People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY FERHAT ABBAS - SETIF1

FACULTY OF SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

THESIS

Presented in Fulfilment of the Requirements

For the Degree of

DOCTORATE

Domain : Computer Science

By

TOUAHRIA Imad Eddine

SUBJECT

Design and Implementation of Real-Time Applications Based on Component Models

Defended on February 02 2022 in front of jury:

BOUBETRA Abdelhak	Professor	University. Bordj Bou-Arreridj	President
KHABABA Abdallah	Professor	University. Ferhat Abbas Sétif 1	Supervisor
BESSOU Saddik	Doctor	University. Ferhat Abbas Sétif 1	Reviewer
LAKHFIF Abdelaziz	Doctor	University. Ferhat Abbas Sétif 1	Reviewer

Declaration of Authorship

I, Imad Eddine Touahria, declare that this thesis titled, 'Design and realization of real-time applications based on component models' and the work presented in it are my own. I confirm that:

1. This work was done wholly or mainly while in candidature for a Phd degree at Ferhat Abbas Sétif-1 University.

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

3. Where I have consulted the published work of others, this is always clearly attributed.

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

5. I have acknowledged all main sources of help.

6. Where the thesis is based on work done by myself.

Signed: ____

Date: ____

"Know how to solve every problem that has ever been solved."

Richard Feynman

Abstract

Real time systems (RTS) are used in a wide variety of applications, ranging from very simple electronic systems to the most complex avionics and medical systems. RTS are time-related, which means the behavior of these systems depends on the timing requirements defined in the design time and refined during execution time. Component based programming is the typical paradigm to develop RTS and this is due to it modular nature, which helps to hide the complexity of such systems and minimizes development costs. One example of RTS is a medical system that monitors a patient vital signs in a high acuity environment, this situation is characterized by the presence of many medical devices where each device has it own pre-defined tasks and required measurements. The presence of more than one device in one clinical situation may affect patient safety because of the lack of communication between these devices and therefore interoperability is needed to avoid hazards and patients health status degradation.

The interconnection of medical devices in a wider network of computational servers is emerging as a new requirement in modern medicine, where the final goal is to monitor and improve patient safety. The heterogeneity of medical devices provided by different suppliers is a key challenge in medical systems, where interoperability and data communication across devices is still under study and specification. The proposed solution in this thesis aims to create a standard interface to cover medical devices heterogeneity, hence, achieve interoperability in a safe way with respect to timing requirements. It's based on the integrated clinical environment (ICE) standard and component technology as a programming paradigm, the architecture focuses on defining an interoperable bus between the patient, medical devices, software applications and the clinician. This work illustrates the component model in detail and validates it with a prototype implementation.

Résumé

Les systèmes temps réel (STR) sont utilisés dans une grande variété d'applications, allant des petits systèmes électroniques aux systèmes avioniques et médicaux les plus complexes. Les STR sont liés au temps, ce qui signifie que le comportement de ces systèmes dépend des exigences temporelles définies au moment de la conception et affinées durant l'exécution. Les composants logiciels sont le paradigme typique pour développer les STR et cela est dû à leurs nature modulaire, chose qui aide à abstraire la complexité de tels systèmes et minimise les coûts de développement. Un exemple de STR est un système médical qui surveille les signes vitaux d'un patient dans un environnement de haute acuité, cette situation est caractérisée par la présence de plusieurs dispositifs médicaux où chaque dispositif à ses propres tâches prédéfinies et mesures requises. La présence de plusieurs dispositifs dans une situation clinique peut affecter la sécurité des patients en raison du manque de communication entre ces dispositifs ,et par conséquent, l'interopérabilité est nécessaire pour éviter la dégradation de l'état de santé des patients.

L'interconnexion des dispositifs médicaux dans un réseau plus large de serveurs informatiques est la nouvelle exigence de la médecine moderne, où l'objectif final est de surveiller et d'améliorer l'état de santé des patients. L'hétérogénéité des dispositifs médicaux fournis par différents fournisseurs est un défi majeur dans les systèmes médicaux, où l'interopérabilité et l'échange de données entre les appareils sont encore sous étude et spécification. La solution proposée dans cette thèse vise à créer une interface standard pour couvrir l'hétérogénéité des dispositifs médicaux, par conséquent, réaliser l'interopérabilité d'une manière sûre et avec prise en charge des contraintes temporelles. La solution est basée sur l'environnement clinique intégré (ECI) et les composants logiciels comme paradigme de programmation, l'architecture vise à réaliser un bus interopérable entre le patient, les dispositifs médicaux, les applications logicielles et le médecin. Ce travail illustre le modèle de composants en détail et le valide avec une implémentation.

Acknowledgements

First and foremost I am extremely grateful to my supervisors, Prof. Abdallah Khababa for his invaluable advice, continuous support, and patience during my PhD study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to express my gratitude to my parents, Professor Mohamed Touahria and my mother Hassiba Sabah, my brother Hichem, my two sisters Khouloud and Nour Hasna. Without their tremendous understanding and encouragement in the past few years, it would be impossible for me to complete my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Context

In recent years, real-time systems are being more integrated in our daily life, varying from car control and mobile phones communication systems to the most complex medical systems. A real-time system must complete it work and deliver services on a timely basis. In other words, real-time systems have stringent timing requirements that must be met [11]. RTS have to meet their functional requirements, which means verifying the timing requirements is not sufficient to qualify a system as real time, as example, a car brake system has to minimize speed without affecting the operation of other components. Component based software engineering (CBSE) has been considered to be a good way to cope with the increasing complexity of RTS by dividing software systems into manageable parts [12].
The increasing complexity of RTS leads to increasing demands with respect to requirements engineering, high level design, early error detection, verification, and maintenance [13]. CBSE is the best solution to hide the complexity of RTS and this is due to the separation of code and interfaces, where the code integrates the system functions and logic while interfaces are the mean to communicate with external components by providing and requiring data or signal, examples of the application of CBSE on RTS are air-traffic management [14], shipboard computing environments [15], and robotics [16].
Defining an appropriate architecture for RTS requires using component technology. Com-

ponent models provide modularity, easy services composability, and differentiates the interface definition from the actual implementation and code. Component frameworks are defined as the basic abstract infrastructures for component construction and management [17]. They vary from a simple structure, to a distributed critical application, and are classified according to their functions: modelling, analysis, simulation, and execution support [18].

Modern medical systems are increasingly using medical devices in the processes of patient diagnosis, monitoring and treatment [19], therefore, the study of medical devices and technologies related to them is important for any research aiming to use medical devices as material. The old generations of analogue electromechanical devices are now being replaced by connected devices and information technology based systems across the diverse array of contemporary medical devices, what used to be passive devices controlled by a human are now complex computing systems embedding sensors and actuators not only to monitor patient vital signs, but also to control the critical physiological processes and functions [20].

In actual medical systems two essential issues are present. First, the communication of data between medical devices and second the transfer of data between those devices and IT infrastructures. Exchanging data between medical devices is still under study and specification because each device has its own hardware and software characteristics, and patient safety standards require more security and compatibility improvements. The communication of measured data from a medical device to an IT infrastructure is possible just between devices and IT infrastructures coming from the same supplier and with predefined characteristics.

Domains like automotive, avionics and military systems are increasingly moving from stand-alone, monolithic systems to integrated platforms by creating communication between micro-controllers, sensors, and actuators [21]. While in the medical domain interoperability is still under specification and research because numerous safety and effectiveness benefits can accrue from achieving interoperability. The trend is to produce medical devices with networking capabilities, predictions estimates by 2020, over 50 billion devices will be connected to the Internet [22], this means, medical devices will exchange and share data to accomplish specific clinical tasks and therefore lower the rates of preventable medical errors, now estimated to be as high as the third leading cause of death in the U.S. [1].

Interoperability is a requirement [23] and a patient safety issue [24]. Many medical devices may be connected to a single patient, therefore clinicians have to monitor devices independently, summarize data, and act on their observations and this can be affected by human mistakes or fatigue. For more than four decades clinical scenarios documented shortfalls in patient care and the associated need for medical device interoperability and this due to the lack of an open communication standard, manufacturers tend to equip their devices with proprietary communication protocols that inhibit cross-manufacturer data exchange between medical devices [25] therefore the development of a standard for medical devices interoperability is a necessity and an urgent need.

For technical as well as legal reasons, interconnecting medical devices dynamically is in most cases still not possible in practice. Usually the legislation requires that devices used together have been tested in every possible situation and certified in that particular combination, and at the same time, testing and certifying all possible combinations of devices in advance, though, is not a viable approach. Also, if the clinic operator would interconnect devices without respecting their intended use he would have to test the combination himself and take responsibility for any failures induced by the interconnection. Commonly, these tasks could not be handled by the clinic operator, especially, as to him a device comes as a black box [26].

In healthcare, some non-PNP interoperability solutions have been implemented mostly between devices and Electronic Health Records (EHRs). These solutions, some utilizing dedicated connection gateways, have been implemented using open networking standards, such as HL7, DICOM and IEEE 11073. Implementations based on these standards, may use profiles, such as those from IHE, and reusable design guidelines and software source code such as from the Continua Health Alliance. Unfortunately these approaches have not led to widespread adoption of PNP interoperability at the point of care for devices from heterogeneous manufacturers. Current solutions include proprietary vertically integrated solutions from single manufacturers and custom hospital developed solutions [27].

Many medical devices on the market support some form of connectivity through serial ports, Ethernet, 802.11b or Bluetooth wireless, etc. Unfortunately, device interoperability is lacking due to the limited implementation of interfacing standards, and connectivity is usually only leveraged by Medical Device Data Systems (MDDSs) that uni-directionally transfer data/events from devices to composite monitors or electronic health records (EHRs) [28]. In medicine, clinicians currently deliver therapy by manually coordinating collections of independently developed devices. Now that many devices marketed today already include some form of network connectivity (serial ports, Ethernet, 802.11 or Bluetooth wireless) clinicians are recognizing the potential to automate device coordination via external control applications [29], Ideally, future medical devices would support plug & play protocols which would allow clinicians to construct networks of medical devices that automatically interoperate to automate life-critical clinical workflows [30].

1.2 Objectives

Numerous efforts in academia and industry have studied different characteristics and implications of plug & play medical systems. Almost all of the prior studies have concluded that achieving plug & play medical systems requires an "open interoperability standard" or a "platform specific solution" similar to what exists in electronics industry.

In this work, the proposed solution is based on the Integrated Clinical Environment (ICE) an "open interoperability standard". The solution aims to achieve interoperability by exchanging messages and data between medical devices and at the same time behave like a real time system in order to deliver results to the different stakeholders by the pre-defined deadlines.

The component model integrates web services to resolve data communication between medical devices and this is by transferring data from hardware to the software level and try to cover differences between medical devices. Web services are also proposed to achieve a certain logic and create communication between the top and low levels of the architecture.

A composed component is integrated to ensure data encryption and conversion between medical devices, web services and the other components, it's the heart of the model where all the primary calculation and basic operations are done and sent to upper levels.

In addition to realizing interoperability, the proposed solution aims to ensure patient safety by delivering accurate results to the clinicians and by respecting timing requirements. Also, hazards resulting from the connection of two incompatible device are studied.

1.3 Scientific publications

The research was conducted by some scientific publications in the form of journal papers, conference papers, and a poster.

An ICE Compliant Component Model for Medical Systems Development

IEEE Computers, Software & Applications in an Uncertain World (COMPSAC) 2017, Turin, Italy

Imad Eddine Touahria, Marisol García-Valls, Abdallah Khababa.

The paper presents the first version of the proposed component model to realize interoperability between medical devices with respect to the real time requirements of medical systems, the model was validated through a prototype implementation of a patient monitoring system.

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

Computing Research Repository (CoRR) 2018, arxiv

Imad Eddine Touahria

The paper analyses a number of previous work on middleware and pay especial attention to the middleware for web-based systems and how they relate to the development of distributed medical systems.

Medical systems, the role of middleware and survey on middleware design

Computing Research Repository (CoRR) 2018, arxiv

Imad Eddine Touahria

This paper reviews the state of the art on middleware as facilitator for interconnection among devices and also describes a number of initiatives and projects that further extend the underlying distribution software towards the clinical domain as device interconnection facilitators.

Improving patient safety by realizing medical devices interoperability

Conference on informatics and applied mathematics (IAM) 2019, Guelma, Algeria

Imad Eddine Touahria, Abdallah Khababa

The poster presented some configurations of medical devices in a clinical situations and the need to avoid hazards to improve patient safety. The poster also presented some improvements of the developed component comparing with it first version.

A Component Based Framework to Enable Medical Devices Communication

Ingénierie des Systémes d'Information journal, IIETA 2021

Imad Eddine Touahria, Abdallah Khababa

This work illustrates the last version of the component model in detail, defines it relation with the integrated clinical environment and details the operation of the mediator.

1.4 Thesis outline

The thesis is divided into six chapters organized as follows:

Chapter 2 illustrates the background concepts related to the work, it defines the set of technologies used to develop a component based model to realize interoperability between medical devices.

Chapter 3 presents interoperability between medical devices, it link with real time systems and technologies used in the development of some initiatives related to the research subject.

Chapter 4 defines the Integrated Clinical Environment (ICE) as the base of the developed component model and shows some of it characteristics, how it deals with interoperability issues and time management in real time configurations.

Chapter 5 provides the component based model architecture to achieve interoperability between medical devices in high acuity patient environment, it also illustrates the set of internal components of the model and their inter-relation.

Chapter 6 validates the proposed component model through the design and implementation of a prototype in an exhaustive way.

Chapter 7 draws some conclusions and describes the future work.

Chapter 2

State of the art

In this chapter the set of concepts related to the research topic are defined including real time systems, medical devices, software components and models, and some modeling languages.

2.1 Real time systems

A RTS is composed of a control system and a controlled system, the controlled system is the physical process that the RTS collects information using sensors and executes actions using actuators. Time scale and deadlines definition vary according to the nature of the controlled process, for example, deadlines in a system controlling a tourist boat are smaller than deadlines for a system directing a TGV.

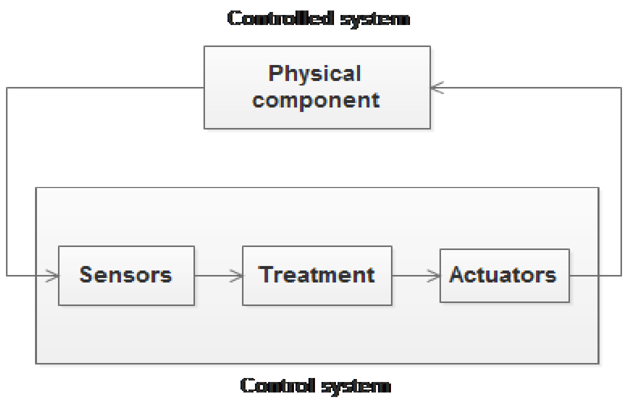

Figure 2.1: Real time system components

It's the speed of evolution of the controlled system that defines the dynamics of the system and the timing requirements, time intervals in the control system are greater than the controlled system. The essential criterion to differentiate a RTS from all other types of systems is the presence of strict time constraints and deadlines for the majority of events taking place in this system.

2.1.1 Timing parameters of RTS

A real time system can have various timing parameters that define the system execution behavior according to time progress. In what follows we define the essential timing parameters used in RTS.

- Deadline (D). It is the maximum time for an activity, operation, or task to complete since a specified reference point that is typically the activation of the operation or task. In an ICE-enabled environment, there are hard deadlines meaning that data arriving after D time units is not valid and becomes irrelevant.
- Execution time of an operation (C). That is the amount of time (processor cycles) needed by a given task or operation to complete.
- Worst case execution time (WCET). Is defined as the biggest execution time of an operation, it is concluded after running a specific operation with varying processor load.
- Period (T) of an operation. It is the elapsed time between two consecutive activations of the operation. Periodic operations or tasks are those that are performed at regular intervals, e.g. sampling oxygen levels, injection of a medicine to the patient.

Deadline values are related to the nature of the activity and the current operation mode (m) where $\mathcal{M}$ is the set of all system operation modes (then, $m \in \mathcal{M}$). For a given activity a_i, its associated deadline value (D_i) can vary depending on the current operation mode. For example if a_i (e.g. measuring sugar levels in blood or connecting/starting a given biometric device) is triggered by the occurrence of an alarm (e.g. a patient having a heart attack), D_i will be shorter that for the same activity in a normal operation mode (e.g. the patient is in stable conditions). Therefore, $D_i = f^D(a_i, m)$ meaning that the deadline of an activity is a function of the activity and the current system operation mode.

2.1.2 Real time systems requirements

- ***Fault tolerance***. A RTS is a critical system, it has to integrate fault tolerance mechanisms where the system has to continue to provide data even some faults are present.
- ***Data accuracy***. In a RTS the arrival of responses within the specified time frame is as important as the accuracy of those responses. If data is correct but provided after time limits it's considered as wrong.
- ***Schedulability***. In order to realize a clear organization of tasks execution order and execution intervals a reliable scheduler must be deployed in the system. The scheduler policy is related to the timing requirements of tasks and the closeness to deadlines.
- ***Stability***. Systems with hard timing requirements can generate serious problems when one component is not stable this means that it provides results that are not accurate or cannot create reliable communication with sensors. Stability is essential where each component of the system must provide the intended results without any surprises.

2.1.3 RTS classification

Due to the complexity of a RTS there are many criterion used to classify them, in this section we present the essential criterion used to define a RTS.

Timing requirements

A RTS has to meet some specific timing requirements, deadlines are defined for the majority of operations executed in the system, exceeding one or more deadlines may disturb the operation of the system and as a result some responses may be worthless. RTS are classified into three types of system according to the strength of the deadlines of operations.

1. ***Hard RTS.*** The production of a response in the specified time limits is essential, it constitutes a design principle and an objective to achieve, the lack of a response to a query is catastrophic and it's more dangerous than an incorrect response or a response with a delay. The system has to meet all the defined deadlines, if the system misses one deadline it's considered as failed and the results are worthless. As example of hard RTS we cite: a car braking system, a system controlling the flow of a medicine into a patient body.

2. ***Firm RTS.*** The response in the specified time limits is important. The system can miss one deadline the results are considered as wrong but the system is not termed to have been failed. As examples of firm RTS we cite: a system controlling transactions in a stock market, a factory with production lines guided by robots.

3. ***Soft RTS.*** Many operations can be affected by a result that has exceeded it deadline, and as a result the reliability of the system is reduced. The system can miss one or more deadline if they are spaced according to the time scale and the results are not worthless, also, the system is not considered as failed. As examples of soft RTS we cite: on-line video streaming system, an Internet telephony application.

Table 2.1: Real time systems differences

	Deadline	**Reliability**	**Operations results**	**Exceeding deadline results**
Hard	Very strict	No reliability	Wrong for all	Threatening of human being life
Firm	Strict	No reliability	Wrong for the majority	Financial and health problems
Soft	Moderately strict	Reduced	Could be used	Degradation of the QoS

Tasks types

A RTS process is composed of a set of tasks executed on a single or multiple processor cores where each task has some specific timing limits. Tasks have some parameters that can de defined to classify the general operation of the system according to these tasks nature. In what follows the different parameters used to classify a RTS are explained.

1. ***Messages exchange.*** Tasks a and b are being executed on a single core processor where they exchange a set of messages $m1, m2, m3, ...$and responses $r1, r2, r3, ...$

Messages can contain data, signal, or

- Synchronous. Task b doesn't continue executing until it receives a response for each message sent to task a, a response has to be provided for a message $m1$ before proceeding to the next message.
- Asynchronous. Task b continues it execution without waiting for a response $r1$ of a message $m1$, message $m2$ can be sent without wating for $r1$.

2. ***Scheduling***. In a RTS a schedule is defined to specify execution order of tasks, this schedule is related to a time interval and deadlines for each task, it defines which task is executed in first and what is data that must be provided for other tasks to continue execution and at the end the whole process results.

   - Preemptive. Task b can stop the execution of task a on the processor even if task b didn't complete it whole execution and the results are temporary.
   - Non preemptive. Task b cannot interrupt the execution of task a, it requests the allocation of the processor but it waits until task b continues it execution.
   - Online. The scheduler execution order is defined at design time and the behavior is static.
   - Offline. The scheduler execution order is defined at run time and the behevior is dynamic.

3. ***Operation modes***.

   - Cyclic. The execution of operations is defined with a time cycle, this cycle is executed each x lap time the actions are periodic, there is no unpredictable events the scheduling is defined at design time.
   - Event driven. The execution is guided by *events* the actions are sporadic, this type of operation mode is mostly used on dynamic environments where the number of sudden events is important.
   - Cyclic and event driven. The execution is triggered each x lap time, sudden events can occur so the schedule is changed to respond to the new events operations.

2.2 Software components

2.2.1 Component definition

A component has many definitions that were used to define it characteristics, implementation and execution details. In what follows we present some of the most used definitions and that are related to this study.

1. *"A component is a reusable, self-contained, piece of software that is independent of any application."* [31]

2. *"A software component is a piece of self-contained, self-deployable computer code with well-defined functionality and can be assembled with other components through its interface."* [32]

3. *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition."* [33]

Studies like [34] and [35] aim to refine definitions of a software components and create discussions about it characteristics. From the above definitions, a component is a modular unit of a software that is independent from any programming language, a component can be executed alone or matched with other components or can be used as execution platform for an application.

Component paradigm appeared to solve same limitations faced by programming paradigms. The first programming paradigm was the *procedure oriented programming (POP)*, a program is a set of functions and procedures it has a top-down logic and a procedure can be called in any part of the program, data and functions are stored in separated memories and code reuse is described by procedure call in different parts of the program.

The disadvantages of procedural oriented programming and that are related to the real time systems programming is when dealing with a big program the reuse of procedures will be hard, another disadvantage is the complexity of a program when modelling real objects and it is not the good solution to program critical and complex systems because

it has a low level of security mechanisms because data is exposed to the whole program. As examples of programming languages, C, PASCAL, FORTRAN and COBOL.
In OOP (Object oriented programming) data and functions are stored in the same memory and code reuse is defined by inheritance between classes. The downside of OOP related to the design of RTS is that it includes more code lines comparing to the procedure oriented programming and this could generate some delays on operations responses, another disadvantage is that OOP is not suitable for critical systems development because it has a low level of security and code reuse may generate redundancy. As examples of OOP, C++, JAVA, Python and Object-Pascal.
Aspect oriented programming (AOP). A program is a set of instructions separated into functional and technical code, the functional code is exposed as *aspects* which is a special treatment that define the program functional operations, it is addressed with *jointpoints* which are points that define where the aspects are inserted. AOP aims to increase modularity by dividing the program into smaller parts so-called *concerns* this mechanism is defined as the separation of cross-cutting concerns. One of the important downside of AOP is that the separation between concerns increases the complexity of the program when dealing with complex applications also the paradigm is not well deployed on debuggers and this may cause faults on code and execution and this in not suitable for critical and complex applications. AOP languages were developed as external libraries to the existing programming languages such as AspectJ with JAVA and AspectC++ with C++.

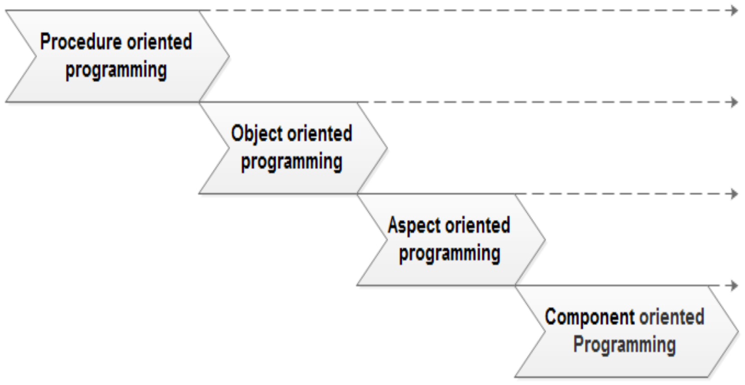
Procedure oriented programming
Object oriented programming
Aspect oriented programming
Component oriented Programming

Figure 2.2: Programming paradigms evolution

According to the previous presentation of programming paradigms the component technology is the most suitable solution to develop critical and real time systems, it modular nature and the separation between code and service through public interfaces is an advantage to design a system as a composed program where the deterioration of one component will not affect the behavior of other components and this is essential when developing RTS. Also, the encapsulation of data is a good solution to ensure an acceptable level of security. Reusability is well defined with component paradigm, a component can be used on many situations with the same code the interfaces will transfer data into the two directions.

2.2.2 Characteristics of component approach

The name *"component"* was taken from the electronic field where a set of electronic components can be gathered together in order to form a PC mother board, thus, software components are made for composition. A component has two parts, a private code that define the behavior of the component and it treatment and a public part that is the interfaces of the component where services and treatment results are connected to other components.

An interface is *provided* or *required* or can be provided and required at the same time it depends on the component model characteristics, a provided interface exports data from inside the component into outside while the required interface imports data into the component. A *port* is the connection point of interfaces it defines how the internal component exchange data with external components.

Figure 2.3 illustrates a component *Order* with two provided interfaces *OrderTracking*, *OrderManagement* and two required interfaces *Client* and *Product*. The picture defines different ways to design a component and it interfaces.

2.2.3 Software component analysis

The analysis of software component approach is very important to set it utility to design and develop critical and RTS. In what follows the good points and downside of software component are presented.

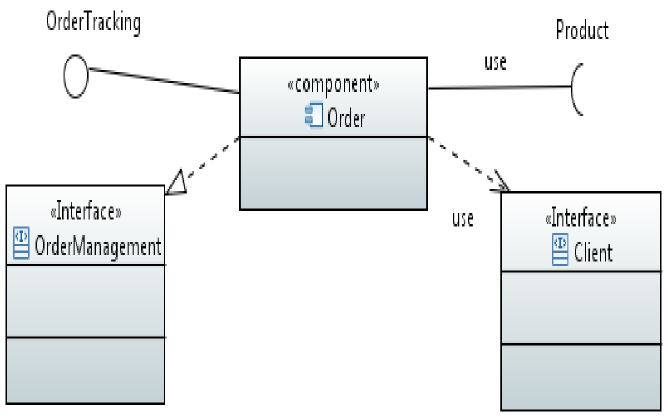

Figure 2.3: Component description with provided and required interfaces

Good points

- ***Reusability.*** Is the essential advantage that software components bring to software development. It is defined as the ability of a system to reuse pre-existing modules, codes or components without modifying the internal content. The component approach is a typical solution for reusability and this is due to the separations between internal treatment and provided results through interfaces and also the independency from any platform.

- ***Independency.*** A component is independent from any programming language, operating system or executing platform, it is related to the specification defined on the component model where the component characteristics are defined. A component can be matched with other component without knowing there internal details, interfaces are the mean to realize this matching.

- ***Encapsulation.*** Aims to hide implementation details from users where software components communicate using interfaces, it facilitates reuse by gathering related properties, functions and operations in a single unit. In component models data encapsulation is not always possible or clear because components can represent both computation and data, or just data, and data encapsulation is not considered as part of composition in general [36].

- ***Extensibility.*** When new requirements appear for an existing system only the affected components are changed in order to respond to new requirements, this saves time and guarantees a good operation level with components that are not affected by the development.

- ***Improvement of software quality.*** Reusability is a key criterion to improve software quality with a new version of the software new improvements are made and quality is raised. A component is improved in every reusability process, therefore, bugs are fixed and other requirements are added to the new version of the software.

- ***Decreases development time and costs.*** Reusability of existing software components means that fewer lines of code are written and fewer time is spent on the software development, in the other side components development time is raised. Reusing components in different projects decreases development costs where resources that are affected for developers are reduced.

Downside

- ***Difficult component integration.*** Finding a component that fulfils a set of requirements in order to be integrated in a system can be a hard task to accomplish, because a component is developed with a set of features that can not match the requirements of the new system.

- ***Component maintenance costs.*** While the software maintenance costs decreases, the component maintenance costs can increase and this due to the various versions of the software and where each one has it specific characteristics. [37]

- ***Final product quality.*** System based on the component based development is assembled from various components where each one is provided by different suppliers, even if each component has a good quality level but the final result of the composition is not known until the software is tested.

2.3 Component models

Component technology is a typical solution to develop critical systems and this due to it modular nature and communication mechanism using interfaces. Component model defines relation between components, how they interact and how they are deployed.
The use of components in critical and real time systems is due to the low cost of such solution for design and development time, the component technology is a solution to hide the complexity of such systems and to minimize response time by reusing existing components and composition operations.
The creation of a real time system pass trough the definition of the software life cycle from the requirements phase to the execution time. So if a system has been developed with timing constraints it has to integrate resource management component that adapts the behavior of the different execution platforms and the final operating device.
Schedulability, predictability, and resource management are the main properties in order to verity the behavior of a RT component model, for this real time and critical systems need to be verified in real conditions and with a real time operating system and databases.
Why not to take an existing component model for real time and critical systems and adapt it for the ICE-based systems ? It's very difficult to have a generic component model for all critical application domains: vehicular domain, medical applications, avionics, etc. and this is due to the differences in the next points:

- Specification priority: security, performance, rapidity, timeliness, data integrity, etc.
- Requirements verifying: composition, flexibility, autonomy, interoperability, etc.
- What is the most important criteria to build in the component model to verify it design: architectural, environment, and studied system differences.
- Differences in real time parameters (hard or soft).
- And the validation criteria (reliability respect with the primary expressed requirements).

2.3.1 Component model definition

The definition of a component model is essentially related to the syntax of the components, their composition strategy and how data and messages are exchanged. In what follows some definitions about component models are presented.

1. *A component model standard describes how to implement components that conform to the specific component model, this last implementation is necessary for these components to execute. [38]*

2. *A component model is the specification of the semantics, the syntax and the composition of components. [39]*

2.3.2 Examples of component models

Studies like [40] aimed to define comparative and classification solution for component models. In this section component models that are related to this study are defined and illustrated in what follows.

Rubus

The Rubus (name of a red arctic berry) component model [41] is developed by Arcticus systems and Mälardalen University for industrial resource constrained and predictable embedded real time systems (ground vehicles) with hard, soft and non real time requirements. Rubus proposes the development of real time systems with three main activities: design, analysis, and synthesis, the real time requirements and resources consumption are defined at design time and verified at run-time.
Rubus uses ports for the communication between components, the data synchronization is done at design time and the model is time and event triggered, it supports static schdulability and defines reusability by defining execution models in the infrastructure level. The run-time environment is expressed with the development of the Rubus-RTOS to support and test the component model in real time conditions. A software circuit (SWC) is a set of functions defined by it implementation and interface, the interaction

between SWCs is provided by assemblies or composites and realized via their interfaces and through ports.

UM-RTCOM

UM-RTCOM is a component model for hard and soft real time systems which supports the real time analysis at design and run time, it describes predictability and supports schedulability with the use of primitives: wait, call , and raise, the real time behavior is described using SDL-RT (Specification & Description Language - Real time). The WCET is described as annotations in the SDL meta-model and calculated on model and implementation, the RT-constraints are calculated at instantiation time.
The UM-RTCOM component model is based on generic (composed of generic and primitive components) and primitives components and supports reusability by the use of configurations slots which are a part of generic components and are public to the user. The communication between components is done via interfaces (described with IDL-CORBA) and events (wait, call, raise), UM-RTCOM describes two types of components "active" for threads and processes, and "passive" for shared resources or execution flow, the implementation of the model is based on RT-CORBA.

SaveCCM

The SaveCCM component model [42] was developed in SAVE project by Mälardalen university for hard real time systems for vehicular domain, it aims to ensure efficiency and flexibility at design time, SaveCCM deals with resources limited systems and describes timeliness as a set of quality properties to be verified and which are contained in components and assemblies. Predictability and analysability are more important than flexibility, and the timing behavior is a primordial functionality that must be verified by all architectural units.
An assembly is a set of components and switches which are a specific type of components, communication between components is made through interfaces which define a set of ports that allow the interaction between the architectural units, components composition and

binding is done at design time so it's static. SaveCCM defines a textual language and an UML 2.0 profile with some modifications to describe the architectural elements and their interconnections.

Koala

Koala [43] was developed by Phillips for resource constrained systems in consumer electronics where a component model and adequate language are proposed. The components composition is done at design time so it's static, it describes reusability and the timing properties are declared as tasks on each component, schedulability is analyzed via a task manager (declared a configuration level) that takes each component timing properties to schedule the execution.

A configuration is composed of compound components which in turn composed of basic components [44], a module is a part of the component that assembles interfaces and doesn't have interfaces, the Koala components may behave like a system so they are autonomous. All operations performed by a component with the external environment are done via the interfaces which are a set of functions described with IDL (COM and JAVA concepts), component description is done with the CDL (Component description language). A repository for Koala components is proposed and which is the *KoalaModel workspace*, reusability can be realized through *"heavy parameterized"* components, a switch defines routes connections between interfaces.

2.3.3 Comparison of component models

In this section we propose three comparative tables of the above mentioned component models and that are: Rubus, UM-RTCOM, SaveCCM, and Koala. Each table illustrates the differences between the models from a specific point of view and with several comparison parameters.

The tables 2.2 and 2.3 define a set of comparison parameters for software engineering development, these parameters are mainly used in software development. the two tables show the differences between the component models studied in this section.

	Robustness	**Reliability**	**Distributed**	**Portability**	**Autonomy**	**Data treatment**
Rubus	N/a	Yes	Yes	Yes	No	No
UM-RTCOM	No	Yes	Yes	Yes	No	No
SaveCCM	No	No	N/a	N/a	No	No
Koala	N/a	Yes	N/a	N/a	Limited	No

Table 2.2: Comparative table with software engineering parameters (1)

	Abstraction	**Flexibility**	**Interoperability**	**Domain**	**Heterogeneity**
Rubus	Yes, on configuration level	N/a	N/a	Vehicles	N/a
UM-RTCOM	Yes, encapsulation	No	N/a	Generic	N/a
SaveCCM	Yes, uses assembly concept	Limited	N/a	Vehicles	N/a
Koala	Yes	Limited	N/a	Electronics	Yes

Table 2.3: Comparative table with software engineering parameters (2)

Indexes. H: Hard, S: Soft, SWC: Software Circuit.

The tables 2.4 and 2.5 define a set of comparison and classification parameters of the component models from a real-time point of view. A system must verify the majority of these parameters in order to be qualified as real-time.

Indexes. P: Provided, R: Required, D: Data, T: Triggering, I: Input, O: Output, DT: Data and Triggering, Syn: Synchronous, Asyn: Asynchronous, Per: Periodic, Spo: Sporadic.

The tables 2.6 and 2.7 define a set of comparison and classification parameters of the component models from a software component point of view.

2.4 Use of Component technology in RTS

The use of new software paradigms is a solution to respond the emerging complexity of computer applications. Component approach is a good solution to minimize the complexity of critical systems and divide it into modules that can be also divided into smaller modules and units, software component hide the technical details of a system and provide it services through public interfaces which can be matched with other applications without

	Predictability	**WCET test**	**Error detection**	**Hard/Soft/Firm**
Rubus	N/a	Yes	Yes (timing errors)	H+S+NON RT
UM-RTCOM	Yes	Yes	Yes (deadline exceeded)	H+S
SaveCCM	N/a	Yes	No	H
Koala	N/a	N/a	No	N/a

Table 2.4: Comparative table with real time criteria (1)

	Schedulability	**Resource management**	**RT analysis**	**Data integrity**
Rubus	Yes, static	Yes, defined on SWC	Yes, design+run time	No
UM-RTCOM	Yes, static	Yes	Yes, design+run time	No
SaveCCM	Limited	Not well described	Yes	No
Koala	Limited	Yes	Yes	No

Table 2.5: Comparative table with real time criteria (2)

knowing it implementation details. Component approach ensures software reuse, whereas, this last notion is not well studied when used with RTS because of their criticality unless when used for the same application domain and more precisely in the same project.
The use of the component as a unique programming approach is not sufficient to cover all the characteristics of real-time systems because of their diversity, complexity, distribution and the weakness of this approach on the points mentioned before. Thus, integrating other programming approaches like SOA (service oriented architecture) is an important solution to realize a reliable system. [24].

2.5 Modeling languages

2.5.1 UML

UML (Unified Modeling Language) [45] is a standard adopted by OMG [46] it's the result of fusion of three modelling technologies Booch [47], OMT [48], and OOSE [49]. UML is used for all types of softwares from the most simple centralized applications to the distributed critical systems like: avionic systems, medical applications, and automotive control systems (in real development, UML has limitations when designing these type of systems). Many propositions for UML improvement has appeared like [50] it present how to use RT-UML to model web services, and [51] it present some modifications on UML to adapt it behavior to model component model for vehicular control system. Real

	Composition	**Interface**	**Ports**	**Reusability**
Rubus	Yes	Yes	D(I/O)+T(I/O)	Yes
UM-RTCOM	Yes	Yes(P/R)	No	Yes
SaveCCM	Yes, static	No	I(D/T/DT)+O(D/T/DT)	Yes
Koala	Yes	Yes(P/R)	No	Yes

Table 2.6: Comparative table with software component criteria (1)

	Connection	**Behavior**	**Run-time**	**Interface desc**
Rubus	Syn/Asyn	Per/Spo	Rubus-RTOS	N/a
UM-RTCOM	Syn/Asyn	Per/Spo	RT-CORBA	Corba
SaveCCM	Syn	Per/Spo	N/a	SaveCCM
Koala	N/a	Spo	N/a	IDL(JAVA+COM)

Table 2.7: Comparative table with software component criteria (2)

time UML is an UML profile that looks to model systems having timing limitations and deals with strict criteria special to this type of systems to be modelled at design time and verified at run time like: schedulability and timing properties/requirements.

2.5.2 SysML

SysML (Systems Modeling Language) [52] is an OMG [46] standard and the result of collaboration of OMG, INCOSE [53] and AP233 [54]. Component diagram is not present on SysML, the internal block diagram or model can be used to model the system as set of parts connected via provided and required interfaces that define input/output ports, parts describe blocks and communicate using data and messages, UML component diagram has no analogue in SysML. This last doesn't include special concepts and descriptions of real time requirements of a complex system they are not prioritized.

2.5.3 MARTE

MARTE [55] is an UML 2.0 profile for Modelling and Analysis of Real Time and Embedded systems, it's an OMG standard to model driven solution that covers the system specification, design, verification and validation. The model is the core conceptual unit of MARTE diagrams, MARTE is divided into four packages composed of sub-profiles each on of them is a set of requirements or properties or concepts that define the basics of

MARTE design.

2.5.4 Comparison of modeling languages

In this section we propose a comparative table of the modelling languages mentioned in the previous section, we aim to identify the main differences.

	Systems type	**Basic design unit**	**Component**	**Model**	**RT analysis**
UML	Generic	Class/object	Yes	Hyper-diagram	No
SysML	Complex	Block/unit	No	diagram	Not prioritized
MARTE	RTE	Model	Yes	package	prioritized

Table 2.8: Modeling languages comparative table

2.6 Medical devices

Medical devices are technology-related units which make them always under study and development following advances in electronics and informatics and requirements expressed by different stakeholders.

Usually the development of a device is a process that begins with a creative idea that is translated into a prototype invention and then it is continually improved over time [19], rarely his process include the study of interoperability of the device and the possibility of it integration with other devices except for devices from the same manufacturer.

2.6.1 Definition

GHTF (Global Harmonization Task Force) [56], FDA (Food and Drug Administration) [57], and WHO (World Health Organization) [58] have the same definition of a medical device and they define it as any instrument, apparatus, implement, machine, appliance, implant, in vitro reagent or calibrator, software, material or other similar or related article that does not achieve its primary intended action in or on the human body solely by pharmacological, immunological or metabolic means and that is intended for human beings for:

- The diagnosis, prevention, monitoring, treatment or alleviation of disease;

Figure 2.4: Operating Room

- The diagnosis, monitoring, treatment, alleviation of, or compensation for an injury;
- The investigation, replacement, modification, or support of the anatomy or of a physiological process;
- Supporting or sustaining life;
- Controlling conception;
- Disinfecting medical devices;
- Providing information for medical or diagnostic purposes by means of in vitro examination of specimens derived from the human body.

2.6.2 Medical devices use

Medical devices can be used in different clinical situations and this depends on their intended use. We distinguish medical devices for general use like a stethoscope or a thermometer, disease-specific medical devices [59] can also be classified to the organ that they meant to be used with, like dental devices, cardiovascular devices, ear, nose and throat devices and so on.

Patient telemonitoring is one of the most common practices in telemedicine in both indoor and outdoor scenarios, and it is hoped that it can increase the quality of the care and the efficiency of services provided. In fact, it should facilitate a continuous or event monitoring of chronic, elderly, under palliative care or have undergone surgery, without them occupying the beds that would be necessary for monitoring (leaving the beds for the use of patients in a more critical condition).

In addition, telemonitored patients can continue to live in their own homes with the subsequent advantages: comfort, more favourable environment, less need for trips to the hospital, etc. Telemonitoring, used appropriately, is expected to decrease healthcare costs [60]. Two barriers to the current expansion of telemonitoring services, both related to interoperability, can be identified and, in our opinion, they make the transition from pilot experiences to clinical use very difficult: 1) Heterogeneity of devices and systems, and 2) difficulty of integration with healthcare information systems used routinely by healthcare professionals [60].

The devices used most frequently in telemedicine applications to measure parameters and biological signals are glucose meters, blood pressure and heart rate meters, pulse oximeters, ECG monitors, digital scales, etc. (see Figure 1). The devices can be fixed, but it is increasingly common for them to be wireless or wearable (with sensors incorporated into clothing, bracelets, etc.), that makes their use more comfortable. These collections of sensors around the patient make up what can be usually described as either a Body Area Network (BAN) or Personal Area Network (PAN). Often, for monitoring elderly patients or those with limited mobility, these PAN or BAN networks are completed with presence detectors, movement sensors, or similar 'Telecare' devices, which combine to form a Home Area Network (HAN) [60].

2.6.3 Concepts related to medical devices

Safety

Safety is defined as freedom from unnecessary risk, where risks are unmitigated hazards [61], it's a property of systems that arises from the interactions between system com-

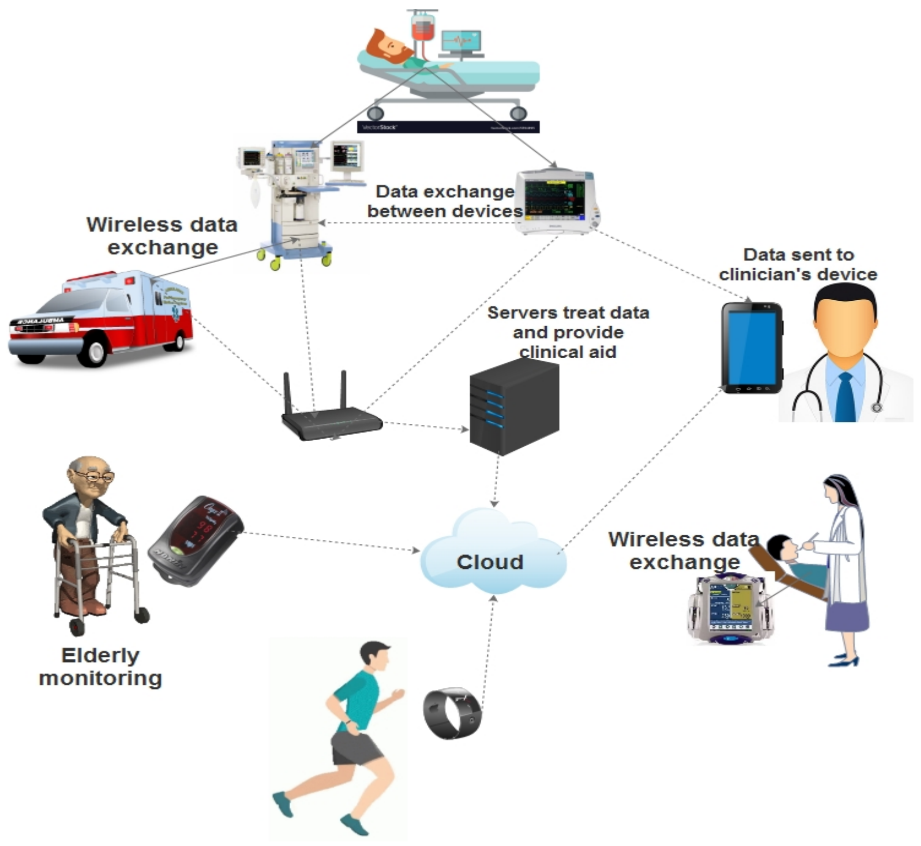
Data exchange between devices
Wireless data exchange
Data sent to clinician's device
Servers treat data and provide clinical aid
Cloud
Wireless data exchange
Elderly monitoring

Figure 2.5: Medical devices use

ponents [62] to constitutes safe inter-device interactions will vary considerably between different clinical scenarios. Safety is critical to ensure that interactions between devices are predictable and that those interactions satisfy the safety requirements of the given clinical scenario.

Medical devices must perform safely and effectively [63], therefore, safety is extremely important when patients' lives are at stake. Since it will not be known specifically which devices will be assembled into the composite system a priori, traditional methods of assessing safety cannot be used [64]. What makes medical devices particularly challenging is that unlike traditional information technology (IT) systems, they are almost exclusively safety- critical systems.

Security

Incorporating security from the very beginning of the design process starts from understanding the threats or potential attacks that the devices face, the implication of such attacks, and defining appropriate responses and mitigations for the attacks, which then are turned into security requirements [65] wireless interfaces of medical devices expose the device to security threats and raises security and privacy concerns for the patient [66]. Researchers have demonstrated that these devices and their wireless interface are vulnerable to cyber attacks that can compromise patient safety, security, and privacy [67]. Various types of attacks on medical devices wireless interface include privacy leakage by eavesdropping, safety and integrity errors by message alteration, and availability issues by battery draining attacks have been demonstrated by researchers [68] However, the introduction of interoperability makes medical devices increasingly connected and dependent on each other.

Therefore, security attacks on IMDs becomes easier to mount in a stealthy manner with potentially devastating consequences [69]. Security of MDs is different and more challenging than regular IT security [70]. In the past, MDs have been success- fully hacked on several occasions. For example, unauthorized commands have been sent wirelessly to an insulin pump [71]. Another insecurity scenario of MDs has emerged in drug pumps that became exposed to a series of remotely exploitable vulnerabilities [72].

Plug and Play

The effort required by a single manufacturer to enable safe end-user integration or connection of two of their own proprietary products is not substantially different from the effort that would be required by two different manufacturers to create plug-and-play interoperable devices, but it does require the cooperation and some level of consistency between medical device manufacturers, SDOs, clinician SMEs, regulatory bodies, and HDO customers.

Plug-and-play interoperability has significant challenges that will not be achieved in the short-term for all systems, but that does not reduce the importance of implementing open, interoperable interfaces with well-defined requirements and clinical purposes that are easy to configure and troubleshoot to enable the ecosystem to transition from closed, proprietary interfaces to plug-and-play.

Device model

The major challenge of getting devices to interoperate is not simply getting the devices to communicate, but getting them to communicate in a meaningful way. An analogy with oral human communication is instructive. Assuming that two people have functioning vocal and hearing apparatus, it is not difficult to see that one could talk and the other could hear. However, if they speak different languages, the communication cannot be meaningful.
Just as importantly, some degree of shared experience and culture is necessary for one person to communicate effectively with another. In addition, the specific environment or context of the communication, as well as a specific representation or model of the parties participating in a conversation, is indispensable not only to meaningful communication but also to confidence that meaningful communication has occurred. Although devices do not have cultures or the richness or expressiveness of human natural language, there are analogues in device communication.

Hazards

A medical device hazard is any potential source of harm [73] . These hazards may affect the patient or the clinician and are related to device risk level and therefore device risk class defined by the FDA [57]. Hazards may be caused by software (Software bug or bad update), hardware (electrodes bad insertion), electric (high voltage), mechanical, environmental (temperature or humidity), incompatibility of devices, wrong information (wrong display of heart beat rate), misuse by the clinician (a leading cause of device incidents reported in the healthcare domain [74]) and that are classified under ISO 62366 [75], radiations (an overdose of insulin from infusion pumps exposed to EMF (Electromagnetic fields) from mobile phones or RFID (radio frequency identification devices) [76]). Hazards may cause device malfunction or even explosion or fire and therefore patient death or injuries, and to avoid that a risk management policy has to be established.

Intended use

Devices are evaluated based on their risk and stated intended use [77]. A medical device has a pre-defined intended use and that varies depending on it hardware and software characteristics, for example, a glucose meter cannot be used to measure heart beat rate because it hasn't the capability to do that. A medical device may also be used in a specific clinical situation, for example two heart beat meters may have the same functionalities but one is intended to be used in a situation where radiations are present.

Medical device intended use is generally mentioned on the labelling and the technical and indications for use manuals, stated words are very important and must be clear without ambiguity to specify the device intended use and it risk class, if you mention that your device is a heart rate monitor it means it is class I device and if you change that to an implantable heart rate monitor it becomes class III.

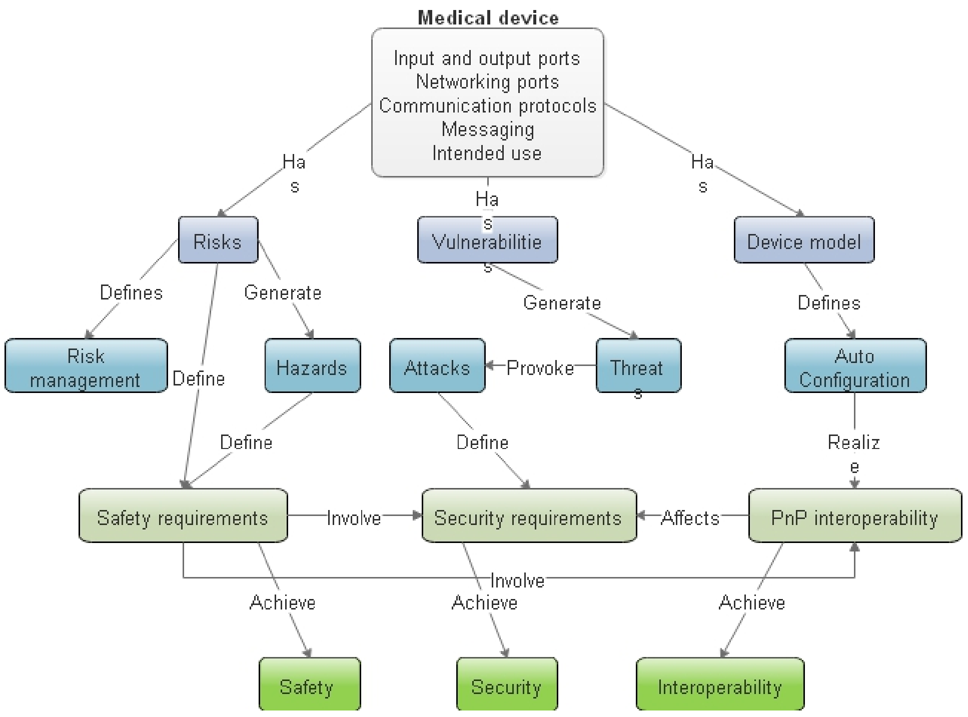

Figure 2.6: Concepts related to medical devices and links between them.

Chapter 3

Interoperability software

In this section we present a set of interoperability software that are used to communicate and transport data in distributed systems, for this we illustrate their definition, characteristics, operating mode, etc.

3.1 Definition

The word *interoperability* is defined according to technology or the context where it's used. Interoperability between two systems is not the same as between two protocols interacting at the application level. In the medical domain and especially The data exchange between medical devices interoperability was defined by academia

3.2 Interoperability architectures

3.2.1 Point-to-point interoperability

In a point-to-point architecture, every system is connected to every other system. This has the advantage of having no central bottleneck or single point of failure. When used in the context of linking one system to a single other system, a point-to- point architecture is efficient and simple to implement. The major disadvantage of this technique when integrating several systems is the number of distinct interfaces needed to integrate even a small number of different systems. Without significant standardization, each system

Table 3.1: Interoperability definitions

Organization	Definition	Medical devices
IEEE[78]/**HL7**[79]/ **ISO**[80]	The ability of two or more systems or components to exchange information and to use the information that has been exchanged.	**Included**
AAMI [81]	Medical device interoperability is the ability of medical devices, clinical systems, or their components to communicate with each other in order to safely fulfil an intended purpose.	**Included**
HIMSS[82]	Interoperability means the ability of health information systems to work together within and across organizational boundaries in order to advance the effective delivery of healthcare for individuals and communities.	**Included**
CEN[83]	Interoperability is the ability of one entity to accept data from another entity and perform a specific task in an appropriate and satisfactory manner without the need for extra operator intervention.	**No**
ASTM[84]	The ability of any system or product to work with other systems or products without special efforts.	**Included**
IHE[85]	IT systems are interoperable if they can properly exchange clearly defined sets of relevant information in the context of a specific clinical situation and perform appropriate actions as described by the IHE specifications.	**No**
IEC[86]	The capability to communicate, execute programs or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.	**No**
FDA[57]	The ability of two or more products, technologies or systems to transmit and/or receive information without special effort or configuration required by users.	**Included**

potentially needs an adapter to talk to every other system and each is likely to need some knowledge of the other systems included in the integration. It is also more difficult to implement any kind of centralized activity monitoring in this kind of environment.

3.2.2 Hub and spoke interoperability

A hub and spoke architecture reduces the number of connections needed to link several systems by linking them all through a central hub. Each system to be integrated needs a single adaptor to connect it to the hub and enable it to connect with every other system connected to the hub. One major disadvantage of the hub and spoke architecture is that the hub is a single point of failure. The hub itself handles the processing and routing of data passed to it by connected systems. The hub becomes complicated as the number of connected systems increases and needs to have knowledge of every system. When a new system is to be added, the hub needs to be modified to support it. However, a major advantage over the point-to-point architecture is that, if a system needs to be removed or replaced, it can be done so with no effect on other systems connected to the hub.

Examples of a hub in a hub and spoke architecture are an integration engine or a shared data repository.

3.2.3 Hybrid models

In practice, a combination of point-to-point and hub and spoke models are often used across local infrastructures due to information systems not being fully flexible and interoperable, and to ensure that communications are as efficient as possible. The requirements for interoperability are obviously avoided if all of the relevant care professionals are able to work within common information systems with common databases. With emerging delivery models to support person-centred treatment and care, this is unlikely to be the case.

3.3 Interoperability dimensions

3.3.1 Data interoperability

Data format interoperability is the ability to exchange information between applications with semantic meaning [21]. Do two systems mutually understand what characters cause terminations or truncations? Do they share the same formatting of dates and times? Do they format medical record numbers as a string or as a numeric and, within a single system, is there consistency in the length of these numbers? Do two systems or applications apply the same default and/or permitted values for individual field elements? Patient safety scenario: A (new) automated microbiology system performs and sends results on 30 lab tests to the laboratory information system (LIS). The (old) LIS is configured to expect/accept only 20 results. Data on items 21–30 are stored to unallocated memory, which cannot be retrieved and reported out to the clinician or patient's EHR [24].

3.3.2 Communication interoperability

Definition: Consistency in transmission and reception of messages between nodes Examples: Does a hospital information system's computerized physician order entry (CPOE)

module share the same format for addresses with its radiology information system (RIS) so that the CPOE knows where to send a message for an order? Do these systems have the same rules for acknowledging that a message has been received completely? Do the systems have the same rules for timeouts and retransmission if a receipt has not been detected? At the message structure level, do the systems share an understanding of when the message begins and ends? Do the systems have a mutually accepted understanding of when a message has errors and when it should be rejected?
Patient safety scenario: A clinician sends an order message over the CPOE to the pharmacy requesting a "STAT" dose of medication for a patient. The pharmacy information system (Rx system) does not receive the message due to a transient network issue. The standard calls for the CPOE system to wait for an acknowledgment (an HL7 ACK message), and display a message to the user that the order transaction was not completed, but this does not happen. The clinician is not aware that the order has never been received, and there is a delay in treating the patient [24].

3.3.3 Semantic interoperability

Definition: Agreement/consistency between systems on the meaning of communicated information. Semantic interoperability requires clear interfaces based on published standards. Systems without the ability to exchange information do not require interfaces. In the middle are systems with non-traditional or proprietary technology that can exchange information, but not based on any accepted standard. Several examples of accepted standards to create modular architecture include the Admission and Patient Discharge (ADT) messages or the Continuity of Care Document (CCD) standard for EHRs as specified by HL7 [23].

3.3.4 Workflow interoperability

Definition: Agreement/consistency on how technology supports/shapes the workflow: • Processing or sequencing tasks between participants according to a set of procedural rules • Formatting or displaying information • User interfaces • Penetration of decision support

Examples: • Semi-automation of workflow involving Indium-labelled leukocyte scintigraphy Processes and information need to be coordinated and passed across CPOE, RIS for scheduling and Modality Work List generation, LIS phlebotomy scheduling, isotope inventory management, ordering, and so on. • Semi-automation of workflow around antimicrobial approval-All systems/ subsystems (CPOE, LIS, telecom/pager) must have a shared model of the work process and the rules/conditions necessary before passing information/status to the next system and step in the process. Patient safety scenario: A clinician requests that

3.4 Protocols

Recent survey [87] has shown that out of 92 health sensors, 86 implement proprietary protocols. Communication protocols can be categorized into two groups. The first one is standard protocol which aims to be used by different manufacturers to produce interoperable devices. In personal healthcare IoT, it is ISO/IEEE 11073 also known as Continua. The second group is proprietary protocol which is developed by a manufacturer for his own products. Confusion may occur between interoperability and standards. As a matter of fact, standards ensure interoperability by default as they can be used by everyone. Regarding proprietary protocols, they can be interoperable once the manufacturer releases necessary tools to ensure the communication with its devices.

For instance, HomeKit or Z-wave are proprietary protocols for home automation. However, lots of manufacturers implement them into their products to ensure interoperability between different services. Implementing standard protocol is very advantageous for final users. They have the ability of buying the sensor they want without having to worry if it is going to work or not with the rest of the used solution. For manufacturers, it spares them the effort of developing a new communication protocol. However, the number of certified devices keeps decreasing since 2009 [88] whereas the number of IoT is growing. This expresses the manufacturers will to implement their own protocols in their products.

3.4.1 MQTT

MQTT (Message Queuing Telemetry Transport) is a simple M2M communication standard developed by International Business Machines Corporation, offers a scalable architecture that can be implemented on limited hardware as well as on unreliable networks [89]. MQTT allows subscribers to indicate interest in topics, and then be notified of updates automatically, preventing inefficient polling schemes. MQTT implementations are available under the liberal Berkeley Software Distribution license, which makes royalty-free commercial usage possible.

Due to the adoption of MQTT by the Eclipse Foundation as the choice for a general purpose open source M2M messaging protocol [17], the popularity of MQTT already exceeds DDS 20 fold in Google Trends, while DDS remains of interest mainly in niche military and similar complex industrial applications. In this paper we therefore consider the use of MQTT for medical device communication. The MQTT protocol is well suited for this purpose as it is scalable, unencumbered by commercial licensing restrictions or requirements, and a good fit for the ASTM interoperability standard [1]. Indeed our approach retraces several elements of the standard.

3.4.2 DDS

DDS (Data Distribution Service) has previously been considered for medical device integration in the Medical Device "Plug-and-Play" Interoperability Program at Massachusetts General Hospital [90]. DDS is a mature protocol developed for military applications, and is certainly a candidate for this type of use. However, the available DDS implementations have commercial license restrictions, and the complexity of the protocol makes it unsuited for deployment on most embedded devices [91].

3.4.3 ZigBee

The ZigBee Alliance is an association of companies working together to enable reliable, cost-effective, low- power, wirelessly networked, monitoring and control products based on an open global standard. The goal of the ZigBee Alliance is to provide the consumer with ultimate flexibility, mobility, and ease of use by building wireless intelligence and capabilities into everyday devices. ZigBee technology will be embedded in a wide range of products and applications across consumer, commercial, industrial and government markets worldwide. For the first time, companies will have a standards-based wireless platform optimized for the unique needs of remote monitoring and control applications, including simplicity, reliability, low-cost and low-power.

3.4.4 DASH7

The DASH7 Alliance was formed to advance the use of DASH7 wireless data technology by developing extensions the ISO 18000-7 standard, ensuring interoperability among devices, and educating the market about DASH7 technology. Formed in 2009, the Alliance now has more than 20 members. Manufacturers, systems integrators, developers, regulators, academia, and end-users all work together to promote the use of DASH7 technology in a wide array of industries and applications. The DASH7 Alliance provides a framework for extensive application development, seamless interoperability, and security for DASH7-enabled transactions. Goals: * Develop improvements and extensions to the ISO 18000-7 standard * Work to ensure that products claiming DASH7 capabilities comply with DASH7 specifications *Encourage the development of products using DASH7 technology * Educate consumers and enterprises globally about DASH7 technology.

3.4.5 Bluetooth

The Bluetooth Special Interest Group (SIG) is a privately held, not-for-profit trade association founded in September 1998. The Bluetooth SIG itself does not make, manufacture, or sell Bluetooth enabled products. The SIG member companies are leaders in the telecommunications, computing, automotive, music, apparel, industrial automation, and network industries. SIG members drive development of Bluetooth wireless technology, and implement and market the technology in their products. The main tasks for the Bluetooth SIG are to publish Bluetooth specifications, administer the qualification program, protect the Bluetooth trademarks and evangelize Bluetooth wireless technology.

3.4.6 Wifi

The Wi-Fi Alliance is a global non-profit industry association of hundreds of leading companies devoted to the proliferation of Wi-Fi technology across devices and market segments. With technology development, market building, and regulatory programs, the Wi-Fi Alliance has enabled widespread adoption of Wi-Fi worldwide. The Wi-Fi CERTIFIED program was launched in March 2000. It provides a widely-recognized designation of interoperability and quality, and it helps to ensure that Wi-Fi enabled products deliver the best user experience. The Wi-Fi Alliance has completed more than 8,000 product certifications to date, encouraging the expanded use of Wi-Fi products and services in new and established markets.

3.4.7 RMI

RMI (Remote Method Invocation) is a JAVA API (Application Programming Interface), and as it name indicate it's used to call remote methods, [92] 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, it's 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 implement the client/server technology.
RMI was proposed with the JAVA release 1.1 in March 1997 and it runs on the port 1099, It uses two transport protocols to transfer data on the network and which are: RMI-JRMP (Java Remote Method Protocol) it's the original version of the transfer protocol, it's language independent and simple to use comparing with the RMI-IIOP (Internet InterORB(Object Request Broker)Protocol) which is the new version of the transfer protocol started to be used from JAVA 2 release 1.3 and it's compatible with CORBA [93].

3.5 Interoperability standards

3.5.1 ICE (The Integrated Clinical Environment) [1]

ICE was charactered by ASTM [84] and promoted by other organizations like MDPNP [94]. ICE is a real-time safety-critical system [95], it follows the concept of a clinical work-place service-oriented medical device architecture and uses DDS [96] as a communication technology and has an open source framework called OpenICE [97]. ICE aim is to realize a PnP network of medical devices for each clinical situation in a safe way taking into account issues like device functional alarms [98], security, workflow and data management. ICE manages data and devices in a centralized way through medical applications [99] which facilitates the task for clinician, in order to build an ICE-based system, a set of requirements have to be respected and that are addressed in ICE requirements documents [100] [101] [102]. Interoperability in ICE is being promoted by many manufacturers and is considered as the most successful standard in medical device interoperability and this is due to the use of syntactic and semantic data exchange mechanisms [69].

3.5.2 HL7 (Health Level 7)

Is defined as a messaging protocol for exchanging healthcare information such as patient demographics, prescribed drugs, or lab tests, it uses JSON, http-based RESTful protocol with predictable URLs. HL7 version 2 [103] is designed to support data messaging in a centralized or distributed environment and propose interfaces communication to stake-holders and suffers from interoperability problems, HL7 version 3 [104] uses XML, web services and promotes semantic interoperability. This last is lacking between version 2 and 3 [23].

3.5.3 DICOM (Digital Imaging and Communications in Medicine) [2]

Was originally created to store and transmit radiographic images (ACR-NEMA [83]), after that DICOM-WS 30 was introduced to support the storage of raw medical waveforms including blood pressure, ECG (supports 12-lead ECG recordings) [105], temperature and audio [106]. DICOM [2] is a file format and an images exchange standard to produce, display, print, store, or transfer medical images among scanners, servers, workstations, printers or any device supporting DICOM file format, it uses binary format to specify data and OO principals to design it, images may be related to radiology, cardiology, pathology and dentistry, DICOM (ISO 12052:2006) is considered as the most successful standard in the medical world today [23].

3.5.4 CDA (Clinical Document Architecture) [3]

Is a part of HL7 version 3 series of standards for presenting patient's healthcare information as clinical documents [23]. It uses XML to define the clinical document structure, parameters and semantics and designed to transfer documents and not data points or lists. CDA is independent from any underlying protocol, so, it promotes interoperability (CDA version 3) among systems and can be used instead of messages to transfer data.

3.5.5 IEEE 11073 [4]

A complement of the IEEE 1073 standard series [107] to establish functional interoperability of point-of-care medical devices, the adoption of IEEE 11073 is limited because of the complexity of lower layers of this protocol [108]. IEEE 11073 is divided into three layers: device data/semantics, communication services and transports and defines standards for each layer like communication model (IEEE 11073-20101), transport profiles (IEEE 11073-30xxx) series, domain information & service model (IEEE 11073-10207) and optimized exchange protocol (IEEE 11073-20601a).

3.5.6 MDPWS (Medical Devices Profile for Web Service) [5]

Uses SOA paradigm for embedded systems, it brings SOA principals for medical devices communications including data encoding (XML), messaging (Simple Object Access Protocol (SOAP)) and devices discovery (Universal Description, Discovery, and Integration (UDDI)) and description (Web Services Description Language (WSDL)), events subscribing end triggering. DPWS [109] does not support safe communication between medical devices , thus, MDPWS was introduced to satisfy safety requirements in the mentioned context [110].

3.5.7 FHIR (Fast Healthcare Interoperability Resources) [6].

A standard developed by HL7 for exchanging healthcare information electronically, FHIR combines HL7 versions 2 and 3 and CDA in order to override incompatibility between HL7 version 2 and 3. It uses XML and JSON to describe structures and http/RESTful to define URLs, the benefit of FHIR comparing with CDA is that it transfers health information faster because it sends specific items of information rather than the whole document like CDA does.

3.5.8 OR.NET [7]

A standard based on SOA and web services [26], it's classified under IEEE 11073 series of standards as IEEE 11073-20701, and considered as one the most important standards [110], it has as implementation openSDC (open Smart Device Connect) which is developed by Dräger and several other manufacturers.

3.5.9 SNOMED (Systematized Nomenclature of Medicine) [8].

Is a collection of medical terms that provide codes and synonyms, it uses clinical documentation and reporting. The aim of SNOMED [8] is to provide a comprehensive, multi-axial and controlled terminology [111] for indexing an entire medical record.

Table 3.2: Interoperability standards

Standards	Interoperability				Type		Security	Safety	How connected	Scope	Adoption	Paradigm
	Technical	Syntactic	Semantic	Social	Open	Proprietary						
ICE	✓	✓	✓		✓		✓	✓	Centralized	Networking, alarms, data, workflow	New	SOA
HL7	✓	✓	✓	✓	✓		✓		Centralized Distributed	Data, messaging, information ontology	Wide	
DICOM	✓	✓			✓					Medical images, waveform,	Wide	OO
CDA	✓		✓		✓					Information models for clinical documents	Medium	
IEEE 11073	✓	✓	✓			✓	✓	✓	Distributed	Data, semantics, Communication, alarms waveform	Limited	SOA
MDPWS	✓	✓			✓		✓	✓	Distributed	Data, messaging, events	Limited	SOA
FHIR		✓		✓	✓		✓		Distributed	Health Information and document model	Medium	WS
OR.NET	✓	✓	✓		✓			✓	Distributed	Data, communications, networking	Medium	SOA and WS
SNOMED		✓	✓	✓	✓			✓		Medical records	Wide	

3.6 Interoperability standards development organisations

Health information interoperability standards are developed by a wide variety of healthcare organizations including regulators, vendors, consultants and healthcare providers. Most often, the development of interoperability standards involves technical committees that define methods and groups representing communities of interest.

3.6.1 HL7

HL7 [79] provides a framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery, and evaluation of health services. HL7 and its members provide a framework (and related standards) for the exchange, integration, sharing, and retrieval of electronic health information.

3.6.2 IHE

IHE (Integrating the Health Enterprise (IHE) [85]. Profiles, which are not standards but provide a common language for purchasers and vendors to use when acquiring or upgrading systems to specify a level of compliance to standards sufficient to achieve efficient interoperability [24].

3.6.3 ISO

ISO [80] is an international standards development and accreditation organisation with a network of national standards institutes in 157 countries. The ISO technical committee, ISO/TC215, was established for the area of health informatics with the scope of: 1. standardisation in the field of information for health, and health information and communications technology to achieve compatibility and interoperability between independent systems 2. ensuring compatibility of data for comparative statistical purposes, (i.e. classifications) to reduce duplication of effort and redundancies. ISO/TC215 collaborates with a number of other SDOs including Comité Européen de Normalisation (CEN) and HL7 and has a membership of over 20 participating countries involved in developing health information and interoperability standards. Standards developed by working groups within ISO/TC215 receive ISO accreditation, thereby ensuring a strong likelihood of international adoption.

3.6.4 CEN

The European Committee for Standardization CEN, is involved in developing multidisciplinary standards including standards for healthcare systems and interoperability. It is a private non-profit organization whose mission is to foster the European economy in global trading, the welfare of European citizens and the environment by providing an efficient infrastructure to interested parties for the development, maintenance and distribution of coherent sets of standards and specifications. TC 251 is the health informatics technical committee in CEN with responsibility for publishing standards addressing aspects of health information representation including messaging, electronic health records and eHealth initiatives. The Committee is also responsible for addressing the European Commission's health interoperability mandate known as Mandate 403. (21) CEN membership consists of most European countries, including Ireland.

3.6.5 OpenEHR

The OpenEHR Foundation (www.openehr.org) is a not-for-profit organisation established in 1999, by Ocean Informatics, Australia and the Centre for Health Information and Multi-professional Education (CHIME) department in University College London, UK. OpenEHR is a virtual community working on interoperability and computability in eHealth. Its mission is to enable semantic interoperability of health information, within and between EHR systems. The openEHR Foundation has published a set of specifications defining a health information reference model, a language for building *clinical models*, or archetypes, which are separate from the software, and a query language. The architecture is designed to make use of external health terminologies, such as SNOMED CT, LOINC and ICD. Its main focus is EHRs and clinical information systems.

3.6.6 LOINC

Initiated in 1994, at the Regenstrief Institute, the Logical Observation Identifiers Names and Codes (LOINC) committee was organised to develop a common terminology for laboratory and clinical observations, to support the growing trend of sending clinical data electronically. Most laboratories and clinical services use HL7 to send their results electronically from their reporting systems to their care systems. However, the tests in these messages are identified by means of their internal, idiosyncratic code values. As a result, receiving care systems cannot fully "understand" and properly file the results they receive unless they either adopt the producer's test codes (which is impossible if they receive results from multiple sources), or invest in the work to map each result producer's code system to their internal code system.

3.6.7 PCHA

The Personal Connected Health Alliance (PCHA), formerly Continua Health Alliance, advocates for global technology standards to enable interoperable solutions for personal connected health. It publishes the Continua Design Guidelines, which provide a flexible implementation framework for 'plug-and-play' or seamless interoperability of personal connected health devices and systems. It is developed by members from technology, medical device, health care industry, consumer electronics, healthcare service and life science companies as well as government agencies. The Continua EHR, PHR, design Guidelines are based on open standards developed by recognized industry groups and standards development organizations. They define interface and standards at those interfaces along with constraints within those standards that enable the secure flow of medical data among sensors, gateways and end services.

3.6.8 Continua Health Alliance

The Continua Health Alliance is a non-profit industry coalition started in 2006. It has more than 180 member companies today. Continua's vision is to build a "system of interoperable personal health solutions." This is done by certifying and branding Continua-enabled products. If a product has the Continua logo on it, then it is certified to work or be interoperable with any other Continua-branded product. Certification comes with rigorous independent testing to the selected Continua standards. The main thrust of Continua currently is the personal telehealth arena, which includes disease management, health and wellness, and aging independently. From a design standpoint, the members' products tend to be small and portable, which drives the engineering toward smaller messaging to keep battery life as long as possible. Additionally, the personal area network (PAN) emphasis for the devices is on wireless protocols that also need to be energy or battery efficient.

3.6.9 MDPNP

• AAMI: systems-level standards, use cases, patient safety, V&V methodology, and technical education. AAMI could provide guidance to other SDOs on important clinical scenarios and promote the testing and improvement of clinical performance in composite devices. • ASTM: high-acuity clinical use case development; requirements and architectural development for systems to serve safety interlock, closed-loop control, and related needs. ASTM F2761 architecture provides for the application of systems engineering best practices, compartmentalizes functions that enable key regulatory pathways, and provides a framework and technical environment for third-party clinical "app" development. • HL7: enterprise integration, messaging and clinical documents. • IEEE: device communication, nomenclature, and device models. • IHE: profiling, enterprise connectivity, demonstration, and testing frameworks and tools. • ISO TC 215: support of selection, continued development, and international propagation of ISO/IEEE 11073 standards. • UL: testing and certification. • DICOM/ISO 12052 and NEMA/MITA: image and clinical information interchange for image and radiotherapy. Interoperability for medical imaging and radiotherapy. • West Wireless: Validation of new technologies using clinical and economic methods demonstrating safety, effectiveness and cost savings, advocacy, and promotion of innovation—particularly for wireless applications

3.6.10 IEC

International Electrotechnical Commission (IEC) 80001. Application of risk management for IT networks incorporating medical devices, which defines the roles, responsibilities, and activities necessary for risk management for IT networks incorporating medical devices to address safety, effectiveness, and data and system security [24].

3.6.11 AAMI

AAMI standards would be based on specific clinical functional requirements (use cases) and on non-functional requirements such as QoS, quality of measurement (precision and accuracy), and, most importantly, safety. Moreover, they would be based on a complete systems-level approach, follow systems engineering best practice, and incorporate standards for user interaction, patient interface, and device-to-device interoperability. The primary emphasis would be on ensuring that the resulting composite device can operate safely and effectively for a specific intended use. For example, if AAMI were to develop a standard to address the PCA clinical scenario (see Section 1.8), such a standard would describe the essential performance requirements for such a composite system: performance, QoS, and interoperability for each part of a composite solution. Standards like these might be recognized by regulatory bodies (e.g., the U.S. FDA), creating a uniform approach for ensuring the safety of composite systems.

3.6.12 HITSP

HITSP was chartered by the U.S. Department of Health and Human Services to provide standards harmonization in the health information technology arena to meet the federal mandate for a universal electronic medical record. HITSP bases each of it interoperability specifications (IS) on use cases as promulgated by the National e-Health Collaborative

(NeHC), formerly known as American Health Information Community (AHIC). The interoperability specifications are meant to be compulsory on any players in the healthcare realm who are contributing to an electronic medical record for a US citizen. Of particular interest to the clinical engineering community is HITSP Remote Monitoring Interoperability Specification (RMON).

3.6.13 NIST (National Institute of Standards and Technology) project

Medical devices have to coordinate their actions in order to realize “closed loop” scenarios and automate clinical workflows [112]. The U.S. National Institute of Standards and Technology projects for advancing medical device communications include: • collaboration with IHE on test tooling for conformance of HL7 messages to IHE PCD domain profiles; and • collaboration with IEEE 11073 committees, the IHE PCD Device Point-of-Care Integration Work Group, and the Continua Health Alliance on software tools for constructing models of device capabilities and on conformance test tooling.

3.7 Interoperability benefits

Adopting a medical devices standard will have many benefits, including :

- Adher to security standards to avoid patient data hacking or misuse.
- Improves the quality of healthcare by using data generated from many devices for analysis and diagnosis.
- Facilitating data exchange among medical devices and systems.
- Coordinating medical devices actions to work in complex clinical situations.
- Avoid hazards and related.
- Provide databases and rules on medical devices that when used in the same clinical situation can hurt the patient.

- Avoid data repetition or multiple measurements of the same vital sign from different devices.
- Support and analyse fauls alarms and record any bad incident in the system.

Benefits of interoperability is will bind together a wide network of real-time, life-critical data that not only transform but become health care [113]. a study made by the U.S. Department of Commerce Technology Administration in 2004 estimates a cost of U$15.8 billion related to the inadequate interoperability between systems in the U.S. Capital Facilities Industry [114] the ultimate goal of interoperability is a seamless flow of information between many disparate devices over a network to and from the intended recipients [115].

Chapter 4

Case study: Medical devices and the Integrated Clinical Environment (ICE)

A real time system has a strict timing requirement, data is exchanged with pre-specified time constraints, if data is provided after deadlines it is considered as non valid and can destabilize the behaviour of the system. The integrated clinical environment is introduced as a real time system where it has a hard deadlines and can threaten human lives if any unexpected behaviour is present.

The interconnection of heterogeneous medical devices is emerging as a new requirement in modern medicine. The final goal is to advance beyond the ad hoc connection of medical devices and realize a reliable network where data is exchanged form end to end without compatibility issues, this is defined under the Integrated Clinical Environment (ICE) [116]. In this chapter, ICE is introduced in details, with the presentation of the different concepts related to it.

4.1 Medical devices differences and incompatibility issues

4.1.1 Medical devices

A medical device is a standalone entity that can operate in any clinical situation without human intervention. A medical device is equipped with electrodes or sensors that collect data from patient body and in some cases it can be equipped with actuators. A medical device vary from a simple mobile entity measuring one vital sign and transferring data with Bluetooth to the most sophisticated equipment measuring many vital signs and transferring data with a Wi-Fi or wired connection to servers in order to analyse data. A medical device is very sensitive entity where even some wireless device can affect it behaviour.Each medical device is provided with a *specific* hardware and software configuration which constitutes the heterogeneity in ICE-based systems, connecting these various devices into the same clinical situation where they coordinate their actions cannot be realized without implementing a software solution that plays the role of *mediator* of medical devices.

Data provided by medical device is connected to computational servers through networking ports like RS 232 port (DB-9, DB-15, and DB-25), RJ 45 port, wireless LAN, Bluetooth, USB, or some special data connection systems developed by suppliers for using data by their own IT systems as Dräger Medibus protocol [117] and Philips Instrument Telemetry System (IIT) [118]. The following are the most common hardware connections used by suppliers to input data into the device: PS/2 where the connection of a keyboard or a mouse is possible, USB, RS 232, digital data entering. Data output formats of medical devices are various we cite: text and binary files, alarms, waveforms, and device settings.

Data provided by medical devices is classified in two types: *medical* and *technical* data, medical data is about the medical signs coming from the patient body and the technical data is about the technical characteristics of the medical device like operating system, device model, type of produced data, etc. By it turn vital signs data can be divided into

two types *measured* and *calculated*, for example we have the next measured values coming from an Ivy 450c monitor; systolic pressure = 140 mm Hg, diastolic pressure = 100 mm Hg, the calculated value from this two measured values is the average pressure which is equal to ((2*DiastolicPressure) + SystolicPressure) / 2 = 113,33 mm Hg.
The medical devices characteristics mentioned above are made after a study of 16 medical devices measuring various vital signs and provided by 9 different suppliers.

4.1.2 Interoperability issues

The growth of use of medical devices in healthcare as an essential tool of diagnosis is important. In a single high acuity situation a patient is connected to one or more medical devices, these devices operate as standalone entities where they store data collected from patient body in their local memory or connect it to some smartphones or to servers that analyse data. Studies like [119], [120] and [121] aim to improve safety by coordinating medical devices actions and avoid patient health status deterioration caused by the lack of a standard model that administrate devices and data exchange protocol that enable a secure medical care delivery.
In a clinical situation two medical devices are connected to a patient: a x-ray machine and a anaesthesia machine ventilator, when operating together these two devices may generate complication on the patient health [121], so, medical staff restarts the ventilator promptly. In order to avoid this kind of issues a *software* solution must be developed and standardized, this software solution will be deployed as an intermediate between medical devices and computational servers. Each medical devices supplier has it own manufacturing policy where standards are used, ICE stands the most reliable solution to override the previous mentioned problems.
Standards were defined to specify bases that researchers from both academia and industry can use to improve the quality of health care and to realize a certain level of interoperability. CDA [3], HL7 [79] are examples of data and documents formats standards specification and Digital Imaging and Communications in Medicine (DICOM) is used for images and related communication [122]. ASTM F2761-09 [123] is a standard defined to integrate

medical devices and equipment into a safe and single medical system. From a communication perspective; MedRTI and DDS are two middleware technologies fluently used to create a flexible communication bridge between the components of a medical system in order to exceed the differences in protocols, implementations and messages encodings. EHR (Electronic Health Record) and EMR (Electronic Medical Record) stands as two concepts that are defined for medical data storage.
The ultimate goal of interoperability is a to realize a seamless flow of information between many disparate devices over a network to and from the intended recipients [115]. Since it connection on the patient body a medical device sends data to computational servers and storage devices without human intervention, when a printer is connected to a PC users do not expect to perform any specific operations in order to run the printer and this is how medical devices must be integrated with other devices.

4.2 Medical devices connectivity with ICE-based systems

The integrated clinical environment aims to improve patient safety by coordinating the actions of medical devices and providing data that is valid and in the specified timing limits, ICE is designed for a single high acuity patient environment and this is due to the risk of patient safety deterioration and the presence of many factors that can affect it health status, as factors we mention: the lack of connection between medical devices or incompatibility of two medical devices as shown in [124], [120] and [121]. A survey realized by the west health organisation [1] illustrates that 74% of nurses strongly recommended that data must be shared between medical devices, 50% of nurses said that they witnessed a medical error caused by the lack of coordination between medical devices.

[1]West health organization. Missed connections: a nurses survey on interoperability and improved patient care http://www.westhealth.org/

4.2.1 ICE standard

ICE aims to create an interoperate network of medical devices and EMR in order to improve patient safety and realize an efficient healthcare delivery. Medical devices will be able to cooperate together, share data with servers and store it in other devices without any specific hardware/software installation. Since it connection the medical device is able to detect other devices and decide the importance of sharing data with those equipment. Medical devices must be conform to ICE standard to interoperate with other devices that are *ICE-compliant* regardless of manufacturer [125] , Wifi, USB, Bluetooth are examples of standards that were used for many years to realize a PnP data connection interfaces. figure 4.1 Illustrates how ICE intermediate the clinician and the patient.

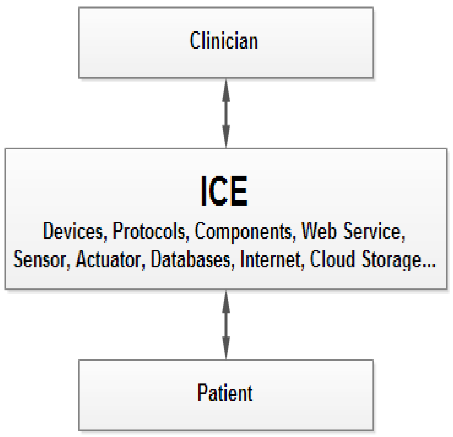
Clinician
ICE
Devices, Protocols, Components, Web Service,
Sensor, Actuator, Databases, Internet, Cloud Storage...
Patient

Figure 4.1: High level view on ICE [10]

ICE was proposed by the CIMIT [116] medical device plug-and-play interoperability project; and ICE functional model was presented by ASTM-F2761 [84], after a draft prepared by [123] in 2008. Julian Goldman [123] defines ICE as "*a medical system that aims to support the care of a single high acuity patient by realizing interoperability between medical devices and other equipment in order to improve patient safety*". OpenICE [97] is the open source implementation of ICE standards that aims to provide a set of tools for developers to create applications related to medical devices connectivity and interoperability, it provides real records of many vital taken from patient it also provides devices simulation and alarms generation.

The medical device Plug-and Play Integrated Clinical Environment (ICE) coordinates

actions of devices provided by different suppliers via a middleware and through a shared network, medical devices suppliers do not have to change their manufacturing policy and produce devices that are compatible with each other the objective is to provide devices thar are *ICE-compliant* in order to realize a PnP standard that is supported by all suppliers.
ICE aims to:

- 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.

4.2.2 Design requirements

ICE stands as an intermediary between the clinician and the patient (medical devices). Many stakeholders are concerned with the interaction of an ICE-based system, these stakeholders vary from human actors like medical staff and the patient, computational actors like servers, data storage devices and data communication equipment, software actors like communication protocols and middleware, software running medical devices and OS, and sensors equipment that are medical devices measuring patient vital signs. All these stakeholders generate various requirements that are related to ICE-based systems design, studies like [95], [126] and [100] aim to gather the requirements from several point of view. In what follows requirements for an ICE-based system design are presented.

1. **Heterogeneity.** It is related to various stakeholders: type of the produced data, software platform (operating system, middleware, and other specific libraries), medical equipment hardware characteristics and communication protocols. Heterogeneity is a design requirement and interoperability is the result of override heterogeneous medical devices differences and realize an interoperable ICE-based systems .

2. **Safety** is a first class requirement for medical systems. *Safety* is the ability of the system to monitor and diagnose the patient in order to avoid the deterioration of his/her health status. In this context, we relate safety to the problems caused by the lack of interoperability between medical devices and the ability of the system to exceed these problems in an intelligent way.

3. **Timeliness.** A medical device must send data in real time to storage devices or computation servers, also, the clinician has to be informed about his patient status without no delays all the deadlines associated to operations are respected. ICE is the place where the time of operations execution intervals are greater than the time taken by data to be sent to computational servers or to communicate data to the clinician. We studied the timing requirement of an ICE-based system that is presented in [10].

4. **Security and privacy.** Given the highly connected nature of most computer-based systems, it is needed to draw architectures that preserve the patient medical data from unauthorized access. Currently, a number of health related systems allow sharing information of Electronic Health Records (EHR) via Internet, even using cloud computing paradigms. *Security* and *privacy* of patient data will be preserved by data encryption mechanisms for the case of data transmission between stakeholders and data access rights when explicit content is requested.

5. **Data accuracy.** The effectiveness of decisions taken by clinicians that use medical systems to diagnose patient health status and safety depends on the *accuracy* of

data provided by medical devices and the integrated applications. Due to the high acuity nature of medical applications data must be *accurate* and that is achieved by avoiding errors caused by formatting, converting, transmission, and execution of processing operations.

6. **Decision support.** An ICE-based system provides logic for supporting *decisions* of clinicians. This logic can be realized by different mechanisms such as machine learning to help analyse data and predict diagnosis/treatment processes easier and more effectively. Reports, notifications, drug injection schedule, real time vital signs data communication, and emergency cases prevention are some of the operations that decision support functionality can facilitate.

7. **Autonomy** is the ability of the system to work and to provide the intended results without human intervention. Implementing self-decision processes is required to enhance safety and to implement autonomy, so that clinicians may concentrate on supervision of the overall care process over the patient. The ontology is a fluent solution that is used in the medical systems to facilitate *autonomy*.

4.2.3 Standard architecture

ICE functional architecture shown in 4.2 was defined in 2009 in ASTM standard [84]. Many components of the architecture can be deployed in one hardware component, ICE standard do not impose or specify one data communication standard but provides examples of various standards that will be required to create an ICE [127].

Medical devices. These devices are developed as standalone units, where they are connected to the patient. Medical devices are provided with different network connection modes and data output ports. An infusion pump is an example of a medical device.

Medical equipment. These are more complex and sophisticated instruments than medical devices; these equipment may be needed for some clinical situations, e.g., a tomography scanner that integrates sensors and a presentation system.

Network controller. It is a middleware that transports data between component of the architecture, it ensures QoS and verifies if the system meet it primary requirements, it is also a mean to discover and connect new inserted devices in the network. The network controller provides an external interface for stakeholders such as EMR (Electronic Medical Record) and information systems, it establishes a common time base for medical devices data.

ICE interface. A data communication port that transports data between medical devices, medical equipment via a Ethernet port, a universal serial bus (USB) port, an RS-232 port or other standard ports. It's a way to realize interoperability between ICE network controller and ICE-compatible equipment.

External interface. This interface is used as the public access port to medical data flowing in ICE, it can be used to connect the system to Internet or to provide data to a professional healthcare facility, the access to medical data should respect ICE security and privacy policies where medical records of patients should be available under some limitations.

Data logger. This entity is used to detect, predict, and record errors that are flowing in the systems. Data saved by the *data logger* is about technical issues that describes abnormal activities of medical devices, equipment, and applications. The data logger component is intended to facilitate system integration, system deployment, and retrospective analysis of performance, incidents and near incidents [123].

ICE supervisor. It is a core component in the architecture it guarantees a reliable flow of data between components and provides a real time scheduling of tasks. The supervisor defines a console where the user can launch applications. provides medical support for clinicians, allowing them to remotely access and control the medical devices. Also, it processes and presents the data provided by medical devices by means of a set of reports and data analysis diagrams.

ICE defines the concept of ***device model*** as "*representation of the capabilities of ICE-compatible equipment that includes the information needed to qualitatively and quantitatively describe, control and monitor its operation*". It is a model-based approach that defines in a declarative meta-data description the hardware/software capabilities of the

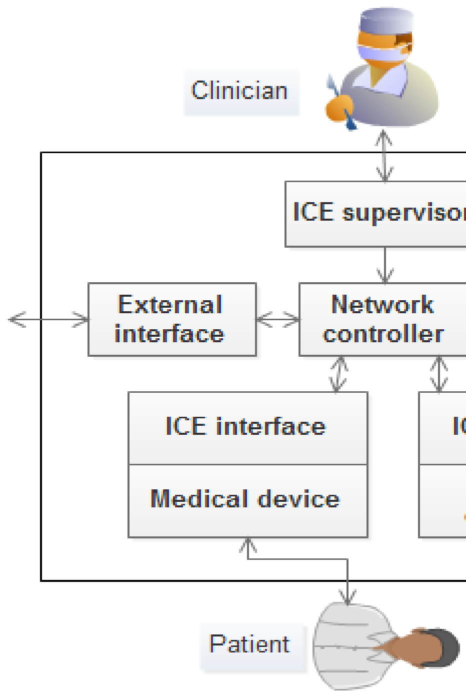

Figure 4.2: Functional model of the integrated clinical environment [9]

medical device and decides if the new connected equipment is suitable for the intended use. To accomplish this the device model describes the device in details: inputs/outputs of the medical device, measured vital signs, units of representation (e.g., mmHg, beats/min), and the bit encoding of the parameter (e.g., 16-bit fixed point, 32-bit floating point), time of measurement, device technical configuration. It also includes data about the QoS in order to guarantee effective results and safe operation,

4.3 Timeliness in ICE

Medical systems that have direct impact on patient safety have hard timing requirements and this is due to the critical situation that can be generated if results arrive after their specified deadlines. Timing requirements are related to many actions: (1) Vital signs measurement and communication time, (2) Time taken by ICE to process data and produce results, (3) data communication time to clinicians. Medical devices perform time-critical tasks. Some of them require high temporal precision such as measuring pulse transit times or controlling intra-aortic balloon pumps; others only require low precision such as sending a stop signal to the PCA pump in case of an overdose [128].

Until the time of this study there is no clear study about timing requirement of ICE-based systems, we defined the timing requirements related to the proposed component model in a previous study [10]. An ICE-based system delays in an ICE-based system can generate issues on the patient health status, also the diagnosis/treatment process is based on the accurate values of vital signs and the the time when those parameters where taken. A medical device generates two types of data *technical* and *medical* for both the exact timing values are essential, when the clinician executes the application that displays alarms history it is needed to gather the exact timing values of the production of alarms and decides the source of these alarms in order to avoid any impact on the patient health status.

The advantage of integrating medical devices into a patient-centric situation is to have real time comprehensive data for the healthcare information system/electronic medical record/electronic health record (HIS/EMR/EHR) [123], this real-time data and can improve patient safety and medical workflows such as: a reliable diagnosis/treatment process, ensures a continuous flow of medical and technical data, and monitoring the performance and activity of medical devices and ICE applications.

Chapter 5

A component model for ICE-compliant medical systems

In this chapter the component model that provides the general framework for designing ICE-enabled applications is presented. From this proposed model (or architecture) specific systems can be provided; a system is an implementation of an architecture or model. The proposed component model supports *data* real-time acquisition, transfer, treatment, and communication. It is based on components as a design unit and (web) services to achieve interoperability.

5.1 Design requirements

The component model presented in this section is based on a set of requirements that leaded to that design, the proposed component model fulfils:

- Support of activities with time constraints.
- Execution predictability; schedulability facilities for ensuring fulfilment of deadlines.
- Availability of data for all stakeholders in the system at any time and regardless of the used device.
- Autonomy, as the system provides results without human intervention.

- Support of data flows in different paths (reflecting the size, type and accuracy of data).
- Heterogeneity of medical equipments that will be able to interoperate.
- Deployment/hardware efficiency, as the architecture has been designed to minimize physical equipments on the patient site in order to facilitate integration.

Critical events may come from clinician devices as well as from patient side devices. The detection of a critical event triggers the execution of some activities that must be completed before their deadlines expire in order to respond the event generated by the device. These activities are mainly related to timely data transmission or processing.
As examples, a clinician device may request a higher resolution data or a specific zoom for an area of intervention; and a patient side device may detect some abnormal vital signs conditions.
Then, device modelling is supported by the proposed model. Precisely the modelled characteristics are: display information (screen size, etc.), power data (battery, etc.), processor capacity (i.e., CPU), and communication capacity (i.e., network media characteristics).

5.2 High level view of the architecture

The component model is divided into three layers due to the diversity of stakeholders (medical devices, data, data adapters, human actors, and software sensors) and the treatment process that data must undergo in order to be used by different components. So the reason of the division of this architecture into layers view is to separate between requirements and to have a clear design view.
The first layer starting from below is dedicated to connect medical devices to the upper components, the second layer is the core of decisions made on the architecture, and finally the third layer directs the behavior of the components by serving like software sensors. Each layer has input and output data exposed via components interfaces, the assembly of components in one layer is realized according to the nature of the main operations realized by those components, the three layers are presented as following:

- Data control and transmission layer (DCTL).
- Diagnostic and decision layer (DDL).
- Control layer (CL).

The figure 5.1 presents a general overview of the component model. It has three layers defined with proper interrelation.

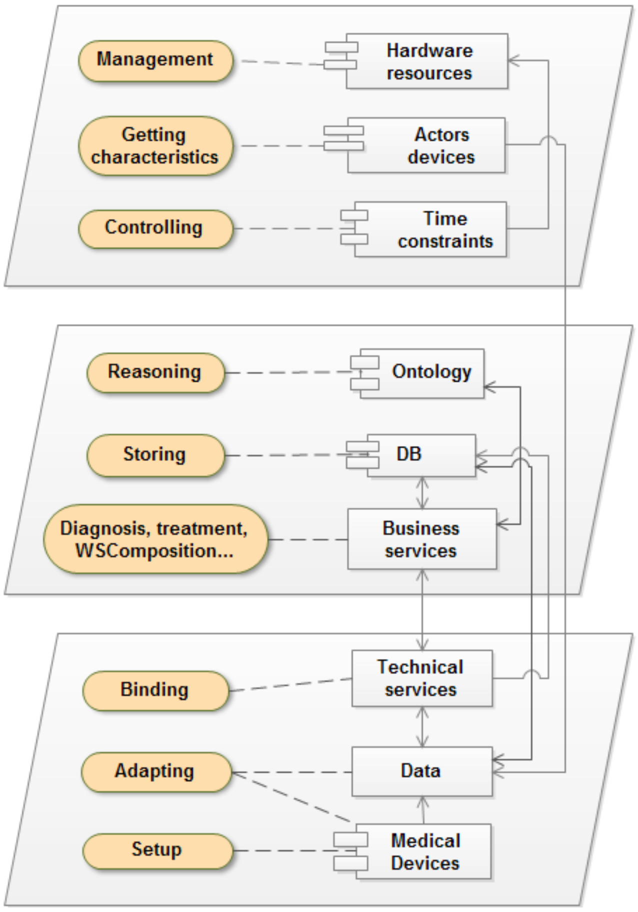

Figure 5.1: General view on the architecture [10]

Each component is defined by operations executed by or on this component, the actors devices component execute operations of getting the characteristics of user's devices and send them to the mediator to adapt data.

5.3 Design internals

The architecture presented in figure 5.1 is detailed in figure 5.2 based on a per-layer basis. The architecture has been designed to minimize IT equipments on the patient place (ambulance, home, clinic) in order to facilitate integration. A single modem or router plus a mediator (hardware and software component) connecting the patient environment to servers is sufficient.

The architecture ensures the respect of time requirements and will also improve response times with the use of some concepts that will be explained later. It is also estimated that the response time in the control system (component model) is greater than the response time of the controlled system (medical devices). For example, the time to sample the blood pressure and to send it is less than the time of its treatment.

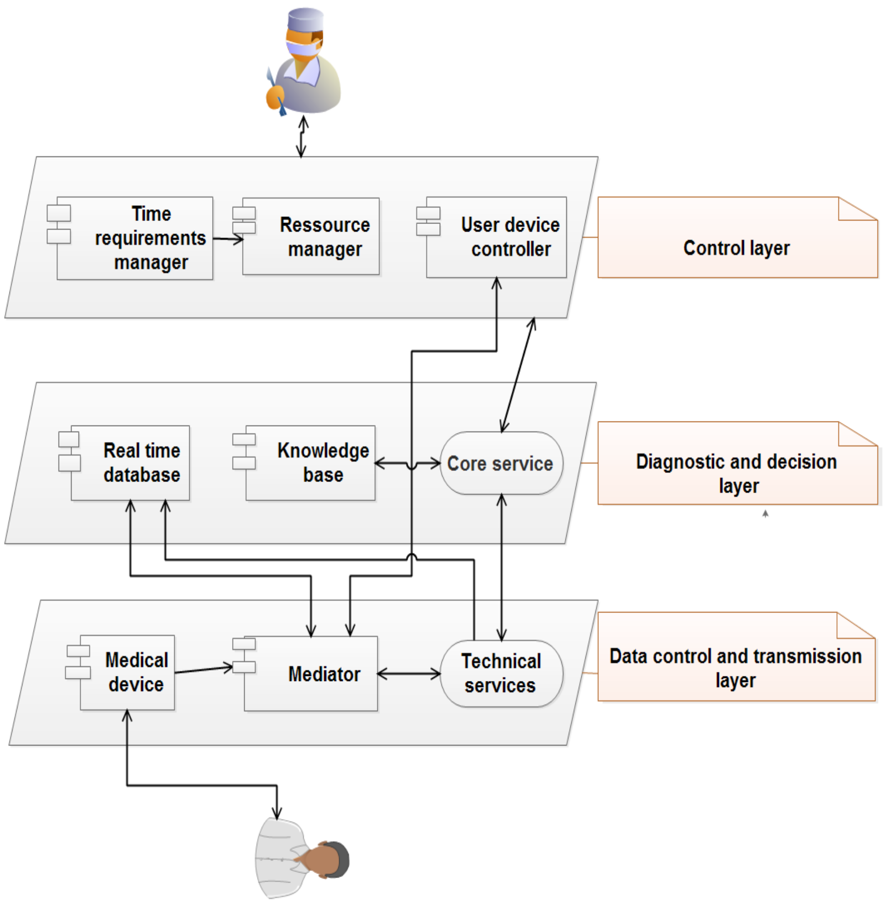

Figure 5.2: A component model for the Integrated Clinical Environment [9]

5.3.1 Data control and transmission layer (DCTL)

The data control and transmission layer is in direct contact with patient side environment, where it serves as device integrator, data carrier and adapter. Medicals devices begin to provide vital signs data after its connection on a patient body, a vital signs monitor is required in each clinical environment so that the clinician present on site could follow the patient's condition or if he wants for example to modify some parameters like waveforms ladder or alarms intervals.

Medical devices are provided by different suppliers, the majority of medical devices are provided with a data connection source like RS 232 port (DB-9, DB-15, and DB-25), RJ 45 port, wireless LAN, Bluetooth, USB, or some special 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 where the connection of a keyboard or a mouse is possible, USB, RS 232, digital data entering. Data output formats of medical devices are various we cite: text and binary files, alarms, waveforms, and device settings.

We can divide the data coming from medical devices into two types: medical and technical data, the medical data is about the medical signs coming from the patient body and the technical data is about the technical characteristics of the medical device like operating system, device model, type of produced data, etc. By it turn vital signs data can be divided into two types measured and calculated, for example we have the next measured values coming from an Ivy 450c monitor; systolic pressure = 140 mm Hg, diastolic pressure = 100 mm Hg, the calculated value from this two measured values is the average pressure which is equal to ((2*DiastolicPressure) + SystolicPressure) / 2 = 113,33 mm Hg.

The medical devices characteristics mentioned above are made after a study on 16 medical devices measuring various vital signs and provided by 9 different suppliers.

DCTL is sown if figure 5.3 and is composed of the following components:

The mediator

The *mediator* is a software and a hardware component that is in direct contact with medical devices plugged on the patient's body. The architecture of this component will

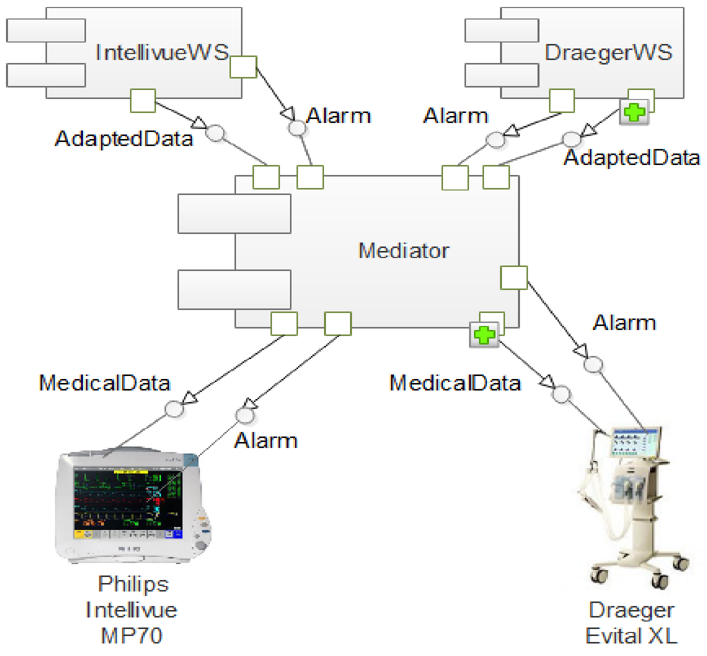

Figure 5.3: Attachment of a Web service to a medical device [9]

be detailed in the next section.

Medical devices

Send the vital signs data to the mediator in real time. In this context, the delay of data transmission varies around milliseconds units (depending on the specific medical device manufacturer) and it must have to be accounted for.

Technical services

Called low level because they are elementary WS, their roles is to receives the data from the mediator, gives access to the medical devices and sends the sampled data to the second layer. For each medical device corresponds a web service that will encapsulate it functionalities, technical and medical data where an intervention on the material corresponds to an intervention on it's web service.

The communication between medical devices with the current technologies and solutions is very hard to achieve, for example to view the data collected from the Hospira infusion pump on the Philips Intellivue MP70 monitor, a physical connection is needed and it may not work because of the differences between the two devices (connection between medical devices is possible just using devices coming from the same supplier or between devices that have optimal characteristics. This connection is realized throw pre-specified

hardware and software characteristics and after reading long installation manuals); we propose to cover this connections problems by the use of this technical services where the communications between medical devices will be ensured via their web services.
Any change on data that will affect its accuracy will be preceded by a storage on the database with dedicated web services in order to keep its integrity. Data is transported to the real-time database on the data saver component to store it with its original format coming from medical devices. If a check has to be done on these data, it is the mediator component that intervenes. Video is displayed on clinician device using streaming because of their large sizes.

5.3.2 Diagnostic and decision layer (DDL)

DDL 5.4 is positioned in the middle of this architecture where any major decision is taken and where all data is stored. it also requires greater physical performance than the other layers. It is on this layer also where response times are longer, it's composed of the following:

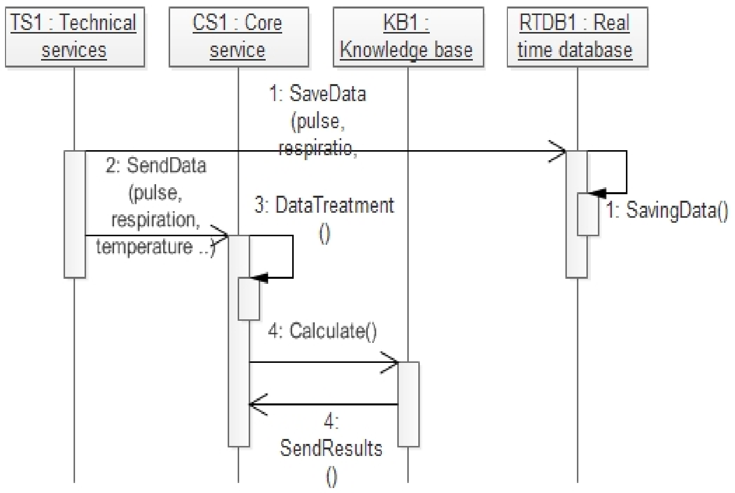

Figure 5.4: Sequence diagram of data flow in DDL [9]

Core service

Is a composed service that provides medical decisions such as disease analysis and drug schedule that constitute a support for clinicians. The core service has as input data provided by the technical services and produces as output data that is used by different components.

The core service is a composed web service that has input data coming from elementary web services of the first layer. and provides as outputs a set of results via web services.

Real time database

Given the large amount of information, a real-time database is essential for the required data search and store. It provides timely data access and guarantees safe simultaneous access by multiple actors. Its difference with an ordinary database is that it allows to attach restrictions to the operations performed on the data. It's Divided into two parts:

Event recorder. Any event that takes place in this architecture is recorded in this component with a date and a reference to the data associated with this event, it also allows the recovery of the system in case of failure or reboot, which is an important criterion of real-time system.

Data saver. Contains data circulating in the system of all types: text, binary and XML files, image, sound, video, characteristics of user devices, backup of states after validation of a disease and a corresponding treatment. For each transaction is associated a deadline and a priority and for every given data a time interval for which its value is valid, for example the glucose level was measured at time t and it is valid from ***t*** to ***t+5***.

Knowledge base

Constitutes the brain of the architecture and aims to guarantee the autonomy of the system and a decision support for clinicians. on the health analyzing computations tasks and the treatment decision services. The reasoning is guided by a set of SWRL (Semantic Web Rule Language) rules from a set of parameters and values that will be the basis of decisions taken in the second layer. It is also a mean of achieving the predictability of the system; that based on vital signs and patient history may identify the patient problems, possible treatment, schedule for treatments and visits, bypassing other processes.

5.3.3 Control (CL)

The CL contains a set of software sensor running between the system and the clinician. It directs and checks the behavior of the model and ensures the respect of time limits.

Following are presented the components of CL:

User device controller

A software component whose role is to check characteristics of the device used by the user to connect: screen size, battery durability, RAM size, power of the CPU, and then sends the data to the mediator.

Both Network Controller and User Device Controller components are very important, acts as software sensors for better management of the user and his device, and to adapt the data that will be communicated to him. They are also an essential means to achieve the time requirements. For example there is no need to send a video clip of one hour time with HD quality to a user who has a smartphone with a weak battery, poor picture quality and a very low internet speed, it will generate exceeded deadlines during data transmission and remote diagnosis process.

Resource manager

Monitors the fulfilment of the operation deadlines for those cases where real-time operation is required. Some applications and service functions have execution priorities and associated finalization deadlines. The resource manager component monitors the schedulability of the different functions of the system.

Time Requirements Manager

This component has the role of observing the compliance with the specified time requirements. This component has the role of observing the compliance with the specified time requirements. model; it fine tunes and adapts the components parameters to fulfill the overall time requirements that are specified in the ICE-based system.

The components of the *time requirements manager* are the following:

Deadline Controller. Execution delays or timeouts are detected by this component. The operation (or process) of connecting a new medical device has an associated deadline. If the operation overruns the specified deadline, the *deadline controller* components detects it and reports it by sending an alarm message to the *alarm and event notification* component. This informs the user/clinician so that action is taken, e.g., manual connection.

Time Synchronizer. In a distributed context integrated by heterogeneous actors such as medical devices and different stakeholders, this component takes the role of synchronizing the clocks of these actors with a central clock. Having a central clock ensures logical synchronization. For applications requiring physical time, UTC (universal coordinated time) receivers can be connected to the system and precisely to the central clock. so as to facilitate access and for better measurement of compliance with time deadlines.

For the components of this last layer, the respect of time requirements is a functional objective, for some of them it is abstract and for others it is discreet. To guarantee a certain level of security a firewall separates the architecture from the user's device, he can access the system via accesses provided by the system administrator. Figure 5.5 shows the component model using UML presentation of the different component and interfaces exchanging data between them.

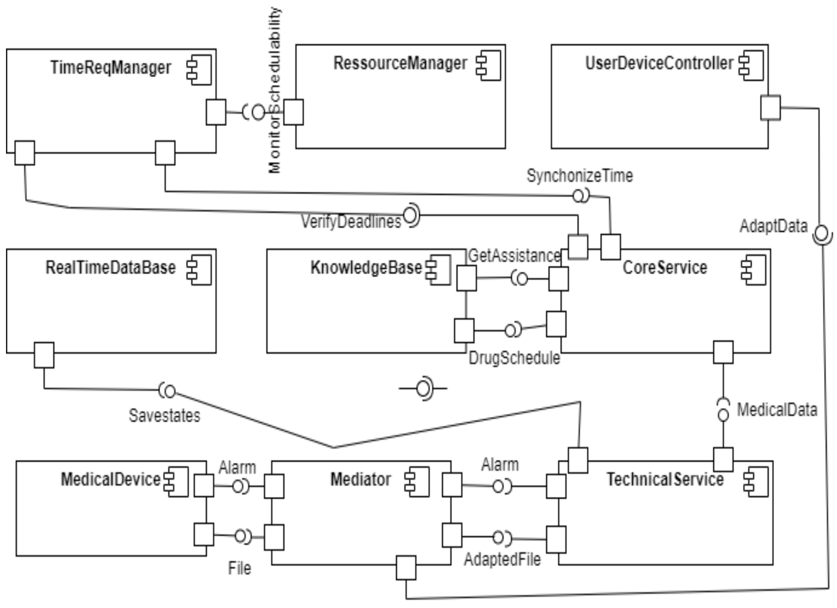

Figure 5.5: UML component diagram of the architecture [9]

5.4 Differences between the proposed component model and studied models

The points mentioned in the following list are differences between the proposed component model and all the models presented in section 2:

- The design of the component model was based on covering the design and the run-time phases of the software, the behavior of the presented models is not clear when dealing with hardware resources, databases, and human.

- The use of *specific* components, because a component that have special tasks to perform is more efficient than a *generic* component for a given application domain. Reusability is ensured for such type of components for systems requiring components to execute special tasks.

- The models do not integrate a data integrity check/treatment mechanism, the component model ensures this mechanism by the *mediator*.

- Autonomy, for the component model there is no need for human intervention, unlike the other models where a description for such parameter is limited (Koala) or not defined (Rubus, UM-RTCOM, SaveCCM).

- Heterogeneity of data and medical devices (sensors). The models do not provide any description in how they deal with different formats of data.

- Connection with sensor/actuator, the component models do not illustrate how they manage communications with sensor/actuator, in order to build a real time system such a concept is primary.

- Unlike the presented component model, interoperability and communication between heterogeneous components and medical devices is not defined by the models.

- A global time synchronizer integration (time synchronizer component) on the component model whereas the presented models do not support such a component (except SaveCCM).

- A *deadline controller component* is integrated on the component model in order to analyse, anticipate deadlines exceeding and prevent other components on the systems to free more hardware resources in order to meet the deadlines.

- A hardware resources component manager is used on the component model in order to communicate with the run-time environment to free more resources for specific or urgent tasks or when the deadlines are near.

Chapter 6

Experiments and results

6.1 Medical devices connector

The mediator is introduced in order to connect heterogeneous medical devices into the same network mediator is a hardware and a software component that has as input data coming from medical devices and as output data provided to the technical services. The mediator is composed of three components each one of them performs operations that are explained in what follows:

- The XML files processing incurred by the parsing and additional message headers leads to high delays [18], so it has to be compressed in order to speed up all the operations that it will undergo, which is the work of the data compressor component (shown in figure 6.1) that reduces the files size without changing their type or quality.

- Data has to be converted from its primary format to another for printing, visualization and processing, and here comes the data converter component (shown in figure 6.1), it converts data types based on a set of parameters such as network load, user device type or battery charge. For example, a text file will be transmitted faster than an Excel file

- Due to the criticality of ICE it must provide accurate results and in time, therefore, the data must undergo an accuracy check performed by the component data correctness (shown in figure 6.1), It is classified as the last component of the mediator so it works as a data checker and validator.

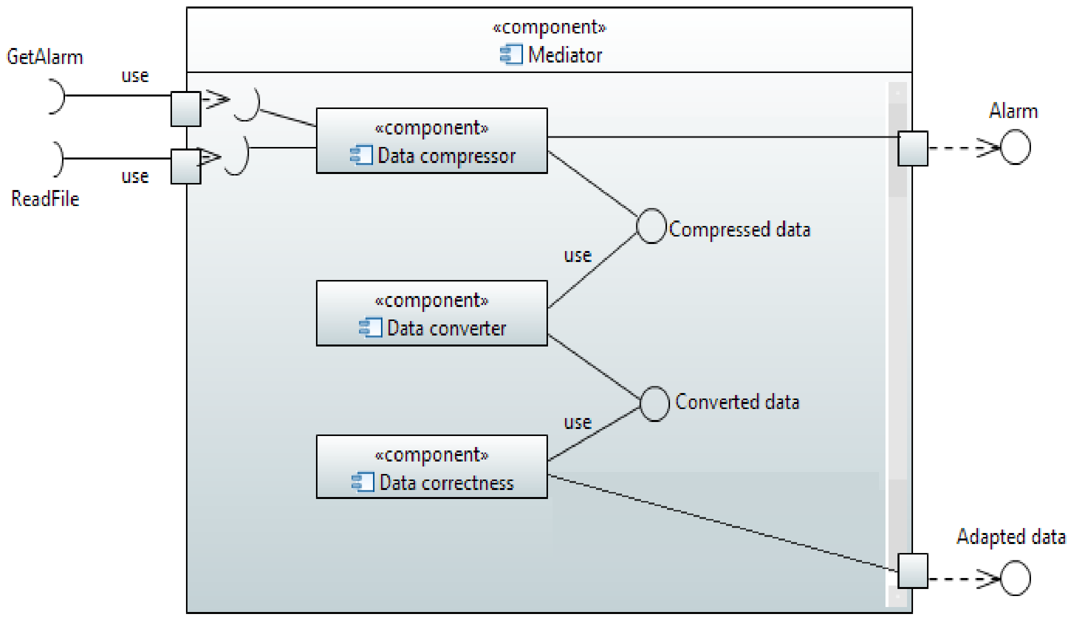

Figure 6.1: Mediator internal structure [9]

6.2 API

The mediator is a composed component that plays an essential role on the presented component model in this paper, it creates communication between medical devices and adapts data according to the clinician device hardware/software characteristics. Figure 6.1 shows the internal structure of the mediator.

The mediator communicates with other components via predefined ports and interfaces while components communicate via interfaces. The mediator has as inputs: files (contain technical and medical data) and alarms, it produces as outputs data (adapted according to the user device and Internet speed) and alarms.

Table 6.1 shows input and output parameters of each component of the mediator. DataCompressor has input files with different formats and sizes generated by the medical devices, it compresses the size of the files to smaller files and treats excel and binary files, DataConverter has as inputs files provided by the DataCompressor, it converts the files

Table 6.1: Mediator internal components inputs and outputs [9]

Component	**Input**	**Output**
DataCompressor	Text, excel and binary files Boolean	Text, excel and binary files Boolean
DataConverter	Output of DataCompressor	Text, excel and binary files
DataCorrectness	Output of DataConverter	Pulse: integer, respiration: integer, temperature: float,..

to a unique file format that is supported by other components. DataCorrectness has as input a unique file format, it generates variables for each measured value.

Table 6.2: Mediator interfaces and data values [9]

Interface	**Input parameters**	**Output parameters**
GetAlarm	IsAlarmRaised: boolean, GetAlarmTime: Date, GetAlarmText: String	AlarmRaised1: boolean, AlarmTime1: Date, AlarmText1: String
ReadFile	GetFileLocation: String, GetFileSize: float, GetFileFormat: String	FileLocation: String, FileSize: float, FileFormat: String
Alarm	AlarmRaised1: boolean, AlarmTime1: Date, AlarmText1: String	AlarmRaised2: boolean, AlarmTime2: Date, AlarmText2: String
CompressedData	GetNewFileLocation: String, GetNewFileFormat: String	FileLocation2: String, FileFormat2: String
ConvertedData	GetNewFileLocation2: String	FileLocation3: String
AdaptedData	GetVitalSignName: String, GetVitalSignValue: String or Integer or Float, GetVitalSignTime: Date	VitalSignName: String, Value: String or Integer or Float, MeasurTime: Date

Table 6.2 shows the interfaces implemented on the component model and data values for each provided and required interface. GetAlarm interface transfers data from medical devices to DataCompressor component while Alarm interface transfers data from DataCompressor component to user device and this is due to the criticality of the alarm. ReadFile interface is responsible of transferring data included in files provided by medical devices into DataComporessor component, CompressedData interface transfers data from DataCompressor component to DataConverter while ConvertedData interface transfers data from DataConverter component to DataCorrectness component. The AdaptedData interface is the link between the mediator internal treatment and DDL.

6.3 Mediator operation process and program design

The mediator is a first class citizen of the component model and therefore it behaviour has to be studied. The operation process of the mediator is shown in figure 6.2, it's starts when the medical device is connected to the network, if the file size is big then it's converted to a smaller one by deleting repetitive lines of the file then sent to the data converter component. This last, receives internet load and convert file type if needed to text for example for faster transmission.

The data correctness component receives a message from the core service if some bad medical data is detected, for example, if some vital signs data are wide out of predefined range. In this case, a restart device message is sent to the clinician or just to verify if electrodes are installed correctly.

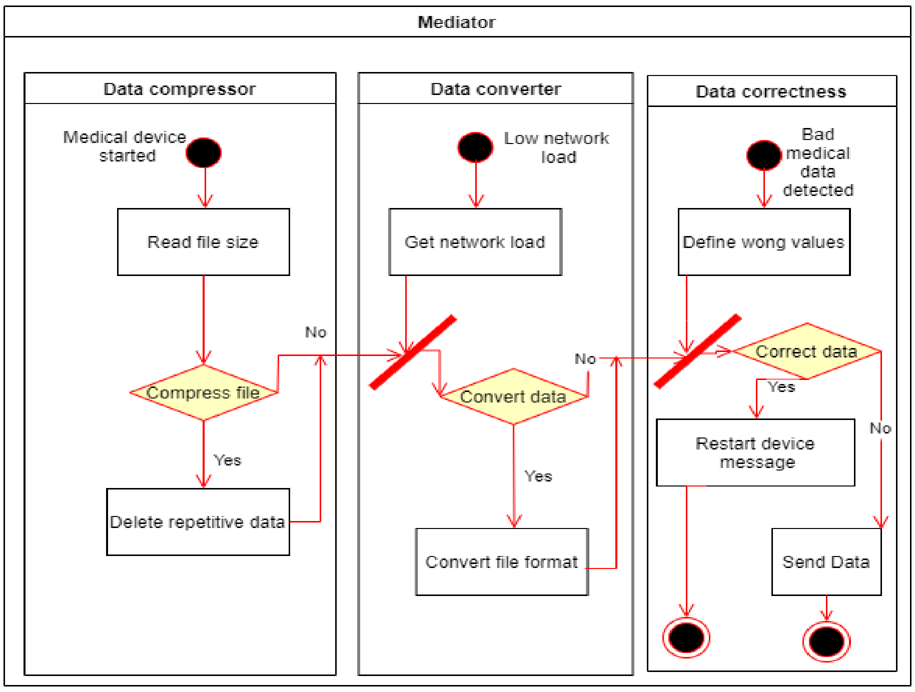

Figure 6.2: Mediator operation process [9]

The communication of data provided by medical devices for processing servers is still hard to achieve, because of the diversity of medical devices (hardware and software), diversity of suppliers policy, multiple data output formats, patient environment limitations, and networking capabilities.

Many initiatives appeared in order to facilitate the communication between medical devices and IT infrastructures. In a clinical situation the presence of medical devices provided by different suppliers and the communication and even the connection of those medical devices to each other is hard or even impossible (connection of MD from different suppliers) where it is needed to read long technical manuals and the intervention of technicians.

6.4 Implementation Configuration

The development of the component model is based on Python 2.7.18, the runtime environment is an Intel core i5- 7300 at 2.20 Ghz with 8 GB of RAM. Tests were performed using the next components and that are illustrated in figure 6.3.

- Processing equipment : 1 Laptop
- Display equipment : 2 LCD
- Medical devices : Microlife blood pressure monitor BP A200 AFIB and Onyx Nonin 3231 USB model
- Adaptation units : 2 Raspberry Pi 3 Model B, Quad Core CPU 1.2 GHz, 1 GB RAM

Figure 6.4 shows a simulated situation on an elderly who is at home and connected to a Microlife blood pressure monitor BP A200 AFIB and an Onyx Nonin 3231 USB model, the two devices are plugged on a raspberry pi 3 model. The component model is deployed on each raspberry, a Wifi connection is used to send data to the cloud. The clinician is is hospital and use his smart phone or his desktop PC to connect to the platform and check patient vital signs.

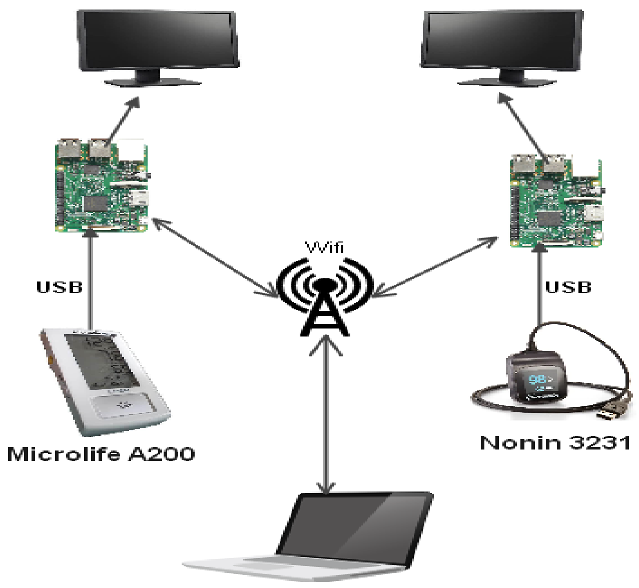

Figure 6.3: Laboratory experiments configuration components [10]

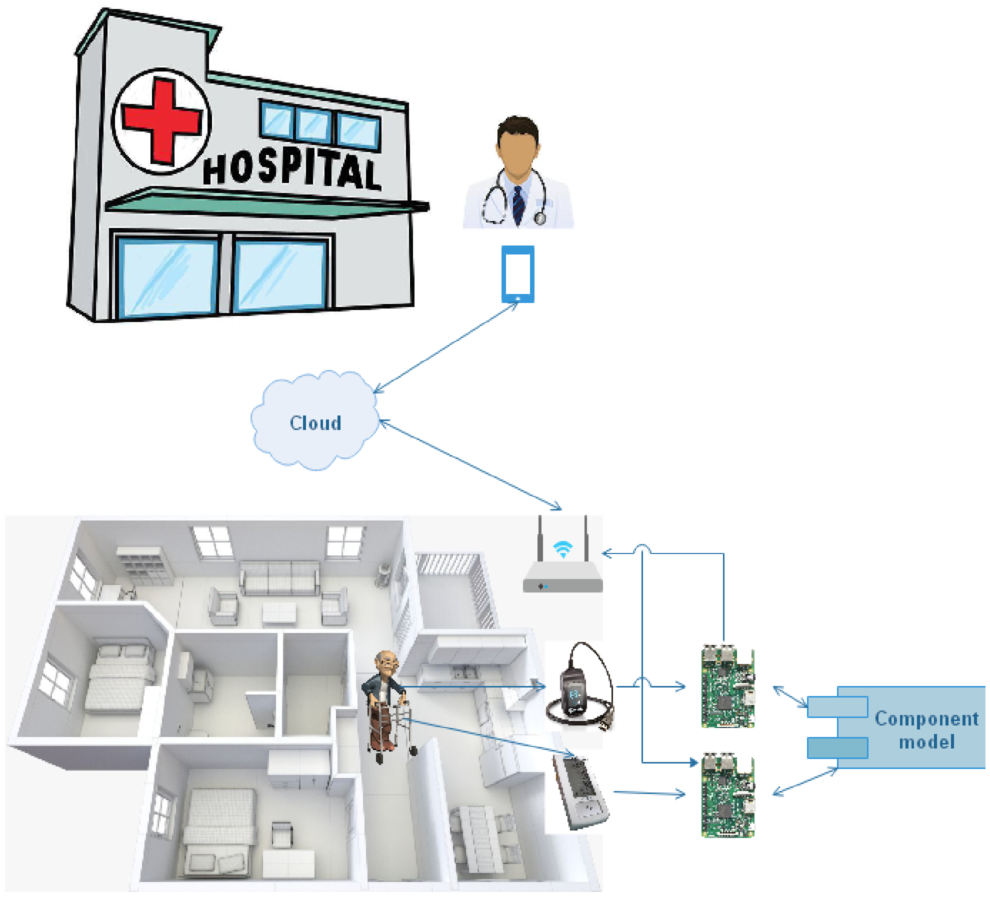

Figure 6.4: Simulated situation

Table 6.3: Number of used elements for the application [9]

Medical devices	**Data output types**	**Mediator**	**Web services**
2	2	1	2

6.5 Some figures

Table 6.3 shows that two medical devices are used and which are the Microlife blood pressure monitor BP A200 AFIB and Onyx Nonin 3231 USB model, the two data output files are an Excel and a text files and there is one mediator (mediator1) to connect medical devices and extract data from them, also, two web services are used (MicroLifeWS and NoninWS) in this case study for each medical device is affected a web service that defines it behaviour and the generated data.

Table 6.4: Medical devices connectivity and data exchange parameters [9]

Execution time	**Exchanged messages**	**Alarms**	**Generated files**
1 minute	19	1	2
10 minutes	155	1	2
60 minutes	917	2	2

Table 6.4 shows some results about data exchange between medical devices and the mediator for a given laps of time, the number of generated files is fix during all the execution process and this because each device generates a file when the execution is started this file is used as a data buffer, in this case there are two files formats a text and an Excel files. After 60 minutes of execution 917 messages were exchanged this illustrates that the component models supports the heterogeneity of medical devices and that data is transferred without problems.

Alarms are an important mean to guarantee patient safety in this case two alarms were generated during 60 minutes of the execution cycle, these alarms were caused by a degradation in the patient vital signs: heart rate registered between 59 and 50 beats per minute for 1 minute and respiration rate between 9 and 11 breaths per minute.

This example shows the efficiency of the component model to guarantee heterogeneity of medical devices by coordinating medical devices actions through the mediator and patient safety monitoring by generating alarms when one or more vital signs are out of the regular range.

Chapter 7

Conclusions and future work

Real time systems aim to provide results before deadlines are achieved, these deadlines are varying according to the system intended use and it relation with threats on human lives. It's up to the system designers to define the timing requirements of a real time system, these timing requirements are defined on design time and verified at execution time. Fault tolerance, data accuracy, schedulability, and stability are the most common requirements of a RTS.

Component-based paradigm is the best solution to program RTS and this due to it modular nature and the septation between code and interfaces. Components hides the complexity of RTS no matter their complexity and their timing requirements. Component approach ensures software reuse, whereas, this last notion is not well studied when used with RTS because of their criticality unless when used for the same application domain and more precisely in the same project.

Modern medical systems integrating medical devices to monitor patient safety in a one high acuity environment are *real time*, therefore, the study of such type of systems must cover the real time aspect. Medical devices have to provide accurate data to the clinician before the deadlines and at the same time make sure that the data is secured from attacks. Many medical devices may be present in one clinical situation which may create incompatibility between data provided by different devices or in same cases a device may affect the operation of another device which is called *hazards* in the medical domain. Many stakeholders tried to propose some solution to create communication bridge between different devices, but no standard has been adopted by manufacturers and this is due to development costs and technology adaptation process.

The integrated clinical environment is a new framework aiming to provide an open solution to exchange data between heterogeneous medical devices.It's a reference model to ensure safety in the development of medical services, it aims to create communication and coordination of medical devices installed on the same high acuity patient environment. ICE-based systems define safety as the ability of the system to avoid the patient health status deterioration because medical devices couldn't operate in the same network, it also covers the heterogeneity of medical devices where each device has a specific hardware/software configuration.

The thesis presented a component model for ICE-based systems where it emphasis on safety and heterogeneity as two essential concepts for the design of medical systems. The paper studied the component model in details and ICE-based systems in particular. The mediator was integrated as a solution for heterogeneity, notification, clinician support and data accuracy as solutions for safety.

The component model was detailed with UML diagrams presenting it internal design with inter-relation and it was validated with a prototype implementation with two medical devices in test where data exchange was illustrated in one clinical situation. Also, the real time behavior of the model was tested where data exchange was measured in predefined laps of time and deadlines were respected.

As future works, the behavior of the component model will be tested with more devices from different manufacturers. Also, data exchange between the model and the clinician will be tested in one real condition, where the clinician will be notified about any event of the system. In addition , other timing requirements will b studied such as worst case execution time and execution time of an operation.

Bibliography

[1] Julian M Goldman. Medical devices and medical systems-essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ice)-part 1: General requirements and conceptual model. *ASTM International*, 2008.

[2] Digital Imaging and Communications in Medicine (DICOM). http://medical.nema.org/, 2020. Accessed: 2019-09-18.

[3] Health Level Seven - clinical document architecture. http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. Accessed: 2017-03-02.

[4] IEEE 11073, personal health devices. https://www.axios.com/health-care-spending-2018-hospitals-doctors-drugs-economy\-b520930a-0cd9-4c68-9dd1-f31cc08ec264.html, 2008. Accessed: 2019-08-01.

[5] IEEE. Medical Devices Profile for Web Service (MDPWS). https://ieeexplore.ieee.org/document/7390446, note = Accessed: 2019-09-16, 2015.

[6] HL7. Fast Healthcare Interoperability Resources (FHIR). https://www.hl7.org/fhir/overview.html, 2011. Accessed: 2019-08-01.

[7] OR.NET interoperability project. https://ornet.org/, 2014. Accessed: 2019-08-01.

[8] SNOMED. http://www.snomed.org/, 2020. Accessed: 2019-08-30.

[9] Imad Eddine Touahria and Abdallah Khababa. A component based framework to enable medical devices communication. *Ingénierie des Systèmes d'Information*, 26(3):pp. 295–302, 2021.

[10] 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 01, pages 278–287, July 2017.

[11] Insup Lee, Joseph Y-T. Leung, and Sang H. Son. *Handbook of Real-Time and Embedded Systems*. Chapman &amp; Hall/CRC, 1st edition, 2007.

[12] Ning Gui, Vincenzo De Florio, Hong Sun, and Chris Blondia. A hybrid real-time component model for reconfigurable embedded systems. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, page 1590–1596, New York, NY, USA, 2008. Association for Computing Machinery.

[13] Ivica Crnkovic and Magnus Larsson. *Building Reliable Component-Based Software Systems*. Artech House, Inc., USA, 2002.

[14] Vincenzo De Florio, Christian Esposito, and Domenico Cotroneo. *Resilient and Timely Event Dissemination in Publish/Subscribe Middleware*, pages 1–20. 01 2012.

[15] Patrick Lardieri, Jaiganesh Balasubramanian, Douglas C. Schmidt, Gautam Thaker, Aniruddha Gokhale, and Thomas Damiano. A multi-layered resource management framework for dynamic resource management in enterprise dre systems. *J. Syst. Softw.*, 80(7):984–996, jul 2007.

[16] D.B. Stewart, R.A. Volpe, and P.K. Khosla. Design of dynamically reconfigurable real-time software using port-based objects. *IEEE Transactions on Software Engineering*, 23(12):759–776, 1997.

[17] Andy Wang. *Component-oriented programming*. Wiley, Hoboken, N.J, 2005.

[18] Petr Hošek, Tomáš Pop, Tomáš Bureš, Petr Hnětynka, and Michal Malohlava. *Comparison of Component Frameworks for Real-Time Embedded Systems*, pages 21–36. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[19] Gregory Campbell*. Statistics in the world of medical devices: The contrast with pharmaceuticals. *Journal of Biopharmaceutical Statistics*, 18(1):4–19, 2007.

[20] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[21] Julian Goldman. "joint workshop on high confidence medical devices, software, and systems (hcmdss) and medical device plug-and-play (md pnp) interoperability". page 29, 05 2008.

[22] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[23] Mark Gaynor, Feliciano Yu, Charles H. Andrus, Scott Bradner, and James Rawn. A general framework for interoperability with applications to healthcare. *Health Policy and Technology*, 3(1):3 – 12, 2014.

[24] M Logan and B Patel. Medical device interoperability-a safer path forward. *AAMI, Arlington, VA2012*, 2012.

[25] M. Dingler, C. Dietz, J. Pfeiffer, T. Lueddemann, and T. Lüth. A framework for automatic testing of medical device compatibility. In *2015 13th International Conference on Telecommunications (ConTEL)*, pages 1–8, July 2015.

[26] N. Decker, F. Kühn, and D. Thoma. Runtime verification of web services for interconnected medical devices. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pages 235–244, Nov 2014.

[27] M. Robkin, S. Weininger, B. Preciado, and J. Goldman. Levels of conceptual interoperability model for healthcare framework for safe medical device interoperability.

In *2015 IEEE Symposium on Product Compliance Engineering (ISPCE)*, pages 1–8, May 2015.

[28] Kejia Li, Steve Warren, and John Hatcliff. Component-based app design for platform-oriented devices in a medical device coordination framework. *IHI'12 - Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, 01 2012.

[29] J Goldman. Advancing the adoption of medical device plug-and-play interoperability to improve patient safety and healthcare efficiency. medical device "plug-and-play" interoperability program. Technical report, Tech. Rep, 2000.

[30] Julian M Goldman. Getting connected to save lives. *Biomedical instrumentation & technology*, 39(3):174–174, 2005.

[31] Robert Orfali, Dan Harkey, and Jeri Edwards. *The essential distributed objects survival guide*. John Wiley & Sons, Inc., 1995.

[32] Andy Ju An Wang and Kai Qian. *Component-oriented programming*. John Wiley & Sons, 2005.

[33] Clemens Szyperski. *Component Software: Beyond Object-oriented Programming*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[34] Bernd Fischer, Matthias Kievernagel, Gregor Snelting, et al. Deduction-based software component retrieval. In *In Proceedings of IJCAI'95 Workshop on Formal Approaches to the Reuse of Plans, Proofs, and Programs*. Citeseer, 1994.

[35] Hironori Washizaki, Hirokazu Yamamoto, and Yoshiaki Fukazawa. Software component metrics and it's experimental evaluation. In *Proc. of the International Symposium on Empirical Software Engineering (ISESE 2002)*, volume 2, 2002.

[36] Kung-Kiu Lau and Faris M. Taweel. *Data Encapsulation in Software Components*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[37] I. Crnkovic. Component-based software engineering - new challenges in software development. In *Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003.*, pages 9–18, June 2003.

[38] Kung-Kiu Lau. *Component-Based Software Development*. WORLD SCIENTIFIC, 2004.

[39] Kung-Kiu Lau and Zheng Wang. Software component models. *IEEE Transactions on Software Engineering*, 33(10):709–724, 2007.

[40] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. V. Chaudron. A classification framework for software component models. *IEEE Transactions on Software Engineering*, 37(5):593–615, Sept 2011.

[41] K. Hanninen, J. Maki-Turja, M. Nolin, M. Lindberg, J. Lundback, and K. L. Lundback. The rubus component model for resource constrained real-time systems. In *2008 International Symposium on Industrial Embedded Systems*, pages 177–183, June 2008.

[42] H. Hansson, M. AAkerholm, I. Crnkovic, and M. Torngren. Saveccm - a component model for safety-critical real-time systems. In *Proceedings. 30th Euromicro Conference, 2004.*, pages 627–635, Aug 2004.

[43] Rob van Ommering. *Koala, a Component Model for Consumer Electronics Product Software*, pages 76–86. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[44] Rob van Ommering, Frank van der Linden, Jeff Kramer, and Jeff Magee. The koala component model for consumer electronics software. *Computer*, 33(3):78–85, March 2000.

[45] UML - Unified Modeling Language. http://www.uml.org/. Accessed: 2016-11-14.

[46] OMG - Object Management Group. http://www.omg.org/. Accessed: 2016-11-14.

[47] Grady Booch. *Object-oriented analysis and design with applications*. Addison-Wesley, Upper Saddle River, NJ, 2007.

[48] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. *Object-oriented modeling and design*, volume 199. Prentice-hall Englewood Cliffs, NJ, 1991.

[49] Ivar Jacobson. *Object-oriented Software Engineering*. ACM, New York, NY, USA, 1992.

[50] M. Emilia Cambronero, J. José Pardo, Gregorio Díaz, and Valentín Valero. Using rt-uml for modelling web services. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 643–648, New York, NY, USA, 2007. ACM.

[51] Jan Carlson, John Håkansson, and Paul Pettersson. Saveccm: An analysable component model for real-time systems. *Electronic Notes in Theoretical Computer Science*, 160:127 – 140, 2006.

[52] SysML - Systems Modeling Language. http://www.uml-sysml.org/. Accessed: 2016-11-14.

[53] INCOSE - International Council on Systems Engineering. http://www.incose.org/. Accessed: 2016-11-14.

[54] Ap232. http://homepages.nildram.co.uk/~esukpc20/exff2005_05/ap233/. Accessed: 2016-11-14.

[55] MARTE - Modeling and Analysis of Real Time and Embedded systems. http://www.omg.org/omgmarte/. Accessed: 2016-11-14.

[56] Global Harmonization Task Force. https://www.imdrf.org/ghtf. Accessed: 2021-01-03.

[57] FDA - Food and Drug Administration. https://www.fda.gov/. Accessed: 2019-09-29.

[58] World Health Organization - Medical Device Full Definition. https://www.who.int/teams/health-product-policy-and-standards/assistive-and-medical-technology/medical-devices. Accessed: 2021-01-03.

[59] World Health Organization et al. A stepwise approach to identify gaps in medical devices (availability matrix and survey methodology): background paper 1, august 2010. Technical report, Geneva: World Health Organization, 2010.

[60] M. Galarraga, L. Serrano, I. Martinez, P. de Toledo, and M. Reynolds. Telemonitoring systems interoperability challenge: An updated review of the applicability of iso/ieee 11073 standards for interoperability in telemonitoring. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6161–6165, Aug 2007.

[61] David Arney, Julian Goldman, Susan Whitehead, and Insup Lee. Synchronizing an x-ray and anesthesia machine ventilator - a medical device interoperability case study. pages 52–60, 01 2009.

[62] Nancy Leveson. A new accident model for engineering safer systems. *Safety Science*, pages 237–270, 2004.

[63] Rajeev Alur, David Arney, Elsa Gunter, Insup Lee, Jaime Lee, Wonhong Nam, Frederick Pearce, Steve Van Albert, and Jiaxiang Zhou. Formal specifications and analysis of the computer assisted resuscitation algorithm (cara) infusion pump control system. *STTT*, 5:308–319, 05 2004.

[64] Andrew L. King, Lu Feng, Sam Procter, Sanjian Chen, Oleg Sokolsky, John Hatcliff, and Insup Lee. Towards assurance for plug & play medical systems. In *Proceedings of the 34th International Conference on Computer Safety, Reliability, and Security - Volume 9337*, SAFECOMP 2015, pages 228–242, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[65] M. Lindvall, M. Diep, M. Klein, P. Jones, Y. Zhang, and E. Vasserman. Safety-focused security requirements elicitation for medical device software. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 134–143, Sep. 2017.

[66] Kevin Fu and James Blum. Controlling for cybersecurity risks of medical device software. *Biomedical Instrumentation & Technology*, 48(s1):38–41, 2014. PMID: 24848148.

[67] A. Mohan. Cyber security for personal medical devices internet of things. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 372–374, May 2014.

[68] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing*, 7(1):30–39, Jan 2008.

[69] Curtis R. Taylor, Krishna Venkatasubramanian, and Craig A. Shue. Understanding the security of interoperable medical devices using attack graphs. In *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, HiCoNS '14, pages 31–40, New York, NY, USA, 2014. ACM.

[70] Johannes Sametinger, Jerzy Rozenblit, Roman Lysecky, and Peter Ott. Security challenges for medical devices. *Commun. ACM*, 58(4):74–82, March 2015.

[71] Dan Kaplan. Black hat: Insulin pumps can be hacked. https://www.scmagazine.com/news/black-hat-insulin-pumps-can-be-hacked/black-hat-insulin-pumps-can-be-hacked, 2011. Accessed: 2019-09-29.

[72] Kim Zetter et al. Hacker can send fatal dose to hospital drug pumps. *Wired Magazine, Boone, IA, accessed Nov*, 9:2015, 2015.

[73] Medical devices - Application of risk management to medical devices. https://www.iso.org/obp/ui/#iso:std:iso:14971:ed-3:v1:en. Accessed: 2021-01-03.

[74] Lucian Leape and Donald Berwick. Five years after to err is human: what have we learned? *JAMA : the journal of the American Medical Association*, 293:2384–90, 06 2005.

[75] Medical devices - Application of risk management to medical devices. https://www.iso.org/obp/ui/#iso:std:iso:14971:ed-3:v1:en. Accessed: 2021-01-03.

[76] 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*, 2013:150143, 01 2013.

[77] K. Lesh, S. Weininger, J. M. Goldman, B. Wilson, and G. Himes. Medical device interoperability-assessing the environment. In *2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPnP 2007)*, pages 3–12, June 2007.

[78] IEEE Standard Computer Dictionary. https://ieeexplore.ieee.org/Xplore/home.jsp. Accessed: 2019-09-16.

[79] Hl7 - Health Level Seven. http://www.hl7.org/. Accessed: 2017-03-30.

[80] ISO - International Standards Organization. https://www.iso.org/home.html. Accessed: 2019-09-15.

[81] AAMI - Association for the Advancement of Medical Instrumentation. https://www.aami.org/. Accessed: 2019-09-16.

[82] HIMSS - Healthcare Information and Management Systems Society. https://www.himss.org/library/interoperability-standards/what-is-interoperability. Accessed: 2019-09-16.

[83] NEN - Dutch Standardization Institute. European Standardization of Health Informatics. CEN/TC251. http://www.frankfurt-am-main-hotels.com/centc251org/. Accessed: 2019-09-15.

[84] ASTM - American Society for Testing and Materials. https://www.astm.org/. Accessed: 2019-09-03.

[85] IHE - Integrating the Healthcare Entreprise. https://www.ihe.net/. Accessed: 2019-09-15.

[86] IEC - International Electrotechnical Commission. https://www.iec.ch/index.htm. Accessed: 2019-09-15.

[87] N. Georgi and R. Le Bouquin Jeannès. Proposal of a health monitoring system for continuous care. In *2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME)*, pages 1–4, Oct 2017.

[88] N. Georgi and R. Le Bouquin Jeannès. Wellness sensors and proprietary protocols, a solution for health monitoring? In *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 301–304, Feb 2017.

[89] A Stanford-Clark and A Nipper. MQTT - MQ Telemetry Transport. https://mqtt.org/, 2014. Accessed: 2019-09-15.

[90] MDPNP. Medical Device Plug-and-Play interoperability progam. https://mdpnp.org/, 2012.

[91] M. Görges, G. A. Dumont, C. L. Petersen, and J. M. Ansermino. Using machine-to-machine / "internet of things" communication to simplify medical device information exchange. In *2014 International Conference on the Internet of Things (IOT)*, pages 49–54, Oct 2014.

[92] RMI - Remote Method Invocation. http://www.oracle.com/technetwork/java/javase/overview/index-jsp-136424.html. Accessed: 2016-11-14.

[93] CORBA - Common Object Request Broker Architecture. http://www.corba.org/. Accessed: 2016-11-14.

[94] MDPAPP. Medical Device Plug And Play Program (MD, PnP) Clinical Scenario Repository. http://www.mdpnp.org/MD_PnP_Program___ClinicalSc.html, 2015. Accessed: 2019-09-01.

[95] W. Zhao. Towards trustworthy integrated clinical environments. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 452–459, Aug 2015.

[96] DDS - Data Distribution Service. https://www.omg.org/spec/#DDS, 2020. Accessed: 2019-09-16.

[97] J. Plourde, D. Arney, and J. M. Goldman. Openice: An open, interoperable platform for medical cyber-physical systems. In *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 221–221, April 2014.

[98] K. K. Venkatasubramanian, E. Y. Vasserman, O. Sokolsky, and I. Lee. Functional alarms for systems of interoperable medical devices. In *2014 IEEE 15th International Symposium on High-Assurance Systems Engineering*, pages 247–248, Jan 2014.

[99] Qais Tasali, Chandan Chowdhury, and Eugene Y. Vasserman. A flexible authorization architecture for systems of interoperable medical devices. In *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, SACMAT '17 Abstracts, pages 9–20, New York, NY, USA, 2017. ACM.

[100] Cinzia Bernardeschi, Andrea Domenici, and Paolo Masci. Towards a formalization of system requirements for an integrated clinical environment. In *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*, MOBIHEALTH'15, pages 46–49, ICST, Brussels, Belgium, Belgium, 2015. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[101] David Arney, Julian M. Goldman, Abhilasha Bhargav-Spantzel, Abhi Basu, Mike Taborn, George Pappas, and Michael Robkin. Simulation of medical device network

performance and requirements for an integrated clinical environment. *Biomedical Instrumentation & Technology*, 46(4):308–315, 2012. PMID: 22839991.

[102] Krishna K. Venkatasubramanian, Eugene Y. Vasserman, Vasiliki Sfyrla, Oleg Sokolsky, and Insup Lee. Requirement engineering for functional alarm system for interoperable medical devices. In Floor Koornneef and Coen van Gulijk, editors, *Computer Safety, Reliability, and Security*, pages 252–266, Cham, 2015. Springer International Publishing.

[103] HL7. Version 2. https://www.hl7.org/implement/standards/product/brief.cfm.productid=185, 2013. Accessed: 2019-09-03.

[104] HL7. Vesrion 3. https://www.hl7.org/implement/standards/product.brief.cfm.product.id=186, 2014. Accessed: 2019-09-03.

[105] Rune Fensli. Evaluation of international standards for ecg-recording and storage for use in tele-medical services. 2006.

[106] T. Hilbel, B. D. Brown, J. de Bie, R. L. Lux, and H. A. Katus. Innovation and advantage of the dicom ecg standard for viewing, interchange and permanent archiving of the diagnostic electrocardiogram. In *2007 Computers in Cardiology*, pages 633–636, Sep. 2007.

[107] M. Glass. Ansi/ieee 1073: Medical information bus (mib). *Health Informatics Journal*, 4(2):72–83, 1998.

[108] Stephan Pöhlsen, Stefan Schlichting, Markus Strähle, Frank Franz, and Christian Werner. A concept for a medical device plug-and-play architecture based on web services. *SIGBED Rev.*, 6(2):6:1–6:7, July 2009.

[109] OASIS. OASIS, Devices Profile for Web Services Version. http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html, 2009. Accessed: 2019-09-16.

[110] M. Kasparick, S. Schlichting, F. Golatowski, 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.

[111] National Institutes of Health et al. Electronic health records overview. 2006. the mitre corporation. mclean, virginia.

[112] Andrew L. King, Sam Procter, Daniel Andresen, John Hatcliff, Steve Warren, William Spees, Raoul Praful Jetley, Paul L. Jones, and Sandy Weininger. A publish-subscribe architecture and component-based programming model for medical device interoperability. *SIGBED Review*, 6:7, 2009.

[113] David W Bates and Asaf Bitton. The future of health information technology in the patient-centered medical home. *Health affairs*, 29(4):614–621, 2010.

[114] NIST GCR. Cost analysis of inadequate interoperability in the us capital facilities industry. *National Institute of Standards and Technology (NIST)*, pages 223–253, 2004.

[115] Bridget Moorman. Medical device interoperability: Overview of key initiatives. *Biomedical Instrumentation & Technology*, 44(2):132–138, 2010.

[116] CIMIT - Center for integration of medicine and innovative technology. https://www.cimit.org/. Accessed: 2016-11-14.

[117] Drager medical. Protocol definition, drager rs 232 medibus, December, 2007.

[118] Philips. Intellivue patient monitor user guide mp20/30, mp40/50, mp60/70/80/90, September, 2008.

[119] R. Marin, P. J. Sanz, P. Nebot, and R. Wirz. A multimodal interface to control a robot arm via the web: a case study on remote programming. *IEEE Transactions on Industrial Electronics*, 52(6):1506–1520, Dec 2005.

[120] David Arney, Julian M Goldman, Abhilasha Bhargav-Spantzel, Abhi Basu, Mike Taborn, George Pappas, and Michael Robkin. Simulation of medical device network performance and requirements for an integrated clinical environment. *Biomedical instrumentation & technology*, 46(4):308–315, 2012.

[121] David Arney, Julian M. Goldman, Susan F. Whitehead, and Insup Lee. *Improving Patient Safety with X-Ray and Anesthesia Machine Ventilator Synchronization: A Medical Device Interoperability Case Study*, pages 96–109. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[122] B. Andersen, M. Kasparick, H. Ulrich, S. Schlichting, F. Golatowski, D. Timmermann, and J. Ingenerf. Point-of-care medical devices and systems interoperability: A mapping of ice and fhir. In *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–5, Oct 2016.

[123] Julian M Goldman. Medical devices and medical systems - essential safety requirements for equipment comprising the patient centric integrated clinical environment (ice) - part1 : general requirements and conceptual model. *ASTM Draft*, 2008.

[124] David Arney, Kunal Bhatia, Sanchit Bhatia, Michael Sutton, Tracy Rausch, Joel Karlinsky, and Julian M Goldman. Design of an x-ray/ventilator synchronization system in an integrated clinical environment. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 8203–8206. IEEE, 2011.

[125] Hamed Soroush, David Arney, and Julian Goldman. Toward a safe and secure medical internet of things. 2016.

[126] Yu Jin Kim, John Hatcliff, Venkatesh-Prasad Ranganath, and Sandy Weininger. Integrated clinical environment device model: Stakeholders and high level requirements. 2015.

[127] T. L. Rausch and T. M. Judd. Using integrated clinical environment data for

health technology management. In *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 607–609, Feb 2016.

[128] David Arney, Sebastian Fischmeister, Julian M. Goldman, Insup Lee, and Robert Trausmuth. Plug-and-play for medical devices: Experiences from a case study. *Biomedical Instrumentation & Technology*, 43(4):313–317, 2009.

Abstract

Real time systems (RTS) are used in a wide variety of applications, ranging from very simple electronic systems to the most complex avionics and medical systems. RTS are time-related, which means the behavior of these systems depends on the timing requirements. Component based programming is the typical paradigm to develop RTS and this is due to it modular nature, which helps to hide the complexity of such systems and minimizes development costs. Medical monitoring systems in a high acuity patient environment are a good example of RTS, where the presence of more than one device in one clinical situation may affect patient safety because of the lack of communication between these devices and therefore interoperability is needed to avoid hazards and patients' health status degradation. The interconnection of medical devices in a wider network of computational servers is emerging as a new requirement in modern medicine, where the final goal is to achieve interoperability across medical devices and improve patient safety.

Résumé

Les systèmes temps réel (STR) sont utilisés dans une grande variété d'applications, allant des petits systèmes électroniques aux systèmes avioniques et médicaux les plus complexes. Les STR sont liés au temps, ce qui signifie que le comportement de ces systèmes dépend des contraintes temporelles. La programmation basée sur les composants est le paradigme typique pour développer les STR et cela est dû à la nature modulaire de ces systèmes, chose qui aide à masquer la complexité et minimiser les coûts de développement. Les systèmes de surveillance médicales dans un environnement de haute acuité sont un bon exemple des STR, où la présence de plus d'un dispositif médical dans une situation clinique peut affecter la sécurité du patient en raison du manque de communication entre ces dispositifs, pour cela, l'interopérabilité est nécessaire pour éviter la dégradation de l'état de santé des patients. L'interconnexion des dispositifs médicaux dans un réseau plus large de serveurs informatiques est la nouvelle exigence de la médecine moderne, où l'objectif final est de réaliser l'interopérabilité entre les dispositifs médicaux et d'améliorer la sécurité des patients.

ملخص

تُستخدم أنظمة الوقت الفعلي (أ.و.ف) في مجموعة متنوعة من التطبيقات، بدءًا من الأنظمة الإلكترونية البسيطة الى الأنظمة الطبية و أنظمة الطيران الأكثر تعقيدًا. أنظمة الوقت الفعلي مرتبطة بالوقت، مما يعني أن سلوك هذه الأنظمة يعتمد على متطلبات التوقيت. تعتبرالبرمجة القائمة على المركباتية أفضل نموذج لتطويرأنظمة الوقت الفعلي وهذا يرجع إلى طبيعتها المعيارية، مما يساعد على إخفاء تعقيد هذه الأنظمة وتقليل تكاليف التطوير. تعد أنظمة المراقبة الطبية لمريض في حالة سريرية مثالًا جيدًا على أنظمة الوقت الفعلي، حيث قد يؤثر وجود أكثر من جهاز طبي واحد في هذه الحالة على سلامة المريض بسبب نقص الاتصال بين هذه الأجهزة، وبالتالي فإن التشغيل التوافقي ضروري لتجنب المخاطر و تدهور الحالة الصحية للمرضى. يظهر الترابط بين الأجهزة الطبية في شبكة أوسع من الخوادم الحاسوبية كمتطلب جديد في الطب الحديث، حيث يتمثل الهدف النهائي في تحقيق إمكانية التشغيل التوافقي للأجهزة الطبية وتحسين سلامة المرضى.