

**CORRELATION AND REAL-TIME CLASSIFICATION OF PHYSIOLOGICAL STREAMS FOR
CRITICAL CARE MONITORING**

by

Anirudh Thommandram

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Electrical and Computer Engineering

in

The Faculty of Engineering and Applied Sciences
Program

University of Ontario Institute of Technology

December, 2013

© Anirudh Thommandram, 2013

Certificate of Approval

CERTIFICATE OF APPROVAL

Submitted by **Anirudh Thommandram**

In partial fulfillment of the requirements for the degree of

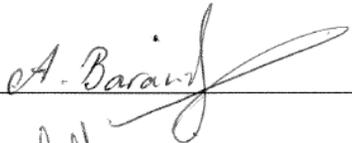
Master of Applied Science in Electrical & Computer Engineering

Date of Defence: **2013/12/02**

Thesis title: Correlation and Real Time Classification of Physiological Streams for Critical Care Monitoring

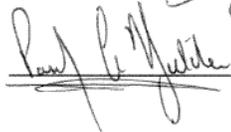
The undersigned certify that the student has presented **his** thesis, that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate through an oral examination. They recommend this thesis to the Office of Graduate Studies for acceptance.

Examining Committee

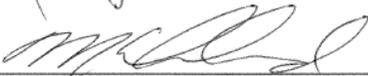


Dr. Ahmad Barari

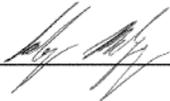
Chair of Examining Committee



Dr. Paul Yelder
External Examiner



Dr. Mikael Eklund
Research Supervisor



Dr. Carolyn McGregor
Research Supervisor



Dr. Andrew James
Examining Committee Member

As research supervisor for the above student, I certify that I have read and approved changes required by the final examiners and recommend the thesis for acceptance:



Dr. Mikael Eklund

Research Supervisor

Abstract

This thesis presents a framework for the deployment of algorithms that support the correlation and real-time classification of physiological data streams through the development of clinically meaningful alerts using a blend of expert knowledge in the domain and pattern recognition programming based on clinical rules. Its relevance is demonstrated via a real world case study within the context of neonatal intensive care to provide real-time classification of neonatal spells. Events are first detected in individual streams independently; then synced together based on timestamps; and finally assessed to determine the start and end of a multi-signal episode. The episode is then processed through a classifier based on clinical rules to determine a classification. The output of the algorithms has been shown, in a single use case study with 24 hours of patient data, to detect clinically significant relative changes in heart rate, blood oxygen saturation levels and pauses in breathing in the respiratory impedance signal. The accuracy of the algorithm for detecting these is 97.8%, 98.3% and 98.9% respectively. The accuracy for correlating the streams and determining spells classifications is 98.9%. Future research will focus on the clinical validation of these algorithms and the application of the framework for the detection and classification of signals in other clinical contexts.

Publications related to this thesis

Thommandram, A., Pugh, J. E., Eklund, J. M., McGregor, C., & James, A. G. (2013, January). Classifying neonatal spells using real-time temporal analysis of physiological data streams: Algorithm development. In Point-of-Care Healthcare Technologies (PHT), 2013 IEEE (pp. 240-243). IEEE.

Thommandram, A., Eklund, J. M., & McGregor, C. (2013, May). Detection of Apnea from Respiratory Time Series Data Using Clinically Recognizable Features and kNN Classification. In Engineering in Medicine and Biology Society (EMBC), 2013 Annual International Conference of the IEEE. IEEE

Kamaleswaran, R., Thommandram, A., Zhou, Q., Eklund, M., Cao, Y., Wang, W. P., & McGregor, C. (2013, June). Cloud Framework for Real-time Synchronous Physiological Streams to Support Rural and Remote Critical Care. In Computer-Based Medical Systems (CBMS), 2013 26th International Symposium on. IEEE

Edward Pugh, Anirudh Thommandram, Eugene Ng, Carolyn Mcgregor, Mikael Eklund, Indra Narang, Jaques Belik, Andrew James, Classifying neonatal spells using real-time temporal analysis of physiological data streams—algorithm development, Journal of Critical Care, Volume 28, Issue 1, February 2013, Page e9, ISSN 0883-9441, <http://dx.doi.org/10.1016/j.jcrc.2012.10.033>.

(<http://www.sciencedirect.com/science/article/pii/S0883944112003899>)

Pugh, J. E., Thommandram, A., McGregor, C., Eklund, M., James, A., Classifying Neonatal Spells Using Real-Time Temporal Analysis of Physiological Data Streams – Verification Tests and Preliminary Algorithm Refinement, Journal of Critical Care, Volume 28, Issue 6, December 2013, Pages e40-e41, doi: 10.1016/j.jcrc.2013.07.036

Table of Contents

CERTIFICATE OF APPROVAL	II
ABSTRACT	III
PUBLICATIONS RELATED TO THIS THESIS	IV
TABLE OF CONTENTS	V
LIST OF FIGURES.....	VIII
LIST OF TABLES	XI
CHAPTER 1 - INTRODUCTION	1
1.1 - CURRENT STATE OF BEDSIDE MONITORS	1
1.2 - PROBLEM STATEMENT	2
1.3 - THESIS OBJECTIVE	3
1.4 - SCOPE OF THESIS	4
1.5 - ORGANIZATION OF THESIS	5
CHAPTER 2 - BACKGROUND AND RELATED WORK	6
2.1 - DATA COLLECTION.....	6
2.2 - DESIGNING INTELLIGENT SYSTEMS AND REAL-TIME CONSIDERATIONS	10
2.3 - REQUIREMENTS	15
2.3.1 - <i>Data Acquisition</i>	15
2.3.2 - <i>Real-Time Analytics</i>	16
2.4 - ARTEMIS FRAMEWORK FOR REAL-TIME DATA CAPTURE AND ANALYTICS	17
CHAPTER 3 - CASE STUDY: CLASSIFYING SPELLS IN THE NICU	20
3.1 - NEONATAL SPELLS	21
CHAPTER 4 - METHODOLOGY	25
4.1 - EVENTS IN INDIVIDUAL STREAMS	26
4.1.1 - <i>Numerical data streams</i>	28
4.1.2 - <i>Waveform data</i>	31
4.1.3 - <i>Artefact Detection</i>	33
4.2 - SYNCING STREAMS	34
4.3 - DETERMINING THE EPISODE PROFILE	36

4.3.1 - Real-time considerations and utilities.....	38
CHAPTER 5 - IMPLEMENTATION: CLASSIFYING SPELLS IN THE NICU	40
5.1 - HEART RATE AND BLOOD OXYGEN SATURATION EVENTS	41
5.1.1 - Raw input data.....	43
5.1.2 - Tunable parameters.....	44
5.1.3 - Absolute Alarm operator	44
5.1.4 - Baseline Operator	45
5.1.5 - Relative alert operator.....	46
5.1.6 - Live display output	47
5.1.7 - Validity stream.....	48
5.2 - BREATHING EVENTS.....	49
5.2.1 - Raw input data.....	50
5.2.2 - Live display output	51
5.2.3 - Detecting breaths.....	52
5.2.4 - Absolute alert operator.....	54
5.2.5 - Relative alert operator.....	55
5.3 - SPELLS CLASSIFICATION	57
5.3.1 - Incoming individual alert data.....	59
5.3.2 - Syncing streams	59
5.3.3 - Generating episode profile.....	60
5.3.4 - Classifying the spell.....	61
5.4 - PRESENTATION OF CLASSIFICATION RESULTS	66
CHAPTER 6 - RESULTS AND DISCUSSION	70
6.1 - EXPERIMENTAL DESIGN.....	70
6.1.1 - Streams Data Display Application.....	70
6.1.2 - Validating individual alerts	72
6.1.3 - Validating classifications.....	74
6.2 - RESULTS.....	74
6.2.1 - Relative change algorithm and breath pause detection.....	75
6.2.2 - Comparison to absolute threshold alerts.....	77
6.2.3 - Spells classification algorithm.....	78

CHAPTER 7 - CONCLUSION	82
7.1 - RESEARCH AND FINDINGS	82
7.2 - SUGGESTIONS FOR FUTURE WORK	84
7.3 - CONCLUDING REMARKS.....	86
CHAPTER 8 - REFERENCES	87
APPENDIX A – POLYSOMNOGRAPHY VALIDATION	92
A.1 - EXPERIMENTAL DESIGN.....	92
<i>A.1.1 - Sleep Study Status Application</i>	<i>94</i>
A.2 - PRELIMINARY RESULTS	96

List of Figures

FIGURE 1. HIGH LEVEL ARCHITECTURE	17
FIGURE 2. STRUCTURE OF A SPELL.....	22
FIGURE 3. OVERALL ARCHITECTURE OF FRAMEWORK	25
FIGURE 4. INDIVIDUAL ALERT MODULES THAT CURRENTLY MAKE UP THE FRAMEWORK’S TOOLKIT	27
FIGURE 5. LOGIC OF ABSOLUTE ALERT STREAM	28
FIGURE 6. DETECTING RELATIVE CHANGE	29
FIGURE 7. LOCKING BASELINE	30
FIGURE 8. LOGIC OF RELATIVE ALERT STREAM.....	30
FIGURE 9. FLOW DIAGRAM DESCRIBING PEAK DETECTION ALGORITHM	32
FIGURE 10. COMPARING MAXES AND LIVE PEAK OUTPUT STREAMS FROM PEAK DETECTION ALGORITHM.....	33
FIGURE 11. LOGIC OF VALIDITY CHECK OPERATOR	34
FIGURE 12. BEHAVIOR OF SYNC MODULE WITH SOURCE STREAMS OF DIFFERENT SAMPLE RATES.....	36
FIGURE 13. EPISODE PROFILE GENERATION FROM SYNCED STREAMS.....	37
FIGURE 14. EXAMPLE CLASSIFICATION OF EVENT PROFILES BASED ON SIMPLE RULES.....	38
FIGURE 15. SCREENSHOTS OF SIMPLE INTERFACE APPLICATION TO CHANGE PARAMETERS OF MULTIPLE RUNNING SAMPLE SPL GRAPHS.....	39
FIGURE 16. SPELLS CLASSIFIER DESIGN USING STREAMS	41
FIGURE 17. RULES FOR DETECTING RELATIVE EVENTS IN INDIVIDUAL STREAMS	41
FIGURE 18. STREAMS GRAPH FOR DETECTING RELATIVE CHANGE IN SpO_2	43
FIGURE 19. SPL OPERATORS FOR HANDLING RAW SpO_2 DATA	43
FIGURE 20. SPL OPERATORS FOR UPDATING PARAMETER VALUES	44

FIGURE 21. SPL OPERATOR FOR DETERMINING ABSOLUTE ALERTS	45
FIGURE 22. SPL OPERATOR FOR DETERMINING BASELINE	45
FIGURE 23. SPL OPERATORS FOR DETERMINING RELATIVE CHANGE ALERTS	46
FIGURE 24. SPL OPERATOR FOR SENDING DATA TO LIVE DISPLAY	48
FIGURE 25. SPL OPERATORS FOR PERFORMING VALIDITY CHECK	49
FIGURE 26. STREAMS GRAPH FOR DETECTING RELATIVE CHANGE EVENTS IN THE RI SIGNAL.....	50
FIGURE 27. SPL OPERATORS FOR HANDING RAW RI DATA	51
FIGURE 28. SPL OPERATORS FOR SENDING DATA TO LIVE DISPLAY	52
FIGURE 29. SPL OPERATORS FOR CALCULATING LOCATIONS OF BREATHS	53
FIGURE 30. DETECTING LOCATIONS OF BREATHS IN A RAW RI SIGNAL USING LIVE PEAK DETECTION	54
FIGURE 31. SPL OPERATORS TO PERFORM THE ABSOLUTE ALERT ALGORITHM FOR AN RI SIGNAL	55
FIGURE 32. SPL OPERATORS FOR DETERMINING MISSED BREATH ALERTS	55
FIGURE 33. LOGIC OF RI RELATIVE ALERT STREAM.....	56
FIGURE 34. STATE DIAGRAM OF RI RELATIVE ALERT DETECTION ALGORITHM	58
FIGURE 35. STREAMS GRAPH FOR CLASSIFYING SPELLS	59
FIGURE 36. SPL OPERATOR FOR HANDLING INDIVIDUAL ALERT DATA	59
FIGURE 37. SPL OPERATORS FOR SYNCING STREAMS.....	60
FIGURE 38. SPL OPERATOR FOR GENERATING EPISODE PROFILE.....	61
FIGURE 39. SPL OPERATORS RESPONSIBLE FOR RULE BASED CLASSIFICATION AND REPORT GENERATION.....	61
FIGURE 40. STATE DIAGRAM OF THE SPELLS CLASSIFICATION PROCESS.....	64
FIGURE 41. CONTENTS OF THE CLASSIFIER SUMMARY FILE	67
FIGURE 42. EVENT SUMMARY FOR A POSSIBLE ISOLATED DESATURATION CLASSIFICATION	67

FIGURE 43. EVENT BUFFER FILE FOR A POSSIBLE ISOLATED DESATURATION EVENT. THIS FILE CONTAINS THE SUMMARY INFORMATION AS WELL AS THE ENTIRE EPISODE BUFFER OF ALERT VALUES.....	68
FIGURE 44. THE STREAMS DATA DISPLAY APPLICATION ENABLES A REVIEWER TO PAN THROUGH HOURS OF HR, RI, AND SPO ₂ DATA TO VALIDATE ALERTS.	71
FIGURE 45. STREAMS DATA DISPLAY APPLICATION SHOWING RAW SPO ₂ , SPO ₂ ABSOLUTE ALERT, SPO ₂ RELATIVE ALERT	73
FIGURE 46. STREAMS DATA DISPLAY APPLICATION SETUP TO VIEW RAW RI, DETECTED BREATHS, AND RI RELATIVE ALERT	74
FIGURE 47. RELATIVE CHANGE ALGORITHM COMPARED TO CLINICALLY ANNOTATED TRACE	76
FIGURE 48. SECTION OF RAW RI ALONG WITH DETECTED BREATHS WHERE MULTIPLE SHORT RI RELATIVE ALERTS IN QUICK SUCCESSION ARE DETECTED	77
FIGURE 49. RELATIVE TO ABSOLUTE THRESHOLD COMPARISON	78
FIGURE 50. DISTRIBUTION OF ALL EVENTS DETECTED BY THE CLASSIFIER	79
FIGURE 51. DISTRIBUTION OF ALL CLINICALLY SIGNIFICANT MULTI-SIGNAL SPELLS EVENTS	80
FIGURE 52. LAYOUT OF EQUIPMENT FOR COLLECTING DATA DURING POLYSOMNOGRAPHY	93
FIGURE 53. SLEEP STUDY STATUS APPLICATION SHOWING SYSTEM IS OPERATIONAL.....	95
FIGURE 54. SLEEP STUDY STATUS APPLICATION SHOWING A SYSTEM ERROR.....	96

List of Tables

TABLE 1. VALUES OF PARAMETERS AND THRESHOLDS FOR RELATIVE CHANGE DETECTION IN HR AND SpO2	42
TABLE 2. THE DIFFERENT SPELLS TYPES AND THEIR CORRESPONDING SEQUENCE OF EVENTS.....	62
TABLE 3. ALL THE STATE TRANSITIONS BEING MONITORED AND THEIR EVENT CODE VALUES FOR USE IN THE CLASSIFICATION STAGE	63
TABLE 4. SUMMARY OF THE ACCURACY OF SPELLS CLASSIFICATIONS BY THE ALGORITHMS	79
TABLE 5. TRUTH DATA FROM A POLYSOMNOGRAPHY LISTING THE TYPES OF EVENTS DETECTED AND THE NUMBER OF OCCURRENCES OF EACH EVENT.....	97
TABLE 6. OUTPUT FROM THE CLASSIFIER ALGORITHM LISTING THE TYPES OF EVENTS DETECTED AND THE DISTRIBUTION OF OCCURRENCES.....	97

Chapter 1 - Introduction

This thesis presents a method for correlation and real-time classification of physiological streams for critical care monitoring. The method is instantiated in a real-time computer software environment enabled by stream computing. Its relevance is demonstrated via a real world case study within the context of neonatal intensive care to provide real time classification of neonatal spells. A neonatal spell is a cardiorespiratory event that presents with variable combinations of cessation of breathing, decrease in blood oxygen saturation, and decrease in heart rate. Common causes of spells include physiological immaturity, respiratory conditions, and infection. A spell may be the only manifestation of a seizure caused by a brain haemorrhage or stroke. Determination of the cause of spells often involves multiple invasive investigations before the cause is established and the real-time classification of spells helps narrow down the list of potential diagnoses thus reducing the number of invasive tests.

1.1 - Current State of Bedside Monitors

In a critical care environment, there is a need for continuous monitoring of the physiological state of patients. But most intensive care units are limited by human resources and technological assistance to perform this in a truly continuous fashion. The techniques for gathering physiological information has come a long way since the 1960s when vital sign monitoring was first implemented at the bedside. Much advancement has been made in the design of sensor circuitry and signal processing algorithms to reduce noise in signals [Thomas et al., 1979]. Today, these devices are a staple in any patient care scenario and can gather many physiological signals simultaneously and display them on a screen for interpretation by humans.

In fact, the number of physiological parameters that a standard bedside monitor collects and displays on screen is staggering. However, even though the amount of raw data displayed is large, the ability to extract useful information is mostly left up to the person interpreting the screen. Even state-

of-the-art patient monitors offer very limited data integration and analysis for non-trivial clinical decision support as they still follow the single-sensor-single-indicator approach (SSSI) [Drews, 2008]. A result of some of the simplistic alarm algorithms present in current patient monitors is that too many false alarms are generated. This point needs to be clarified – the alarms thrown are “correct” in detecting anomalies in the physiological signal. However, this does not necessarily mean that an audible alert was necessary. Anomalies are more than likely to be artefacts caused by noise from sensors. Even with a clean signal, an alert based on a simple threshold breach of a parameter may be of no clinical significance in the context of the patient and other physiological data streams [Koski et al., 1990].

It is this context that is lost in current standard bedside monitors. The monitors lack the capability of correlating several data streams into one, “smarter” alert. The high output of clinically insignificant audible alerts may have additional unintended consequences such as the phenomenon of alarm fatigue [Meredith & Edworthy, 1995; Tsien, 1997b].

1.2 - Problem Statement

The traditional method for developing such algorithms or models is performing retrospective analysis of the physiological data and clinical annotations. The data is lined up with the annotations and panning through from start to finish one can see the signal patterns that correspond with each annotation. One of many machine learning algorithms such as neural networks [Guez & Nevo, 1996], decision trees [Tsien et al., 2000], and fuzzy logic [Mirza et al., 2010; Becker et al., 1994; Kickert & Mamdani, 1978] is applied to the raw signals and the corresponding annotations. While technically this approach seems sound, there are several limitations to the method. One of the main issues is that the physiological data is collected by the bedside monitor, but the annotations are collected separately. This can cause the timestamps of data and events to be unsynchronized. While the offset may not be very much and quite possibly insignificant to a human observer, when that data is used to train models and algorithms it

introduces error and uncertainties. To mitigate such errors, some assumptions are made on the data and this leads to results that are potentially clinically invalid.

There is another major concern when using black box modeling systems for classifying conditions from physiological signals. Such systems make classifications based on patterns and similarities learned from training data and so the actual behaviors in the signal are not clearly defined. This adds to the uncertainty in a way that cannot be related to the classification process performed by human experts. The way these processes define features is not provided for the human to examine and it does not work the same way that human decision making does when analysing the same data streams. Many attempts at implementing clinical decision support systems are faced with this issue and clinical experts that the systems are designed to support are not confident in the output [Waterson, 1988].

1.3 - Thesis Objective

In order to bring bedside monitor alerts into a new age, a system that uses real-time streaming of physiological data together with algorithms that check the correlation of several parameters of different sampling frequencies is required. A framework for developing clinically significant alerts using a blend of expert knowledge in the domain and pattern recognition programming based on clinical rules is the objective of this thesis.

This methodology is implemented for the task of automated recognition and classification of neonatal spells. The physiological data streams of heart rate (HR), respiratory impedance (RI) and blood oxygen saturation (SpO_2) are captured from the bedside monitors and events are detected in the individual streams, followed by analysing the correlation between the events in the different streams, resulting in a classification of the type of spell that has occurred.

1.4 - Scope of Thesis

The work presented in this thesis is part of the larger Artemis Project that is in progress in the Health Informatics Research Laboratory (HIRL) at UOIT. In the larger project of Artemis, there are several topics surrounding real-time clinical monitoring. Research on applying automated monitoring techniques for apnoea of prematurity [Catley et al., 2011], developing an accurate pain profile for neonates [Naik et al., 2013], and monitoring conditions related to retinopathy of prematurity [Cirelli et al., 2013] are only a subset of the projects being worked on. In addition to the clinical conditions being studied, there is research on ways to visualise data and how best to present algorithm outputs to clinicians [McGregor et al., 2013].

The framework developed as part of this thesis will be used for developing rule-based algorithms for many clinical conditions, however this thesis is limited to validating the framework's functionality in the context of neonatal spells. While the development of the algorithms for correlating and classifying physiological streams on a real-time platform is the main contribution of this thesis, the design of the clinical rules is outside the scope of this thesis. That is the contribution of Dr. Edward Pugh.

Dr. Pugh is a neonatologist who is a research fellow in the Division of Neonatology at The Hospital for Sick Children in Toronto. For his Masters in Health Informatics he has been designing the requirements for a real-time automatic detection system for classifying neonatal spells. His expert primary knowledge in the subject of neonatal spells will be used to accurately translate relevant events in physiological signals into automated detection algorithms.

Also to note is that the presentation of algorithm outputs and results to clinicians is not in the scope of this research. The framework designed in this thesis outputs classifications onto a text-based report file. While the report is legible and clearly understandable, the file is optimized for machine

reading and to be used to develop custom visualizations and user interfaces for clinicians. This task is the research work of Rishikesan Kamaleswaran, another Artemis team member.

1.5 - Organization of Thesis

This thesis presents some background on the medical condition that has been used to demonstrate the framework that enables the correlation and real-time classification and the software tools employed in the framework. This and an overview of the related work completed in the field of automated classification of physiological data are presented in Chapter 2. Chapter 3 presents some background for the case study of classifying neonatal spells as well as an overview of the research on automated detection schemes for spells. The next chapter details the methods for the contribution of this thesis work for developing algorithms for the purpose of producing clinically significant alerts and their implementation. Chapter 5 demonstrates the use of the developed framework in implementing algorithms to classify neonatal spells. Chapter 6 presents the testing and validation procedures and the results of implementing the framework. Chapter 7 concludes this thesis with summary findings of the research in support of the research questions proposed. Suggestions for future work are proposed in the conclusion also.

Chapter 2 - Background and Related Work

Before describing the requirements of a new and improved patient monitoring system, a review must first be performed of studies that have already been completed in this area of research. Developments in critical care monitoring have been ongoing in both industry as well as academia. In this chapter, a review of the research related to correlation and real-time processing of data streams is presented. The review is categorized into two core components of an intelligent decision support system and establishes the need for the framework proposed in this thesis.

2.1 - Data collection

Collection of the physiological data is the first step in the development of a patient monitoring system. As technology has progressed the amount of physiological data as well as clinical information about patients has grown significantly. As such, developing systems that record this data securely and at a suitable sampling rate is a large research topic on its own. In this section, some of these systems are described.

Sukuvaara et al. developed a system called DataLog which would connect to bedside monitors through an RS232 serial interface to collect physiological signals every five seconds. In addition, they also had direct access to the patient data management system of the intensive care unit (ICU) and collected clinical information with recorded times. They performed some trending analysis to the signals and combined it to heuristic if-then rules to create a knowledge-based alarm system [Sukuvaara, 1993]. Capturing a data point once every five seconds is not enough to make accurate assessments for complex behaviors such as spells. Also, only numeric signals are collected with DataLog and no waveform data is captured which is an important component of detecting conditions in real-time.

One of the major contributions to the collection of physiological data in critical care environments came from Moody et al in the mid-90s. They developed customized software to log the

signals coming from the Hewlett Packard CMS (“Merlin”) bedside monitors that were being used in the medical, surgical, and cardiac ICUs of Boston’s Beth Israel Hospital. They implemented this by using a pair of RS232 serial interface cards in the monitor and communicating the data to a standard PC with a Digiboard PC/4e serial interface over 38400 baud. At this maximum output rate that the monitors can support, they were able to record three ECG signals each sampled at 500 Hz and four or five other signals sampled at 125 Hz, in addition to periodic measurements and alarm messages. Clinical data derived from the patient’s medical records was also captured. The physiological data and clinical information were stored in a relational database called MIMIC, which has been made available to the public for research [Moody & Mark, 1996]. While the amount of data collected is impressive, their approach was to strictly record and store the data for the purpose of retrospective analysis. There was no functionality to serve the data for any on line processing.

Tsien et al. set out to better understand the alarms in a pediatric intensive care unit and to assess the positive predictive value of different monitoring devices. In their prospective study, they collected numerical data, from SpaceLabs Medical bedside monitors, that was sent to a bedside laptop every five to ten seconds. On that laptop, they also recorded annotations of relevant clinical events onto a Microsoft Access database. This was done manually by a trained observer [Tsien, 1997a; Tsien 2000]. As with the previously described research, this project involved only recording the data for the purpose of retrospective analysis. The annotations collected from a separate source are unified with the physiological data through a series of post-collection processing programs. Also, collecting data directly from patient monitors using a laptop is not a methodology that is easily scalable.

It is interesting to note that RS232 or serial port has been the most widely used interface when collecting data from bedside monitors. This has been noted by Chambrin as early as 1989 [Chambrin, 1989]. RS232 has remained the standard in communications with bedside monitors for over 15 years.

Even with the advent of networked monitors that are connected to a hospital's local area network (LAN), most prototype systems proposed by researchers still use the RS232 interface. This is because of its position as a de facto standard in the medical computing community [Eddleman et al., 1990]. It could also be because the device's LAN network connection is being used for the operations of the hospital staff such as central monitoring feeds and in order to collect data without interrupting hospital processes, researchers resort to using the serial interface.

Still, as early as 1996, researchers began to investigate the capabilities of a networked system with remote data collection possibilities. Wang et al. developed an online monitoring system whereby they could send collected physiological signals from bedside monitors over the Internet to be displayed on web pages [Wang et al., 1996]. While this opened many doors to the possibilities of remote online analytics, it also began the discussion on security of sensitive data over the internet. The updating cycle for the data in this study was one minute, which is far too slow to perform any useful analytics but as a pilot study the researchers have shown that it can be feasible to collect data from patient monitors remotely.

Saeed et al. took advantage of the fact that hospitals were upgrading their ICUs with network capable monitors and equipment. They designed a system that collected physiological and clinical data from the hospital's information management system for the purpose of creating a temporal ICU patient database called MIMIC II. They monitored patients admitted to an 8-bed medical intensive care unit and an 8-bed coronary care unit. The physiological data consisted of four continuously monitored waveforms (two leads of ECG, arterial blood pressure, pulmonary artery pressure) sampled at 125 Hz, 1-minute parameters (HR, BP, SpO₂, cardiac output), as well as monitor-generated alarms. All this data is collected by Philips patient monitors and gets transmitted to a Philips Information Center Database Server. They captured the relevant clinical data by interfacing with the Philips central monitoring

system that was being used to house information such as laboratory results, nurses' text notes, medications, fluid balance, and patient demographics [Saeed et al., 2002]. The MIMIC II database has been widely used in research as it is a rich source of data, but an important thing to note is that the steps taken to achieve this are specific to this case and the model isn't as easily generalised to different technologies and hospitals. The research team worked closely and had the assistance and cooperation of the manufacturer in order to interface with these proprietary systems. The strength in their approach is the ability to vary the presentation of data depending on what kind of research the data is being used for. Users of the database can extract a detailed record of a single signal, or for more temporal analysis data from many signals can be displayed in one view. However this ability to provide data temporally can only be done after considerable pre-processing and data fusion and is inherently retrospective.

The extent to which a well-networked hospital can benefit from continuous monitoring systems is evident in the real-time, continuous physiological data acquisition system for the study of disease dynamics in the pediatric ICU described by Goldstein et al in 2003. They collected data from 16 Philips Merlin patient monitors which were sampling slowly varying signal averages such as heart rate at 0.98 Hz and sampling other signals such as ECG at 500 Hz and respiratory and pressure waveforms at 125 Hz. This data is sent through the intranet of the Oregon Health & Science University for display and analysis, in addition, it is also sent over the Internet to the Portland State University and the University of Pittsburgh Medical Center for display and analysis remotely [Goldstein et al., 2003]. Their system was only designed to collect and store data from multiple monitors simultaneously and not to perform any online analysis however they went on to perform some linear and nonlinear analysis techniques on the data retrospectively. As well, clinical annotations were combined with the physiological data retrospectively.

2.2 - Designing intelligent systems and real-time considerations

Researchers in patient monitoring techniques have always tried to add complexity and intelligence to better aid medical professionals. Even the development of an electronic patient monitor itself was an advance in monitoring as prior to being connected to several sensors, a patient's condition was monitored by putting one's finger on the patient's pulse and listening to the chest through a stethoscope [Samuels, 1986; Meredith & Edworthy, 1995]. As patient monitors began to accurately measure physiological signals, they started incorporating alarm systems to notify clinicians if any of the signals were outside a normal range. However, to this day these alarm systems on modern monitors that only detect numeric threshold breaches of a specific signal. There are many reported problems with the number of auditory alarms going off in intensive care units, many of which are unnecessarily loud and continuous [Meredith & Edworthy, 1995]. This occurs because of the simple nature of the algorithms and a lack of clinical context. Also historically, manufacturers have a 'better safe than sorry' attitude with regards to generating an alert and making it as loud as possible [Meredith & Edworthy, 1995]. Generally, these alarm systems do not consider multivariate interactions, partly due the limited processing capacity to build the temporal layers necessary to associate multiple physiological events with multiple clinical conditions [Sukuvaara et al., 1993]. Therefore, a major area of research is being focussed on how to incorporate artificial intelligence techniques and related methods in the field of physiological data monitoring. In this section, some of these systems are described.

The knowledge based alarm system developed by Sukuvaara et al. was designed for monitoring patients undergoing cardiac surgery. One of its components, DataLog, was described in the previous section. The other part of their system is InCare which is a knowledge-based alarm system that implemented 87 rules to provide highly specific alarms. The rules would use the latest values of physiological signals in addition to detecting trends in the data over time. One of the factors the researchers considered is that to make it sufficiently intelligent, InCare had to operate even with

incomplete data which is a common occurrence during intensive monitoring due to the movements of patients for tests and procedures. To this end, multiple rules and multiple conditions in the rules were combined by logical OR operators to preserve the multi-route structure of inference [Sukuvaara et al., 1993]. They used the prototype system in a trial with ten patients and then upgraded the rule base and tested it again with 15 patients. The sensitivity was 100% in both tests, and the specificities increased from 20% to 73% [Koski et al., 1994]. While the alarms generated by InCare take into account multiple variables, it only checks the current state or current trend of a specific signal. It cannot correlate temporal events that happen in sequence and this is a necessary component of classifying complex conditions. However, it should be noted that this research was towards a replacement of the bedside monitor so a constraint of the research is for the processing to be achieved on a microcomputer.

Another approach is to use classification trees to detect patterns in data. Tsien et al. explored decision tree induction and decision tree-guided logistic regression on multiple physiologic data signals in order to detect whether or not artifacts were present on any of the signals. The data was preprocessed by abstracting features in the raw data such as moving average, median, best fit linear regression slope, standard deviation, maximum, minimum, range. Each value was calculated over a sliding window of three-minutes, five-minutes and ten-minutes and fed as inputs to the C4.5 decision tree learning algorithm. Their results were shown to be promising as the receiver operating characteristics (ROC) curve area was at least 85% for successfully detecting artifacts in the heart rate, blood pressure, and CO₂ signals [Tsien et al., 2000]. The researchers later went on to collect an annotated data set recorded by a trained human observer and put it through a decision tree induction system as well as a neural network classifier system to detect true alarms in the systolic blood pressure signal. The decision tree system achieved an area under the ROC curve of 94.34% while the neural network achieved an ROC curve area of 98.98% [Tsien, 2000a]. The researchers recognise that there is a temporal aspect which differentiates true alarms to the fleeting nature of false alarms, and chose their

feature set as eight statistical measures for three different time intervals on five signals. While this is a multi-signal approach that is taking into account the behavior of signals over a period of time, the reasons for choosing many of the parameters such as the number of time intervals, is completely arbitrary. The arbitrary nature of feature selection may be suitable for machine learning algorithms, but they do not translate well as clinical decision support tools for anything more complex than simple artefact detection.

Mylrea et al. (1993) discuss the potential of using neural networks to improve pattern recognition to reduce the number of false alarms in anesthesia-related events. They compare rule-based analysis to pattern recognition methods such as neural networks and concluded that for complex medical problem solving neural networks would work better because they are able to assimilate data and provide meaningful alarms on a real-time basis without first analyzing and summarizing the data [Mylrea et al., 1993]. The neural network implemented in this study used 25 features from three analog waveform signals to detect when critical events have occurred. The issue then is their definition of complex medical problem solving. While pattern recognition methods may be more suitable for finding anomalies or critical events, when it comes to analysing the sequence of patterns in multiple signals to classify clinical conditions such methods introduce far too much uncertainty when trying to define the relationships between characteristics of multiple signals over time as compared to rule-based expert systems.

One of the more widely used algorithms used in analysing physiological data is fuzzy logic. It has been used as far back as 1978 to incorporate the “experience” of a human process operator in the design of a controller. Using a set of linguistic rules to describe the control strategy, an algorithm is constructed with the words used defined as fuzzy sets. This approach advocates the possibility of implementing “rule of thumb” experience, intuition and heuristics [Kickert & Mamdani, 1978]. It was

originally used predominantly in industrial control, consumer electronics and robotics. In biomedical engineering, it has been applied in controlling pacemaker rates in cardiac support systems [Sugiura et al., 2009] and monitoring the dynamics of a left ventricular assist device [Yoshizawa, 1992]. Becker et al. devised a fuzzy controller to evaluate the interdependencies of a patient's vital signs in order to replace simple threshold alarms to intelligent alarms. They acquired the knowledge base from 14 cardiac anaesthetists and formulated a set of state variables and membership functions to come up with 188 fuzzy rules [Becker et al., 1994]. Zong et al. used fuzzy logic to model the relationship between arterial blood pressure and ECG to reduce the number of false alarms in the ICU [Zong et al., 1999]. Mirza et al. developed a fuzzy controller for detecting critical events during anesthesia administration. They formulated membership functions for heart rate, blood pressure and pulse volume and created seven rules that act on the fuzzy sets to determine the level of alarm/warning necessary [Mirza et al., 2010]. While these methods show that a close match to human experts in detecting events is possible, it also demonstrates the challenge in translating such systems to the bedside for routine clinical use. The fuzzy logic systems excel at characterizing specific signals but as a tool for classifying complex conditions their clinical implementation has been minimal.

While there are clearly some methods that more popular than others, there has been much research activity in applying many other machine learning and classification techniques to physiological data. Bloom noted however, that it is nearly impossible to interpret automatically collected data to produce intelligent alarms and identify particular conditions without identifying the specific clinical context in which the data are obtained. To identify such changes in context, he used cluster analysis, discriminant analysis and statistical predictors on electrocardiogram (ECG) data [Bloom, 1993]. Gather et al. studied the correlation between physiological variables to detect known association patterns in haemodynamic time series data. They employed statistical graphical models based on partial coherences between different signals. It is a unique approach and showed success in characterizing

particular clinical states with particular partial correlation models [Gather et al., 2002]. Thommandram et al. used the k-nearest neighbours algorithm (kNN) classification of respiratory time series data to determine whether certain one minute epochs contained apnoea or not. The area under the ROC curve was 96.04% [Thommandram et al., 2013a]. The commonality in these approaches is the heavy dependence on black box modeling systems where clinical context regarding patient condition is not a factor in the algorithm. This makes results difficult to defend at the bedside or as a clinical tool no matter how well they seem to perform on historical datasets.

Very few studies on frameworks for real-time multi-signal classification of conditions were found in the literature. Zhang trained classification trees and neural networks for individual patients and suggested that the trained algorithms could then work prospectively in real-time [Zhang, 2003]. This does not meet the real-time standards set in this thesis where online learning and classifications need to happen simultaneously. Fried et al. compared several methodologies designed for online detection of trends for classification of patient state, but found that not one single statistical methodology could model all the patterns in physiological time series data [Fried et al., 2001]. This comparison of online detection models makes it evident that a more complex framework for developing models for automating existing, functioning rules is required. Such a framework would have to combine simple statistical methods with pattern specific models to accomplish classification of complex conditions.

Most of the studies described above perform analysis retrospectively. This allows them to train their models on a training set of data and test it on another portion of data. While this is sound methodology, it does not translate easily to online or real-time analysis. Performing online learning and real-time classifications requires the use of different approaches. Based on the limitations in current

systems described in this literature review, a set of requirements for data acquisition and analytics are formulated for achieving the ability to correlate and classify physiological data streams in real-time.

2.3 - Requirements

2.3.1 - Data Acquisition

For an intelligent monitoring system, it is crucial to be able to process as many physiological signals as possible at a high sampling rate. This will ensure that algorithms will not have to deal with data loss and be forced to make assumptions for the gaps in time for low frequency sampled data.

For waveform data such as ECG, it is important to have as high a sampling rate as possible. For most modern monitors, this is 500 Hz. This high resolution allows for analysis into small deformations in the waveform and subtle anomaly detection. For other waveforms such as respiratory impedance (RI), blood pressure, and blood oxygen saturation the sampling rate does not need to be as high as features and anomalies in these signals are not as subtle as ECG and can be discerned with a sampling rate as low as 50 Hz.

For numerical data streams, it is vital to break past the one minute sampling frequency. Too much information is lost when the raw data consists of an average value for an entire minute. This makes trend detection more difficult and the analysis more speculative. Numerical streams such as HR, SpO₂, blood pressure (BP) and many others need to be collected at a rate of 1 Hz in order to perform useful analytics with them.

It has been observed that many research endeavours in developing analytics algorithms usually also involves developing the system to interface to specific patient monitors as well. This unnecessarily complicates the project and detracts from the focus of the research, which is in the analysis. Thus far it has been an inevitable task as different machines have different data export capabilities and interface

protocols. Although, when designing a framework that supports a wide range of users, the data interface with the monitors needs to be separated from the algorithm development environment.

2.3.2 - Real-Time Analytics

The data acquisition is only the first step in the process of creating an intelligent monitoring system. Arguably, the most critical component of such a system is the analytics. As complex as the bedside monitors are in terms of the sensors onboard the alerts are simplistic and most only detect events in single signals. The nature of real clinical conditions makes it very necessary to move beyond single signal event detectors and develop algorithms that can correlate multiple physiological data streams in order to classify conditions.

Another obstacle that has been a common theme in the literature is the difficulty of taking a system that is trained retrospectively and then being able to run it on live data in real-time. One of the issues operating in real-time is synchronising different streams so that the signals are always in phase. When running an algorithm retrospectively, an almost universal step was the preprocessing of data and annotations to match up the timestamps so that the system could be trained properly. A method needs to be developed to handle the time variability of incoming data in a real-time and robust way.

Considerations must be taken for adapting the research methods to extend beyond the laboratory environment and into the hospital to the bedside. This involves making analysis processes transparent and understandable by clinicians. To accomplish this, there needs to be less dominance of black box modeling systems and more input from medical professionals in a rule based expert system. There are advantages to using black box methods but as the literature review above shows, a great deal of resistance is met when implementing the work commercially [McGregor et al., 2009]. A new balance between the different approaches is required.

It is clear that to support real-time analysis of multiple physiological streams simultaneously there is a need for a robust data acquisition system as well as a sophisticated software environment to perform complex analytics for the cross correlation of features required for real-time monitoring associated with many clinical conditions. The next section details the Artemis platform that met this need by providing a platform within which the instantiation of the framework of this thesis could be demonstrated.

2.4 - Artemis framework for real-time data capture and analytics

The platform used in this research, known as Artemis [McGregor et al., 2011], is a multidimensional, online health analytics framework designed to support intensive care environments. The framework supports the acquisition, collection, transmission, real-time processing, storage and retrospective analysis on wave form and numeric physiological data streams combined with supporting clinical information including laboratory results and observations. A diagram of the Artemis framework is shown in Figure 1.

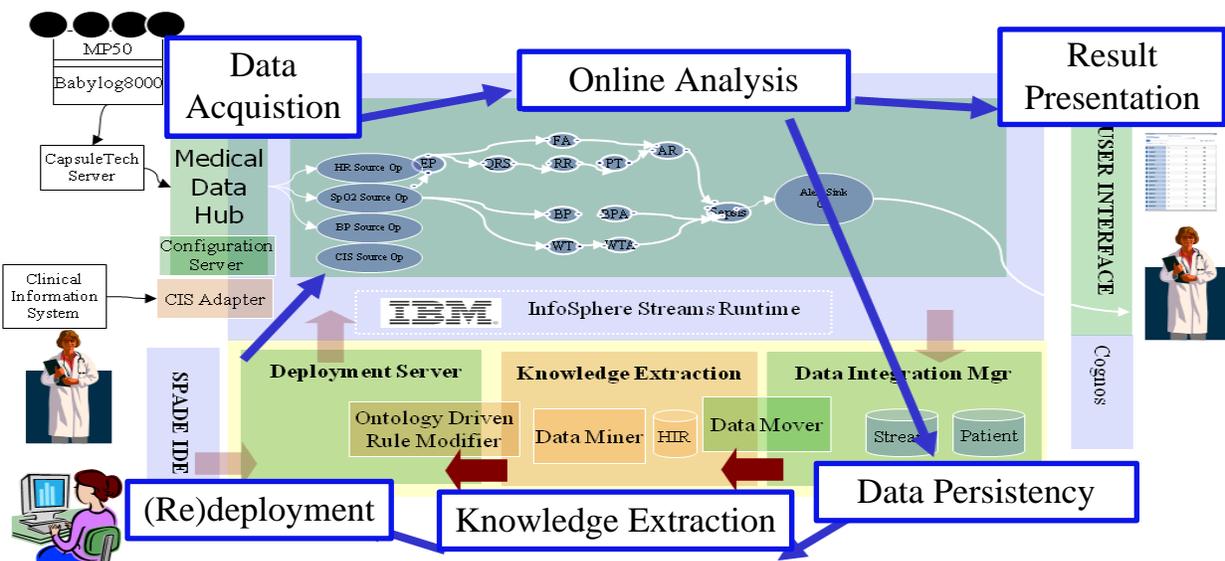


Figure 1. High level architecture [McGregor et al., 2011]

The Artemis framework relies on the acquisition of physiological data from existing bedside medical devices contained within the NICU. These devices, which are already enabled to output their data for collection, can be connected to via an Ethernet and/or serial port.

Within the current Artemis framework, an existing medical software data collection system supplied by CapsuleTech is used. This system connects the outputs from the monitors to the input data stream of Artemis where they are parsed in real-time. The data collected by Artemis varies based on the medical monitors being used and their capabilities. In general, the data streams collected include but are not limited to heart rate (HR) at 1 Hz, respiratory impedance (RI) waveform at 62.5 Hz, blood oxygen saturation (SpO₂) at 1 Hz, multiple blood pressure signals such as mean blood pressure (MBP) and systolic blood pressure (SBP) and diastolic blood pressure (DBP) at 1 Hz, electrocardiogram (ECG) at 500 Hz, respiratory rate (RR) at 1 Hz, and carbon dioxide (CO₂) waveform at 62.5 Hz.

The real-time stream processing component of Artemis is built upon IBM InfoSphere Streams (Streams), which is IBM's scalable middleware component for handling multiple streams of high volume, high rate data. In Streams, real-time algorithms for processing the data are constructed as sets of operators interconnected as a graph. Each operator takes one or more data streams as input and produces one or more output streams. Complex analysis is enabled through the correlation of the behaviours shown by the data across multiple physiological streams. As a result, this approach can provide support for multiple concurrent real-time clinical management needs. Through the provision of stream operators, diagnostic algorithms can be developed with a focus on core evidence-based hypothesis design.

Additional features of Artemis include the persistent storage of the physiological data streams together with the derived analytics. This stored data can be queried to support real-time clinical management and the development of new diagnostic algorithms as part of clinical research studies.

Streams uses an application specific programming language known as the Streams Processing Language (SPL) to enable stream-based applications. In this environment, the applications, commonly called “graphs”, are deployed which run operators to process the streams in parallel to each other as they enter the system. Each operator represents a step in the logic of an SPL graph and communicates with other operators through packets called tuples using a simple message format. This breakdown of a program into steps and operators is what allows Streams to distribute processing over multiple computing elements. This system is highly scalable and effective at handling large amounts of data.

The Artemis framework has been deployed in the Neonatal Intensive care Unit (NICU) of The Hospital for Sick Children in Toronto, Ontario, Canada. It has been capturing physiological data streams and patient health records since August 2009 [Blount et al., 2010]. A cloud based version of Artemis, known as Artemis Cloud was deployed at Women and Infants Hospital, Providence Rhode Island in 2010 (McGregor et al., 2011). Artemis and Artemis Cloud are currently being tested by the NICU, Children’s Hospital of Fudan University since 2012 (Kamaleswaran et al., 2013). In addition to capturing data, the environment has also been used for a clinical research study for identifying late onset neonatal sepsis (LONS) using heart rate variability analysis [McGregor et al., 2012].

The above observations highlight the decision to use Artemis as the chosen platform in the proposed framework to collect data from patient monitors and serve it to a real-time engine for analytics. The data acquisition portion of Artemis handles the interface protocols with various manufacturers and allows researchers to focus on developing algorithms. Artemis also contains the InfoSphere Streams analytics engine which enables the processing of signals in a real-time fashion. Its design within the Artemis framework allows for data to be collected remotely as well. This makes performing real-time analysis with data from multiple sites much more feasible.

Chapter 3 - Case Study: Classifying Spells in the NICU

To demonstrate the capability of the proposed real-time data stream processing methods as proposed in this thesis, I chose to implement algorithms that can classify neonatal spells in the neonatal intensive care unit (NICU). The NICU is an intensive care unit that specializes in the care of premature and critically ill mature newborn infants. Due to the very sensitive nature of the patients, the level of monitoring in a NICU is much higher than with regular medical care. In addition, patients in the NICU are left to sleep for longer periods of time than other ICUs as interventions are bundled together, where possible, to occur when the babies are awake. This reduces the amount of artefacts in the signals as compared to other populations. From a research standpoint, this translates to access to high quality physiological data to base the algorithm development around.

The NICU provides care for premature infants and critically ill mature newborn infants born after 37 weeks gestation. A baby is classified as premature if it is less than 37 weeks gestational age. Premies, as they are also called, are at a stage in their life where they are incredibly vulnerable and require the most care to develop and grow properly. Generally, the earlier the baby is born, the higher the risk of complications. Also, the weight of the baby at birth is an important factor in determining the risks. Some of the health problems preterm babies face after birth include [“Premature babies”, n.d.]:

Respiratory distress syndrome – This is a breathing problem that is most common in babies born before 34 weeks of pregnancy. These babies do not have surfactant, a protein that that stabilises the small air sacs in the lungs and prevents them from collapsing at the end of expiration.

Infections – premature babies are at increased risk for infection because of immaturity of their immune systems and the use of invasive intensive care technologies.

Intraventricular hemorrhage – This is bleeding within the fluid filled spaces in the center of the brain.

Necrotising enterocolitis – an ischaemic injury of the intestinal mucosa of the gastrointestinal tract.

Apnoea – This is a pause in breathing for 20 seconds or more.

Retinopathy of prematurity – This is vasoproliferative disorder of the immature retina that can lead to vision impairment or blindness.

Some of the long-term health care problems premature babies can have include chronic lung disease, cerebral palsy, and vision and hearing loss. The risk of developing these conditions is heavily dependent on the neonate's degree of immaturity, size, and the occurrence of infection at any time and/or the complications of immaturity.

For the case study implemented in this thesis the focus was on neonatal apnoea, also called spells.

3.1 - Neonatal spells

The term neonatal spell is frequently used in the NICU as a synonym for apnoea, however it encompasses other physiological events that also trigger a monitor alarm such as a fall in heart rate or a decrease in blood oxygen saturation. Apnoea is defined by the American Academy of Pediatrics as a pause in breathing for more than twenty seconds or a pause in breathing of any duration that is associated with cyanosis, pallor, bradycardia, or marked hypotonia [“AAP”, 2003]. The alarm systems on modern monitors detect numeric threshold breaches and therefore, they are not capable of alarming for instances of this second more complex definition of apnoea. As a result nurses in the NICU are relied on to clinically assess if a baby is apnoeic. It is not practical for nurses to observe a baby continuously so the term spell has come into use to document events where the monitor alarms but a cessation of breathing was not observed.

In his recent research within the Artemis team, Pugh has determined that spells may be classified as central apnoea, vagal apnoea, obstructive apnoea, obstructive central apnoea, central

obstructive apnoea, isolated blood oxygenation desaturation or isolated bradycardia [Pugh et al., 2013a]. The only accurate way to recognise and classify these spells is with polysomnography, which uses multiple sensors and continuous clinical observation to determine the type of spell that is occurring. The pattern of change of the recorded variables is classified using coded algorithms based on well-defined physiological principles [“AASM”, 2007]. Polysomnography is not practical in the NICU environment because it often requires relocating the baby and having a sleep technician monitor multiple channels of bio-signals requiring many electrode attachments to patients as well as performing intrusive operations to measure some physiological variables. It is a resource intensive operation and as a result, it is reserved only for the most difficult cases to diagnose. Figure 2 shows an example of a timeline of events that lead up to a spell that would go undetected by conventional threshold techniques. It starts with all physiological signals in a stable condition. This is shown as the baseline phase. Then the respiratory impedance wave, the blue line, shows a pause in breathing in the Resp pause phase. This is then followed by a drop in blood oxygen saturation. At this point, a spell can be defined. However, since none of the signals breached its individual alarm thresholds yet, the bedside monitor would not raise an alert.

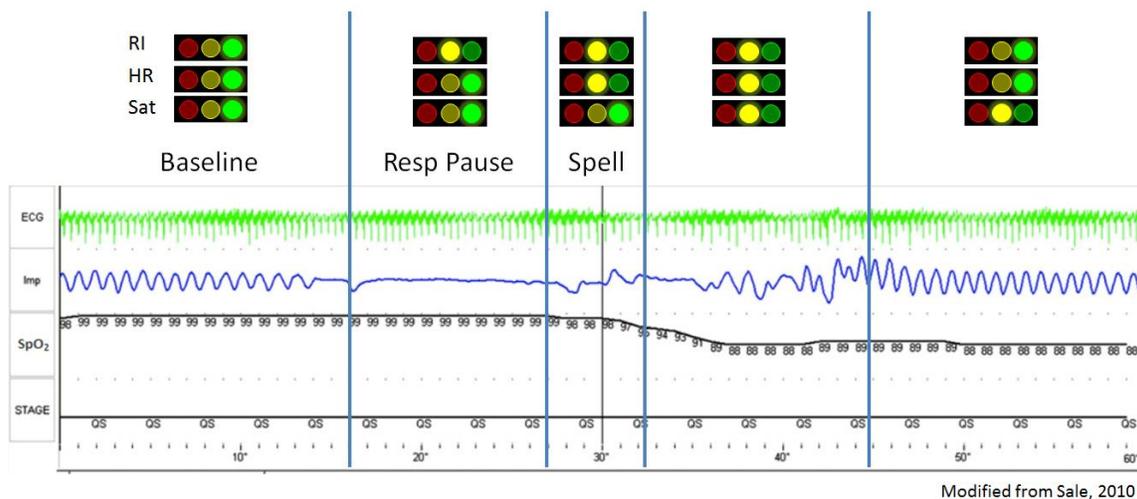


Figure 2. Structure of a spell

This is not the first time automated detection of apnoea has been attempted. There is quite a history in intelligent systems for the classification of apnoea and it varies widely in the number and types of physiological signals used, as well as the type of system identification techniques.

Most automated detection schemes involve black box modeling systems such as artificial neural networks. While some of these systems perform well, their adoption in practice is very low, possibly because of the lack of traceable rules inherent to black box systems [McGregor et al., 2011]. Obstructive sleep apnoea and hypopnoea can be automatically detected by extracting wavelet-based features in electrocardiogram (ECG) recordings and finding patterns using a feed forward neural network [Khandoker et al., 2009]. Schluter et al. developed a decision tree classifier towards an approach for automatic sleep stage scoring and apnoea detection by using rules formulated for sleep technicians for manual scoring. They used derivative dynamic time warping (DDTW) to perform pattern matching in an attempt to detect the same shapes in the signal as a sleep technician would detect [Schluter et al, 2010]. It has been hypothesized that simple features can still be sufficient for accurate detection if extracted from a variety of signals. Belal et al. used a combination of HR, RR and SpO₂ features and fed them to a neural network to find correlations [Belal et al., 2011]. All of these schemes don't really make use of the logic by which human experts detect features in signals. They simply use the results and train pattern matching algorithms. This is why these approaches remain in the realm of novel research as supposed to transitioning to the bedside.

Apart from the ECG, the RI waveform is believed to be very valuable for apnoea detection as it is directly correlated to breathing effort. Lee et al. recognized that RI signals also contain fluctuations caused by the beating of the heart. These fluctuations are misinterpreted as breaths by bedside monitors and cause inaccurate alarms [Lee et al., 2012]. They developed a cardiac filter that involved resampling the RI at the frequency of the ECG to reduce the effect of the periodic fluctuations caused by

the heart. Some research has moved in a different direction, taking the respiratory measurement from locations other than the chest. Ansari et al. discovered that it is possible to extract a reliable RR using signal processing from impedance measured across the arm [Ansari et al., 2009]. While this method resolves the cardiac effect, it is prone to movement artefacts. This research takes a different approach for filtering this artefact by incorporating a threshold cut-off that ensures only major peaks caused by breathing are detected and minor peaks caused by heart rate influence are ignored [Thommandram et al., 2013b].

In order to develop an accurate means to detect and classify spells and have it adopted for clinical practice, this thesis proposes rule-based temporal analysis methods in a hierarchical model. Each individual physiological signal will be processed independently initially to find any significant events. The duration and order of the events from multiple signals will be processed by the next level in the hierarchy in order to output a spells classification.

The approach within this thesis is unique in its modular architecture. By using Streams, algorithms can be developed in a distributed manner and linked together to perform complex classifications. Independent low level alerts can be channeled to any algorithm that needs them. This also allows us to expand or broaden the analysis as required. Spells and many other conditions can be classified by automating the rules used by human experts because of the main contribution of this thesis, which is the ability to build complex rule sets from the ground up by defining physiological events and building their temporal correlations in a hierarchical approach.

Chapter 4 - Methodology

This chapter presents the methodology for the framework that has been designed to support the real-time classification of physiological streams for critical care monitoring. Over the course of the literature review in the previous chapter, there were key observations made towards the technical requirements of developing such a system. This framework as a contribution of this thesis addresses these requirements. The proposed framework is a library of operators programmed in SPL supporting analysis of physiological signals as necessary for the direct support and association with clinical conditions. The system that instantiates the framework was developed using InfoSphere Streams Studio, an SPL development environment created by IBM [Ballard et al., 2012]. In this section, the designed framework for supporting real-time classifications of physiological signals is described in detail. A diagram of the overall architecture of the framework is shown in Figure 3 below.

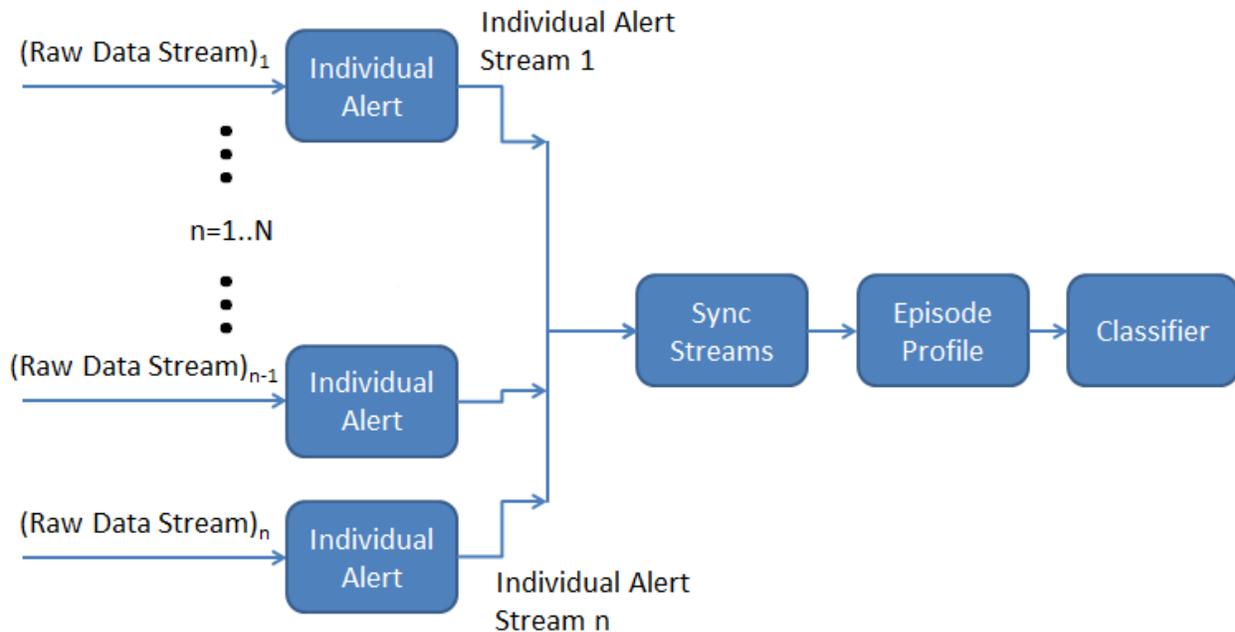


Figure 3. Overall architecture of framework

The first element within this framework is each raw physiological data stream. Currently within Artemis the raw streams are one of two types, numerical periodic data and waveform data. But the framework extends to any type of input data stream. These raw data streams are then processed through individual alert operators. These are any of multiple algorithms that are coded to detect different kinds of anomalies or features in a specific data stream. The individual alerts represent the first step of the hierarchy of classifying complex conditions. The framework supports any number of individual alerts to be instantiated. As well, multiple individual alerts can be fed the same raw data stream. This enables various features to be detected on a single data stream simultaneously. Once all individual signal feature streams have been configured, the next step in the framework is to synchronise the streams. This operator reads the timestamps for every individual alert stream and acts as a buffer while outputting data points from all the streams in one message with one synchronised timestamp. The next step is to determine an episode profile for any events that may be occurring. This operator provides the ability to observe correlations between events in multiple streams. It builds a buffer of synchronised alert messages starting with the first message where any alert was raised and ending with the last message where all alerts have returned to normal. This buffer is the episode profile and is sent to the final block of the framework that is the classifier. In this block, any rule based logic can be implemented to analyse the episode profile and determine a label for the whole event. In the following subsections, each block in the framework is described in detail.

4.1 - Events in individual streams

One of the first steps in a complex classification framework is to detect anomalies or features in individual physiological streams. The individual events will be processed for correlation downstream in the logic, but the detection of events in a specific stream is an independent process. Any number of individual feature streams can be configured for a source raw stream. For example, if a classification rule required the detection of a dramatic fall in heart rate (HR) and also needed to know the value of the

heart rate variability, then two individual feature streams or individual alert streams would be configured on the raw physiological stream of HR. This forms the base for a highly extendable event detection system as various specific event rules can be packaged as tools in the framework to be applied to algorithms as needed. Figure 4 below shows some of the developed individual feature detector modules that are now part of the framework's toolkit. The decision to implement these specific individual alert modules was driven by the case study described in the next chapter. Any future classification algorithm that uses the proposed framework will have access to the existing modules for generating individual alerts for various data types, as well as the capability to program new ones.

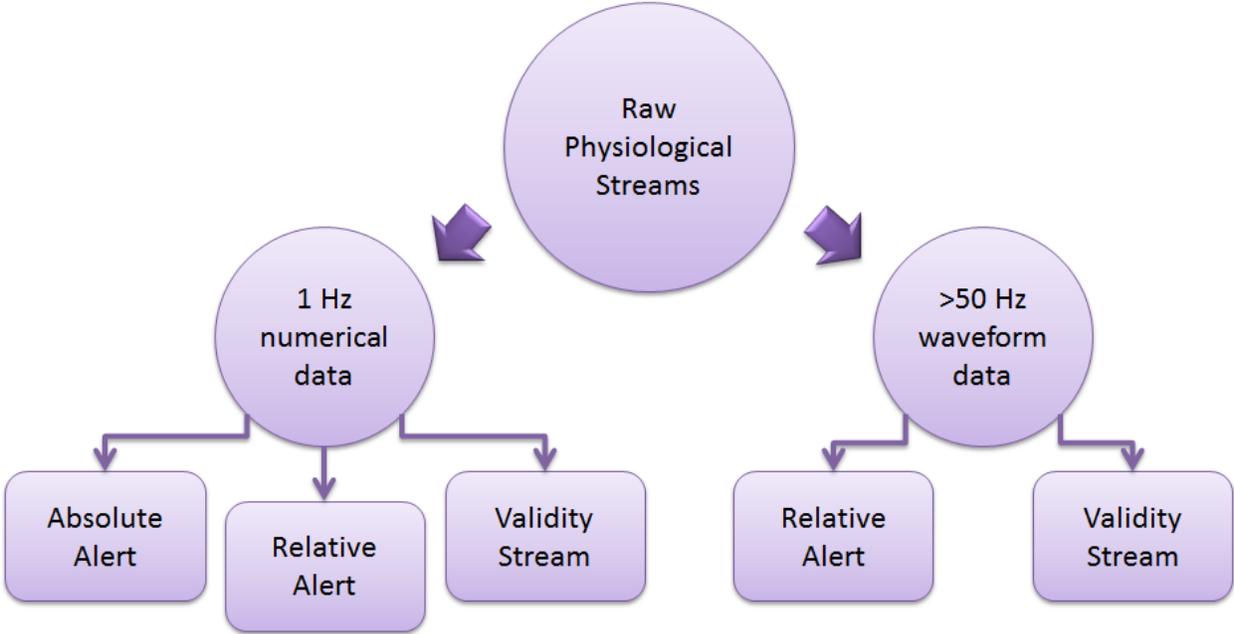


Figure 4. Individual alert modules that currently make up the framework's toolkit

4.1.1 - Numerical data streams

For numerical data streams, several individual event detectors have been developed. These event detectors are independent in the sense that no context is taken with respect to other signals. The context is provided during the correlation and classification phase downstream in the algorithm.

Absolute Alert

The first event detector to be implemented is the absolute threshold breach detector. This module analyses a signal to see if the value is within set parameters. It has an output port on which it sends the absolute alert value stream. The alert value is zero if the source data is within threshold conditions and one if a threshold has been breached.

The absolute threshold breach detection is very simple and reflects what is currently used in most bedside monitors. The logic of the absolute threshold detector is shown in Figure 5 below.

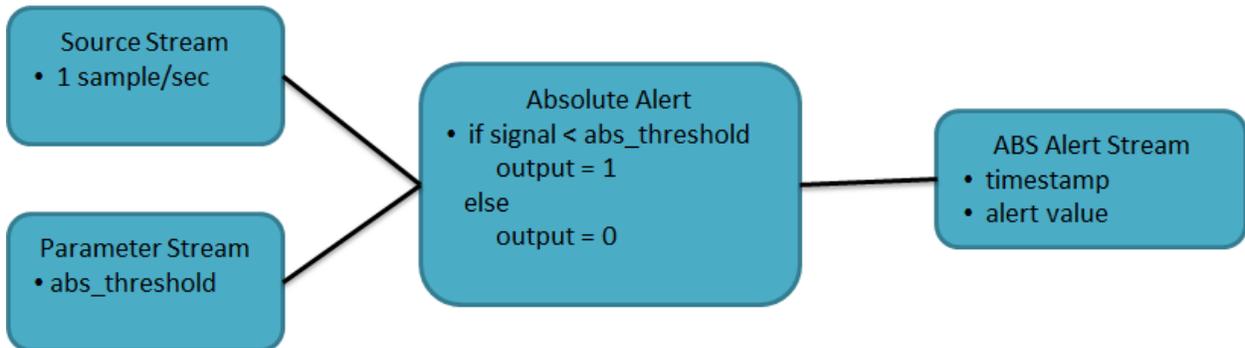


Figure 5. Logic of absolute alert stream

Relative Alert

A more useful approach is to detect relative changes in the signal such as drastic falls or drastic rises. This is a more complex alert and requires much more computation. To do this, a sliding baseline is

determined on an ongoing basis as data enters the module. This sliding baseline is similar to a moving average of the signal and is used to determine the patient's current "normal" state. Every new value of the physiological stream is then compared to this baseline and if there is a change of more than a certain percentage, then an event is noted in the module's output stream. Figure 6 below demonstrates the detection algorithm for when values are holding within the limits of the baseline.

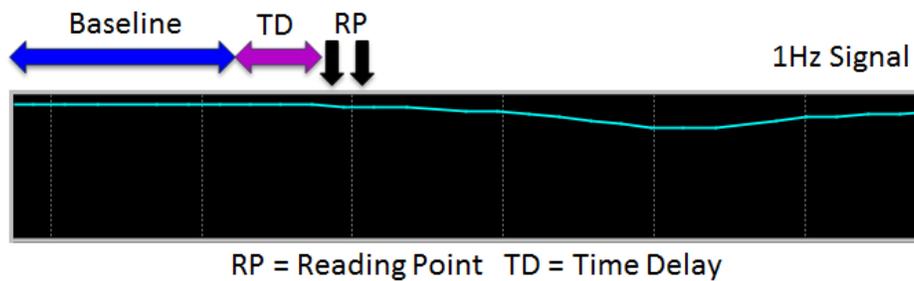


Figure 6. Detecting relative change

Defining when the alert value should be zero and when it should be one is not as simple as it is with the absolute threshold. This is because if an alert is raised when the reading point is out of range with the baseline, and the baseline continues recalculating, eventually the baseline will have shifted closer to the value the reading point was at when the alert was raised. To properly determine the exit conditions for this alert, more sophisticated logic is required. The normal state prior to the alert needs to be preserved or remembered to serve as a basis for comparison. When an alert is raised, the baseline is locked and acts as a reference state to which the patient needs to recover in order for the alert to terminate. Figure 7 below shows a baseline locked due to a detected event.

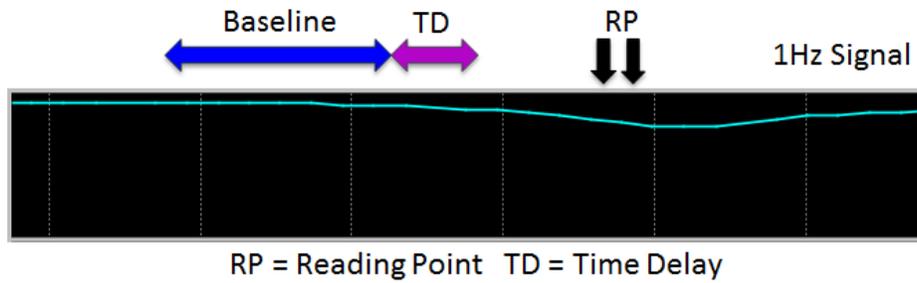


Figure 7. Locking baseline

This covers most of the cases where the signal deviates from the baseline and returns to baseline, but the case where the signal never returns is not handled. Since the characteristics of the physiological data are always changing, it is entirely possible that the signal stabilizes at a different, but still normal level. The algorithm monitors this case by maintaining another baseline with a longer time window that doesn't lock when an event starts. If the signal stabilizes at a value for a set duration, then the event is considered finished and a new normal is obtained. The logic of the relative alert is shown in Figure 8 below.

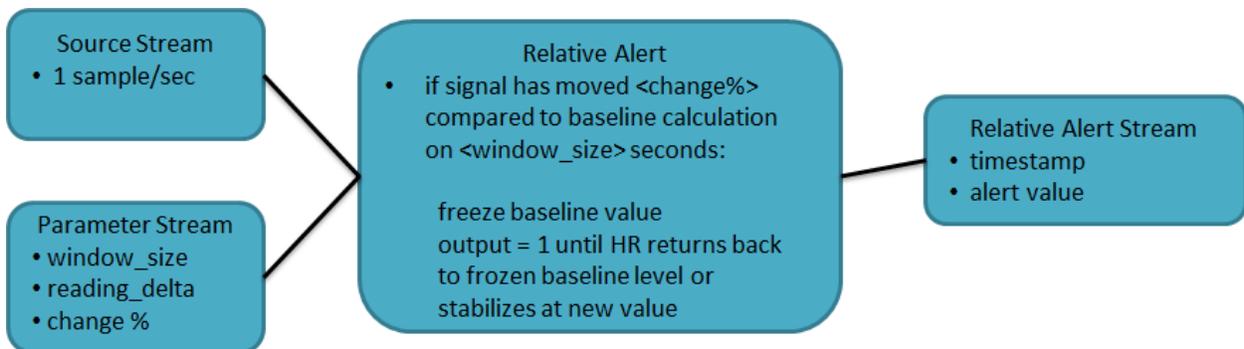


Figure 8. Logic of relative alert stream

4.1.2 - Waveform data

For waveform data the desired features to detect are very different. Most of the interpretations of waveform data by a human expert use patterns in the peaks and valleys in the waveforms. Since the framework is based on enabling rule-based classification systems, it is vital to provide the peaks and valleys as features for use in classification rules. To facilitate this, a peak detection algorithm has been coded into a Streams operator that takes raw signal values as input.

Peak Detection

There are many algorithms for finding peaks in time series data, but the approach selected is simple enough to be performed on real-time data but robust enough in its performance. The popular zero-derivative method is not used because the noise inherent in this real-life signal produces many accidental zero-crossings. Usually to get rid of these false detections, some kind of smoothing is applied with a low pass filter. But this changes the original signal, and as discovered is really not necessary. The detection of peaks in this case is not really the same as finding maxima and minima in the mathematical sense; the system just has to identify the peaks that seem obvious to the human. To do this, a value for maximum and minimum are kept updated by comparing every new value. When searching for peaks, the new value is checked to see if it is less than the stored maximum by a certain difference. If it is, then it is concluded that the signal has changed direction and a search for the next valley is started. When searching for valleys, the new value is checked to see if it is greater than the stored minimum by a certain difference. If it is, it is concluded that a valley was reached and the signal changed direction and a search for the next peak is started. This process continues indefinitely on incoming data. Since the source signal is physiological in nature, it is expected that there will be some noise so a simple threshold cut-off parameter is set to ignore minor peaks and valleys and only detect major ones. When a peak or valley is detected a tuple is output to the Maximums and Minimums streams respectively. These two streams allow the development of algorithms that use the locations and amplitudes of peaks and values

in waveform data. A flow diagram describing the logic of the peak detection algorithm is shown in Figure 9.

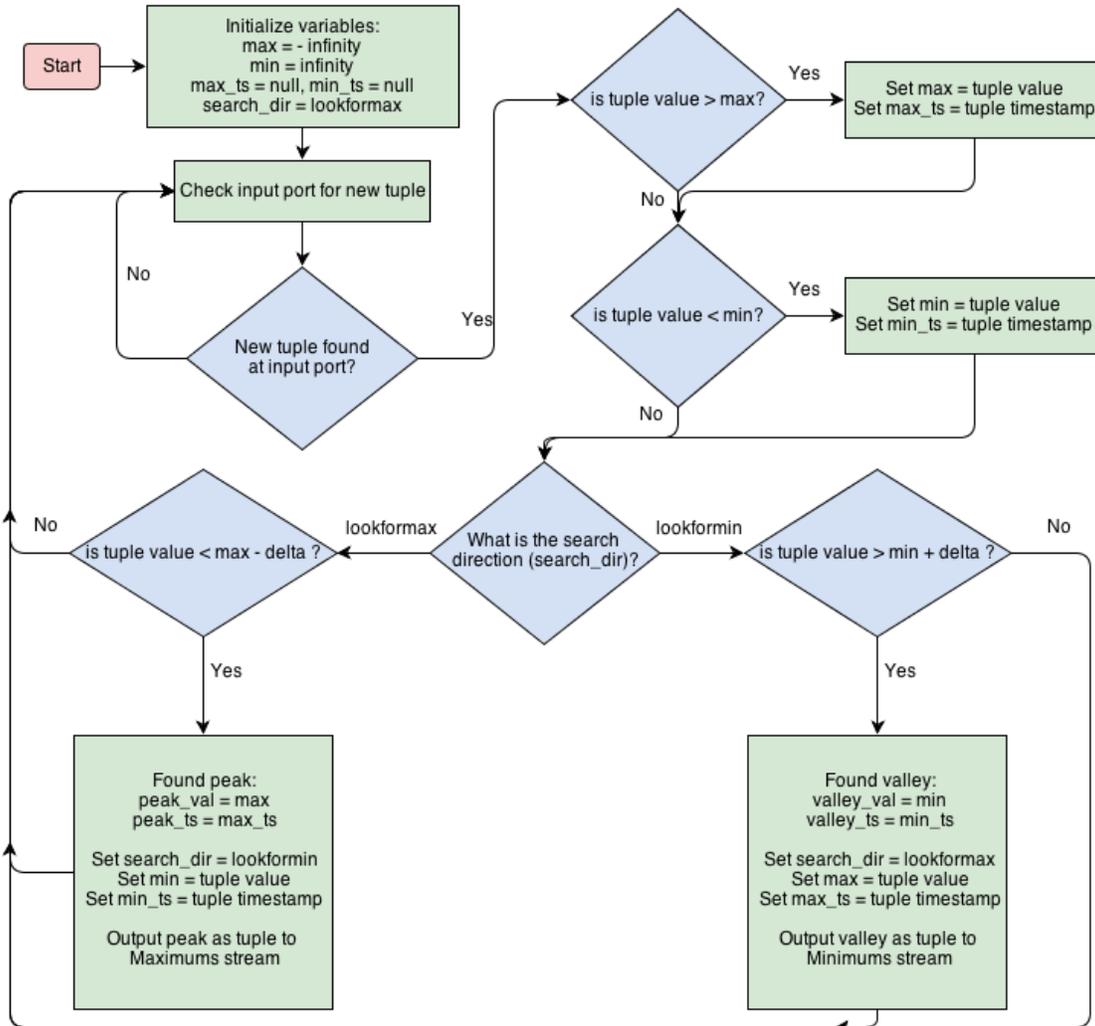


Figure 9. Flow diagram describing peak detection algorithm

However, there is a case where the Maximums and Minimums streams are not enough. The trouble is in the fact that the peak detection operator only outputs a tuple to these streams only when a peak or valley has occurred. It does not emit any tuples in between peaks. That means that to find out how much time has elapsed since the last peak, you would have to wait for another peak to form because only then would the operator output another tuple. This is because the way Streams operators

behave, logic code is only executed when a tuple arrives at an input port. For a lot of applications, simply knowing the time and amplitude of peaks is enough to perform a trend analysis. However there are cases where knowing how much time has elapsed since the last peak is crucial. These are usually when a certain peak to peak time is expected and an alert needs to be raised because that time has passed without a new peak. This live peak location stream is designed to output a tuple for every tuple on its input port. Along with the timestamp, the location value is one if there is a peak at that timestamp or zero otherwise. A comparison of the maxes operator output stream and the live peak locations output stream is shown in the Figure 10. Based on the live peak locations, an individual alert can be designed for waveforms to detect relevant events. For example, an operator can be designed to output a relative alert if too much time has elapsed without a new peak.

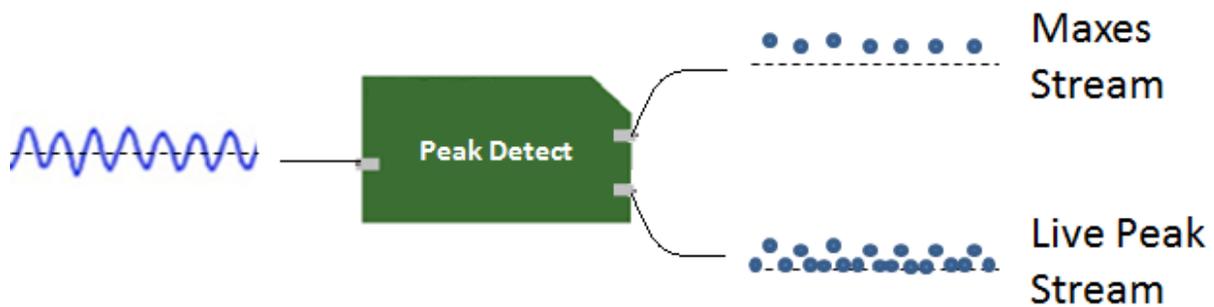


Figure 10. Comparing Maxes and Live Peak output streams from peak detection algorithm

4.1.3 - Artefact Detection

Another feature added in the framework is the ability to detect and filter artefacts in the raw signal. There are times when a sensor is disconnected but the monitor still measures a signal and sends it to the computer. In these cases, the signal value is a nonsensical number. For example, if the SpO₂ sensor is still plugged into the monitor but fails to make proper contact with the skin, the values obtained for the

signal will change from a number 0-100 to exactly 8,388,607 for example on Philips Intellivue series devices. To allow classification operators to have access to this piece of information, a set of signal validity operators were developed for both numerical data types as well as waveforms. Currently, the only artefact being detected is the presence of 8,388,607 values. As input, the operator takes the raw data. For every input tuple, it produces an output. Figure 11 demonstrates the output of the validity stream as leads off events are detected. As more sensors get integrated into the system, the need for detecting more complex artefacts will be apparent. To this point, the validity module has been developed to output a percentage confidence based on all the artefact blocks within the module. Since the only block employed right now is the leads off detection, the validity module for any signal only outputs either one (100% confidence of validity) if the sensor is properly connected, and zero (0% confidence of validity) if the sensor isn't properly connected to the body.

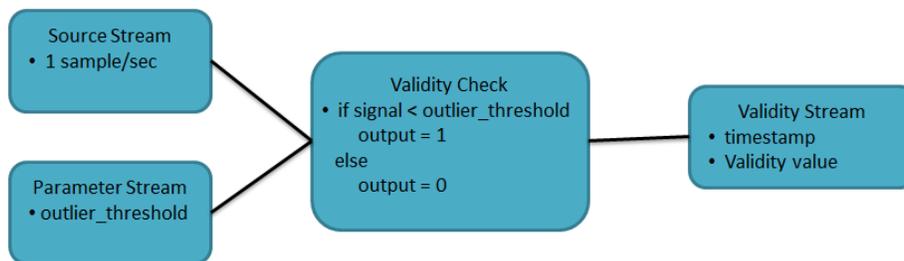


Figure 11. Logic of validity check operator

4.2 - Syncing streams

Once data signals have individual alerts, it is time to correlate the alerts to find meaning and insight into the event as a whole. This is a key step to support classifications because up till now all the processing on a signal was bound to only that signal with no link to any other physiological data. By synchronising all the relevant signals and their validity streams, analysis on which alerts came in what order and when signals recovered back to normal can be done.

Intuitively, syncing multiple signals shouldn't be an issue. After all, the data is streaming in from the machines at the same time so the times of the latest sample of each physiological signal would be very close. But it is not sufficient to rely on real incoming time to sync signals. While the framework is geared towards real-time analysis, by not constraining data timestamps to real-time but rather to use the measurement time allows the framework to be equally suited for running algorithms through data retrospectively. With this design, the framework does not take into account how many real seconds have passed since the last input tuple. It compares the sample's timestamp to the previous timestamp to calculate the time elapsed. This also enables rapid replay of retrospective data for analysis as the computation time is only limited to processing capability. So it would be possible to analyse hours of retrospective data in minutes.

The syncing module of the framework has parameters specifying the different alert streams required to be grouped for classification. The module has to consider that different types of input streams may have different data rates. It down-samples the waveform data to speed of the numeric signals but since it is not down-sampling the raw waveform but rather the relative alert stream of that waveform, temporal features have already been extracted. The sync operator takes in multiple inputs each with a timestamp and alert value or validity value and outputs one stream where each tuple contains a timestamp and an alert value from every input as well as the validity signal values. Figure 12 below shows an example of what the sync module does with inputs from three relative alert streams of varying sampling frequencies. The output of the sync operator represents the state of individual signals at any time and is the starting point for temporal analysis of the data.

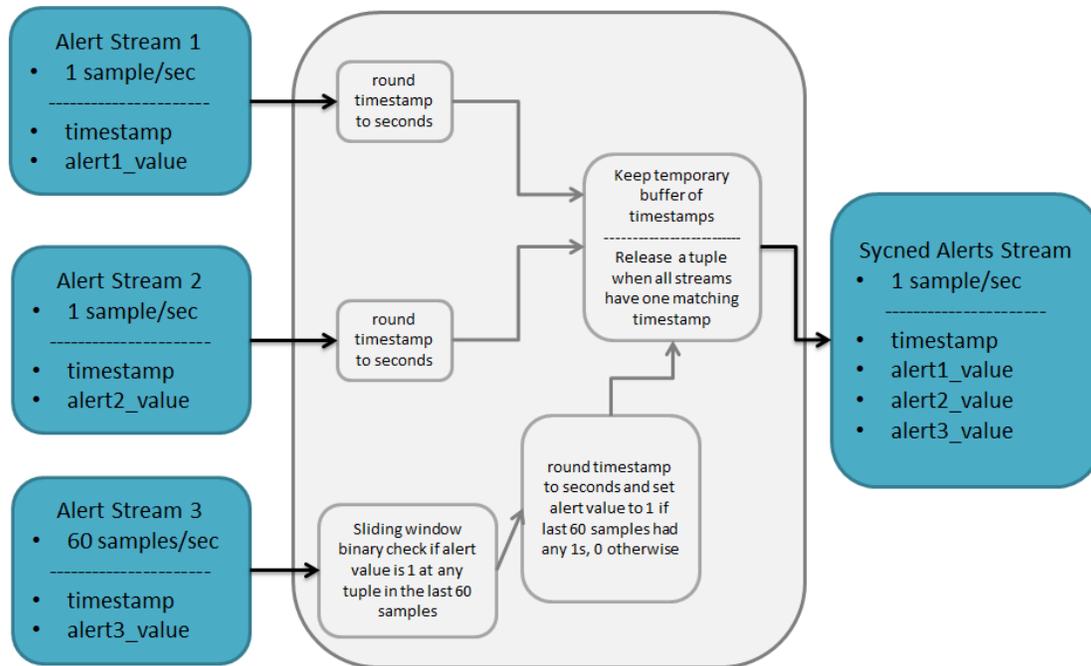


Figure 12. Behavior of sync module with source streams of different sample rates

4.3 - Determining the episode profile

The next step in the pipeline of the framework is the creation of the episode profile. This module monitors the synced signals stream and creates a buffer when any of the individual alerts are thrown. This marks the beginning of an episode. Tuples are added to the buffer until all individual alerts have recovered. This marks the end of an episode. The entire episode buffer is output in a tuple for a rule based classifier to analyse the temporal patterns in the episode profile. Figure 13 shows the logic behind the episode profile generation operator. The episode profile module also has access to the validity streams as they would be one of the features in the synced stream. For any point in an episode if any of the validity streams goes to 0, then the episode is determined to be invalid and it will not be forwarded to the classifier operator to be analysed. In addition to the episode buffer, this operator also calculates some metrics regarding the episode and forwards them to the classifier block to help with the classifications. The metrics include an event id, the start and end time of the event, the duration, and

the number of signals affected. These metrics save the classifier block one scan of the buffer which helps with efficiency.

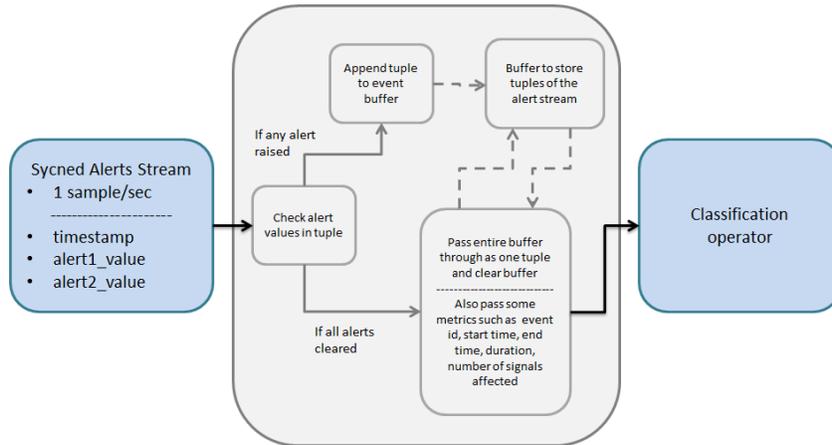


Figure 13. Episode profile generation from synced streams

The final step in the framework pipeline is the classifier block. This module takes the episode buffer containing the profile of an episode and applies rules to determine a classification for the event. The classification usually is based on the sequence of events in the episode. The event buffer is scanned and a code is given to each transition of state of an alert. Then the event code sequence is sorted by timestamp. This results in a temporal view of what transpired in the event by means of showing how alerts in one signal affected alerts in others over time. This sorted event code sequence is then fed to a rule based classifier to assign a label to the event. The classifier will analyse the sequence of events and match them against a knowledge base of event sequence-logic pairs to determine the classification. Figure 14 below show an example of such a classifier analysing a simple episode profile.

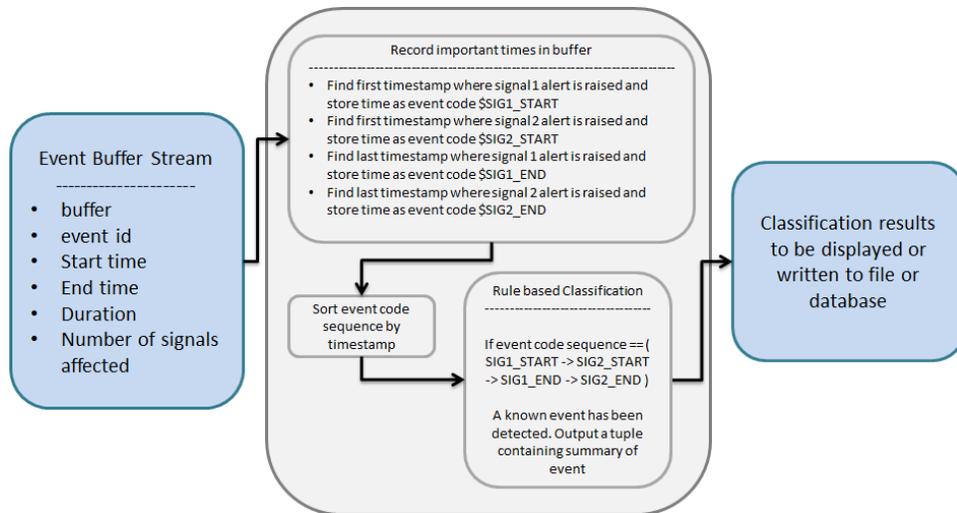


Figure 14. Example classification of event profiles based on simple rules

This rule based classifier is looking for events where signal 1 raises an alert, followed by signal 2, followed by a recovery in signal 1 and finally a recovery in signal 2. It receives a tuple with an episode buffer that matches this pattern. It can now output a tuple with a summary of the event including a label, time of start and time of end. Different classifiers can be written to analyse the same episode profile stream and the classifiers can run in parallel in real-time. Classifiers can also be added into the framework’s library of modules.

4.3.1 - Real-time considerations and utilities

With so many individual alert algorithms running on multiple patients simultaneously, it is very probable that parameters and thresholds need to be adjusted on a patient to patient basis. Streams was designed mainly as an analytics engine and user interfaces are not common components of the system. However, due to the need to change thresholds and other parameters on an ongoing basis, Streams operators were designed to open network sockets and act as a server. Also, an application was developed to display parameters and change them in real-time. Below is a figure of the simple interface application that was created as part of this research.

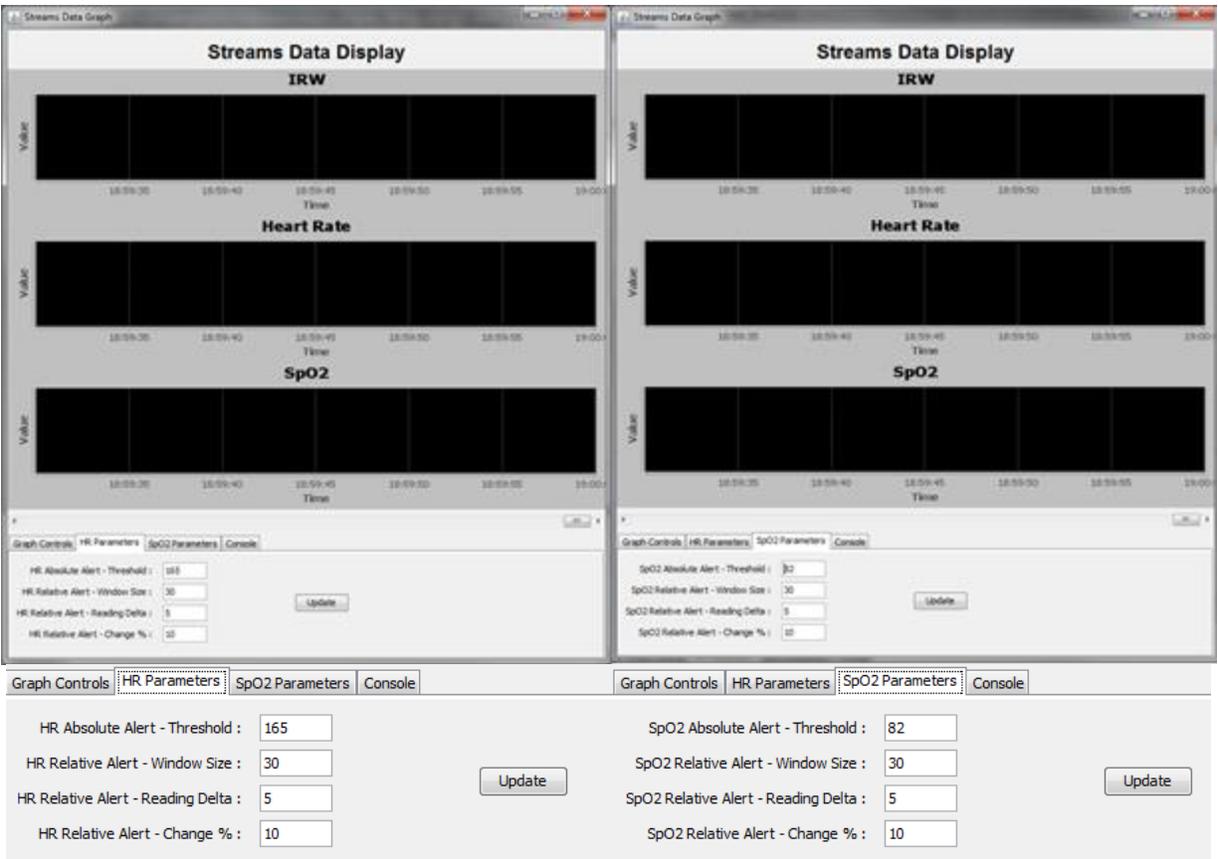


Figure 15. Screenshots of simple interface application to change parameters of multiple running sample SPL graphs

The application is written in Java and communicates over the TCP/IP network to Streams where there is an operator listening on a specific port for commands to change parameters. Currently, only the absolute alert and relative alert operators for numeric data types is supported. The changeable parameters are the absolute threshold, the relative alert baseline window size, the relative alert reading delta, and the relative alert change percentage. But the framework allows for this functionality to be enabled for any new feature detector that is implemented by way of standardized parameter interface protocols.

Chapter 5 - Implementation: Classifying spells in the NICU

In this chapter, the implementation of the designed framework for the task of classifying neonatal spells is described. An overview of the state of technology in the bedside monitors in the NICU is given in the Background and Related Work chapter of this document. Also in that chapter is a summary of what neonatal spells are and how they have been approached by automated systems thus far. By using Streams and the new framework we can now develop an algorithm to perform complex classifications such as spells using rule-based temporal analysis methods in a hierarchical model. This approach is unique and is well suited to be adopted as a useful clinical decision support tool.

A main step in developing a rule-based system is designing the actual rules. The process of discovering the optimal rule set is outside the scope of my research as it requires deep clinical knowledge of the condition being analysed. This insight is contributed by Dr. Edward Pugh. Combining his expert primary knowledge in the subject and an exhaustive literature review on the various types of spells and how their structures differ and how they are detected in the medical community, a set of rules have been devised. These rules describe a sequence of events that would have to take place in order for a specific type of spell to have occurred.

For the spells classification system, three sources of data are used – the heart rate (HR), the blood oxygen saturation (SpO_2), and the respiratory impedance (RI). These three sources of data are fed into Streams as live streams. The HR and SpO_2 are of the 1 Hz numeric signal type and the RI is a waveform signal coming in at approximately 62 samples per second. Using the full suite of algorithms described in the Methodology chapter, each individual raw data stream is processed to extract features, detect relative changes and anomalies. These single signal events are referred to as level 1 events. They are then combined and the sequence of events is evaluated to assign the entire episode a classification.

Using the architecture described in the previous chapter, the design of the spells classifier is shown in the Figure 16.

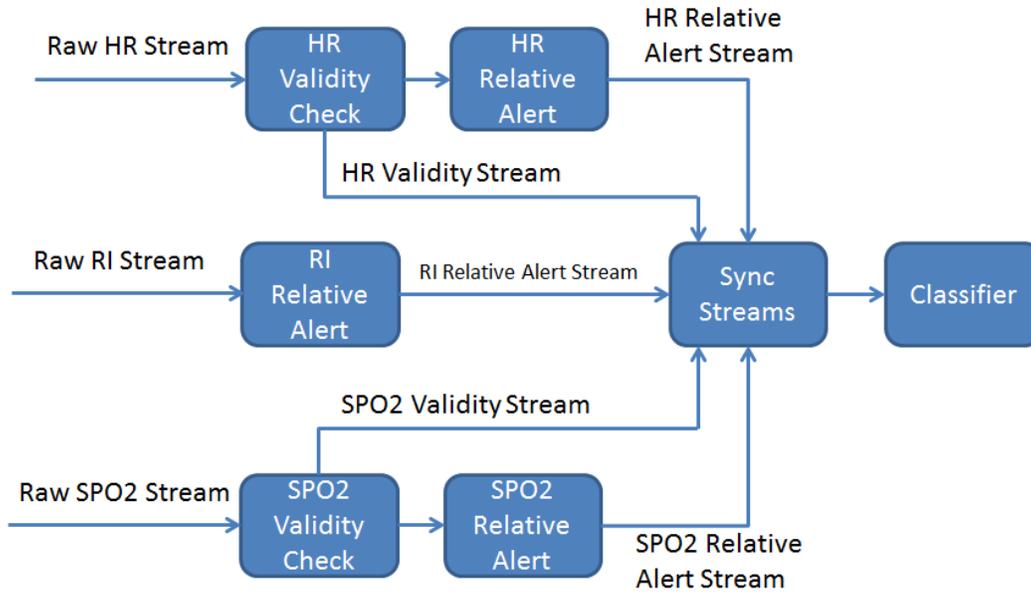


Figure 16. Spells classifier design using Streams

We start by designing the single signal event processing blocks for relative change detection. Below is a figure that details the rules for detecting level 1 relative events in the RI, HR, and SpO₂ used by Pugh et al. [Pugh et al., 2013].

RI		Respiratory pause for > 2 breaths
HR		HR falls more than 10% of baseline
SpO ₂		SpO ₂ falls more than 10% of baseline

Figure 17. Rules for detecting relative events in individual streams

5.1 - Heart rate and blood oxygen saturation events

To detect a relative change in the HR or SpO₂ signal, a fall of more than 10% of the baseline needs to be detected. The designed framework as described in the Methodology chapter provides the Streams

operators to perform the necessary calculations to detect such a fall. What is required is the implementation of two relative change detection modules to the raw HR and SpO₂ respectively. The parameters for the module will have to be set. The parameter values for HR and SpO₂ are shown in Table 1. While the values for the parameters are just a starting point for performing experiments they are not completely arbitrary and were decided by Dr. Pugh based on his clinical expertise of the characteristics of each physiological signal as well based on some experiments conducted with varying parameters. These parameters are designed to be configurable and will be tuned as experimental results are analysed.

Table 1. Values of parameters and thresholds for relative change detection in HR and SpO₂

Parameter	Description	Value in HR graph	Value in SpO ₂ graph
<u>window_size</u>	Size of sliding window to calculate baseline with	30 seconds	30 seconds
<u>read_delta</u>	Difference in time between baseline window and reading point to compare the average to	5 seconds	5 seconds
<u>change_pct</u>	Percentage fall from baseline required to raise relative alert	15 %	3 %
<u>exit_pct</u>	Percentage of the baseline a value is required to be within to be considered a return to baseline	7 %	2 %

Once the two relative change detection modules have been configured, the level 1 event detection system for HR and SpO₂ is complete. Each module will output the relative alert stream along with the data validity stream for that physiological signal.

Streams is well suited for such processing of real-time time series data. Figure 18 below shows the Streams graph for detecting relative change events in the SpO₂ signal. The Streams graph for the HR relative change algorithm is identical except with differences in the parameters as shown in Table 1.

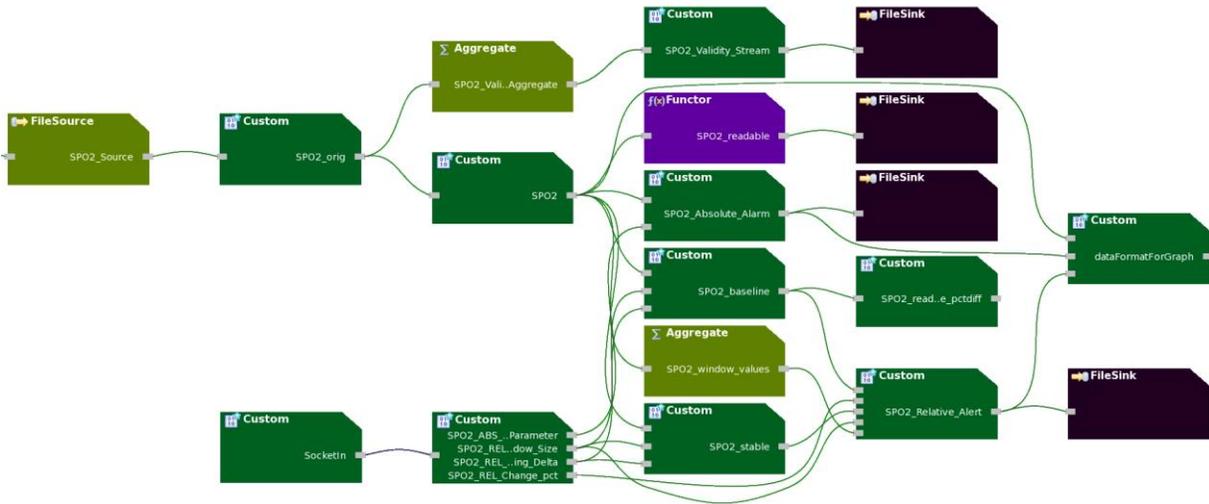


Figure 18. Streams graph for detecting relative change in SpO₂

5.1.1 - Raw input data

Incoming raw data is read through a network socket or a file depending on whether the algorithm is being run prospectively or retrospectively. The algorithm is largely unchanged between the two because all operations involving changes in time uses the timestamp attribute tied to each data sample instead of the actual time changes. Raw data is formatted into tuples containing a pair of data values – a timestamp and physiological value. The SPL operators responsible for handling and distributing the raw data are shown in Figure 19.



Figure 19. SPL operators for handling raw SpO₂ data

5.1.2 - Tunable parameters

The relative change detection module has several parameters that can be set at instantiation or during runtime. These parameters are the absolute alert threshold, the relative alert baseline window size, the relative alert reading delta, and the relative alert percent change. The use of each parameter will be explained in the following sections. In order to be able to modify these parameters during run time, the SPL graph has to communicate with external applications. This is done through a TCP socket interface. The SPL operators that handle this communication are shown in Figure 20.

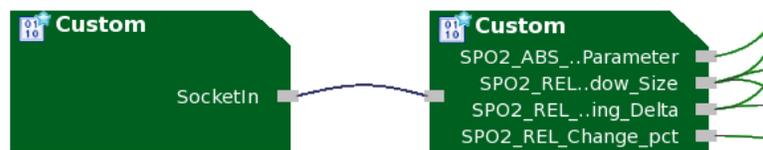


Figure 20. SPL operators for updating parameter values

Messages are received from the external Java interface application and parsed to determine the new value of parameters. Any time a new parameter change message arrives, a tuple is sent to any operator in the graph that is using that parameter.

5.1.3 - Absolute Alarm operator

The incoming data is sent to the Absolute Alarm operator which performs the comparison with the absolute threshold parameter and outputs the absolute alert stream over a socket or to a file. The SPL operator responsible for performing this calculation is shown in Figure 21.

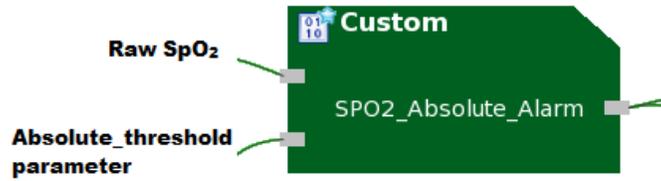


Figure 21. SPL operator for determining absolute alerts

5.1.4 - Baseline Operator

The formatted raw data is connected to the baseline calculating operator shown in Figure 22. This operator reads the timestamp of incoming tuples and keeps a reference of the values for the last 30 seconds. This is the `Window_size` parameter and it can be modified to tune the behavior of the alert. As new tuples come in, it removes the oldest values from the back of its window and pushes the new value at the front of the window. When there is a full window of data, the average of the window is calculated to be the baseline value. The reading point with which the baseline average is compared with is ahead of the baseline window by a few seconds. This parameter is called the `Reading_delta` and has a value of five seconds. Once there is data for a full baseline window average and a valid reading point five seconds ahead of the window, there is enough information for the operator to output a tuple containing the percent difference between the reading point and baseline average.

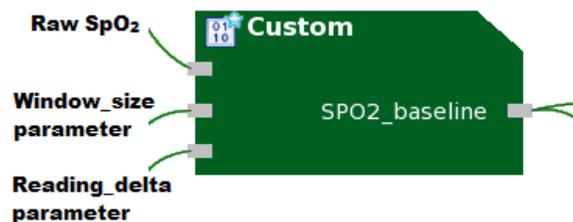


Figure 22. SPL operator for determining baseline

5.1.5 - Relative alert operator

This operator is the focal point of relative change detection as it takes in baseline window values and the percent difference and decides whether a level 1 event should be triggered and when the event should finish. The SPL operators responsible for performing the relative change detection are shown in Figure 23.

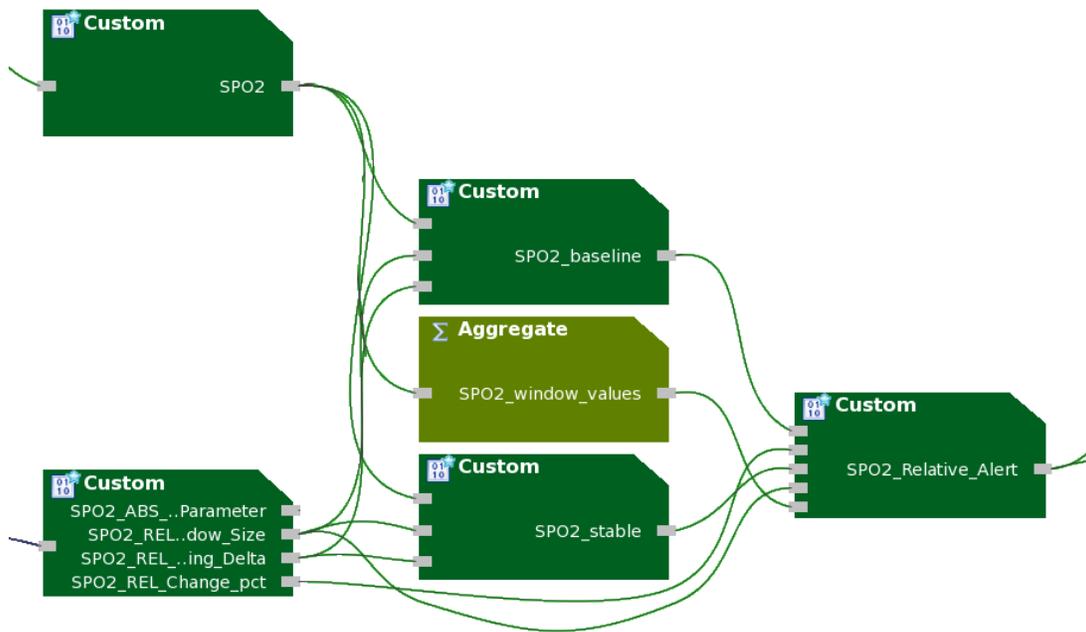


Figure 23. SPL operators for determining relative change alerts

It checks to see if the percent difference is greater than the relative change percent parameter and if it is, it outputs a tuple with the timestamp and an alert value of 1 indicating an in-alert state. It also stores the value of the baseline average at this time as the frozen baseline average to check future values for recovery. If the percent difference is within the limits and the operator is currently not in an alert state, the operator outputs a tuple with the timestamp and an alert value of 0 indicating a non-alert state.

If the operator is in an in-alert state, it checks incoming tuples to see whether the values have recovered to the frozen baseline. To ensure that the signal has indeed recovered back to the baseline and it is not just a single value that just happened to breach the normal limit, an exit condition window is employed. In essence the new values must be within the limits of the frozen baseline for ten seconds in order for the operator to justify the recovery.

Another way for an event to recover is if the value has stabilized around a new value. This is considered a new normal and is valid due to the nature of the physiological signals. To facilitate this, similar to the tracking baseline window another sliding window of larger length (90 seconds for SpO₂) is used to keep track of how stable values are. This stream is also fed to the relative alert operator so it can check whether new values have stabilized somewhere even if it is not the same range as the frozen baseline. A stable baseline is found when the raw values are within two percent of each other in a 90 second window. Raw values are determined to be within the limits of a new stable if they are within one percent of the stable baseline for 10 seconds. If this condition is achieved, it is concluded that the signal has found a new stable baseline and the event can be ended.

The raw data is also windowed using an Aggregate operator. This operator then sends each full window of samples as a tuple to the next operator, which calculates the baseline average. The Relative Alert operator uses the windowed values, the baseline value and the latest value to determine whether a relative fall has occurred.

5.1.6 - Live display output

The external software that is used to update algorithm parameters at run time can also receive data from Streams. It does this using a second network socket connection to the application. The raw values, the absolute alert stream, and the relative alert stream are joined and formatted into a more

convenient structure for sending to the external application for live on-screen display of streams. The SPL operators that perform this function are shown in Figure 24.

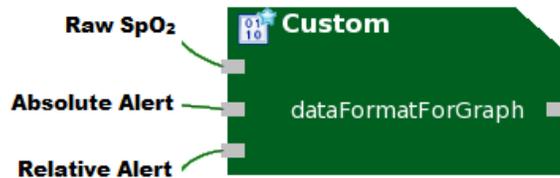


Figure 24. SPL operator for sending data to live display

5.1.7 - Validity stream

In addition to the relative alert, there is a stream that presents a metric for the quality of the data received. As described in the Methodology chapter, for HR and SpO2 this operator checks the incoming value to see whether it is 8,388,607 indicating a poor signal or a leads off scenario. If the incoming signal is valid, the operator outputs a validity value of one for that timestamp and if it detects any 8,388,607 in the last 30 seconds it outputs a validity of zero for that timestamp. This means that a signal returning from a leads off scenario needs to present valid signal values for 30 seconds for the validity operator to justify its validity. This validity stream will be used alongside the output of the relative alert stream when performing classifications. The SPL operators for performing the validity check are shown in Figure 25.

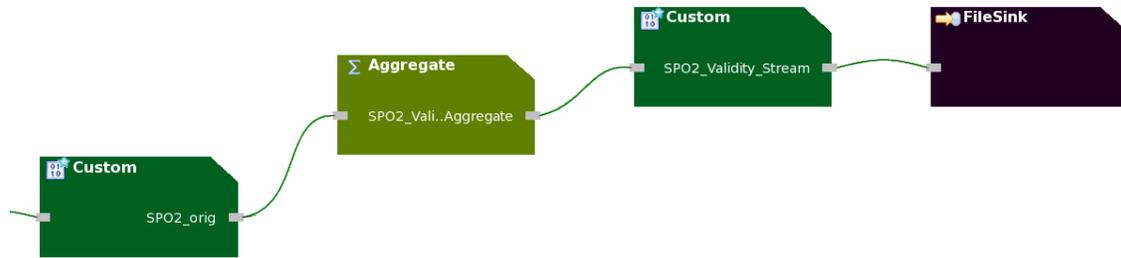


Figure 25. SPL operators for performing validity check

5.2 - Breathing events

In addition to the HR and SpO2 relative events, classifying spells requires the analysis of the respiratory impedance (RI) waveform. The RI waveform represents the expansion and contraction of the chest as the patient is breathing with the peaks in the signal corresponding to maximum chest expansion in each breath. The rule for detecting a relative alert in the RI signal is that a respiratory pause is defined when there is no breath for the duration of time it took for the previous two breaths. Implementing this in Streams involves the use of the live peak locations stream in the developed framework as well as several sliding windows keeping track of the time it took for the previous two breaths and the time that has elapsed without a breath. Figure 26 below shows the operators and connections in the Streams graph for detecting pauses in breathing using the RI wave.



Figure 26. Streams graph for detecting relative change events in the RI signal

Incoming data is preprocessed to get to a form usable for analysis. Once the data is in timestamp and value pairs it goes through the peak detection portion which outputs a breaths stream indicating the locations of every breath. This stream is connected to the relative alert operator which measures the time between breaths and determines when a missed breath has occurred.

5.2.1 - Raw input data

Incoming raw data is read through a network socket or a file depending on whether the algorithm is being run prospectively or retrospectively. RI data comes in from the monitor at a rate of 62.5 samples per second. This odd number is the result of the way patient monitors send their data to Artemis. The format for incoming data is a timestamp and an array of 16 values. There are 4 such packets received every 1024 milliseconds. Hence the data rate is 62.5 samples per second. Some preprocessing is required to make this data in a format that is more convenient to use. To do this, the four packets are combined and all values are assigned an interpolated timestamp. The end result of this operator is a

stream of tuples each consisting of one timestamp and one value. The SPL operators responsible for handling and distributing the raw data are shown in Figure 27.

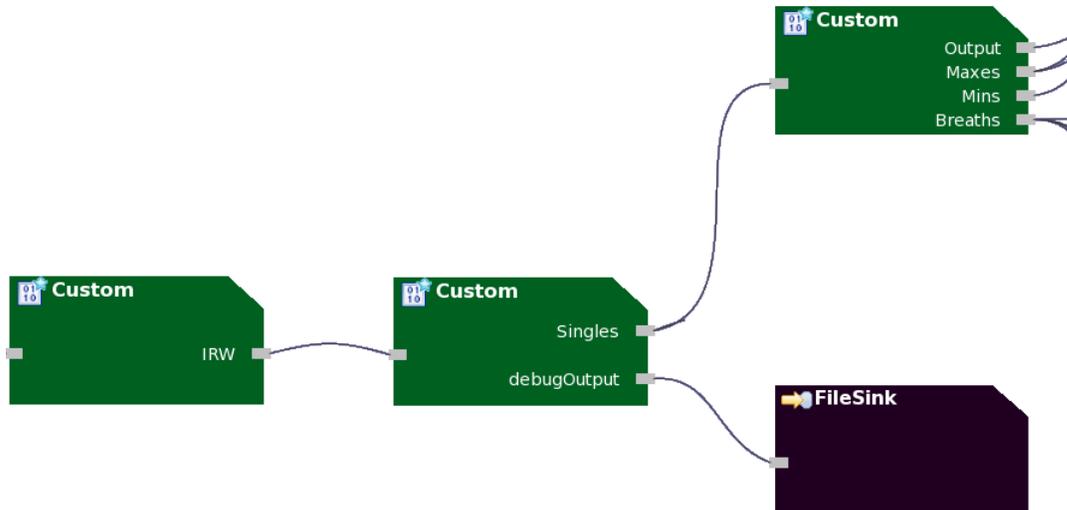


Figure 27. SPL operators for handling raw RI data

5.2.2 - Live display output

Source RI data is sent through an operator to the external streams display application. In the same way as the HR and SpO₂ graphs send data out, a network socket connection is made from the operator. Some preprocessing is done to reduce the amount of data being sent to the display. This is because 62.5 samples are not required to reproduce the RI waveform with enough detail for human observation. Down-sampling is performed by an Aggregate operator with a tumbling window of 0.5 seconds for which one value is sent. The SPL operator for sending data to the live display is shown in.

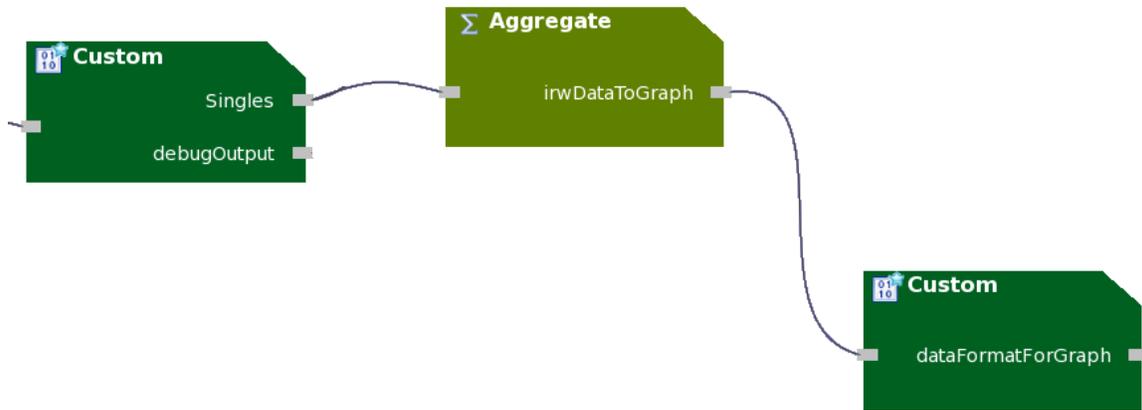


Figure 28. SPL operators for sending data to live display

5.2.3 - Detecting breaths

Implementing the generic peak detection module described in the Methodology chapter, the breath detection operator performs peak detection on the raw RI signal to find the starts of breaths. Figure 29 shows that the breath detection operator's only input is the raw RI after the timestamps are interpolated and each tuple is a timestamp and value. The operator's outputs include the maximums and minimums stream as well as the Breaths stream, which is basically the live peak stream described in the Methodology.

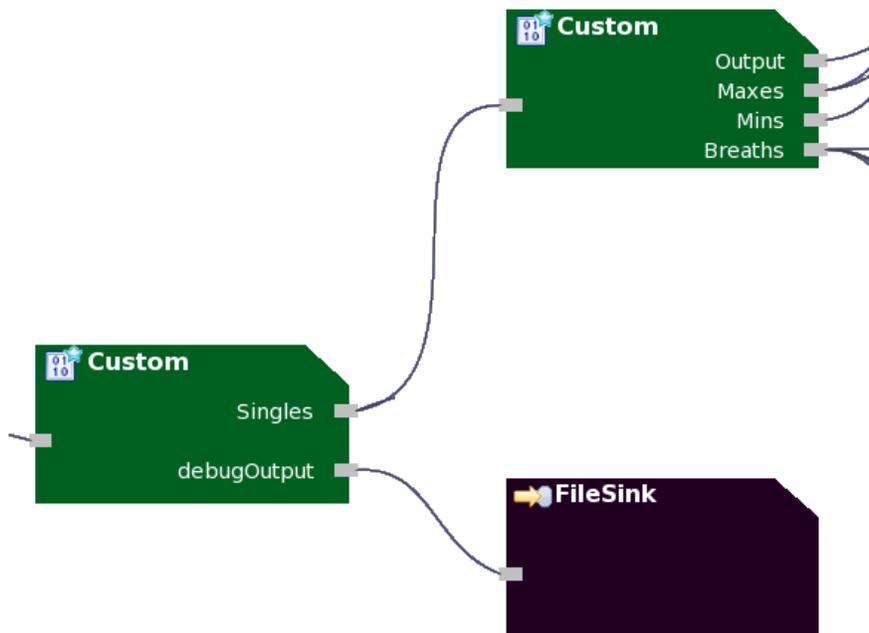


Figure 29. SPL operators for calculating locations of breaths

To determine when there has been a pause in breathing, the locations of each breath and the time between breaths must first be calculated. Generally, the peaks in the RI waveform are the starts of breaths however there is some cross-talk from the heart activity as the physical movement of the chest is also affected by the beating heart which causes minor peaks in the RI signal. This has to be filtered out to accurately measure the breath using RI. This is achieved by incorporating the threshold cut-off functionality in the framework that ensures only major peaks in the waveform are recognised as breaths and minor peaks are ignored. The delta parameter in the peak detection module from the framework is configured such that a peak is only detected as a breath if it is higher than its surrounding points by at least 60% of the range of the signal. Below is a figure of the raw RI signal and the detected breaths using live peak detection module described in the Methodology chapter.

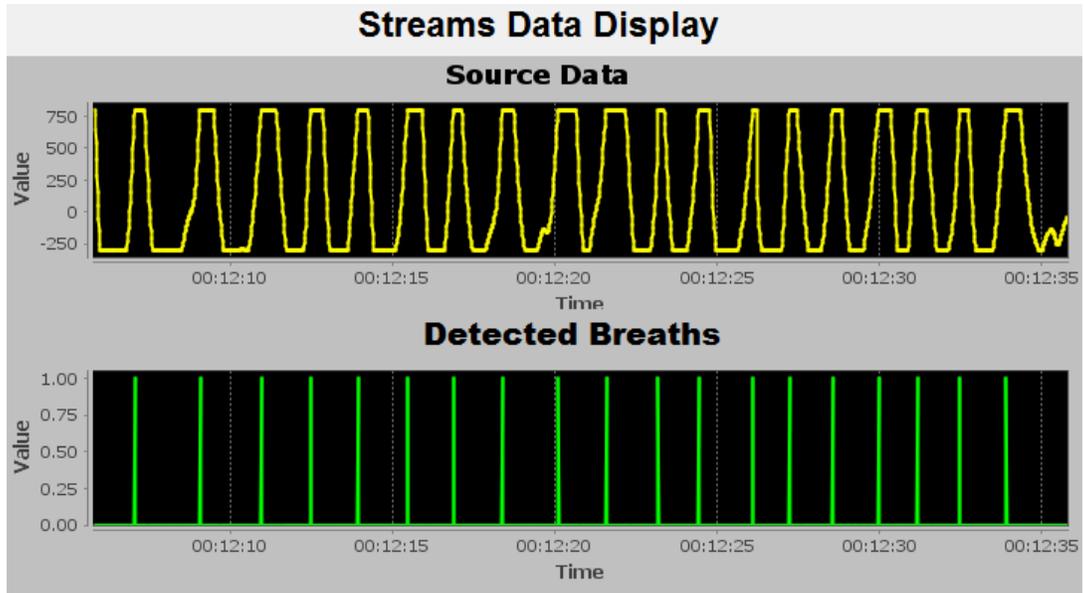


Figure 30. Detecting locations of breaths in a raw RI signal using live peak detection

5.2.4 - Absolute alert operator

The absolute alert is configured as an alert that is raised when 15 seconds of no breathing have gone by. It takes the live peak stream, or the detected breaths stream in this case, and sends it to an Aggregate operator. This operator aggregates the detected breaths stream with a sliding window of 15 seconds and sends the number of breaths detected in the window to the absolute alert operator. The absolute alert operator reads the incoming tuples and if any window contained zero breaths, an absolute alert is raised. The alert is returned when a full breath has been detected again. The SPL operators to perform the absolute alert algorithm are shown in Figure 31.

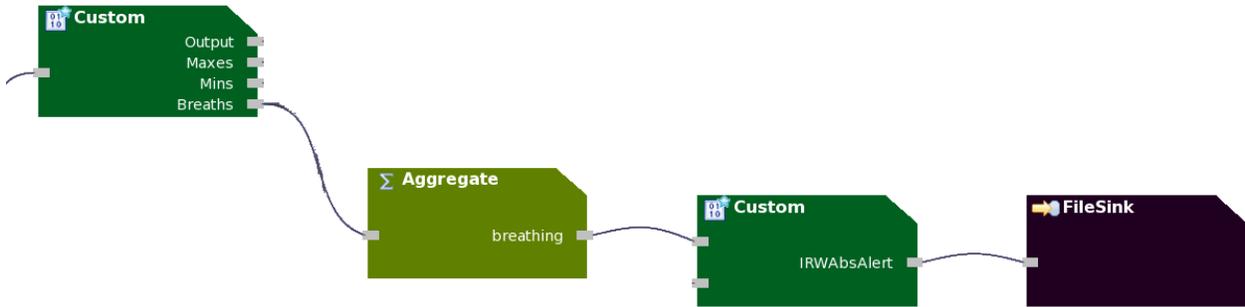


Figure 31. SPL operators to perform the absolute alert algorithm for an RI signal

5.2.5 - Relative alert operator

This is the main operator involved in generating the level 1 event stream for RI. It receives tuples indicating the locations of breaths and is in charge of determining when a breath has been missed and how much time has gone by before normal breathing continues. The SPL operators responsible for performing the missed breath detection are shown in Figure 32.

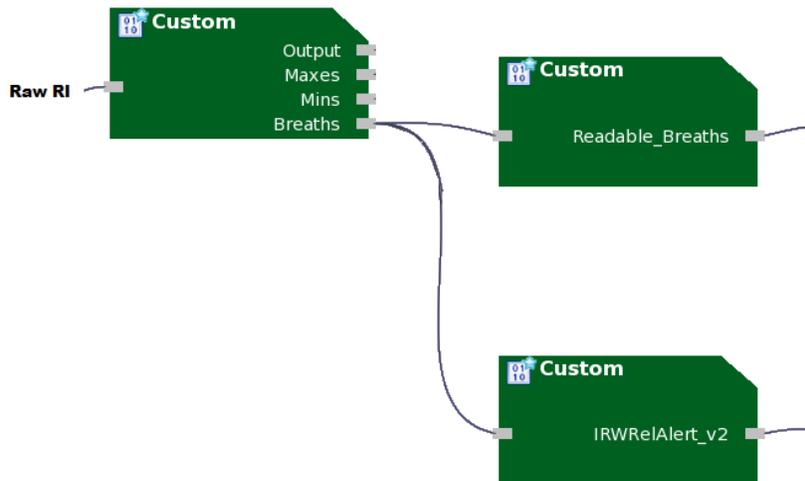


Figure 32. SPL operators for determining missed breath alerts

It operates as kind of a state machine first checking for a breath, then checking for a second breath, then continuously recalculating the time for the last two breaths and checking the time elapsed since the last breath. Figure 33 shows the output of the RI relative alert operator's logic. It begins with A, where the signal is stable and a two breath time is calculated. B shows how the two breath time is kept updated by recalculating at every new breath. C shows the alert being raised when the latest two breath times has elapsed and no new breath was found. D shows a complete breath being taken in under less time than the two breath time, so the alert is returned to zero.

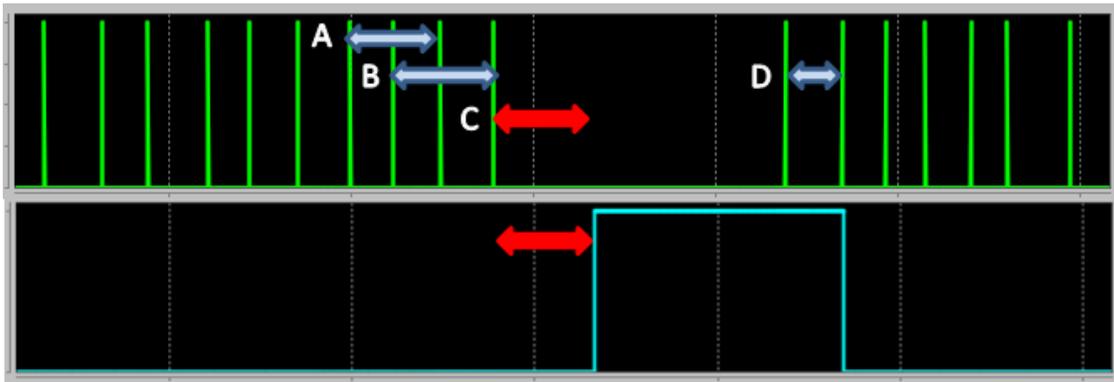


Figure 33. Logic of RI relative alert stream

The code is structured as a state machine that starts in state 0 where it is searching for the first breath. When an incoming tuple has a breath value of 1, the timestamp is recorded and the state is switched to 1. In state 1, the operator reads the incoming tuples until another breath value of 1 appears. When it does, the timestamp is recorded as the second breath and the state is switched to 2. During state 2, the operator is searching for the next breath so it can calculate the time it took for the last two complete breaths. Once it finds the third breath and calculates the two breath time the state switches to 3. In state 3, the timestamp of every incoming tuple is checked and if the breath value is 1, the two breath time is recalculated and the operator stays in state 3. If the breath value is 0 the operator checks how

much time has elapsed since the last breath. If it has been more than the two breath time, and event start is recorded and the operator goes into state 0 with alarm value 1. The operator then checks incoming breaths until there has been one complete breath within the last valid two breath time. This marks the end of an RI level 1 event. The entire logic of the operator is shown in Figure 34 as a state diagram.

5.3 - Spells classification

Now that all three individual alerts are configured, they can be linked to a sync module to get one stream of all alert values and validity values. This stream can then be connected to the episode profile module, whose logic is described in the Methodology chapter. The output of the episode profiler is an episode buffer. The creation of the episode buffer represents the end of the framework's standard procedures and the start of spells specific rules based classification. Figure 35 shows the operators and connections in the Streams graph for classifying spells.

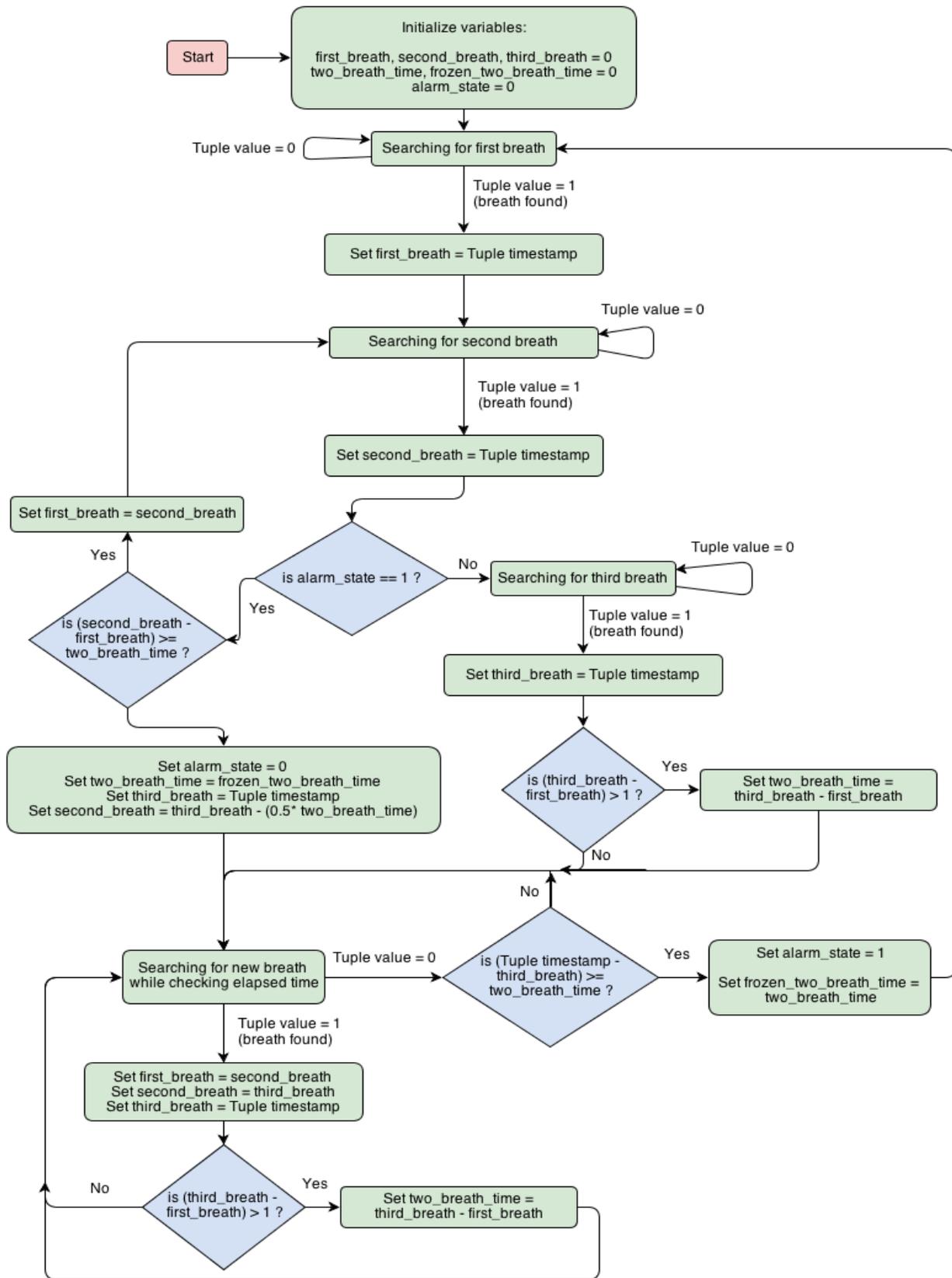


Figure 34. State Diagram of RI relative alert detection algorithm

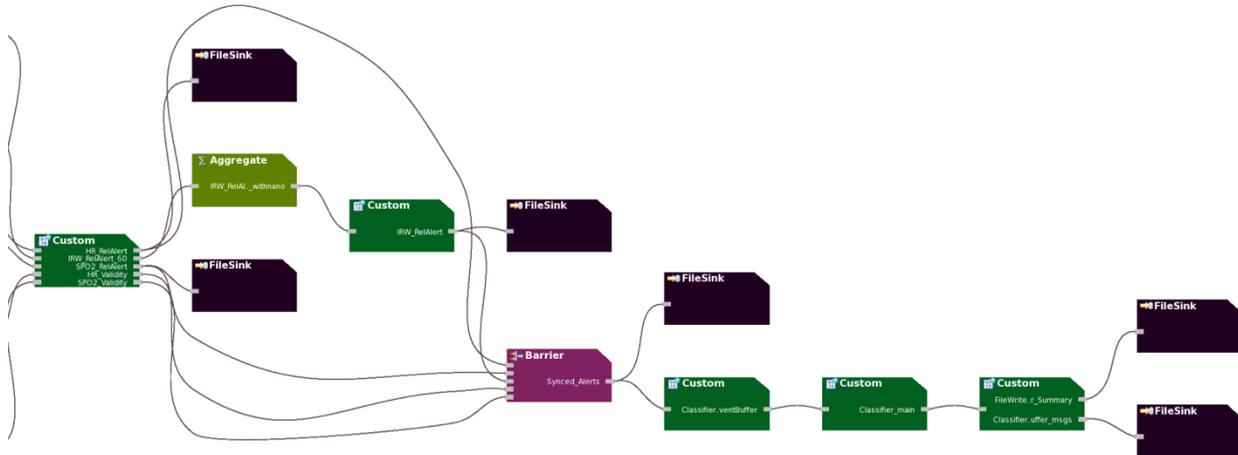


Figure 35. Streams graph for classifying spells

5.3.1 - Incoming individual alert data

Incoming individual alert data is read from concurrently running individual alert graphs or from a file depending on whether the algorithm is being run prospectively or retrospectively. The incoming data consists of the three alert streams, each one sending tuples containing timestamp and alert value pairs. In addition there are the two validity streams whose format is a timestamp and validity value. The SPL operators responsible for collecting and distributing the individual alert data are shown in Figure 36.

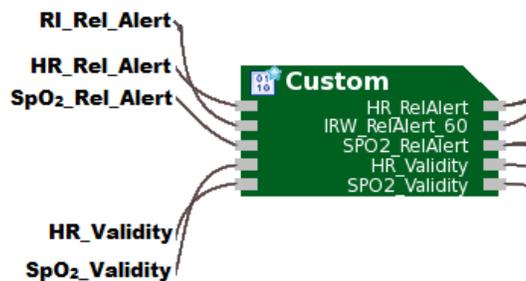


Figure 36. SPL operator for handling individual alert data

5.3.2 - Syncing streams

In order to make any comparisons between the three alerts, a vital step is to sync the event streams. At the end of the sync process, the tuples emitted should have all five values and only one corresponding

timestamp. The Barrier operator in Streams is used to check the timestamp of the five incoming streams and waiting for a match on all five ports and when a match is found, one tuple with the above format is emitted. The SPL operators involved in syncing the streams are shown in Figure 37.

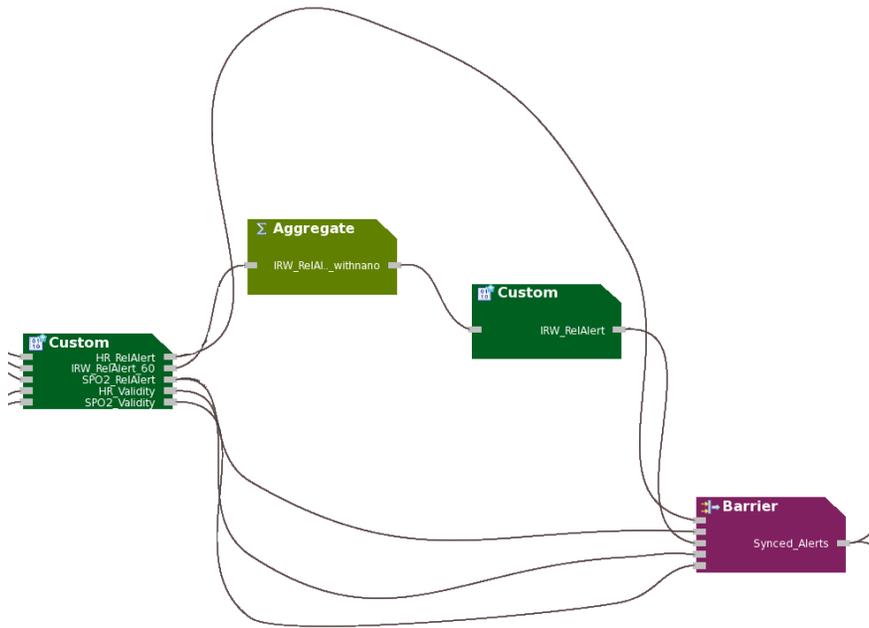


Figure 37. SPL operators for syncing streams

Also, a consideration has to be made for the fact that RI data is coming at a faster rate than the HR and SpO₂. As described in the methodology chapter, since temporal features have been extracted in the RI signal, the alert can be down sampled to match the rate of the other streams. A one second window is taken and if there was an RI alert value of one at any point within the second, that second would be labeled an alert value of one. This effectively converts the RI alert stream into a format identical to the other two alert streams. In addition to the three alert streams, the validity streams are synced as well.

5.3.3 - Generating episode profile

Once the events and validity streams are synced, they go through the episode detection phase. The purpose of this operator is to segment off the time and alert signals to send to the classifier to do analysis

on. The rules and logic for performing this are described in the Methodology chapter. Figure 38 shows the SPL operator responsible for reading the synced alerts and generating the episode profile.

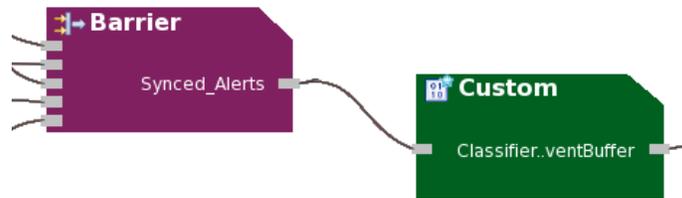


Figure 38. SPL operator for generating episode profile

5.3.4 - Classifying the spell

This operator performs the penultimate step in the entire algorithm. The SPL operators responsible for performing the rule based classification and generating the final report with event logs are shown in Figure 39.



Figure 39. SPL operators responsible for rule based classification and report generation

This Streams operator takes the episode buffer and first checks how many signals were affected in the episode. If only one signal was affected, the classification becomes very straight forward. If only the RI was affected, the episode is classified as an isolated RI pause. If only the HR was affected, the episode is classified as an isolated bradycardia. If only the SpO₂ was affected, the episode is classified as an isolated desaturation.

If more than one signal was affected in the episode, then the classification is much more complicated. The episode is scrolled through from start to finish, recording the start and end times of

the three individual alerts. In the case of RI pauses, because there can be multiple RI pauses within one episode the number of pauses is recorded for output in the spell summary.

Once the start and end times of the relative alerts is found, they are sorted by timestamp to determine the event sequence. This sequence is encoded and sent to the classification determination part of the code. One of the goals for this research was to make a system that would be embraced by the medical community and this required the type of classifier to not be a complete black box. A rule based approach was chosen. By using clearly defined rules, the results of an automated classifier can be decomposed and easily explained to physicians in a report style document. The rules describe a sequence of events that would have to take place in order for a specific type of spell to have occurred. Table 2 below shows the various spells and their corresponding sequence of events as determined by Dr. Pugh.

Table 2. The different spells types and their corresponding sequence of events

Type of spell	Fall from baseline			Recovery from baseline		
	HR	RI	SpO ₂	RI	HR	SpO ₂
Central	2	1	3	4	5	6
Vagal	1	1	2	3	3	4
Obstructive	1 (Incr.)	-	2	-	3	4
Obstructive Central	1 (Incr.)	3	2	4	5	6
Central Obstructive	2	1	3	4	5 (Fall)	6
Desaturation	-	-	Absolute	-	-	-
Bradycardia	Absolute	-	-	-	-	-
Possible Isolated Desaturation	-	1 U 2	1 U 2	3	-	3
Possible Isolated Bradycardia	1 U 2	1 U 2	-	3 U 4	3 U 4	-

The rule based logic for the classification is described in the Methodology chapter. Table 3 lists the code values corresponding to the various alert state transitions.

Table 3. All the state transitions being monitored and their event code values for use in the classification stage

Expression Name	Description	Event code value
HR_FALL_START	This code represents the start of an HR relative fall event	0
HR_FALL_RECOVER	This code represents the end of an HR relative fall event as it recovers back to baseline	1
HR_RISE_START	This code represents the start of an HR relative rise event	2
HR_RISE_RECOVER	This code represents the end of an HR relative rise event as it recovers back to baseline	3
SPO2_FALL_START	This code represents the start of an SpO2 relative fall event	4
SPO2_FALL_RECOVER	This code represents the end of an SpO2 relative fall event as it recovers back to baseline	5
SPO2_RISE_START	This code represents the start of an SpO2 relative rise event	6
SPO2_RISE_RECOVER	This code represents the end of an SpO2 relative rise event as it recovers back to baseline	7
RI_PAUSE_START	This code represents the start of the first RI pause event	8
RI_PAUSE_RECOVER	This code represents the end of the last RI pause event	9

The sequence of state transitions is defined as the event sequence code. Figure 40 shows the classification logic of how final spells classifications are made based on the event sequence code.

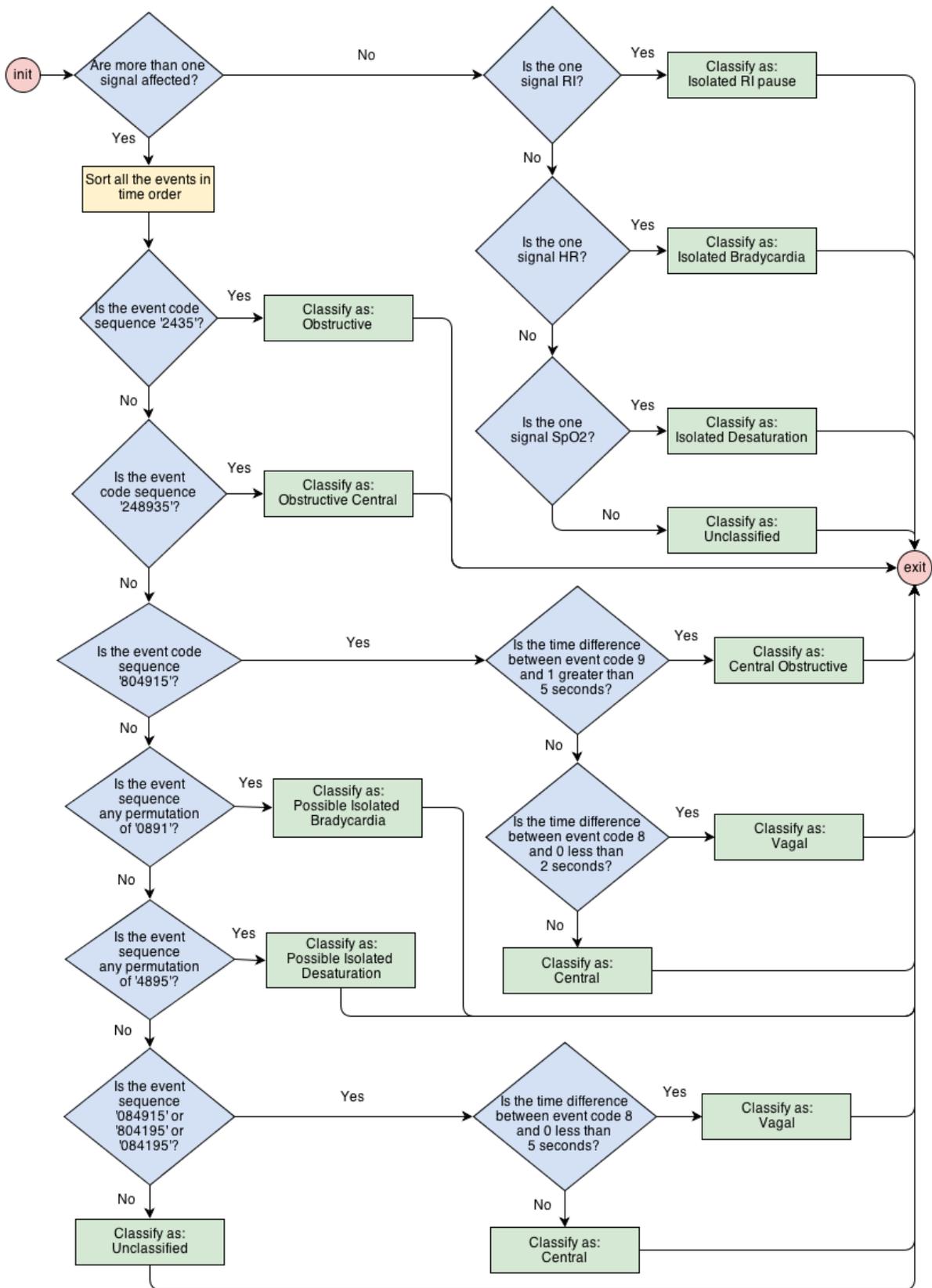


Figure 40. State diagram of the spells classification process

The logic begins with checking how many signals were affected in the event. This is because if it is found that only one signal was affected, the classification process is much quicker. If only one signal is affected, the classifier checks whether it is RI, HR, or SpO₂ and ends the process with a classification of isolated RI pause, or isolated bradycardia, or isolated desaturation respectively. However if more than one signal is affected, the next step is to sort all the event transitions in time order. From here, it is possible to observe the chain of events in the episode. Referring to the table with the event sequences for various types of spells, it is possible to do a simple lookup with if-statements to check whether a buffer follows the sequence for a specific spell type. For example, if the order of events shows that an HR_RISE_START event was first, followed by an SPO2_FALL_START and then an HR_RISE_RECOVER and finally an SPO2_FALL_RECOVER event, then the event sequence according to the encoding table in Table 3 would be “2435”. In the classifier logic, the same encoding is performed so a simple check to see if the event code matches will result in a spells classification.

Some computational considerations were made to effectively describe some events. For a classification of vagal apnoea the HR drop and the RI pause should happen simultaneously. However the chance of an automated system finding simultaneity when timestamps are measured to the millisecond is low. This is a pragmatic challenge for the technology. The key is to realise that the term simultaneous in this case really means contextually concurrent and so in the classifier it is defined as two events occurring within two seconds of each other.

Another such interpretation is presented in the sequence for central obstructive apnoea episodes. In the recovery phase of a central obstructive apnoea episode, the HR continues to fall. To make the link between the recovery phase in the automated classifier and the recovery phase in the physiological sense, a closer look at how episodes are segmented is taken. Since an episode cannot reach the classifier block without having all alerts go back to 0, there is no way the HR alert can be 1 at

the end of an episode. However, if the HR continued to fall and reached a different baseline it would take much longer for the alert to recover to 0. For classifying central obstructive events, this behaviour can be identified in the episode buffer by measuring the time between the end of the RI pause event and the end of the HR event. If the difference is greater than five seconds, it is determined that the HR took a long time to recover after the rest of the signals have recovered. This is a signature of a central obstructive episode and so the classification can be made. The trick with rule based expert systems is to be able to interpret the real world rules in such a way that is computationally feasible to implement.

5.4 - Presentation of Classification Results

The previous chapter presented a model for the application of the framework through the developed system to the case study problem of this research, namely neonatal spells. This chapter presents the results of using this approach on physiological data captured from premature infants using Artemis. This research was part of the “Real-time, multidimensional temporal analysis of complex high volume physiological data streams for the identification of condition onset predictors for nosocomial infection in the neonatal intensive care unit”. REB File No.: 1000013657 at SickKids and File No.: 09/002 at UOIT.

Figure 41 presents some example output of the individual alert algorithms as logged for the entirety of a patient study in addition to the final classifications. This was prepared initially for debugging purposes through the validation steps of the algorithm. This format was necessary to support the experiments as detailed in the next section. As noted in chapter 1, the user interface for clinicians providing only clinical relevant information via an analytics interface is outside the scope of this research and is part of the research of Artemis team member Rishikesan Kamaleswaran. The study report consists of a main text file called “Classifier Summary” which logs every classification made by the system. The content of the summary file is shown below in Figure 41.

```

631 Event detected (#211)--> Start: Thu Jul 18 00:32:07 2013 (1374121927), End: Thu Jul 18 00:32:18 2013 (1374121938)
632 -> Duration(sec): 11, #Signals Affected: 2, Signals: - SPO2 - - IRW - , Classification: { SPO2 Fall->RI Pause->RI
Recover->SPO2 Recover (#_RI: 1, RI_time: 3s) } >>> Possible Isolated Desaturation
633
634 Event detected (#212)--> Start: Thu Jul 18 00:32:21 2013 (1374121941), End: Thu Jul 18 00:32:24 2013 (1374121944)
635 -> Duration(sec): 3, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
636
637 Event detected (#213)--> Start: Thu Jul 18 00:32:45 2013 (1374121965), End: Thu Jul 18 00:32:49 2013 (1374121969)
638 -> Duration(sec): 4, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
639
640 Event detected (#214)--> Start: Thu Jul 18 00:33:40 2013 (1374122020), End: Thu Jul 18 00:35:10 2013 (1374122110)
641 -> Duration(sec): 90, #Signals Affected: 2, Signals: - HR - - SPO2 - , Classification: { HR Rise->SPO2 Fall->HR
Recover->SPO2 Recover } >>> Obstructive
642
643 Event detected (#215)--> Start: Thu Jul 18 00:35:47 2013 (1374122147), End: Thu Jul 18 00:35:52 2013 (1374122152)
644 -> Duration(sec): 5, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
645
646 Event detected (#216)--> Start: Thu Jul 18 00:36:02 2013 (1374122162), End: Thu Jul 18 00:36:05 2013 (1374122165)
647 -> Duration(sec): 3, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
648
649 Event detected (#217)--> Start: Thu Jul 18 00:36:07 2013 (1374122167), End: Thu Jul 18 00:36:12 2013 (1374122172)
650 -> Duration(sec): 5, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
651
652 Event detected (#218)--> Start: Thu Jul 18 00:36:14 2013 (1374122174), End: Thu Jul 18 00:36:22 2013 (1374122182)
653 -> Duration(sec): 8, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
654
655 Event detected (#219)--> Start: Thu Jul 18 00:36:57 2013 (1374122217), End: Thu Jul 18 00:37:04 2013 (1374122224)
656 -> Duration(sec): 7, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
657
658 Event detected (#220)--> Start: Thu Jul 18 00:37:23 2013 (1374122243), End: Thu Jul 18 00:37:31 2013 (1374122251)
659 -> Duration(sec): 8, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
660
661 Event detected (#221)--> Start: Thu Jul 18 00:37:55 2013 (1374122275), End: Thu Jul 18 00:38:35 2013 (1374122315)
662 -> Duration(sec): 40, #Signals Affected: 2, Signals: - HR - - IRW - , Classification: { HR Fall->RI Pause->RI
Recover->HR Recover (#_RI: 2, RI_time: 18s) } >>> Possible Isolated Bradycardia
663
664 Event detected (#222)--> Start: Thu Jul 18 00:39:05 2013 (1374122345), End: Thu Jul 18 00:39:08 2013 (1374122348)
665 -> Duration(sec): 3, #Signals Affected: 1, Signals: - IRW - , Classification: Isolated RI pause
666

```

Figure 41. Contents of the classifier summary file

There is a line in the file for every event detected. The event is numbered and the start and end time as well as the duration of the full event are noted. Also, the number of signals affected is logged. This is because for further analysis all events with only one signal affected can be filtered out to leave behind only complex multi-signal episodes. For single signal events, the signal affected is noted, and finally the classification is recorded. In the case of multi-signal events, the number of signals and names are noted. Then the event sequence is recorded with easily understandable labels describing each individual alert's state. Figure 42 below shows the summary for a possible isolated desaturation event.

```

Event detected (#211)--> Start: Thu Jul 18 00:32:07 2013 (1374121927), End: Thu Jul 18 00:32:18 2013 (1374121938)
-> Duration(sec): 11, #Signals Affected: 2, Signals: - SPO2 - - IRW - , Classification: { SPO2 Fall->RI Pause->RI
Recover->SPO2 Recover (#_RI: 1, RI_time: 3s) } >>> Possible Isolated Desaturation

```

Figure 42. Event summary for a possible isolated desaturation classification

It is shown that the first event to occur is a fall in SpO₂. This is followed by a pause in breathing, then a recovery of breathing and finally by a recovery of SpO₂. The event summary also shows the number of respiratory events that occurred during the duration of the episode. Finally, the result of the classification is shown as a “Possible Isolated Desaturation”.

Apart from the summary file, there is another text file that is recorded that contains a more detailed report including the contents of the entire buffer of alert values for the episode. A figure of the buffer file for the event described above is shown below.

```

1 EventBuffer: event #211
2
3 Start: Thu Jul 18 00:32:07 2013 (1374121927)
4 End: Thu Jul 18 00:32:18 2013 (1374121938)
5 Duration(sec): 11
6 # Signals Affected: 2
7 Signals Affected: - SPO2 - - IRW -
8 Classification: { SPO2 Fall->RI Pause->RI Recover->SPO2 Recover (#_RI: 1, RI_time: 3s) } >>> Possible Isolated
  Desaturation
9
10 {readingtime=(1374121926,0,0),HR_AlertValue=0,SPO2_AlertValue=0,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
11 {readingtime=(1374121927,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
12 {readingtime=(1374121928,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
13 {readingtime=(1374121929,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
14 {readingtime=(1374121930,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
15 {readingtime=(1374121931,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
16 {readingtime=(1374121932,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
17 {readingtime=(1374121933,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}
18 {readingtime=(1374121934,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=1,HR_ValidityValue=1,SPO2_ValidityValue=1}
19 {readingtime=(1374121935,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=1,HR_ValidityValue=1,SPO2_ValidityValue=1}
20 {readingtime=(1374121936,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=1,HR_ValidityValue=1,SPO2_ValidityValue=1}
21 {readingtime=(1374121937,0,0),HR_AlertValue=0,SPO2_AlertValue=1,IRW_AlertValue=1,HR_ValidityValue=1,SPO2_ValidityValue=1}
22 {readingtime=(1374121938,0,0),HR_AlertValue=0,SPO2_AlertValue=0,IRW_AlertValue=0,HR_ValidityValue=1,SPO2_ValidityValue=1}

```

Figure 43. Event buffer file for a possible isolated desaturation event. This file contains the summary information as well as the entire episode buffer of alert values

Since the start and end time of the episode is noted, the raw signal values for the duration can also be pulled from the database and made available for viewing. The separation of output data into these three levels enables the creation of intuitive user interfaces to display classifications. The development of such interfaces is not in the scope of this thesis, but it has been envisioned that the study summary would be displayed first and any episode that needs to be viewed in greater detail can be clicked on to bring up data from the event buffer file. That, combined with the raw signal data can be displayed

together in a synchronised format for a user to view the entire decomposition of the event and fully understand the classification process.

In addition to the textual report generated, validation of the algorithms and classifications can be done with the aid of custom software described in the next chapter.

Chapter 6 - Results and Discussion

In this chapter, the results from testing the framework are presented. The process by which the output of the framework is validated is described in the Experimental Design section.

6.1 - Experimental Design

The testing and validation of the designed framework is done within the context of the NICU case study for real-time spells detection. The validation methods are presented below in two steps. In the first phase, the algorithm output of the individual alerts is compared to a neonatologist's annotations on a raw stream data set. This includes the relative change detection algorithms for heart rate and blood oxygen saturation as well as the breath pause detection in the respiratory impedance signal. The second phase of validation is performed by comparing the neonatologist's annotations of spells events with the framework's final classification report.

As the first phase of validation of the algorithms, the relative change algorithms for HR, SpO₂, and RI are run retrospectively through data collected by Artemis. The database is screened for babies that are suitable for the study. This means filtering patients that are suitably premature to fit the parameters used in the algorithm. A 24-hour sample of one patient's HR, SpO₂, and RI is acquired. The signals are then run through the individual alert graphs and their output is validated using a custom programmed data display and review application. Then the signals are run through the entire classification algorithm and the output report of the framework will be compared to annotated episodes of apnoea in the same data set.

6.1.1 - Streams Data Display Application

The patient data is stored in a database and formatting it for visual display traditionally requires software tools such as SPSS or MATLAB or at the least a spreadsheet software such as Excel. To make the process of annotating the patient data easier for the neonatologist, an application was developed as

a contribution of this thesis to display the three relevant physiological signals in a synchronised time format. This is similar to how the patient bedside monitor displays signals, and it enables the reviewer to pan through hours of data more comfortably than the traditional software tools. Below is a figure of how the software presents the HR, SpO₂ and RI signals for review.



Figure 44. The Streams Data Display application enables a reviewer to pan through hours of HR, RI, and SpO₂ data to validate alerts.

The slider bar can be dragged to pan the data forward and backward in time. For more precise control, a textbox and left and right arrows allows the user to enter the number of seconds to pan.

The software is configurable in the sense that it is not limited to displaying the raw HR, raw SpO₂, and raw RI. It can load the raw HR on the top display, HR absolute alert on the second display, and

HR relative alert on the bottom display. This allows the reviewer to compare the outputs of the individual alerts with the traditional threshold system.

In addition to displaying the data, the application allows for the user to click on any point on the signal to input an annotation. The timestamp of the clicked point is retrieved and printed in the textbox. A note can be entered next to the timestamp to describe any event that may be occurring. This text can then be copied out to an external document or report.

6.1.2 - Validating individual alerts

First, the individual alerts are run separately without the classification phase. The output of the Streams graphs running the individual alert graphs is a text file containing rows of timestamp and alert value pairs. This text file is loaded onto the data display program. For ease of comparison, the raw signal is loaded on the top graph, the absolute alert is loaded in the center graph, and the relative alert is loaded on the bottom graph. Since the three graphs move in sync, it is simple for a neonatologist to validate the alerts based on the behavior of the raw signal. Below is a figure of the data display application setup for reviewing SpO₂ alerts.



Figure 45. Streams Data Display application showing raw SpO₂, SpO₂ absolute alert, SpO₂ relative alert

For the case of validating RI events, things are slightly different. The main steps of the algorithm for detecting breathing events are detecting the locations of the start of breaths, and then to calculate the time between breaths. For the purpose of validating this stream, the algorithm was configured to record the locations of the breaths onto a text file. With the addition of the raw RI signal and the final relative alert signal, a neonatologist can review the accuracy of breath detection as well as the accuracy of the breathing alert detection. Figure 46 shows the data display application setup for reviewing RI events.



Figure 46. Streams Data Display application setup to view raw RI, detected breaths, and RI relative alert

6.1.3 - Validating classifications

The framework outputs its classifications as a report in text files. This output contains a summary of all the events as well as a more in depth log of each event. A neonatologist can view the summary file along with the streams display application showing the individual alerts. A second instance of the streams display program can be launched and configured to display the raw signals for the same time frame. By reviewing the sequence of events detected along with the raw signals, the final classification can be validated.

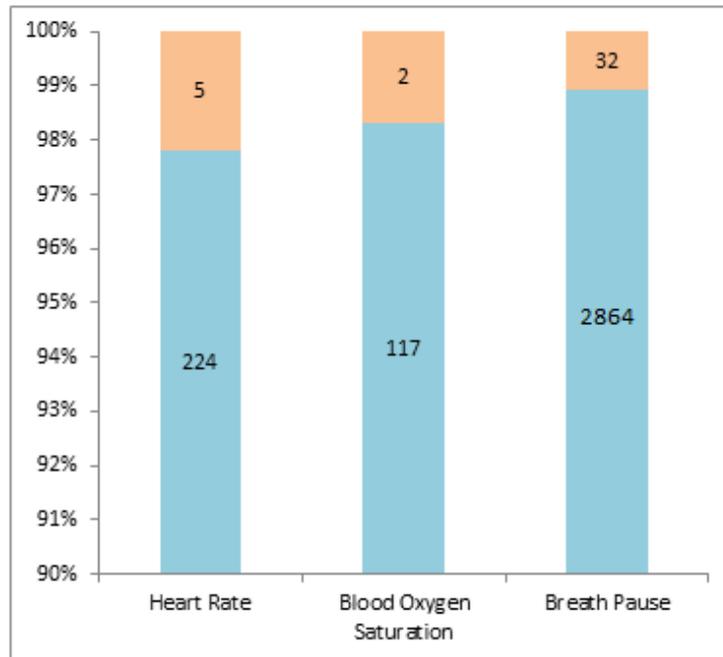
6.2 - Results

A suitable patient was found in the Artemis database who was reported to be experiencing spells. A 24-hour sample of this neonatal patient was used to evaluate the accuracy of the event detection

algorithms for HR, SpO₂, and RI. The data was processed using the framework algorithms and the results were reviewed for accuracy by Dr. Edward Pugh. For the relative change alerts in the heart rate and blood oxygen saturation, annotations were done on the raw traces blind to the results of the relative change algorithms. For detecting pauses in breathing from the respiratory impedance signal however, the 24 hours were manually scanned viewing the raw trace along with the breath detection and the relative alert outputs and verifying the events along the way. While this may not be ideal, it was taken into consideration that the number of breaths taken by a premature baby over 24 hours is an unreasonable amount of data for one person to review and annotate.

6.2.1 - Relative change algorithm and breath pause detection

In this sample, first the HR and SpO₂ relative alerts are validated. There were 229 relative change alerts detected in the HR and 119 events detected in the SpO₂ signals over the course of the 24 hours. Clinically significant relative falls were manually determined by Dr. Pugh and compared with the relative falls detected by the algorithms. A high level of correlation was found. The algorithm detection correlated 97.8% for HR and 98.3% for SpO₂. Figure 47 illustrates the accuracies of the three relative alert algorithms.



Modified from Pugh et. al., 2013

Figure 47. Relative change algorithm compared to clinically annotated trace

For validating the breathing events, the same 24 hour sample was scanned using the streams display application checking if the breaths have been marked correctly in respect to the respiratory impedance waveform. The relative alert for RI is fired any time there is a missed breath in the time it took for the previous two complete breaths. The total number of breath pause events detected by the algorithm was 2896 and it was determined that the algorithm detected missed breaths in the RI waveform with an accuracy of 98.9%. It was found that this alert was raised very frequently. But as we will see in the results of the classification stage, many of the RI alerts are filtered out because they are isolated events that last a short amount of time, mostly recovering after missing only one breath. Below is a figure showing a section of the raw RI, the locations of the detected breaths, and the RI relative alert where multiple short alerts in quick succession are detected.



Figure 48. Section of raw RI along with detected breaths where multiple short RI relative alerts in quick succession are detected

6.2.2 - Comparison to absolute threshold alerts

When the relative change algorithms for HR and SpO₂ were in the process of initial validation, a comparison of relative alerts to the absolute alerts generated by bedside monitors was made [Thommandram et. al., 2013b]. A 24-hour sample of heart rate and blood oxygen saturation data was collected and processed using the designed algorithms. There were 339 relative falls in HR and 32 relative falls in SpO₂ over the course of the 24 hours. The relative changes were compared to the results of manually determined clinically significant falls in heart rate with relative falls detected by the algorithm. The accuracy of the algorithm was determined to be 99.45%.

Another result to note is the number of absolute threshold breaches found in the HR and SpO₂. In the 24 hour sample, 1117 absolute change alerts were detected in HR and 92 absolute change alerts were detected in SpO₂. What this means is because the algorithm is highly accurate in finding clinically relevant changes in HR and SpO₂, the number of alarms being fired at the bedside can potentially be

reduced by a factor of almost 3 times. This would be a vast improvement on its own over what is fast becoming an alarm riddled environment. But the framework has the potential to drop that down even more because individual alerts are correlated and only then is any kind of alert is raised.

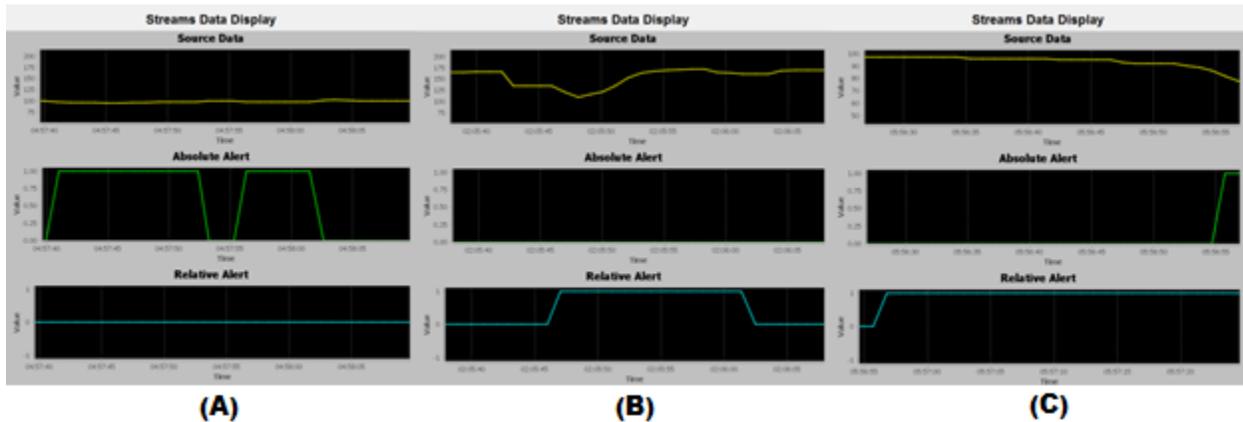


Figure 49. Relative to Absolute threshold comparison

Figure 49 above compares the performance of the relative change signal and the absolute threshold based signal currently used by bedside monitors. (A) shows a patient’s HR hovering around the threshold causing inaccurate alarms. (B) shows a dramatic drop in HR from 165 to 115 beats/minute in fewer than 10 seconds that goes undetected by the monitor using only absolute thresholds. (C) shows a drop in saturations that is detected 20 seconds in advance of the threshold breach.

6.2.3 - Spells classification algorithm

When we combine the three relative alerts into the classifier stage, we get the final report of all the episodes that were detected. The summary of all the episodes detected using the 24-hour sample from section 6.2.1 is shown in the table and figures below. This distribution of all the events detected by the classifier is shown graphically in Figure 50. A comparison of manually annotated spells events with the clinically significant spells detections made by the classifier is shown in Table 4.

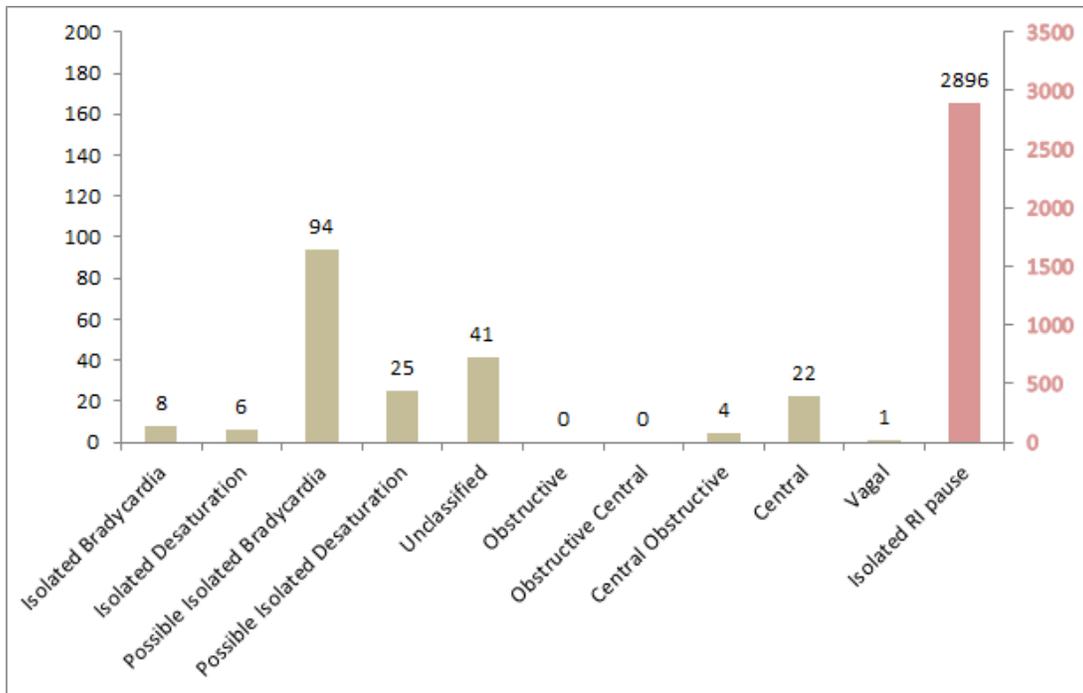


Figure 50. Distribution of all events detected by the classifier

Table 4. Summary of the accuracy of spells classifications by the algorithms

Type of spell	Classifier	Neonatologist	Accuracy
Central	22	21	95.46%
Central Obstructive	4	4	100%
Vagal	1	1	100%
Desaturation	6	6	100%
Overall accuracy:			98.9%

It can be seen in the above figure that the vast majority of events detected were isolated RI pauses. Because they were isolated events with no change in the HR or SpO₂, they can be considered clinically irrelevant. Including them in the final report is not considered as false detection of spells. The sensitive detection of breathing pauses is made apparent in the high number of possible isolated desaturations and possible isolated bradycardia events. These events that would be isolated HR or SpO₂ events if it not for a missed breath during the event. One or two missed breaths during a bradycardia or

desaturation are not defined as a spell but rather a slight variation of an isolated episode. However, an isolated episode can be defined as a spell if the conditions for a breach of the absolute threshold are met. It is important to note why desaturations were listed in the spells classification table. An isolated desaturation or isolated bradycardia can be classified as a spell if the relative fall coincided with an absolute threshold breach. The algorithms for detecting relative change in HR and SpO₂ also output a stream for the absolute alert. This alert file was given to the neonatologist to scan alongside the raw data in the viewer application. It was with this additional alert stream that the accuracy of the isolated events was determined. While strictly speaking the classifier Streams graph is not performing the sync on the absolute streams to make this classification right now, it is a minor addition that will be done as the next course of action.

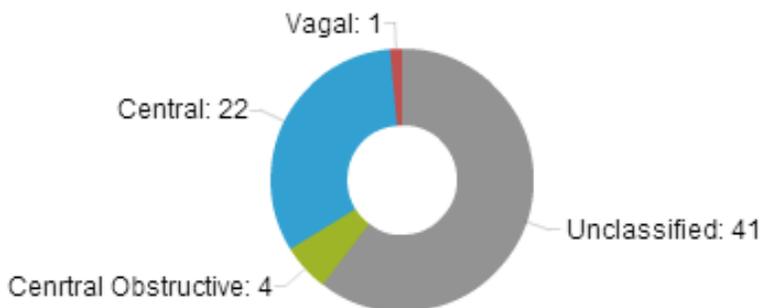


Figure 51. Distribution of all clinically significant multi-signal spells events

After filtering out the isolated and possible isolated events, with Figure 51 it can be seen that a total of 68 multi-signal episodes were detected. 22 central episodes were found, four central obstructive events, and one vagal episode. These episodes were checked by the neonatologist to confirm the classification. All the episodes were determined to be accurate except for one Central event. One central event that the classifier detected was considered unclassifiable by the human expert. For any case that the expert was unsure of the classification, the detailed event report containing the supporting

evidence was consulted. It was determined that tuning of parameters could lead to better classifications on a patient to patient basis.

An important result to note is the number of unclassified events that were found. These consist of event sequences that are not specified in the rules for classifications table. Based on some consistent event sequences being present, it is possible that these are legitimate spells and that some tuning of the rules will allow the classifier to recognize the events.

Chapter 7 - Conclusion

In a critical care environment, there is a need for continuous monitoring of the physiological state of patients. Current bedside monitors are capable of acquiring a great deal of data and displaying it on a screen, but even state of the art monitors offer very limited data integration and analysis for non-trivial clinical decision support. This thesis has presented the design of a framework for the correlation and real time classification of physiological data streams for critical care monitoring. The limitations of traditional algorithm development methods such as machine learning are described and the decision to develop clinically significant alerts using a blend of expert knowledge in the domain and pattern recognition programming based on clinical rules is highlighted. The methods are implemented for the task of automated detection and classification of spells. The performance of the classifier is evaluated and in doing so, the benefit of this framework as a novel approach in developing the next generation of intelligent alarms at the bedside is justified. This thesis concludes with a summary of findings and suggestions for future research.

7.1 - Research and findings

First, a literature review was performed to uncover the need for a framework that can perform multi-signal analysis of non-trivial conditions in real-time. A set of requirements for data acquisition and analytics for achieving the ability to correlate and classify physiological data streams in real-time is formulated based on the limitations in current systems described in Chapter 2. To demonstrate the practical application of the proposed framework, the classification of spells in the NICU was chosen as the task for implementation. Details on the specific events that the framework will be used to identify are described in Chapter 3 along with a survey of existing automated solutions and their performance in this task.

A rule based classification approach was chosen to best provide evidence on results to the medical professionals the framework is used to assist. Chapter 4 describes in detail the various components of the framework starting from identifying anomalies in individual streams, synchronising the streams based on timestamps, and then determining classifications based on the sequence of events in multiple signals.

The implementation of the generalised framework for the task of automated spells classification is explored in Chapter 5. The clinical rules for classifying spells were formulated by a neonatologist based on accepted physiological principles and an extensive literature review as part of his research work. Modules were designed to detect threshold breaches as well as relative changes in numerical data streams such as heart rate and blood oxygen saturation. For waveform data such as respiratory impedance, different modules were designed to detect anomalies representing abnormal breathing patterns. The individual events from the three streams are synced and correlated to generate episode profiles which are then assigned a classification based on a comparison to a truth table of various event sequences and their corresponding spells.

The output of the individual alert streams as well as the final classifier for a test sample consisting of 24 hours of patient data is shown in Chapter 6. Also, the results of the verification of the classifier outputs in comparison to data annotated by a neonatologist are discussed. The design of the framework and its implementation to the task of classifying spells demonstrated that rule based algorithms are a feasible approach in a field dominated by black box systems for developing next generation algorithms for complex multi-signal temporal events. The algorithms for detecting relative change events in individual signals outperformed the standard threshold breach detectors used in current bedside monitors in terms of reducing the risk of alarm fatigue in NICUs as well as generating alerts that were clinically significant. The rule-based algorithms for performing the spells classifications

performed comparatively to some black box techniques applied to specific conditions. However it is important to note the approach demonstrated in this thesis has the advantages of performing analyses for multiple conditions simultaneously without the need to retrain for each condition specifically. In addition, this methodology has the advantage of providing clinically relevant evidence to medical professionals regarding classifications. This will increase the clinicians' confidence in the algorithm and therefore has much greater potential than current automated systems for being deployed in a clinical setting.

In addition to the framework for designing algorithms, another contribution of this thesis is the Streams Data Display application. This program provides an interface to view live streaming data and the alert signals. It allows for changing algorithm thresholds and parameters for feature-condition in real-time. This is very useful as the alerts can be fine-tuned on a patient-to-patient basis very quickly. The application also is capable of loading retrospective data for viewing and playback, as well as for making annotations in the data. It is a valuable tool for validation testing of algorithms.

7.2 - Suggestions for future work

The research presented in this thesis is part of the larger Artemis project that is ongoing in the Health Informatics Research Laboratory (HIRL) at UOIT. The framework for temporal analysis of multi-signal events was designed to be generic and applicable across many clinical contexts. However, only a few modules for detecting specific events have been developed so far mainly because the initial development was driven by the case study of classifying spells.

One area for future work would certainly be to validate the output of the designed spells classifier in a clinical setting. As was mentioned, polysomnography is currently the gold standard method for classifying different kinds of spells. A comparison of the designed classifier with the results of a polysomnographic evaluation would show how close the system is to being used as a decision

support tool in the NICU. This work has actually been commenced and is a part of the research of Dr. Edward Pugh. A portable version of the Artemis system along with the real-time framework and spells classification algorithms has been designed to run in parallel with the polysomnography equipment. Details on this system are shown in Appendix A. The tests will provide some valuable data with which the algorithms parameters can be tuned for optimal performance.

There are improvements that can be made to the spells classification algorithms as well. Currently, only the relative change alerts along with the breathing alerts are being used to determine the type of spell. However, if the absolute alerts were synced and sent to the classifier block as well, more events can be accurately labeled. For example, it was determined that an isolated bradycardia or isolated desaturation event is considered a spell if and only if the relative fall was accompanied by a threshold breach. Since the threshold breaches are already being detected by the absolute alert modules, the simple addition to the rule set in the classifier block would enhance its functionality.

On a similar note, another level in the hierarchy can be added to make more complex classifications. Currently, individual alerts are detected and their sequences are grouped into an episode and that episode is classified. The episode is ruled by the starting of any alert and the ending of all alerts. However, there are conditions such as periodic breathing that this current design cannot determine because an episode of periodic breathing would encompass multiple episodes of RI pause classifications. Periodic breathing is a variation in breathing generally characterized by a pause in breathing for ten to 20 seconds followed by several rapid short breaths. In the current design, it was noted the large number of RI pauses that get detected. Adding another layer in the classifier that can analyse the frequency and duration of the isolated RI pause episodes can lead to a determination of periodic breathing.

Arguably the main area for future research is designing more rules suitable for automation with the designed framework. So far only the rules for the classification of spells have been implemented in the framework, but in the future there is a potential to have several algorithms running for detecting various conditions simultaneously. The process for adding algorithms to detect new conditions involves people with the clinical knowledge to determine the physiological changes that can be monitored and the behavior of these changes that characterize a classification. That clinical knowledge can then be translated into automatable rules which can then be programmed into the framework using the existing modules or by creating new modules. At the HIRL at UOIT, there are several projects that are working on this. There is research undergoing in classifying sleep/wake cycling in infants, retinopathy of prematurity, and describing novel premature infant pain profiles.

7.3 - Concluding remarks

The research in automated detection and classification of complex multi-signal events in physiological data has been constrained to specific solutions and single classifications. As well, most of work has focused on the research context and their implementation in a commercial clinical setting is limited. This thesis provides a framework to develop algorithms to make classifications of physiological signals in a way that is effectively presentable to medical professionals by employing a rule based approach and generating a report for each classification showing the supporting evidence.

This contribution was shown to accurately detect and classify a medical condition such as spells in neonates. It has the potential to continuously monitor for many conditions that would otherwise go unnoticed and to fundamentally change the notion of what critical care monitoring can provide as a clinical decision support tool.

Chapter 8 - References

- AAP Committee on Fetus and Newborn. (2003). Apnea, Sudden Infant Death Syndrome, and Home Monitoring. *Pediatrics*, 111(4), 914–917. Retrieved from <http://pediatrics.aappublications.org/content/111/4/914.full.html>
- AASM *Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications*. (2007).
- Ansari, S., Najarian, K., Ward, K., & Tiba, M. H. (2009). Extraction of Respiratory Rate from Impedance Signal Measured on Arm: A Portable Respiratory Rate Measurement Device. *2009 IEEE International Conference on Bioinformatics and Biomedicine*, 197–202. doi:10.1109/BIBM.2009.68
- Ballard, Chuck et al. IBM InfoSphere Streams: Assembling Continuous Insight in the Information Revolution. 2011. IBM Redbooks. 10 Oct. 2011. 25 Sept. 2012. Retrieved from <http://www.redbooks.ibm.com/abstracts/sg247970.html>
- Becker, K., Rau, G., Petermeyer, M., Kalff, G., & Zimmermann, H. (1994). Fuzzy Logic Approaches to Intelligent Alarms, (December).
- Belal, S. Y., Emmerson, A. J., & Beatty, P. C. W. (2011). Automatic detection of apnoea of prematurity. *Physiological Measurement*, 32(5), 523–42.
- Bloom, M. J. (1993). Techniques to identify clinical contexts during automated data analysis. *International journal of clinical monitoring and computing*, 10(1), 17–22.
- Blount, M., Ebling, M. R., Eklund, J. M., James, A. G., McGregor, C., Percival, N., ... & Sow, D. (2010). Real-time analysis for intensive care: development and deployment of the artemis analytic system. *Engineering in Medicine and Biology Magazine, IEEE*, 29(2), 110-118.
- Catley, C., Smith, K., Mcgregor, C., James, A., & Eklund, J. M. (2011). A framework for multidimensional real-time data analysis: a case study for the detection of apnoea of prematurity. *International Journal of Computational Models and Algorithms in Medicine*, 2(March), 16–37. doi:10.4018/jcmam.2011010102
- Chambrin, M. C., Ravaux, P., Chopin, C., Mangalaboyi, J., Lestavel, P., & Fourrier, F. (1989). Computer-assisted evaluation of respiratory data in ventilated critically ill patients. *International journal of clinical monitoring and computing*, 6(4), 211–5.
- Cirelli J, Graydon B, McGregor C, James A. “Analysis of Continuous Oxygen Saturation Data for Accurate Representation of Retinal Exposure to Oxygen in the Preterm Infant”, Courtney K, Shabestari O, Kuo A, editors. *Enabling Health and Healthcare Through ICT*. Vancouver: IOS Press; 2013. p. 126–31
- Drews, F. A. (2008). *Patient Monitors in Critical Care : Lessons for Improvement*, 1–13.

- Eddleman, D. W., Tucker, D. M., & McEachern, M. (1990). A patient monitoring system designed as a platform for application development. *International journal of clinical monitoring and computing*, 7(4), 233–40.
- Flower, A., Moorman, J. R., Lake, D. E., & Delos, J. B. (2010). Periodic heart rate decelerations in premature infants. *Experimental biology and medicine (Maywood, N.J.)*, 235(4), 531–8. doi:10.1258/ebm.2010.009336
- Fried, R., Gather, U., & Imhoff, M. (2001). Online pattern recognition in intensive care medicine. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*, 184–8. Retrieved from <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2243299&tool=pmcentrez&rendertype=abstract>
- Gather, U., Imhoff, M., & Fried, R. (2002). Graphical models for multivariate time series from intensive care monitoring. *Statistics in medicine*, 21(18), 2685–701. doi:10.1002/sim.1209
- Goldstein, B., McNames, J., McDonald, B. a, Ellenby, M., Lai, S., Sun, Z., ... Scلابassi, R. J. (2003). Physiologic data acquisition system and database for the study of disease dynamics in the intensive care unit. *Critical care medicine*, 31(2), 433–41. doi:10.1097/01.CCM.0000050285.93097.52
- Guez, A., & Nevo, I. (1996). Neural networks and fuzzy logic in clinical laboratory computing with application to integrated monitoring.
- Kamaleswaran, R., Thommandram, A., Zhou, Q., Eklund, M., Cao, Y., Wang, W. P., & McGregor, C. (2013, June). Cloud Framework for Real-time Synchronous Physiological Streams to Support Rural and Remote Critical Care. In *Computer-Based Medical Systems (CBMS), 2013 26th International Symposium on*. IEEE
- Khandoker, A. H., Gubbi, J., & Palaniswami, M. (2009). Automated scoring of obstructive sleep apnea and hypopnea events using short-term electrocardiogram recordings. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 13(6), 1057–67. doi:10.1109/TITB.2009.2031639
- Kickert, W. J. M., & Mamdani, E. H. (1978). Analysis of a Fuzzy Logic Controller, 1, 29–44.
- Koski, E. M. J., Mäkivirta, A., Sukuvaara, T., & Kari, A. (1990). Frequency and reliability of alarms in the monitoring of cardiac postoperative patients. *International Journal of Clinical Monitoring and Computing*, 7(2), 129–33. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/2373943>
- Koski, E. M., Sukuvaara, T., Mäkivirta, a, & Kari, a. (1994). A knowledge-based alarm system for monitoring cardiac operated patients--assessment of clinical performance. *International journal of clinical monitoring and computing*, 11(2), 79–83. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/7930853>
- Lee, H., Rusin, C. G., Lake, D. E., Clark, M. T., Guin, L., Smoot, T. J., ... Delos, J. B. (2012). A new algorithm for detecting central apnea in neonates. *Physiological measurement*, 33(1), 1–17. doi:10.1088/0967-3334/33/1/1

- McGregor, C., Smith, K. P., & Eklund, J. M. (2009). A survey of recent physiological monitoring and transmission to support the service of critical care. 2009 22nd IEEE International Symposium on Computer-Based Medical Systems, 1–7. doi:10.1109/CBMS.2009.5255404
- McGregor, C., Catley, C., James, A., & Padbury, J. (2011). Next Generation Neonatal Health Informatics with Artemis. Medical Informatics Europe (MIE) 2011. Stud Health Technol Inform. 169:115-9.
- McGregor, C., Catley, C., & James, A. (2012). Variability Analysis with Analytics Applied to Physiological Data Streams from the Neonatal Intensive Care Unit. Proc 2012 25th IEEE International Computer-Based Medical Systems (CBMS 2012), pp1-5.
- McGregor, C., James, A., Eklund, J.M., Sow, D., Ebling, M., Blount, M., (2013), “Real-time Multidimensional Temporal Analysis of Complex High Volume Physiological Data Streams in the Neonatal Intensive Care Unit”, The 14th World Congress on Medical and Health Informatics (MedInfo, 2013), Copenhagen, p 362-6
- Meredith, C., & Edworthy, J. (1995). Are there too many alarms in the intensive care unit? An overview of the problems. *Journal of advanced nursing*, 21(1), 15–20. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/7897067>
- Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63, 81–97. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/8022966>
- Mirza, M., Gholamhosseini, H., & Harrison, M. J. (2010). A fuzzy logic-based system for anaesthesia monitoring. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, 2010*, 3974–7. doi:10.1109/IEMBS.2010.5627987
- Moody, B., & Mark, R. G. (1996). A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring, 657–660.
- Mylrea, K. C., Orr, J. A., & Westenskow, D. R. (1993). Integration of Monitoring For Intelligent Alarms in Anesthesia: Neural Networks - Can They Help?
- Naik, T., Bressan, N., James, A., & McGregor, C. (2013). Design of temporal analysis for a novel premature infant pain profile using artemis. *Journal of Critical Care*, 28(1), e4. doi:10.1016/j.jcrc.2012.10.024
- Premature babies | March of Dimes. (n.d.). Retrieved from <http://www.marchofdimes.com/baby/premature-babies.aspx>
- Pugh, E., Thommandram, A., Ng, E., McGregor, C., Eklund, M., Narang, I., ... James, A. (2013). Classifying neonatal spells using real-time temporal analysis of physiological data streams—algorithm development. *Journal of Critical Care*, 28(1), e9. doi:10.1016/j.jcrc.2012.10.033

- Saeed, M., Lieu, C., Raber, G., & Mark, R. G. (2002). MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in cardiology*, 29, 641–4. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/14686455>
- Sale, S. M. (2010). Neonatal apnoea. *Best Practice & Research Clinical Anaesthesiology*, 24(3), 323–336.
- Samuels, S. I. (1986). An alarming problem. *Anesthesiology*, 64(1), 128. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/3942327>
- Schluter, T., & Conrad, S. (2010). An Approach for Automatic Sleep Stage Scoring and Apnea-Hypopnea Detection. *2010 IEEE International Conference on Data Mining*, 1007–1012. doi:10.1109/ICDM.2010.60
- Sugiura, T., Mizushina, S., Kimura, M., Fukui, Y., & Harada, Y. (2009). A fuzzy approach to the rate control in an artificial cardiac pacemaker regulated by respiratory rate and temperature: A preliminary report. Retrieved from <http://informahealthcare.com/doi/abs/10.3109/03091909109016207>
- Sukuvaara, T., Koski, E. M., Mäkivirta, a, & Kari, a. (1993). A knowledge-based alarm system for monitoring cardiac operated patients--technical construction and evaluation. *International journal of clinical monitoring and computing*, 10(2), 117–26. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/8366312>
- Thomas, L. J., Clark, K. W., Mead, C. N., Ripley, K. L., Spenner, B. F., & Oliver, G. C. (1979). Automated cardiac dysrhythmia analysis. *Proceedings of the IEEE*, 67(9), 1322–1337. doi:10.1109/PROC.1979.11450
- Thommandram, A., Eklund, J. M., & McGregor, C. (2013a). Detection of Apnea from Respiratory Time Series Data Using Clinically Recognizable Features and kNN Classification *.
- Thommandram, A., Pugh, J. E., Eklund, J. M., McGregor, C., & James, A. G. (2013b). Classifying neonatal spells using real-time temporal analysis of physiological data streams: Algorithm development. In *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE* (pp. 240-243). IEEE.
- Tsien, Christine L. (2000a). Event discovery in medical time-series data. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*, 858–62. Retrieved from <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2243881&tool=pmcentrez&rendertype=abstract>
- Tsien, C L, Kohane, I. S., & McIntosh, N. (2000). Multiple signal integration by decision tree induction to detect artifacts in the neonatal intensive care unit. *Artificial intelligence in medicine*, 19(3), 189–202. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/10906612>
- Tsien, Christine L. (1997a). An A n n o t a t e d Data Collection System to Support Intelligent Analysis of Intensive Care Unit Data, 111–121.
- Tsien, Christine L. (1997b). Reducing False Alarms in the Intensive Care Unit : A Systematic Comparison of Four Algorithms, 22(6), 1997.

- Tsien, Christine L. (2000b). TrendFinder : Automated Detection of Alarmable Trends Chairman ,
Departmental Committee on Graduate Students TrendFinder : Automated Detection of Alarmable
Trends by.
- Wang, K., Kohane, I., Bradshaw, K. L., & Fackler, J. (1996). A real time patient monitoring system on the
World Wide Web. *Proceedings : a conference of the American Medical Informatics Association / ...
AMIA Annual Fall Symposium. AMIA Fall Symposium*, 729–32. Retrieved from
<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2233054&tool=pmcentrez&rendertype=abstract>
- Waterson, C. K. (1988). Anesthesia Monitoring DEVELOPMENT DIRECTIONS IN INTEGRATED ANESTHESIA
MONITORING The role of monitoring in anesthesia is to integrated approach to the development
of anes seemingly unrelated juxtaposition . indicate the need for further study of the, 1765–1766.
- Yoshizawa, M. (1992). An Automatic Monitoring And Estimation Tool For The Cardiovascular Dynamics
Under Ventricular Assistance. *Proceedings of the Annual International Conference of the IEEE
Engineering in Medicine and Biology Society*, 364–366. doi:10.1109/IEMBS.1992.595595
- Zhang, Y. (2003). Real-Time Analysis of Physiological Data and Development of Alarm Algorithms for
Patient Monitoring in the Intensive Care Unit.
- Zong, W., Moody, G., & Mark, R. (1999). Reduction of False Blood Pressure Alarms by use of
Electrocardiogram Blood Pressure Relationships.

Appendix A – Polysomnography Validation

A.1 - Experimental Design

At present, the gold standard for detection and classification of neonatal spells is polysomnography. This is an expensive process involving the patient spending the night in a sleep laboratory connected to devices recording several physiological signals and being monitored by trained professionals. Polysomnography is not practical in the NICU environment so it is reserved for the most difficult cases to diagnose.

As part of the validation process for the framework and case study algorithm, a portable version of the Artemis platform to be placed in the room of the sleep study was created. This portable system consists of two laptops on a private network isolated from the hospital's network. One laptop is responsible for the collection of physiological data from the medical monitor and displaying a status on screen. The second laptop is responsible for storing that data into a DB2 database and also running the data through Streams and the framework algorithms. Figure 52 below shows the layout and functionality of the portable system.

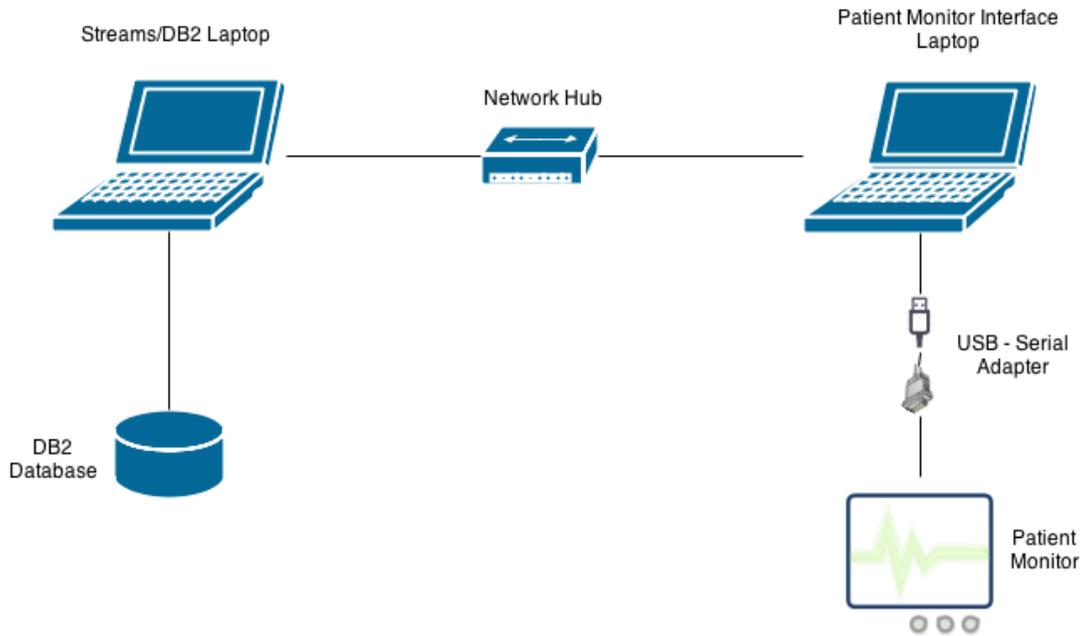


Figure 52. Layout of equipment for collecting data during polysomnography

The Patient Monitor interface laptop operates on Microsoft Windows and runs the software to interface with patient monitors. The laptop is physically connected to the device via a USB-serial adapter. It captures data from the monitor and serves the data over a network socket. The laptop also contains configuration applications to set up data collection parameters depending on which device is connected. When setting up for a sleep study or polysomnography, custom software described in the next section is launched to serve as a real-time system status display.

The Streams/DB2 laptop operates on RedHat Enterprise Linux and runs the IBM Infosphere Streams software and the DB2 database software. In Streams, a graph is started to connect to a network socket and receive data from the windows laptop. Another graph forms a connection to a DB2 database and writes the raw data as well as the output of any algorithm graphs that may be running to DB2 tables using secure authenticated connections. There are scripts on the machine that automate some functions of streams such as starting and stopping graphs and also scripts that automate

interactions with the database. These scripts can be triggered through the custom software residing on the other laptop.

The entire system is run on an offline network that does not have to be connected to the hospital's network to function. The laptops are configured with static IP addresses and the network hub manages the local network. This provides a security assurance to the hospital that the data cannot accidentally leave the hospital through the Internet. The data resides in the DB2 database with enterprise level security measures and the export of data from the system requires the use of a secure storage medium such as an encrypted hard drive.

Once the sleep study has completed, the data from the polysomnography is created into a report for the attending physician. This report contains a detailed log of every event that took place over the course of the study, including desaturations and episodes of apnoea. This truth data will be compared to the report generated by the designed framework for concordance.

A.1.1 - Sleep Study Status Application

Since sleep studies are not frequently performed in the hospital, it was important to make sure our equipment was functioning without flaws during the sleep study nights. As I would not be permitted to stay in the room during the study, an application was developed to run on one of the laptops that would remain open for the sleep technician to be alerted if data collection has stopped for any reason. The figure below shows a screenshot of the application.

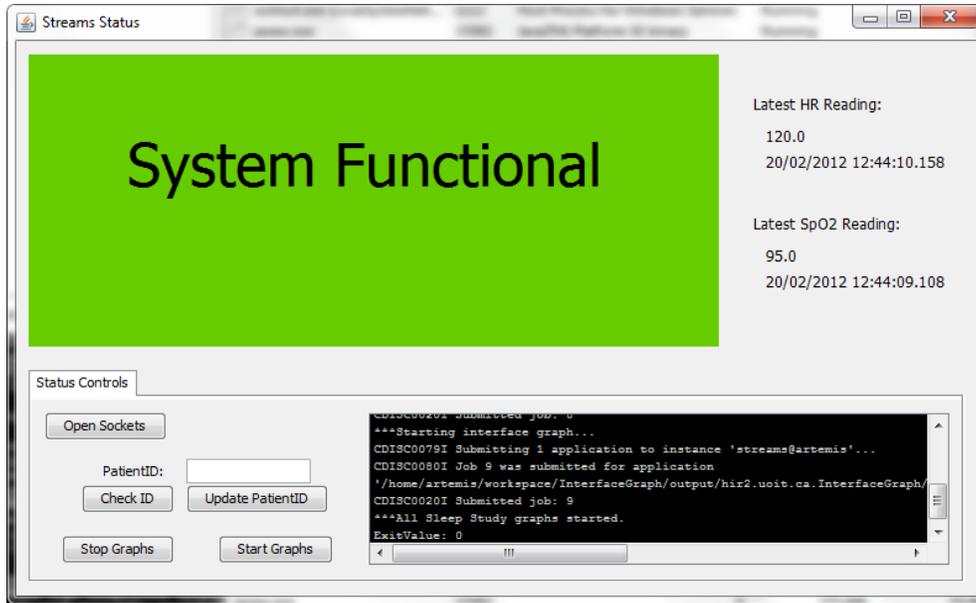


Figure 53. Sleep Study Status application showing system is operational

The application displays the latest timestamp and value of the heart rate (HR) reading and the blood oxygen saturation (SpO₂) reading. These give an indication of whether the data collection is keeping up. In addition, in the event that there has been an error writing to the database, the big green rectangle will turn red and read “System ERROR” as shown in the figure below and provide a contact number to notify. This allows us to diagnose the issue immediately and bring the system back up with minimal down time.

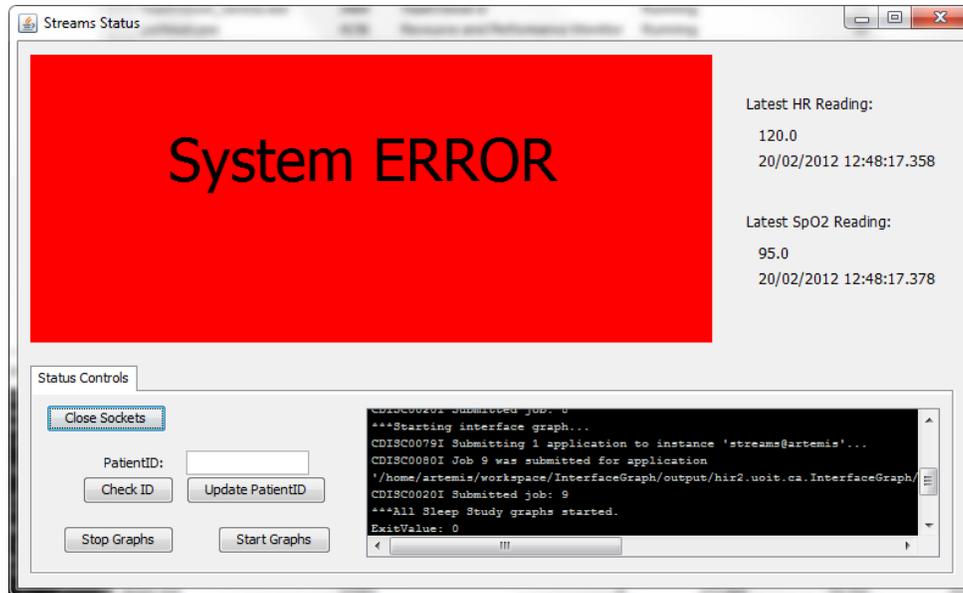


Figure 54. Sleep Study Status application showing a system error

The sleep study status application also provides the functionality to start and stop the Artemis Streams graphs which reside on the second laptop. This is done by designing several shell scripts which execute commands on the networked machine through a secure shell connection (SSH). This reduces the time needed to restart the system by a drastic amount. Another feature of the application is to update the Patient ID used for storage in the database directly from the application. Traditionally this requires a row modification on a DB2 database table, but by using Java JDBC it was possible to automate the functionality from within the application. Again, this greatly reduces the time required to set up the environment for a sleep study.

A.2 - Preliminary Results

Being sufficiently satisfied with the results of the neonatologist's validation, phase 2 of the validation process was to perform the comparison with the gold standard of polysomnography. The test was performed by collecting patient data side by side with the polysomnography equipment and running the data through the framework algorithms.

Data was successfully collected for one sleep study as of this writing. The truth data consisted of a detailed table of the times and types of events during the course of the night. Below is a table showing the different kinds of events and the number of each detected. A typical polysomnography records many more types of events, but this table only displays types of events similar to the types being detected by the framework’s classifier.

Table 5. Truth data from polysomnography listing the types of events detected and the number of occurrences of each event

Type of Event	Number of occurrences
Oxygen Desaturation	168
Hypopnea	42
Obstructive Apnea	10
Mixed Apnea	6
Central Apnea	143
Periodic Breathing	13

In comparison, the output of the framework report contained no multi-signal spells classifications. Below is a table showing the different events and the number of each detected.

Table 6. Output from the classifier algorithm listing the types of events detected and the distribution of occurrences

Total events detected	531
Single-signal events:	
Isolated RI pause	463
Isolated Bradycardia	4
Isolated Desaturation	2
Multi-signal events:	
Possible Isolated Bradycardia	17
Possible Isolated Desaturation	12
Unclassified	33

At first observation, there seems to be a poor correlation between the two results. But with the assistance of Dr. Pugh, it was noticed that there is a pattern in the way the SpO₂ alerts are firing. The framework seemed to catch very few of the desaturations and the ones that did line up with the truth data seemed to be delayed by a few seconds. Such a discrepancy would certainly cause the proposed algorithm to malfunction as not detecting desaturation events will inevitably lead to not detecting spells events.

Upon further investigation into what could cause the algorithm to miss desaturations, it was observed that even the raw SpO₂ signal captured by the Artemis framework did not show as many visual desaturations as the raw SpO₂ captured by the polysomnography equipment. It was this revelation that led to the discovery that there is an averaging feature on the bedside monitor that we were capturing data from. This parameter affects the raw SpO₂ values being collected from the patient monitor. It was set very high and so the raw signal was already greatly smoothed and detecting dramatic falls was all but impossible.

While the initial testing did not yield any positive correlation, we did learn a lot about more about the format and contents of the polysomnography result data. As well, many issues were discovered regarding differences in monitor settings between patient monitors in the research laboratory and patient monitors in the NICU.