Requirements Document

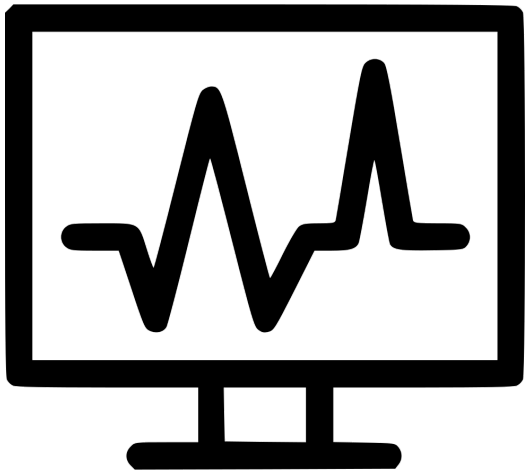
12/11/2017

Dr. Fatemeh Afghah

Version 2.0

James Todd, Nathan Payton-McCauslin and

Alexander Grzesiak



Data Menders

Accepted as baseline requirements for the project:

For the client: _____ For the team: _____

Table of Contents

1.0 Introduction

This document serves to outline specific requirements that we must meet throughout our project to deliver a complete package as well as introduce the reader to the necessity and importance of the project and our vision for a solution going forward.

Data Menders is a capstone team aimed to help facilitate the reduction of false alarms in intensive care units (ICUs) through the analysis and interpretation of signals to determine the true nature of an alarm. In a typical ICU, nurses are constantly scrambling to tend to fires that may not exist. Many of these non existent fires are false or meaningless alarms. A false alarm is an alarm that is triggered to medical staff about a patient that is interpreted as a danger when it is really benign. These false alarms can create "alarm fatigue" in medical staff since they are forced to respond to numerous alarms throughout the day. In other words, false alarms create a "cry wolf effect" with medical staff, this is to say that nurses become desensitized to alarms after responding to numerous alarms and thus they stop responding in the correct way. False alarms and the resulting alarm fatigue can result in interruption of patient care, depressed immune systems through sleep deprivation and even death. In fact, the Emergency Care Research Institute placed false alarms at number one on their list of the Top 10 Health Technology Hazards for the years 2012, 2013, and 2015. To put this quantitatively, from 2005 to 2008, the FDA database received 566 reports of patient deaths related to alarms of monitoring devices. The idea for the project comes through real life experiences pertaining to our client and sponsor and an obviously glaring weakness in the healthcare field.

Our client and sponsor, Dr. Fatemeh Afghah, is an assistant professor in the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. In addition to teaching, her research areas include: wireless communications, game theoretical optimization and biomedical signal processing. Her current research focuses on developing predictive modeling techniques using game theory and graph theory to optimize the performance of current medical diagnosis methods. Her expertise with biomedical signal processing and medical diagnosis methods are invaluable to our project's success.

2.0 Problem Statement

Automated monitoring has revolutionized care in modern ICU units around the world because they allow continuous monitoring of patient vital signs such as: blood pressure, pulse and ECG signals. These automated monitoring devices watch these vital signs and alert medical staff if there is an anomaly by way of a medical alarm. This is great news for medical staff as they can become more productive by being able to work on other tasks, however, for all the good automated monitoring does for ICU units, it's also prone to false alarms. False alarms here occur from any number of things: sensors become loose, muscles have spasms, monitors are faulty, incorrect placement of sensors by staff, etc.

As you can imagine, false alarms have been addressed by medical professionals and engineers, and attempts have been made to reduce them. However, the majority of attempts so far have been focused on improving hardware such as improving monitors or increasing the accuracy of sensors. The problem with this approach is that, as mentioned earlier, people are imperfect. Our project focuses on the much more realistic idea that regardless of the sophistication of the hardware, patients will still move around and staff won't always hook everything up correctly and thus false alarms will be triggered.

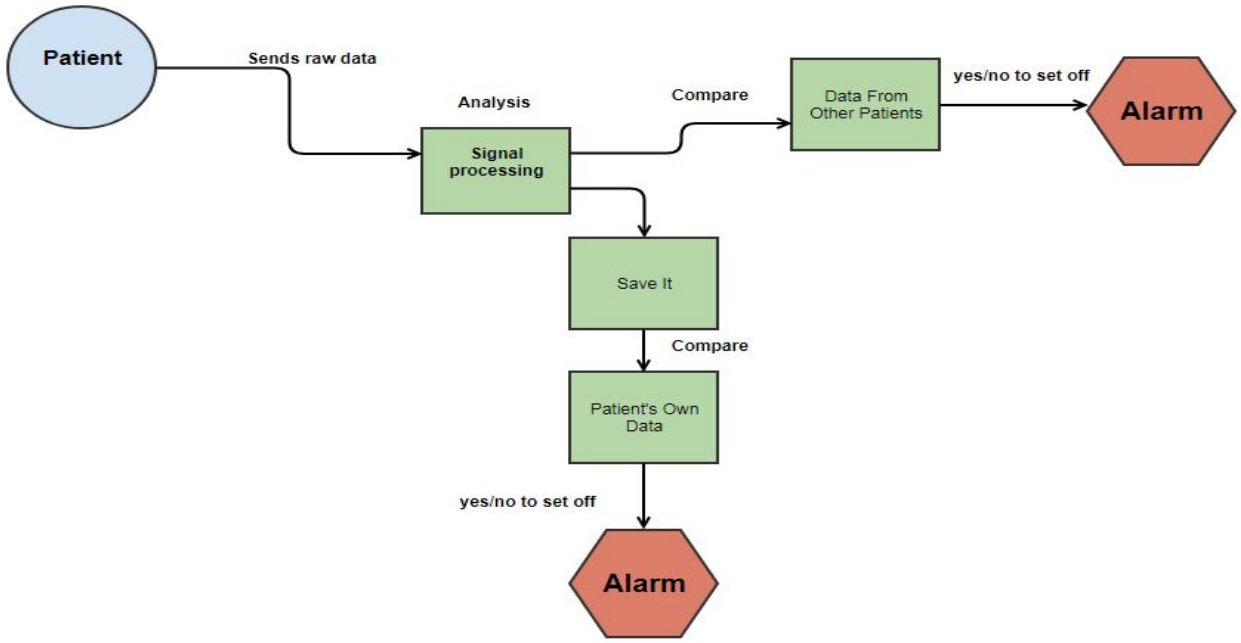
This is where a piece of software would come in handy to analyze multiple signals from a patient and determine what kind of care is called for. Our software will alleviate the stress put on ICU medical staff and patients and as a result save lives.

3.0 Solution Vision

The goal for this project is to be able to take in patient's data and feed it into our software that would perform analyses on it, and determine if an alarm needs to be triggered or not. The patient's data will then be compared between two sets of data, global and local. The Global data set is the set of data that the machines now within the ICU use to detect alarms. The local data is collected from patients as they stay within the ICU. That data will then be used to help predict if the patient is in need for an alarm to go off or not. Another way to help predict alarms, which our team will look into, but not implement, is prediction. What this will do is try to predict if a patient has a problem before a problem happens.

By using our software, patient's living in the ICU can drastically change for the better. This is for multiple reasons, but mainly because there are more accurate results for the patient's health.

The diagram below shows an overview of how our software will work. Starting with the patient, and ending with the alarms being triggered.



The diagram, as stated above, starts with the patient. Data is gathered from the patient, such as an ECG signal, and sent to processing. Once processing is done, one or two things will happen. The first thing is the data is compared to the global set of data. This will happen if there is no other information on the patient. The other thing that will happen is the data of the patient is saved. If there is enough data saved on the patient, then when new incoming data comes in, it can be compared against the saved data. This is known as the local set. Now whenever data is compared to a set of data, global or local data, it can be determined if an alarm should go off or not. Finally, if an alarm is needed, then an alarm will go off.

The last thing about the solution vision that needs to be talked about is that our part in this entire project is to make a package for it, and package will do local and global data.

4.0 Project Requirements

After detailing the solution in the section above, there are two main parts that encompass this system. First, the back end of the application will provide all the logic required to successfully determine whether an alarm should be triggered. This is the part of the program that will give it the title “smart”. Many different signals will be processed, and through analysis, the alarm could be omitted possibly with a less severe alarm being given saving nurses time in an ICU. This feeds into the second part of the application. Once the system has resolved the alarm, it needs to give some sort of feedback to the user pertaining to the alarm; this is where the user interface

comes in. Once an alarm is triggered, the nurse needs to be able to quickly see why an alarm was triggered and take the appropriate steps in order to deal with it.

The following sections will go into detail concerning the back and front end and all that they entail. A third requirement is discussed here, relating to maintainability. A user manual summarizing the features of the software coupled with guides on how to use the software will be coupled with the software to provide user support.

4.1 Functional Requirements

When a patient is connected to any number of machines, each one is doing its job without any knowledge of the others. This is disadvantageous as covered above. If we could combine signals into one centralized application, more could be done to effectively classify conditions.

4.1.1 Back End

As mentioned, the back end will do all the heavy lifting to give the best result possible. It consists of input, pre-processing, processing or detection, and data integration. After going through these four steps, a patient and a certain condition or lack of condition will be properly analyzed and given the required attention deserved.

4.1.1.1 Signal Input

The first thing that has to happen before any algorithm can do it's job is to obtain some sort of input. In an ICU, there are many different monitors all working to make sure there is nothing abnormal about a patient. Some monitors include an ECG machine, blood oxygen monitor, glucose level monitor, blood pressure monitor, and many more. For some patients, all available metrics may not be necessary. This means that we will need to implement a way of only using available inputs for a particular patient. This modular design will ensure the best result can be given with any set of inputs. If a new monitoring technique is invented and implemented in a hospital, our program will be able to adapt to this new environment and utilize the added signal. It may need further optimization, but the program will use all data available. As covered later in this section, the more inputs, and thus data we can get on a patient, the more likely we are to correctly diagnose a condition. For example, if a resource lacking ICU only has a patient connected to an ECG machine and blood oxygen sensor, our software will correctly identify heart conditions so long as it is connected. If our program sees the only two sensors producing a signal consistent with a patient experiencing a problem, an alarm will sound. But, if an input was added that monitored blood pressure and that sensor was still experiencing normal patient vital signs, an alarm could be signaled, but perhaps not as dire as before. As you can see, adding support for any number of signals is something that has to be implemented in our solution.

4.1.1.2 Pre-Processing

After our program takes any number of inputs from patient monitoring devices, there is a certain level pre-processing that needs to take place before an analysis can be done on the data. The three main pre-processing steps are denoising, which isn't required, but sometimes useful, windowing the signal into smaller more manageable pieces, and transforming the signal into multiple different domains.

Noise in a signal can come from many different sources from the hospital's own Wifi signal, which operated on the same frequency as some ICU monitors, electrical interference in the monitor's own power supply, movement, breathing rhythms, and many more. These all contribute to a signal that has unwanted artifacts. That being said, seldom will the signal need to be denoised because in doing so, key features of the signal may be lost due to removing too much signal data. Noise is something that does have the possibility to corrupt a signal, but if the noise is to the point where the signal is too obstructed by unwanted noise, the signal will most likely be less unusable with or without denoising. This being said, there are certain frequencies that can be removed that correspond to historic noise sources that have very predictable frequencies.

Another key thing that happens in the pre-processing phase is the technique of windowing. A patient's ECG signal consists of many heartbeats, and it would be unreasonable to use every heartbeat when trying to perform some analyses. There are techniques in which we will use the patient's past data in determining if a condition is real for a particular patient, but this will be covered later. When we are determining if a set of current conditions for a patient is indicative of an alarm, the immediate past signals of a patient are the more relevant ones. This translates to a window being placed around the previous 3 or 4 heart cycles. This not only keeps the data set more manageable, but it also isolates any condition that might appear in a patient's ECG signal.

Once we have a manageable window for the signal the next step involves taking the signal coming from the patient and breaking it up into different domains, which can be better utilized by analysis methods. An ECG signal coming from a patient is in its most basic domain, which is time domain. Time domain means that for any given time, there is exactly one amplitude of the voltage being received from the ECG. When this signal is processed by a transform (Fourier transform for example), the signal is shifted into frequency domain, which means that for any given frequency, there is exactly one amplitude corresponding to the "amount" of that particular frequency in the signal. Transforming the signal into a different domain can reveal information about heart conditions that are otherwise undetectable if only looking at the time domain for a

signal. For example, if a hypothetical heart condition caused the P-Q transition to occur more rapidly, this would cause a spike in a certain frequency that could be picked up when looking at the signal in frequency domain. Our program will utilize multiple methods of transformations because not all heart conditions are distinguishable from a single transform.

The last step the signal will go through in the pre-processing phase is to compute key features. In this context feature refers to any sort of metric we can use to detect abnormalities. They include, but are not limited to mean, mode, max, min, standard deviation, skewness, and entropy. Each of these features can give some insight about the variations that can occur when a patient's condition is present.

4.1.1.3 Data Processing and Detection

Related to feature extraction, once the multitudes of features have been identified, our program is going to have to find the most relevant features to use when making a decision about a patient and a possible condition. Not only is the correct set of features going to have to be chosen, the correct set of features with respect to the optimal data transform is also going to have to be taken into account. This is done through extensive data mining to sift through the features which also feed into a machine learning algorithm, but this will be covered in a later section.

Once the relevant features have been identified, the process of determining whether or not a certain set of conditions should trigger an alarm. The premise around successfully concluding that an alarm should be sounded is comparing a certain signal coming from a patient, or more accurately comparing a set of features, to a different healthy set to see if there are deviances signaling that an alarm should be triggered. There are two approaches our program will take when coming to this conclusion. The first is if the patient is so new to an ICU there is no previous data to consult when making an alarm decision. The next is the case where the patient does in fact have a certain amount of reference data that can be used to make a more informed, and thus a more accurate, decision if a set of conditions should trigger an alarm.

In the case there is no patient data or that the patient is so new to an ICU that their data is not steady enough to give "normal" results relative to that patient, other data will have to be used. When this occurs, our program will use historically healthy patient data to create a baseline for what should be expected out of a normal signal. The obvious problem with this approach is that a certain patient is not going to have the same values for a set of features as the aggregate of historically healthy patients.

Then there is the case in which offers the most informed alarm decision making. If the patient has been in an ICU for hours, our program will know what set of features are used as well as the sensitivity of those features when triggering alarms. This will ensure that if a patient comes into

an ICU, given a long enough period to get a baseline, their conditions that are deemed normal won't require triggering an alarm or may trigger a less severe alarm.

These methods of using patient data will be highly dependant on machine learning. The process of extracting the most useful features from the most useful transforms is something that will take training and tweaking. If the algorithm is trained using both healthy data and not so healthy data from the patient, not only can a baseline be determined so that abnormalities can be detected, but abnormalities that are unique to the patient that do not require an alarm can be found.

After all these aspects are factored into the final product, the back end will be able to take any number of inputs, do any necessary data pre-processing, extract all required features, and finally use the patient's own data to determine if an alarm should be triggered.

4.1.2 Front End

Once we have made a decision as to whether or not an alarm is necessary, there needs to be a way for the nurse to quickly see what the condition is as well as the signals that led up to the alarm. This is where a well designed User Interface(UI) comes in. Our monitor will need to display the typical information that a regular ICU monitor might display such as the inputted signals, alarms that may be going off, plus any options to silence alarms or do a simple query into a particular signal. Along with these standard tools our UI will need to be able to modify underlying parameters used in the alarm determination. This is because if there are certain parameters that are too strict, conditions that are not worthy of alarms may get mistaken as serious conditions. Similarly, if the parameters are too lax, conditions that require medical attention may get passed up, but these cases are covered later in section 5.

4.1.3 User Manual

After the back end and the front end are married into one unified program to detect and reduce false alarms, the only aspect missing is how to use everything. This is why we will be designing a user manual since the users of this software may not be as technologically inclined as the developers. This user manual will handle everything from how to tweak the parameters mentioned above to first time startup. The exact content of this manual will be determined at a later date when the software is near completion and the exact specs are realized.

4.2 Performance (non-functional) Requirements

4.2.1.1 Signal Input

The signal input part of this project should be simple to have done. With the endgame in mind, all external devices will connect to a new device, our device, and stream their data, the signal, into our device. This will all be done in real time as well, and the only user interaction of this will be the user plugging in other devices to our device. From here, our device will have what it needs to do pre-processing.

4.2.1.2 Pre-Processing

The pre-processing program needs to be able to understand what signal it is looking at, and from there, it can decide what pre-processing technique to use on that signal. This can be done in a number of ways, for example, if the plug-in on the device has labels on it for each signal that can be processed, then all that is needed is the system to be set up right to pre-process the signals correctly. Or there can be an algorithm made to understand what signal is coming in. The other processing techniques such as denoising, windowing, and transforming will all have their own processes. Each of these processes will be done within parallel so that there is nothing being bogged down when processing. Once this is all done, the processed signals will go through feature extraction methods and have the data it needs to continue on to the next part, Data Processing. The system will need to act in real time when processing signals, as soon as the data comes in, it needs to be prepped for Data Processing and Detection. There is no form of user interaction here, all of this is done by software.

4.2.1.3 Data Processing and Detection.

Here, the program will analyze the features that were extracted and judge them. This means, given a signal, a set of features would have been extracted from a signal that would give the best results if someone would need an alarm to go off or not. If the features of that signal would indicate that an alarm is needed, then depending on what other signals stated, if there are other signals, and it was all yes there needs to be an alarm, then an alarm will sound. This process will need to be in real time as well. The system will know if a feature indicates if something is bad or not by known data sets of features. The processed features will be compared to the known data set of features, which then it will state if there is need for concern with the patient. Where the data sets come from were explained in 4.1.1.3.

4.2.2 Front end

The front end of this code will need to have an easy to use interface for the nurses to use with ease. All current stats of the patients will be monitored with User Interface. This will be done in real time, and it would most likely be recommended to train the nurse about the User Interface before using it.

4.2.3 User Manual

No software here needed, just a comprehensive guide to the software in play.

4.3 Environmental Requirements

Our project uses specific environments and with these environments comes a number of constraints. These constraints are entirely software related at this point due to the fact that our project outcome is a software package. These constraints are:

- Using MATLAB programming language.
- Using the MATLAB library WFDB.

The programming language being MATLAB is a constraint that was given to us by our mentor due to previous code for similar projects having been already written in MATLAB. This allows us to utilize some work that other researchers have already done to some degree, which gives us the ability to use our time for more project specific tasks. MATLAB is also used often for her research, and thus she has an understanding of the way it works.

Similar to one of the reasons why we are constrained to using MATLAB, the MATLAB library WFDB contains a number of functions to do wave analyses and manipulate the MIT ECG database signals. Again, this allows us to not have to reinvent the wheel whenever we need to use a method to achieve our project goals. An example of this is the function called ecgpuwave.m in that library, which is a QRS detection function used in peak detection. This is useful for our project because peak detection is used in analysis of heart rate variability, which is a feature we need to use in conjunction with others for our analysis.

5.0 Potential Risks

All of our risks for this project revolve around our software being inaccurate. Specifically, if our software is not accurate enough, it can lead to false positives or worse yet, false negatives.

False positives are safe, but means our software isn't working as intended. This also means that our software is not reducing the number of false alarms. This risk is much more likely to occur than false negatives, is not life threatening, and could be caused by a patient being new, so pre-existing conditions are unknown and not allowing our software to make comparisons using local data. We can mitigate this risk by gathering more data on the patient and possibly having less strict parameters.

False negatives are more serious, in this case real alarms that could represent danger for a patient is taken as a false alarm and result in medical staff not responding to the alarm. In this case, lives could be lost and our number one priority is to mitigate this risk. Possible causes of this kind of risk are bugs in our algorithms or design oversights as well as signal corruption. We can mitigate this risk by being diligent in the design of the software architecture and ruthless testing of our algorithms. If we run into this problem while testing, we can fix it by issuing parameter tuning. If the problem persists, making changes to our overall design.

6.0 Project Plan

Our project is going to be done within two phases. Each phase will have sub parts to them to be completed. Phase one will mostly be understanding signal processing and the different techniques you can use. Using multiple techniques on ECG signals, we will collect data known as features from the processed signals, sanity check the signals, process other signals and collect mass amounts of data.
Phase one is expected to be finished in the fall semester.
Phase 1 milestones.

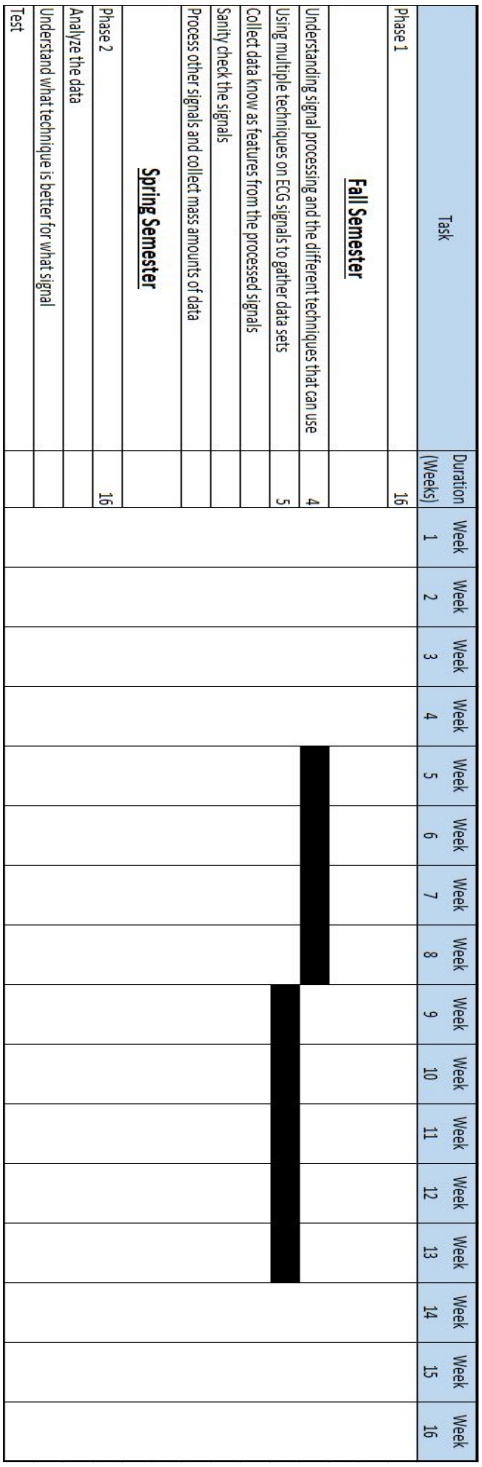
1. Understanding signal processing and the different techniques that can be use
2. Using multiple techniques on ECG signals to gather data sets
3. Collect data known as features from the processed signals
4. Sanity check the signals
5. Process other signals and collect mass amounts of data

Phase two will be more on analyzing that data we collected in phase one. In phase two, the goal will be to understand what signal processing techniques will be better to use on each signal that will be processed. By doing so, we can better predict a false alarm.
Phase two is expected to be finished in spring semester.
Phase two milestones.

1. Analyze the data.

2. Understand what technique is better for what signal
3. Test.

Here is a Gantt chart to show our path plan in a visual way.



7.0 Conclusion

To conclude this document, the false alarms that the ICU have to deal with are costing people their lives and stressing people out. The goal of this project is to help reduce that. Since false alarms can cause nurses to get a "cry wolf effect", the plan is to reduce that by analyzing the inputs of the devices that the patients are connected to. By having a device look at all of the inputs, a more accurate alarm can be set. This document contained an outline to accomplish our goal. The section of this document related to functional requirements give an in depth plan for the development for this software, and the project plan shows how that will be accomplished. From this outline, the development of this software should be relatively straight forward. Of course, there will be aspects of mathematical concepts that need to be learned and understood; which if that would be the only hurdle for our team, then we should be able to make do. In the end, when successful, this software will be used to help reduce false alarms in the ICU, and in doing so, help reduce patient stresses and deaths.