and interact with a physical process. Embedded computers monitor and control physical processes in real time or quasi real time, usually with feedback loops, where physical processes affect computations and vice versa [2]. The consolidation of CPS, partly promoted by the boom in the *Internet of Things* (IoT) [3], [1] *big data*, and the development of *wireless sensor-actuator networks* (WSAN), is generating new business models in a broad range of domains, including healthcare. Hence, *healthcare CPS* could not only prevent diseases but also proactively deal with diseases. On the one hand, healthcare CPS enable the monitoring of biosignals through miniature wearable sensors (aka *biosensors* and *bodysensors*), which could automatically detect a disorder or problem and generate an alarm that a patient or a doctor receives on his/her mobile phone or computer without human intervention. CPS are usually characterized by the implementation of feedback loops with cognitive and learning capabilities [4] such that these systems can act autonomously and rapidly without human intervention. On the other hand, healthcare CPS may facilitate immediate access for physicians and other health care professionals to patient information and medical records, as well as real-time access to data coming from biosensors, for efficient decision-making and treatment. With this twofold objective, in recent years, healthcare CPS solutions have proliferated to enable remote control of chronic patients, to enable patients to take an active role in

[1] The term "IoT" was coined back in 1999 by Kevin Ashton, although the technology advanced later, in 2011, when IPv6 protocol made it possible to connect "340 billion, billion, billion, billion unique IPv6 addresses".

treatment by providing information and training, and consequently, to reduce the costs to public health associated with repeated consultations with healthcare professionals. This proliferation of solutions has brought about extensive research focused on improving the construction of healthcare CPS.

Research so far can be categorized from multiple points of view. A good example is the work by Carroll [5] that presents a model for assessing the value and potential impact of healthcare software ecosystems from multiple stakeholders' perspectives. From a software engineering perspective, we propose to assess software architecture for the following reasons: software architectures are the core of systems; the lack of focus on architecture is bound to engender suboptimal design-decisions [6]; and an inaccurate architectural design leads to the failure of large software systems [7]. As architectures play a key role in software construction, research on software architectures for healthcare CPS is critical.

This paper reports a systematic literature review (SLR) on *software architectures for healthcare CPS* to evaluate existing solutions. SLRs identify, evaluate, and interpret all available relevant research on a specific objective or question by using a rigorous and auditable methodology that includes systematic protocols for searching, analysis, and synthesis [8].

To date, there has not been an SLR on software architectures for healthcare CPS, making it difficult to assess the maturity of current solutions and to identify trends, gaps in research, or future dimensions. Both European and American initiatives—for example, the European Commission's Horizon 2020 Programme and the National Science Foundation's National Institutes of Health—emphasize the need to draw up a research agenda for healthcare CPS, for example, for ICT solutions for active and healthy ageing and eHealth innovation in empowering the patient.

The research synthesis has resulted in a knowledge base of current software architectures for healthcare CPS. The results of this SLR are beneficial both for researchers who want to identify relevant studies and existing gaps and for practitioners who want to understand the available software architectures that have already been put in practice.

Our ultimate goal is to build a reference architecture for healthcare CPS, hence this SLR pays special attention to the following areas: (i) the architecture needs in this domain; (ii) the characterization of software architectures—including architectural views, architecture styles, main components, and implementation technologies; (iii) case studies where these architectures have been evaluated; and (iv) current challenges from the architectural point of view.

The structure of the paper is as follows: Section 2 describes the background; Section 3 describes the method used for the literature review. Section 4 reports the contributions from primary studies. Section 5 summarizes key findings as well as implications both for researchers and practitioners and Section 6 includes a discussion. Finally, conclusions are presented in Section 7.

2 Background

2.1 Healthcare Cyber-Physical Systems

The term cyber-physical systems (CPS) emerged around 2006, when it was coined by Helen Gill at the National Science Foundation in the United States [2]. CPS refer to the next generation of embedded Information and Communication Technology (ICT) systems that are interconnected and collaborative (cooperative and negotiable), and that provide users and businesses with a wide range of smart applications and services [4]. To that end, CPS integrate computing, communication, and control—denoted by the symbol C3—with real world's objects and physical processes. CPS intend to monitor, control, and interact with a physical process [2], enabling effective and fast feedback loops between sensing and actuation, possibly with cognitive and learning capabilities [4] and leveraging autonomous behavior. To sense and act on the physical environment, CPS are often built on sensor/actor networks, mostly wireless networks [9]. CPS are more and more interconnected through the IoT [3], which encompasses the extension of the Internet into the physical realm, resulting in a global network that interconnects thousands or even millions of "things", or smart objects [10]. Therefore, CPS and IoT are connected terms: CPS is a broader, more foundational and durable term because it is not limited to any particular implementation (such as the Internet) [2]. IoT is a network for connecting things that is able to monitor and process information to offer smart services, whereas CPS includes the control of a physical process—thus, such systems interact with control and management devices—with the Internet as a means and not as the core. Despite the novelty of these concepts, in recent years, CPS/IoT has been applied to a multitude of vertical domains, including smart grids, smart transport, Industry 4.0, digital healthcare, and smart cities, which are large cyber-physical systens(see Fig. 1).

Focusing on digital healthcare, *Connected Health* [5][11], or *eHealth* (in its different variants, such as telemedicine, telehealth, and mHealth), refers to health services and information whose aim is to improve health and that are delivered or enhanced through the Internet and related technologies. This aims to reduce costs compared to a traditional medical service. In particular, telemedicine is defined as the use of ICT to provide clinical services from a distance, whereas mHealth aims the same objective but is supported by mobile devices (cell phones or digital personal assistants). These terms are closer to what we call *healthcare CPS*, but they are not the same. Healthcare CPS is the intersection of eHealth and related variants with CPS and IoT (see Fig. 1), which mainly focus on monitoring biosignals to control physical processes and to provide smart healthcare services. Compared to eHealth, (i) healthcare CPS involves much more physical components, and (ii) cyber-physical system components exchange information with each other—that is why the communication is added in the symbol C3. CPS aims to improve quality in healthcare by providing real-time supervision of patients, who can be monitored and controlled from a distance and who can receive a faster emergency response.

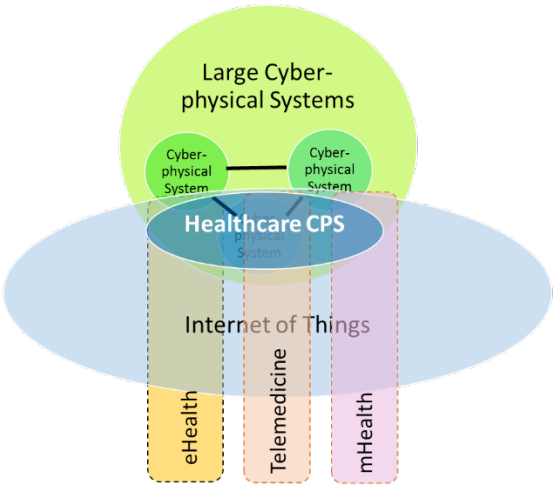

Fig. 1 Identification of the field of study

The rapid appearance in the market of less and less intrusive, miniature (healthcare) wearable sensors (aka biosensors and bodysensors) have made the remote monitoring of patients possible and have opened the door to a multitude of new business models for the healthcare field. The basic function of bodysensors is to monitor the vital signs of patients; then these data are processed and analyzed by a software system that visualizes the information for physicians so that they can provide remote assistance. The most common biosignals to be monitored are vibrations, beats per minute, sounds, gases, movement or lack of movement, body temperature, blood pressure, heart rate, electrocardiogram (ECG), blood pressure, galvanic skin response (sweating), and patient's position; hence excessive sweating may indicate that you are experiencing anxiety or dehydration, a change in position may indicate a fall or faint, and high blood pressure may indicate chest pains and dizziness. Most advanced systems can provide assistance for the decision-making process while determining a diagnosis and its respective treatment, and they can even provide reasoning and autonomous behavior.

According to [12], healthcare cyber-physical systems are categorized by application (assisted, controlled), architecture (infrastructure, data requirement, composition system), sensing (sensor type, method, parameter), data management (data integration, data storage, data processing), computation (modeling, monitoring), communication (scheduling, protocol), security (privacy, encryption) and control/actuation (decision-making, mechanism). Given the importance of architecture to hide the important decisions and characteristics of systems, this paper focuses on software architecture for healthcare CPS, namely software systems that collect, process, analyze, control, and visualize biosignals to provide smart health care services. Specifically, this paper identifies, analyzes, and compares the software architecture of existing healthcare CPS.

2.2 Software Architecture

Software architectures describe the structure of software systems by hiding the low-level details and abstracting the high-level important features [13], accommodating both functional and non-functional requirements. The design, specification, and analysis of the structure of software-intensive systems have become critical issues in software development [14], emerging as a solution for the design and development of large and complex software systems. In fact, it has long been recognized that architecture has a strong influence over the lifecycle of a system and as a critical element in successful development as well as successful evolution of software-intensive systems [15][16].

Due to the importance of the architecture level of systems development, a reliable consensus with a precise definition of a system's architecture was required. The IEEE 1471-2000 (2007) standard [17] and the subsequent ISO/IEC/IEEE 42010 (2011) standard provide a basis for the architectures of software-intensive systems in terms of standardization of elements and practices for architectural description. A set of concepts related to software architectures defined by the IEEE 1471-2000 standard are used in this SLR as follows. A *system* encompasses individual applications, systems, subsystems, systems of systems, product lines, or ecosystems. A system has one or more stakeholders. A *system stakeholder* is an individual, team, or organization with interests in a system [17]. The fundamental organization of a system is manifested through its *architecture*: the system's components, their relationships to each other and to the environment, and the principles guiding the system's design and evolution. A *component* is considered as a black box showing a high level of encapsulation and abstraction as well as the interactions with which it is restricted through its interfaces. The description of architectures is usually organized into one or more views—for example, Kruchten's 4+1 view model [18]. A *view* is a representation of

a whole system from the perspective of a related set of concerns of the system stakeholders [17]. Finally, an *architectural style* entails high level patterns and principles commonly used for applications, that is, reusable solutions to commonly occurring problems in software architecture within a given context. Therefore, a style describes component types together with a set of constraints on how they can be combined. According to [19], architectural styles can be categorized into communication (e.g., SOA, message bus), deployment (e.g., client/server, N-Tier, multitenant cloud) and structure (e.g., component-based, layered architecture). Next, common architecture styles, which were extracted during the SLR, are described to facilitate understanding the SLR reporting.

2.2.1 Service-oriented Architecture (SOA)

Service-oriented architecture offers the benefits of the loose coupling of a service and its provider, adaptability through service composition and orchestration and encapsulation for integration in a highly distributed system [20]. SOA prescribes how computing entities interact in such a way that (i) a requestor only knows what the service's job is and how to request it, and (ii) the service is the only one that knows its implementation.

Many companies are finding that making their applications highly available, scalable, modifiable, and agile is still challenging. *Microservices* is an emerging architectural style 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*" [21]. This is known as *componentization via services* [22]. The principle of the microservices architecture is akin to the Unix principle: *Do one thing and do it well* [22]. 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. This supports the rapid/agile and reliable evolution of an application.

Finally, REST is a software architectural style of the World Wide Web that provides a coordinated set of constraints to the design of web services in a distributed hypermedia system that can lead to a higher-performing and more maintainable architecture [23]. Hence, REST is the architecture more frequently used to create web service APIs (application programming interface) that can be used by any device or client that implements HTTP (hypertext transfer protocol). This makes REST simpler than previous alternatives such as SOAP (Simple Object Access Protocol) and XML-RPC (remote procedure call).

2.2.2 Message bus

The message bus architecture prescribes the use of a message oriented middleware approach to centralizing communication between software systems or subsystems in such a way that some systems generate messages and many other systems consume these messages. The message-oriented middleware can implement message queues, publish/subscribe or relay patterns.

2.2.3 N-Tier

N-Tier segregates functionality into separate segments being a tier located on a physically separate computer. Client/Server architectures, or 2-Tier architectures, segregates the system into two applications, where the client makes requests to the server. In many cases, the server is a database with application logic represented as stored procedures [19]. *Tiers usually* are *monolith, as they implement diverse functions that are combined into a single package deployed onto hardware pre-scaled for peak loads* [21].

2.2.4 Cloud Computing – Multi-tenant SaaS architecture

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": Infrastructure as a Service (IaaS); Platform as a Service (PaaS); and Software as a Service (SaaS) [24]. Multi-tenant SaaS is an architectural style based on a modular design to in which a single instance of an application serves multiple users (tenants). Tenants may be allowed to customize some parts of the application to realize their individual and diverse requirements [25].

3 Systematic Literature Review: The method

A review of the state of the art on software architecture for healthcare CPS was conducted by following the guidelines proposed by Kitchenham [8], who suggests three main phases of a systematic review process: *(i) planning the review*, the phase that aims to develop a review protocol (see Section 3.1); (ii) *conducting the review*, the phase that executes the protocol planned in the previous phase (see Section 3.2); and (iii) *reporting the review*, the phase that is responsible for relating the review steps to the community (see Section 4). The execution of the overall process involves iteration, feedback, and refinement of the defined process.

3.1 First Phase: Planning the Review

This phase consists of the development of a review protocol. The review protocol defines the methods for undertaking a specific SLR, reducing the possibility that the review is driven by research expectations. Protocol development must specify (i) *the review objective and research questions*, (ii) *the search strategy*, (iii) *the explicit inclusion and exclusion criteria*, (iv) *the criteria for evaluating each study*, and (v) *the data extraction strategy and the strategy for synthesizing extracted data*. All these steps are described in the following subsections.

3.1.1 Review objective and research questions

The review objective is to identify and compare current software architectures for healthcare CPS, paying special attention to the architecture needs in this domain and the characterization of software architectures—for example, architectural views, architecture styles, main components, implementation technologies, case studies where architectures were evaluated and current challenges from the point of view of the architecture. The research questions that were identified to achieve the outlined objective are as follows:

RQ1: Why are CPS necessary in healthcare?
RQ2: What architecturally significant requirements should be met?
RQ3: What are the existing software architectures for healthcare CPS and their main characteristics?
RQ4: In which case studies were these software architectures evaluated?

3.1.2 Search strategy

A formal search strategy was required to find the entire population of scientific papers that might be relevant to the identified research questions. The formal definition of this search strategy allowed us to carry out a replicable and open review of external assessments. The search strategy consisted of defining the search space. In this study, the search was performed in six electronic databases (ACM Digital Library, IEEE Xplore, Science Direct, Springer, Web of Science, and PubMed). The searches in this space retrieved a list of primary studies.

Once the search space had been defined, it was necessary to define the search terms to be introduced into the search inquiry forms of electronic databases. The search terms were software architecture, telemedicine, eHealth, cyber-physical systems, and Internet of Things. Acronyms for each of these terms were also added in the search inquiry. Therefore, the research string pattern was the following:
("SOFTWARE ARCHITECTURE") AND (TELEMEDICINE OR EHEALTH OR HEALTHCARE) AND (CPS OR "CYBER PHYSICAL SYSTEM" OR "CYBER-PHYSICAL SYSTEM" OR IoT OR "INTERNET OF THINGS")

3.1.3 Inclusion and exclusion criteria

The review protocol also specifies inclusion criteria (IC) and exclusion criteria (EC) that determine whether each potential study should be considered for the SLR. The lists of IC and EC for the present SLR are shown below:

- Inclusion Criteria (IC)
  - IC1. Type of studies: scientific material written in English, according to the research string pattern defined in the previous section
  - IC2. Documents in the area of software architecture for healthcare CPS
  - IC3. Documents in the area of software architecture for cross-domain CPS that present a detailed case study of the application of those architectures in healthcare systems
  - IC4. The scientific material that has been published up to March 2017
- Exclusion Criteria (EC)
  - EC1. Scientific material is a general term. This SLR excluded short papers, experience reports, summaries of workshops, and papers in the form of abstracts, tutorials, or talks that do not provide enough of the detail required in a systematic literature review
  - EC2. Documents in the area of software architecture that do not provide an architectural solution
  - EC3. Documents in the area of CPS/IoT that mention eHealth as an example but do not present an architecture
  - EC4. False positives: documents that do not match the search string

- EC5. Poor arguments: studies with low relevance to the research
- EC6. Reduction ad absurdum: studies that do not fulfill the IC

3.1.4 Quality, rigor, and relevance assessment

To guide the interpretation of findings in the included studies, and to determine the strength of inferences, we evaluated the scientific quality criteria of the selected studies in order to determine the relevance of the results obtained in the review. These criteria indicate the credibility of an individual study when synthesizing results. The results of the quality assessment of the included studies can reveal potential limitations of the current search and guide future research in the field [26][27].

Kitchenham's guidelines suggest performing a quality assessment of each study; the assessment complements the IC and EC defined above. However, there is no universal agreed definition of "quality." The Critical Appraisal Skills Programme (CASP)[2] defines criteria for assessing the quality of qualitative research. The present systematic review used the quality criteria defined for CASP and those proposed by Dybå et al. [6]. The criteria cover three main issues: *rigor, credibility,* and *relevance*. We summarize the quality assessment in Table 1.

Table 1. Quality criteria. Adapted from [6]

1. Is there a clear statement of the aims and objectives of the research? (YES/NO)
2. Is there an adequate description of the context in which the research was carried out? (YES/NO)
3. Is there an adequate description of the proposed contribution, method, or approach? (YES/NO)
4. Is there a clear statement of findings? (YES/NO)
5. Is the evidence obtained from experimental or observational studies? (YES/NO)
6. Is the study of value for research or practice? (YES/NO)

3.1.5 Data extraction and synthesis strategies

The data extraction step of the SLR allows the identification of relevant information and required data that should be extracted from each of the primary studies in order to answer the research questions. Therefore, in this SLR, the process was carried out by reading each paper thoroughly and extracting relevant data, which were subject to bibliographical management in a set of forms for storing key concepts regarding the aims, findings, and conclusions of each of the studies included. The synthesis process consists of organizing the key concepts to enable comparisons across studies and the reciprocal translation of findings into higher-order interpretations. To that end, we defined the characterization schema shown in Fig. 2. This figure shows the items that have guided the data extraction and synthesis strategies. As mentioned before, the architecture characterization is based on the key concepts defined by IEEE 1471-2000. These concepts are stakeholders, core functional and non-functional features, quality attributes, architectural views, architecture styles, component types, and implementation technologies.

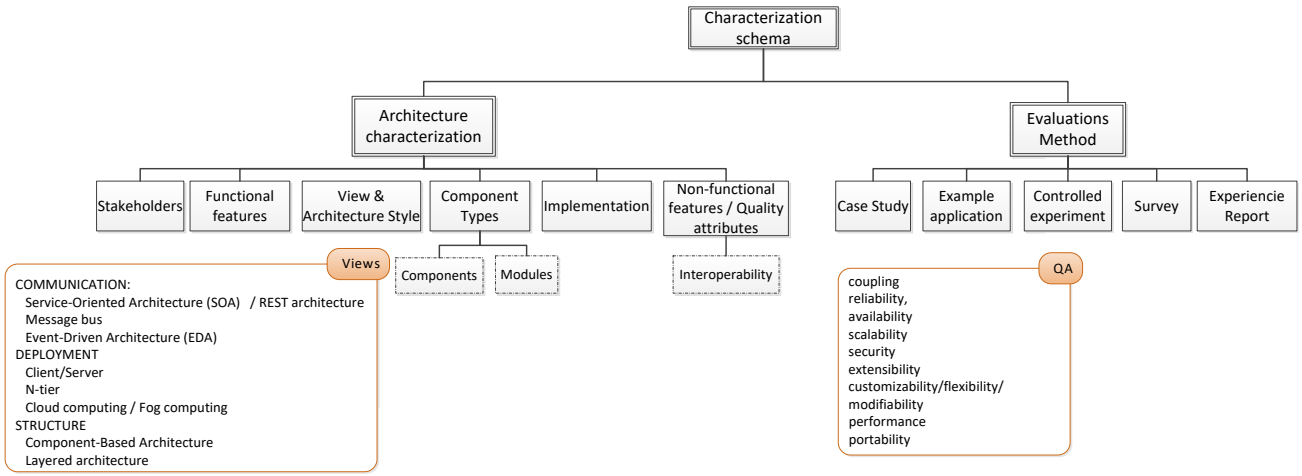

Fig. 2 Software architecture characterization schema

[2] Critical Appraisal Skills Programme (2017). CASP Qualitative Research Checklist. [online] Available at: URL http://www.casp-uk.net/casp-tools-checklists

3.2 Second Phase: Conducting the Review

The second phase of the SLR consists of the following sequence of steps: (i) *search for primary studies*; (ii) *selection of primary studies* by applying inclusion/exclusion criteria; (iii) *quality, rigor, and relevance assessment*; (iv) *data extraction* and *data synthesis*. These steps are described in the following sections:

3.2.1 *Search for primary studies*

Following the guidelines described in Section 3.1.2, a search for primary studies was carried out, using the automatic search functions in the electronic databases identified in the search strategy. We searched documents in the electronic databases according to the defined research string pattern and used a form to classify and write each document's title. The results of the search were also recorded using Electronic Sheet, Mendeley, and EverNote.

3.2.2 Selection of primary studies

The search retrieved 73 results. After retrieving the unduplicated population of scientific papers relevant to the research questions, we obtained 71 unduplicated results. Then, we selected the publications relevant to the review objective according to the inclusion and exclusion criteria defined in Section 3.1.3. In a first round, we evaluated the title, abstract and conclusions of the 71 unduplicated results. In a second round, we read the full text of each paper and retrieved five further papers from the citations of the previous publications. On evaluating these items and applying IC1–4 and EC1–6, 50 studies were rejected. Hence, a total of 21 relevant studies (see Table 2) were entered into a spreadsheet. The inclusion and exclusion decision for each paper was recorded for further discussion and reassessment. The SLR retrieved 21 primary studies, which have been identified from P1 to P21 (see Table 3). Fig. 3 shows descriptive data for primary studies. The topic is new, hence the number of primary studies is relatively low.

Table 2. Study selection

Database	1st round	2nd round	Included	Excluded
ACM	1		0 (0.00%)	1 (100%)
IEEE	5	5	9 (90.00%)	1 (10.00%)
Science Direct	30		6 (20.00%)	24 (80.00%)
Springer	25		1 (4.00%)	24 (96.00%)
Web Of Science	7		1 (14.29%)	6 (85.71%)
PubMed	5		4 (80.00%)	1 (20.00%)
Total	73	5	**21** (26.92%)	57 (73.08%)

Table 3. List of article identifiers

Id.	**Ref.**	**Id.**	**Ref.**	**Id**	**Ref.**	**Id**	**Ref.**
P1	[28]	P7	[34]	P13	[40]	P19	[46]
P2	[29]	P8	[35]	P14	[41]	P20	[47]
P3	[30]	P9	[36]	P15	[42]	P21	[48]
P4	[31]	P10	[37]	P16	[43]		
P5	[32]	P11	[38]	P17	[44]		
P6	[33]	P12	[39]	P18	[45]		

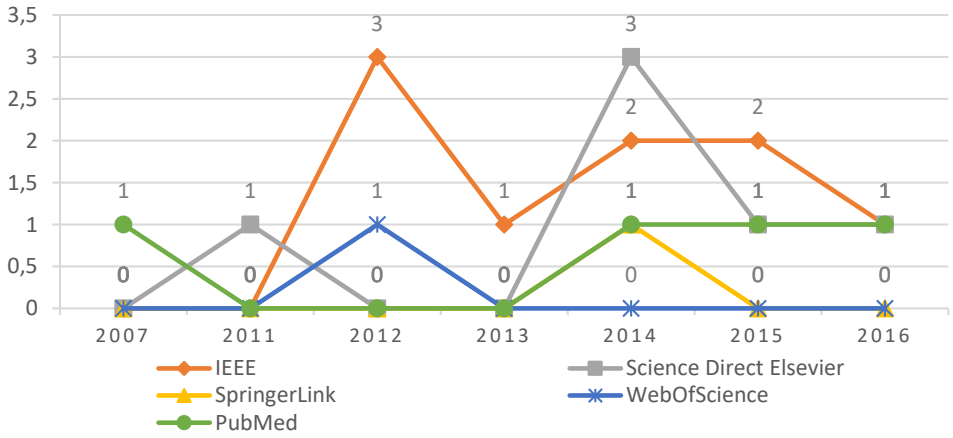

Fig. 3 Papers by year and by database

3.2.3 Quality rigor and relevance assessment

Each study selected was assessed on the basis of the quality criteria defined in Section 3.1.4; 100% of the studies (21 scientific papers) passed the defined quality criteria.

3.2.4 Data extraction and data synthesis

The data extraction and synthesis was carried out by reading the 21 primary studies and extracting relevant data as set out in Section 3.1.5. In order to keep the information consistent, the data extraction for the studies was managed using a form shown in Table 4.

Table 4. Data Extraction for each study

Extracted data	Description
Identity of study	Identifier (P1-P21)
Bibliographic data	Author, year of publication, title, source of publication
Key concepts	eHealth, telemedicine, software architecture, cyber-physical system, IoT
Type of study	Book, journal paper, conference paper, workshop paper
Focus of the study	Main topic area, concepts, motivation, objective
Architecture characterization	Characterization according to the schema shown in Fig. 2
Evaluation method & validity analysis	Case study, controlled experiment, survey, etc.

4 Third Phase: Reporting the Review

The systematic review retrieved 21 scientific papers. They constitute the primary studies that were used to answer the four RQs proposed in section 3.1.1. Next, we present the reporting of each one of these questions.

RQ1: Why are CPS necessary in healthcare?

The World Health Organization indicates that approximately 2 billion people will be 60 or older by 2050 and 80% will be in underdeveloped countries [49]. [P3] highlights that, according to the World Health Organization, chronic obstructive pulmonary disease (COPD) will be the third most common cause of death in 2030. Additionally, the European Union reveals that the main chronic pathologies are cardiovascular disease (CVD), which, together with respiratory disease, is the leading cause of death in Europe. Thus, the main motivation of eHealth systems is the monitoring of patients, especially those with chronic diseases, by doctors and health centers to provide timely remote assistance [P1]. Furthermore, the greater the amount of patient information, the greater the accuracy of decision-making in diagnosis and prescribing of treatments. The applications are extensive, including heart attack control, diabetes control, asthma, Parkinson's, obesity, detection of accidents, rehabilitation after interventions, treatment reminders (pills), emotional and cognitive recognition to facilitate social interactions, location of people with orientation disorders, and medical assistance in case of natural disaster such as earthquake. A particular type of eHealth systems such as telemedicine systems are becoming more important for a variety of reasons, such as caring for an ageing population [P2], providing timely diagnosis, and adjusting rehabilitation therapies [P4]. "Vital signs are indicators that reflect the physiological state of vital organs (brain, heart, lungs). They express immediately the functional changes that occur in the body, changes that otherwise could not be qualified or quantified" [50] and are fundamental for determining the state of a patient.

As the number of vital signs to be monitored increases, these signs are more complex, and it is necessary to control a complex physical process (e.g., a diseases and its treatment), novel solutions emerge. Some primary studies describe their solutions in terms of Cyber-Physical Systems [P6][P13][P18], Internet of Things [P1][P3][P15][P16][P17][P19][P21], and others in terms of Future Internet (FI) [P7]. Similar to the concepts of CPS and IoT, FI is envisioned as a system in which heterogeneous resources are communicated to each other to provide services to the real world. In fact, all these terms emphasize the heterogeneity issue. In the CPS/IoT domain, communication occurs between heterogeneous devices (e.g., bodysensors), where the wide variety of manufacturers has generated a need for software platforms and middlewares that support the integration of such devices.

RQ2: What architecturally significant requirements should be met?

Software platform and middleware healthcare CPS should meet the architectural requirements described by primary studies:

- Heterogeneity and dynamic discovery of sensor. Healthcare CPS requires the ability to utilize a large variety of body sensors through platforms that support high-level interfaces to access the sensors in a transparent way, hiding specifications and protocols to the users and the applications that make use of them [P1][P4][P5][P7][P13][P15]. In addition to heterogeneity support, *sensor discovery* is also important for dynamically integrating new devices into healthcare CPS [P4][P5][P11][P15].
- Scalability. The huge amount of data that is expected to be generated by the sensors of eHealth systems requires scalable storage and processing capabilities [P3][P4][P10][P13][P18][P19][P21]. Real-time data streams from bodysensor networks (BSN) may trigger contextual and situational events. Managing and processing these events (aka *big data analytics*) can be useful for decision-making [P2][P13][P18][P19][P21] and may, in turn, trigger services as a response to these events.
- Interoperability. Data transmission to cloud storage and processing services often requires improved data quality and involves data conversion into standard formats (for example, HL7[3]) to deal with interoperability issues, data cleaning, redundancy reduction, and aggregation [P13][P18].
- Dynamic reconfiguration. Criticality of the healthcare domain requires reconfiguring systems to adapt software to changes in system environments at runtime, i.e., without disturbing the operation of those parts of the system unaffected by the change. This means that a system is aware of its environment and can self-adapt its behavior to the environmental conditions [P1]. The paper [P5] contributes to the building of new context-aware applications that interpret the state of the physical world in realtime in order to allocate computing resources efficiently.
- Security and privacy. Authenticity and integrity are key given the need for confidentiality of data [P16][P18]. Papers [P4][P5][P10][P13] include in their solutions specific layers for security (confidentiality and authenticity), data integrity, and data encryption.

RQ3: What are the existing software architectures for healthcare CPS and their main characteristics?

To answer RQ3, we used the characterization schema defined in Fig. 2. Therefore, our description and comparison of existing software architectures for healthcare CPS is based on the following categories: stakeholders, core functional features, architectural styles and views, components, implementation technologies, interoperability issues, and other core non-functional features and quality attributes.

Stakeholders

The *EcoHealth* (Ecosystem of Health Care Devices) platform [P1] defines a set of stakeholders who participate in the system: (i) **device manufacturers**, who develop device *drivers* compliant with the EcoHealth API in order to make possible the connection between devices and the platform; (ii) **doctors/clinicians**, who continuously monitor the data collected from patients via the EcoHealth platform and use this information to improve diagnosis and response in emergencies; (iii) **patients**, who provide information to the platform via attached body sensors; and (iv) **system administrators** of users and hospital activity. The architecture proposed in [P2][P15] and [P17] simplifies this stakeholders' definition by focusing only on patients and doctors. In this paper, device manufacturers are not considered as the approach is based on the use of the sensors on board smartphones, and therefore, drivers are already embedded. In addition to patients, doctors and administrators, [P3] also identifies social workers and family caregivers who have full access to patient information

[3] HL7 Inc. Official Web Page, [online] Available at: http://www.hl7

anytime and anywhere. In line with [P2], device manufacturers are not considered, because the architecture described in [P3] only supports the exchange of data between biosensors and a mobile device through the Bluetooth wireless technology standard. The BodyCloud platform [P4][13] includes new stakeholders in addition to patients and doctors: **cloud providers** and **developers**. This platform provides developers with web-based programming abstractions for rapid development and deployment of BSN (body sensor network) applications in a broad range of domains, including health care monitoring, emergency management, fitness monitoring, and behavior surveillance.

In contrast to the previous papers, [P6] only focuses on an interface to automatize the reading of vital signs in *electronic medical records* (EMR) systems; thus the only client is a **health center** and the end user is the **medical staff**. Similar to [P4], [P6] includes a new stakeholder: the developer/maintainer as the intended recipient of the readability, adaptability, and well-definedness of the interface. [P8] considers doctors, pathologists, pharmacists, nurses, and relatives as stakeholders of the system. [P9] includes the nurses who provide aid servive in the ambulance, and the doctors who supervise the vital signs and analyze the case remotely. [P10], which presents an electro cardiogram mobile application for athletes, includes doctors, physicians, and sports coaches. The iSenior system [P11] proposes a solution for the monitoring of the elderly, inside or outside the nursing home; thus the stakeholders that this work describes are caregivers, system administrators, and patients. Finally, [P14] focuses on the implementation of humanoid robots for the care of diabetics, offering caregivers a set of services for monitoring their patients.

Core functional features

The EcoHealth platform [P1] provides doctors with management of all information about their patients as well as data coming from bodysensors attached to them, medical records, and notifications about problems, symptoms, or anomalies in the vital signs measured through the sensors. EcoHealth supports heterogeneity of sensors, and therefore heterogeneity of the data that are managed, as long as (EcoHealth-compliant) drivers exist for those sensors.

The system presented in [P2] supports the monitoring of Parkinsonian patients by means of a mobile app able to automatically detect a motor disorder called *freezing of gait* (FoG[4]) by means of a set of sensors on smartphones and to generate an alarm when an FoG is detected. This system manages very specific types of data, such as step length, step frequency, and freeze index, through a specific type of sensors onboard smartphones. The system also supports clinicians by providing a web app that allows them to consult all the information about their patients' FoG episodes as well as to update patients' pharmacological and rehabilitation therapies. Patients can also see their pharmacological and rehabilitation therapies, and they receive alerts as soon as a change involving any therapy has happened.

Similar to [P2], the system for monitoring chronic patients described in [P3] supports early diagnosis by means of a smart mobile device able to process data from Bluetooth sensors, such as pulse oximeters, accelerometers, and body temperature sensors. The device is also able to activate alarms to be sent to the chronic patient's caregivers and medical personnel. Therefore, both [P2] and [P3], unlike [P1], support *in-house* and *outdoor* monitoring. The system [P3] also supports social workers and family caregivers by providing a web app for consulting the associated patient information. Finally, the system provides a utility for (natural language-based) searching of health information in medical sources.

The *BodyCloud* platform [P4] supports the management and storage of large amounts of data streams coming from bodysensor networks that monitor patients' health status (e.g., heart attacks, diabetes, and asthma), as well as the processing and online and offline analysis of these data to support physicians in decision-making. The data are shared in real time with doctors who are responsible for a particular patient, and they can provide instructions to patients through alarms. The novelty resides in the fact that this platform offers developers workflow-based programming abstraction of data analysis—that is, developers can configure and customize how data have to be processed and analyzed. Similarly to [P4], the Health-CPS proposed in [P18] manages large amounts of data. The system supports collection of data through a set of *data nodes,* and these data are sent to the cloud through the adapters, real-time and off-line analysis, processing data mining algorithms, and automatic learning.

The Knowledge-Aware Service-Oriented *(KASO) Middleware* [P5] has been deployed with two core functions in a healthcare scenario: tracking patients and medical staff through a virtual perimeter around sanatoriums by using a bracelet node, and monitoring biosignals through two biomedical sensors, a heart-monitor sensor and a body temperature sensor. Additionally, a set of intermediate nodes were deployed all around the sanatorium area to monitor the temperature, humidity, and light of the environment in order to make inferences from both the data provided by biomedical sensors and environmental sensors. The middleware offers (i) discovery mechanisms for dynamically integrating new devices into the healthcare scenario, (ii) context-awareness for dynamic and efficient allocation of resources, and (iii) programming abstractions for rapid prototyping of scenarios based on sensors and actuators.

[P6] aims to automate the readings of vital signs using biosensors for existing electronic medical records (EMR) systems. To that end, [P6] designs and implements a cyber-physical interface that captures vital signs for a type of encounter through a set of custom vital signs forms that collect specific medical data.

The *e-SURAKSHAK* system [P8] is capable of real-time, continuous, and autonomous monitoring of patients' vital signs. These vital signs come from 'wireless embedded Internet devices' (WEIDs) and are stored periodically in an electronic database and a flash memory. Furthermore, this system implements a web-based GUI to visualize measurements in real time, it being possible to configure a WEID to make periodic measurements. It also includes an alarm to notify medical staff about network connectivity problems.

[4] "*FoG is a motor block that manifests as a sudden and transient inability to generate effective stepping despite the intention to walk"*

The *Patient Monitoring System* [P9] supports patient health monitoring, video streaming, and audio communication through VoIP from an ambulance on an emergency call. The authors propose an application to monitor the patient's vital signs without delay in data transmission, thereby allowing the doctor to begin the analysis and treatment while the patient is still in the ambulance. These services are available through two GUIs, one for the doctor at the hospital showing real-time data acquisition from medical instruments and voice and live camera feed, and one for ambulance personnel that shows navigation information.

The *ECG Android App* [P10] provides patients with a visualization of their electrocardiogram (ECG) waves. A patient uses ECG electrodes, and the data measured are sent to mobile devices and stored in the patient's private cloud or a specific medical cloud.

The *iSenior* system [P11] supports monitoring, alerting, and requesting assistance. An elderly patient carries a personal monitoring device (Elderly Monitoring Device or EMD) that enables caregivers to monitor the patient's biosignals in real time, to analyze historic data, to locate people in case they need assistance, and to receive alerts. The system has these two functions: (i) *data collection and storage:* the EMD generates information that is stored in the database and can be viewed through a web-based user interface; and (ii) *alarm management:* the system allows doctors to set up alarms over the deployed sensors by means of rules on the parameters under observation. When the threshold defined by a doctor is met, the alarm sends an SMS message or an email to a predefined group of cell phones and email addresses and shows the alarm in the web interface. The system is configured to send an alert in the event of low battery; (iii) *localization:* the system allows users to be located and their trajectories to be followed on Google Maps; (iv) *remote access:* the system implements authentication and access to the functionalities of the system; (v) *user profiles:* the system implements three profiles: caregivers who have permission to verify and analyze data, a system administrator with access to all information, and end-users (the patients); (vi) *auto-configuration:* the system allows EMD devices to be registered by adding them to the database when they are powered on.

The *Activity as a Service* framework [P13] is built on the BodyCloud platform [P4], thus enabling the collection, computing, and storing of sensor data, workflow-based programming of data analysis, and visualization of results. The authors of this framework define Activity as a Service as "a full-fledged cyber-physical framework to support community, online and off-line human activity recognition and monitoring in mobility." [P13.] The novelty of this service is that the high-level BodyCloud abstractions are specialized by defining three models depending on the (pre)processing carried out locally or in the cloud. Thus, the three models are *full cloud* (raw data are sent to the cloud, which performs all the necessary processing), *mix cloud* (there is some pre-processing on the body-side before the data are sent to the cloud), and *full local* (data are collected and processed in full on the body-side and then sent to the cloud for long-term storage and historical visualization only). In [P13], the authors apply the programming abstractions to human monitoring applications for posture and activities monitoring and step counting (including authentication, geolocation, and computation of the time percentage spent for each performed activity), physical energy estimation (a kcal estimator), automatic fall detection (including an alarm system that connects to Facebook and sends a notification to predefined friends as relatives or caregivers), and smart wheelchair support (wheelchair overturn detection body posture recognition).

[P14] incorporates humanoid robots to support emergencies in the treatment of diabetes. The platform provides the following features: (i) the *application access interface*, which provides unified access to disease management hub applications when it receives a request; (ii) *validation* of incoming requests; (iii) *authentication* using HTTPS session, cookies, or access keys, which—if correct— forwards the request to service resolution; (iv) *service resolution*, which generates an instance of the service; (v) *authorization*, in which the authorization unit verifies that the object has permission to access the service, and (vi) *service execution*.

[P15] presents a system called *CardioNet*, which includes a monitoring system that facilitates remote supervision of patients with cardiovascular diseases through certain portable devices. This system defines an ontology to semantically interpret acquired medical data as well as to exchange data seamlessly between autonomous entities (general practitioner cabinets, hospitals, laboratories, emergency units, medical personnel, and patients). The ontology can be adapted for the treatment of other diseases. CardioNet allows patients and doctors to interact in three ways: (i) online (patients and doctors interact through a web interface), (ii) off-line (the system provides asynchronous communication between these actors), and (iii) face-to-face. Additionally, the system allows remote consultation and administration of patients' records.

Unlike the other proposals in which monitoring is more focused on vital signs, *VIRTUS* [P16] proposes to monitor the patient's movements, that is, to count the number of steps and to identify postures during the day. With this information, it is possible to analyze (i) the time lapse between activities, and ii) a patient's reaction after the administration of a drug.

[P19] proposes telemonitoring of health-related parameters— speed and direction of movement, pulse, and blood oxygen saturation— allowing users to access processed data from any device with Internet access (including PCs and tablets). Finally, the project [P21], unlike the others, implements "Intelligent Building" for the processing of the large amounts of data obtained from sensors connected to the body (e.g., wrist, ankle, heart, and chest).

Architectural styles and views; components

Most of the primary studies under review describe software architectures for healthcare CPS using (i) the *deployment view* (aka. physical view) through the architectural styles N-tier, client/server, and cloud computing; (ii) the *structure view* (aka. implementation view) through the architectural styles component-based architecture and layered architecture; and (iii) the *communication* view (aka. process view) through the architectural styles service-oriented architecture (SOA), message bus, event-driven architecture, and REST architecture.

The architecture described in [P1] is a three-tier architecture: a network of wireless body sensors (the **perception tier**) communicates with a set of components referred as drivers (commonly referred as the **gateway tier**) that subsequently communicate via the Internet with the *EcoHealth* platform (the **application tier**). The architecture's implementation view of the EcoHealth platform [P1] is composed of a set of modules (aka. middleware services). The *devices connection module* integrates physical devices (wireless bodysensors) to the platform through the drivers that different manufacturers develop according to a defined API (the EcoHealth API). These drivers can access data by continuous polling or under request after a notification. They are built upon REST principles, and are based on the protocol HTTP (compliant with the web of things paradigm). In this way, bodysensors are viewed as web resources accessed, controlled, and monitored via the Internet. The *data manipulation module* registers data sent to EcoHealth via HTTP PUT requests from drivers. The *storage module* includes a relational database and a file system for storing all data used in EcoHealth that can take advantage of storage cloud-based infrastructures. The *common services module* consists of a set of general services such as security and notifications (e.g., event-based notification mechanisms that report problems, symptoms, or anomalies in the measured vital signs). Finally, the *visualization and management module* provides a web interface for giving stakeholders (doctors, patients, and system administrators) access to core functional features.

The architecture described in [P2] is also a three-tier architecture: there are sensors onboard smartphones, and the smartphones act as gateways that send data to the core component, that is, the clinical remote server. The implementation view of the architecture is composed of the following components: a smartphone app, a database, and a web app. The smartphone app monitors gait parameters, detects FoG episodes, generates alarms in response to them, saves these data in a local database, and periodically uploads these data to a central database of the clinical remote server.

The architecture described in [P3], like [P1] and [P2], presents three tiers: vital sign sensors and their connectivity constitute the **perception tier**; a smart mobile device constitutes the **gateway tier**, which captures vital signs from sensors and sends data to the medical platform; and finally this medical platform constitutes the **application tier**. The implementation view of the application tier has the following modules: a *smart mobile device* that, in addition to capturing vital signs from sensors and sending this information to the medical platform, implements a rule-generation system able to process intermediate data and generate alarm messages to be sent to the patient's caregivers or medical personnel; *data storage and processing modules* that are able to manage massive amounts of data; an *interoperability and messaging platform* for the delivery of information to all stakeholders via SMS, voice, and PUSH technology; a *website platform* that allows patient information to be consulted; and a *health-related information searcher* in various medical sources.

The BodyCloud platform [P4] is a multi-tier architecture. To the best of our knowledge, it is a four-tier architecture. The lower-level layer is the **perception tier** constituted by the body sensor network (BSN), which is composed, in turn, by a set of sensor nodes that measure bio-signals and send raw or pre-processed data to a coordinator. Sensor nodes provide features that are more advanced than previous work, such as a standard interface to the diverse sensor drivers,and the customization and extension of the platform to support new physical sensors and signal processing services. The **gateway tier** is for the coordinator; it manages the sensor nodes (e.g. the coordinator can notify the discovery of new sensors), collects, stores and analyzes the data received from sensors, and act as a gateway to connect the BSN with a clinical remote server. These two tiers are called the *body-side*. The **application tier** supports a distributed real-time system for the monitoring and analysis of BSNs data streams as well as a system for visualizing the output of the data analysis through advanced graphical reporting. The former is deployed on Cloud through an SaaS (Software as a Service) model and is called the *cloud-side*, whereas the latter is deployed as an HTML/Android client application and is called the *viewer-side*. Finally, it is possible to consider a new tier, the **programming abstraction tier**, which provides developers with programming abstractions for rapid prototyping of BSN applications and is called *analyst-side*. To that end, the authors have defined a set of programing abstractions as follows: *group* to formalize a BSN data source; *workflows* to define actions to be realized on the input data —for example, storage, processing, analysis—; *views* to formalize outputs for viewers; and *modality* to formalize the interaction between the input data, the actions, and the output data. This last layer is a novel contribution in the domain of healthcare systems.

KASO Middleware [P5] is also a three-tier architecture, although the tiers are described as layers (see Fig. 4). KASO Middleware is designed from an agent paradigm called Perceptual Reasoning Agents (PRAs). The lowest layer corresponds to the **multifunctional embedded layer** (where the hardware platform is located), the real-time operating system to hide the hardware heterogeneity from higher layers, and the network protocol layer that transmits packets in a multi-hop communication. The intermediate layer is the **middleware layer,** which is made up of three subsystems: (i) *framework services* composed of *query services*, through which information can be consulted on specific parameters, a *runtime manager service* in charge of load/unload and start/stop components and PRAs, a *security service* to manage the access permissions and encryption of the information, and a *configuration service* to set up parameters at low level; (ii) *communication services* formed of three modules enabling low-level resource discovery, service exposure and discovery, and internet-working communications; and (iii) *knowledge management services* (KMS) implemented by a broker and an orchestrator of services that implement an API offered to developers of PRAs. The top layer is the **agent layer**, where PRAs are defined.

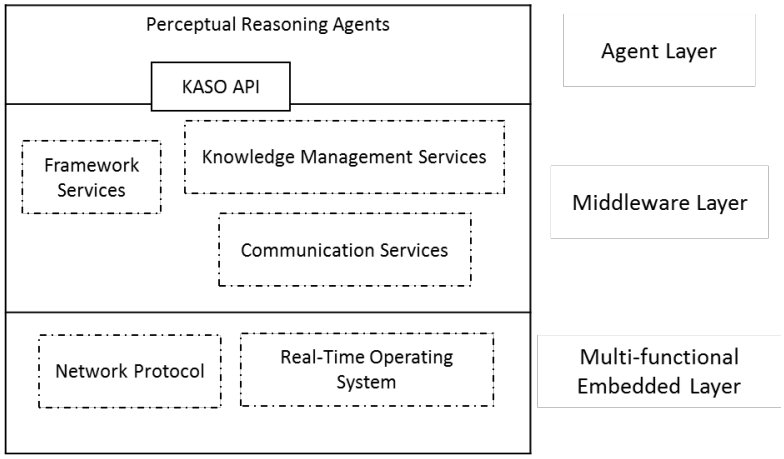

Fig. 4 Architectural system for KASO middleware [P5]

The paper [P6] proposes the design of a cyber-physical interface for automating biosignal readings in EMR systems, by describing a three-tier architecture. It is necessary to highlight that the authors refer to tiers as layers (see Fig. 5). Hence, they describe the following architecture decomposition. The *interface layer* includes (i) the *vital signs reading station* (an application that reads the biosensors and stores the information in the internal memory of the device where it is deployed), and (ii) the *custom vital signs form* (which supports the automatic reading of biosignals from the reading station). The *application logic layer* is responsible for collecting data from the reading station, displaying them in the form, and sending them to the data layer. The *data layer* denotes the existing EMR system.

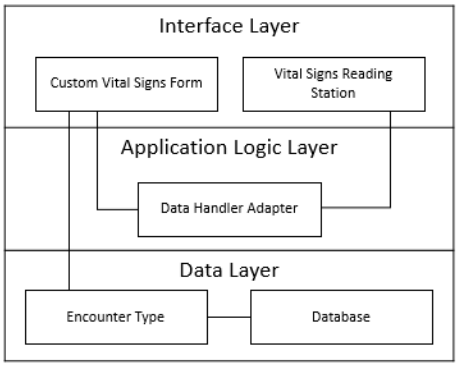

Fig. 5 Architectural system for automating biosignal readings in Electronic Medical Record (EMR) systems [P6]

PRIME [P7] proposes a two-layer architecture, consisting of the **communication layer**, which exploits the overlay using devices in an overlay network built on top of low-level wireless communication technologies, and the **API programming layer**, which provides the abstractions and operations to implement P-RESTful (Pervasive REST) applications.

E-SURAKSHAK [P8] consists of a service-oriented architecture (SOA) with three tiers. The **environmental tier** (equivalent to the perception tier of previous work) is composed of WEIDs and routers, and senses vital health parameters of patients. The **service tier** is composed of basic and complex analysis functions. Finally, the **control tier** stores and processes information gathered and provides a web-based GUI.

The Patient Monitoring System (PMS) [P9] also presents a SOA that is implemented using an OSGi framework.[5] OSGi is an initiative focused on the interoperability of applications and services based on the component integration platform (service platform) providing a service-oriented, component-based environment. The *services layer* connects bundles—OSGi components— in a dynamic way through a publish-find-bind model for plain old Java objects. The *life-cycle layer* implements the API to install, start, stop, update, and uninstall bundles. The *modules layer* defines how a bundle imports and exports code. The *security layer* guarantees secure access. Finally, the *execution environment* defines which methods and classes are available in a specific framework. The PMS defines a set of bundles for secure login, video streaming, patient medical monitoring, patient devices, audio communication and GPS navigation.

[5] OSGi, "Osgi alliance," [online] Available at: http://www.osgi.org

ECG Android App [P10] implements an end-to-end system architecture. It is necessary to highlight that the authors describe a three-tier architecture but refer to tiers as layers. Hence, they describe the following architecture decomposition: the **hardware layer** includes microcontrollers and sensors; the **application layer** implements the Model/View/Controller of the Android App for the interaction with users, retrieves the bio signal data from the hardware layer, stores the data in a buffer and a binary file, and transfers the file from the android device to an FTP server on the **cloud layer**.

The iSenior system [P11] implements the following modules: (i) *portal*: this module deploys the interface that allows authentication and access to iSenior. The portal displays information on vital signs from Elderly Monitoring Device through records and alerts, provides history analysis, and tracks patients using Google Maps, alarm configuration, and monitoring functions; (ii) *kernel*: this module manages communication between the remote medical server and EMDs; (iii) *alarm processing*: this module is responsible for activating alerts by detecting abnormal conditions, sending SMS alerts, and implementing raw data processing algorithms

[P15] also proposes a three-tier architecture, which is composed of a *modular portable device prototype* (the **perception tier**), which includes the module processing unit **(**responsible for signal conditioning and processing), the *data acquisition* module, which measures the parameters, and the *data communication* module, which is responsible for sending the data obtained to the gateway or access point. The **gateway** manages communication; this proposal developed two communication protocols—device to gateway and gateway to server. The modules for *activity recognition* and *complex patient monitoring* both implement health services (the **application tier**). The former identifies the activity that is being realized by patients (e.g., walking or running), while the latter monitors a set of eight parameters (e.g., ECG, blood pressure, airflow, and oxygen level).

VIRTUS [P16] proposes a layered architecture based on SOA, which is composed of: *wearable devices*, which collect the data related to the activity of the patient; *middleware*, which allows the connection between the modules of the system; *SmartDevice*, which is the interface between the user, the system, and the remote center; and the *central unit* or *project manager software* in charge of classifying the data and generating daily reports and feedback to the patient.

Health-CPS [P18] proposes similar architecture to VIRTUS [P16] with the following layers: the *data collection layer*, which consists of data nodes, adapters, and an interface for multisource heterogeneous data; the *data management layer*, which contains a distributed file storage (DFS) module and a distributed parallel computing (DPC) module; and the *application service layer*, which provides the visual presentation of results and an open unified API for developers.

μWoTOP [P17] implements a layered architecture: the *IoT ecosystem layer*, which is responsible for adapters identifying the potential resources (sensors and actuators), classifying them based on their topology, and announcing them to the upper layers; the *Web of Things middleware layer*, whose objective is to provide functionalities so that the designers and developers of services can design, develop, and deploy smart spaces; and the *resource composition and orchestration layer*, which is responsible for deploying the necessary components so that services and functionalities can be offered.

[P19] proposes a decentralized cloud architecture with three tiers: *sensors and actuators* (the **perception tier**), the **gateway tier** in charge of the communication between resource management and Remote Telemetry Units (RTUs), and presentation and application services (the **application tier**).

Finally, [P21] also proposes a layered architecture: the *data collection layer* is responsible for obtaining and processing the data; the *communication layer* provides end-to-end connectivity, aggregation of measured data from other devices, and transformation into the format; the *processing layer* performs statistical calculations; the *management layer* consists of a medical expert system that suggests actions based on the data generated by the processing layer; and the *service layer* provides access to the end user (hospitals, emergency services, ambulances, and police stations), as well as providing an analysis based on the patient's medical history.

Implementation details

The EcoHealth platform [P1] is implemented in the Java programming language and it is deployed on a JBoss application server. Data is stored in a MySQL relational database using the Java Persistence API (JPA) specifications implemented in the Hibernate framework. The data to be managed are described as follows: medical records, historical data from bodysensors, notifications, patient-related information, and information about emergency services. The web interface provided is implemented with the JavaServer Faces (JSF) technology. According to the authors' specifications, drivers should implement a RESTful interface for clients (humans or applications), specifically the RESTEasy implementation for the REST architectural style and the Java API for RESTfulWeb services (JAX-RS). Finally, the authors use the Java Authentication and Authorization Service (JAAS) specifications implemented in the JBoss server to provide user authenticity, confidentiality, and integrity.

In the architecture described in [P2], the novelty and complexity of the contribution is the fuzzy logic algorithm and the fuzzy inference system—implemented through a smartphone app—to determine a FoG episode based on a set of gait parameters: step length, step frequency, freeze index, and energy. The smartphone app was developed for iOS and Android operating systems. Data are stored in a MySQL relational database in the clinical remote server. The information to be stored is as follows: web app users' personal details, patient monitoring data from the smartphone app, and pharmacological and rehabilitation therapies assigned by clinicians to patients. (The reader can refer to [P2] for further details about the entity–relationship model).

The novelty of [P3] resides in the use of a distributed non-relational database to store very large amounts of data, in the following way. A smart mobile device implements continuous polling of sensors. The data are stored using files whose records have a format of a tuple of three elements (sensor_id, time_stamp, value), and each record occupies 10 bytes of memory. When the file size reaches 500

KB, it is closed and another file is created to save the most recent data. The data files saved in the smart device are synchronized on a server on the Internet—that is, the data are transmitted to the server via a secure, distributed P2P file synchronization mechanism (*BitTorrent Sync* protocol). Due to the large amount of data to be processed on the server side, [P3] uses a RIAK[6] database, which is a distributed, scalable, open-source *key/value* non-relational database. In RIAK, the only unit of data storage are *objects*, which can be isolated through virtual key spaces (*buckets*). Therefore, data are organized as follows: (i) a bucket for users where the *key value* is an identifier and the *value* is an object containing the user data and a list of sensors identifiers associated with this user, and (ii) a bucket for sensors where the key value is formed by the user and device identifiers and the value is an object containing the necessary data to identify the most recent data transmitted by one sensor.

[P4] integrates Cloud Computing and BSNs. The cloud-side has been developed using Google App Engine (GAE). Specifically, the BodyCloud uses (i) the GAE Datastore API for the persistence of data streams, (ii) the GAE Task Queue API for complex event processing to implement ad hoc data analysis applications and integrate third-party data mining workflows (e.g., WEKA [51]), and (iii) OAuth 2.0 [52] for the authorization system. The interaction between the components of this solution—body and cloud (data acquisition), cloud and viewer (data visualization), and cloud and analyst (definition of new data analysis services)—is supported through a RESTful web service, implemented using the Restlet Framework.

The novelty of KASO Middleware [P5] resides in the possibility of dynamic reconfiguration of resources (e.g., sensors) depending on the environmental conditions. To that end, this approach defines a set of Knowledge Management (KM) services that depend on an Information Model of the system that describes low-level and high-level resources. The data is stored in Knowledge Bases (KB)—Mote Knowledge Base, Contextual Knowledge Base, and System Knowledge Base—which are deployed on sensor and actuator nodes (SA nodes), contextual nodes (C nodes), and gateways, respectively. The KB are structured through an ontology over an RDF/OWL 2, and the services are mapped over standardized REST web services.

[P6] implemented a cyber-physical interface to automate the readings of biosignals in EMR systems. The implementation of this interface focused on a proprietary and licensed ERM: Centricity Practice Solution®, which is deployed on a Citrix server. This work implemented a wrapper (or adapter) to extend the functionality of the ERM, that is, to automate the population of EMRs from the sensors and store the data into the databases.

The core features of the Patient Monitoring System (PMS) [P9] are service-orientation and service-aggregation using an Eclipse OSGi framework and the Java class loader architecture for deploying the services at runtime. The PMS includes a GPS navigation system accessed via the internet through TCP/IP using an ad hoc network. It supports (i) video streaming through IP cameras with filtering and a transmission ratio of 1:1 (the authors are not using any proprietary software), (ii) patient health monitoring by preconfiguring the OSGi framework so that any medical monitoring device sends its data over the ad hoc network and displays the results in the form of a graph, and (iii) audio communication through VoIP.

The ECG Android App [P10] uses Filestream and Filetable technologies for storing unstructured data. It implements a microcontroller, which obtains the biosignals of a patient and sends them to the mobile device wirelessly using Bluetooth technology. The data are stored in a binary file encrypted on the device's secure digital (SD) card, after which the patient can upload the binary files to a centralized private cloud using an FTPES secure server (via FTP).

As a novelty, the [P15] presents a neural network to determine a type of activity to be realized by patients, such as sitting, standing, walking, running, or an undefined activity.

The distinctive features of VIRTUS middleware [P16] are (i) the use of Java programming language, (ii) the use of XMPP (eXtensible Messaging and Presence Protocol) instant messaging, and (iii) the use of an OSGi framework.

Finally, [P21] uses real-time processing and MapReduce technologies (Apache Spark and Hadoop) to process data and to analyze and calculate statistical parameters.

Interoperability

In [P1] the authors define three levels of interoperability: (i) "*in the lower level it is necessary to seamlessly integrate a myriad of heterogeneous physical devices with each other"*, (ii) "*at an intermediate level, data provided by the devices need to be easily made available on the Internet"*, (iii) "*at a higher level, a standardized programming model can provide transparency assembling and transforming information from sensing devices without requiring any specific knowledge regarding the specificities of these physical devices and the networking environment*."

The *EcoHealth* [P1], the BodyCloud [P4], the KASO [P5], and the Activity-aaService [P13] platforms deal with the three levels by (i) encapsulating the heterogeneity of devices into drivers to be complaint with an EcoHealth API, (ii) making the data (*feeds*) provided

[6] [online] Available at: http://basho.com/products/

by the devices available to the Internet through the HTTP protocol and RESTful APIs, and (iii) structuring feeds using XML or JSON. The EcoHealth platform uses the Extended Environments Markup Language (EEML)[7] to address data interoperability issues between body sensors from different manufacturers. EEML is an XML-based language for describing sensor data collected from a device, building, system or space in a structured form (in this case, human body vital signs collected from body sensors). However, none of these platforms addresses standardization for medical records, nor are relational models for historical data described in detail.

[P2] and [P8] only contemplate interoperability at the lower and intermediate levels. In [P2], the integration of sensors is encapsulated by smartphones—as this approach is based on sensors onboard smartphones—and sensor data can be accessed via HTTP. In [P8], the authors define the interoperability in the environmental and service tiers. In the former, they define *Edge Routers* and *WEIDs* associated with patients, whereas in the latter, they establish services independent of the context or the state of the other services.

[P3] and [P10], by contrast, implement a microcontroller-based smart mobile device with a Bluetooth module, and thus do not focus on supporting heterogeneity of sensors.

The proposed solution in [P6] is based on the Spot Vital Signs LXi® reading station, which allows the reading of vital signs from a reduced set of compatible sensors.

To preserve interoperability with other solutions, mechanisms have been developed to allow VIRTUS to exchange data with modules of platforms that use different communication protocols than XMPP (eXtensible Messaging Presence Protocol), such as web services for SOA. The proposed system [P19] integrates a home gateway into a cloud middleware, thereby allowing heterogeneous devices to integrate data from other healthcare platforms.

The system [P20] proposes integration using the HL7's CDA (Clinical Document Architecture) standard, because this is the world's leading medical ICT standard that is envisaged as providing the umbrella for medical data semantic interoperability.

Core non-functional features and quality attributes

The EcoHealth platform described in [P1] promises (i) loosely coupled modules, (ii) reliability, availability, and scalability depending on storage cloud-based infrastructures, and (iii) security in terms of user authenticity, as well as integrity and confidentiality by encrypting data transmitted by sensors via Internet (e.g., only the assigned doctor can access a patient's record and the data provided by his/her bodysensor). [P2] mainly focuses on providing a set of non-functional features for the monitoring of Parkinsonian patients that had not previously been addressed. The authors considered such factors as keeping the cost low (by using sensors onboard a smartphone) and providing flexibility (by making it non-intrusive for the patient). Finally, this solution also deals with the issue of authentication.

The system [P3] proposes the handling of large amounts of information using a distributed, scalable, open-source non-relational database called RIAK. The system that the authors describe can collect about 100 samples per second from sensors. The BodyCloud approach [P4] includes (i) rapid prototyping through a set of web-based programming abstractions that enable the formalization of BSN applications, (ii) extensibility and customizability, making the changes with minimal disruption to the users already using the system, (iii) flexibility, enabling the discovery of new sensors, and (iv) scalability, which relies on *platforms as a service*.

One of the non-functional features that the e-SURAKSHAK [P8] offers is the improvement of system reliability through redundant edge routers. The Patient Monitoring System (PMS) [P9] protects the transmission of sensitive information and prevents access by unauthorized personnel through the security functions of an OSGi framework. The OSGi framework also provides access and reuse for independence between components and supports dynamism and interoperability to promote system scalability.

The proposed ECG Android App [P10] addresses non-functional features such as performance, privacy/security, portability, scalability, flexibility, and cost. The proposed solution is able to monitor the ECG at low cost. The data storage on the SD card of the mobile phone allows improvements in performance and scalability. In addition, the binary file format is optimized for fast and compact analysis. ECG sampling is performed at regular intervals, which provides energy efficiency and prolongs the life of the sensors and the mobile device.

The paper [P13] highlights the robustness of the designed algorithm for monitoring physical activity. The authors highlight the minimal maintenance of the battery and the speed of the Android smartphone and the accelerometer sensor node. The processing module for smart wheelchairs is based on Arduino; it provides low energy consumption, a high sampling rate, and high processing capabilities.

The eHealth platform [P14] provides secure, accurate, and seamless data exchange. In addition, the authors mention that they have demonstrated the capacity of the disease management hub to feed patients when their health goals are not met. Another factor to mention is the emotional support provided to patients through patient–robot dialogues. Furthermore, the platform includes data quality assessment through the following dimensions: accuracy, completeness, consistency, timeline, and usability.

A distinctive feature of VIRTUS [P16] is its reliable, scalable, and secure communication channel, which avoids the loss of data and takes advantage of the characteristics of the publish/subscribe paradigm when the subscriber is off-line.

Health-CPS [P18] includes data format transformation and encryption. Preprocessed data is encrypted so that only authorized devices can decrypt the data.

[7] [online] Available at: http://www.eeml.org/

RQ4: In which case studies were these software architectures evaluated?

Nine of the papers (approx. 42.8%, specifically [P1][P4][P6][P7][P8][P9][P10][P11] and [P15]) evaluate their respective approaches in case studies that focus on monitoring heart rate using electrocardiogram (ECG) sensors. They were selected for two reasons: (i) the case study has become an exemplar in this area, and (ii) the case study has a wide range of potential benefits for patients, because heartbeats are used to diagnose a multitude of diseases such as heart problems, pulmonary pathologies, cardiovascular diseases, and even stress.

Specifically, [P1] briefly describes the elements that make up the case study: (i) an electrocardiogram sensor and a blood pressure oscillometric sensor over the *e-Health Sensor Platform V2.0* (a biometric shield for Arduino available from Cooking Hacks); (ii) a driver developed for *Arduino Uno*; and (iii) the EcoHealth platform, that is, the main component. The driver implements continuous polling of sensors regarding the measured heart rate and blood pressure, and then collects and sends the data measured by these sensors to the EcoHealth platform for historical data storage. Finally, the use case defines the possibility of defining triggers associated with these feeds in order to send alert messages to doctors when the measured heart rate is greater than 160 beats per minute (bpm) and/or the measured blood pressure is greater than 160/100 mmHg. This case study is useful for characterizing the platform, but empirical data regarding the performance, scalability, reliability, or availability of the EcoHealth platform are not provided.

[P6][P7] and [P8] also monitored blood pressure, since asymptomatic hypertension usually goes unnoticed, causing serious diseases such as heart disease and stroke, and it can lead to heart attacks even for hypertensive people. Monitoring allows prevention through medical diagnosis, generating timely responses and therefore effective treatment, since the data obtained can be shown as a graph that highlights conditions considered as serious. Response time is key in these systems. [P4] carried out a performance evaluation by emulating a set of clients that send sensor data streams simultaneously. The evaluation consisted in a sensor, which generated 6000 data values a minute, and 10 clients, each one making 10 requests with a 60-second interval between each one, each one sending a dataset with 6000 samples. Five test were run with 10, 20, 30, 40 and 50 clients. The tests show that the response time increases linearly between 10 and 40 clients but the increase is not linear when there are 50 clients or more—the authors identified a bottleneck at 50 clients. Finally, [P9] defined a case study for emergency calls in cases of cardiac arrest.

The proposal [P15] includes the monitoring of eight parameters (ECG, blood pressure, airflow, oxygen level, temperature, transpiration, muscle activity, and body position) to identify the activities performed by patients, such as sitting, standing, walking, running, and an '*undefined'* activity. The experiments demonstrate that automatic recognition of activities is possible through wearable sensors, but the success rate depends on the position of the sensor on the body and the number of training sessions of the neural network capable of identifying the activity. The authors mention that the inclusion of the 'undefined' activity category significantly increased the recognition of the other categories.

[P2] proposes monitoring motor disorders applied to Parkinsonian patients suffering from *freezing of gait*. Patients experience episodes where they cannot walk, and this results in injuries caused by falls. The smartphone app showed a high performance in real-time FoG detection (with a mean sensitivity of 93.08% and a mean specificity of 90.98%) due to the fuzzy logic-based detection algorithm. The tests were performed on 6 Parkinsonian patients with a total of 81 FoG episodes. It is important to take into account the complexity of the identification of FoG episodes: because the episodes are not constant, the patients do not know what the paralysis really is.

[P9] and [P13] executed performance tests. In the case of the Patient Monitoring System (PMS) [P9], the authors used the VisualVM profiling tool to record the performance of the system and the Windows Task Manager tool to corroborate the results. The authors mention that after approximately 4 minutes of execution of the PSM, the CPU usage increased by a significantly low percentage (7%) and the live process count rose by only 6%. The amount of memory used was also low, at approximately 18 MB. Therefore, they demonstrated a low consumption of resources.

In the proposal [P13], tests were performed on three commercial smartphones and one tablet (a Samsung Galaxy Next Turbo, a Samsung Galaxy Tab 2, a Samsung Galaxy S4, and a Xiaomi MI2S, which we will refer to as devices 1, 2, 3, and 4, respectively). Regarding *CPU load*, device 4 suffered a consistent CPU load, device 1 showed a high overload, and there was no significant overload with the other devices. Another factor evaluated was *processing time*. The authors mention that device 3 gave poor results in comparison to the others but was faster in executing the classifier algorithm after being updated to Android 5. No devices other than 1 used computation drifts. With regard to the *RAM*, a maximum allocation of 50 MB was not considered as critical parameter. A relevant factor not evaluated in the proposal [P9] is the *discharge of the battery*. All the devices presented similar behaviors, showing the implementation of low power. Finally, the authors evaluated the *transmission time* required to stream activity data to our cloud platform. They invoked the data transmission function once per second and used a stable Wi-Fi connection. The results indicated that high-performance devices are not required for proper cloud transmission.

Finally, it is necessary to highlight that [P16] to [P21] do not present any case studies.

5 Key Findings

This SLR has retrieved a set of existing solutions for healthcare CPS/IoT. These solutions provide users and businesses with a wide range of smart services and features for health and emergency applications and also for sports and wellness. From this SLR, we have identified the following set of services: patient information management; real-time analysis of data collected from bodysensors attached to patients and real-time feedback to users; complex analysis and detection of problems or anomalies in the data measured in real time; notifications for reporting those problems or anomalies; assistance for the decision-making process in determining a diagnosis and associated treatment; online consultations; and integration with external systems for medical records management and administration of hospital activity.

Healthcare CPS are generally **three-tier architectures**, as shown by ten primary studies ([P1][P2][P3][P5][P6][P8][P10][P14][P15] and [P19]) and multi-tier architectures ([P4]), which represents approx. 52.4%. These systems are composed of wearable wireless networks of bodysensors, sometimes also actuators (the ***perception tier***); sensors and actuators communicate with components, commonly referred to as gateways, that support the heterogeneity of protocols of these devices (the ***gateway tier***); and these gateways subsequently communicate through the Internet with remote (telemedicine/medical) servers (the ***application tier***). Other primary studies focus on a layered architecture description ([P7][P9][P11][P16][P17][P18][P21], which represents 33.3%) and others do not specify layers and tiers. It is necessary to highlight the difference between the terms *layer* and *tier*, although they are frequently used interchangeably.[8] Both terms partition the concerns of an application into stacked groups or segments, but a tier is located on a physically separate computer. It is clear that perception, gateway, and application tiers are physically distributed. In the same way, it is possible to identify, as [P4] does, different layers inside a tier—for example, the presentation, security, and persistence layers. As each tier is independent of the other tiers, this architectural style makes these systems easier to maintain and scale out. As layers enable clearly defined functional segments, this architectural style improves cohesion, loose coupling, and reusability. Most of the papers highlight these quality attributes, but they lack evidence to support these general assumptions. We can therefore conclude from the analysis of the primary studies that both architectural styles are extensively used, but further evaluation is desirable.

One of the limitations of the solutions under review was the fact that the existing biosensors, and medical wearable sensing devices in general, work only with the software provided by the device supplier. This limitation was mainly emphasized by [P15], which mentions that "*any attempt to re-use and integrate such devices in different applications fails because communication protocols are not openly available, or the device interfaces are proprietary solutions*". The lack of standardization is a well-known problem that most governments include in their research agendas. Other domains, such as automotive and energy grids, have achieved standardization in two respects: (i) limiting the number of industrial network protocols, and (ii) adapting Ethernet interfaces to ensure compatibility between devices and upper tiers. Although such an approach is possible from a hardware point of view, it is difficult to implement in software due to the limited memory capacity of microcontrollers [P15]. For example, in healthcare CPS, where most biosensors are wearables with restrictive processing and memory capacity, it is quite difficult to implement complex data acquisition procedures together with HTTP support. In those healthcare CPS where it is possible, Arduino boards can transform microcontrollers into complex monitoring systems, as [P15] demonstrates in an experiment.

Approximately 38% of primary studies—[P1][P2][P3][P4][P5][P16] and [P19], and also [P15] in another experiment—deal with this problem with gateways and middleware that have processing and communication functionalities. Specifically, [P15] shows an experiment using a device together with a gateway for data acquisitions of health-related parameters, data processing, and data transmission that were developed with an open architecture. In this case, the device-gateway protocol is a message-based protocol in which messages are translated into HTTP requests using a dedicated XML scheme, conforming to an ontology for cardiovascular information. In [P2] and [P3], health monitoring devices use Bluetooth to communicate with smartphones, so smartphones act as gateways that send the information to upper tiers. Efforts were also made to build advanced middleware platforms. In this regard, it is remarkable that some of the primary studies of this SLR call their solutions "*middleware platforms*." This is the case for EcoHealth [P1] and KASO [P5], among others. The motivation is to emphasize their role as abstraction layers that hide the details of WSANs from the upper applications layers to which these middleware provide services. The middleware aims to deal seamlessly with a myriad of heterogeneous sensors and smart devices (e.g., concentrators and sinks) from different manufacturers, each one providing a different interface, thus creating operational barriers in order to use them in a holistic way [P1]. Therefore, the proposed middleware—according to the primary studies that we reviewed—aims to enable (i) interoperability among heterogeneous sensors and smart devices to communicate with Internet services and users, and (ii) integration with other systems (e.g., medical applications). From this point of view, these solutions focus on the middle tier (the gateway tier) but also offer advanced features for data processing, analysis, and visualization.

Finally, the tier/layer that BodyCloud [P4] and RPIME [P7] define to provide developers with programming abstractions is unusual. The former provides developers with programming abstractions for rapid prototyping of BSN applications whereas the latter specializes in implementing P-RESTful (Pervasive REST) applications. The programing abstraction allows developers, mainly data analysts, to formalize BSN data sources, input data, actions to be realized on the input data (e.g., storage, processing [P18], analysis), and output data. In this way, they can program customized analysis workflows over the data.

[8] [online] Available at: https://msdn.microsoft.com/en-us/library/ee658117.aspx

In addition to the tier/layer decomposition, it is important to analyze how **communication** between architectural components is realized. Most approaches are based on the following architectural styles (see Fig. 6): service-oriented architecture [P8][P9][P16] and REST architectures [P1][P4][P5][P7][P13][P17], which represents approx. 43%—, and message bus or event-driven architectures (e.g., the publish/subscribe pattern [P17]) over cloud infrastructures [P1][P3][P4][P5][P10][P13][P18][P19], which represents approx. 38%. In fact, as healthcare CPS/IoT demands more processing and storing capabilities, there is a trend toward using cloud infrastructures [P1]. This work synthesizes the benefits of the cloud as follows: scalability, availability, performance, and on-demand resource usage. Cloud computing fits well as an enabling technology for healthcare CPS/IoT, as it presents a flexible stack of computing, storage, and software services under a pay-per-use model. Software as a Service (SaaS) can provide a high-quality experience for patients, physicians, and other caregivers anytime, anywhere, and seamlessly. Papers [P3][P4][P5][P10][P13] present cloud-based solutions mainly to deal with *scalability* issues (see Fig. 8); however, the massive amounts of data that can be collected from bodysensor networks could result in *latency* in networks. To deal with these issues, other approaches have emerged, such as *fog computing*. Fog computing could minimize latency in networks by analyzing data close to the devices that collected the data before these data are sent to the cloud— in other words, by means of data pre-processing or filtering. [P13] does not use the term fog computing, but the authors propose a similar approach by defining different modes of operation—full cloud, mix cloud, and full local. In this regard, we identify a need for more research into which biosignals could be pre-processed close to patients and how much latency can be reduced, which could be of great value to the eHealth community. The authors of [P13] demonstrate the *effectiveness* of their cloud-based framework for monitoring human activity as well as for *performance* evaluation. However, most of the primary papers lack evidence that supports these general assumptions.

In addition to cloud storage, the management of large amounts of data collected from WSN, or more specifically from BSN, requires new approaches, such as *big data*. The term big data is sometimes used to refer to the storage of large amounts of non-structured data on NoSQL databases, as in [P3]. That work makes reference to big data, but only reports an experience using the NoSQL RIAK database. However, big data entails much more. Big data is storage but also analytics, such as real-time stream analytics and predictive analysis through machine learning algorithms. Additionally, it is necessary to deal with the four dimensions of big data: *volume*, *velocity*, *variety*, and *veracity*. In healthcare CPS/IoT, veracity and accuracy of readings and analysis of vital signs are especially important for correct assessment and treatment of patients. The work [P2] is the only one that evaluates the accuracy and effectiveness of the algorithm it defines.

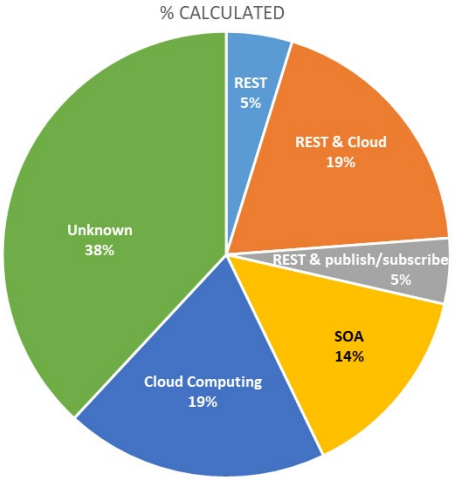

Fig. 6 Use of architectural styles

In the field of healthcare, and where large-scale analytics are concerned—for example, discovering patterns in diseases—two key challenges are how to preserve *privacy* and *security*. Papers [P4][P5][P10][P13] include in their solutions specific layers for security (confidentiality and authenticity), data integrity, and data encryption (see Fig. 8). In fact, security issues are identified as a key challenge by most of the papers (notably [P16] and [P18]).

Finally, two critical features of the architecture are the runtime discovery of new devices into an IoT environment (for example, a new sensor) and the dynamicity of healthcare CPS/IoT for being reconfigured at runtime according to changes in the environment (for example, self-optimization and self-healing). The first issue has been addressed by [P4][P5][P11] and [P15], and the second issue has been addressed by [P1][P5] (see Fig. 8). In the latter case, runtime reconfiguration is limited to dynamic and efficient allocation of resources (aka self-optimization).

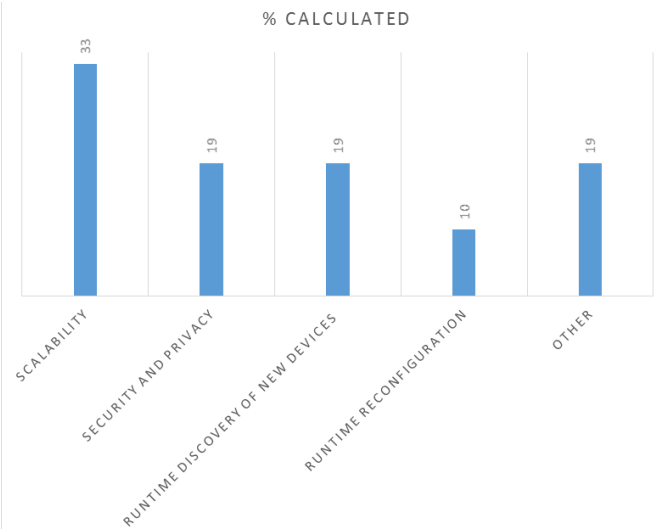

Fig. 7 Architecturally significant requirements

As a result of completing the SLR, we have obtained the architectural knowledge to address building a reference architecture for healthcare CPS/IoT. Reference architectures capture the high-level architectural design for many software applications. As such, they describe common architectural elements that can be used in the design of a concrete architecture of an application domain [53]. Fig. 8 shows an initial proposal for a reference architecture for healthcare CPS from the primary studies under review. This architecture is four-tier architecture: perception tier, middleware/gateway tier, application tier, and programming abstraction tier. Following a bottom-up description, the figure shows a gateway that supports several communication protocols—specifically, body area networks (BAN) and wireless personal area network (WPAN)—as well as an advanced middleware platform. This middleware is composed by a set of mod-ulesthat provide: REST device connection, device dynamic discovery, historical and real-time data access, expert system—e.g. engines for big data processing and machine learning—, data handler adapters, and cloud file storage and parallel computing. A service layer uses these modules to provide users high-level services, such as real-time monitoring, alarm & actuation, decision support, and reporting, possibly interacting with other systems, such as medical servers. By last, the programming abstraction tier provides data analysts with the capacity of customizing data analysis workflows.

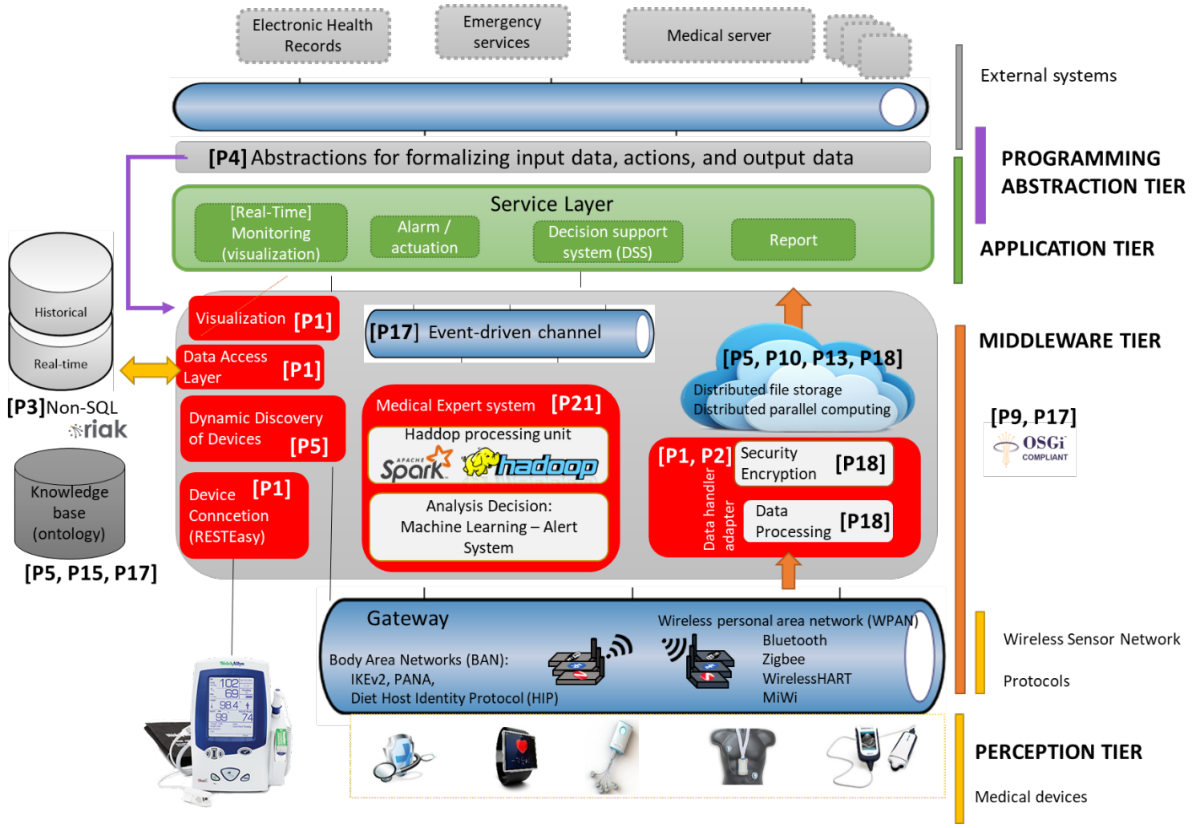

Fig. 8 Reference architecture for healthcare CPS

This section ends with the Table 5 that shows the projects that provided financial support for the primary studies, as detailed in the acknowledgements sections and in other references in the papers. We observed that most of these publications were not the results of large projects focused on applying CPS/IoT to the healthcare domain. We assumed that this was due to the novelty of this research area, and we confirmed this assumption by searching for large projects on healthcare CPS/IoT within the European Union's Horizon 2020 Programme. We found a large number of projects starting in 2017 and ending in 2020 (such as Internet of Food and Farm 2020 and ACTIVAGE: ACTivating InnoVative IoT smart living environments for AGEing well). Therefore, after 2020, there will be a higher number of papers in this area, making this SLR a state-of-the-art starting point that could be updated in the coming years with the results of these new projects.

Table 5. Projects that supported the primary studies from the financial support

ID		Project
P3	National (SP)	*Virtual Cloud Carer Project*
		iPHealth
P5	FP6 European Programme	USWN: *Solving Major Problems in MicroSensorial Wireless Networks*
	ITEA	DiYSE: *Do-it-Yourself Smart Experiences*
P6	National (US)	Alliance of Chicago Health Services
P7	FP7 European Programme	SMSCOM: *Self-Managing Situated Computing*
P13	Horizon 2020 European Programme	*INTER-IoT*
	National (IT)	*Smart Personal Mobility Systems for Human Disabilities in Future Smart Cities*
	National (IT)	*Locubirehab: Low cost ubiquitous rehab assistant*
P15	European	*FIRST – Future Internet Research Services and Technologies*
P17	National (SP)	-
P19	FP7 European Programme	*eWall: eWall for Active Long Living* WALL for Active Long Living – eWALL
P21	National (unknown)	-

6 Discussion

The reference architecture for healthcare CPS/IoT proposed in Fig. 8 provides a holistic view of the different approaches that were analyzed in the SLR. It provides practitioners with an integrated set of software artefacts (e.g. components, technologies) that could be reused for building new healthcare CPS/IoT. It also provides high-level architectural design decisions, such as the partition of the system into tiers and layers. The high-level architectural design is a multi-tier architecture. Specifically, this architecture is four-tier including, in addition to *perception*, *middleware* and *application* tiers, rapid prototyping and customization of BSN applications and data analysis workflows through an *abstraction programming* tier.

To guarantee the interoperability between heterogeneous devices from different manufacturers that compose the perception tier, the reference architecture enforces to have a gateway that hides the details of WSANs—specifications and protocols—to upper layers. The middleware tier also offers advanced services, such as, real-time data acquisition, big data real-time processing, machine learning, (distributed) parallel computing, and (distributed) storage, to upper application tiers that make use of them. Security encryption is a key component for real-time data acquisition, whereas expert systems are supported by using machine learning techniques, supported in turn by non-SQL databases for big data requirements of storage and processing. Several of these services could be distributed in cloud infrastructures. This decision involves legal and compliance issues and implies loss of control when moving to the cloud. As proposed by several primary studies, several advanced services of the middleware tier could be implemented following a service-oriented architecture style through an OSGi-compliant technology. Being OSGi-compliant implies high modularization through self-describing components called bundles that communicate each other through OSGi services in a container that manages their life-cycle. This facilitates deploying new bundles at runtime, i.e. dynamism, as well as interoperability and scalability.

All these advanced services are used by uppers applications that range from patient information management, diagnosis and treatment assistance, and real-time feedback to patients, to integration with external systems for medical records management and administration of hospital activity.

As conclusion, this architecture proposes a service-oriented architecture, built on a multi-tier infrastructure, and integrated with distributed cloud services. When this architecture is going to be instantiated, architects should consider trade-offs between using an OSGi-compliant container vs. a conventional service-oriented bus, a centralized middleware architecture vs. a distributed architecture, on-premise vs. cloud, etc. Therefore, the proposed reference architecture should be validated in several scenarios and case studies, especially

to analyze and compare these critical trade-offs, evaluating its strengths and weaknesses in terms of interoperability, scalability, availability, and performance, among others. From this empirical analysis, it would be feasible to provide a set of recommendation guidelines about an implementation of this reference architecture based on the quality attributes.

Finally, we have identified two issues that the primary studies have not addressed and that a reference architecture for healthcare CPS/IoT should take into account:

- The first one is the *multi-tenancy* as key technology to leveraging the economics of scale for [54]. Native multi-tenancy supports all tenants by a single shared application instance over various hosting resources [54]. An aspect to evaluate are pros/cons of multi-tenancy approach in eHealth.
- The second one is *microservices* and *nanoservices*. Microservices is an emerging architectural style for the development of distributed systems that intends to deal with high availability, scalability, modifiability, and agility. *"A microservices application is decomposed into independent components called microservices, that work in concert to deliver the application's overall functionality*" [21]. This is known as componentization via services [22]. The principle of the microservices architecture is akin to the Unix principle: *Do one thing and do it well* [22]. Each microservice has well-defined contracts (typically RESTful) for other microservices to communicate and share data with it. Nanoservices (aka. *serverless*) takes to the extreme the concept of microservices by focusing on Functions as a Service (FaaS). Cloud providers such as AWS Lambda, Google's Cloud Functions and Azure's Functions offers a variety of programming languages and tools to support these architecture styles.

Both multi-tenancy and microservices have demonstrated their benefits in other domains [55]. The use of big data in healthcare is, along with security, one of the most challenging issues. This is partly because it necessarily involves a form of knowledge that is very specialized, namely data science, which results, as Fig. 9 shows, from the union of knowledge in a particular domain of application (mathematics and computer science). Within computer science, it is necessary to pay special attention to software architectures, especially to new approaches such as *lambda architectures*, which are data-processing architectures composed of three layers: batch processing, speed (real-time) processing, and a serving layer that responds to queries from the outputs given by the batch and speed layers. There is a myriad of technologies around big data. In other domains, there are examples of lambda architectures for analytics of IoT data with technologies such as Apache Spark, Cassandara, Kafka, and Akka. Therefore, the application of big data to healthcare CPS presents numerous issues to be resolved.

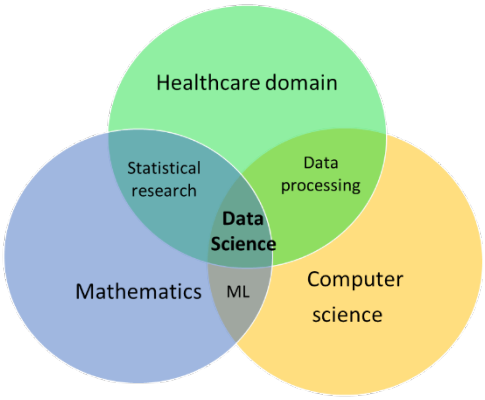

Fig. 9 Big data Venn diagram [source: Palmer, Shelly. Data Science for the C-Suite. New York: Digital Living Press; 2015]

Regarding the data that are managed in healthcare CPS, it is also necessary to keep in mind the requirements of data interoperability. From our knowledge in other CPS/IoT domains—e.g., smart grids [56]—we know of the existence of specific layers for translation between data formats to guarantee the interoperability and communication of devices based on different electrical standards. In the same way, American and European electronic medical records show considerable differences, and none of the primary studies addressed this interoperability issue.

In general, these gaps identified in the section have still not reached a mature development state. This requires additional research efforts.

7 Threats to Validity

This section addresses potential biases and the actions taken to minimize their effects. Trico et al. [57] define three kinds of biases: identifying articles, selecting studies, and obtaining accurate data. This work addresses these issues in the following ways: several factors can affect the identification of articles, such as, the criteria of the reviewers and editors of journals and conferences, industry-sponsored research in some areas, the place of publication, biased indexing studies in literature databases, inadequate or incomplete searches, articles that are cited more often than others are, and studies that generate multiple publications. In this SLR, we used different electronic literature databases to include the maximum number of sources and to minimize the impact of the above-mentioned biases.

For selecting studies, the inclusion and exclusion of studies was completed by two of the authors of the present study, and those with discrepancies were resolved by taking a decision among all the authors. Finally, to improve data accuracy, we selected papers that were published in journals, conference proceedings, or workshop proceedings according to a peer-review process. Additionally, the extraction and analysis of the papers were performed by the three authors to constrat the analysis conclusions and avoid personal bias.

8 Conclusions and Future Work

This paper presents the results of a systematic literature review on software architectures for Healthcare CPS. The results show the proposals of different architectures, most of them, 3-tier architectures: the perception tier, the gateway/middleware tier, and the application tier. These architectures focus on supporting important features in any domain but especially relevant to healthcare. These features are interoperability to support the heterogeneity of body sensor networks, automatic discovery of new devices in this networks, dynamic reconfiguration according to changes in systems' environments, security and privacy, scalability of storage and processing, and big data analytics. Quality attributes also play a fundamental role, such as robustness, reliability, availability and performance. Cloud computing, SOA, and REST architectures are the most widely used architecture styles.

Most of the architectures have been evaluated on real or simulated case studies. However, we observe a lack of metrics that provide evidence of the goodness of the proposed architecture' solutions.

As future work, we plan working on offering smart health services on top of the already existing middlewares, to try to cover the gaps that this SLR has identified. These new services can be integrated adopting new architecture styles, such as microservices and serverless, and should cover the analysis of large volume of data.

Acknowledgment

This work is supported by the Spanish fund MESC (DPI2013-47450-C2-2-R) and CROWDSAVING (TIN2016-79726-C2-1-R), H2020 CPSELabs (644400), and in part by the Salesian Polytechnic University (http://www.ups.edu.ec/).

9 References

[1] WHO global report. Preventing chronic diseases: a vital investment, World Health Organization (2005)

[2] Lee EA. The past, present and future of cyber-physical systems: A focus on models. *Sensors* 2015; **15**(3):4837-4869.

[3] Kramp T, van Kranenburg R, Lange S. Introduction to the Internet of Things. *Enabling Things to Talk* 2013, Springer Berlin Heidelberg; 1-10. DOI:10.1007/978-3-642-40403-0_1.

[4] Lee EA. Cyber Physical Systems: Design Challenges. *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*, *isorc 2008*, IEEE, 2008; 363-369.

[5] Carroll N. Key success factors for smart and connected health software solutions. *Computer* 2016; **49**(11):22-28. DOI:10.1109/MC.2016.340.

[6] Dybå T, Dingsøyr T. Empirical studies of agile software development: A systematic review. *Information and software technology* 2008; **50**(9):833–859.

[7] Bowers J, May J, Melander E, Baarman M, Ayoob A. Tailoring XP for large system mission critical software development. *Extreme Programming and Agile Methods XP/Agile Universe* 2002, Springer; 100–111.

[8] Kitchenham B. Procedures for performing systematic reviews. *Keele, UK*, *Keele University* 2004; **33**(2004):1-26.

[9] Yick J, Mukherjee B, Ghosal D. Wireless sensor network survey. *Computer Networks* 2008; **52**(12):2292-2330.

[10] Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks* 2012; **10**(7):1497-1516.

[11] Richardson I. *Connected health: people, technology and* processes, *Lero-TR-2015-03.* Lero Technical Report Series, University of Limerick, 2015.

[12] Haque SA, Aziz, SM, Rahman M. Review of cyber-physical system in healthcare. *International journal of distributed sensor networks* 2014; **10**(4):217415. DOI:10.1155/2014/217415

[13] Perry DE, Wolf AL. Foundations for the study of software architecture. *ACM SIGSOFT Software engineering notes* 1992; **17**(4):40–52.

[14] Garlan D. Software architecture. *Wiley Encyclopedia of Computer Science and Engineering* 2001.

[15] Brown A, McDermid J. The art and science of software architecture. *Proceedings of First European Conference on Software Architecture*, ECSA'07, Springer-Verlag Berlin Heidelberg 2007; volume 4758 of Lecture Notes in Computer Science:237-256:

[16] Kruchten P, Obbink H,Stafford J. The past, present, and future for software architecture. *IEEE Software* 2006; **23**(2):22-30.

[17] IEEE Std 1471-2000 (2007). ISO/IEC Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems. ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15, (pp. c1-24).
[18] Kruchten PB. The 4+1 view model of architecture. *IEEE Software* 1995; **12**(6):42-50.
[19] Chapter 3: Architectural Patterns and Styles. https://msdn.microsoft.com/en-us/library/ee658117.aspx
[20] Erl T. SOA Design Patterns (1st ed.), Prentice Hall PTR, Upper Saddle River, NJ, USA, 2009
[21] Russinovich M. Microservices: An application revolution powered by the cloud, 2016 https://azure.microsoft.com/es-es/blog/microservices-an-application-revolution-powered-by-the-cloud/
[22] Lewis J, Fowler M.. Microservices, 2014 http://martinfowler.com/articles/microservices.html
[23] Feng X, Shen J, Fan Y. REST: An alternative to RPC for web services architecture. *First International Conference on Future Information Networks*, ICFIN Oct 2009; 7-10.
[24] Botta A, De Donato W, Persico V, Pescapé A. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems* 2016; volume 56:684–700. DOI=http://dx.doi.org/10.1016/j.future.2015.09.021
[25] Cai H, Cui L., Shi, Y, Kong L, Yan, Z. Multi-tenant service composition based on granularity computing. *2014 IEEE International Conference on Services Computing,* SCC 2014; 669-676.
[26] Dybå T, Dingsøyr T. Strength of evidence in systematic reviews in software engineering. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement,* ESEM '08, 2008; 178-187.
[27] Keele S. Guidelines for performing systematic literature reviews in software engineering. *Technical report*, Ver. 2.3 EBSE Technical Report. EBSE, 2007
[28] Maia P, Batista T, Cavalcante E, Baffa A, Delicato FC, Pire PF, Zomaya A. A Web Platform for Interconnecting Body Sensors and Improving Health Care. *Procedia Computer Science* 2014; volume 40:135-142. DOI: http://dx.doi.org/10.1016/j.procs.2014.10.041.
[29] Pepa L, Capecci M, Verdini F, Ceravolo MG, Spalazzi L. An architecture to manage motor disorders in Parkinson's disease. *2nd World Forum on Internet of Things (WF-IoT),* Milan, 2015 IEEE; 615-620. DOI: 10.1109/WF-IoT.2015.7389124
[30] Páez DG, Aparicio F, de Buenaga F, Ascanio J. Big Data and IoT for Chronic Patients Monitoring. *International Conference on Ubiquitous Computing and Ambient Intelligence* 2014; volume 8867:416–423. DOI:10.1007/978-3-319-13102-3_68.
[31] Fortino G, Parisi D, Pirrone V, Di Fatta G. BodyCloud: A SaaS approach for community Body Sensor Networks. *Future Generation Computer Systems* 2014; volume 35:62-79. DOI:http://dx.doi.org/10.1016/j.future.2013.12.015.
[32] Corredor I, Martínez JF, Familiar MS. Bringing pervasive embedded networks to the service cloud: A lightweight middleware approach. *Journal of Systems Architecture* 2011; **57**(10):916-933, ISSN 1383-7621. DOI:http://dx.doi.org/10.1016/j.sysarc.2011.04.005.
[33] Méndez EO, Ren S. Design of cyber-physical interface for automated vital signs reading in electronic medical records systems. *2012 IEEE International Conference on Electro/Information Technology*, Indianapolis, IN, 2012; 1-10. DOI: 10.1109/EIT.2012.6220696.
[34] Caporuscio M, Ghezzi C. Engineering future internet applications: The Prime approach. *Journal of Systems and Software* 2015; volume 106:9-27, ISSN 0164-1212. DOI:http://dx.doi.org/10.1016/j.jss.2015.03.102.
[35] Rao IH, Amir NA, Dagale H, Kuri J. e-SURAKSHAK: A Cyber-Physical Healthcare System with Service Oriented Architecture. *2012 International Symposium on Electronic System Design*, ISED 2012, IEEE; 177–182. DOI:10.1109/ISED.2012.66
[36] Ragavan SV, Haider K, Shanmugavel M. Healthcare Telematics Service Implementation Using OSGi Framework. *Procedia Computer Science* 2014; volume 42:168-174, ISSN 1877-0509. DOI:http://dx.doi.org/10.1016/j.procs.2014.11.048.
[37] Mohammed J, Lung CH, Ocneanu A, Thakral A, Jones C, Adler A. Internet of Things: Remote Patient Monitoring Using Web Services and Cloud Computing. *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, Taipei, 2014; 256-263. DOI:10.1109/iThings.2014.45
[38] Rodrigues A, Silva JS, Boavida F. iSenior—A Support System for Elderly Citizens. *IEEE Transactions on Emerging Topics in Computing* 2013; **1**(2):207-217. DOI:10.1109/TETC.2013.2294917
[39] Hu X, Wang L, Gao P, Dong X, Ji Z, You F, et al. Study on Disease Screening and Monitoring System Based on Wireless Communication and IOT, *2012 Spring Congress on Engineering and Technology*, Xian, 2012; 1-6. DOI:10.1109/SCET.2012.6341961
[40] Gravina R, Ma C, Pace P, Aloi G, Russo W, Li W, Fortino G. Cloud-based Activity-aaService cyber–physical framework for human activity monitoring in mobility. *Future Generation Computer Systems* 2017; volume 75. DOI:http://dx.doi.org/10.1016/j.future.2016.09.006.
[41] Al-Taee Ma, Al-Nuaimy W, Muhsin ZJ, Al-Ataby A. Robot Assistant in Management of Diabetes in Children Based on the Internet of Things. *IEEE Internet of Things Journal* 2017; 4(2):437-445. DOI:10.1109/JIOT.2016.2623767
[42] Sebestyen G, Hangan, A, Oniga S, Gal Z. eHealth solutions in the context of Internet of Things. *2014 IEEE International Conference on Automation, Quality and Testing, Robotics* 2014*;* 1-6. DOI:10.1109/AQTR.2014.6857876
[43] Bazzani M, Conzon D, Scalera A, Spirito MA, Trainito CI. Enabling the IoT paradigm in e-health solutions through the VIRTUS middleware*. 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom* 2012*;* 1954-1959. DOI:10.1109/TrustCom.2012.144
[44] Corredor I, Metola E, Bernardos AM, Tarrío P, Casar JR. A lightweight Web of Things open platform to facilitate context data management and personalized healthcare services creation. *International journal of environmental research and public health* 2014; **11**(5):4676-4713.
[45] Zhang Y, Qiu M, Tsai CW, Hassan MM., Alamri A. Health-CPS: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal* 2017*,* 11(1):88-95. DOI:10.1109/JSYST.2015.2460747

[46] Suciu G, Suciu V, Martian A, Craciunescu R, Vulpe A, Marcu I, et al. Big data, internet of things and cloud convergence–an architecture for secure e-health applications. *Journal of medical systems* 2015; **39**(11):141.
[47] Blazona B, Koncar M. HL7 and DICOM based integration of radiology departments with healthcare enterprise information systems. *International journal of medical informatics* 2007; volume 76:S425-S432. DOI:https://doi.org/10.1016/j.ijmedinf.2007.05.001
[48] Rathore MM, Ahmad A, Paul A, Wan J, Zhang D. Real-time Medical Emergency Response System: Exploiting IoT and Big Data for Public Health. *Journal of medical systems* 2016; **40**(12):283.
[49] World Health Organization, Global age-friendly cities: a guide, 2007.
[50] Penagos SP, Salazar LD, Vera F, Control de signos vitales.
[51] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: An update. *ACM SIGKDD explorations newsletter* 2009; **11**(1):10-18. DOI:http://doi.acm.org/10.1145/1656274.1656278
[52] Hardt D. The OAuth 2.0 Authorization Framework, IETF RFC 6749, October 2012
[53] Eklund U, Eriksson A, Bosch J. A Reference Architecture Template for Software-Intensive Embedded Systems. *Proceedings of the WICSA/ECSA Companion Volume* 2012; 104–111. DOI:10.1145/2361999.2362022
[54] Guo CJ, Sun W, Huang Y, Wang ZH, Gao B. A Framework for Native Multi-Tenancy Application Development and Management. *The 9th IEEE Int. Conf. on E-Commerce Technology and The 4th IEEE Int. Conf. on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*, Tokyo, 2007; 551-558. DOI:10.1109/CEC-EEE.2007.4
[55] Díaz J, Pérez J, Pérez J, Garbajosa J. Conceptualizing a framework for cyber-physical systems of systems development and deployment. *Proccedings of the 10th European Conference on Software Architecture Workshops*, (ECSAW '16). ACM, New York, NY, USA, Article 1 , 7 pages. DOI:https://doi.org/10.1145/2993412.3004852
[56] Pérez J, Díaz J, Vidal C, Rodriguez D, Fernández D. Self-Balancing Distributed Energy in Power Grids: an Architecture based on Autonomic Computing. *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS'14)* 2014, IEEE Computer Society; 2398- 2407.
[57] Tricco AC, Tetzlaff J, Sampson M, Fergusson D, Cogo E, Horsley T, Moher D. Few systematic reviews exist documenting the extent of bias: a systematic review. *Journal of Clinical Epidemiology* 2008; **61**(5):422–434.