

# An incomplete and biased survey on selected resource management for distributed applications as basis for interoperability of medical devices

Imad Eddine Touahria<sup>1,2</sup>

<sup>1</sup> Department of Telematics engineering, Universidad Carlos III de Madrid, Leganes, Madrid, Spain

<sup>2</sup> Department of computer science, Ferhat Abbas Setif-1 University, Setif, Algeria  
100370038@alumnos.uc3m.es imad.touahria@univ-setif.dz

**Abstract.** The popularity and wide spread of IoT technology has brought about a rich hardware infrastructure over which it is possible to run powerful applications that were not previously imagined. Among this infrastructure, there are the medical hardware that is progressively advancing but at a slower pace. Nevertheless, medical devices are more powerful now to run more sophisticated functions and applications and exchange big data with external systems in a secure and safe fashion. Towards the design of an architecture for interoperability of medical devices, this paper initially focuses on the background work that is taken by the author for this objective. The paper briefly describes the role of the software in the advances of medical systems and their possibilities for interoperability. It focuses attention on the distribution software layer that is responsible for connectivity, efficiency, and time-sensitivity in the basic operation of medical systems such as exchange of information and commands across devices and systems. The paper analyses a number of previous work on middleware (mostly performed at his research group and also in a broader research community), and pay especial attention to the middleware for web-based systems and how they relate to the development of distributed medical systems.

**Keywords:** Middleware · Medical devices · Distributed medical applications · Interoperability · Integrated Clinical Environment · Medical standards.

## 1 Introduction

As more IoT devices populate everyday life and penetrate on every system and society, the available hardware infrastructures become progressively richer. The new application paradigms such as *Social Dispersed Computing* introduced in [62] envision the creation of highly complex systems that will provide powerful and intelligent solutions to many of today's problems such as mobility, transport, energy, health, smart living, etc.

The amount of patients in need of continuous monitoring by 2025 will be 1.2 billion [91] approximately. This indicates the need for providing more sophisticated logic to build intelligent systems with very efficient operation; for this it is needed to find more efficient ways for designing and developing medical devices and their corresponding software that enables continuous monitoring through smart devices (mobile or bedside) that will improve the quality of care and safety.

The integration of newer hardware and software technology in health care systems is an opportunity for decreasing the levels of mortality and will provide, overall, an improved and smarter patient care.

Health monitoring solutions can be provided on many available devices, from a small medical application deployed on a PDA (Personal Digital Assistant) to measure heart rate to complex systems deployed on a hospital server that ensures monitoring of a patient with critical or chronic diseases and clinician decision support on diagnosis. Medical devices are an example of the new era of smart health care, as these devices offer new possibilities for physicians (on the user side) and provide new opportunities for research on the design and development of health care software that is data-centric and provides efficient interoperability.

Critical clinical scenarios are deeply in need of the installation of medical devices especially in cases where patients are located in remote sites and they are in need of diagnosis services in a remote way. The systems that will support these scenarios will have to integrate devices that will be mobile [23, 84, 94] or fixed close to the monitored patient such as the bed or somewhere around her/his home or living site [81]. Manufacturers provide numerous alternatives for hardware and software of medical systems with a high degree of variability. These differences may generate some integration problems to build holistic clinical solutions. The network of the care center (i.e., hospital, etc.) has different computational servers, switches, data recorders and medical devices/equipment; the latter ones have to be integrated in the network as Plug and Play (PnP) devices. Manufacturers have to adhere to standards for medical systems (operation, data exchange, etc.) in order to achieve integrability in the network.

Improving the assistance given to patients depends greatly on the capacity of medical systems to collect large data amounts in real-time, exchange them, process them, and create new knowledge that could assist the medical decisions. Collecting data from multiple sensors and devices can only be done if *interoperability* is achieved effectively, efficiently, and in a timely manner.

Interoperability is fundamental in the medical domain. At the same time, it is a great challenge given the enormous variety of actors ranging from medical devices to computational units, to human actors. As defined in [35], *medical device interoperability* is the ability to safely, securely, and effectively exchange and use information among one or more devices, products, technologies, or systems. The data that is exchanged can be used for a number of purposes such as display patient information monitoring, storage, interpretation, analysis, and take autonomous action on or control another product. OR.NET [7] and ICE [1] are

two examples of projects whose objective is to provide a safe and interoperable network of medical devices. This will be later described in this paper.

For safety reasons, medical systems have traditionally been deployed in isolation. However, the proliferation of networking technology and distribution software and middleware has also reached eHealth in such a way that medical systems are becoming more open. To support such interconnection in a *safe* way, the involved actor agreed on the necessity of a common environment to facilitate interoperability and safety. The most accepted one is the Integrated Clinical Environment (ICE), that is a new solution involving a number of stakeholders where the final goal is to realize an interoperable network of medical devices in a *safe* way [105]. To fulfill this goal, manufacturers have to use standards such as ICE (Integrated Clinical Environment) or other medical-device interoperability projects. Communication standards are beneficial for manufacturers, health care providers, and users: According to the west health organization [9], the take up of functional interoperability in medical devices can save up to \$30 billion dollars.

Having said the above, this paper focuses on the impact of the software technology on the development of intelligent health solutions and their associated medical systems. By making a biased survey centered in the work of some researchers, the paper exposes selected contributions of a limited part of the research community that target at developing distribution software; this is an exercise to analyze these specific contributions that feed the author's background and will take them as basis for producing a new enhanced solution tailored to the needs of medical device interoperability.

### 1.1 Paper structure

The paper is structured as follows. Section 2 describes the role of the distribution software for developing distributed medical applications, emphasizing the web technology that facilitates interoperability via standard protocols. Section 3 describes state of the art on middleware design and distribution, focusing on the work of the author's group that serves as his reference for improving middleware design for medical systems; also, other technologies are listed that are relevant for his current work. Section 5 describes illustrates some examples of frameworks for medical devices coordination and interoperability projects. Section 6 presents the conclusions and the future work that will be part of the author's thesis.

## 2 Distribution software design for medical systems

Medical systems experiment the same difficulties of the rest of systems as far as its interoperability is concerned. Connectivity problems arise due to incompatibility of the software level (e.g., operating system, networking protocols, format of exchanged data, etc.). This problem has been well identified by the community that develops and designs medical systems and a number of projects have appeared that are studying the way of solving interoperability problems and

providing safe solutions at the same time. The distribution software typically takes care of these type of technical issues, so the middleware community has a lot to say in this picture. Solutions will have to consider ICE, that is the most popular and widely adopted standard for medical device interoperability. Most of its implementations are based on publish-subscribe middleware such as the Data Distribution System [90]. Nevertheless, there is a long road towards obtaining proper solutions for efficient communication and interoperation in a timely, flexible, and reconfigurable manner.

On the author's road to provide one of this solutions, this paper performs a study of a reduced set of solutions in the research to design and develop real-time middleware. The paper analyzes the research on distribution software of his reference distributed real-time systems group that will be used as the background to the research on this field introduced there. This work has the intention of clearly exposing some selected problems and solutions to real-time middleware design for general distributed applications and also for cyber-physical systems. The analyzed work starts with techniques for resource management that are focused at operating system level, the design of software engineering solutions for middleware, service oriented techniques, real-time reconfiguration, modeling of cyber-physical systems and real-time online verification, and hardware acceleration embedded into the middleware logic. Taking this as a solid baseline, the author will extend these works to suit the needs of medical systems.

## 2.1 Middleware definition

The term "*middleware*" has been used under different contexts. At first, it was described as a layer of software located between platforms and applications in distributed systems [20]; it is widely defined as an software layer that connects two or more heterogeneous applications, systems, or structures [24] which, by the end, provides an interface to transfer data and commands among the different stakeholders. In the medical domain, HL7 is one of the most important middleware technologies (referred to in sections 5.1 and 5.2).

Middleware is a reusable software layer that renders standard interfaces and protocols to frequently encountered problems like heterogeneity, interoperability, scalability, fault-tolerance, security, resource-sharing [93]. The middleware technology aim to protect the application layer from problems generated by the lower layers and that are devices heterogeneity, data encoding, security encryption algorithms. Also, middleware constitutes a way to achieve *interoperability*, and this is done by handling the heterogeneity of computer architectures, operating systems, programming languages, and networking protocols to facilitate application development and management [74].

## 2.2 Middleware technologies

**RMI.** RMI (Remote Method Invocation) [8] is a JAVA API (Application Programming Interface), and as its name indicates it is used to call remote methods, [8] defines RMI as a technology to create communication between Java-based

applications, where Java objects can be invoked on the same machine or on different machines deployed on different networks, it supports object polymorphism, its easy to write, secure and mobile, and supports distributed garbage collection. RMI is based on TCP/IP technology and was proposed like a version for the method called RPC (Remote Procedure Call). The two machines that are exchanging data with RMI technology have to run a JVM (Java Virtual Machine), RMI is object oriented and implements the client/server technology.

**CORBA** . CORBA (Common Object Request Broker Architecture) [2] is an OMG standard and is one of the first developed middlewares, it allows different softwares or applications developed with different programming languages and present in different locations to communicate together via an interface broker. The core concept in CORBA is Object Request Broker (ORB), ORB allows a client application to request services from a server application without knowing the details of the application server or the configuration of the network where it's deployed. CORBA uses Interface Description Language (IDL) to define the interfaces of objects which facilitates the communication between different actors.

**iLand**. iLand [106] is an open source middleware that has been applied in industrial prototypes, including medical systems. It follows the classical principles of a layered middleware design; though its architecture is independent of the underlying communication network protocol, the reference implementation of iLand uses a DDS backbone. iLand includes a number of enhanced functions to support dynamically re-configurable applications based on services: light-weight services in the real-time version and web services in the soft and best effort version with QoS guarantees.

**DDS**. DDS stands for Data Distribution Service [90], it is a middleware protocol and API standard for data-centric connectivity provided by OMG (Object Management Group) standard. DDS serves as middleware architecture for a publish/subscribe messaging pattern and integrates the components of a system together, providing low-latency data connectivity, extreme reliability, and a scalable architecture that business and mission-critical Internet of Things (IoT) applications need. DDS is *data centric*, which is ideal for the Internet of Things. Data centricity means that DDS knows what data it stores and controls how to share that data.

### 2.3 Middleware position in distributed medical solutions

The development of distributed applications has been greatly enhanced by middleware technology, one important realization of this last in medical systems is the automatic handling of patient records. Current practice often uses health information systems (HIS) and electronic health record (EHR) in an informal

manner with adhoc protocols and interoperability solutions in order to develop clinical systems [65].

Middleware technology has an important role in medical systems in what follows we mention some of those roles:

- Improves data transfer mechanisms between medical devices and the computational servers where applications are deployed.
- Defines a secure layer that defines algorithms and encryption methods and therefore enhance data security.
- The programmer is more concentrated on the programming details because the hardware details are abstracted by the middleware.
- Achieves interoperability by connecting medical devices, computational servers, applications and data storage solutions.
- Patient data must be private, middleware can be the intermediate that ensures a certain level of privacy and especially when data is transferred into third party stakeholders.
- Improves the portability of applications [108], where an application can be executed under many platforms.

Over the last years, middleware has proven successful to address the ever increasing complexity of distributed systems in a reusable way [78], thus, middlewares are taking an important place in complex systems design like medical applications that controls medical devices that have a direct impact on patient safety or a medical system that controls drug injection in patient body.

Middleware is characterized by abstracting the low level details of the communication protocols and the hardware characteristics of devices to programmers [73]. The distributed nature of medical applications requires the use of technologies such as middlewares that provides a bridge between layers or levels, the hardware layer can be connected to the application layer via a *middleware* and the protocols level can be connected to the security level via a "*middleware*", so the use of the middleware depend on the context of it application. By abstracting the low layers problems and concentrating on the application layer the programmer can generate better solutions. According to [100], middleware technology can be classified into tow types and this depends on the level of use, *low level* middleware exists between sensor nodes and medical devices and *high level* middleware connects diffrent application and tra,sfer data between them.

### 3 Baseline

This section analyses the state of the art work on which the author will build on. Here, it is described the work of the author's group related to the design of resource management strategies for middleware in order to build time-sensitive and efficient systems, as well as real-time middleware design for cyber-physical systems and medical systems.

**Resource management and components for real-time.** This section presents techniques and algorithms to manage the computational resources; it constitutes

the lowest abstraction level related to efficient resource assignment and enforcement techniques to avoid interference in the execution among several application tasks.

Controlling the execution at thread or task level to make an efficient resource usage is essential in distributed applications. There are a number of contributions on the architecture of real-time middleware from a real-time perspective. One of the first ones was named Hola-QoS described in [45] and [39]; using real-time scheduling for distributed actions [16]; real-time quality of service management [92]; mode changing policies for timely execution [43, 44]; agents modeling real-time properties [25]; identification of Linux kernel properties for improving locking [26]; architecting open source projects for Linux [27]; analysis of temporal behavior at bus level within a multiprocessor [75]; reconfiguration scheduling [69].

Other contributions for service oriented systems have been [17]; or component-based modeling over QoS networking [29]; have progressively shed light over how to handle execution in systems using more abstract modeling like encapsulation through services and separating application from networking responsibilities.

On the distribution software side, a number of works have been published to fine tune the resource management policies inside the distribution software. On [96] and [97], a solution to build real-time component replacement for OSGi systems is provided. The work [42] presents a higher abstraction to separate concerns in managing thread-level resources flexibly. Also, [53] provides a higher abstraction resource management policy based on a quality of service specification inside a middleware to flexibly reconfigure applications based on services. Based on this, [49] provides a survey on real-time service composition in distributed systems and proposes a new solution to achieve real-time composition, further enhanced by [51] laying down different alternatives for reconfiguration; and [48] describes two alternatives for application reconfiguration built inside the middleware one at task level and the other at service level. Other approaches describe garbage collection techniques inside the middleware such as [18] that support real-time execution even during system maintenance time.

Other works provide higher level designs of specific applications based on real-time middleware such as [50] that uses the Data Distribution Software for remote control.

As part as the new virtualization paradigms that provide new execution infrastructures where applications can coexist, we find [52, 102] that studies the performance and [60] that provides predictable cloud computing.

**Cyber-physical systems and IoT spheres.** Cyber-physical systems and IoT are highly related because both refer to the monitoring and actuation over physical objects. In the case of medical systems, the physical objects are the patients and the sensors and actuators that monitor them are medical devices (that are very close to IoT devices). Medical devices are part of a medical cyber-physical system (MCPS) that monitor the physical conditions of patients and actuates on them or help in deciding how a human physician will actuate on the patients, providing advice for a recommendation system or a decision support system. Cyber-physical systems have to employ rigorous design techniques because they

are critical systems; for this, they have to use formal methods to verify that the system properties are kept at all times.

CPS have to be systems capable of evolving and should, therefore, be flexible. This is so because CPS in the medical domain target the monitoring of humans; humans have mostly an unpredictable behavior and this may raise unstable situations and unexpected events can be triggered. This means that these systems cannot be designed offline fully as not all the situations are known at design time. CPS have to support evolution and flexibility that are important challenges. It is extremely challenging to integrate the uncertainty raised by the unknown monitored conditions of a patient, etc., and the real-time reaction that is needed, as all the possible situations that could happen when the system is in operation can not be known a priori. In [72], a formal design is described based on Petri nets to model systems that can evolve; this technique was also explored in [71]. A different formal method approach is applied in [22] and [21]. Also, [70] inserts online verification mechanisms based on Petri nets inside a distribution middleware.

In the above works, the focus is placed on the design of the software component interactions relative to their timing properties and other behavioral parameters that are modeled. However, the communication across the above components is a key aspect that must be analyzed to achieve communication and interaction infrastructures that support timely interaction and variable conditions such as load peaks or coexistence of components with heterogeneous resource usage patterns. For this purpose, there are also a number of contributions for the design of distribution middleware for CPS as middleware is the key software layer that is capable of abstracting distribution and interaction, masking situations where a node can receive a peak of requests from other nodes; the systems must be resilient to these and other situations, and they have to continue to work at all times. The design of adaptive middleware is provided in OmaCy architecture [47]. In [54] and [58], an analysis of this problem is outlined. In [56], the design of a scheme for attending simultaneous requests is provided. In [46], a model for integrating the Data Distribution Software with single board computers and Raspberry Pi is provided; this is further reworked in [61] for a different domain such as avionics. Also, there have been a number of dedicated research contributions to building real-time facilities in middleware such as [58,59], among others; or building abstractions for utilization of multiple interaction paradigms such as [95] or [67].

**Medical systems.** It is fundamental to analyze the position of middleware within medical systems in order to develop safe execution solutions that also provide timely operation. Failure to meet these requirements may yield to hazards to lives. Service-oriented architectures such as iLAND [68] (that has been well proven in a number of critical domains) have been integrated with ICE in [73]. The reconfiguration capacity [40] and timely communication of iLAND have been proposed to be the core interoperation backbone for ICE. A number of studies for profiling the actual performance of communication middleware such as [64] has been particularized for medical systems in a number of works such



as [66] for the Internet Communications Engine and [98] for AMQP. Moreover, a number of improvements to their execution by making the middleware aware of the underlying hardware structure have been undertaken in [55] and the benefits of this acceleration in medical systems for remote patient monitoring has been exemplified in [57] for eHealth services and in [41] for audio surveillance.

It is important to bare in mind that designing medical systems requires detailed design and validation of properties as they are critical systems. There are different design and development frameworks for critical systems that have to be applied considering not only the functional properties of the application-level logic, but also the non-functional properties of the whole software stack. Some frameworks exist like [38,63] that support web-based monitoring of critical software projects like the development of medical systems, including their set of libraries like the distribution middleware.

## 4 Key definition of *Medical Devices*

According to the WHO (World Health Organization) definition of “medical device” [10], there are a number of possible hardware systems containing software and hardware parts that sample patient data through reading of vital signs, use networking protocols to exchange data or share it, and have different possibilities on mobility.

The development of medical devices undergoes strict regulations that are established by the FDA (Food and Drug Administration) [3]. The role of FDA has faced criticism on its lack of innovation due to the strict regulations; this is the case of X-Ray machines innovation critics [32]. On the other side of the Atlantic, the European Union adopted the MDD (Medical Device Directive) as the governing set of standards and regulations for medical-devices manufacturer. Some of these directives are the Medical Device Directive (MDD 93/42/EEC), the In Vitro Diagnostic Medical Device Directive (IVDMDD98/79/EC) and the Active Implantable Medical Device Directive (AIMDD 90/385/EEC) [36]. Overall, in different countries, regulation have gone through different paths; however, those countries most involved in medical device regulation established the Global Harmonization Task Force (GHTF) and, after that, the International Medical Device Regulators Forum (IMDRF) [6] appeared with the goal of enforcing a faster medical device regulatory harmonization and convergence at international level in what concerns safety, performance and quality of medical devices.

According to the EU Borderline Manual [33], the following types of software should generally be classified as medical devices:

- picture archiving and communication systems;
- mobile apps for processing ECGs;
- software for delivery and management of cognitive remediation and rehabilitation programs;
- software for information management and patient monitoring; and

- mobile apps for the assessment of moles (e.g. making a recommendation about any changes).

Also, the EU provides recommendations on types of software that should generally not be considered as medical devices. These are mobile apps for communication between patient and caregivers while giving birth or for viewing the anatomy of the human body; software for interpretation of particular guidelines or mobile apps for managing pictures of moles like recording updates and changes over time.

As indicated by the World Health Organization [10], software can be considered as a medical device. Moreover, IMDRF [6] defined the concept of “*Software as a Medical Device*” that is a software intended for use in different devices as medical functions that perform them without being part of a hardware medical device. This follows the classical role of the software engineering vision and software as a service paradigm. It should be noticed that this study considers that a software is a medical device when the functional properties of the software are enough to handle a clinical situation in which the software service is used as a whole medical device.

In the context of this paper, software is a key part, and middleware is a part of software stacks. As the importance of software increases in the health domain [86], the importance of middleware also raises. Given the cyber-physical systems relation to medical devices, also the medical software has vital safety requirements that force it to adhere to strict parameters concerning data accuracy, integrity, security, and verification. This means that its design should follow strict rigorous validation and verification techniques just as CPS do.

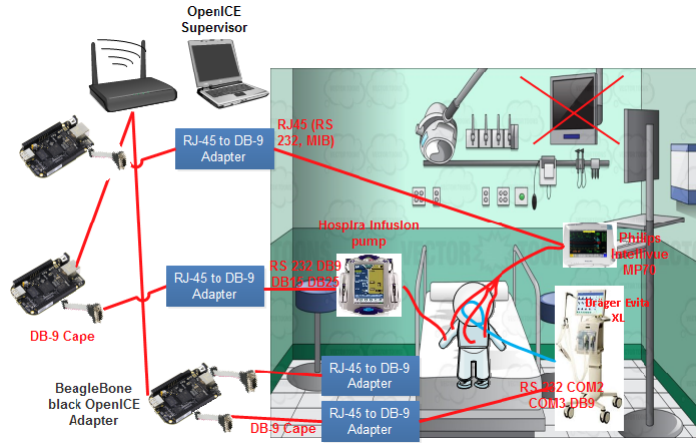
#### 4.1 Analysis of medical devices characteristics

**Safety.** *Safety* [82] is the most important parameter in the development of medical systems. For this, their design must comply to standards like: European MDD [11, 12], ISO 13485 [15], IEC 62366 [13], ISO 14971 [14] and others.

The high variability (e.g., displays, sensors, actuators, communication capacities, and even materials) across different devices challenges their design.

To overcome the communication challenges, medical devices are progressively provided with standard input/output ports, leaving behind the proprietary data output ports that would highly vary across suppliers of medical devices. The most common ports are RS 232 port (DB-9, DB-15, and DB-25), RJ 45, wireless LAN, Bluetooth, USB, or some proprietary data connection systems developed by suppliers for using data by their own IT systems. The following are the most common hardware connections used by suppliers to input data into the device: PS/2 (for supporting keyboard or a mouse inputs), USB, RS 232, and digital data input [105].

**Timing requirements.** Medical systems are time-sensitive. It is the case that different systems have different requirements for timing behavior, ranging from



**Fig. 1.** Hardware configuration of medical devices according to ICE architecture

QoS (quality of service) and best effort to hard real-time guarantees. Therefore, medical devices should support different levels of real-time guarantees in the collection of data, the processing of data, visualization of medical data and alarms, and actuation on the patient or system. The respect of timing requirements by a medical devices is related to the ability of it software to treat and transfer data with consideration to the quality of it hardware components and the good installation of electrodes and cables.

The timing requirements of a medical device include the execution time of an operation (C), deadline (D), and the period (T) of an operation [105], these timing requirements are defined at design time (requirements collection phase) and verified at run time (system operation). Manufacturers can only specify and guarantee the individual devices standalone real time properties which can be distributed via ISO/IEEE 11073 medical device description format [30], but, when connected in a network or to other devices the real timing measurements can be get after executing the system.

**Electromagnetic compatibility.** In some clinical cases, more than one medical device can be integrated into the same OR (Operating Room), also, many medical devices can be used for different patients into a small clinical spot. Electromagnetic fields (EMF) may cause problems on electronic devices [87] and generate safety hazards in the medical devices, the level of those hazards increases when the affected MD has an important role in the treatment process, such as: implantable infusion pumps and cardiac pacemakers. An example of a hazard generated by radio-frequencies is an overdose of insulin from infusion pumps exposed to EMF from mobile phones or RFID (radio frequency identification devices, usually emitting RF EMF of 0.82.4 GHz) [79].

The radio-frequencies generated by a medical device or another source (infrared, microwave or ultrasound therapy) can affect the patient body and the body of others-the physiotherapist or bystanders, if they find themselves within the EMF [76], thus, medical devices must be designed to operate in specified intervals of radio frequencies where their behavior will not be affected by the differences of frequencies. The impact of EMF can generate wrong results of medical devices, the deterioration of the hardware components of another MD, and also can cause cancer for clinicians that are exposed for long periods to EMF.

**Data accuracy.** The final goal of a medical device is to monitor patient safety, inject drugs into patient body in some cases, intervenes in surgical operations, and provide data that is accurate for ulterior use by clinicians and staticians. The big flow of data in healthcare applications generates a need to have well-defined description of rich and structured data required to represent the variety of data used in clinical environment [107] and thereby data that is *correct*. To have accurate data the clinician responsible of the good use of the device must insure the good installation of electrodes and cables on the patient body, the appropriate environmental conditions to the MD and the absence of interoperability issues.

## 5 Interoperability alternatives

HIMSS (Healthcare Information and Management Systems Society) [4] defines *interoperability* in healthcare at three levels:

- Data exchange between two different information technology system. This includes the ability to exchange the data, to receive it and to interpret the data. This is a foundational interoperability level.
- Structure or format of the exchanged data. This refers to the standards for message formats. It implies that the data content and meaning, as well as its operational purpose, is preserved and unaltered.
- Semantic. This level leverages the two levels above and the coding of the data (that includes its vocabulary) in order to allow the receiver systems to interpret the data.

All these levels are common to general purpose middleware applied to the specific domain of health care.

### 5.1 Data exchange standards

Communication and data exchange between medical devices is a basic support for *interoperability*. The basic data exchanged is the EHR. As such, there were a number of EHR incentive programs for building efficient data management in this respect. Two health IT standars competed years ago for this purpose:

**CDA.** This stands for Clinical Document Architecture that was backed up by the Health Level Seven International (HL7).

**CCR.** This stands for Continuity of Care Record that was empowered by the American Society for Testing and Materials (ASTM).

There were a number of proposals for harmonizing the two; however, CDA won over CCR in a first battle. Nowadays, it exists **C-CDA** that stands for Consolidated CDA format that is e by the National Coordinator for Health Information Technology (ONC).

Some other alternatives are shown below.

**DICOM** (Digital Imaging and Communications in Medicine) is a standard developed by the National Electrical Manufacturers Association (NEMA) to store, retrieve, and transfer medical digital information exchange, basically imaging data from imaging equipment, from various medical devices such as scanners, printers, network hardware etc [31].

**HL7 version 2** is a standard designed for messaging in a centralized or distributed environment. Also, it supports the proposition of interfaces for communicating to third parties that do not adhere to data exchange standards. HL 7 standard is one of the most popular ones; precisely, 95% of hospitals, 95% of medical related equipment and information systems in the whole America use it. It is also in use in Germany, Japan and other developed countries [85].

**HL7 version 3** is a newer version of HL7 version 2 that uses the eXtensible Markup Language (XML) as a powerful tool in the web to transfer data. Also, version 3 integrates web services and the Web Services Description Language (WSDL) [89]. Version 3 of HL7 promotes semantic interoperability defined a more explicit methodology for the development of messages [99].

## 5.2 Health specific seamless-communication standards

Following, a number of protocols that provide seamless interoperability in medical systems is provided.

**HL7.** This standard is again listed here as its main goal is to provide seamless integration across a network of medical devices and also in a way that is fully secure. HL7 defines data transfer formats for interoperability across medical devices and HIS (Health Information Systems).

**ISO/IEEE 11073** designates a set of standards for plug and play interoperability of medical devices. ISO/IEEE 11073 defines a common framework for the establishment of a unified data structure model [83]. ISO/IEEE 11073 (X73PHD) [5, 103] objective is to standardize Personal Health Devices (PHDs) and allow semantic interoperability of medical devices by defining the structure of data and the protocol for information delivery between individual medical devices (such as Glucose meter, Weight scale, Blood pressure monitor, etc.) and the manager (computer, smart phone, set top box, etc.), which collects and manages the information from the individual medical devices [88].

**FHIR** (Fast Healthcare Interoperability Resources) of HL7. It is an environment and framework for EHR exchange that integrates both the market needs and the more up-to-date technologies [77]. FHIR targets REST (Representational State Transfer) architectural style as presented by Fielding [34]. For this purpose, it

models the actors of healthcare scenarios as resources: medical software, clinicians modeling, medical devices, medications, or IT structures, among others. FHIR is the most recent standard of the series HL7 v2, HL7 v3, CDA developed by HL7 [19].

### 5.3 Web-services based interoperability

**DPWS** (Devices Profile for Web Services) uses SOA (Service Oriented Architecture) for providing interoperable, cross-platform, cross-domain, and network-agnostic access to devices and their services [37]. DPWS is used for embedded devices with limited resources by enabling Web services using IoT applications. DPWS requires WSDL (Web Service Description Language) and SOAP (Simple Object Access Protocol) to communicate the device services, but it does not need a registry like UDDI (Universal Description, Discovery and Integration) for services discovery. DPWS aims to achieve interoperability by using the loosely coupled concept of Web services over the MD operation and data encryption.

**MDPWS** (Medical Devices Profile for Web Services) is a part of IEEE 11073-20702 series of standards and uses the principles of DPWS but for medical devices interoperability domain with some modifications like the restricted security mechanisms of MDPWS comparing with DPWS, e.g. the usage of client authentication with HTTP authentication is withdrawn in favor of using X.509.v3 certificates [80]. MDPWS uses the principles of DPWS with respect to the high acuity patient environment and the complexity of medical devices.

### 5.4 Software frameworks for medical systems

**OSCP** OSCP (Open Surgical Communication Protocol) [104] is based on the data transmission technology Medical Device Profile for Web Services (MDPWS, standardized as IEEE 11073-20702) and the Domain Information and Service Model (standardized as IEEE 11073-10207). As mentioned before, MDPWS is based on a SOA architecture and it allows devices to detect and find other medical devices in a local network using WS-Discovery.

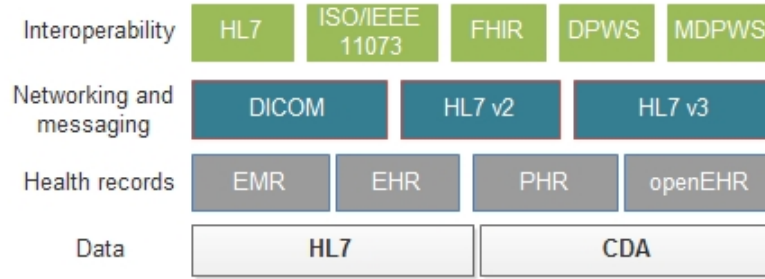
The description of devices is based on formal notations and the creation of device description templates. As it is based on formalisms, it supports using different logics to verify correctness and properties of the systems and the devices such as LTL (Linear Temporal Logic, Smart Assertion Logic for Temporal Logic (SALT), regular LTL (RLTL), etc.

**ICE** (Integrated Clinical Environment) [1] architecture was defined in 2009 in ASTM (American Society for Testing and Materials) F2761 standard. ICE was designed for bridging the gap defined by the high heterogeneity of medical devices. ICE aims at providing a set of specifications and architecture that implements a plug and play atmosphere to create networks of medical devices and to create a communication gateway between them, where messages and commands

are exchanged appropriately. To survive to the next generation of medical applications and systems, manufacturers will have to adhere to the ICE specifications. One of the most important design decisions of ICE is that medical devices have a network output port and must produce data that can be managed through ICE interfaces. The following are the main ICE objectives:

- Improve patient safety by coordinating medical devices actions and avoid incorrect medical decisions generated by a faulty device operation.
- Ensure support for clinicians in their monitoring and treatment operation, where clinical aid information is generated by a set of workflows implemented in the ICE framework logic.
- Create a flexible communication bus between medical devices, servers running medical applications, and the clinicians.
- Implement an interoperable network of medical devices and computational servers where data and messages are exchanged in real time.
- Define standards for the hardware and software characteristics or dimensions of medical devices that will be used by manufacturers to produce medical devices that comply with ICE.

In ICE-based systems *safety* is the ability to implement interoperability between heterogeneous medical devices in a single high acuity patient environment where communication is done via software or hardware interfaces. ICE aims to improve patient *safety* by elaborating and deploying interoperability of the medical devices, thus, creating an interoperate communication bus between heterogeneous medical devices where messages and commands are exchanged.



**Fig. 2.** Hardware configuration of medical devices according to ICE architecture

**OR.NET.** OR.NET is a solution developed by German academics and industrials for medical devices integration and medical systems interoperability in the operating room and its surroundings. The objective of OR.NET is to develop basic concepts for the secure dynamic networking of computer-controlled medical devices in the operating room and clinic [7]. In the end, these concepts are evaluated and transformed to standards. OR.NET aims to create a service-oriented

architecture for the safe and secure dynamic interconnection of medical devices in the OR context [101].

Besides OSCP, OR.NET also allows the use of different communication protocols (e.g. DICOM and HL7 Version 2). OSCP explicitly does not try to replace these widespread protocols. Instead, dedicated gateways are specified that enable the operation of the DICOM and HL7 protocols despite the separation of OR network and the hospital network.

## 6 Conclusion and future works

This paper has analyzed the baseline for design of middleware that applies both to general distributed applications and it is later taken to constitute a take-off platform for designing the needed adjustments for medical systems. The analysis is confined to the research group of reference for the author, comprising the fields that are part of embedded research curricula [28] such as resource management techniques, middleware design, cyber-physical systems, and software engineering techniques for functional design considering non-functional properties. Also, technologies related to medical devices are studied in details from data retrieving to processing and final decisions for clinicians. Requirements of medical devices good operation are presented and interoperability as the core concept in medical devices communication is detailed with examples.

This work is the base for future researches aiming to study and define the position of middleware technology in medical systems and especially in medical devices communications, here ICE is the objective of development. An architecture for ICE-based systems development is under study, this work is the outline for medical devices communications and data exchange through real implementations in clinical environment.

## References

1. CIMIT - Center for Integration of Medicine and Innovative Technology. <http://www.cimit.org/>. Accessed: 2018-07-02.
2. Corba - common object request broker architecture. <http://www.corba.org/>. Accessed: 2018-11-14.
3. FDA - US Food and Drug Administration. <https://www.fda.gov/>. Accessed: 2018-07-13.
4. Healthcare information and management systems society. Accessed: 2018-12-12.
5. Ieee 11073 Personal health devices. <http://www.11073.org/>. Accessed: 2018-08-16.
6. International Medical Device Regulators Forum. <http://www.imdrf.org/>. Accessed: 2018-07-13.
7. OR.NET - Operating Room project. <http://ornet.org/>. Accessed: 2018-07-01.
8. Rmi - remote method invocation. <http://www.oracle.com/technetwork/java/javase/overview/index-jsp-136424.html>. Accessed: 2016-11-14.
9. West health organization. <http://www.who.int/en/>. Accessed: 2018-07-12.



10. World Health Organization - Medical Device Full Definition. <http://www.who.int/medical-devices/full-definition/en/>. Accessed: 2018-07-12.
11. European Council - Council Directive 1993/42/EC. ed.Luxembourg: Official Journal of the European Union, 1993.
12. European Council - Council Directive 2007/47/EC. ed.Luxembourg: Official Journal of the European Union, 2007.
13. International Electrotechnical Commission - IEC 62366:2007. Medical devices - Application of usability engineering to medical devices, ed. Geneva,, 2007.
14. International Organization of Standardization - ISO 13485:2012. Medical devices - Application of risk management to medical devices, ed. Geneva, 2012.
15. International Organization of Standardization - ISO 13485:2012 Medical Devices, Quality management system. Requirements fo regulatory purposes, ed. Geneva, 2012.
16. Alejandro Alonso, Marisol García-Valls, and Juan Antonio de la Puente. Assessment of timing properties of family products. In *Development and Evolution of Software Architectures for Product Families, Second International ESPRIT ARES Workshop, Las Palmas de Gran Canaria, Spain, February 26-27, 1998, Proceedings*, pages 161–169, 1998.
17. Gaetano F. Anastasi, Tommaso Cucinotta, Giuseppe Lipari, and Marisol García-Valls. A qos registry for adaptive real-time service-oriented applications. In *2011 IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2011, Irvine, CA, USA, December 12-14, 2011*, pages 1–8, 2011.
18. Pablo Basanta-Val and Marisol García-Valls. A simple distributed garbage collector for distributed real-time java. *The Journal of Supercomputing*, 70(3):1588–1616, 2014.
19. D. Bender and K. Sartipi. Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 326–331, June 2013.
20. Philip A. Bernstein. Middleware: A model for distributed system services. *Commun. ACM*, 39:86–98, 1996.
21. Marcello M. Bersani and Marisol García-Valls. The cost of formal verification in adaptive CPS. an example of a virtualized server node. In *17th IEEE International Symposium on High Assurance Systems Engineering, HASE 2016, Orlando, FL, USA, January 7-9, 2016*, pages 39–46, 2016.
22. Marcello M. Bersani and Marisol García-Valls. Online verification in cyber-physical systems: Practical bounds for meaningful temporal costs. *Journal of Software: Evolution and Process*, 30(3), 2018.
23. A. Bestbier and P. R. Fourie. Development of a vital signs monitoring wireless ear probe. In *2018 3rd Biennial South African Biomedical Engineering Conference (SAIBMEC)*, pages 1–5, April 2018.
24. Toni A. Bishop and Ramesh K. Karne. A survey of middleware. In *in Proc. 18th Int. Conf. Computers and Their Applications*, pages 254–258, 2003.
25. Alberto Montilla Bravo and Marisol García-Valls. Fipa-based qos negotiator for nomadic agents. In *Mobile Agents for Telecommunication Applications, 4th International Workshop, MATA 2002 Barcelona, Spain, October 23-24, 2002, Proceedings*, pages 216–226, 2002.
26. Peter T. Breuer and Marisol García-Valls. Static deadlock detection in the linux kernel. In *Reliable Software Technologies - Ada-Europe 2004, 9th Ada-Europe International Conference on Reliable Software Technologies, Palma de Mallorca, Spain, June 14-18, 2004, Proceedings*, pages 52–64, 2004.

27. Peter T. Breuer and Marisol García-Valls. Raiding the Noosphere: the open development of networked RAID support for the linux kernel. *Softw., Pract. Exper.*, 36(4):365–395, 2006.
28. Paul Caspi, Alberto L. Sangiovanni-Vincentelli, Luís Almeida, Albert Benveniste, Bruno Bouyssounouse, Giorgio C. Buttazzo, Ivica Crnkovic, Werner Damm, Jakob Engblom, Gerhard Fohler, Marisol García-Valls, Hermann Kopetz, Yasmine Lakhnech, François Laroussinie, Luciano Lavagno, Giuseppe Lipari, Florence Maraninchi, Philipp Peti, Juan Antonio de la Puente, Norman Scaife, Joseph Sifakis, Robert de Simone, Martin Törngren, Paulo Veríssimo, Andy J. Wellings, Reinhard Wilhelm, Tim A. C. Willemse, and Wang Yi. Guidelines for a graduate curriculum on embedded software and systems. *ACM Trans. Embedded Comput. Syst.*, 4(3):587–611, 2005.
29. M. A. de Miguel, J. F. Ruiz, and M. Garcia. Qos-aware component frameworks. In *IEEE 2002 Tenth IEEE International Workshop on Quality of Service (Cat. No.02EX564)*, pages 161–169, May 2002.
30. C. Dietz, T. Lueddemann, M. E. Dingler, and T. C. Lueth. Automated risk detection for medical device networks with hard real time requirements. In *2016 IEEE/SICE International Symposium on System Integration (SII)*, pages 471–476, Dec 2016.
31. A. J. Dinu, R. Ganesan, A. A. Kebede, and B. Veerasamy. Performance analysis and comparison of medical image compression techniques. In *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 738–745, Dec 2016.
32. Karen B Ekelman et al. Technological innovation and medical devices. 1988.
33. European Union. Manual on borderline and classification in the community regulatory framework for medical devices. v1.18, December 2017.
34. Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887.
35. Foof and Drug Administration. Medical Device Interoperability. <https://www.fda.gov/MedicalDevices/DigitalHealth/ucm512245.htm>. Accessed: 2018-07-01.
36. Elaine French-Mowat and Joanne Burnett. How are medical devices regulated in the european union? *Journal of the Royal Society of Medicine*, 105(1\_suppl):22–28, 2012.
37. K. Fysarakis, D. Mylonakis, C. Manifavas, and I. Papaefstathiou. Node.dpws: Efficient web services for the internet of things. *IEEE Software*, 33(3):60–67, May 2016.
38. Javier García-Munoz, Marisol García-Valls, and Julio Escribano-Barreno. Improved metrics handling in sonarqube for software quality monitoring. In *Distributed Computing and Artificial Intelligence, 13th International Conference, DCAI 2016, Sevilla, Spain, 1-3 June, 2016*, pages 463–470, 2016.
39. Marisol García-Valls. Calidad de servicio en sistema multimedia empotrados mediante gestión dinámica de recursos. Universidad Politécnica de Madrid, 2001. PhD thesis (In Spanish).
40. Marisol García-Valls. A proposal for cost-effective server usage in CPS in the presence of dynamic client requests. In *19th IEEE International Symposium on Real-Time Distributed Computing, ISORC 2016, York, United Kingdom, May 17-20, 2016*, pages 19–26, 2016.
41. Marisol García-Valls. Integrating multicore awareness functions into distribution middleware for improving performance of distributed audio surveillance. *Advances in Engineering Software*, 2019.

42. Marisol García-Valls and Alejandro Alonso. Integration of system-level policies and mechanisms for quality of service management for web-based environments. In *Proceedings of the IADIS International Conference WWW/Internet 2002, ICWI 2002, Lisbon, Portugal, November 13-15, 2002*, pages 653–657, 2002.
43. Marisol García-Valls, Alejandro Alonso, and Juan Antonio de la Puente. Mode change protocols for predictable contract-based resource management in embedded multimedia systems. In *International Conference on Embedded Software and Systems, ICESS '09, Hangzhou, Zhejiang, P. R. China, May 25-27, 2009.*, pages 221–230, 2009.
44. Marisol García-Valls, Alejandro Alonso, and Juan Antonio de la Puente. A dual-band priority assignment algorithm for dynamic qos resource management. *Future Generation Computer Systems*, 28(6):902 – 912, 2012.
45. Marisol García-Valls, Alejandro Alonso, José Ruiz, and Angel Groba. *An Architecture of a Quality of Service Resource Manager Middleware for Flexible Embedded Multimedia Systems*, pages 36–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
46. Marisol García-Valls, Javier Ampuero-Calleja, and Luis Lino Ferreira. Integration of data distribution service and raspberry pi. In *Green, Pervasive, and Cloud Computing - 12th International Conference, GPC 2017, Cetara, Italy, May 11-14, 2017, Proceedings*, pages 490–504, 2017.
47. Marisol García-Valls and Roberto Baldoni. Adaptive middleware design for CPS: considerations on the os, resource managers, and the network run-time. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware, ARM@Middleware 2015, Vancouver, BC, Canada, December 7-11, 2015*, pages 3:1–3:6, 2015.
48. Marisol García-Valls and Pablo Basanta-Val. A practical solution for functional reconfiguration of real-time service based applications through partial schedulability. In *REACTION 2012, First International Workshop on Real-time and distributed computing in emerging applications, Proceedings, San Juan, Puerto Rico, December 4, 2012*, 2012.
49. Marisol García-Valls and Pablo Basanta-Val. A real-time perspective of service composition: Key concepts and some contributions. *Journal of Systems Architecture - Embedded Systems Design*, 59(10-D):1414–1423, 2013.
50. Marisol García-Valls and Pablo Basanta-Val. Usage of DDS Data-Centric Middleware for remote monitoring and control laboratories. *IEEE Trans. Industrial Informatics*, 9(1):567–574, 2013.
51. Marisol García-Valls and Pablo Basanta-Val. Comparative analysis of two different middleware approaches for reconfiguration of distributed real-time systems. *Journal of Systems Architecture - Embedded Systems Design*, 60(2):221–233, 2014.
52. Marisol García-Valls and Pablo Basanta-Val. Analyzing point-to-point DDS communication over desktop virtualization software. *Computer Standards & Interfaces*, 49:11–21, 2017.
53. Marisol García-Valls, Pablo Basanta-Val, Marga Marcos, and Elisabet Estévez-Estévez. A bi-dimensional qos model for SOA and real-time middleware. *Comput. Syst. Sci. Eng.*, 28(5), 2013.
54. Marisol García-Valls, Paolo Bellavista, and Aniruddha S. Gokhale. Reliable software technologies and communication middleware: A perspective and evolution directions for cyber-physical system, mobility, and cloud computing. *Future Generation Comp. Syst.*, 71:171–176, 2017.

55. Marisol García-Valls and Christian Calva-Urrego. Improving service time with a multicore aware middleware. In *Proceedings of the Symposium on Applied Computing, SAC 2017, Marrakech, Morocco, April 3-7, 2017*, pages 1548–1553, 2017.
56. Marisol García-Valls, Christian Calva-Urrego, Juan Antonio de la Puente, and Alejandro Alonso. Adjusting middleware knobs to assess scalability limits of distributed cyber-physical systems. *Computer Standards & Interfaces*, 51:95–103, 2017.
57. Marisol García-Valls, Christian Calva-Urrego, and Ana García-Fornes. Accelerating smart eHealth services execution at the fog computing infrastructure. *Future Generation Comp. Syst.*, –:–, 2018.
58. Marisol García-Valls, Antonio Casimiro, and Hans P. Reiser. A few open problems and solutions for software technologies for dependable distributed systems. *Journal of Systems Architecture - Embedded Systems Design*, 73:1–5, 2017.
59. Marisol García-Valls and Tommaso Cucinotta. Real-time and distributed computing in emerging applications. foreword by the general chairs of reaction 2012. *Journal of Systems Architecture - Embedded Systems Design*, 61(5-6):267–268, 2015.
60. Marisol García-Valls, Tommaso Cucinotta, and Chenyang Lu. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture - Embedded Systems Design*, 60(9):726–740, 2014.
61. Marisol García-Valls, Jorge Domínguez-Poblete, Imad Eddine Touahria, and Chenyang Lu. Integration of data distribution service and distributed partitioned systems. *Journal of Systems Architecture - Embedded Systems Design*, 83:23–31, 2018.
62. Marisol García-Valls, Abhishek Dubey, and Vicent Botti. Introducing the new paradigm of Social Dispersed Computing: Applications, technologies and challenges. *Journal of Systems Architecture*, 91:83 – 102, 2018.
63. Marisol García-Valls, Javier García-Munoz, and Julio Escribano-Barreno. An extensible and collaborative framework for monitoring software quality in critical systems. *Information and Software Technology*, –:–, 2019.
64. Marisol García-Valls, Daniel Garrido, and Manuel Díaz. Impact of middleware design on the communication performance. In *Green, Pervasive, and Cloud Computing - 12th International Conference, GPC 2017, Cetara, Italy, May 11-14, 2017, Proceedings*, pages 505–519, 2017.
65. Marisol García-Valls, Natividad Herrasti, Christof Jouvray, and Aintzane Armentia. Flexible and timely on-line integration of medical services using iland middleware. *SIGBED Rev.*, 2017.
66. Marisol García-Valls, Natividad Herrasti, Christophe Jouvray, and Aintzane Armentia. Flexible and timely on-line integration of medical services using iland middleware. *SIGBED Review*, 14(2):53–60, 2017.
67. Marisol García-Valls and Felipe Ibáñez-Vázquez. Integrating middleware for timely reconfiguration of distributed soft real-time systems with ada DSA. In *Reliable Software Technologies - Ada-Europe 2012 - 17th Ada-Europe International Conference on Reliable Software Technologies, Stockholm, Sweden, June 11-15, 2012. Proceedings*, pages 35–48, 2012.
68. Marisol García-Valls, Iago Rodríguez Lopez, and Laura Fernández-Villar. iland: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. *IEEE Trans. Industrial Informatics*, 9(1):228–236, 2013.

69. Marisol García-Valls, Alejandro Alonso Muñoz, and Juan Antonio de la Puente. Time-predictable reconfiguration with contract-based resource management. In *23rd International Conference on Advanced Information Networking and Applications, AINA 2009, Workshops Proceedings, Bradford, United Kingdom, May 26-29, 2009*, pages 494–499, 2009.
70. Marisol García-Valls, Diego Perez-Palacin, and Raffaella Mirandola. Extending the verification capabilities of middleware for reliable distributed self-adaptive systems. In *12th IEEE International Conference on Industrial Informatics, INDIN 2014, Porto Alegre, RS, Brazil, July 27-30, 2014*, pages 164–169, 2014.
71. Marisol García-Valls, Diego Perez-Palacin, and Raffaella Mirandola. Time-sensitive adaptation in CPS through run-time configuration generation and verification. In *IEEE 38th Annual Computer Software and Applications Conference, COMPSAC 2014, Vasteras, Sweden, July 21-25, 2014*, pages 332–337, 2014.
72. Marisol García-Valls, Diego Perez-Palacin, and Raffaella Mirandola. Pragmatic cyber physical systems design based on parametric models. *Journal of Systems and Software*, 144:559–572, 2018.
73. Marisol García-Valls and Imad Eddine Touahria. On line service composition in the integrated clinical environment for ehealth and medical systems. *Sensors*, 17(6):1333, 2017.
74. Kurt Geihs. Middleware challenges ahead. *Computer*, 34(6):24–31, June 2001.
75. Angel M. Groba, Alejandro Alonso, José A. Rodríguez, and Marisol García-Valls. Response time of streaming chains: Analysis and results. In *14th Euromicro Conference on Real-Time Systems (ECRTS 2002), 19-21 June 2002, Vienna, Austria, Proceedings*, page 182, 2002.
76. Krzysztof Gryz and Jolanta Karpowicz. Environmental impact of the use of radiofrequency electromagnetic fields in physiotherapeutic treatment. *Roczniki Panstwowego Zakladu Higieny*, 65 1:55–61, 2014.
77. N. Hong, K. Wang, L. Yao, and G. Jiang. Visual fhir: An interactive browser to navigate hl7 fhir specification. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 26–30, Aug 2017.
78. Valerie Issarny, Mauro Caporuscio, and Nikolaos Georgantas. A perspective on the future of middleware-based software engineering. In *2007 Future of Software Engineering, FOSE '07*, pages 244–258, Washington, DC, USA, 2007. IEEE Computer Society.
79. Jolanta Karpowicz and Krzysztof Gryz. An assessment of hazards caused by electromagnetic interaction on humans present near short-wave physiotherapeutic devices of various types including hazards for users of electronic active implantable medical devices (aimd). *BioMed research international*, 2013, 2013.
80. M. Kasparick, S. Schlichting, F. Golasowski, and D. Timmermann. Medical dpws: New ieee 11073 standard for safe and interoperable medical device communication. In *2015 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 212–217, Oct 2015.
81. A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In *2009 31st International Conference on Software Engineering - Companion Volume*, pages 141–151, May 2009.
82. M. Lepmets, T. McBride, and F. McCaffery. Towards safer medical device software systems: Industry-wide learning from failures and the use of safety-cases to support process compliance. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 193–198, Sept 2016.

83. Hai-Long Li, Zhi-Bin Duan, Jin-Zhong Cui, and Zhen-Wei Chen. A design of general medical data adapter based on iso/ieee 11073 standards. In *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 404–407, Dec 2015.
84. J. Lu, W. Hu, M. Song, X. Zhan, and X. Liu. Mobile medical service system based on portable devices. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 1530–1535, Aug 2015.
85. X. Lu, Y. Gu, J. Zhao, N. Yu, and W. Jia. Research and implementation of medical information format conversion based on hl7 version 2.x. In *2011 International Conference on Computer Science and Service System (CSSS)*, pages 2440–2443, June 2011.
86. M. Mchugh, F. Mccaffery, and V. Casey. Software process improvement to assist medical device software development organisations to comply with the amendments to the medical device directive. *IET Software*, 6(5):431–437, October 2012.
87. Kjell Hansson Mild, Tommi Alanko, Gilbert Decat, Rosaria Falsaperla, Krzysztof Gryz, Maila Hietanen, Jolanta Karpowicz, Paolo Rossi, and Monica Sandstrm. Exposure of workers to electromagnetic fields. a review of open questions on exposure assessment techniques. *International Journal of Occupational Safety and Ergonomics*, 15(1):3–33, 2009. PMID: 19272237.
88. J. Nam, W. Seo, J. Bae, and Y. Cho. Design and development of a u-health system based on the iso/ieee 11073 phd standards. In *The 17th Asia Pacific Conference on Communications*, pages 789–793, Oct 2011.
89. R. Noumeir and J. F. Pambrun. Hands-on approach for teaching hl7 version 3. In *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, pages 1–4, Nov 2010.
90. OMG. Data distribution software, v1.4. <https://www.omg.org/spec/DDS/>, 2015.
91. World Health Organization. *Diet, nutrition, and the prevention of chronic diseases: report of a joint WHO/FAO expert consultation*, volume 916. World Health Organization, 2003.
92. Clara Otero-Pérez, Liesbeth Steffens, Peter van der Stock, Sjr van Loo, Alejandro Alonso, José Ruiz, Reinder Brill, and Marisol García-Valls. Qos-based resource management for ambient intelligence. 2003.
93. L. Qilin and Z. Mintian. The state of the art in middleware. In *2010 International Forum on Information Technology and Applications*, volume 1, pages 83–85, July 2010.
94. V. Randazzo, E. Pasero, and S. Navaretti. Vital-ecg: A portable wearable hospital. In *2018 IEEE Sensors Applications Symposium (SAS)*, pages 1–6, March 2018.
95. Iago Rodríguez-López and Marisol García-Valls. Architecting a common bridge abstraction over different middleware paradigms. In *Reliable Software Technologies - Ada-Europe 2011 - 16th Ada-Europe International Conference on Reliable Software Technologies, Edinburgh, UK, June 20-24, 2011. Proceedings*, pages 132–146, 2011.
96. Julio Cano Romero and Marisol García-Valls. Scheduling component replacement for timely execution in dynamic systems. *Softw., Pract. Exper.*, 44(8):889–910, 2014.
97. Julio Cano Romero and Marisol García-Valls. On the free, safe, and timely execution of component-based systems. *CoRR*, abs/1512.04844, 2015.

98. Paloma Rubio-Conde, Diego Villarán-Molina, and Marisol García-Valls. Measuring performance of middleware technologies for medical systems: Ice vs AMQP. *SIGBED Review*, 14(2):8–14, 2017.
99. M. I. Sabar, P. M. Jayaweera, and E. A. T. A. Edirisuriya. International interoperability through unified universal hl7 v3 green messaging. In *2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 112–118, Aug 2015.
100. Mangal Sain, HoonJae Lee, and Wan-Young Chung. Middleware for ubiquitous healthcare information system. In *2009 11th International Conference on Advanced Communication Technology*, volume 03, pages 2325–2328, Feb 2009.
101. J. Schlamelcher, M. Onken, M. Eichelberg, and A. Hein. Dynamic dicom configuration in a service-oriented medical device architecture. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1717–1720, Aug 2015.
102. Rosbel Serrano-Torres, Marisol García-Valls, and Pablo Basanta-Val. Virtualizing DDS middleware: Performance challenges and measurements. In *11th IEEE International Conference on Industrial Informatics, INDIN 2013, Bochum, Germany, July 29-31, 2013*, pages 71–76, 2013.
103. ISO Staff. Iso/ieee 11073-20601: Health informatics–personal health device communication; part 20601 application profile-optimized exchange protocol. *Ginebra, Suiza*, 2010.
104. The Institute for Software Engineering and Programming Languages. Oscp – open surgical communication protocol. <https://www.isp.uni-luebeck.de/oscp>. Accessed: 2018-12-12.
105. I. E. Touahria, M. García-Valls, and A. Khababa. An ICE compliant component model for medical systems development. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 278–287, July 2017.
106. M. Garcia Valls, I. R. Lopez, and L. F. Villar. iland: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. *IEEE Transactions on Industrial Informatics*, 9(1):228–236, Feb 2013.
107. Sandy Weininger. Integrated clinical environment device model: Stakeholders and high level requirements.
108. W. Wolf. Middleware architectures for distributed embedded systems. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 377–380, May 2008.