Pontifícia Universidade Católica
do Rio de Janeiro

**Chrystinne Oliveira Fernandes**

# An Architecture for E-Health Systems that supports Patient Monitoring and Caregivers Notification based on a Reasoning Model to avoid Alarm Fatigue



**Tese de Doutorado**

Thesis presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática.

Advisor: Prof. Carlos José Pereira de Lucena

Rio de Janeiro

December 2019


**Chrystinne Oliveira Fernandes**

# An Architecture for E-Health Systems that supports Patient Monitoring and Caregivers Notification based on a Reasoning Model to avoid Alarm Fatigue

Thesis presented to the Programa de Pós-graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática. Approved by the Examination Committee.

**Prof. Carlos José Pereira de Lucena**
Advisor
Departamento de Informática – PUC-Rio

**Profª. Simone Diniz Junqueira Barbosa**
Departamento de Informática – PUC-Rio

**Prof. Jorge Calmon de Almeida Biolchini**
PUC-Rio

**Prof. Ricardo Choren Noya**
IME

**Prof. Jean-Pierre Briot**
CNRS

Rio de Janeiro, December 16th, 2019

**Chrystinne Oliveira Fernandes**

The author received her Bachelor degree in Computer Science from the Universidade Federal da Paraíba (UFPB) in 2006. She also received her Master degree in Computer Science from PUC-Rio in 2015. During her academic career, she participated in several research projects. Since April 2014, she acts as a Researcher and Developer at the Software Engineering Laboratory (LES), working on projects involving the development of innovative e-Health solutions in Internet of Things, Multi-agent Systems, Machine Learning and Data Science fields. Her main research interests are: Software Engineering, E-Health Systems, Artificial Intelligence and Data Science.

Bibliographic data

Fernandes, Chrystinne Oliveira

An Architecture for e-health systems that supports patient monitoring and caregivers notification based on a reasoning model to avoid alarm fatigue / Chrystinne Oliveira Fernandes ; advisor: Carlos José Pereira de Lucena. – Rio de Janeiro : PUC-Rio, Departamento de Informática, 2019.

125 f. : il. color. ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2019.

Inclui referências bibliográficas

1. Informática – Teses. 2. Fadiga de alarme. 3. Sistemas de saúde eletrônicos. 4. Monitoramento de pacientes. 5. Sistemas de alerta. 6. Inteligência artificial. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro.

CDD: 004

## Acknowledgements

To my advisor, Professor Carlos Lucena, for all his support, and for the opportunities that he gave me during my Master's and Ph.D. I feel honored to have an advisor that is a reference in the Computer Science field and such an amazing and inspiring person. After working with him for 6 years, I can confirm that he deserves every nice word that I heard about him during this journey.

My sincere gratitude to my external advisor from Kings College London, Dr. Simon Miles - who is a brilliant researcher -, for all his feedback, guidance, and patience. I could not have had better support during my period abroad. (My sincere thanks also to Ingrid Nunes, who introduced me to him).

My deep gratitude to all the professors of the Department of Informatics from PUC-Rio for the inspiration and knowledge that I acquired during this journey, especially Professors Hugo Fuks, Noemi Rodriguez, Simone Barbosa, Hélio Lopes, Ruy Milidiú, Eduardo Laber, and Alessandro Garcia. Many thanks also to Guido Lemos, from UFPB.

I thank also João Oliveira, Pergentino Alves, Edson Sales, and Renato Crivano for all their support.

I would like to thank also my family, friends, colleagues from KCL and PUC-Rio (especially Leonardo Sousa), and my English teacher Marta.

Finally, I thank the PUC-Rio, LES, CNPq, and FAPERJ for their support and funding of this research.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

# Abstract

Fernandes, Chrystinne Oliveira; Lucena, Carlos José Pereira de (Advisor). **An Architecture for E-Health Systems that supports Patient Monitoring and Caregivers Notification based on a Reasoning Model to avoid Alarm Fatigue.** Rio de Janeiro, 2019. 125p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Estimates show that 80% to 99% of alarms set off in hospital units are false or clinically insignificant, representing a cacophony of sounds that do not present a real danger to patients. These false alarms can lead to an alert overload that causes a health care provider to miss important events that could be harmful or even life-threatening. As health care units become more dependent on monitoring devices for patient care purposes, the alarm fatigue issue has to be addressed as a major concern in order to prevent healthcare providers from undergoing alarm burden, as well as to increase patient safety. The main goal of this thesis is to propose a solution for the alarm fatigue problem by using an automatic reasoning mechanism to decide how to notify members of the health care team. Our specific goals are: to reduce the number of notifications sent to caregivers; to detect false alarms based on alarm-context information; to decide the best caregiver to whom a notification should be assigned. This thesis describes: a model to support reasoning algorithms that decide how to notify caregivers in order to avoid alarm fatigue; an architecture for health systems that supports patient monitoring, reasoning and notification capabilities; and three reasoning algorithms that decide: (i) how to notify caregivers by deciding whether to aggregate a group of alarms; (ii) whether, or not, to notify caregivers with an indication of a false alarm probability; (iii) who is the best caregiver to notify considering a group of caregivers. Experiments were used to demonstrate that by providing a reasoning system that aggregates alarms we can reduce the total of notifications received by the caregivers by up to 99.3% of the total alarms generated. These experiments were evaluated through the use of a dataset comprising real patient monitoring data and vital signs recorded during 32 surgical cases where patients underwent anesthesia at the Royal Adelaide Hospital. We present the results of this algorithm

by using graphs generated with the R language, which show whether the algorithm decided to deliver an alarm immediately or after a given delay. We also achieved the expected results for our reasoning algorithm that handles the notifications assignment task, since the algorithm prioritized the caregiver that was available and was the most experienced and capable of attending to the notification. The experimental results strongly suggest that our reasoning algorithms are a useful strategy to avoid alarm fatigue. Although we evaluated our algorithms in an experimental environment, we tried to reproduce the context of a clinical environment by using real-world patient data. As future work, we aim to evaluate our algorithms using more realistic clinical conditions by increasing, for example, the number of patients, monitoring parameters, and types of alarm.

## Keywords

Alarm Fatigue; E-Health Systems; Patient Monitoring; Alert Systems; Artificial Intelligence.

# Resumo


Fernandes, Chrystinne Oliveira; Lucena, Carlos José Pereira de. **Uma Arquitetura para Sistemas de Saúde Eletrônicos que Suporta o Monitoramento de Pacientes e a Notificação de Cuidadores com base em Raciocínio Automático para evitar a Fadiga de Alarme.** Rio de Janeiro, 2019. 125p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Estimativas informam que 80% a 99% dos alarmes disparados em unidades hospitalares são falsos ou clinicamente insignificantes, representando uma cacofonia de sons que não apresenta perigo real aos pacientes. Estes falsos alertas podem culminar em uma sobrecarga de alertas que leva um profissional da saúde a perder eventos importantes que podem ser prejudiciais aos pacientes ou até mesmo fatais. À medida que as unidades de saúde se tornam mais dependentes de dispositivos de monitoramento que acionam alarmes, o problema da fadiga de alarme deve ser tratado como uma das principais questões, a fim de prevenir a sobrecarga de alarme para os profissionais da saúde e aumentar a segurança do paciente. O principal objetivo desta tese é propor uma solução para o problema de fadiga de alarme usando um mecanismo de raciocínio automático para decidir como notificar os membros da equipe de saúde. Nossos objetivos específicos são: reduzir o número de notificações enviadas à equipe de cuidadores; detectar alarmes falsos com base em informações de contexto do alarme; decidir o melhor cuidador a quem uma notificação deve ser atribuída. Esta tese descreve: um modelo para suportar algoritmos de raciocínio que decidem como notificar os profissionais de saúde para evitar a fadiga de alarme; uma arquitetura para sistemas de saúde que suporta recursos de monitoramento, raciocínio e notificação de pacientes; e três algoritmos de raciocínio que decidem: (i) como notificar os profissionais de saúde decidindo quando agrupar um conjunto de alarmes; (ii) se deve ou não notificar os profissionais de saúde com uma indicação de probabilidade de falso alarme; (iii) quem é o melhor cuidador a ser notificado considerando um grupo de cuidadores. Experimentos foram realizados para demonstrar que, ao fornecer um sistema de raciocínio que agrupa alarmes semelhantes e recorrentes, pode-se reduzir o total de notificações recebidas pelos

cuidadores em até 99.3% do total de alarmes gerados, sem perda de informação útil. Esses experimentos foram avaliados através do uso de um conjunto de dados reais de monitoramento de sinais vitais de pacientes registrados durante 32 casos cirúrgicos nos quais os pacientes foram submetidos à anestesia, no hospital Royal Adelaide. Apresentamos os resultados desse algoritmo através de gráficos gerados na linguagem R, onde mostramos se o algoritmo decidiu emitir um alarme imediatamente ou após um determinado delay. Para a tarefa de atribuição de notificações realizada pelo nosso algoritmo de raciocínio que decide sobre qual cuidador notificar, também alcançamos nossos resultados esperados, uma vez que o algoritmo priorizou o cuidador que estava disponível no momento do alarme, além de ser o mais experiente e capaz de atender à notificação. Os resultados experimentais sugerem fortemente que nossos algoritmos de raciocínio são uma estratégia útil para evitar a fadiga de alarme. Embora tenhamos avaliado nossos algoritmos em um ambiente experimental, tentamos reproduzir o contexto de um ambiente clínico utilizando dados reais de pacientes. Como trabalho futuro, visamos avaliar os resultados de nossos algoritmos utilizando condições clínicas mais realistas, aumentando, por exemplo, o número de pacientes, os parâmetros de monitoramento e os tipos de alarme.



## Palavras-chave

Fadiga de Alarme; Sistemas de Saúde Eletrônicos; Monitoramento de Pacientes; Sistemas de Alerta; Inteligência Artificial.

## Summary

# List of Figures

# List of Tables

## List of Abbreviations

AF: Alarm Fatigue

AI: Artificial Intelligence

BP: Buffering Period

CSP: Constraint-Satisfaction Problem

CSV: Comma-Separated Values

DoA: Depth of Anesthesia

EEG: Encephalography

FAI: False Alarm Indicator

FAP: False Alarm Probability

GA: General Anesthesia

GPS: Global Position System

ICU: Intensive Care Unit

IT: Information Technology

IoT: Internet of Things

MAS: Multi-Agent Systems

MNI: Minimum Notification Interval

NP: Notification Period

OR: Operating Room

POCD: Postperative Cognitive Dysfunction

POD: Postoperative Delirium

RQ: Research Question

SQ: Sub-Question

# 1
# Introduction

Information Technology (IT) has already provided significant benefits to the healthcare sector, but there are still many areas where the application of IT could offer further critical improvements. For example, a worldwide hospital problem nowadays is the alarm fatigue (Cvach, 2012), which has recently been receiving attention from the industry, the healthcare sector and the academic community.

Alarm fatigue involves a lack of response due to an excessive number of non-critical alarms being received by healthcare personnel, resulting in sensory overload and desensitization[1] (Cvach, 2012). To illustrate the severity of this problem, in Canada and the United States, where this issue has been treated as a major patient safety concern (Jones, 2014), it was reported that during a 12-day period of analysis of the alarm system at The Johns Hopkins hospital in Baltimore, there was an average of 350 alerts per bed per day. In fact, in one Intensive Care Unit (ICU), the average was 771 alerts per bed per day. Such numbers indicate a severe sensory overload for the healthcare staff, with serious consequences for the well-being of the patients if an alarm is ignored.

In this thesis, we present a new approach to cope with the alarm fatigue problem, its most common causes, adverse consequences, and strategies as compared with other solutions published in the literature. Our proposed solution for addressing this issue uses an artificial intelligence approach based on an automatic reasoning system that decides how to notify caregivers about anomalies detected by a patient monitoring system where a large volume of alarms could lead to alarm fatigue. In other words, we are using information technology to assess the validity of alarms and to notify the most suitable member of the healthcare staff about the alarms that truly need attention.

This chapter presents our problem definition, motivation, goals, research questions, main contributions and the thesis organization.

[1] Reduction or elimination of individual´s negative reaction to stimulus (Myles et al., 2007).

## 1.1
## Problem Definition

A concern in a great number of hospitals and a complaint among possibly every medical team that works with physiologic monitors, alarm fatigue is an undesired sensory overload experienced by physicians and other health providers. This problem is caused by the excessive numbers of alerts generated from physiologic monitor devices in audible or visual form. Alarm fatigue represents a substantial issue that can bring undesired consequences to healthcare environments. For instance, we can cite the desensitization of the medical team in relation to alerts, which can lead to longer response times to handle the anomalies as well as the overlook of critical events. These examples illustrate unintended behaviors in the context of a sensory overload that may culminate in an unsafe patient environment.

## 1.2
## Motivation, Goals and Research Questions (RQ)

Estimates show that 80% to 99% of alarms set off in hospital units are false or clinically insignificant. Such alarms represent a cacophony of sounds that do not even represent a real danger to patients. Consequently, these false alarms can lead to alarm burden and can compromise the health providers' attention (Cvach, 2012; Drew et al., 2014; Jones, 2014; Tanner, 2013). For example, they can lead health providers to miss relevant alarms that announce significant, harmful, or life-threatening events. As the healthcare units become more dependent upon physiologic monitoring devices used for patient care, the alarm fatigue issue has to be addressed as a major concern. In this case, feasible strategies need to be provided to prevent the alarm burden for the healthcare providers, as well as to increase patient safety.

Our main goal in this thesis is to propose a solution to contribute to the mitigation of the alarm fatigue problem. The proposed solution relies on an automatic reasoning mechanism to decide how to notify health providers. We aim at reasoning about whether to notify caregiver teams with an indication of a False Alarm Probability (FAP). The FAP label added to the notification can help the caregivers to prioritize their work, especially when they are under alarm fatigue conditions. As another contribution of our work, we aim to reduce the bedside

monitor's alarm burden by grouping a set of identical (or too similar) notifications. Finally, we also aim at reasoning about which caregiver to notify, considering their degree of experience, availability, geolocation, and current workload circumstances.

To pursue our main goal, we defined the following Research Question:

> **RQ.** How can an automatic reasoner determine how to best notify caregivers about anomalies detected by a patient monitoring system where a large volume of notification leads to alarm fatigue?

Because of its complexity, we divided our main research question in 4 Sub-Questions (SQ) as follows:

> **SQ1.** How to reason about whether to group alarms to avoid alarm fatigue?
>
> **SQ2.** How can an automatic reasoning system calculate an indication of FAP for an alarm generated by sensors and monitoring devices?
>
> **SQ3.** How to reason about whether, or not, to add an indication of FAP to a notification that could be visualized by the healthcare team?
>
> **SQ4.** How to reason about who to notify within the caregiver teams?

We conducted three case studies in order to answer our sub-questions. The results of each case study, which comprise a reasoning algorithm and its evaluation, are presented as individual chapters, later on.

By answering the sub-questions, we provided examples of strategies that can be utilized to notify caregivers in order to mitigate alarm fatigue. These strategies were combined into the notification model that composes the architecture provided in this thesis.



## 1.3 Main Contributions

In this work, we try to fill the gap of having feasible solutions to mitigate the alarm fatigue problem by focusing on the issues of excessive alarms and false positive alarms that are known to be a serious problem that still remains unsolved. As a strategy to mitigate the alarm fatigue issue, in this thesis we present a new approach to monitor patients by using an intelligent notification process supported by a reasoning mechanism that mainly assists health providers in deciding: 1- whether to group a set of alarms that occurs within a short period of time in order to deliver them together; 2- whether to include an indication of FAP to a notification

that can be visualized by caregivers in order to assist them in their decision-making process about which alarm to prioritize next; 3- who the best caregiver is to receive the notification within a set of caregivers, based on real-time circumstances in an ICU - including information about alarms, caregivers' and patients' circumstances.

Below, we list our main contributions:

**C1.** An architecture well suited for health systems that supports patient monitoring, reasoning and notification capabilities;

**C2.** A model to support reasoning algorithms that decide how to best notify caregivers to avoid alarm fatigue;

**C3.** A reasoning algorithm that specifies how to notify caregivers by grouping a set of alarms;

**C4.** A reasoning algorithm that decides whether to notify with an indication of a false alarm probability;

**C5.** A reasoning algorithm that decides who to notify considering a group of caregivers.

The contributions C1 and C2 are related to our main Research Question RQ1, while the contribution C3 is associated to the Sub-Question SQ1, the contribution C4 is related to the Sub-Questions SQ2 and SQ3 and, finally, the contribution C5 is associated to the SQ4.



## 1.4 Thesis Organization

This thesis is organized as follows:

Chapter 2 presents a literature review. As related work, we present studies that provide strategies to deal with the alarm fatigue problem and we discuss how these works are distinct from our approach.

Chapter 3 presents: (i) our proposed architecture for health systems that supports patient monitoring, reasoning and notification processes; (ii) a more formal description of the main features of these systems, e.g., anomaly detection, alarm triggering, and notification; (iii) the Alert Fatigue-aware Notification Model, the model we developed to support our reasoning algorithms.

Chapter 4 presents examples of applications we built by following the architecture to show the implementation of data collection, data visualization and data analysis features.

Our main contributions are presented in Chapters 5, 6 and 7, where we describe our reasoning algorithms to solve our main research question and sub-questions, along with their evaluations.

Finally, we conclude this thesis and present possible future work in Chapter 8.

# 2
# Background and Related Work

A critical concern in hospitals that use monitoring devices to track patient health is alarm fatigue. Tens of thousands of alerts may go off throughout a hospital each day, and yet the majority of these audible or visual alerts are false or nuisance alarms, indicating conditions that do not require clinical intervention (Cvach, 2012; Drew et al., 2014; Jones, 2014; Tanner, 2013). Alarm fatigue represents a substantial issue that can bring undesired consequences to health care environments. For instance, the desensitization of a health care team to alerts can lead to longer response times for handling anomalies as well as possibly missing life-threatening events. These examples illustrate the fact that sensory overload is very likely to produce an unsafe environment for patients.

According to Sowan et al., the key issues causing alarm fatigue and decrease in trust of alarm systems are as follows: the high incidence of nuisance alarms, the confusion in locating the device sending out the alarm, unit layouts that hinder alarm response, the inadequacy of alarm systems to alert nurses of changes in patients' conditions, and the complexity of new monitoring systems, among others. The most important issues interfering with alarm recognition and alarm response ranked by the nurses cited by Sowan et al. were as follows: (1) frequent false alarms, (2) difficulty in understanding alarm priorities, and (3) noise competition from nonclinical devices.

Caring for patients and managing alarms simultaneously is a very complex and demanding task, especially when health providers are caring for multiple patients at the same time and have been exposed to a high number of alarms generated by physiological monitors. In addition to dealing with frequent alarms, health care providers also perform other activities, such as medication administration, patient assessments, and note updates. Over time, they become fatigued and errors may occur because of decreased attentiveness (Shanmugham, Strawderman, Babski-Reeves, & Bian, 2018).

Considering the aforementioned scenario, a commonly recommended solution to mitigate alarm fatigue is to adjust alarm parameters on monitors to suit each patient's condition rather than using default settings (Shanmugham et al., 2018). The works of Shanmugham et al. and Sowan et al. are examples of studies that assess the effect of modifying the default alarm settings provided by the device manufacturers. According to their findings, the nurses' perceived workload was lower when the clinical alarm threshold limits were modified according to the patients' clinical conditions. They also concluded that the modification of alarm settings affects the number of alarms accurately addressed, care providers' experience, and overall satisfaction.

Another strategy suggested by Sowan et al. to reduce the number of false alarms and alarm fatigue is educating staff regarding alarm management. The authors showed that their changes in default alarm settings significantly reduced 24% of the total number of the target alarms after their interventions, which included the following: (1) re-education of ICU bedside nurses on the appropriate use of the monitors, and (2) change of default settings of some parameters on the cardiac monitors - including the addition of an alarm delay by increasing the period between the alarm detection and its triggering.

However, despite the achievement of a significant reduction in the alarm rate, they deem that the change of default settings and better education regarding cardiac monitors are insufficient to improve alarm system safety.

Scientific studies show that the quality of medical device alarms is unsatisfactory, and it affects quality of care and patient safety. One root cause is the poor quality of alarm-generating algorithms. Therefore, from a clinical perspective, major improvements in alarm algorithms are urgently needed (Imhoff, Kuhls, Gather, & Fried, 2009).

To pursue this goal, different methods have been proposed and investigated for use in the alarm systems of medical devices, mostly from the fields of statistics and Artificial Intelligente (AI). Imhoff et al. gave a brief overview of different methods, including statistical approaches and AI methods.

Regarding the methodological approaches to alarm management, Imhoff et al. present the 4 areas in which alarms can be improved: (1) signal acquisition, that

is, the interface between patient and medical devices; (2) alarm generation, that is, the algorithms that determine an alarm situation; (3) alarm validation, that is, determining whether the alarm is actually valid; and (4) integration of multiple alarms, for example, from different devices, into 1 or few alarms.

Successful quality improvement approaches included alteration in default monitor presets, daily electrode change, alarm customization, alarm management education, change in policy, histogram-based pulse oximetry (SpO2), alarm tailoring, improved displays to aid nurse-patient assignments, and the use of notification delays (Winters et al., 2018). Notification delays are performed with a middleware situated between the alarming medical device and the clinicians' receiver equipment such as a mobile phone. Several studies found that introducing alarm delays prior to the notification process could decrease "false alarms" by 25%–67% (Winters et al., 2018). Regarding the reduction of the total alarms, considering the effects of these interventions, alarm quantities decreased between 18.5% and as much as 89%, according to Winters et al..



## 2.1
## Alarms and the Impact of Alarm Safety in Patient Care

Alarms are utilized to improve patient safety and quality of care, by detecting changes early and requiring appropriate action. However, the medical literature contains many studies that show that up to 90% of all alarms in critical-care monitoring are false positives. The vast majority of all threshold alarms in the ICU does not have a real clinical impact on the care of the critically ill (Imhoff et al., 2009).

Many studies have recorded the number of alerts being triggered nowadays in intensive care units during a period of time in order to analyze the impact of alarm safety in patient care as a consequence of the excessive volume of alarms. For instance, Lawless analyzed alarm soundings that occured in an ICU during a 7-day period, recorded by ICU staff (Lawless, 1994). In his experiments, he categorized alarms into three types: false, significant (resulted in change in therapy), or induced (by staff manipulations; not significant). He showed that, within the total of 2,176 alarm soundings, 1,481 (68%) were false, 119 (5.5%) were significant and 576 (26.5%) were induced. His results showed that over 94% of

alarm soundings in a pediatric ICU may not be clinically important. Based on his findings, the author concluded that current monitoring systems are poor predictors of untoward events.

In addition to the excessive number of alarms in ICUs, another alarm-related problem presented by Sendelbach & Funk is the high number of different alarm signals in an Operating Room (OR). In 1983, each patient in an ICU could have up to 6 alarms. By 1994, up to 33 different alarms were identified, and by 2011, this number increased to over 40 different alarm signals in an ICU (Sendelbach & Funk, 2013). Currently, this number can reach 120 separate alarm devices in an OR that are stand-alone, uncorrelated, and non-prioritized (Sendelbach & Funk, 2013).

The main problem of having so many different devices triggering alarms is that it is not feasible for nurses to identify all of them, which means that this increase has occurred despite staff having difficulty in learning all available alarm signals in their work environment. The work of Shanmugham et al. pointed out this problem, showing that nurses have difficulties in learning more than six different alarm signals. Therefore, in addition to the excessive number of alarms, staff can only identify between 9 to 14 out of the 23 alarms found in the OR, and, on average, 10 to 15 out of 26 alarms triggered in the ICU, which contributes to the alarm overload problem.

According to Shanmugham et al., Kerr & Hayes recognized that the excessive number and many diverse types of alarms were bringing adverse consequences to patient care, such as: (i) the reduction of the effectiveness of alarms, (ii) creation of confusion and distraction for caregivers, who were having difficulties in responding to alarms, and (iii) the deterioration of patient care, putting patients in a more unsafe environment.

At last, a third alarm-related problem we are focusing on this thesis is the excessive number of false alarms. Studies have indicated that false and/or clinically insignificant alarms range from 80%-99% (Cvach, 2012). False alarms are frequently triggered by erroneous or absent patient data (Tanner, 2013). These types of alarms can be caused by events such as patient movement or repositioning in bed and by poor placement of sensors, such as pulse oximeter.

Along with the already mentioned alarm-related problems that can affect patient care, there is more information in ICUs that are considered critical for the healthcare team, such as: (i) the perceived alarm urgency, and (ii) the perceived true alarm rate of the alarm system (Tanner, 2013). Tanner showed that perceived alarm urgency contributes to the nurses' alarm response; but nurses also use additional strategies to determine response, including the criticality of the patient, signal duration, uncommonness of the alarming device, and workload.

Regarding the perceived true alarm rate of the alarm system, an important finding of the work of Tanner is the link between the impact of the perceived true alarm rate of the alarm system by caregivers and its influence in patient care. The author showed that the nurses' response to alarms follows the perceived true alarm rate of the alarm system. According to the author, if the true alarm rate is perceived to be 10% reliable, then the response rate will be about 10%.

Although alarm safety is a critical issue that needs to be addressed to improve patient care, the hospitals have not given serious consideration on how its staff should be using, setting and responding to clinical alarms, according to the ECRI Institute (ECRI Institute; Keller, 2012). Currently, this complex and overwhelming scenario is still a problem that culminated in an unsolved health problem known as Alarm Fatigue, which we next describe.



## 2.2 Alarm Fatigue

By definition, Alarm Fatigue consists of the lack of response due to excessive numbers of alarms in hospital environments, especially in ICUs, resulting in sensory overload and desensitization (Cvach, 2012). This issue has the potential to compromise patient safety (Keller, 2012), since frequent alarms are distracting and interfere with a clinician's performance of critical tasks. Excessive false positive alarms may lead to apathy, resulting in less likelihood that real events may be acted on. The presence of medical devices generate enough false alarms to cause a reduction in responses, leading to a scenario in which caregivers disable, silence and/or ignore the alarms (Keller, 2012) or are slow to respond (Cvach, 2012; Kerr & Hayes, 1983).

In Table 1, we summarized the information we presented about alarm-related issues, comprising its causes, consequences to the staff, consequences to the patient's care and avoidance strategies (Cvach, 2012).

Table 1- A summary of alarm-related issues.

| Alarm-related issues | Causes | Consequence to the staff | Consequence to the patient care | Avoidance strategies |
|---|---|---|---|---|
| Excessive False Positive Alarms (FPAs) | Can be attributed to patient manipulation (motion artifact) | Apathy and desensitization;<br><br>Mistrust | Reduction in responding;<br>Lack of caregiver response;<br>Real events being less likely to be acted on | Suspension of alarms for a short period prior to patient manipulation;<br>Statistical methods should be suitable to decrease the number of FPAs |
| Frequent insignificant or irrelevant alarms | Use of the default alarm settings;<br><br>Poor staff education on alarm management | Distraction;<br><br>Reduction in trust | Disruption of patient care;<br><br>Disabling of alarm systems by staff | Eliminating nonessential alarms;<br>Adjusting alarm parameters on monitors to suit patients' conditions;<br>Staff education on alarm management |

## 2.3
## Statistical and AI-related Approaches

According to Imhoff et al., the quality of medical device alarms is unsatisfactory, affecting quality of care and patient safety. Since the low quality of alarm-generating algorithms is one of the main causes of this problem, major improvements in alarm algorithms are urgently needed (Imhoff et al., 2009).

To achieve this goal, a variety of alarm suppression algorithms have been developed and successfully applied in laboratory and the clinical environment to avoid alarm fatigue, such as: relevance vector machine learning, statistical metrics, time series analysis, spectral regression, feature selection, and other classifiers (Winters et al., 2018). Imhoff et al. showed different methods that have been proposed for use in the alarm systems of medical devices, including statistical approaches, such as: improved data preprocessing, robust signal extraction,

segmentation, median filter, statistical process control, and time series analysis for pattern detection, among others. AI methods also have been investigated, and includes approaches based on machine learning, neural networks, random forests, fuzzy logic, and Bayesian networks.

Another strategy to avoid alarm fatigue is to use notification delays that are performed through the use of a middleware between the alarming medical device and the clinicians' receiver device such as a mobile phone or a tablet. Several studies found that introducing alarm delays before notifying caregivers could decrease "false alarms" by 25%–67% (Winters et al., 2018). Regarding the reduction of the total alarms, considering the effects of these interventions, alarm quantities decreased between 18.5% and as much as 89%, according to Winters et al..

Other examples of promising proposed approaches are the application of contextuality, and the integration of alarms to create smart alarms with improved data presentation through human factors engineering (Winters et al., 2018).

According to Imhoff et al., one of the main areas in which alarms can be improved is alarm validation. In this thesis, we are given our contribution to the alarm validation area in Chapter 6. As our methodological approach to deal with alarm validation, we try to fill the gap of having feasible solutions for mitigating the alarm fatigue problem by focusing on the issue of false positive alarms that is known to be a serious problem that still remains unsolved.

# 3
# System Architecture and The Alarm Fatigue-aware Notification Model

Before presenting our reasoning algorithms, we outline important concepts of the monitoring and notification processes we developed to cope with remote patient monitoring. In this chapter, we illustrate a more formal description of the components of the architecture that will be used in our systems, e.g., the anomaly detection, the alarm-triggering, and the notification processes. We cover all the features of our architecture with the Reasoning module that is introduced in this chapter but is explained in more detail in Chapters 5, 6 and 7.

## 3.1.
## System Architecture

The default function of our notification system is to notify a group of caregivers about anomalies detected in a patient's vital signs. The anomaly detection process works through continuous monitoring of each patient's vital signs using data acquired from sensors (P_DATA). To verify if an anomaly occurs, the readings are evaluated against anomaly thresholds configured for each patient. If a reading for a patient is above a maximum or below a minimum threshold value, then the reading is considered to be anomalous and the system triggers an alarm that is sent to the healthcare team. The anomaly detection process and its related concepts such as anomalies, alarms and notifications, are more formally defined in the next subsections.

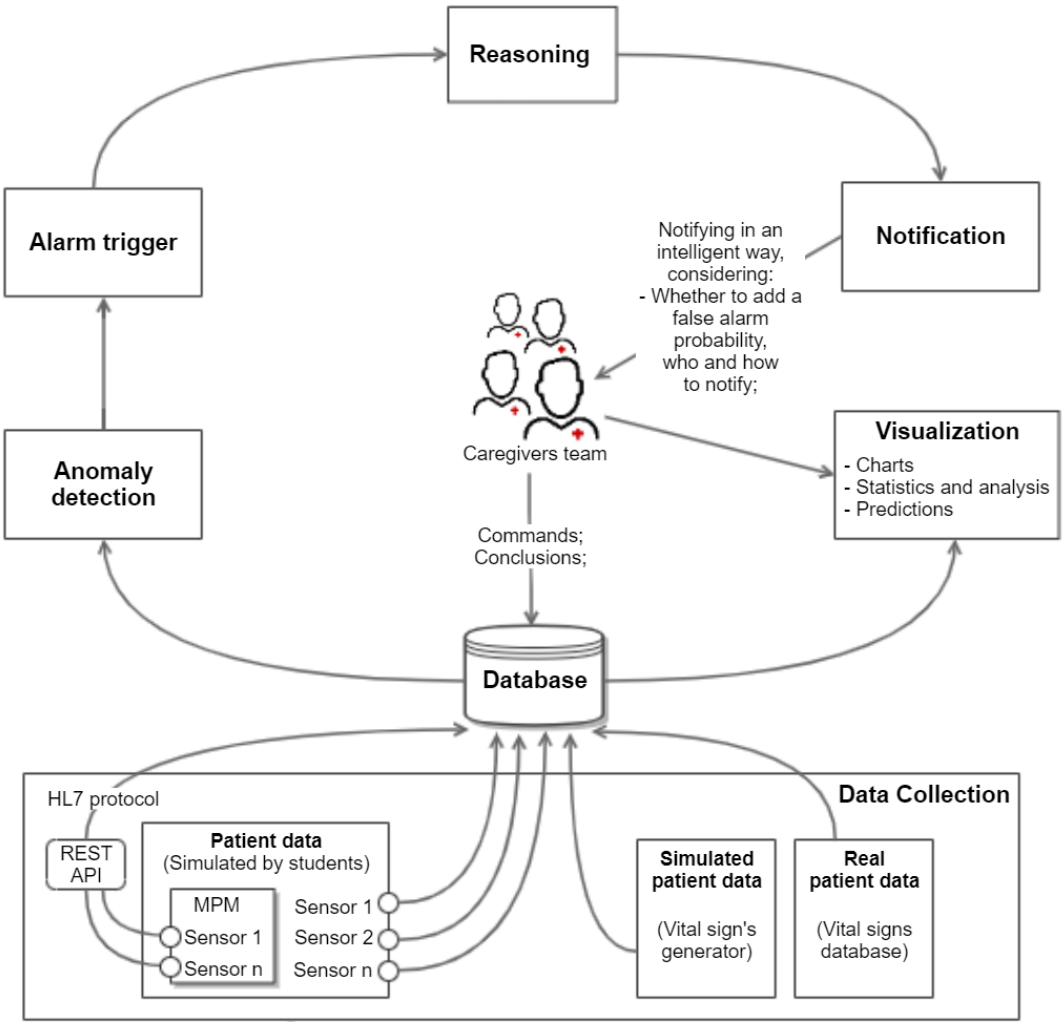



Figure 1 - An architecture designed for health care systems that support patient monitoring and notification capabilities. MPM: Multi-parametric Monitor; API: Application Programming Interface.

### 3.1.2
### Defining Thresholds and Anomalous Values

Anomaly thresholds for the sensors must be configured before starting to monitor a patient. A threshold is a minimum and maximum limit for a reading of a sensor S for a patient P, and an anomaly is a value either below or above those limits. An anomaly or anomalous value $v \in AV(S,P)$ triggers an alarm that is sent to the health care team. The threshold value for sensor S connected to a patient P is designated threshold(S,P) and the minimum and maximum values are v_min(S,P) and v_max(S,P), respectively. We formally defined anomalies using set theory as shown later.

Let AV(S,P) be the set of values that represent patient P's anomalous values for the sensor S. Let us also consider that these values from S belong to the set of real numbers. The AV(S,P) set is formally defined as shown in Equation (1):

$$AV(S,P) = \{ v \mid v \in \mathbb{R}, v\_min_{S,P} > v > v\_max_{S,P}\} \quad (1)$$

Where:

- the inequalities $v < v\_ min_S$ and $v > v\_ max_{S,P}$ comprise the thresholds for sensor S and patient P;
- $v\_min_{S,P} \in R$, which represents the minimum limit, that is, the value below which a sensor reading v is considered an anomalous value;
- $v\_max_{S,P} \in R$, representing the maximum limit, that is, the value above which v is considered an anomaly.

We can define an anomaly detected by sensor as the function An(v) = b that maps real numbers into Booleans ($f : R \rightarrow Boolean$) where $v \in R$ and is the value that represents a sensor reading, and b={true, false} as shown below.

$$An(v) = true, \text{ if } v \in AV_{S,P}; \text{ false, otherwise.} \quad (2)$$

### 3.1.3 Defining Alarm, Anomaly Detection, and Notification Events

In our system, we define the concepts of anomaly detection, alarm triggering, and notification in terms of events, which are represented as α, β, and μ, respectively.

The occurrence of an event α = "anomaly detected" means that the function An(v) assumes the value "true" at a given time defined as ANOMALY_DETECTION_TIME ($T_\alpha$). The event β = "alarm triggering", in its turn, is defined as the action of triggering an alarm to indicate that an anomaly has been detected. The time when an event β occurs is referred as ALARM_TRIGGERING_TIME ($T_\beta$). The third event we define in this section is μ = "notification". μ is the action of sending a notification to a set of caregivers to inform them that an alarm has been triggered. The time when an event μ occurs is referred to as NOTIFICATION_TIME ($T_\mu$).

Associated with the occurrence of these events, we have the delays ALARM_TRIGGERING_DELAY ($D_\beta$) and NOTIFICATION_DELAY ($D_\mu$), where $D_\beta$ represents the delay between anomaly detection and its indication through an alarm triggering and $D_\mu$ is the delay between an alarm triggering and its notification to the caregivers. We show in Equations (3) and (4) how the delays $D_\beta$ and $D_\mu$ are calculated according to the time at which the events α, β, and μ occur.

$$D_\beta = T_\beta - T_\alpha \quad (3)$$

$$D_\mu = T_\mu - T_\beta \quad (4)$$

We can summarize the abovementioned explanation through the event-trigger rules presented in Equations (5) and (6):

$$\varphi_1: \ \alpha \rightarrow \beta \quad (5)$$

$$\varphi_2: \ \beta \rightarrow \mu \quad (6)$$

Where α, β, and μ are the events; the symbol "→" represents the action triggers; φ1 indicates that, when the event $\alpha$ occurs, the event β is automatically triggered after the delay $D_\beta$; and $\varphi_2$ indicates that event $\mu$ is automatically triggered $D_\mu$ time after β occurs.

The parameterization of the events α, β, and $\mu$ is defined as follows.

$$\alpha = < \text{type}, T_\alpha > \quad (7)$$

$$\beta = < \text{type}, \alpha, T_\beta > \quad (8)$$

$$\mu = < \text{type}, \beta, T_\mu > \quad (9)$$

Where the parameter $\alpha$ for β event represents the event α; and the parameter β for μ event represents the event "alarm triggering" β.

### 3.1.4
### Modeling Anomaly Detection, Alarm-Triggering, and Notification

To illustrate the anomaly detection, alarm-triggering, and notification processes, we present a state-transition diagram in Figure 2. This figure presents a visual representation of the following: (1) the possible states of the anomaly

detection, alarm-triggering, and notification processes; (2) the events such as inputs that may result in transitions between states; and (3) the transitions between states. We also show the conditions an event requires in order to trigger a transition.

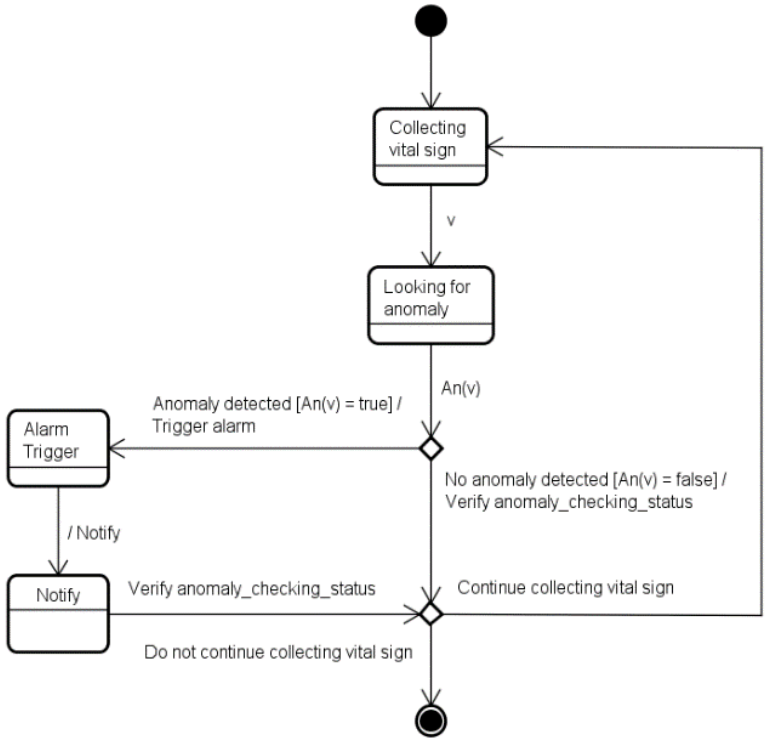



Figure 2 - The state-transition machine showing the states involved in the anomaly detection, alarm-triggering, and notification processes.

To formalize the concept of an anomaly, we present, through the state-transition machine in Figure 3, the possible states for an anomaly. Figure 3 presents the current anomaly detection process, showing the 3 possible states of an anomaly: no anomaly, anomaly alerted, and anomaly notified. The interconnecting arrows represent the transitions between states, and the labels on the arrows represent the events that make the transitions occur.

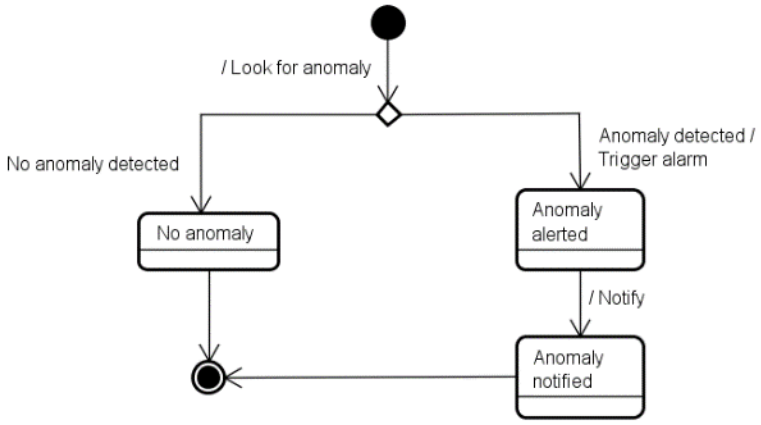



Figure 3 - The state-transition machine showing the possible states for an anomaly.

Now that the basic concepts anomaly, alarm, anomaly detection, and notification needed for the reasoning process have been defined, in the next section we present our reasoning model.

## 3.2. Adding Reasoning to the System

### 3.2.1 A Brief Description of Using Reasoning to Cope with Alarm Fatigue

In this section, we provide a brief description of how we apply a reasoning engine to the alarms generated by the monitoring devices being used to track a patient's health status to minimize alarm fatigue. The software system contains a component that reads the vital signs (the reader) accompanied by a reasoning engine that decides how to notify the health care team. The reader can be set to ignore all the non-anomalous vital signs to focus only on the anomalous values that can require attention from the caregivers' team. An anomalous reading is then passed to the reasoning engine that decides how to handle the reading. For example, the reading could be used to cause an alarm to be triggered immediately because the patient's situation is deemed critical; or readings could be accumulated as the situation is not critical but can be attended to within a certain time period.

### 3.2.2. Updating the Anomaly Detection, Alarm triggering and Notification Process through the Addition of Reasoning

Before presenting the reasoning algorithms, we show, in Figure 4, how the reasoning process interacts with the anomaly detection, alarm- triggering, and notification processes.

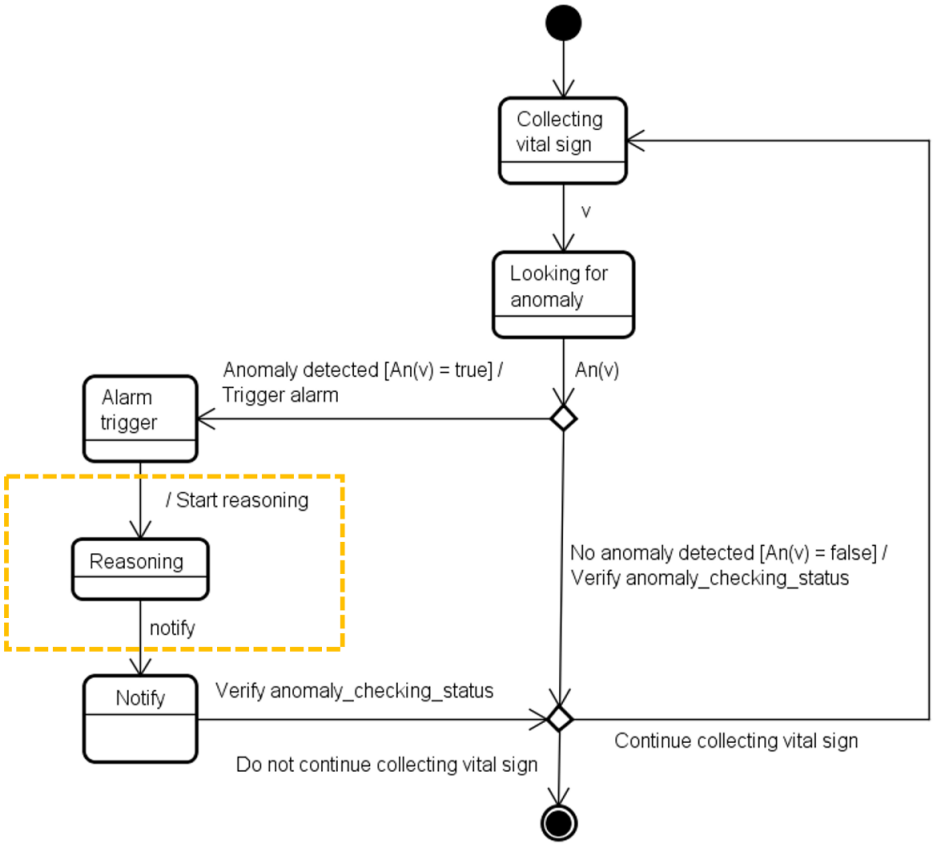



Figure 4 - Illustration of the inclusion of the state “Reasoning” (inside the hatched rectangle) that determines when an alarm trigger(s) causes a notification.

Figure 5 is an update of Figure 2 including information related to the reasoning activity.

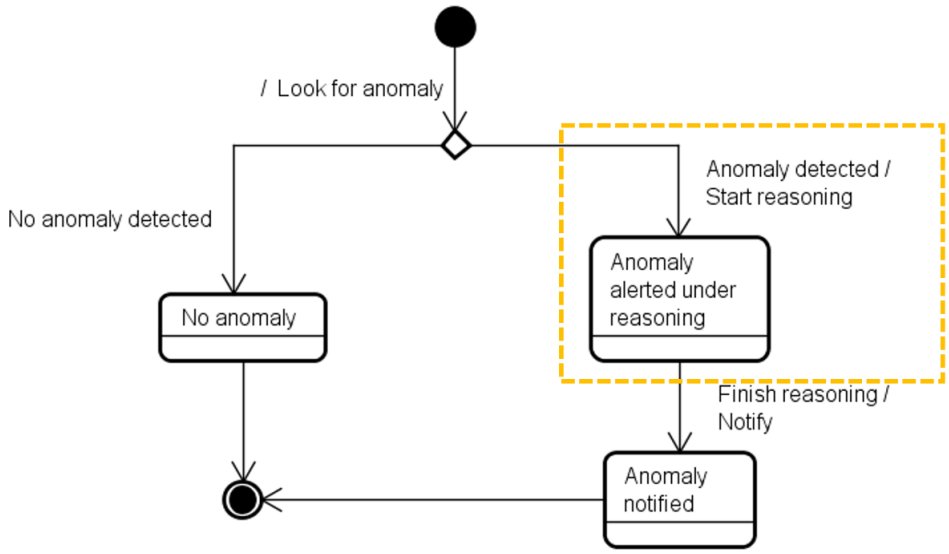



Figure 5 - Illustration of the inclusion of the new state “Anomaly alerted under reasoning” (inside the hatched rectangle) as another possible state for an anomaly.

To deal with the decision-making processes occurring during reasoning, we developed the Reasoner entity that is an instance of our reasoning algorithm. The Reasoner is responsible for managing the entire notification process. A high-level representation of the decision-making processes is shown in Figure 6.

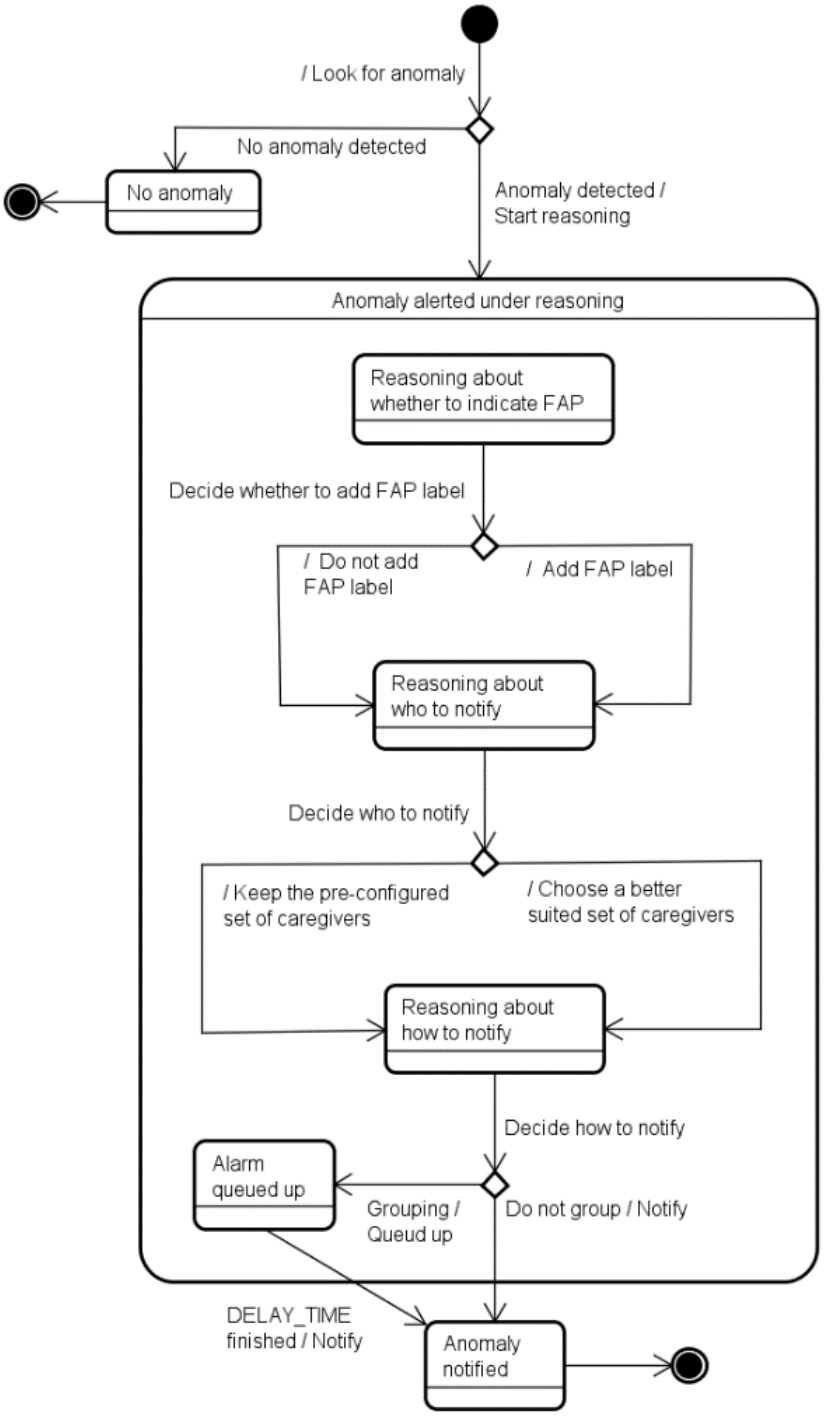



Figure 6 - A high-level representation of the decision-making processes used during reasoning. FAP: false alarm probability.

### 3.2.3.
### The Alarm Fatigue-aware Notification Model

The Alarm Fatigue-aware Notification Model (Figure 7) is designed to support reasoning algorithms that decide on the best approach to notify caregivers about anomalies detected by a patient monitoring system where a large volume of alarms could possibly lead to alarm fatigue. The reasoning algorithms, which are the focus of this research, decide on how to notify the healthcare team by determining: (i) whether to aggregate alarms to avoid alarm fatigue while not compromising patient safety, (ii) whether to add a FAP label to the notification, and (iii) who to notify within the group of caregivers.

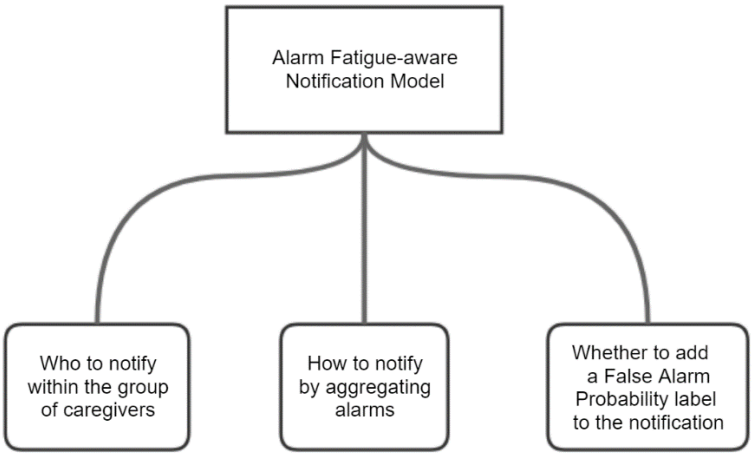



Figure 7 - Representation of our alarm fatigue-aware notification model.

# 4
# Examples of Applications

This chapter provides examples of how the data collection and the visualization modules of our architecture were implemented by showing examples of systems we built during this research. In these works, we carried out two experiments by using modern technologies to monitor brain activity and by performing Encephalography (EEG) in a wireless way through the use of wearable devices to capture EEG data. We used MindWave Mobile Headset devices to monitor electrical activity of the brain, and to collect data generated by these devices (e.g., concentration and meditation levels, and brainwave patterns). In the first example, we promoted technological support for the electric Depth of Anesthesia (DoA) monitoring activity during an intraoperative period by using EEG sensors and software agent technology (Luck, 2003). In the second example, we assembled a dataset gathered from the EEG data capture. These data were pre-processed, analyzed and visualized through proper graphs. We also provided statistical calculations such as mean, median and moving average of attention and mediation values, and we were able to make predictions based on the EEG data results.



## 4.1.
## Case Study I: Smart Depth of Anesthesia Monitoring

The present section describes the development of a medical system applied to the general domain of remote patient monitoring and focused on the anesthesiology care domain. We present a solution to perform Depth of Anesthesia (DoA) monitoring activities during the intraoperative period in a remote, autonomous and wireless way, by using EEG sensors and software agent technology (C. O. Fernandes, Lucena, & Silva, 2017).

### 4.1.1. Motivation

When a patient undergoes a surgical procedure under General Anesthesia (GA), it is expected that such a procedure will be successful in all of its stages - pre, inter and postoperative. As important as the patient's preparation for surgery and the postoperative phase - when recovery occurs - are, is the time when the procedure actually takes place. In this case, it is expected that, throughout the procedure, the three pillars of anesthesia will be kept at adequate levels: analgesia, immobility and unconsciousness. This means that the medical staff strives to ensure that the patient does not feel pain, remains immobile and is not aware of what is happening during all intraoperative period.

To ensure that such expectations are met, the intraoperative period requires a high level of collaboration between the medical teams involved in the procedure, i.e., the surgical, anesthesia and monitoring teams. Therefore, it is necessary that the three teams work with proper vigilance and agility and are able to act proactively to avoid situations of risk for the patient, detect and take quick action in the event of an unexpected situation.

The monitoring team must continuously monitor the patient’s depth of anesthesia after administration of anesthesia (Figure 8). Any changes in the expected levels of analgesia, immobility and unconsciousness should be immediately reported to the other teams so that the anesthesia team can intervene, with the adoption of strategies that guarantee the return to the adequate level.

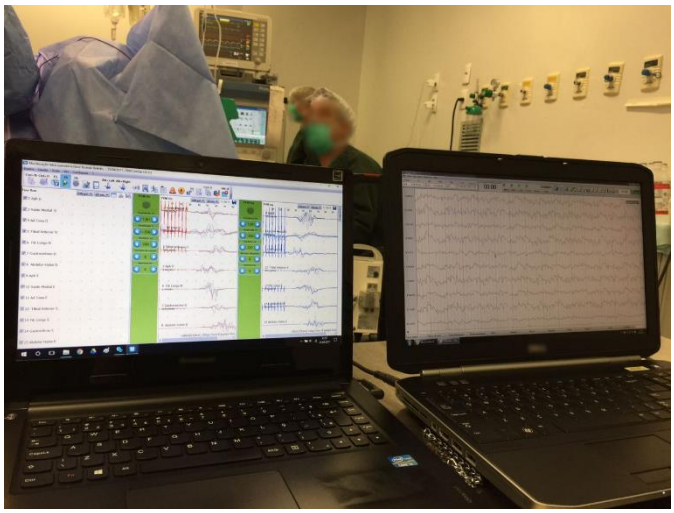


Figure 8 - The illustration of an intraoperative monitoring process of DoA.

Our solution aims to assist the DoA monitoring process during the intraoperative period by providing technological resources so that the DoA monitoring can be performed in a proactive, remote and autonomous way, trying to avoid risky situations for the patient and increasing patient safety.

### 4.1.2
### Depth of Anesthesia Monitoring

General anesthesia is a reversible state of induced coma, which comprises the three pillars of anesthetic depth shown above. During GA, these pillars need to be maintained alongside with physiological stabilization (Purdon et al., 2013). In this case, monitoring physiological patient data such as heart rate, body temperature, saturation and blood pressure has been a critical factor in reducing morbidity and mortality in anesthesiology, as well as other consequences of using inadequate doses of anesthetics, which will be discussed in section Clinical DoA Monitoring.



#### 4.1.2.1
#### DoA Monitoring Techniques

In order to assess DoA levels and guide the anesthetics administration process, the following DoA monitoring techniques have been applied in hospitals during intraoperative period: 1- Clinical DoA monitoring using multi-parametric monitors; 2- Monitoring of Expired Fraction of Anesthetic Gases; 3- Electrical DoA Monitoring.

##### 4.1.2.1.1
##### Clinical DoA Monitoring

In clinical monitoring, some physiological parameters have been used to assess DoA levels, such as: blood pressure, heart rate, respiratory pattern changes, somatic and skeletal motor activity, sweating, tearing, pupillary diameter and vasomotor skin reflexes. Although sympathetic stimulation is not always a consequence of painful stimulus, tachycardia, hypertension, sweating and tearing are usually considered signs of inadequate analgesia. However, depending on the patient's clinical conditions and the medications been used, these parameters may have little influence on the DoA assessment.

In addition to the physiological parameters, the motor response in voluntary commands' response along with the reaction to painful stimuli are also ways to evaluate the potency of anesthetic agents. The presence of these signs is an important indication of anesthetic inadequacy, which makes the patient susceptible to the risk of arousal and intraoperative awareness that are undesired conditions.

#### 4.1.2.1.2
#### Monitoring of Expired Fraction of Anesthetic Gases

Monitoring the anesthetic agent administration process, especially inhaled gases, has become routine in hospitals due to the modules incorporated into the multi-parametric monitors. The study of Nunes et al. compares the three DoA monitoring techniques regarding anesthetic consumption.

#### 4.1.2.1.3
#### Electrical DoA Monitoring

Normally performed simultaneously with clinical monitoring, electrical DoA monitoring through EEG is considered one of the most feasible approaches to tracking brain states under GA.

In general, we can highlight the following objectives of the electrical DoA monitoring activity: 1- Regulate the consumption of anesthetics, aiming to reduce the excessive administration of anesthetic agent and decrease the anesthetic recovery period, as well as the adverse effects of anesthesia such as: nausea, vomiting, headache, cognitive dysfunctions, especially in the elderly; 2- Avoid intraoperative awakening; 3- Study the relationship between electrical activity of the nervous system and mortality; 4 - Minimize the residual effects of drugs on cognition, considering Postoperative Delirium (POD) and Postoperative Cognitive Dysfunction (POCD); 5- Avoid post-traumatic stress syndrome due to superficial anesthesia (Nunes et al., 2015).

In this case study, we choose the Electrical DoA Monitoring technique as our strategy to handle our research problem of performing DoA monitoring during simulated intraoperative period.

#### 4.1.2.2
#### Consequences of Inappropriate Depth of Anesthesia

As mentioned above, DoA monitoring is necessary to ensure its adequate level, since too superficial, or deep levels of anesthesia can be disastrous in the short and long term. The following Table 2 classifies intercurrences that occur as a consequence of insufficient, or excessive dosages of anesthetics. Anesthesia awareness, also called unintended intraoperative awareness, occurs under GA when a patient becomes cognizant of some or all events during surgery, and recalls those events. Because of the routine use of neuromuscular blocking agents during GA, the patient is often unable to communicate with the surgical team when this occurs (The Joint Commission, 2004).

Table 2 - Example of anomalies that can occur as adverse effects of anesthesia.

| **Anesthetic dose** | |
|---|---|
| **Excessive Anesthetic Dose** | **Insufficient Anesthetic Dose** |
| Cardiovascular and respiratory depression;<br>Cognitive impairments in patients with low neuronal reserve | Unintended Intraoperative Awareness |

### 4.1.3.
### Open Issues and Related Work

Despite its broad application, brain state monitoring is not a totally accepted practice in anesthesia care nowadays because there are no markers or indicators that reliably track changes in the level of consciousness in patients under GA (Purdon et al., 2013). This issue makes DoA monitoring still an open problem because there is no precise mapping of how anesthetics impact the brain.

Interest in performing EEG as a monitoring tool during GA has increased considerably. This practice has been fomented by the development of technological tools that have produced tangible progress in the creation of anesthetic-depth monitors (Rampil, 1998). However, since EEG reading has not become part of the routine anesthesiology practice, a simpler approach is used: current depth-of-anesthesia monitors compute proprietary indices that reduce the EEG to a single number (BIS) intended to represent a patient's level of unconsciousness, varying from 0 to 100 (Purdon et al., 2013). Since the monitors have proprietary algorithms,

they work as a black-box approach that obscures structure in the EEG (Lehmann, Thaler, & Boldt, 2001). An adequate patient's level of unconsciousness is assumed when BIS numbers vary at intervals from 40-60. Consequently, any values smaller than 40 and greater than 60 are detected as an anomaly condition and reported to the medical team.

Instead of using these indices, some studies have created other approaches to map the EEG reading process. As an example, the work of Purdon et al. aim to better understand how anesthetics affect the brain in terms of unconsciousness, mapping the EEG reading process through a process of induction to, and emergence from, anesthetics in a very slowly way. This work involves some concepts that have been called EEG signatures and have been defined to be markers used as indicators of loss of consciousness. However, according to the authors, it has been difficult to specify EEG signatures because most anesthesia-related EEG data comes from clinical settings in which GA induction is performed rapidly, i.e., when the loss of consciousness occurs within 30-60 seconds. To overcome this obstacle, the work of Purdon et al. designed an experiment to study the relationship between EEG activity and the loss and recovery of consciousness over a two-hour period. They recorded EEGs in 10 volunteers during gradual induction of, and emergence from, unconsciousness while executing tasks to assess conscious behavior.

The results presented by Purdon et al. provided insights into the mechanisms of induced unconsciousness, establishing EEG signatures of brain states that track transitions in consciousness precisely, and suggesting strategies for monitoring the brain activity of patients receiving GA during the intraoperative period. According to the authors, the EEG signatures they have identified can be computed in real-time, are easy to recognize, and can be interpreted in a way that relates directly to the mechanisms through which this anesthetic is postulated to induce unconsciousness.

### 4.1.4. Research Goals

#### 4.1.4.1. General Goals for our SmartDoAMonitoring App

In general, we aim to promote a technological support for the electric DoA monitoring activity during an intraoperative period, in a simulated environment, in order to process DoA monitoring automatically, proactively and wirelessly. By using wearable devices with EEG sensors and software agent technology, we hope our solution contributes to facilitate risk management, increase patient safety and offer a more accessible monitoring solution, replacing large and expensive equipment for smaller devices with greater mobility such as wearable sensors. We also expect that our solution will bring the concept of personalized medicine, where the monitoring parameters are defined according to each patient individually.



#### 4.1.4.2. Specific Goals for our App

Our specific goals are: 1- Create a database as a result of the EEG sensing process; 2- Allow sensing data to be accessed remotely, monitored and visualized in real time from mobile devices (smartphones and tablets); 3- Enable alerts to the medical team when DoA anomalous values occur, notifying the professionals interested in receiving this information; 4- Evaluate our agents' performance in its monitoring and notification tasks.

### 4.1.5. Methodology

To meet our defined goals and achieve the expected results, providing an E-Health System capable of performing sensing, visualization and monitoring activities in the DoA monitoring context, the SmartDoAMonitoring Application was generated as an IoT4Health (Chrystinne Oliveira Fernandes & Lucena, 2017) instance. IoT4Health is a flexible software framework (Markiewicz & de Lucena, 2001) we developed to generate a range of Internet of Things (IoT) (Atzori, Iera, & Morabito, 2010) applications in the Remote Patient Monitoring domain. As a software framework, IoT4Health offers extensibility points for the generation of E-

Health applications as multi-agent systems (Wooldridge, 2009) that are designed to perform patient monitoring activities autonomously.

#### 4.1.5.1 EEG Sensing Process

In the sensing process, we used an Arduino microcontroller (*Arduino UNO*) Uno R3 (Figure 10) to collect the EEG data from the Mindwave headset (Figure 9). Because the headset sends the EEG data via Bluetooth and the Arduino model we were using did not provide communication capabilities in this technology, we coupled the Arduino with a Bluetooth module (*Sparkfun*) to enable communication between the two devices. The BlueSMiRF module has been programmed to pair with and connect to the headset. Once connected to the microcontroller with the Bluetooth module, Mindwave was able to send a stream of data through the ThinkGear protocol (*ThinkGear Serial Stream Guide*). In this case, the Arduino was programmed (in C language) to process the data stream received by Mindwave, interpreting it through the use of the ThinkGear protocol and storing the information in a database.

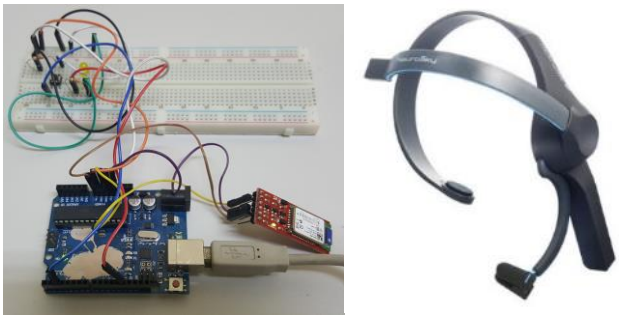


Figure 9 - Devices used to collect EEG data: our hardware prototype with the Arduino Uno R3 Microcontroller and the BlueSMiRF Sparkfun Bluetooth Module, on the left; MindWave Mobile Headset from NeuroSky on the right;

The Mindwave measures and outputs the following data: Received Package Timestamp (TSLP); Information about quality of signal (PoorQuality); Raw EEG Data (RAW); Processed information that corresponds to the brainwaves patterns (Delta Power, Theta Power, Low Alpha Power, High Alpha Power, Low Beta Power, High Beta Power, Low Gamma Power, High Gama Power); Attention; Mediation; Blink Strength (Strength of detected blink); Mental Effort (measures

how the subject's brain is working); Familiarity (measures how well the subject is learning a new task).

From the information provided by the device, we decided to assemble our dataset, capturing and storing the following data subset: PoorQuality, Attention, Meditation, Brain waves (Delta, Theta, LowAlpha, HighAlpha, LowBeta, HighBeta, LowGamma, MidGamma) and TSLP.

#### 4.1.5.2 EEG Monitoring, Anomalies Detection and Notification Processes

Our DoA monitoring goal was to detect anomalies in the EEG data provided by the Mindwave device to assess if the pillars of GA were adequate. As strategies for monitoring the brain activity of patients receiving GA by the Electrical DoA Monitoring technique some monitoring parameters such as BIS number and EEG signatures could be used in real hospital environments.

However, as a strategy for monitoring the patients' brain activity in our simulated environment, we selected the Attention EEG data from our database to serve as the main parameter of our monitoring and anomalies detection activities. We utilized this strategy because we did not find how the BIS number is calculated. As we said previously, it is derived by a proprietary algorithm and works as a black-box approach. However, both the BIS and MindWave Attention values vary from 0 to 100. Similarly to the BIS approach, we considered that a normal patient condition is assumed when the Attention level varies at intervals from 40-60, e.g., all occurrences of attention outside this interval are reported as anomalies in DoA and trigger a notification message to health providers. So, our developed solution can be posteriorly applied to other monitoring parameters such as BIS number – if we can calculate it in future – in order to have a more useful comparison.

To measure the ability of our tool to respond proactively and in real-time to adverse conditions and its capacity to notify health providers in case, for example, of anomalies in patients' EEG signs (Table 3) the following step-by-step experiment was conducted:

Step1. Initially, five measurement points related to the tasks performed by agents were identified in the SmartDoAMonitoring App and were labeled Timestamps (T1-T5) as follows:

T1. Our app collects the EEG data and the monitoring agent analyses them, searching for anomalies. If no anomaly is detected, the system remains in a loop collecting and analyzing more data until an anomaly is found. Once an anomaly is detected the application continues to T2.

T2. This point is reached when the monitoring agent detects an EEG data anomaly and then calls the notification agent.

T3. The notification agent initiates the routine to notify the health care providers;

T4. The notification agent sends information about the detected EEG anomaly to the patient's health care providers.

T5. The health care providers receive the notification message on their mobile phones or tablets.

Step 2. The SmartDoAMonitoring App is executed and the timestamps are measured and registered.

Step 3. Our delays captured by the agents' execution tasks are defined as follows:

Delay 1: DAI- Detection Anomaly Interval (DAI = T2−T1). The anomaly's detection delay in the monitoring routine.

Delay 2: NSI- Notification Start Interval (NSI = T3−T2). The delay between the anomaly detection and the initiation of the notification routine.

Delay 3: NP- Notification Period (NP = T4−T3). Duration of the notification routine by agents.

Delay 4: NRI- Notification Routine Interval (NRI =T5−T4). Time elapsed between the sending of the notification and its receipt by the health provider.

These delays were calculated to serve as a concrete measure of how quickly and proactively the solution can respond to the environment, as well as to support the assertion that this system performs anomaly detection in real time.

#### 4.1.5.3
#### Patient Monitoring and Anomalies Detection

Our solution transforms large-scale sensor data into more significant information that meets specific application requirements for patient monitoring task. In this case, the large volume of data (collected from sensors in the sensing

phase) is processed autonomously (by software agents) in the monitoring phase. The monitoring parameters are contained in a knowledge base specified by medical experts. By consulting this database, the agents are able to detect anomalies, that is, physiological patient data outside the expected limits. In the monitoring stage, each knowledge base, handled by the agents monitoring a particular patient is built based on the assessment of each individual patient, bringing us to the concept of personalized medicine.

The information resulting from the monitoring phase - the anomaly - triggers a notification to the end user of the system - the health professionals in charge of responding to these events. Notifications are also performed automatically by software agents and can be delivered via SMS, email or Bluetooth messages.

#### 4.1.5.4
#### EEG Sensing Technology

Electroencephalography has been used by neuroscientists and psychologists to monitor brainwaves since 1937 (Purdon et al., 2013). Usually, encephalograms require patients to remain immobile as movements interfere with the brain's impulses. Fortunately, current technology allows scientists to observe how the brain works in much more realistic configurations. This is justified by the fact that brainwave detection technologies have evolved considerably, allowing encephalography realization in a wireless and mobile mode.

It is now even possible to monitor the mental state of patients through autonomous wearable sensors. Modern technology eliminates the wires between the device reading EEG signals and the computer, smartphone or other device that collects, analyzes and processes these data. For this experiment, we utilized wearable devices to wirelessly monitor brain electrical activity.

### 4.1.6.
### Results

To confirm the fulfillment of our research goals in this case study, we will present the architecture of the SmartDoAMonitoring App and the results of its main functionalities, namely sensing, visualization, monitoring and notification.

#### 4.1.6.1
#### The SmartDoAMonitoring App's Architecture

The following Figure 10 shows the architecture of the application, structured in 3 layers: L1- Data persistence layer; L2- Communication layer between layer L1 and layer L3; L3- Application Layer manages the application data for the collection, visualization, monitoring and notification processes.

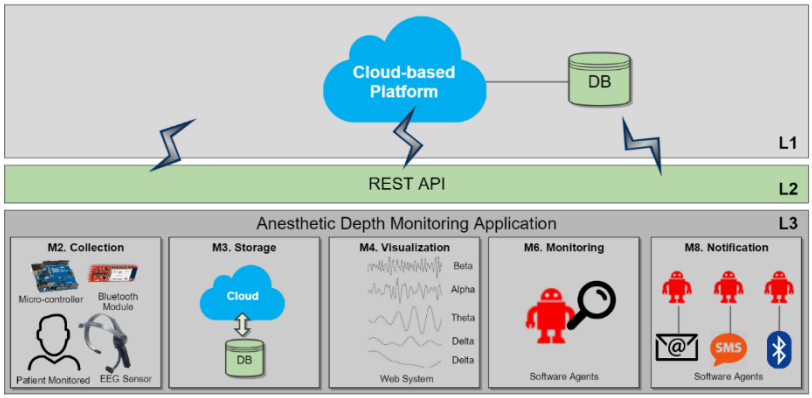



Figure 10 - The Smart Depth of Anesthesia Monitoring Application Architecture.

#### 4.1.6.2
#### Sensing Results: The Smart DoA Monitoring App's Database

As a result of the sensing process, we have an effective capture of the electrical activity collected by Mindwave, via Bluetooth. Table 3 shows the structure of the SmartDoAMonitoring App dataset and some examples of the data collected during our sensing phase.

#### 4.1.6.3
#### Visualization Results: Real-time EEG Data Streams

Figure 11 and Figure 12 show the visualization results comprising times series representing EEG Data streams.

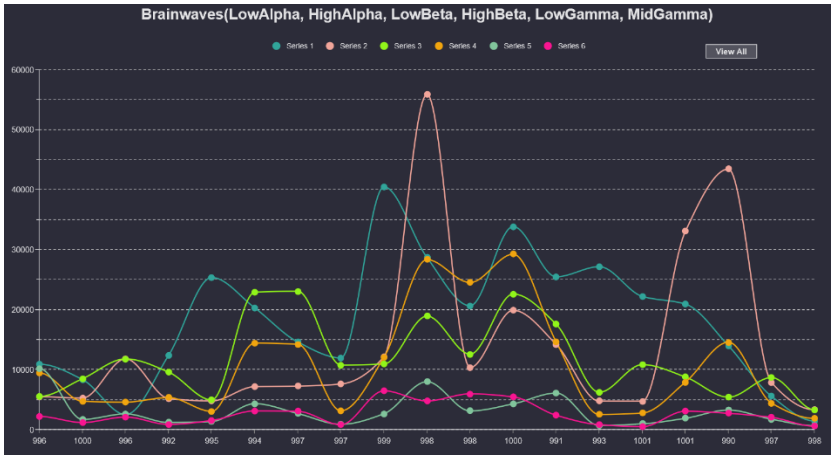



Figure 11 - Times series representing brainwaves patterns exhibited in real-time.

### 4.1.6.4
### Monitoring Results: Anomalies Detection and Notification

To confirm the fulfillment of the main research goal, we conducted the experiment described above and tabulated the relevant results. Table 4 shows examples of timestamps for agents' behavior and task delays for EEG sensing, attention monitoring and notification activities, where the EEG Attention level is monitored to evaluate DoA levels. In case of anomalies detection for attention values (illustrated in red circles in Figure 12) the agents act by sending SMS messages to the healthcare providers (Figure 13).



Table 3 - Illustration of the SmartDoAMonitoringApp dataset structure.

| Poor Qual. | Attention | Meditation | Delta | Theta | Low Alpha | High Alpha | Low Beta | High Beta | Low Gamma | Mid Gamma | TSLP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 40 | 30 | 196432 | 60147 | 10941 | 5514 | 5568 | 9479 | 10191 | 2227 | 996 |
| 0 | 53 | 23 | 498363 | 115836 | 8379 | 5299 | 8525 | 4763 | 1738 | 1221 | 1000 |
| 0 | 53 | 27 | 144949 | 23748 | 2548 | 11834 | 11824 | 4622 | 2658 | 2129 | 996 |
| 0 | 54 | 34 | 1826689 | 68866 | 12433 | 5217 | 9615 | 5372 | 1226 | 902 | 992 |
| 0 | 38 | 38 | 747555 | 73652 | 25398 | 4817 | 4982 | 3081 | 1384 | 1560 | 995 |
| 0 | 37 | 30 | 210412 | 437968 | 20324 | 7202 | 22952 | 14467 | 4360 | 3148 | 994 |
| 0 | 27 | 48 | 578487 | 54416 | 146371 | 7289 | 23104 | 14258 | 2717 | 3080 | 997 |
| 0 | 34 | 61 | 89963 | 15677 | 11993 | 7637 | 10805 | 3170 | 916 | 854 | 997 |
| 0 | 53 | 77 | 475890 | 9452 | 40504 | 12196 | 10967 | 12096 | 2683 | 6502 | 999 |
| 0 | 63 | 93 | 231761 | 316959 | 28718 | 55932 | 18963 | 28373 | 8026 | 4817 | 998 |
| 0 | 78 | 75 | 379599 | 60849 | 20685 | 10395 | 12611 | 24556 | 3153 | 5998 | 998 |
| 0 | 94 | 75 | 51382 | 41581 | 33890 | 19928 | 22637 | 29279 | 4328 | 5499 | 1000 |
| 0 | 93 | 60 | 146485 | 63617 | 25516 | 14234 | 17593 | 14652 | 6129 | 2420 | 991 |
| 0 | 84 | 70 | 318539 | 36694 | 27196 | 4782 | 6263 | 2557 | 722 | 857 | 993 |
| 0 | 60 | 74 | 395389 | 52250 | 22242 | 4771 | 10872 | 2802 | 1006 | 531 | 1001 |
| 0 | 44 | 80 | 265306 | 37419 | 20958 | 33109 | 8810 | 7952 | 1920 | 3103 | 1001 |
| 0 | 41 | 84 | 295009 | 66752 | 14009 | 43472 | 5443 | 14610 | 3301 | 2747 | 990 |
| 0 | 47 | 67 | 193669 | 71237 | 5590 | 7878 | 8708 | 4447 | 1769 | 2083 | 997 |
| 0 | 47 | 50 | 438916 | 52121 | 1482 | 3352 | 3339 | 1903 | 685 | 601 | 998 |

Table 4 - Results for monitoring and notification tasks performed by agents.

| Timestamp T1 | Timestamp T2 | Timestamp T3 | Timestamp T4 | Timestamp T5 | DAI (s) | NSI (s) | NP (s) |
|---|---|---|---|---|---|---|---|
| 2017-05-14-142415 | 2017-05-14-142418 | 2017-05-14-142418 | 2017-05-14-142419 | 2017-05-14-1415 | 3 | 0 | 1 |
| 2017-05-14-142421 | 2017-05-14-142425 | 2017-05-14-142425 | 2017-05-14-142427 | 2017-05-14-1416 | 4 | 0 | 2 |
| 2017-05-14-142433 | 2017-05-14-142436 | 2017-05-14-142436 | 2017-05-14-142439 | 2017-05-14-1416 | 3 | 0 | 3 |

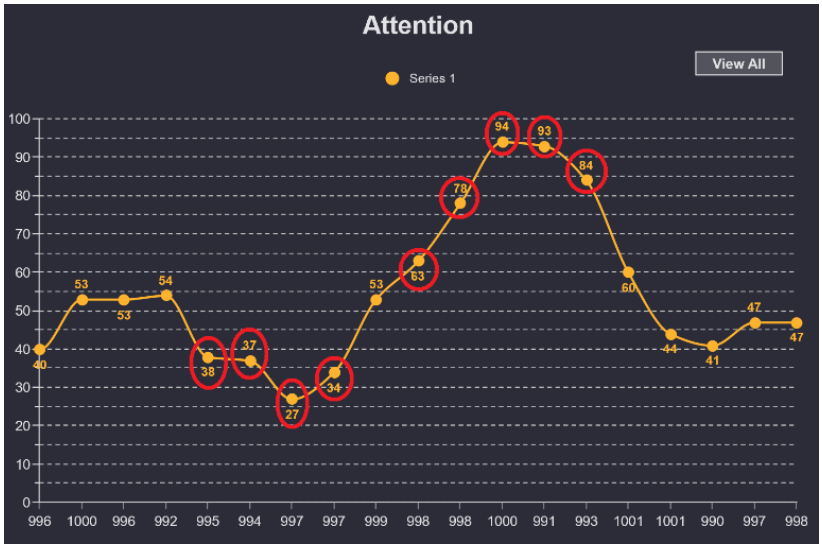



Figure 12 - Times series representing Attention levels with the illustration of the anomalies detection process shown in red circles.

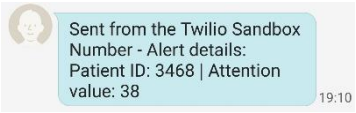



Figure 13 - SMS Message sent by notification agent in anomalies detection task.

### 4.1.7. Discussion

We can conclude that we achieved our expected results and reached our research goals, since our solution proved to be effective for its purpose. It has been effective in achieving the challenge of performing, in simulated conditions, DoA monitoring in an autonomous way by using wireless sensing and agent-based technologies. All simulated electrical DoA monitoring was performed through the use of wearable devices with Bluetooth communication capabilities, which allows more mobility and flexibility to carry out the sensing. Our solution is an alternative to expensive and reduced mobility equipment used in DoA monitoring. As a result of the monitoring phase, we have an effective detection of anomalies in EEG data

and consequent notifications to the professionals involved. This task was performed effectively and autonomously by software agents

As future work, we can utilize, as they become available, BIS number and EEG signatures to serve as baselines for anomalies values (monitoring parameters) in our monitoring process instead of the use of MindWave Attention level. In addition, we can contemplate the development of applications that carry out Electrical DoA monitoring, using other techniques mentioned in this thesis, e.g., clinical monitoring of DoA and monitoring of the expired fraction of anesthetic gases. To monitor the expired fraction of anesthetic gases, in the sensing phase, we can use gas sensors to monitor the gases expelled by the patient after anesthesia.

For clinical monitoring, we can use the biometric sensors we are currently working with to monitor physiological data, such as blood pressure, heart rate, respiratory pattern changes, pulseoxiometry, among others. We also intend to use the multiparametric monitors we are also currently working with to perform this monitoring activity (*LES PUC-Rio*).

As we mentioned above, one of the consequences of very deep anesthesia is that it can affect the patient's cardiovascular system. In this case, we can use sensors that monitor the cardiovascular system to control this negative consequence of anesthesia, simultaneously with electrical DoA monitoring techniques. We also aim to utilize the Mindwave device to develop other apps, such as: monitoring of patients with epilepsy, in order to report situations of abnormality in his/her patterns of brain electrical activity and predict new epileptic seizures; monitoring of sleep disorders; monitoring of brain activity of people with depression, amont others.

The database that was created as the result of the sensing and monitoring processes could be used as an input dataset for training and validation of a model to be used with machine learning techniques to predict the occurrence of anomalies. The knowledge produced at the end of the prediction phase could be used to support decision-making processes.

## 4.2. Case Study II: Statistical Analysis and Predictions on Monitoring Brainwave Activity

This Section describes our second example of aplication (Chrystinne Oliveira Fernandes, Moreira, Barbosa, & de Lucena, 2017) developed to perform

brainwave monitoring activity. We present next the motivation, research goals, methodology and results of this experiment carried out to make statistical analysis and predictions based on EEG results.

### 4.2.1. Motivation

Physicians have been monitoring brainwave activity to recognize sleep patterns and epileptic seizures. Moreover, devices with EEG sensors are also being used not only for medical purposes but also for entertainment in order to detect emotional states such as excitement, frustration and boredom, all of which help create patterns that can be used in games, for example. Similarly, such devices are being used to create applications that interact with virtual objects, such as pushing or lifting these objects, based on mental effort levels. Some examples of other applications are: assisting people with motor paralysis to interact with the world; alerting individuals suffering from migraines about an impending headache; adjusting computerized learning to suit the student's individual rhythm.

In this case study, we utilized Mindwave devices to monitor the electrical activity of the brain, to collect data offered by the device (attention levels, meditation levels, brainwave patterns) and to assemble a dataset through this collection process. Further, we explored these data to perform pre-processing, analysis and prediction activities.

There is a broad spectrum of publications in the field of brain monitoring ranging from sensor design (Abhishek, Poojary, Rao, & Narayanan, 2013) to injury prevention (Goldman et al., 2009), and drug use detection (Craig, Tran, Wijesuriya, & Nguyen, 2012).

Our work is related to the work of Craig et al., but instead of identifying just fatigue by analyzing brain wave graphs, we are more interested in identifying individuals and activities. Since we can identify an individual by its brain wave pattern, we can use this as part of a biometric security authentication process as future work.

### 4.2.2. Research Goals

#### 4.2.2.1. Main Goal

The main goal of this experiment was to perform pre-processing, analysis, prediction and visualization activities in EEG data collected through the use of Mindwave.

#### 4.2.2.2. Specific Goals

We defined the following specific goals:

1. Assemble a dataset comprising the following information: a) EEG data provided by Mindwave devices; b) Processed data generated through pre-processing activity; c) Annotation of user name, shift (day or night) and activity performed by the participants of the experiment during the collection process;
2. Filter the dataset by pre-processing the EEG data to achieve the subset capable of providing the best analysis and prediction results;
3. Plot pre-processed data, choosing proper graphs to visualize attention and meditation levels, brainwave patterns and statistical calculations such as mean, median and moving average of attention and mediation values;
4. Analyze the distribution of attention and meditation levels by shift (day or night) and by the activity performed by the user, taking into consideration each participant, individually;
5. Perform prediction experiments to identify the subject that was using the EEG sensor as well as the shift and activity information. Shift and activity predictions were also performed considering each subject individually.



### 4.2.3. Methodology

#### 4.2.3.1 EEG Sensing

The first step of this work was the specification of the dataset we utilized to perform analysis and prediction activities in the EEG data collected through the wearable sensors. The dataset structure (specification of columns and data types) was defined based on the data provided by Mindwave, as well as on information

that were added to enrich the dataset, via features derivation mechanisms and annotations. Our dataset was populated with the EEG data collected from two participants of the machine learning experiments and are herein referred to as Subject 1 and Subject 2.

#### 4.2.3.2 Data Collection

We developed two applications to collect data: (i) via Arduino Micro-controller, and (ii) via Mobile App.

Our first data collection strategy received Mindwave data via Bluetooth, with the use of the hardware prototype shown in Figure 10.

Collection files had 3600 registers approximately, since the duration of each collection performed by the participants took approximately one hour. The frequency of collection in both applications was one collection per second.

The Arduino implementation did not provide an annotation support for each data collection section and lacked the description of the activity and the name of the user performing the experiment. Our initial data collection infrastructure needed to be improved to support further features such as activity annotation, user annotation, and ubiquity.

To overcome those limitations, we developed an Android application using the android studio SDK to collect data. In this application, we did not use the mindwave SDK; instead, we developed a parser for the mindwave's thinkgear protocol in pure Java language to extract the information we needed from the bitstream. A series of performance tests were done to evaluate the ability of the application to maintain a constant rate of decoding, since Java uses garbage collection that is not controlled by the programmer and could cause some interference with the data collection sustainability rate. After some fine-tuning, this problem was solved and the impact of the garbage collection was kept at a minimum in terms of delay.

Our application, as shown in Figure 14, asks the user to inform the activity and the username prior to starting the data collection, which, in turn, will activate the start capture button.

The application worked standalone in the Android system, without transmitting data, just saving the information obtained in the local file system, so

we did not have concerns about security issues regarding data transmission after collection, and the brainwave device uses standard Bluetooth technology.

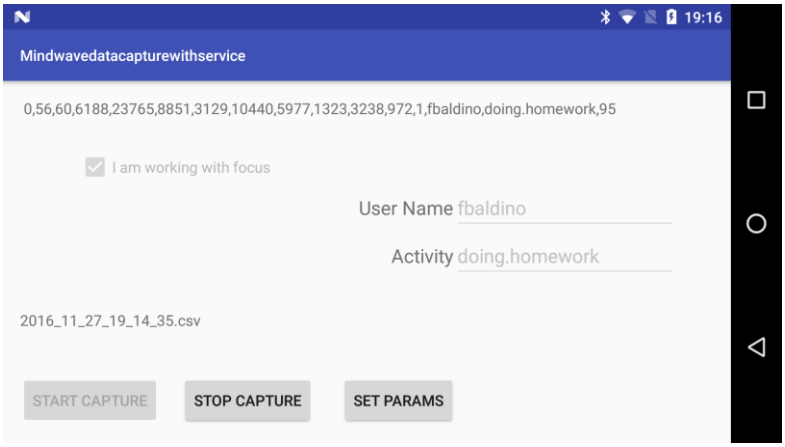



Figure 14 - Mobile App developed to capture EEG data provided by Mindwave. This solution works on Android smartphones.

#### 4.2.3.4
#### Data Storage and Data Processing

The pre-processing activities were performed through the following steps:

1. Transforming old contents of data files collected via Arduino into a standard format corresponding to the dataset structure specified in this work and storing them in .csv files;
2. Renaming previous files of data collection to a defined pattern for file names (YYYY_DD_MM_HH_MM_SS.csv), which was utilized in visualization and analysis scripts;
3. Deleting: a) data with quality of signal lower than 100%; b) data whose quality of signal was 100% but had extremely low values to brainwave patterns (lower than 100, in absolute values);
4. Creating a new column in dataset to contain information about the shift (day/night) in which the collection occurred. We considered day collections those that were performed between 6:00 and 17:59 and night collections those occurred between 18:00 and 5:59;
5. Finally, removing file collections with few data, i.e., with extremely short periods of collection;

#### 4.2.3.5
#### Dataset Annotations

We defined the following annotations for our app: (i) username ("Username"); (ii) information about how focused a user was on his/her activity at the time ("I´m working with focus") and (iii) the activity performed by the user at

the collection time ("Activity"). These activities were comprised of the following categories:

1. Reading papers;
2. Having classes;
3. Working;
4. Watching TV;
5. Watching movies;
6. Playing videogames;

Figure 14 shows samples of annotations ("Username", "Activity", "I'm working with focus").

### 4.2.4. Results

#### 4.2.4.1 The Pre-processed Dataset

The resulting dataset of the pre-processing phase was an only .csv file with 12.7 MB, which was stored in the Azure Machine Learning Studio platform (*Azure Machine Learning Studio*). The dataset was structured with 101.094 rows and 23 features. Table 5 shows our dataset structure.



Table 5 - Some samples of our dataset.

| **Index** | 1 | 2 | 3 |
|---|---|---|---|
| **PoorQuality** | 0 | 51 | 0 |
| **Attention** | 53 | 53 | 64 |
| **Meditation** | 34 | 34 | 43 |
| **Delta** | 869278 | 1473793 | 2433755 |

| | | | |
|---|---|---|---|
| **Theta** | 131673 | 71673 | 129964 |
| **LowAlpha** | 9404 | 27216 | 108148 |
| **HighAlpha** | 50536 | 10793 | 20345 |
| **LowBeta** | 93913 | 32962 | 46389 |
| **HighBeta** | 50798 | 14194 | 60667 |
| **LowGamma** | 16638 | 10852 | 31269 |
| **MidGamma** | 7856 | 4224 | 5956 |
| **TLSP** | 993 | 994 | 997 |
| **Focused** | 1 | 1 | 1 |
| **Username** | 2 | 2 | 2 |
| **Activity** | reading | reading | reading |
| **Sequence** | 1 | 2 | 3 |
| **TimeoffsetFStart** | 993 | 1987 | 2984 |
| **DateTimeCStart** | 2016-10-13T12:30:37 | 2016-10-13T12:30:37 | 2016-10-13T12:30:37 |
| **DataCaptureOK** | 1 | 0 | 1 |
| **ImpossibleData** | 0 | 0 | 0 |
| **GoodData** | 1 | 0 | 1 |
| **Night** | 0 | 0 | 0 |

#### 4.2.4.2 Analysis

Our analysis activities were performed in R language, using the RStudio (*RStudio*) tool, comprising the following steps:

1. Statistical analysis to indicate the percentage of: a) Bad records, i.e., data with quality of signal lower than 100%, which corresponded to values higher than 100, in absolute values); b) Impossible records, i.e., data with quality of signal equal to 100% (equal to 200, in absolute values, but with brainwave patterns Delta, Theta, etc lower than 100, in absolute values); c) Good records, i.e., data kept after elimination of bad records and impossible records;
2. Calculation of mean, median and moving average for attention levels, considering each activity and each collection file containing records of approximately one hour of duration per collection);
3. Calculation of mean, median and moving average for meditation levels, considering each activity and each collection file containing records of approximately one hour of duration per collection).

Figure 15 displays statistic results from the pre-processing phase, with the following information: Impossible Records (IR); Possible Records (PR); Bad Capture (BC); Good Capture (GC). Script 1 shows the script we utilized to pre-process our dataset.

| DateTime | IR | PR | %IR | BR | GR | %BR | BC | GC | %BC | Total |
|---|---|---|---|---|---|---|---|---|---|---|
| 2016-10-13 17:51:57 | 0.00 | 3274 | 0.00 | 16.00 | 3258 | 0.49 | 16.00 | 3258 | 0.49 | 3274 |
| 2016-10-15 15:52:45 | 0.00 | 1926 | 0.00 | 151.00 | 1775 | 7.84 | 151.00 | 1775 | 7.84 | 1926 |
| 2016-10-18 20:03:42 | 0.00 | 2605 | 0.00 | 154.00 | 2451 | 5.91 | 154.00 | 2451 | 5.91 | 2605 |
| 2016-10-18 21:02:15 | 0.00 | 4138 | 0.00 | 202.00 | 3936 | 4.88 | 202.00 | 3936 | 4.88 | 4138 |
| 2016-10-19 21:40:38 | 0.00 | 2462 | 0.00 | 96.00 | 2366 | 3.90 | 96.00 | 2366 | 3.90 | 2462 |
| 2016-10-22 14:42:49 | 0.00 | 2955 | 0.00 | 2134.00 | 821 | 72.22 | 2134.00 | 821 | 72.22 | 2955 |
| 2016-10-22 16:04:50 | 0.00 | 4628 | 0.00 | 75.00 | 4553 | 1.62 | 75.00 | 4553 | 1.62 | 4628 |
| 2016-10-23 20:27:07 | 0.00 | 5638 | 0.00 | 428.00 | 5210 | 7.59 | 428.00 | 5210 | 7.59 | 5638 |
| 2016-10-25 20:57:04 | 0.00 | 2727 | 0.00 | 103.00 | 2624 | 3.78 | 103.00 | 2624 | 3.78 | 2727 |
| 2016-10-25 22:29:30 | 0.00 | 3952 | 0.00 | 237.00 | 3715 | 6.00 | 237.00 | 3715 | 6.00 | 3952 |
| 2016-10-26 09:59:26 | 0.00 | 3562 | 0.00 | 213.00 | 3349 | 5.98 | 213.00 | 3349 | 5.98 | 3562 |
| 2016-10-26 20:39:40 | 0.00 | 4301 | 0.00 | 102.00 | 4199 | 2.37 | 102.00 | 4199 | 2.37 | 4301 |
| 2016-10-27 21:17:52 | 0.00 | 2920 | 0.00 | 363.00 | 2557 | 12.43 | 363.00 | 2557 | 12.43 | 2920 |
| 2016-10-28 16:11:09 | 10.00 | 2516 | 0.40 | 27.00 | 2499 | 1.07 | 17.00 | 2509 | 0.67 | 2526 |
| 2016-10-28 17:50:58 | 0.00 | 1971 | 0.00 | 349.00 | 1622 | 17.71 | 349.00 | 1622 | 17.71 | 1971 |
| 2016-10-28 21:09:57 | 0.00 | 3557 | 0.00 | 2650.00 | 907 | 74.50 | 2650.00 | 907 | 74.50 | 3557 |
| 2016-10-29 09:42:10 | 14.00 | 3655 | 0.38 | 172.00 | 3497 | 4.69 | 158.00 | 3511 | 4.31 | 3669 |
| 2016-10-30 19:50:46 | 16.00 | 4059 | 0.39 | 53.00 | 4022 | 1.30 | 37.00 | 4038 | 0.91 | 4075 |
| 2016-10-30 21:04:44 | 12.00 | 3604 | 0.33 | 148.00 | 3468 | 4.09 | 136.00 | 3480 | 3.76 | 3616 |
| 2016-11-01 09:57:45 | 0.00 | 5704 | 0.00 | 53.00 | 5651 | 0.93 | 53.00 | 5651 | 0.93 | 5704 |
| 2016-11-07 09:23:49 | 7.00 | 1844 | 0.38 | 55.00 | 1796 | 2.97 | 48.00 | 1803 | 2.59 | 1851 |
| 2016-11-07 09:38:15 | 9.00 | 3276 | 0.27 | 355.00 | 2930 | 10.81 | 346.00 | 2939 | 10.53 | 3285 |
| 2016-11-07 10:45:26 | 14.00 | 3011 | 0.46 | 133.00 | 2892 | 4.40 | 119.00 | 2906 | 3.93 | 3025 |
| 2016-11-07 10:49:02 | 15.00 | 3034 | 0.49 | 320.00 | 2729 | 10.50 | 305.00 | 2744 | 10.00 | 3049 |
| 2016-11-08 14:09:23 | 16.00 | 2409 | 0.66 | 195.00 | 2230 | 8.04 | 179.00 | 2246 | 7.38 | 2425 |
| 2016-11-14 10:42:41 | 11.00 | 3041 | 0.36 | 554.00 | 2498 | 18.15 | 543.00 | 2509 | 17.79 | 3052 |
| 2016-11-21 09:19:46 | 19.00 | 5083 | 0.37 | 392.00 | 4710 | 7.68 | 373.00 | 4729 | 7.31 | 5102 |
| 2016-11-21 17:37:13 | 16.00 | 4119 | 0.39 | 50.00 | 4085 | 1.21 | 34.00 | 4101 | 0.82 | 4135 |

Figure 15 - Statistical analysis for pre-processing activity.

Script 1. Script of statistical analysis for dataset pre-processing.

```
PrepareAndStandarizeDataFrame = function(df,fname){
  localdf <- within(df, TimeoffsetFileStart<-cumsum(TSLP))
  localdf[,"DateTimeCaputreStart"]<-
                GetFileDateAndTime(fname)
  localdf[,"X"] <-NULL
  dropcolums <- c("X")
  localdf[,"DataCaptureOK"]<- localdf$PoorQuality == 0
  localdf[,"ImpossibleData"]<-(localdf$PoorQuality == 0) &
     (localdf$Theta<MindWavePowerThreshold &
     localdf$LowAlpha<MindWavePowerThreshold &
     localdf$HighAlpha<MindWavePowerThreshold &
     localdf$LowBeta<MindWavePowerThreshold &
     localdf$HighBeta<MindWavePowerThreshold &
     localdf$LowGamma<MindWavePowerThreshold &
     localdf$MidGamma<MindWavePowerThreshold)
  localdf[,"GoodData"]<-localdf$DataCaptureOK &
                (!localdf$ImpossibleData)
  return (localdf[,!(names(localdf)  %in% dropcolums ) ])
}
```

### 4.2.4.3 Visualization

In the visualization section we present the results for our analysis activities, using part of the graphs we generated to plot the results for statistics calculation of mean, median and moving average.

All graphs were generated using R language. Graph shown in Figure 16 used the zoo (Zeileis, Achim; & Grothendieck, Gabor) library to plot data. Graphs shown in Figure 17 and Figure 18 used the ggplot2 (Wickham, 2009) library and a generic plot function developed for those specific graphs, where parameters were used to specify the graph data and title. Graph in Figure 19 used a generic procedure created by using ggplot2, to show density. Graphs in Figure 20 and Figure 21 demanded the creation of a function to compute movable average centered, ahead or delayed. Every execution of this function creates a new column in the graph data, which is melt using the melt function from the package reshape2 (Wickham, 2009) and plotted using the ggplot2 basic graph function.

Figure 16 shows times series representing, from top to bottom of the graph, brainwave patterns Delta, Theta, LowAlpha, HighAlpha, LowBeta, HighBeta, LowGamma and MidGamma, respectively.

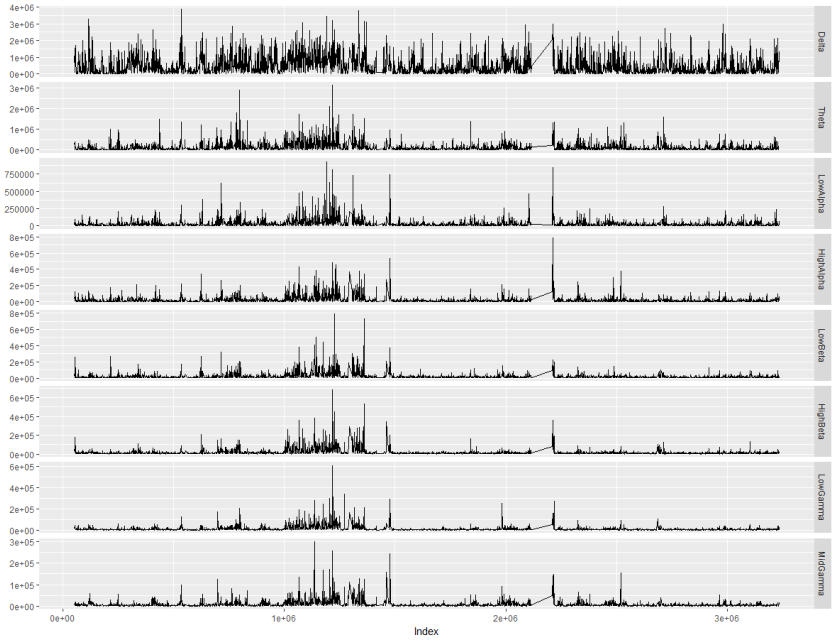



Figure 16 - Time series representing brainwave patterns.

Figure 17 shows times series (plotting in blue color) representing attention levels for subject 2. This collection was made during the day while the subject was working. The visualization also indicates the mean (plotted in red) and the median (plotted in grey) for his/her attention.

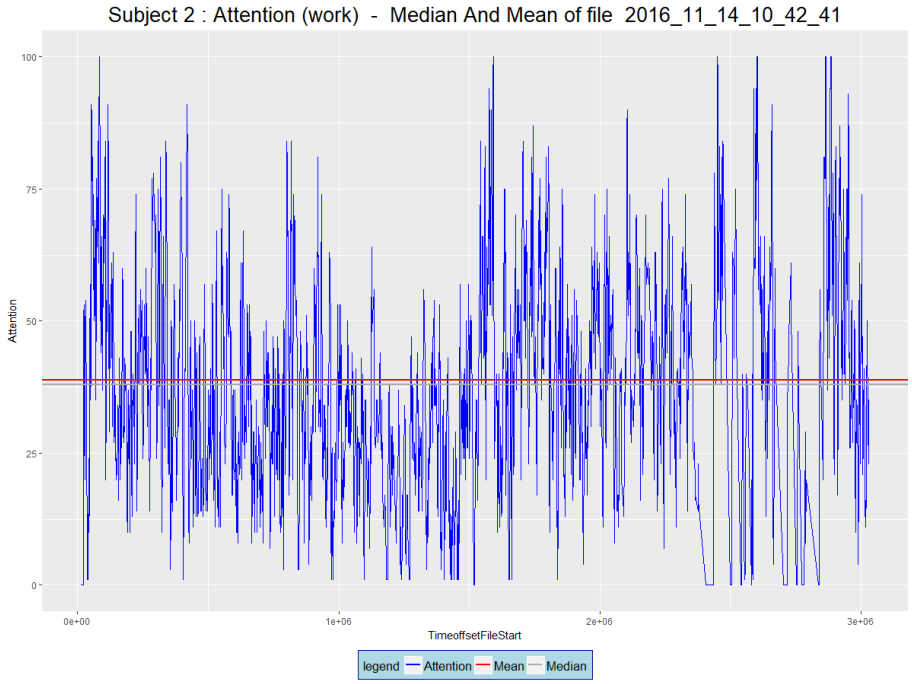



Figure 17 - Time series representing attention levels (axis y) over time (axis x). The mean is plotted in blue color and the median in red.

Figure 18 shows times series (plotting in gray color) representing meditation levels for subject 2. This collection was made during the day while the subject was working. The visualization also indicates the mean (plotted in red) and the median (plotted in blue) for his/her attention.

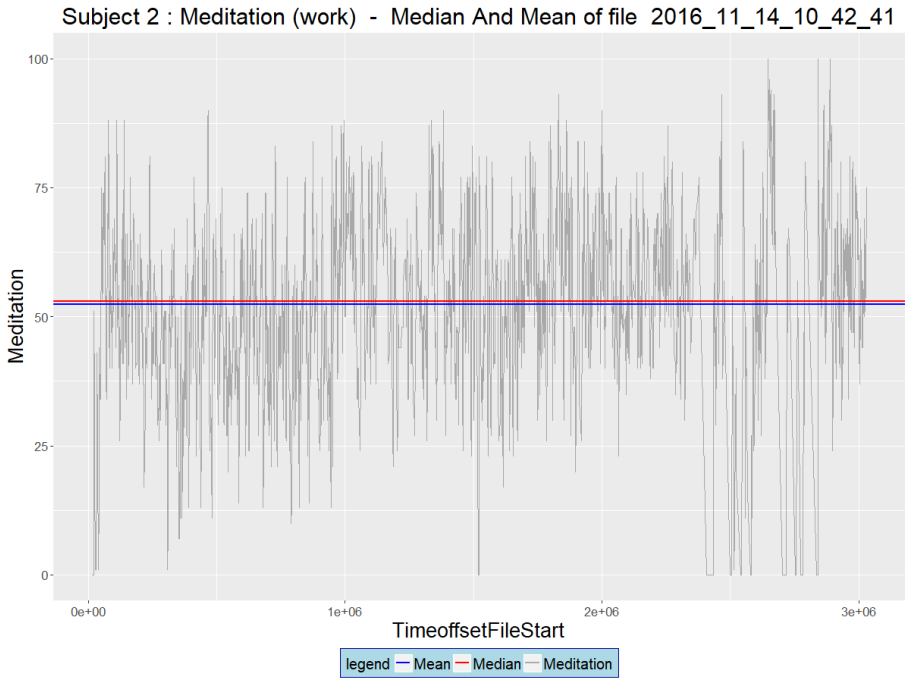



Figure 18 - Time series representing meditation levels (axis y) over time (axis x). The red line represents the mean and the blue line represents the median.

Figure 19 represents the distribution of attention for subject 1. This graph was generated considering all the data collected from this subject.

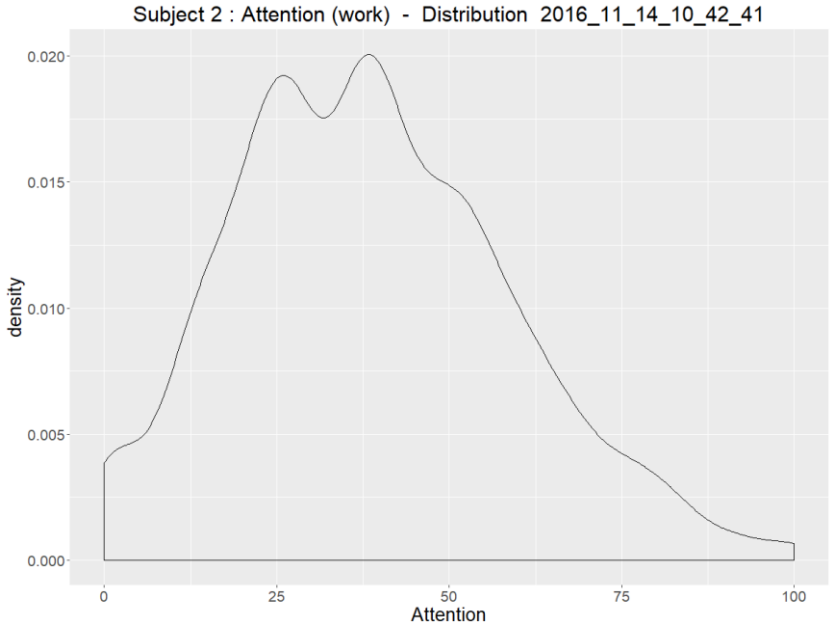



Figure 19 - Distribution of Attention for Subject 1.

We plotted two graphs of the moving average with a window of 50 elements, of attention and meditation shown in Figure 20 and Figure 21, respectively.

The option to use movable average was to minimize fluctuations that normally occur during the execution of activities. Another option we chose was to use the moving average delayed (picking the last fifty measures), the centered average (calculating the average of the 25 previous elements and the 25 following elements) and the average ahead (which calculates the following 50 elements average).

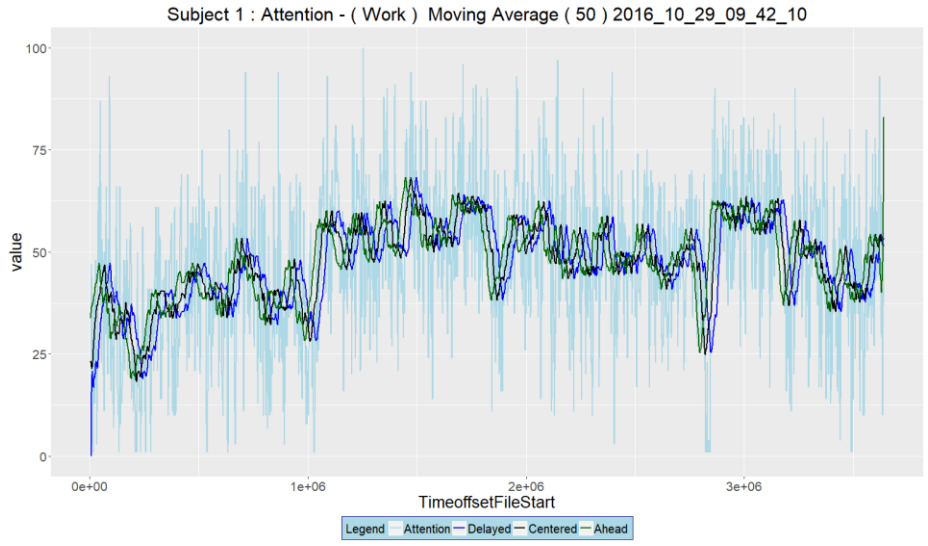



Figure 20 - Attention moving average for subject 1. The ahead mean is plotted in blue, the centered mean is plotted in black and the delayed mean in green.

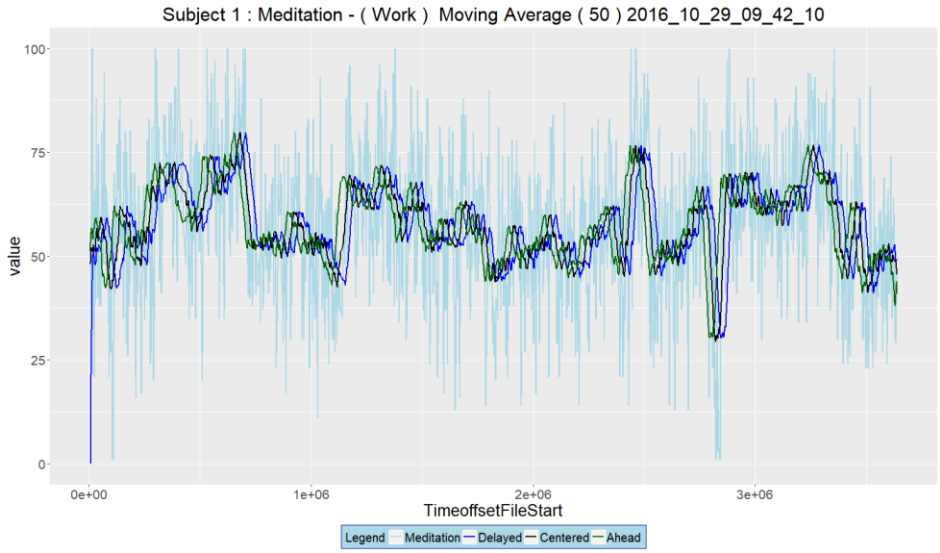



Figure 21 - Meditation moving average for subject 1. The ahead mean is plotted in blue, the centered mean is plotted in black and the delayed mean in green.

By analyzing those graphs we noticed that meditation and attention oscillate at the beginning of the capture, like a cold-starting machine, and that at some point they stabilize at just about an average level. This level then oscillates and stays at a

higher level for a while. However, after performing the activity for some time, both meditation and attention levels drop to an all-time low. We do not have enough data to support by the sensor data alone that the subject was fatigued, but the subject reported being fatigued while performing the experiment.

#### 4.2.4.4 Predictions

We performed the three following machine learning experiments, using the Azure Machine Learning Studio platform:

I- Identifying the user. In this experiment, we aimed at identifying which participant was using the sensor , based on his/her EEG data;

II- Identifying the shift. It was developed to identify the shift (day or night) in which the user was using the wearable sensor, based on his/her Electroencephalography.

III- Identifying the activity. Based on his/her brainwave patterns, we aimed at identifying the activity performed by the user of the sensor.

The experiments II e III were performed considering each participant individually.

##### Experiment I – Identifying the User

For model training, we utilized the Two-Class Support Vector Machine in the standard configuration offered by ML Studio (Figure 22). The Two-Class Support Vector Machine option creates a binary classification model using Support Vector Machine (SVM) algorithm (Pal & Mather, 2005).


Figure 22 - Configuration workflow of Experiment I in Azure ML Studio.

Initially, we filtered our dataset keeping only "Good records" (that is, in this step we deleted bad records and impossible records). This step removed 10,786 samples (approximately 10.67 % of the samples) in total and it was performed via the "Split Data" option available in ML Studio. It divided the dataset into two distinct subsets. In this filtering step we utilized relative expression.

The split of the dataset into two subsets for training/validation and final test of our model was conducted using random split. The first subset (for training) comprised 90% of all data and the second one (for final test) comprised 10% of all data. The training subset was also split into two other subsets using random split, resulting in a subset for model training (85% of the total) and other subset for model validation (15% of the total).

1. Total of instances (total of samples): 90,308;
2. Total of instances for training: 90% of all samples (81,277);
3. Total of instances for final test: 10% of all samples (9,031);
4. Total of instances for model training: 85% of all samples available for training (69,085);
5. Total of instances for model validation: 15% of all samples available for validation (12,192);

All the experiments followed this splitting strategy.

The execution time of the model training was 3 seconds, approximately.

Table 6 shows the accuracy results for our models in the validation and final test phases.

Table 6 - Accuracy results for our validation model and final test model.

| | Validation | Final test |
|---|---|---|
| **Accuracy** | 97.70 % | 98.20 % |
| **Precision** | 99.70 % | 99.90 % |
| **Recall** | 89.00 % | 91.20 % |
| **F1 Score** | 94.00 % | 95.40 % |
| **Positive Label** | subject1 | subject1 |
| **Negative Label** | subject2 | subject2 |
| **True Positive** | 2206 | 1642 |
| **False Negative** | 274 | 158 |
| **False Positive** | 6 | 2 |
| **True Negative** | 9706 | 7229 |

## Experiment II – Identifying the Shift (Subject 1 Results)

As in Experiment I, for model training we utilized the Two-Class Support Vector Machine algorithm in the standard configuration offered by ML Studio (Figure 23). The execution time of the model training was 5 seconds, approximately.

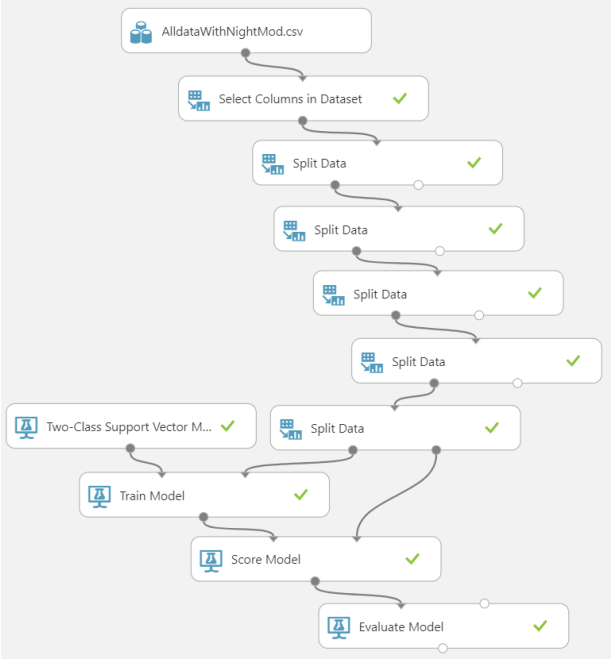



Figure 23 - Configuration workflow of Experiment II in Azure ML Studio.

In Experiment II, we had the following splits for subject 1:

1. Total of instances (total of samples): 18,247;
2. Total of instances for training/validation: 90% of all samples (16,422);
3. Total of instances for final test: 10% of all samples (1,825);

4. Total of instances for model training: 85% of all samples available for training (13,959);
5. Total of instances for model validation: 15% of all samples available for validation (2,463).

For model training, we utilized the Two-Class Support Vector Machine in the standard configuration offered by ML Studio (Figure 23). The execution time of the model training was 4 seconds, approximately.

Table 7 shows the accuracy results for our models in the validation and test phases.

Table 7- Accuracy results for our validation model and final test model

| | **Validation** | **Final test** |
|---|---|---|
| **Accuracy** | 100 % | 100 % |
| **Precision** | 100 % | 100 % |
| **Recall** | 100 % | 100 % |
| **F1 Score** | 100 % | 100 % |
| **Positive Label** | 1 | 1 |
| **Negative Label** | 0 | 0 |
| **True Positive** | 1022 | 744 |
| **False Negative** | 0 | 0 |
| **False Positive** | 0 | 0 |
| **True Negative** | 1441 | 1081 |



## Experiment II – Identifying the Shift (Subject 2 Results)

In Experiment II, we had the following splits for subject 2:

1. Total of instances for subject 2 (total of samples): 72,061;
2. Total of instances for training/validation: 90% of all samples (64,855);
3. Total of instances for final test: 10% of all samples (7,206);
4. Total of instances for model training: 85% of all samples available for training (55,127);
5. Total of instances for model validation: 15% of all samples available for training (9,728).

Table 8 shows the accuracy results for our validation and test phases.

Table 8 - Accuracy results for our validation model and final test model

| | **Validation** | **Final test** |
|---|---|---|
| **Accuracy** | 77.80 % | 78.10 % |
| **Precision** | 70.80 % | 71.00 % |
| **Recall** | 80.90 % | 79.60 % |
| **F1 Score** | 75.50 % | 75.00 % |
| **Positive Label** | 1 | 1 |
| **Negative Label** | 0 | 0 |
| **True Positive** | 3331 | 2374 |
| **False Negative** | 785 | 608 |

| | | |
|---|---|---|
| **False Positive** | 1377 | 971 |
| **True Negative** | 4235 | 3253 |

## Experiment III – Identifying the Activity (Subject 1 Results)

In this experiment, for model training we utilized the Multiclass Logistic Regression algorithm in the standard configuration offered by ML Studio (Figure 24).

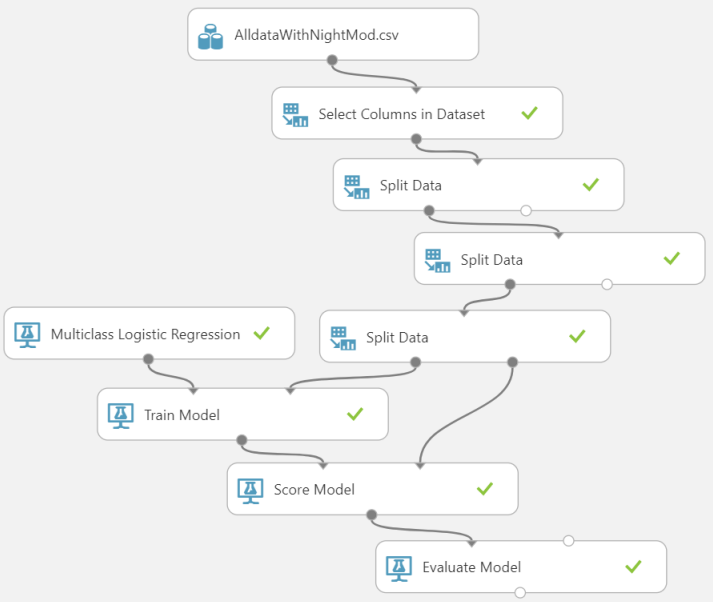



Figure 24 - Configuration workflow of Experiment III in Azure ML Studio.

In Experiment III, we had the following splits for subject 1:

1. Total of instances for subject 1 (total of samples): 18,247;
2. Total of instances for training/validation: 90% of all samples (16,422);
3. Total of instances for final test: 10% of all samples 1,825);
4. Total of instances for model training: 85% of all samples available for training (13,959);
5. Total of instances for model validation: 15% of all samples available for validation (2,463).

Table 9 shows the accuracy results for our models in the validation and final test phases, while Tables 10 and 11 present the confusion matrix results of our validation and final test phases, respectively.

Table 9 - Accuracy results for our validation model and test model

| | Validation | Final test |
|---|---|---|
| **Overall accuracy** | 100 % | 100 % |
| **Average accuracy** | 100 % | 100 % |
| **Micro-averaged precision** | 100 % | 100 % |
| **Macro averaged precision** | 100 % | 100 % |
| **Micro-averaged recall** | 100 % | 100 % |
| **Macro-averaged recall** | 100 % | 100 % |

Table 10 - Confusion matrix results for our validation model

| | Playing | Having class | Watching | Working |
|---|---|---|---|---|
| **Playing** | 100% | | | |
| **Having class** | | 100% | | |
| **Watching** | | | 100% | |
| **Working** | | | | 100% |

Table 11 - Confusion matrix results for our test model

| | Playing | Having class | Watching | Working |
|---|---|---|---|---|
| **Playing** | 100% | | | |
| **Having class** | | 100% | | |
| **Watching** | | | 100% | |
| **Working** | | | | 100% |



## Experiment III – Identifying the Activity (Subject 2 Results)

In Experiment III, we had the following splits for subject 2:

1. Total of instances for subject 2 (total of samples): 72,061;
2. Total of instances for training/validation: 90% of all samples (64,855);
3. Total of instances for final test: 10% of all samples 7,206);
4. Total of instances for model training: 85% of all samples available for training (55,127);
5. Total of instances for model validation: 15% of all samples available for validation (9,728).

The execution time of the model training was 23 seconds, approximately. Table 12 shows the accuracy results for our models in the validation and final test phases. Tables 13 and 14 show the confusion matrix for our models in the validation and final test phases.

Table 12 - Accuracy results for our validation model.

| | Validation | Final test |
|---|---|---|
| **Overall accuracy** | 87.20 % | 87.64 % |
| **Average accuracy** | 93.60 % | 93.82 % |
| **Micro-averaged precision** | 87.20 % | 87.64 % |
| **Macro averaged precision** | 85.30 % | 85.77 % |
| **Micro-averaged recall** | 87.20 % | 87.64 % |
| **Macro-averaged recall** | 86.22 % | 86.83 % |

Table 13 - Confusion matrix results for our validation model.

| | Reading | Surfing | Having class | Watching |
|---|---|---|---|---|
| **Reading** | 89.3 % | | 9.4 % | 1.2 % |
| **Surfing** | | 99.9 % | | 0.1 % |
| **Having class** | 43.9 % | | 56.1 % | |
| **Watching** | 0.4 % | | | 99.6 % |

Table 14 - Confusion matrix results for our test model.

| | Reading | Surfing | Having class | Watching |
|---|---|---|---|---|
| **Reading** | 89.5 % | | 9.2 % | 1.3 % |
| **Surfing** | | 100 % | | |
| **Having class** | 41.4 % | | 58.6 % | |
| **Watching** | 0.8 % | | | 99.2 % |



### 4.2.5. Discussion

In our experiment we explored the brainwave sensor data, and although only two subjects took part in the experiment, we were able to use machine learning to identify each subject, the period of the day he/she was using the wearable sensor and his/her activity with a good accuracy. Furthermore, just by analyzing the movable mean graphs of attention and meditation gives us an idea of how fatigue is reflected in the sensors reading.

As future work, we are considering the possibility of using raw data provided by Mindwave. Other planned improvements are: (i) the use of tools (such as Rescue Time) to support the annotation task; (ii) the definition of the subjects' profiles by identifying which ones have a day-shift profile and which ones have a night-shift profile, based on the time of the day when they are more productive in performing their activities.

Another possibility is to make recommendation about the best time of the day to perform a given activity. We could, for example, suggest the best time of the day to write a paper, taking into account the time of the day in which the subject is most focused and his attention level is at its best.

We also consider the improvement of the data collection app to inform the user when the data quality is not good, as well as the implementation of a machine learning algorithm to detect fatigue.

# 5
# Reasoning about How to Notify to Avoid Alarm Fatigue

In this chapter, we discuss the algorithm we developed to mitigate alarm fatigue by reasoning about how to notify caregivers by grouping similar notifications (Fernandes, Chrystinne, Miles, Simon, Cowan, Donald, & Lucena, C.J.P. de, 2019). Our proposed solution to address this issue decides how to notify caregivers about anomalies detected by a patient monitoring system where a large volume of alarms could lead to alarm fatigue. In other words, we aim to reduce the number of notifications received by health care staff, so they can be focused on the activities that truly require attention. Our experiments were configured to alert nurses and were evaluated through the use of a dataset comprising a wide range of real patient monitoring data recorded during 32 surgical cases where patients underwent anesthesia at the Royal Adelaide Hospital (Liu, Görges, & Jenkins, 2012).

Our main goal for this chapter is to find out whether to group a set of alarms that occurs within a short period of time to deliver them together without compromising patient safety. Our specific goal is to avoid that alarms of the same type for the same patient can be alerted more than once within a short period by using a notification delay strategy.

To pursue our goals, we aim at addressing the following Sub-Question:

**SQ1:** How to reason about whether to group alarms to avoid alarm fatigue?

## 5.1
## Explaining the Reasoner

The main concept behind the reasoner is to choose the best way to notify caregivers to avoid alarm fatigue. As has been mentioned, the default behavior of our anomaly detection process is to trigger an alarm every time an anomaly occurs. For example, a notification would occur even though a number of other alarms of the same type are occurring. However, even though an alarm has been triggered by

our patient monitoring system, the decision of how to notify the caregivers is decided by the Reasoner, using the following rule R1, which states:

- R1. Our system must limit to one the number of notifications (of the same type for the same patient) that caregivers can receive within a defined period of time.

We define Minimum_Notification_Interval (MNI) as the minimum interval of time between receiving 2 notifications by the caregivers. The R1 rule is only applied when we are considering notifications of the same type (TYPE_β) for the same patient P.

Let $\mu_j$ and $\mu_{j-1}$ be 2 notifications of the same type for a given patient P. As shown in Equation (9), a notification can be formally defined as $\mu$=<TYPE_$\mu$, $T_\mu$, $P$>, and in this case we can assume that TYPE_$\mu_j$ is equal to TYPE_$\mu_{j-1}$ and also that $P_j$ is equal to $P_{j-1}$. The time $T_\mu$, at which the notification occurs, allows 2 notifications to be distinguished from each other. The MNI can be formally defined in terms of the notifications $\mu_j$ and $\mu_{j-1}$ as shown below:

$$T_{\mu j} - T_{\mu j-1} >= \text{MNI} \Leftrightarrow (\text{TYPE\_}\mu_j = \text{TYPE\_}\mu_{j-1}) \wedge (P_j = P_{j-1}) \qquad (10)$$

The MNI value must be configured for each patient individually based on patient's context (both of the alarm sources, and patient's criticality).

## 5.2 The Inputs for Our Reasoning Algorithm Related to a Notification

After explaining rule R1, we define the inputs ($I$) for our algorithm as follows:

- I1—CURRENT_ALARM_TRIGGERING_TIME ($T_{\beta r}$). Let $\beta_r$ be the current alarm that has been triggered and is involved in the reasoning process, so the algorithm can decide whether to add a delay to its delivery. The first input for our algorithm is $T_{\beta r}$, that is, the time when the alarm $\beta_r$ was triggered.
- I2—LAST_NOTIFICATION_TIME ($T_{\mu k}$). Let $\mu_k$ be the last notification (of the same type as $\beta_r$) received by the caregivers. The second input for our

reasoning algorithm is the time when caregivers received $\mu_k$, which we represent as $T_{\mu k}$.

As we only consider here current alarms under reasoning and last notifications of the same type and from the same patient, we assume that the alarm types and patients are identical, that is, TYPE_$\beta_r$ = TYPE_$\mu_k$ and $P_{\beta r}$=$P_{\mu k}$.

Another definition is the Last_Notification_Period (LNP), which is the period of time between the 2 inputs for our reasoning as shown in Equation (11).

$$LNP = T_{\beta r} - T_{\mu k} \quad (11)$$

## 5.3
## The Outputs of Our Reasoning Algorithm Related to a Notification

We next define the outputs (O) for our reasoning algorithm as the 2 properties of notifications that can vary depending on the circumstances under which they occur:

O1—NOTIFICATION_DELAY ($D_\mu$). As discussed previously, in Equation (4), $D_\mu$ is the period of time between the alarm triggering event and the delivery of that notification to the caregivers.

O2—NOTIFICATION_DATA (DATAμ). DATAμ refers to the type of data a notification might contain, which depends on the context of the alarm-triggering process, and it might range from a single alarm $\beta_j$ to a set of alarms βSET.

As much as possible, we try to keep the NOTIFICATION_DELAY at a minimum so as not to prejudice patient safety. However, to avoid alarm fatigue, the value for this property can range over an acceptable range of time defined as the BUFFERING_PERIOD, indicating that a DELAY_PERIOD ($\varepsilon$) might be added to the delivery time of the notification under specific conditions (defined in the next section). The BUFFERING_PERIOD is the period of time one or more alarms can be delayed (ie, be held in a buffer) before being delivered to caregivers. See Equation (12).

$$0 < \text{BUFFERING\_PERIOD} < \text{MNI} \quad (12)$$

From Equation (12), we show that an alarm might need to be delayed up to a period equal to MNI. However, the BUFFERING_PERIOD specified for an alarm or a set of alarms should not surpass the value of MNI.

## 5.4 Defining the Grouping Criteria for Notification Delivery—When We Shall Put an Alarm Into Our Buffer

As we said previously, the Reasoner decides the way of delivering the alarm under reasoning ($\beta_r$) by making choices about whether to add a delay $\varepsilon$ to its delivery and whether to group $\beta_r$ with other alarms. To make these choices, the Reasoner must take into consideration our defined inputs ($T_{\beta r}$ and $T_{\mu k}$). By analyzing these inputs, the Reasoner decides whether to queue the current alarm $\beta_r$, based on the following grouping criteria:

- Criteria 1. A same-type alarm was already notified within the MNI.

If caregivers were already notified in the LNP, then the current alarm $\beta_r$ must be queued up into a buffer for the period BUFFERING_PERIOD. After BUFFERING_PERIOD has passed, $\beta_r$ is delivered along with other possible alarms in the buffer as a unique notification.

Just to clarify, when the circumstances for the alarms do not meet the abovementioned grouping criteria, a notification containing an individual alarm is sent to the caregivers as soon as an alarm has been triggered, that is, immediately after $T_{\beta r}$.

As important as it is to avoid alarm fatigue, the Reasoner must handle the notification delivery process without putting patients at risk. In this case, the delay added to the notification delivery must not prejudice the requirements established regarding patient safety.

## 5.5 The Pseudocode for Our Reasoning Algorithm About How to Notify

The pseudocode for our reasoning algorithm about how to notify is shown in Textbox 1.

Textbox 1. The pseudocode for the reasoning about how to notify.

```
DEFINE LNP, Tβr, Tμk, MNI;
```

```
// Receive Input CURRENT_ALARM_TRIGGERING_TIME Tβr;
INPUT Tβr;
// Receive Input LAST_NOTIFICATION_TIME Tμk;
INPUT Tμk;
// Calculate LNP
LNP = Tβr - Tμk;
// If LNP is equals to Tβr (meaning that no notification μk occurred to the patient in the last MNI-period) or LNP is higher than or equal to MNI (which means that a notification μk occurred more than MNI-period ago) then notify βr immediately. Otherwise, put βr into the buffer
If (LNP == Tβr || LNP >= MNI) then
        //There is no need for putting βr into the buffer. Notify it immediately
        Notify(βr);
Else {
        // We need to put βr into the buffer and deliver it after some delay
        QueuedUp(βr)
        // If βr is the first alarm been put into the buffer then {
        Ìf (isAlarmTheFirstOneQueuedUp(βr)) then {
                // Define buffer's property STARTING_TIME as the time the alarm was triggered;
                STARTING_TIME:= Tβr;
                // Create a new thread for handling the buffer in parallel. This thread needs to
                // control the BUFFERING_PERIOD (BP) for notifying caregivers after BP has passed
                Create a new thread;
                Start BUFFERING_TIME;
                If BUFFERING_PERIOD has passed then
                //Release the content of buffer to caregivers by wrapping the set of alarms
                //(alarmsSet) into a single notification and sending it
                        Notify(alarmsSet);
        }
}
```

## 5.6 Methods

In this chapter, we present a new approach to cope with the alarm fatigue problem. Our proposed solution focuses on an automatic reasoner that is used to

decide how to notify caregivers about anomalies detected by a patient monitoring system through the use of a notification delay strategy.

To confirm the fulfillment of the main research goal, the experiment described next was conducted and results are tabulated in the Discussion section.

### 5.6.1 Hypotheses

We defined the following hypotheses for our case study:

**H1.** The caregivers should not receive more than one notification about the same type of anomaly for the same patient within the MNI.

**H2.** Patient safety will not be compromised by the use of the reasoning algorithm about how to notify.

### 5.6.2 Methodology

To illustrate the operation of our reasoning algorithm, we conducted 5 experiments to evaluate how the algorithm works under different scenarios, considering mainly the number of alarms generated in each experiment.

### 5.6.3 Applications Settings

As shown in Table 15, to run an experiment, we need to define the following settings for our application scenarios:

- The number of wards occupied by patients (NUMBER_OF_WARDS).
- The number of patients being monitored (NUMBER_OF_PATIENTS) by a caregivers team.
- The number of sensors used during monitoring (NUMBER_OF_SENSORS).
- The interval in which the sensor readings are being monitored (SENSORS_READING_INTERVAL).

- The number of sensor readings (NUMBER_OF_READINGS). This information, along with the SENSORS_READING_INTERVAL, tells us how long the patients in our experiment are being monitored.

Table 15 - Defining the configuration for our 5 experiments.

| | |
|---|---|
| **Number of wards** | 1 |
| **Number of patients** | 1 |
| **Number of sensors** | 1 |
| **Sensors reading interval (ms)** | 1000 |
| **Number of readings** | 60,000 |

We also need to define the thresholds for each sensor and the MNI, considering each patient individually (Table 16). As has been mentioned earlier, the MNI is defined by taking into account both of the alarm sources, and the patient's criticality to respect patient safety constraints. In our simulated environment, we defined the MNI value as 5 minutes for every patient and we assume the delivery of the type of anomalies triggered in our context (which are related to heart rate values) can be delayed up to this period without representing any danger for the patients.

All the inputs for our reasoning were provided through a vital signs streaming app, we developed for streaming vital signs retrieved from a dataset comprising real patient data. The dataset provides clinical anesthesia monitoring data from 32 entire surgical cases, including a wide range of vital signs variables, such as electrocardiograph, pulse oximeter, capnograph, noninvasive arterial blood pressure monitor, airway flow, and pressure monitor, and in a few cases, a Y-piece spirometer, an electroencephalogram monitor, and an arterial blood pressure monitor. The monitoring data were collected using Philips IntelliVue MP70 and MP30 patient monitors and Datex-Ohmeda Aestiva/5 anesthesia machines. In this dataset, a single stream of raw monitoring data was recorded in a comma-separated values (CSV) text file format at a sampling resolution of 10 milliseconds (Liu et al., 2012).

We evaluated our algorithm by using data that we selected from 3 out of the 32 surgical cases in the dataset (cases 4, 7, and 14). Experiment 1 was conducted using data from case 4, while, in experiment 2, we utilized data from case 14, and, finally, experiments 3-5 were executed using data from case 7. In all the

experiments, we utilized the version of processed data available in the CSV format for monitoring patients based on their heart rate parameter at 1-second intervals (our algorithm uses this frequency instead of the 10-millisecond sampling resolution available at the dataset). However, the number and type of vital signs used in every experiment could vary to simulate other configurations for sensors and monitoring devices in an ICU.

To define when a given heart rate reading represented an anomalous value that should trigger an alarm, we defined the thresholds in Table 16 for each patient.

Table 16 - Defining the anomaly thresholds of heart rate sensor for each patient.

| Experiment | Patient_ID | Min_heartrate | Max_heartrate |
|---|---|---|---|
| 1 | 1 | 60 | 100 |
| 2 | 2 | 55 | 100 |
| 3 | 3 | 50 | 105 |
| 4 | 4 | 50 | 100 |
| 5 | 5 | 50 | 102 |



## 5.7 Results

### 5.7.1 Application Details - Technologies Utilized

The application was developed in the Java language along with the use of the RabbitMQ (*RabbitMQ*) message broker. RabbitMQ is an open-source message broker that accepts, stores, and forwards messages. The basic concepts behind this technology are Queue, Producer, and Consumer (Figure 25). A Queue is essentially a large message buffer that stores the messages, while a Producer and a Consumer are both user applications. The former is a program in charge of sending messages to the queue through the exchanges, and the latter consists of a program that receives messages from the queue. A program can be both a Producer and a Consumer at the same time.

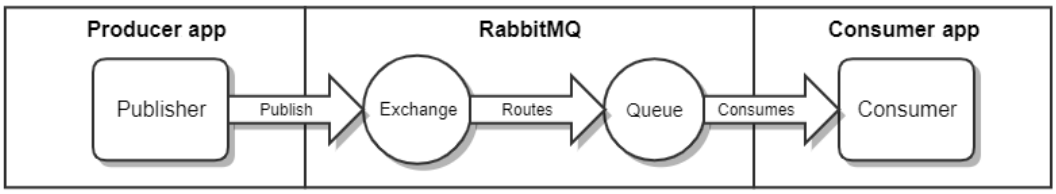



Figure 25 - Basic concepts and information flow in RabbitMQ.

As can be seen from Figure 25, a broker receives messages from publishers (producers) and routes them to the consumers. The information flow involved in this process occurs in 2 steps, described as follows:

- Step 1. The producers send messages to exchanges that act by distributing messages to queues using rules called bindings.
- Step 2. The broker either delivers messages to consumers subscribed to queues or consumes pull messages from queues on demand.

In this application, we used the Advanced Message Queuing Protocol 0-9-1 Java client provided by RabbitMQ, which is an open and general-purpose protocol for messaging.

Owing to the high volume of notifications we are dealing with in our application, we decided to utilize a solution that could take care of the nonfunctional requirements of our system. By using a solution to handle problems related to scalability and safety, we could focus on the functional requirements of our application. Therefore, we decided to use the RabbitMQ to meet the high availability, throughput, and scale requirements of our application domain. This message broker solution offers features related to data safety such as reliable delivery, which means it can ensure that messages are always delivered, even encountering failures such as network failures and consumer application failures.

### 5.7.2 Explaining How Our Application Works

In a high abstraction level, the main idea of this app is to have an application that sends alarms to a broker that routes them to a consumer app that represents the receiving of these alarms by the health care team.

We chose the type of exchange called *topic* for routing the messages. The topic exchange routes messages to one or many queues based on matchings between

a message routing key and the pattern that is used to bind a queue to an exchange. We declared one queue named *sensor_readings* to where the publisher sends the data and the consumer receives data. We also declared the binding key for our consumer (ie, the class that is consuming heart rate data) as *#.heartrate* (Figure 26).

The routing key is defined based on the pattern <patientID>.<heartrateValue>. For example, we could have a routing key as *16.88*, representing a patientID=16 and heartrateValue=88.

The notifications sent to health providers are created based on this message. In this case, the final notification received by nurses contains information related to the patient, such as identification, location, and vital signs.

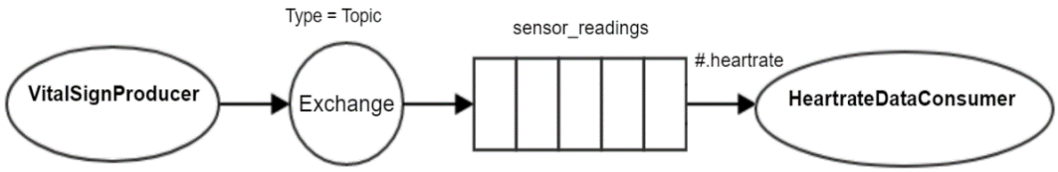



Figure 26 - RabbitMQ scheme utilized in our application.



### 5.7.3
### Application Modeling - Class Diagram

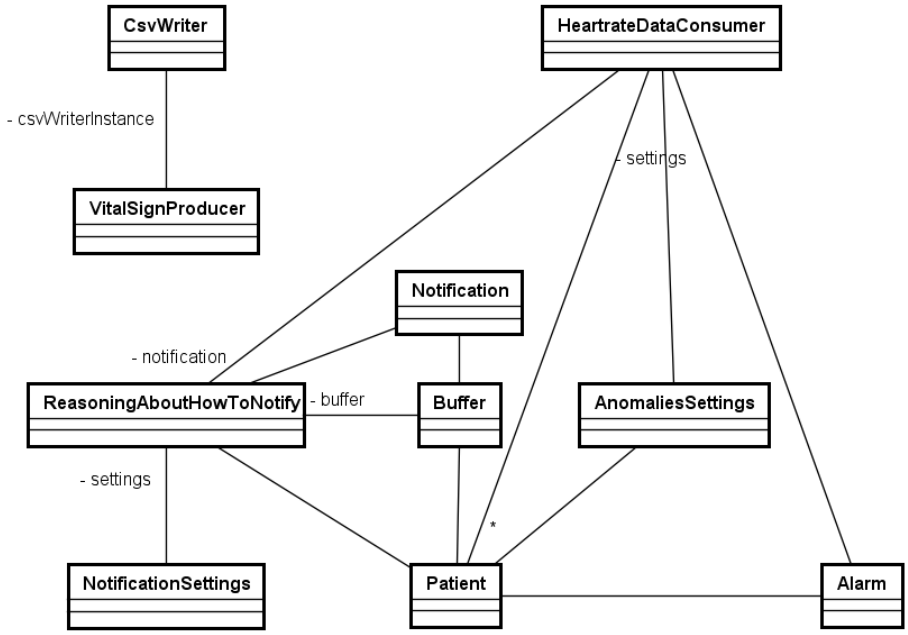
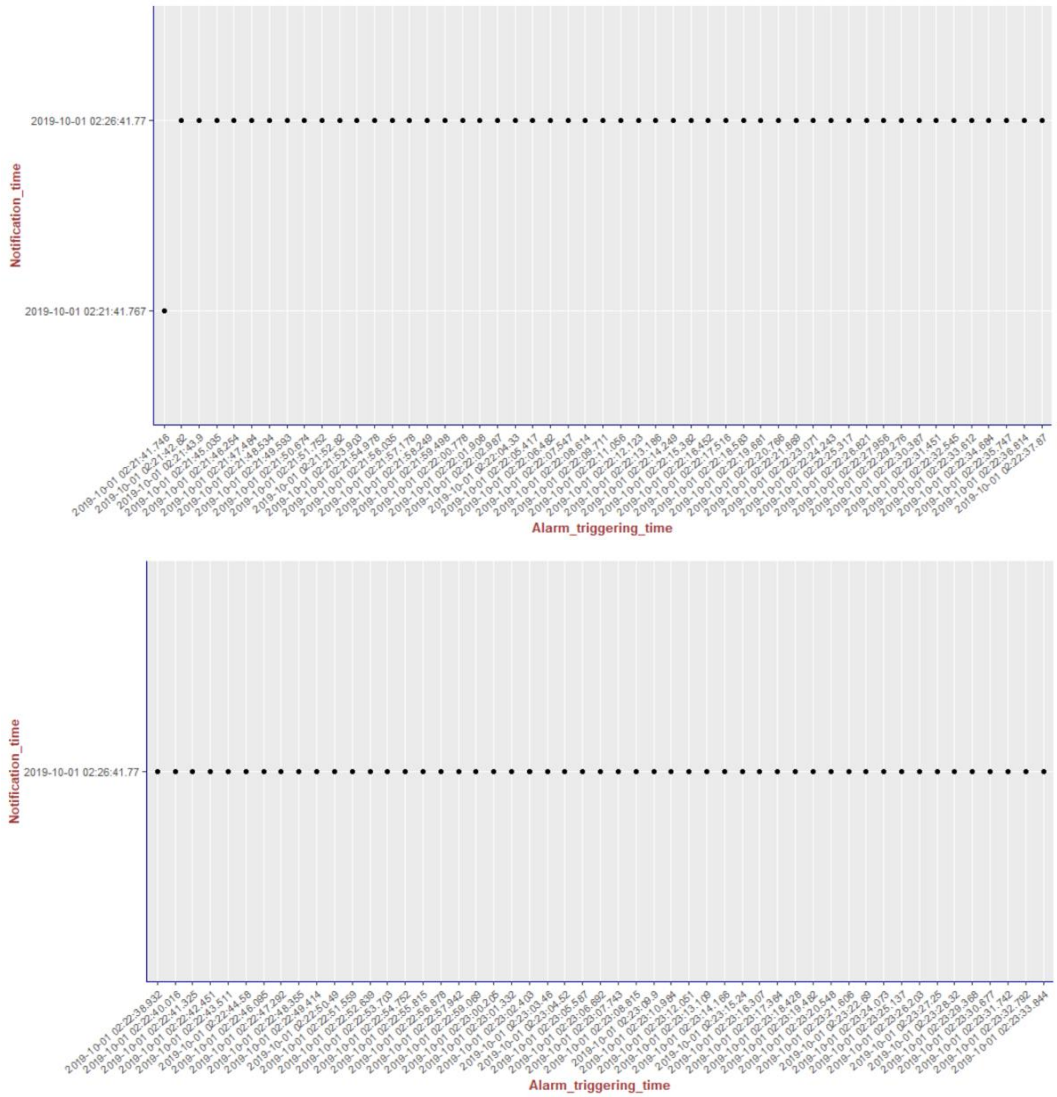



Figure 27 - The class diagram for our application, where the consumer application monitors a specific vital sign based on the anomalies settings defined for each patient.

In Figure 27, as can be seen from the class diagram for our application, the consumer application monitors a specific vital sign based on the anomalies settings defined for each patient. The consumer app invokes the reasoning mechanism through the ReasoningAboutHowToNotify class, which knows how to notify based on the defined notifications settings (eg, the MNI value configured for each patient).

We present the results of our algorithm by using graphs we generated using the R language and the ggplot2 library. The graphs shown in Figure 28 illustrate the delivery process of all notifications related to the patient monitored in experiment 5 (PatientID=5). We show whether the algorithm decided to deliver an alarm immediately or after a delay by grouping alarms to deliver them together.

To better visualize the results of experiment 5 through the graphs, we split the output data of our algorithm for this experiment (comprising a total of 204 alarms) into 4 pieces of data containing 51 alarms each. Thus, we plot each piece of data into a graph, showing the alarm triggering time through the *x*-axis and the notification time on the *y*-axis. As can be seen from Figure 28, the occurrence of the first notification (NotificationID=1) of an alarm of heart rate for patient 5 occurred at the notification time *2019-10-01 02:21:41.767*, that is, almost immediately after the occurrence of the first alarm (that occurred at the alarm triggering time *2019-10-01 02:21:41.746*). Following the strategy of our reasoning algorithm, the next notification of an alarm of heart rate for this patient should not be received by the caregivers before MNI. As in this experiment MNI corresponds to 5 min, the timestamp for the next delivery of a heart rate alarm related to patient 5 should occur at least 5 min after *2019-10-01 02:21:41.767*. As can be seen in Figure 28, the next heart rate alarms for patient 5 were held in the alarms buffer and delivered together at the timestamp *2019-10-01 02:26:41.77* as a unique notification (NotificationID=2) with a delay of approximately 5 min.



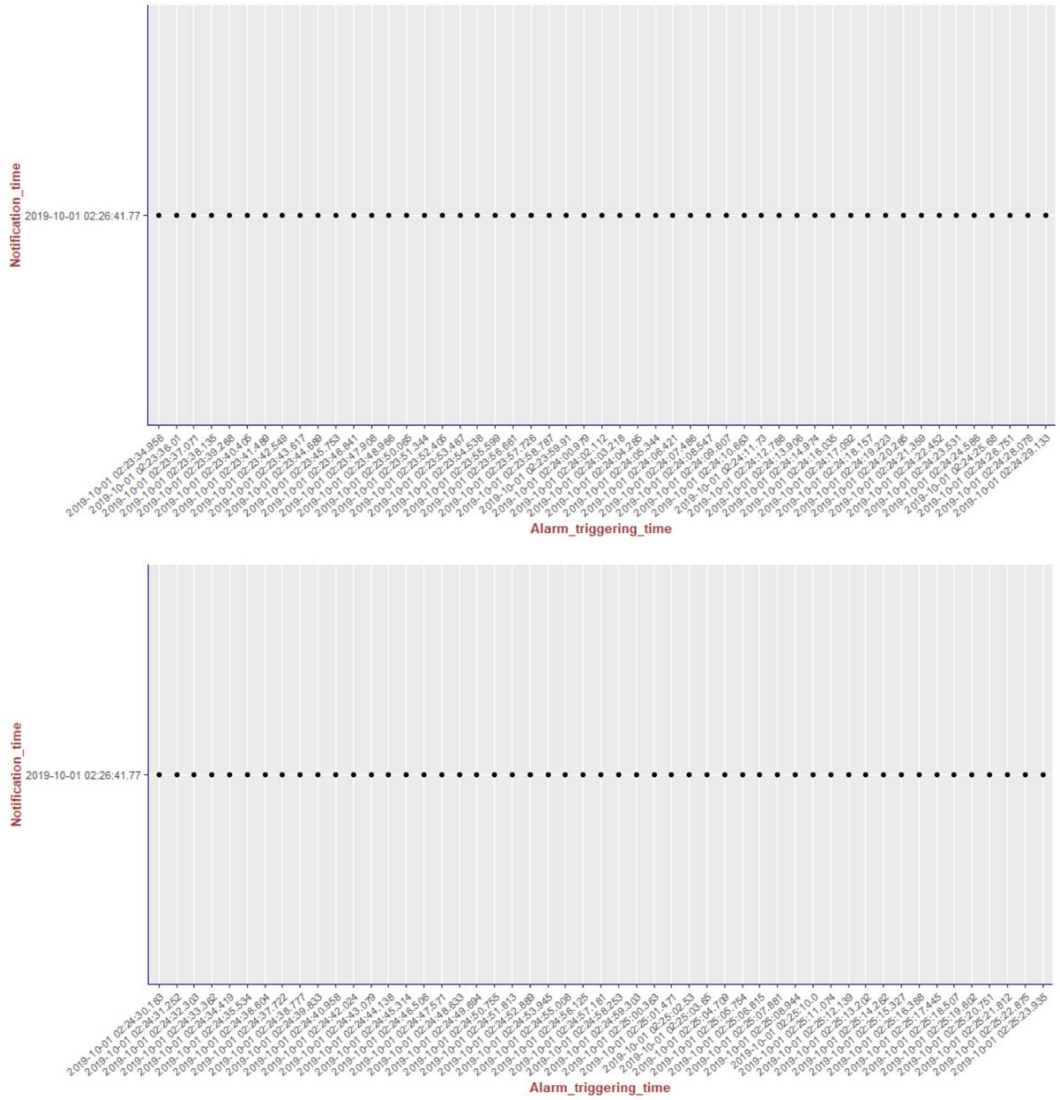



Figure 28 (a-d) - Illustrations of the results of the alarm triggering and delivery processes related to the patient monitored in our experiment 5 (PatientID=5).

Figure 29 illustrates the results of the delivery processes related to all patients monitored in our experiments (PatientID = 1,2,3,4, and 5, respectively).

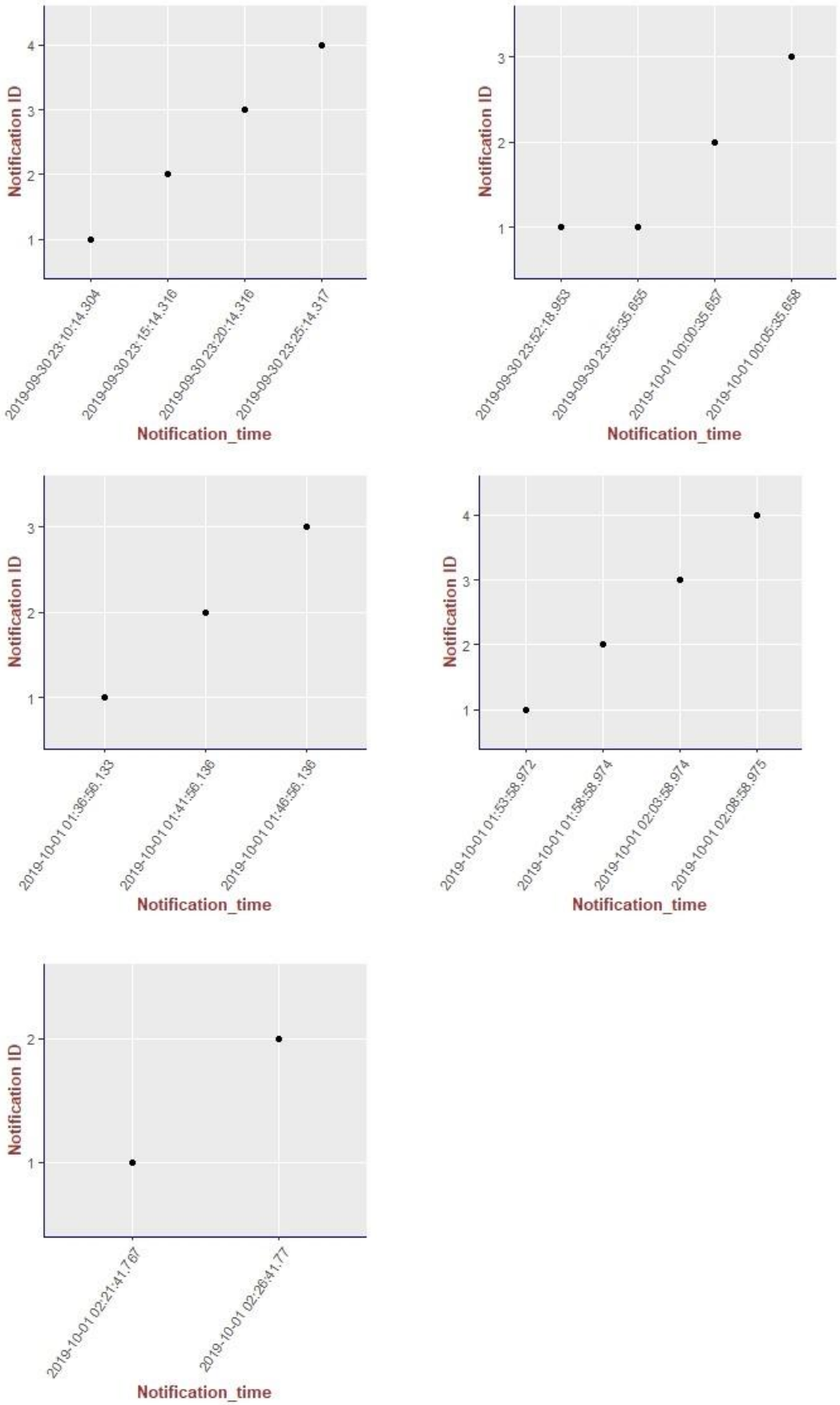



Figure 29 - (a-e). Illustrations of the results of the delivery process related to all our experiments.

We show the results for all of our experiments summarized in Table 17, where we can compare the number of alarms triggered by our system in each experiment with the number of notifications delivered to the caregivers.

Table 17 - Results of our experiments to evaluate our reasoning algorithm about how to notify caregivers considering the reduction of the number of notifications received by them.

| Experiment | Number of heart rate alarms | Number of heart rate notifications | Notifications in relation to the total of alarms (%) | Reduction in alarms received (%) |
|---|---|---|---|---|
| 1 | 407 | 4 | 0.9 % | 99.0 % |
| 2 | 423 | 3 | 0.7 % | 99.2 % |
| 3 | 308 | 3 | 0.9 % | 99.0 % |
| 4 | 586 | 4 | 0.6 % | 99.3 % |
| 5 | 204 | 2 | 0.9 % | 99.0 % |

## 5.8 Discussion

### 5.8.1 Conclusions

The first hypothesis (H1) we want to evaluate with this case study says that the caregivers should not receive more than one notification about the same type of anomaly for the same patient within the defined MNI. By executing our reasoning algorithm throughout the experiments, we saw that H1 holds for all of them, as within all the occurrences of notifications for each patient, there is no occurrence of a notification of the same type within the defined MNI. We support this affirmation by presenting, in Figure 29, a summary of the results from our experiments using graphs containing all notifications that occurred in each experiment. As can be seen, considering all experiments, there was no occurrence of delivery of notifications of the same type for the same patient that occurred before the specified delay, that is, the MNI value of 5 minutes.

The hypothesis H2, in turn, stating that patient safety will not be compromised by the use of the reasoning algorithm about how to notify also holds, as the notification interval (MNI) we defined is no longer than 5 min. This means that a group of alarms that are occurring to a given patient can be held in a buffer for, at most, 5 min before the buffer is fully released to the caregivers as a unique notification. However, in order not to prejudice patient safety, the first occurrence

of an alarm is always delivered to the caregivers immediately after its occurrence. In this case, only the next occurrences of the alarms are delivered to caregivers with the addition of a given delay.

In Table 17, we made a comparison between the number of alarms triggered by our system and the number of notifications delivered to the caregivers, in each experiment. These results show that the reduction of the notifications received by the caregivers can be up to 99.3% (582/586) of the total of alarms, with a mean of 99.17% (1912/1928) of reduction in the number of total alarms, considering all the experiments.

According to Winters et al, nearly all studies assume that a reduction in the number of total alarms and/or false alarms will reduce alarm fatigue (Winters et al., 2018). Thus, by presenting these results, we expect that our algorithm can be used as a useful strategy for avoiding alert fatigue. We also expect our approach can be useful for helping to prevent its negative consequences, such as disruption of patient care, disabling of alarm systems by staff, reduction in responding, lack of caregiver response, and real events being less likely to be acted on, among others.

# 6
# Reasoning about How to Detect False Alarms by Analyzing Alarm-context Information

In this chapter, we discuss how we can reason about adding an indication of a false alarm probability to a notification (FAP_LABEL) (Fernandes, Chrystinne & Lucena, C.J.P. de, 2019). The FAP_LABEL is calculated according to false alarm indicators we defined based on our literature review. The idea is to use these indicators to decide when an anomaly reported through a notification could be analyzed with a low priority level. In this case, the FAP_LABEL could be utilized to assist caregivers in prioritizing the next alarm to attend to.

## 6.1.
## Problem Definition

In the systems built through our software framework (e.g., the depth of anesthesia monitoring app), the anomaly detection process worked by triggering an alarm every time an anomaly occurs, independently of the circumstances (Chrystinne Oliveira Fernandes & de Lucena, 2015; Chrystinne Oliveira Fernandes, de Lucena, de Lucena, & de Azevedo, 2016). However, many times these alerts are false alarms that do not represent real danger for patients. In this case, the lack of use of any intelligent filter to detect an indication of false alarms before alerting health providers can culminate in a context of a sensory overload for the medical team. This context can result in alarm fatigue and compromise the health providers' attention, leading them to miss relevant alarms that might indicate significant harmful events.

As a strategy to mitigate the alarm fatigue issue, in this chapter we present a new approach to monitor patients by using a notification process supported by a reasoning mechanism. This mechanism associates a FAP to alarms based on its real-time context information, including: (i) information about a patient's circumstances, such as his/her repositioning in bed, and localization (which is tracked in real-time by the use of wearable devices with Global Positioning System

(GPS)), and (ii) information about sensors, including battery charge life, the last time the patient's skin was prepared to receive electrodes and the last time electrodes were changed, among others.

After receiving this context information as inputs, the Reasoner's work begins by analyzing each alarm and calculating the FAP associated to it according to the false alarm indicators we defined. Thus, the Reasoner uses the FAP calculated for each alarm to decide whether to include an indication of false alarm probability (FAP_LABEL) to a notification that can be visualized by caregivers.

## 6.2. Goals and Contributions

This chapter's main goal is to propose a solution to mitigate alarm fatigue by using an automatic reasoning mechanism to assist caregivers in their decision-making process of choosing the next alarms to which they should respond. Our specific goal is to attribute a false alarm probability to an alert based on the context in which it has been generated, such as: patient's conditions and information about monitoring devices and sensors. We aim at reasoning about the probability of an alarm being a false alarm in order to decide whether to enrich the notifications sent to caregivers with this information (FAP_LABEL).

We addressed the following Sub-Questions:

**SQ2.** How can an automatic reasoning system calculate an indication of FAP for an alarm generated by sensors and monitoring devices?

**SQ3.** How to reason about whether, or not, to add an indication of a false alarm probability to a notification that could be visualized by the healthcare team?

The main contributions achieved in this chapter were: (i) a list of the false alarm indicators we defined that can be utilized and possibly extended by other researchers; (ii) a novel approach to assess the probability of a false alarm using statistical analysis of multiple inputs representing the alarm-context information; (iii) a reasoning algorithm that uses alarm-context information to detect false alarms in order to decide whether to notify caregivers with an indication of FAP to avoid alarm fatigue.

## 6.2
## Material and Methods

With regard to methodology, we present a new approach to mitigate the alarm fatigue issue. We developed an application that attributes a FAP to the alarms based on the false alarm indicators we defined. Our reasoning algorithm uses the calculated FAP to decide whether to include an indication of FAP to a notification (FAP_LABEL) before sending it to caregivers in order to assist them in the complex task of choosing the next alarms to which they should respond.

To achieve our main research goal, the experiment described below was conducted and results are displayed in the Discussion session.

### 6.2.1
### Hypotheses

We defined the following hypotheses for our case study:

> **H3.** Our reasoning algorithm should associate a FAP value to every alarm generated by sensors and monitoring devices in our experiments.
>
> **H4.** Our reasoning algorithm should add an indication of a false alarm probability (FAP_LABEL) based on which the reasoner should decide whether, or not, to notify caregivers.
>
> **H5.** Patient safety should not be compromised when, and if, the reasoning algorithm decides to add a FAP_LABEL to the notification.



### 6.2.2
### Reasoning Model to Decide Whether to Include a FAP Label to a Notification

In our system, a notification is a type of message that is sent to caregivers and contains information about a detected alarm (or a group of alarms). A FAP is a false alarm probability associated to an individual alarm that we calculate according to the false alarm indicators we describe next. While a FAP_LABEL, on the other hand, corresponds to the probability of a notification containing a false alarm.

We calculate the FAP of every alarm triggered by our system. However, the reasoning algorithm decides whether to include the indication of FAP to a

notification (as the FAP_LABEL) based on the false alarm indicators. The FAP_LABEL is the piece of information that can be visualized by caregivers.

The inputs for our algorithm are a notification and its context information, including information about the patient's conditions and sensors. After receiving these inputs, the Reasoner starts working by analyzing the notification content and calculating the FAP_LABEL associated to it.

The processes to calculate the FAP and FAP_LABEL are described below. Figure 30 presents a state machine diagram of the FAP reasoning process considering each alarm individually. In Figure 30, we present the reasoning modeling process that decides whether to notify caregivers through a FAP_LABEL indication.

### 6.2.3 Explaining how we calculate FAP based on the False Alarm Indicators (FAI)



To calculate the FAP associated to each alarm, we defined four indicatives of false alarms based on the information we gathered in our literature review. According to Kerr & Hayes, the main events that cause false alarms are patient movement or repositioning in bed and poor placement of sensors. Another common issue that triggers alarms is related to technical problems, such as the lack of a battery in the monitoring devices.

The four false alarm indicators defined in this case study represent information about (i) the duration of a sensor battery and the last time it was changed, (ii) the last time the patient's skin was prepared to receive electrodes and the last time they were changed, (iii) the patient's mobility, and (iv) the patient's position in bed. To calculate the false alarm indication percentage, in our experiment we considered that each indicator has the same weight. We list below our false alarm indicators:

- **FAI1:** Sensor battery false alarm indicator (SENSOR_BATTERY_FAI). This is an indication of the FAP associated to the battery charge level of the sensors attached to the patient;

- **FAI2:** Placement of sensor false alarm indicator (PLACEMENT_OF_SENSOR_FAI). FAI2 is related to the placement of a sensor, i.e., if a sensor is properly in touch with the patient's skin;
- **FAI3:** Patient mobility false alarm indicator (PATIENT_MOBILITY_FAI). This indicator is related to patient mobility, which means that it can evaluate the probability that the alarm has been triggered due to his/her movement from the bed to other places;
- **FAI4:** Patient repositioning false alarm indicator (PATIENT_REPOSITIONING_FAI). This indicator can be used to calculate the FAP related to patient repositioning, i.e., if the alarm has been sent simply because the patient may have changed his/her position in bed;

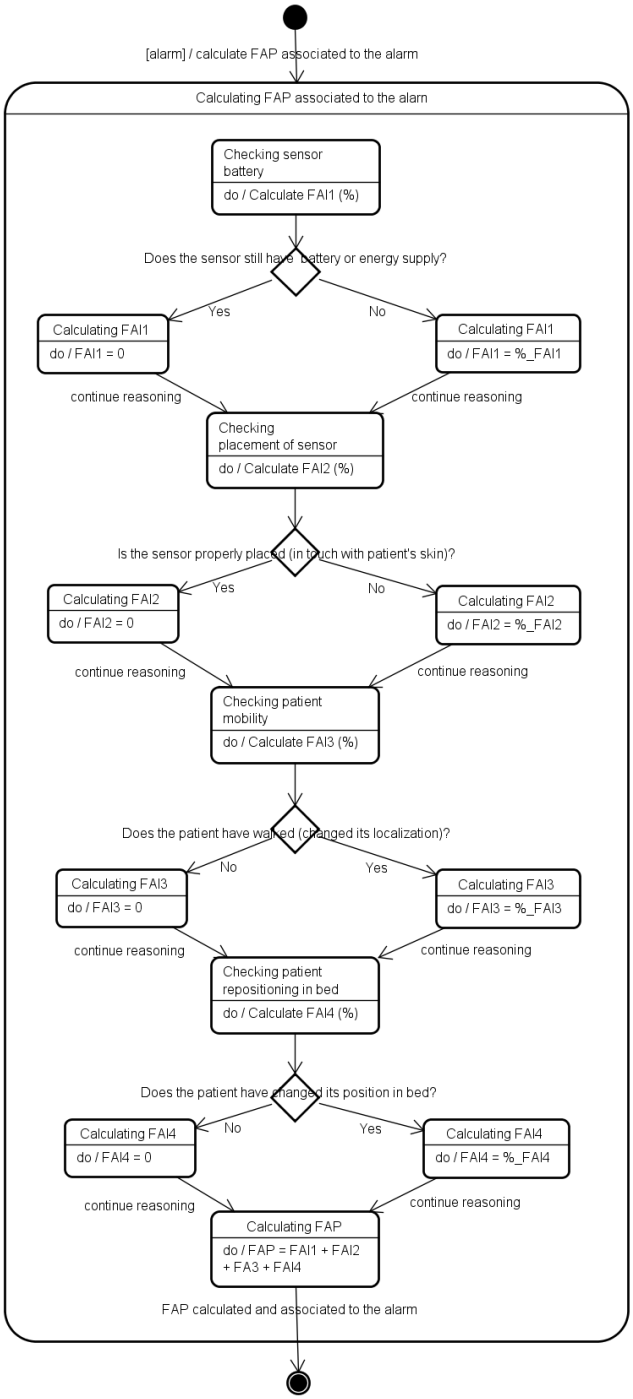



Figure 30 - State Machine Diagram showing how we calculate the FAP associated to an alarm.

### 6.2.4 Inputs for our reasoning algorithm about whether to add a FAP_LABEL

As shown in Table 18, we defined eight inputs for our algorithm. There are four types of information that need to be manually inserted into our system by caregivers (Inputs 1-4), two types of data automatically collected via sensors

(Inputs 5 and 7), and, finally, two inputs (Inputs 6 and 8) that are retrieved from the database by the system as historical patient data.

Every input mentioned above is related to one of the four false alarm indicators, as described below:

Table 18 - Inputs for our reasoning algorithm.

| Input | Input name | It used to calculate the following FAI | Description | Type of related monitoring device |
|---|---|---|---|---|
| 1 | LEVEL_OF_ BATTERY | FAI1 (SENSOR_ BATTERY_FAI) | Level of battery for each monitoring device (including multi-parametric monitors) | Monitoring devices that use battery |
| 2 | LAST_TIME_ BATTERY_ CHANGED | FAI1 (SENSOR_ BATTERY_FAI) | Last time device's battery was changed | Monitoring devices that use battery |
| 3 | LAST_TIME_ SKIN_ PREPARATION | FAI2 (PLACEMENT_OF _SENSOR_FAI) | Last time skin preparation occurred | Sensors that use electrodes |
| 4 | LAST_TIME_ ELECTRODES_ CHANGED | FAI2 (PLACEMENT_OF _SENSOR_FAI) | Last time electrodes were changed | Sensors that use electrodes |
| 5 | CURRENT_ PATIENT_ LOCALIZATION | FAI3(PATIENT_ MOBILITY_FAI) | The current patient's localization | Sensors used to track patient localization |
| 6 | LOG_LAST_ PATIENT_ LOCALIZATION | FAI3 (PATIENT_ MOBILITY_FAI) | A log of patient's last localization | Sensors used to track patient localization |
| 7 | CURRENT_ PATIENT_POSI-TION_IN_BED | FAI4 (PATIENT_REPO-SITIONING_FAI) | The current position a patient occupies in a bed | Sensors used to track patient position in bed |
| 8 | LOG_LAST_ PATIENT_POSI-TIONS_IN_BED | FAI4 (PATIENT_REPO-SITIONING_FAI) | The last positions a patient has occupied in a bed | Sensors used to track patient position in bed |

### 6.2.5
### Output of our reasoning algorithm

There is one output of our algorithm:

- Output1: The probability of an alarm be false (FAP);

### 6.2.6
### Application's Details – Technologies utilized, Scenario and Settings

To test our reasoning algorithm, we developed a system comprising an application (the Producer App) that sends alarms to a broker who routes them to consumer applications that receive these alarms on behalf of the healthcare team. The system was developed in the Java language using the RabbitMQ message broker. The reason we decided to use RabbitMQ to handle the features related to data safety and scalability is to allow us to focus mainly on our functional requirements, since we are dealing with a high volume of alarms in our system.

### 6.2.7
### Applications Scenario

The application scenario consists of a group of four patients being monitored in an ICU by using sensors and monitoring devices, such as: multi-parametric monitors (Figure 24), wearable devices, and external sensors that can be utilized with micro-controllers (Figure 31).

In our simulated scenario, a group of caregivers is in charge of dealing with the alarms generated. In addition to other demanding tasks, caregivers need to update the system with information about monitoring devices (i.e., the inputs for our reasoning algorithm that need to be manually inserted by them). However, our algorithm also receives other inputs automatically via sensors, such as the patient's localization, and the patient's position in bed. To retrieve the patient's localization, we used a wearable device, such as a GPS-equipped bracelet. The patient's position, in turn, is given via a body position sensor (Figure 32-c) (*E-Health Sensor Platform*). Figures 31 and 32 show examples of monitoring devices we utilized during this research to collect biometric patient data.

### 6.2.8
### Monitoring Devices to Collect Biometric Patient Data

The CM100 Efficia Philips monitor (*CM100 Efficia Philips Monitor*) is commonly utilized to collect vital signs, such as ECG, breathing, temperature, noninvasive blood pressure (PNI), oximetry (SpO2), capnography (EtCO2), invasive blood pressure (IBP).


Figure 31 - The CM100 Efficia Philips Monitor.

The e-Health Sensor Platform Complete Kit (*E-Health Sensor Platform*) (Figure 32-c) contains an e-Health Sensor Shield (Figure 32-b) compatible with Arduino (Figure 32-a) and Raspberry Pi (*Raspberry Pi*) microcontrollers along with 10 sensors to collect biometric data (Figure 32-c): pulse, oxygen in blood, airflow (breathing), body temperature, electrocardiogram (ECG), glucometer, galvanic skin response, blood pressure, patient position (accelerometer) and muscle/electromyography sensor (EMG).

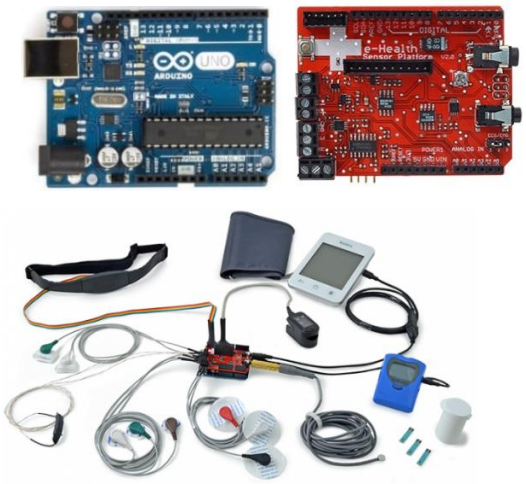



Figure 32-a. Arduino micro-controller. 32-b. e-Health Sensors Shield. 32-c. e-Health Sensor Platform Complete Kit.

### 6.2.9
### Application Settings

In our simulated environment, patients were monitored through the use of two sensors: heartrate and temperature. The sensor readings were generated by the vital signs simulator we developed. Regarding the sensor data simulated for each sensor, the temperature readings were generated randomly by the simulator within the 35.0-

42.0 range and the heartrate readings were randomly selected from the 40-188 range. To define when a given temperature and heartrate reading represented an anomalous value that should trigger an alarm, we defined the thresholds shown in Table 19 for each patient.

Table 19 - Defining the anomaly thresholds of temperature and heartrate sensors for each patient.

| Patient_I | Min_temperatur | Max_ | Min_heartrat | Max_ |
|---|---|---|---|---|
| 1 | 35.5 | 39.0 | 60 | 100 |
| 2 | 35.0 | 38.5 | 55 | 95 |
| 3 | 35.5 | 39.5 | 60 | 100 |
| 4 | 35.5 | 38.5 | 50 | 100 |

In our experiment, we set at 75% the FAP_NOT_MIN (i.e., the value used as a reference to decide whether to add the FAP_LABEL to the notification). This means that every time the calculated FAP for an alarm is higher than or equal to 75%, our Reasoner adds the FAP_LABEL to the notification. Otherwise, we set the FAP_LABEL in our dataset as "UNDEFINED", meaning that it will not be included in the notification as an additional piece of information for caregivers (see Table 20 and Table 21). We chose to use this strategy because we believe that only if this value is significant it will be useful to send the caregivers this false alarm indication. Since we are working with an experimental version of our system, the choice of 75% for the FAP_NOT_MIN was selected arbitrarily. However, it is important to say that the medical staff can configure this value according to their preferences.



## 6.3 Results

We present, in Tables 20 and 21, the results from our experiments. We illustrate a part of the output of our reasoning algorithm showing the first ten notifications related to the temperature and heartrate vital signs, respectively. As one can see, FAP values were attributed to the alarms, and FAP_LABEL were added to notifications by the Reasoner. The first four columns represent, respectively: Notification ID (NID), Ward ID (WID), Patient ID (PID) and Alarm ID (AID).

Table 20 - Results of our experiments for notifications related to alarms of temperature.

| NID | WID | PID | AID | Sensor Type | Sensor Value | Alarm_ timestamp | FAP | Notificatio_ timestamp | FAP_ LABEL |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | Tempe-rature | 35.0 | 2019-07-02 21:51:06.291 | 50.0 | 2019-07-02 21:51:06.334 | UNDE-FINED |
| 2 | 1 | 4 | 2 | Tempe-rature | 42.0 | 2019-07-02 21:51:08.328 | 25.0 | 2019-07-02 21:51:08.328 | UNDE-FINED |
| 3 | 1 | 3 | 4 | Tempe-rature | 41.0 | 2019-07-02 21:51:12.457 | 50.0 | 2019-07-02 21:51:12.457 | UNDE-FINED |
| 4 | 1 | 2 | 9 | Tempe-rature | 41.0 | 2019-07-02 21:51:43.223 | 75.0 | 2019-07-02 21:51:43.223 | 75.0 |
| 5 | 1 | 1 | 12 | Tempe-rature | 42.0 | 2019-07-02 21:52:03.697 | 50.0 | 2019-07-02 21:56:06.334 | UNDE-FINED |
| 5 | 1 | 1 | 15 | Tempe-rature | 42.0 | 2019-07-02 21:52:20.053 | 100.0 | 2019-07-02 21:56:06.334 | 100.0 |
| 5 | 1 | 1 | 16 | Tempe-rature | 41.0 | 2019-07-02 21:52:24.135 | 75.0 | 2019-07-02 21:56:06.334 | 75.0 |
| 5 | 1 | 1 | 17 | Tempe-rature | 35.0 | 2019-07-02 21:52:32.309 | 25.0 | 2019-07-02 21:56:06.334 | UNDE-FINED |
| 5 | 1 | 1 | 18 | Tempe-rature | 42.0 | 2019-07-02 21:52:42.594 | 50.0 | 2019-07-02 21:56:06.334 | UNDE-FINED |
| 5 | 1 | 1 | 20 | Tempe-rature | 41.0 | 2019-07-02 21:52:50.774 | 50.0 | 2019-07-02 21:56:06.334 | UNDE-FINED |

Table 21 - Illustration of the results of our experiments, with the addition of FAP and FAP_LABEL for the first ten notifications related to heartrate vital signs.

| NID | WID | PID | AID | Sensor Type | Sensor Value | Alarm_ timestamp | FAP | Notification_ timestamp | FAP_ LABEL |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | Heart-rate | 108.0 | 2019-07-02 21:51:09.375 | 75.0 | 2019-07-02 21:51:09.39 | 75.0 |
| 2 | 1 | 1 | 2 | Heart-rate | 145.0 | 2019-07-02 21:51:11.432 | 25.0 | 2019-07-02 21:51:11.432 | UNDE-FINED |
| 3 | 1 | 4 | 6 | Heart-rate | 123.0 | 2019-07-02 21:51:21.721 | 50.0 | 2019-07-02 21:51:21.722 | UNDE-FINED |
| 4 | 1 | 3 | 8 | Heart-rate | 116.0 | 2019-07-02 21:51:25.827 | 50.0 | 2019-07-02 21:51:25.827 | UNDE-FINED |
| 5 | 1 | 2 | 3 | Heart-rate | 156.0 | 2019-07-02 21:51:15.539 | 0.0 | 2019-07-02 21:56:09.397 | UNDE-FINED |
| 5 | 1 | 2 | 5 | Heart-rate | 159.0 | 2019-07-02 21:51:19.667 | 50.0 | 2019-07-02 21:56:09.397 | UNDE-FINED |
| 5 | 1 | 2 | 7 | Heart-rate | 44.0 | 2019-07-02 21:51:23.776 | 75.0 | 2019-07-02 21:56:09.397 | 75.0 |
| 5 | 1 | 2 | 9 | Heart-rate | 164.0 | 2019-07-02 21:51:27.874 | 50.0 | 2019-07-02 21:56:09.397 | UNDE-FINED |
| 5 | 1 | 2 | 16 | Heart-rate | 184.0 | 2019-07-02 21:51:44.254 | 25.0 | 2019-07-02 21:56:09.397 | UNDE-FINED |
| 5 | 1 | 2 | 23 | Heart-rate | 51.0 | 2019-07-02 21:52:00.641 | 0.0 | 2019-07-02 21:56:09.397 | UNDE-FINED |

## 6.5
## Discussion

### 6.5.1
### Conclusions and Future Work

In this chapter, we tried to fill the gap of having feasible solutions to mitigate the alarm fatigue problem by focusing on the issue of false positive alarms, known to be a serious problem that yet remains unsolved. We presented a reasoning algorithm to detect false alarms based on alarm-context information provided automatically by the use of sensors and wearable devices and manually by the inputs of caregivers.

We created a database of simulated alarm-context information to establish a basis for the development of our algorithm in order to confirm the hypotheses H3 and H4 in experimental settings. As we can see in Tables 25 and 26 (in the FAP column), every alarm generated by the sensors and monitoring devices in our experiment had a FAP value associated to it by our reasoning algorithm. Our algorithm also added an indication of a false alarm probability (FAP_LABEL) to the notifications sent to caregivers. This information is available in the FAP_LABEL column of our dataset (See Tables 25 and 26).

Regarding the hypothesis H5, which declares that patient safety will not be compromised by the use of the reasoning algorithm about whether to add a FAP_LABEL to a notification, we can assume that H5 is confirmed, since our algorithm does not stop an alarm from being triggered even when the FAP found is considered very high. We can see an example of this information in the sixth row of Table 20, where the alarm (AlarmID=15) still triggered a notification (NotificationID=5) even though it had a calculated FAP of 100%.

As future work, we are planning to evolve our solution to support an optimized version of our reasoning algorithm that calculates the optimal FAP_NOTIF_MIN based on the real-time volume of alarms being triggered in an ICU.

Another plan for future work is to develop a machine learning-based algorithm capable of predicting both FAP and FAP_LABEL based on a dataset that

contains the ICU information history, such as patients' conditions, sensors and alarms.

# 7
# Reasoning about Who to Notify

In this chapter, we discuss about how to choose the best caregiver to notify within a set of caregivers based on real-time circumstances in an ICU, including: (i) alarm-context information, (ii) patients' conditions (e.g., their level of severity), and (iii) caregivers' information such as their experience, along with other circumstances that change dynamically (e.g., how far they are, physically, from the patient that needs help and how many notifications they have already received previously).

This chapter's main goal is to propose a solution for mitigating alarm fatigue by using an automatic reasoning mechanism to choose the best caregiver to be assigned to a given notification. The implementation attributes a value correspondent to the benefit of each possible assignment and decides the best option within all of the benefits by using a greedy strategy.

To pursue our goal, we defined the following Sub-Question:

**SQ4.** How can we reason about who to notify within the caregivers team?

We formalized this problem as a Constraint-Satisfaction Problem (CSP) and we present, in this chapter, one example of how it can be solved. We designed a case study where patients' vital signs were collected through a vital signs' generator that also simulates anomalies that trigger alarms. We conducted five experiments to test our algorithm considering different situations for an ICU. In each experiment, we vary the number of patients, number of caregivers, caregivers' capacity etc. The evaluation of our algorithm was made through the comparison between the results of the choices made by our reasoning algorithm and another strategy that we call "blind" strategy, which randomly assigns caregivers to notifications.

## 7.1
## Problem Definition

In our previous chapter, we mentioned the most common alarm-related issues that may lead to Alarm Fatigue, including the excessive number of alarms, alarms generated by many different types of alarm devices, and the high percentage of false alarms. In addition to these issues, the healthcare teams need to deal with other critical information in ICUs, such as the perceived alarm urgency, and the perceived true alarm rate of the alarm systems.

All of this information has to be processed by the healthcare teams who need to follow their own strategies to properly answer to alarms. Caregivers are consistently under pressure: they should analyze the high volume of inputs they are receiving in order to answer to them quickly and correctly, by making decisions in real-time about the response to the next alarm based on their context information. This scenario may culminate in an overwhelmed and fatigued healthcare team that is desensitized and slow to respond to alarms. Under alarm fatigue conditions, the staff may ignore and/or silence alarms, putting patients in risky situations (Keller, 2012).

To assist caregivers in their daily routine activities of responding to alarms, we provide a reasoning mechanism that decides who is the best caregiver to notify based on the analysis of alarm-context information, patients' and caregivers' conditions. The idea is to assign a member of a caregivers' team as the receiver of a given alarm, by choosing the member of the team that is the most capable of attending to the alarm, given the circumstances in an ICU.

## 7.2
## Constraint-Satisfaction Problem

Constraint-Satisfaction is a powerful framework used for expressing and solving search problems. The idea is to find a consistent assignment of values to a predefined set of variables. The variables typically have pre-enumerated domains of discrete values and a set of constraints over subsets of these variables that limits their possible values (Mittal & Falkenhainer, 1990).

A number of problems in Artificial Intelligence and other areas of computer science can be viewed as special cases of the Constraint-Satisfaction Problem

(Kumar, 1992). In this work, we are focusing on the type of CSP that can be stated by a given set of variables, a finite and discrete domain for each variable, and a set of constraints, along with soft constraints or preferences. Our goal is to find one assignment to the variables such that the assignment satisfies all the constraints.

A CSP is typically specified by a set of variables $V = \{v_1, \ldots, v_n\}$ and a set of constraints on subsets of V limiting the values that may be assigned in a consistent manner. Each variable $v_i$ has an associated domain $D_i = \{d_{i1},\ldots,d_{ini}\}$ which specifies its set of possible values. The constraint satisfaction task is to find assignments of values for $\{v_1,\ldots,v_n\}$ that simultaneously satisfy all the constraints (Mittal & Falkenhainer, 1990). Considering our problem, the set of variables V would be represented by the set of receivers of the notifications, while the domain D, in turn, would consist of the set of possible receivers, i.e., caregivers to whom a notification can be assigned to.

## 7.3 Modeling our Problem

For this chapter, we describe the formalization for our research problem as a constraint-satisfaction problem. We present below our modeling process based on the notation we defined in this work.

As we mentioned above in Equation (9), we define a notification in terms of the alarm that triggers it and the timestamp in which the notification is sent. Let $N = \{n_1, n_2,...,n_n\}$ be the set of notifications that should be sent to a member of a caregivers' team notifying him/her about anomalies that happened to a given set of patients $P = \{p_1, p_2,\ldots, p_n\}$ in an ICU during a period of one day. We also have a set of caregivers $C = \{c_1, c_2,\ldots,c_n\}$ to whom each alert should be assigned. Examples of information that can be used to decide the best caregiver to choose from the available ones at the time the alert is triggered are: the probability for an alarm to be true, the severity of a patient's condition, and the caregiver's ability to deal with an alarm based on their experience, among other factors.

Our problem is to create an assignment e that attributes a given notification n to a given caregiver c. In this case, an assignment e can be defined as e = (n,c), where $(n,c) \in E$, and E is the set of all the assignments that satisfies all the constraints defined for this problem (13).

$$e = (n,c) \qquad (13)$$

Where:

- $E = \{(ni,ci), \dots , (nk, ck)\}$, with $ni, nk \in N$ (the set of Notifications) and $cj, ck \in C$ (the set of caregivers);

Our goal is to find out who is the best receiver for each alarm given the circumstances of alarms, patients and caregivers, without putting patients at risk and caregivers in alarm fatigue conditions. Before showing the solution we arrived at, we show an illustration of a possible solution for a generic assignment problem, i.e., a set of assignments E that we are assuming it satisfies all its constraints.

Suppose we have a set of ten notifications $N = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}\}$ and a set of six caregivers $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. Consider that one of them - $c_5$ -, is not available to receive notifications. We show below how a possible assignment can be done and the final set of assignments $E_1$.

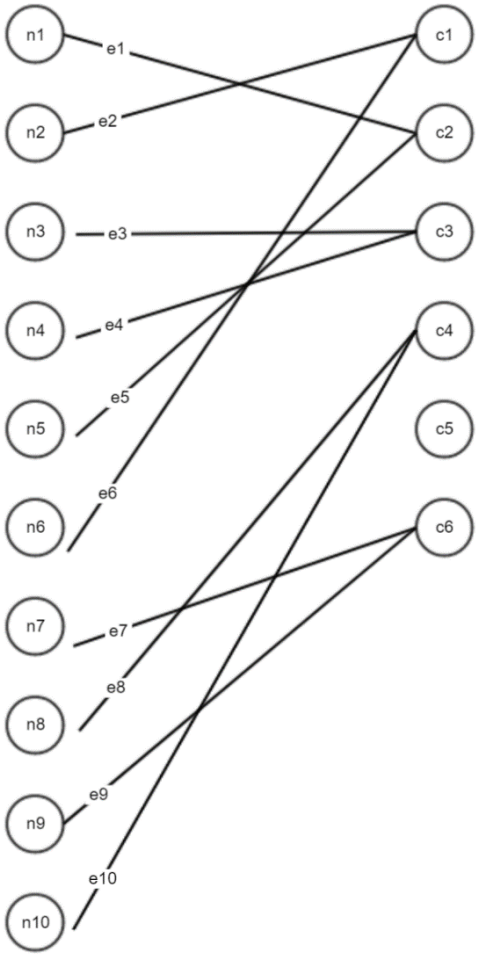



Figure 33 - Example of a possible assignment $E_1$.

The final set for $E_1$ is: $E_1 = \{(n_1,c_2), (n_2,c_1), (n_3,c_3), (n_4,c_3), (n_5,c_2), (n_6,c_1), (n_7,c_6), (n_8,c_4), (n_9,c_6), (n_{10},c_4)\}$.

## 7.4
## Explaining how we choose who to notify

Let's now consider a more general case of our problem, where a given notification $n_i$ in a set of notifications $N = \{n_1,n_2,\ldots,n_e\}$ should be assigned to a given caregiver $c_i$ within the set of caregivers $C = \{c_1, c_2, \ldots, c_n\}$.

Our strategy for deciding the best assignment $e_i = (n_i,c_i)$ for each notification $n_i$ is to calculate the benefit function *benefit(e)* for each possible assignment, where the possible assignments are those that remain, after excluding the ones that do not respect our constraints, as listed below (Table 22):

Table 22 - List of our constraints and soft constraints (preferences).

| Constraint | Type | Description |
|---|---|---|
| C1 | Unary | Caregivers should only receive notifications according to his/her capacity CAPACITY(c). CAPACITY(c) gives us the maximum number of notifications a caregiver c can receive |
| C2 | Unary | A caregiver can only receive a notification $n_i$ if he/she is available at the instant the notification occurs |
| C3/SC3 | Unary | A notification should be preferably assigned to the caregiver that has more experience (measured in years) |
| C4/SC4 | Unary | A notification should be preferably assigned to the caregiver that is physically closer to the patient |
| C5/SC5 | Unary | A notification should be preferably assigned to caregivers that have received less notifications |

Let $b_e$ be the benefit of the assignment e, i.e., $b_e$ is the benefit of sending a notification $n_i$ to a caregiver $c_i$. Our goal is to calculate the benefit for all the possible assignments for a notification $n_i$ (by doing this calculus immediately after $n_i$ has occured) in order to choose the best caregiver to receive $n_i$. With this strategy, we aim at maximizing the following sum of benefits for a set of notifications assigned to caregivers during a period of one day:

$$\sum_{e \in E} b_e X_E, \quad X_E = \begin{cases} 1, & \text{if notification e is assigned} \\ 0, & \text{otherwise} \end{cases} \qquad (14)$$

The benefit function Benefit(e) is calculated as follows:

$$\textbf{Benefit (e)} = 1/\text{Probability}(a,\text{false}) \times \text{Severity}(p) \times \text{Ability}(c) \times 1/\text{Distance}(c,p) \times 1/\text{NumberOfNotificationsReceived}(c) \qquad (15)$$

Where:

- Probability(a, false): The probability of an alarm a to be “false”;
- Severity(p): The severity of a patient p;
- Experience(c): The ability of a caregiver c to deal with a notification n based on his/her experience;
- Distance(c,p): The distance between a caregiver c and a patient p;
- NumberOfNotificationsReceived(c): The number of notifications received by a caregiver c.

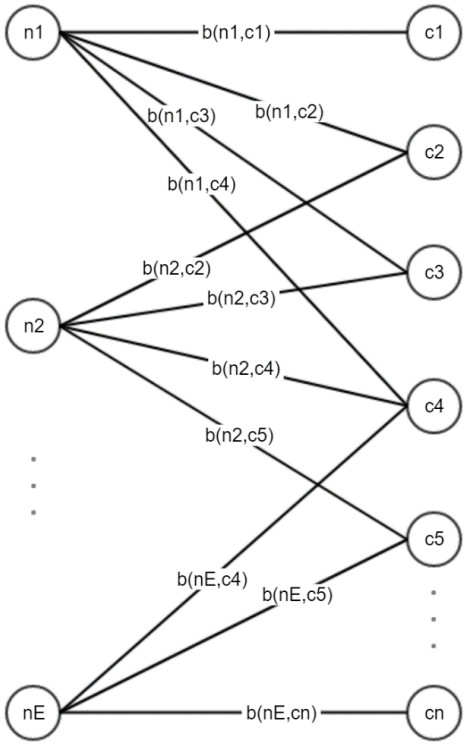



Figure 34 - Illustration of possible assignments, where $b(n_1,c_1)$ is the benefit for assigning the notification $n_1$ to caregiver $c_1$.

The sum of all notifications assigned $X_n$ to a given caregiver should be lower than, or equal to, his/her capacity to attend to. Equation 16 shows the restriction of the number of notifications that can be assigned to a caregiver c.

$$\forall \text{ caregiver } c \in C, \qquad (16)$$
$$\sum_{nT} X_{nc} <= \text{CAPACITY}(c)$$

Where:

- $X_n$ specifies whether a notification n was assigned to a caregiver c. The above sum specifies that the total of notifications assigned to a given caregiver must be lower than, or equal to, its capacity (CAPACITY(c));
- nT is the total number of notifications assigned to c;

Every notification n should be assigned to one caregiver. Equation 17 shows the restriction of the number of caregivers that can receive a given notification n.

$$\forall \text{ notification } n \in N, \qquad (17)$$
$$\sum_{(n,c) \in E} X_{nc} = 1$$

## 7.5 Methods

Our proposed solution represents a new approach for mitigating the alarm fatigue issue that focuses on a reasoning algorithm that is used to decide the best caregiver to whom a notification should be assigned.

We describe next the experiment we developed as an example of solution for this problem. The results are presented in the Discussion session.

### 7.5.1 Goals

Our main goal with this case study is to decide which caregiver is the most capable of attending to a given notification based on alarm-context information and information about patients' and caregivers' conditions.

Our specific goal is to maximize an objective function that gives us the sum of the benefits of the notification assignments that occurred during one day.

### 7.5.2 Hypotheses

We defined the following hypotheses for our case study:

> **H6.** A notification assignment should prioritize the caregiver that is the most capable of attending to it within the group of caregivers available at the time the notification occurs, considering the probability of a notification to be false, patient's severity, caregivers' experience, the distance between caregivers and patients, and the number of notifications caregivers have received.
>
> **H7.** The assignment of notifications to caregivers should be limited to his/her capacity of receiving notifications.

### 7.5.3 Applications Scenario

As we mentioned above, in our case study, patients' vital signs were collected by using a vital signs' generator that also simulates anomalies that trigger alarms. We conducted five experiments to test our reasoning algorithm considering

different situations for an ICU. In each experiment, we changed the configuration of our simulated environment (ICU) in terms of number of patients, number of caregivers, and number of sensor readings to evaluate the response of our algorithm to different circumstances (Table 23).

Table 23 - Configuration of the experiments comprising number of patients, caregivers and sensor's readings in each experiment.

| Experiment | Number of Patients | Number of Caregivers | Number of sensor readings |
|---|---|---|---|
| 1 | 2 | 7 | 1200 |
| 2 | 4 | 7 | 2400 |
| 3 | 8 | 9 | 4800 |
| 4 | 16 | 14 | 9600 |
| 5 | 32 | 30 | 19200 |

Other additional information, such as patients' severity, caregivers' availability, experience and capacity, was also pre-configured in each experiment. Tables 24 and 25 present the initial configuration for caregivers and patients defined in experiment 2, whose results we will discuss later on.

Table 24 - Initial configuration for the set of caregivers in experiment 2.

| Caregiver ID | Experience level | Is_Available | Ward | Capacity |
|---|---|---|---|---|
| 1 | 3 | true | 1 | 260 |
| 2 | 1 | true | 8 | 260 |
| 3 | 2 | true | 5 | 260 |
| 4 | 4 | true | 7 | 260 |
| 5 | 5 | true | 6 | 260 |
| 6 | 4 | true | 2 | 260 |
| 7 | 2 | true | 3 | 260 |
| 8 | 2 | true | 4 | 260 |
| 9 | 3 | True | 9 | 260 |

Table 25 - Initial configuration for the set of patients in experiment 2.

| Patient ID | Severity | Ward |
|---|---|---|
| 1 | 2 | 2 |
| 2 | 4 | 3 |
| 3 | 3 | 4 |
| 4 | 5 | 5 |
| 5 | 1 | 6 |
| 6 | 2 | 7 |
| 7 | 3 | 8 |
| 8 | 4 | 9 |

## 7.6 Results

In the next sections, we describe the results of the experiments conducted to evaluate our application - developed in the Java language using the RabbitMQ broker - to test our algorithm.

### 7.6.1 Inputs for our reasoning algorithm about who to notify

We defined the following inputs for our reasoning algorithm (Table 26).

Table 26 - Inputs for our reasoning algorithm.

| Input | Input name | Description |
|---|---|---|
| 1 | Patients | The set of patients (P) |
| 2 | Notifications | The set of notifications (N) |
| 3 | Caregivers | The set of caregivers (C) |
| 4 | Severity | Severity of a given patient p (Severity(p)) |
| 5 | Probability | Probability of a notification n to be false (Probability(n,false)) |
| 6 | Experience | Caregivers' experience (Experience (c)) |
| 7 | Distance | Distance between patient and caregiver (Distance(c,p)) |
| 8 | Capacity | Caregivers' capacity (CAPACITY(c)) |
| 9 | NotificationsReceived | The number of notifications received by a caregiver NumberOfNotificationsReceived(c) |

### 7.6.2 Output of our reasoning algorithm

There is one output of our algorithm (Table 27).

Table 27 - Output of our reasoning algorithm.

| Output | Output name | Description |
|---|---|---|
| 1 | Receiver | The identity of a caregiver to whom a notification should be assigned |

### 7.6.4 Evaluation

The evaluation of our algorithm was made through the comparison between the results of the sum of benefits of the assignments chosen by our reasoning algorithm and the sum of benefits for the assignments made by another strategy we called "blind" strategy, which randomly assigns caregivers to notifications. As we explained previously, every time an alert is triggered, our reasoning algorithm calculates the benefit for all the possible assignments to the available caregivers and chooses the one that has the higher value (as a greedy strategy). The benefit is

calculated by computing the Benefit (e) function shown in Equation 15. On the other hand, the blind version randomly assigns a caregiver to a notification independently of the circumstances. After the blind algorithm chooses an assignment, we calculate the assignment's benefit in order to have the sum of all benefits for this approach so we can use as a parameter of comparison with our reasoning algorithm.

In this case, for each experiment, we calculated the sum of benefits for both versions of the algorithms: the blind version and the reasoning version we are providing as a solution for the alarm fatigue issue. We considered each sensor separately to calculate the sum. The results are presented in Table 28.



Table 28 - The final sum of benefits for all assignments made by the reasoning and the blind algorithms in our five experiments.

| Experiment | Sensor | Algorithm | Number of alerts | Sum of benefits |
|---|---|---|---|---|
| 1 | Temperature | Blind | 297 | 387.50 |
| | Temperature | Reasoning | 305 | 663.08 |
| | Heartrate | Blind | 449 | 585.77 |
| | Heartrate | Reasoning | 446 | 855.38 |
| 2 | Temperature | Blind | 639 | 581.09 |
| | Temperature | Reasoning | 868 | 674.16 |
| | Heartrate | Blind | 631 | 1302.47 |
| | Heartrate | Reasoning | 881 | 1568.70 |
| 3 | Temperature | Blind | 1289 | 707.13 |
| | Temperature | Reasoning | 1722 | 787.22 |
| | Heartrate | Blind | 1307 | 1577.44 |
| | Heartrate | Reasoning | 1719 | 1804.46 |
| 4 | Temperature | Blind | 2471 | 1479.34 |
| | Temperature | Reasoning | 3417 | 1740.36 |
| | Heartrate | Blind | 2431 | 2772.10 |
| | Heartrate | Reasoning | 3473 | 3207.85 |
| 5 | Temperature | Blind | 5005 | 1530.16 |
| | Temperature | Reasoning | 6873 | 1844.85 |
| | Heartrate | Blind | 4896 | 3900.05 |
| | Heartrate | Reasoning | 6823 | 4638.95 |

We present below a comparison of the sum of benefits for the two approaches evaluated in this case study. Table 29 shows the percentage of the improvement we achieved for the sum of benefits we aimed at maximizing. The comparison column represents how much higher the sum of benefits achieved by our reasoning algorithm is when compared to the blind version.

Table 29 - Percentage of the improvement of the sum of benefits we achieved by using our reasoning algorithm compared to the results for the blind algorithm.

| Experiment | Sensor | Algorithm | Comparison (in %) |
|---|---|---|---|
| 1 | Temperature | Reasoning | 71.12 |
| | Heartrate | Reasoning | 46.03 |
| 2 | Temperature | Reasoning | 124.14 |
| | Heartrate | Reasoning | 132.69 |
| 3 | Temperature | Reasoning | 123.08 |
| | Heartrate | Reasoning | 129.22 |
| 4 | Temperature | Reasoning | 87.39 |
| | Heartrate | Reasoning | 84.32 |
| 5 | Temperature | Reasoning | 154.88 |
| | Heartrate | Reasoning | 151.45 |

Regarding the total of notifications received by each caregiver, we present below the results for the reasoning algorithm and for the blind algorithm in Tables 30 and 31, respectively.

Table 30 - Notifications received by each caregiver for the reasoning algorithm in Experiment 2.

| Caregiver ID | Caregiver's experience | Total of notifications received |
|---|---|---|
| 5 | 5 | 260 |
| 4 | 4 | 260 |
| 6 | 4 | 260 |
| 1 | 3 | 236 |
| 9 | 3 | 183 |
| 3 | 2 | 160 |
| 7 | 2 | 160 |
| 8 | 2 | 110 |
| 2 | 1 | 90 |

Table 31 - Number of notifications of heartrate received by caregivers in Experiment 2 for the blind algorithm.

| Caregiver ID | Caregiver's Experience | Total of notifications received |
|---|---|---|
| 3 | 2 | 206 |
| 4 | 4 | 204 |
| 2 | 1 | 199 |
| 8 | 2 | 193 |
| 9 | 3 | 189 |
| 7 | 2 | 189 |
| 5 | 5 | 188 |
| 6 | 4 | 184 |
| 1 | 3 | 170 |

## 7.7 Discussion

### 7.7.1 Conclusions and Future Work

Regarding the hypothesis H6 that says that an assignment of a notification should prioritize the caregiver that is available and is the most experienced and capable of attending to it, we can assume it holds for our reasoning algorithm. This result can be seen in Table 15 above. In our experiments, the levels of experience range from 1 to 5, where 5 is the highest level and 1 is the lowest. As one can see, caregivers with higher levels of experience received more notifications than the ones with lower levels. The caregiver with the highest experience (level 5) received 260 notifications, while the one with the lowest experience (level 1) received 90 notifications. It is important to point out that other information (i.e., inputs 4-10 shown in Table 26) also had impact on this result.

In turn, the results for the blind algorithm in the same Experiment 2 show that the caregiver that received more notifications (a total of 206) was one with a low level of experience (level 2), while the one with the highest level of experience was only the 7th caregiver to receive more notifications within a group of 9.

This result means that, compared to the random strategy for assignments, our reasoning algorithm achieved a better result in terms of prioritizing the assignments we wanted to make based on our defined criteria: patient's severity, the distance between caregivers and patients, caregivers' experience, the probability

of a notification to be false, and the number of notifications caregivers have received. It is important to point out that our reasoning algorithm had taken into account all of these criteria to calculate the best benefit for an assignment, even though we chose the caregivers' experience parameter to illustrate that we achieved our goal in our prioritization task.

The hypothesis H7 analyzed in this case study (that says that the assignment of notifications to caregivers should be limited to his/her capacity of receiving notifications) was also confirmed, since for our reasoning algorithm we limited the number of notifications a caregiver can receive (Capacity(c)) in each experiment in order to respect this constraint. In Experiment 2, the maximum number of notifications of heartrate that a caregiver could receive was 260. As we can see in Table 30, there is no caregiver with more than 260 notifications received. In fact, after caregivers 5, 4, and 6 (i.e., CaregiverID=5, CaregiverID=4, and CaregiverID=6) had reached their capacity, the algorithm made them non-available to receive other notifications, and this constraint was respected by the system in all our conducted experiments.

As future work, we are planning to evolve our reasoning algorithm to deal with resource negotiation, in the sense that, if we reach a situation where all caregivers are non-available to receive more notifications (i.e., if all of them reach their capacity), we can start a negotiation for new resources. The idea is to ask for members of other caregivers teams to be allocated in teams that are over their capacity to respond by doing all of the negotiation process automatically through the system.

Another future plan is to evaluate the distribution of the notifications to the caregivers' teams made by the algorithms. We expect that our solution may offer a better distribution of notifications compared to the blind version. Therefore, we aim to analyze the results of our experiments in order to confirm our expectations.

# 8
# Final Remarks and Future Work

Alarm safety is a complex problem to solve, influenced by a number of factors that extrapolate technology challenges and limitations, such as human influences, difficult patient conditions, a wide variety of environmental conditions, and even staffing cultures (Keller, 2012). Alarm hazards are still a big challenge for members of the healthcare teams in ICUs. As practice settings continue to become more technology driven, effective interventions for alarm hazards in ICU settings are crucial. Feasible strategies should be provided in order to allow nurses to respond to the call to ensure patient safety in an increasingly complex care environment (Tanner, 2013).

As healthcare units become more dependent upon monitoring devices for patient care purposes, the alarm fatigue issue has to be addressed as a major concern for the healthcare team, as well as to enhance patient safety. Nonetheless, although alarm safety is a critical issue that needs to be addressed to improve patient care, hospitals have not given serious consideration to the manner in which their staff should respond to clinical alarms.

The lack of use of any intelligent filter to detect recurrent, irrelevant and/or false alarms before alerting healthcare providers can culminate in a complex and overwhelming scenario of a sensory overload for the medical team.

This thesis proposes a new approach to cope with the alarm fatigue problem. The solution we provide to manage this undesirable issue uses an automatic reasoning mechanism to decide how to notify caregivers about anomalies detected by a patient monitoring system where a large number of alarms might lead to alarm fatigue.

## 8.1
## Main Contributions

The main contributions described in this thesis are:

1. An architecture for health systems that support patient monitoring, notification, and reasoning capabilities;
2. A model to support reasoning algorithms that decide how to notify caregivers to avoid alarm fatigue without compromising patient safety;
3. A reasoning algorithm that specifies how to notify caregivers by deciding whether to aggregate a group of alarms;
4. A reasoning algorithm that specifies how to detect false alarms and notify caregivers with an indication of false alarm probability;
5. A reasoning algorithm to decide who is the best caregiver to notify within the set of available caregivers in an ICU.

Regarding the third contribution, we demonstrated through our experiments that providing a reasoning system can reduce the notifications received by the caregivers by up to 99.3% of the total alarms generated. These experimental results strongly suggest that this reasoning algorithm is a useful strategy to avoid alarm fatigue.

In the fourth contribution, we showed that the Reasoner entity was able to calculate FAP values to alarms based on false alarm indicators in order to reason about whether to notify caregivers with a FAP label indication without putting patients in risky situations. Experiments were conducted to demonstrate that we could reason about how to detect false alarms by analyzing alarm-context information.

In our fifth contribution, we showed that our reasoning algorithm achieved better results in terms of assignments prioritization when compared to the blind strategy of notification assignments. We defined the following prioritization criteria: caregivers' experience, probability of a false notification, patient's severity, distance between caregivers and patients, and number of notifications caregivers have already received. We demonstrated that our algorithm prioritized the caregivers that were the most capable of attending to the notification within the group of available ones. The results of our experiments showed that, in our simulated environment, caregivers with higher levels of experience received more notifications than the ones with lower levels of experience.

## 8.2
## Future Work

As a future plan, we aim at evolving our solution provided to reason about how to detect false alarms based on alarm-context information. We aim to support an optimized version of our reasoning algorithm that calculates the optimal FAP_NOTIF_MIN based on the real-time volume of alarms triggered in an ICU. Another plan is to develop a machine learning-based algorithm capable of predicting both FAP and FAP_LABEL based on a dataset that contains the ICU information history, such as context information about patients' conditions, sensors, and alarms. The idea is to provide a reliable classification system in which caregivers may trust so the FAP label added to the notification can help them prioritize their work, especially when they are under alarm fatigue conditions.

Another future work is to evolve our reasoning algorithm about who to notify to deal with resource negotiation, in the sense that, if we reach a situation in which all caregivers are unavailable to receive more notifications (i.e., if all of them reach their capacity), we could start a negotiation for new human resources. The idea is to ask that members of other caregivers teams be allocated to teams that are over their capacity to respond to alarms, where all of the negotiation process would be done automatically through the system.

Our last future plan is to evaluate the distribution of the notifications to the caregivers teams made by the reasoning algorithm that decides who to notify. We expect that our solution may offer a better distribution of notifications compared to the blind version. Therefore, we aim to analyze the results of our experiments in order to confirm our expectations.

Note that our system is experimental and does not consider security, something that needs to be taken very seriously in an operational healthcare alarm system.

# 9
# References

Abhishek, B., Poojary, A. G., Rao, M. V. A., & Narayanan, S. (2013). Low Power Portable EEG for Continuous Monitoring with Active Electrodes. *Proceedings of the 2013 Texas Instruments India Educators' Conference*, 332–339. https://doi.org/10.1109/TIIEC.2013.66

*Arduino UNO*. (n.d.). Retrieved from https://www.arduino.cc/en/Guide/ArduinoUno

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805. https://doi.org/https://doi.org/10.1016/j.comnet.2010.05.010

*Azure Machine Learning Studio*. (n.d.). Retrieved from https://studio.azureml.net/

*CM100 Efficia Philips Monitor*. (n.d.). Retrieved from https://www.philips.com.br/healthcare/solutions/value-products/efficia

Craig, A., Tran, Y., Wijesuriya, N., & Nguyen, H. (2012). Regional brain wave activity changes associated with fatigue. *Psychophysiology*, *49*(4), 574–582. https://doi.org/10.1111/j.1469-8986.2011.01329.x

Cvach, M. (2012). Monitor Alarm Fatigue: An Integrative Review. *Biomedical Instrumentation & Technology*, *46*(4), 268–277. https://doi.org/10.2345/0899-8205-46.4.268

Drew, B. J., Harris, P., Zègre-Hemsey, J. K., Mammone, T., Schindler, D., Salas-Boni, R., … Hu, X. (2014). Insights into the problem of alarm fatigue with

physiologic monitor devices: a comprehensive observational study of consecutive intensive care unit patients. *PloS One*, *9*(10), e110274. https://doi.org/10.1371/journal.pone.0110274

ECRI Institute. (n.d.). *ECRI Institute*. Retrieved from https://www.ecri.org

*E-Health Sensor Platform*. (n.d.). Retrieved from https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical

Fernandes, C. O., & de Lucena, C. J. P. (2015). An Internet of Things Application with an Accessible Interface for Remote Monitoring Patients. In A. Marcus (Ed.), *Design, User Experience, and Usability: Interactive Experience Design* (pp. 651–661). Cham: Springer International Publishing.

Fernandes, C. O., de Lucena, C. J. P., de Lucena, C. A. P., & de Azevedo, B. A. (2016). Enabling a Smart and Distributed Communication Infrastructure in Healthcare. In Y.-W. Chen, C. Torro, S. Tanaka, R. J. Howlett, & L. C. Jain (Eds.), *Innovation in Medicine and Healthcare 2015* (pp. 435–446). Cham: Springer International Publishing.

Fernandes, C. O., & Lucena, C. J. P. D. (2017). A Software Framework for Remote Patient Monitoring by Using Multi-Agent Systems Support. *JMIR Med Inform*, *5*(1), e9. https://doi.org/10.2196/medinform.6693

Fernandes, C. O., Lucena, C. J. P. de, & Silva, D. de S. e. (2017). Smart depth of anesthesia monitoring with EEG sensors and agent-based technology. *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*

*(SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 1–8. https://doi.org/10.1109/UIC-ATC.2017.8397455

Fernandes, C. O., Moreira, F. B., Barbosa, S. D. J., & de Lucena, C. J. P. (2017). What Your EEG Wearable Sensors Can Tell About You? *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, 2:1–2:10. https://doi.org/10.1145/3109761.3158389

Fernandes, Chrystinne, & Lucena, C.J.P. de. (2019). *Reasoning about How to Detect False Alarms by Analyzing Alarm-context Information*. (Submitted to publication on the Journal of Medical Internet Research - ISSN 1438-8871)

Fernandes, Chrystinne, Miles, Simon, Cowan, Donald, & Lucena, C.J.P. de. (2019). *Artificial Intelligence Technologies for Coping with Alarm Fatigue in Hospital Environments Because of Sensory Overload*. (JMIR - Journal of Medical Internet Research - ISSN 1438-8871)

Goldman, A. M., Glasscock, E., Yoo, J., Chen, T. T., Klassen, T. L., & Noebels, J. L. (2009). Arrhythmia in heart and brain: KCNQ1 mutations link epilepsy and sudden unexplained death. *Science Translational Medicine*, *1*(2), 2ra6. https://doi.org/10.1126/scitranslmed.3000289

Imhoff, M., Kuhls, S., Gather, U., & Fried, R. (2009). Smart alarms from medical devices in the OR and ICU. *Best Practice & Research. Clinical Anaesthesiology*, *23*(1), 39—50. https://doi.org/10.1016/j.bpa.2008.07.008

Jones, K. (2014). Alarm fatigue a top patient safety hazard. *CMAJ : Canadian Medical Association Journal = Journal de l'Association Medicale Canadienne*, *186*(3), 178. https://doi.org/10.1503/cmaj.109-4696

Keller, J. P. (2012). Clinical alarm hazards: a “top ten” health technology safety concern. *Journal of Electrocardiology*, *45*(6), 588–591. https://doi.org/https://doi.org/10.1016/j.jelectrocard.2012.08.050

Kerr, J. H., & Hayes, B. (1983). An “Alarming” situation in the intensive therapy unit. *Intensive Care Medicine*, *9*(3), 103–104. https://doi.org/10.1007/BF01772574

Kumar, V. (1992). Algorithms for Constraint-Satisfaction Problems: A Survey. *AI Magazine*, *13*, 32–44.

LAWLESS, S. T. (1994). Crying wolf: False alarms in a pediatric intensive care unit. *Critical Care Medicine*, *22*(6), 981–985.

Lehmann, A., Thaler, E., & Boldt, J. (2001). [Is measuring the depth of anesthesia sensible? An overview on the currently available monitoring systems]. *Anasthesiologie, Intensivmedizin, Notfallmedizin, Schmerztherapie : AINS*, *36*(11), 683—692. https://doi.org/10.1055/s-2001-18048

*LES PUC-Rio*. (n.d.). Retrieved from http://www.les.inf.puc-rio.br/

Liu, D., Görges, M., & Jenkins, S. A. (2012). University of Queensland Vital Signs Dataset: Development of an Accessible Repository of Anesthesia Patient Monitoring Data for Research. *Anesthesia & Analgesia*, *114*(3). Retrieved from https://journals.lww.com/anesthesia-analgesia/Fulltext/2012/03000/University_of_Queensland_Vital_Signs_Dataset__.15.aspx

Luck, M. (2003). *Agent Technology - Enabling Next Generation Computing: A Roadmap for Agent Based Computing*. Retrieved from https://www.xarg.org/ref/a/0854327886/

Markiewicz, M. E., & de Lucena, C. J. P. (2001). Object Oriented Framework Development. *XRDS*, *7*(4), 3–9. https://doi.org/10.1145/372765.372771

Mittal, S., & Falkenhainer, B. (1990). Dynamic Constraint Satisfaction Problems. *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1*, 25–32. Retrieved from http://dl.acm.org/citation.cfm?id=1865499.1865503

Myles, B., Cooper Swanson, T., Holverstott, J., & Duncan, M. (2007). *Autism spectrum disorders : a handbook for parents and professionals*. Westport, Conn: Praeger.

Nunes, R. R., Fonseca, N. M., Simões, C. M., Rosa, D. M., Silva, E. D., Cavalcante, S. L., … Stefani, L. C. (2015). Consenso brasileiro sobre monitoração da profundidade anestésica. *Brazilian Journal of Anesthesiology*, *65*(6), 427–436. https://doi.org/https://doi.org/10.1016/j.bjan.2015.10.001

Pal, M., & Mather, P. M. (2005). Support vector machines for classification in remote sensing. *International Journal of Remote Sensing*, *26*(5), 1007–1011. https://doi.org/10.1080/01431160512331314083

Purdon, P. L., Pierce, E. T., Mukamel, E. A., Prerau, M. J., Walsh, J. L., Wong, K. F. K., … Brown, E. N. (2013). Electroencephalogram signatures of loss and recovery of consciousness from propofol. *Proceedings of the National Academy of Sciences*, *110*(12), E1142–E1151. https://doi.org/10.1073/pnas.1221180110

*RabbitMQ*. (n.d.). Retrieved from https://www.rabbitmq.com/

Rampil, M., Ira J. ..MS. (1998). A Primer for EEG Signal Processing in Anesthesia. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, *89*(4), 980–1002.

*Raspberry Pi*. (n.d.). Retrieved from https://www.raspberrypi.org/

*RStudio*. (n.d.). Retrieved from http://www.rstudio.com

Sendelbach, S. E., & Funk, M. (2013). Alarm fatigue: a patient safety concern. *AACN Advanced Critical Care, 24 4*, 378-86-8.

Shanmugham, M., Strawderman, L., Babski-Reeves, K., & Bian, L. (2018). Alarm-Related Workload in Default and Modified Alarm Settings and the Relationship Between Alarm Workload, Alarm Response Rate, and Care Provider Experience: Quantification and Comparison Study. *JMIR Hum Factors*, *5*(4), e11704. https://doi.org/10.2196/11704

Sowan, A. K., Gomez, T. M., Tarriela, A. F., Reed, C. C., & Paper, B. M. (2016). Changes in Default Alarm Settings and Standard In-Service are Insufficient to Improve Alarm Fatigue in an Intensive Care Unit: A Pilot Project. *JMIR Human Factors*, *3*(1), e1. https://doi.org/10.2196/humanfactors.5098

*Sparkfun*. (n.d.). Retrieved from https://www.sparkfun.com/products/12577

Tanner, T. (2013). The Problem of Alarm Fatigue. *Nursing for Women's Health*, *17*(2), 153–157. https://doi.org/https://doi.org/10.1111/1751-486X.12025

The Joint Commission. (2004). *Preventing, and managing the impact of, anesthesia awareness*. *32*, 1–3.

*ThinkGear Serial Stream Guide*. (n.d.). Retrieved from http://developer.neurosky.com/docs/doku.php?id=thinkgear_communications_protocol

Wickham, H. (2009). *Ggplot2: Elegant Graphics for Data Analysis* (2nd ed.). Springer Publishing Company, Incorporated.

Winters, B. D., Cvach, M. M., Bonafide, C. P., Hu, X., Konkani, A., O'Connor, M. F., … The Society for Critical Care Medicine Alarm and Alert Fatigue Task

Force. (2018). Technological Distractions (Part 2): A Summary of Approaches to Manage Clinical Alarms With Intent to Reduce Alarm Fatigue. *Critical Care Medicine*, *46*(1). Retrieved from https://journals.lww.com/ccmjournal/Fulltext/2018/01000/Technological_Distractions__Part_2___A_Summary_of.17.aspx

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). Wiley Publishing.

Zeileis, Achim;, & Grothendieck, Gabor. (n.d.). *zoo: S3 Infrastructure for Regular and Irregular Time Series*. Retrieved from https://www.jstatsoft.org/article/view/v014i06