A Distributed Architecture for the Monitoring and Analysis of Time Series Data

by

Ruairi Donagh O'Reilly, B.Sc.

THESIS

Presented to the College of Science, Engineering and Food Science
National University of Ireland,
Cork

for the Degree of

Doctor of Philosophy

September 15, 2015

Declaration

In signing this declaration, I am confirming, in writing, that the submitted work is entirely my own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

Name:

Signed:

Date:

Abstract

It is estimated that the quantity of digital data being transferred, processed or stored at any one time currently stands at 4.4 zettabytes (4.4×2^{70} bytes) and this figure is expected to have grown by a factor of 10 to 44 zettabytes by 2020 [1]. Exploiting this data is, and will remain, a significant challenge. At present there is the capacity to store 33% of digital data in existence at any one time; by 2020 this capacity is expected to fall to 15%. These statistics suggest that, in the era of *Big Data*, the identification of important, exploitable data will need to be done in a timely manner.

Systems for the monitoring and analysis of data, *e.g.* stock markets, smart grids and sensor networks, can be made up of massive numbers of individual components. These components can be geographically distributed yet may interact with one another via continuous data streams, which in turn may affect the state of the sender or receiver. This introduces a dynamic causality, which further complicates the overall system by introducing a temporal constraint that is difficult to accommodate.

Practical approaches to realising the system described above have led to a multiplicity of analysis techniques, each of which concentrates on specific characteristics of the system being analysed and treats these characteristics as the dominant component affecting the results being sought. The multiplicity of analysis techniques introduces another layer of heterogeneity, that is heterogeneity of approach, partitioning the field to the extent that results from one domain are difficult to exploit in another.

The question is asked can a generic solution for the monitoring and analysis of data that: accommodates temporal constraints; bridges the gap between expert knowledge and raw data; and enables data to be effectively interpreted and exploited in a transparent manner, be identified?

The approach proposed in this dissertation acquires, analyses and processes data in a manner that is free of the constraints of any particular analysis technique, while at the same time facilitating these techniques where appropriate. Constraints are applied by defining a workflow based on the production, interpretation and consumption of data. This supports the application of different analysis techniques on the same raw data without the danger of

incorporating hidden bias that may exist.

To illustrate and to realise this approach a software platform has been created that allows for the transparent analysis of data, combining analysis techniques with a maintainable record of provenance so that independent third party analysis can be applied to verify any derived conclusions.

In order to demonstrate these concepts, a complex real world example involving the near real-time capturing and analysis of neurophysiological data from a neonatal intensive care unit (NICU) was chosen. A system was engineered to gather raw data, analyse that data using different analysis techniques, uncover information, incorporate that information into the system and curate the evolution of the discovered knowledge.

The application domain was chosen for three reasons: firstly because it is complex and no comprehensive solution exists; secondly, it requires tight interaction with domain experts, thus requiring the handling of subjective knowledge and inference; and thirdly, given the dearth of neurophysiologists, there is a real world need to provide a solution for this domain.

Acknowledgements

I would like to thank my supervisor, Prof. John Morrison for his encouragement and support throughout my time in the Centre for Unified Computing. Also, my examiners, Dr Carolyn McGregor and Dr John Herbert, for their time, constructive feedback and interesting discussion.

A number of thanks are due to my colleagues in the CUC: Phil, for his guidance, assistance and mentoring in my early days; Brian, for accommodating and assisting with technical work; and Stefan for acting as a sounding board for my many ideas and sharing his extensive knowledge on all things technical.

I would like to thank Michelle for her patience, proofreading and perseverance with me while I completed the PhD. Also a note of thanks to Dr Rainville and Dr Cotter for their interesting conversation over the years. Finally, a last minute thanks to my brother Finbarr, for stepping in with his proofreading skills at the last minute.

Contents

List of Figures

List of Tables

Publications arising from this work

[1] P. D. Healy, R. D. O'Reilly, G. B. Boylan, and J. P. Morrison, "Web-based remote monitoring of live EEG", in *Proceedings of the 12th International Conference on E-Health Networking, Applications and Services*, (Lyon, France), July 2010.

[2] R. D. O'Reilly, P. D. Healy, G. B. Boylan, and J. P. Morrison, "An agent framework for the analysis of streaming physiological data", in *Proceedings of the 3rd International ICST Conference on Electronic Healthcare*, (Casablanca, Morocco), December 2010.

[3] P. D. Healy, R. D. O'Reilly, G. B. Boylan, and J. P. Morrison, "Interactive annotations to support collaborative analysis of streaming physiological data", in *Proceedings of the 24th International Symposium on Computer-Based Medical Systems*, (Bristol, UK), June 2011.

[4] R. D. O'Reilly, J. P. Morrison, and C. McGregor, "A system for the transmission, processing and visualisation of EEG to support Irish neonatal intensive care units", in *Electrical & Computer Engineering (CCECE), 2012, 25th IEEE Canadian Conference on* Electrical & Computer Engineering, (Montreal, Canada), pp. 1–5, IEEE, June 2012.

[5] R. D. O'Reilly, D. Power, P. D. Healy, J. P. Morrison, and G. B. Boylan, "Scrutiniseit: A search-based approach to EEG seizure detection", in *eTELEMED 2013, The Fifth International Conference on eHealth, Telemedicine, and Social Medicine*, (Nice, France), pp. 310–313, March 2013.

Chapter 1

Introduction

Technologies for the monitoring and analysis of data are rapidly developing. New paradigms, algorithms and techniques for the acquisition and processing of data are continually emerging; examples include MapReduce, BigTable and MongoDB. These technologies are used in domain-specific contexts. Advances, of a practical or theoretical nature, are subsequently applied within that context as they are discovered.

There is a tendency to adopt a domain-specific approach to the application of emerging tools, which makes it difficult to subsequently apply those techniques in a broader context. This phenomenon is corroborated by Wagstaff, who states that in the field of machine learning, the pursuit of algorithmic performance has taken precedence over the communication of results to the domains from which the data has originated [3].

Similarly the development of monitoring and analysis systems is seen as a means to an end with the design and development of a system being justified by its application in a specific domain. The monitoring and analysis of data is often a secondary concern with the proposed analysis technique being the primary. This is evident throughout the literature and can be seen in systems that assist with medical decision support [4, 5, 6], act as a tool for knowledge discovery [7, 8, 9] or those used to demonstrate the suitability of a paradigm to the task [10, 11, 12, 13].

The isolated treatment of similar concepts in different fields has a number of disadvantages. It is wasteful, as resources are consumed to explore concepts which may have already been explored in a similar field. It discourages reusability, as the scope of the advancement is limited to the field from which it arises. It is self-perpetuating, as each advancement increases the separation between the fields. Consequently, the tools and techniques that are available to one field are denied to another.

The limitations of this isolated approach are avoidable and constitute missed opportunities to further our understanding of the domain in question and any other field that

may benefit from the transfer of knowledge acquired outside that field. This dissertation sets out to illustrate that through a combination of careful engineering and by viewing the monitoring and analysis of data from a holistic perspective, encompassing a comprehensive workflow analysis, distinguishing between data capture and analysis and incorporating an auditable trail of information generation, it is possible to successfully address these issues and enable each advance to deliver benefit across multiple domains.

Moreover, given that feedback loops are characteristic of many application domains and are used typically to drive a system towards convergence, it is imperative that these loops are transparently applied so that appropriate convergence paths result.

1.1 Monitoring And Analysis Of Data

Monitoring involves the acquisition of data from one or more sources over a period of time. Analysis is the means through which the acquired data can be interpreted. The data that is being monitored, the means through which it is monitored and the analysis that takes place determine the design, development and performance of a monitoring and analysis system. Central to such a system are three components, the *Data Producer* (DP), the *Data Consumer* (DC) and the *Interpreting Component* (IC). The DP is the entity generating data or the source from which data is acquired. The IC is the entity that takes data from a data source, interprets it and provides the resulting information to the DC. The DC is the entity that subsequently acts on that information.

The DPs, ICs, and DCs together with a description of their relationships, form part of a system, which will be called the *Interpreting Environment* (IE), in which data is monitored, analysed and the results acted on. An IE will be required to:

i) Bridge the gap between expert knowledge and raw data, that is to enable experts to access data from which knowledge is synthesised and to allow the reincorporation of that knowledge for subsequent use through the provision of appropriate feedback loops.

ii) Maintain an expert knowledge base, enabling the evolution of that knowledge base and preserving the integrity of both pre-existing knowledge and newly acquired data. This is a complex challenge, whose solution may involve a combination of approaches including algorithms to detect nonsense data and peer review to inject a degree of quality assurance.

iii) Accommodate the unique perspective of each DC. This may necessitate the development of appropriate consumer-specific interfaces.

iv) Dynamically accommodate changes in the rate of data production, processing and consumption, which in turn may lead to real-time or near real-time system operation.

v) Dynamically incorporate varying numbers of DPs, ICs and DCs.

Without loss of generality, a DP and a DC can be viewed as independent entities, however the IC relies heavily on characteristics of each. As such, it is a challenge to create efficient and effective ICs that are capable of generically supporting the complex relationships that may exist between arbitrary DPs and DCs.

Therefore, an IE can be a complex entity and may include components that handle physical communications, provide data storage, embody analysis tools, incorporate expert knowledge and accommodate temporal constraints resulting from the need to process ephemeral data in a timely manner.

Existing systems (see Section 1.2) that may be viewed as IEs (as defined here) tend to be point solutions, solving one particular problem without regard for related problems, and as such fall short of the ideal generic IE. They are limited in the types of data they can handle, the quantity of data they can process, the types of analysis techniques they can employ and the domains to which their solution can be readily employed.

A few take a generic perspective, but do so in a manner that does not allow their implementations to be readily compared to other similar systems. They do not provide the means to reduce the complexity associated with the configuration of systems of that nature. Similarly the systems do not support the requirements necessitated for a complete monitoring and analysis system [4, 8], as can be realised by the proposed IE.

Related work advances particular techniques, and approaches, in specific contexts. It does not do so in a generic manner. This limits the proposed advance to that context failing to enable the multitude of alternate but closely related domains to take advantage of the knowledge derived [3].

Consequently, opportunities are lost to incorporate, share and make use of expert data across these diverse domains and peer support for the verification and validation of interpretations cannot be leveraged. Tapping this valuable resource is itself an important motivation for the creation of a generic IE.

1.2 Related Work

Active areas of related research include the following:

1. Stream Computing (see Section 1.2.1)

2. Complex Event Processing (see Section 1.2.2)

3. Remote monitoring and analysis systems (see Section 1.2.3)
4. Decision support (see Section 1.2.4)
5. Knowledge discovery (see Section 1.2.5)
6. Complex Software Systems (see Section 1.2.6)
7. Agent Frameworks (see Section 1.2.7)

While each research area is different by definition, there is a significant overlap in the design and development of systems for realising them. In reviewing application-specific instances, it is noted that many include functionality that is outside the scope of a monitoring and analysis platform, but benefits the domain in which it is applied. Functionality that enhances an IE and has the potential for broad application should be considered for inclusion.

The monitoring and analysis of data is closely related to fields that deal with the timely processing of data. Two fields of study that have made significant contributions to the ephemeral processing of data are Stream Computing (see Section 1.2.1) and Complex Event Processing (see Section 1.2.2), both of which influenced the design and development of the IE.

1.2.1 Stream Computing

Stream Computing is a computing paradigm based on the processing of data streams in a manner that tries to minimise the quantity of storage that is required, this enables the processing of large volumes of data when compared to the quantity of storage available [14, 15]. Processing of data streams in real time enables real-time analytics and information to be provided. The underlying data model takes a transient approach to data acquired, as the quantity of data is so great that persisting it to disk would be impractical. Instead the query running against the data is altered as data is processed, after which the data is discarded.

The database management system (DBMS) used in a Stream Computing platform emphasises different characteristics than that of a traditional DBMS. It is based on the *DBMS-active, human-passive (DAHP)* model [16], whereby data is acquired from external sources and the DBMS's role is to act on that data, as opposed to the *human-active, DBMS-passive (HADP)* model [16], whereby the DBMS is treated as a repository of data and the initiation of queries is the human's role. This enables the platform to react to data analysis in real time and return insights from that data, both of which are relevant to a monitoring and analysis system.

Stream Computing is seen as a potential means of handling massive data generation and has been applied to a number of domains including healthcare. For instance, in [17] a Stream Computing platform for the analysis of physiological data in real time is presented. It is used to turn data processed into information for medical professionals, to provide an infrastructure for processing large quantities of data and returning valuable information in an online manner [17].

The platform has been deployed to a NICU in a real clinical setting, where the intent is to develop "a Decision Support System to assist in detecting subtle changes in physiological data" [18]. Stream Computing provides a means of enabling data driven knowledge discovery of streaming physiological data at scale, which is a promising way to further our understanding of human ailments [19]. Stream computing shares a number of the characteristics that embody a monitoring and analysis system and as such a large overlap exists which will have a direct influence on the development of the IE.

1.2.2 Complex Event Processing

The perspective taken by Complex Event Processing (CEP) [20] is different to that of Stream Computing. Continuous data streams are viewed as a stream of events in which identical patterns can occur that are indicative of higher level events occurring in the real world. There is a separation between the providers and consumers of events and a means of querying relationships between events is also provided. This enables the development of dynamic systems that are heavily interconnected and capable of operating at scale [20]. As such there is a significant overlap with the envisioned IE and so a number of systems that utilise CEP were reviewed.

CEP in relation to the acquisition and processing of RFID data streams is discussed in [21]. A number of challenges affecting the management and processing of the data are outlined: i) Duplicates within the data stream require filtering; ii) RFID observations have to be aggregated in order to derive meaningful information from them; iii) The RFID data is temporal, streaming and high volume, and so requires a suitable means of processing that data.

An event oriented framework was developed that: enabled aggregation through the generation of complex events (compounded primitive/complex events); allowed the specification of events with temporal constraints; and the detection of events with temporal constraints. This enabled the three challenges to be addressed and was shown to be both efficient and scalable [21].

In [22] an application for the timely processing of data is referred to as an information flow processing system (IFP). A survey of such systems is presented with the two main

models that contribute to the application being identified as the data stream processing model and the complex event processing model. It is argued that neither of these models sufficiently address the needs of an IFP engine, as a hybrid of the two is necessitated by a "fully-fledged" implementation.

An IFP engine is defined as a "a tool that operates according to a set of processing rules which describes how incoming flows of information have to be processed to timely produce new flows as outputs" [22]. The key characteristics emphasised include: real-time processing of information; a language in which to express how information is processed; and a scalable solution to be capable of handling geographically dispersed nodes that have to cooperate [22]. As such an IFP shares many of the characteristics seen in a distributed system for the monitoring and analysis of data.

1.2.3 Remote Monitoring And Analysis Systems

Remote monitoring and analysis systems are defined as those that are concerned with the analysis of data and the mitigation of geographic constraints. They are used to provide remote analysis, thereby addressing a need for timely data analysis, and/or a means to achieve greater efficiencies. They constitute the foundations of modern monitoring and analysis platforms and their architecture closely resembles that outlined in Section 1.1. There are numerous examples of remote monitoring and analysis systems, a number of which discussed are of particular relevance to the use case.

An early method of remote monitoring of Electroencephalogram (EEG) and other physiological data involved analogue transmission over telephone lines [23, 24, 25, 26]. This approach involves the modulation of EEG signals to audible frequencies before transmission over the public switched telephone network [27]. Although these systems allow for real-time monitoring, they are constrained by poor bandwidth to a limited number of channels, incur high telephone charges when used long-distance, and suffer from signal degradation due to noise.

An electrophysiological data monitoring and testing device for neurological intensive care units was presented in [28]. The device, which is mounted at the patient's bedside, does not sample continuously; instead it records segments of data at predetermined intervals. The resulting segments can then be viewed using a web browser within the hospital intranet. It also supports functionality for the further processing of the acquired signals, including spectral analysis, burst counting and loose lead detection.

A neurosurgery intensive care unit (ICU) monitoring system developed at UCLA [29] provided a web interface to physiological signal data. Remote monitoring of data is facilitated through the dynamic generation of plots that are viewed as images embedded in

HTML pages. The system lacks the immediacy available through the use of more advanced client-side technologies such as Java or Flash. Although the system supports EEG, the extent to which EEG specific features (such as filtering and montaging) are supported is unspecified.

TeleEEG [30] is a remote monitoring system that allows EEG data to be stored in a standardised format. Files can then be transferred easily between TeleEEG instances at geographically dispersed medical centres. An online viewing facility is provided via the dynamic generation of plots that are viewed as images embedded in web pages. This viewing system is similar to the UCLA system, but is somewhat more advanced in that support for EEG-specific features such as montaging is provided. However, the system does not provide real-time monitoring capability.

A system for the remote diagnosis of epilepsy through the use of a high-bandwidth, point-to-point link between Japan and the US is presented in [31]. The system allows segments (5–20 minutes) of ictal and interictal EEG data to be transferred quickly between locations, along with corresponding video files of patient activity and medical image data such as MRI, CT and PET. Teleconferencing capabilities are also provided in order to facilitate collaborative diagnosis. The aim of this system is to support telepresence at case conferences rather than real-time remote monitoring and analysis.

The BRIAN system [32] allows for interactive, but non-real-time, remote monitoring using compressed digital transmission. The system can adjust the quality of the signal to take advantage of available bandwidth. Data are viewed using custom viewing software on the client. Caching is not performed on the client; each screenful of data is fetched on demand. The possibility of performing computationally intensive analysis of captured EEG data using Grid technologies has been examined [33]. The system has also been extended [34] to support web-based review using a Java applet, eliminating the need to install custom software and supports the display of annotations that were made prior to data acquisition (data uploaded to server). The web-based system, although also non-real-time, is fully featured and incorporates advanced features such as spike and eye-blink detection. Limited caching is performed through prefetching of the most probable next screenful of data.

The e-babies project [6] began as a system for integrating and analysing real-time data collected from the various monitors found in the NICU. The system was then expanded to provide data warehousing and remote monitoring [35, 36, 37] via a web interface. The monitoring of both real-time physiological data and video streams is supported. Java applets are provided for viewing incoming physiological data streams. The video quality and data sampling rates can be adjusted automatically based on available bandwidth in order to reduce congestion. Alternatively, signal quality can be maximised if potential delays in transmission are acceptable. However, the system does not, as yet, support EEG.

In many cases, these systems provide a means to bridge the gap between expert knowledge and raw data but their capacity to do so for more complex bio-signals, such as EEG, in real time is questionable. Furthermore, none of the aforementioned systems provide a means to collate, or incorporate, the expert knowledge garnered during the analysis of data. As such, these systems are not capable of fulfilling the requirements of an IE.

1.2.4 Decision Support

A Decision Support System (DSS) is a computer-based information system that supports decision making activities. It typically combines data acquired by a monitoring system with a repository of expert knowledge (or an algorithmic representation of that knowledge) that enables assistance to be provided to the expert carrying out the analysis. This assistance can be in the form of a query submission to a knowledge base or the continual interrogation of data as it is being acquired. A DSS can also be used to assist with the retrospective analysis of data.

DSS's are a rational progression in the evolution of a monitoring and analysis platform. They enable expert knowledge to be utilised in a reusable manner. This is beneficial to all parties as the DSS provides assistance to the analysis phase while the expert knowledge contained within it is simultaneously undergoing a process of verification and validation through its use. Furthermore in some instances, DSS's are capable of adding newly acquired expert knowledge (either through the actions of a human expert or the automated processing of the data) to the system's knowledge repository. A growing knowledge repository can in some cases increase the overall performance of the system by increasing the quantity of pertinent data available for classification and comparison.

In [4] a platform for medical decision support in the ICU is presented, referred to as *Intensive Care Agent Platform* (ICAP). Its architecture is middleware focused, enabling the distribution of agents to multiple workstations seamlessly. Simplified integration, and scalability in the platform are realised by adopting an agnostic approach to the underlying operating system (implemented in Java EE and CORBA) and an agent-based architecture. This enables the execution of medical decision support agents and their distribution across the workstations that ICAP has access to. Eight requirements outlined for the platform include: data/work-flow integration, generic operation, task management, distributed workload, computational efficiency, robustness, security and a configurable topology. These are typical concerns of a monitoring and analysis platform.

The primary application of the platform is managing the computational resources available to medical decision support agents and assigning them appropriately. Its contributions come in the form of transparent load balancing, task scheduling, real-time compilation of

agents and a migratory capacity for said agents.

In [5] the need for methodology that supports "scientific homogeneity and accountability of healthcare decisions and actions" is articulated. It is argued that existing knowledge management tools do not readily integrate with clinical DSSs to the extent that there is a lack of "pragmatic" tools offering clinicians assistance in real time. As such, the combination of an ICU monitoring and analysis platform with a DSS is proposed. The *Intelligent Clinical Information Management Support (ICIMS)* system enables the combination of clinical information and processing tasks. These processing tasks encapsulate knowledge-based system techniques that are being prototyped. This provides a means for the integration of solutions that enable clinical assistance. Furthermore, it allows these knowledge-based system techniques to be assessed by personnel working with the system.

An interesting form of DSS that is often utilised for specific conditions is Multi-Agent DSS. One such system, which diagnoses cardiac disorders, is presented in [38]. Agents that incorporate new domain knowledge and methodologies can be incorporated without structural changes to the multi-agent system. An agent in the system, denoted CATS (chaotic analysis of time series), uses a methodology based on non-linear dynamics that performs offline analysis of Electrocardiographic (ECG) data. An ontology-based intelligent healthcare agent for respiratory waveform recognition was presented in [39]. The agent uses fuzzy matching against an ontology to recognise and classify respiratory waveforms.

An agent server for the analysis of physiological data streams from NICUs is presented in [40]. The agent-based intelligent support system examines streaming physiological data as well as previously acquired physiological data and clinical history data in order to detect trends and patterns that may indicate the onset of clinical conditions.

An agent-based distributed DSS for the diagnosis and prognosis of brain tumours is presented in [41]. The agent-based architecture provides a distributed network from which clinical data can be collected and classifiers, designed to identify brain tumours, used to process the data. The results of this classification are presented to the user via a graphical user interface. Agent functionality is abstracted to ensure platform independence. The argument is put forward that, due to the increased quantity of data available to the agents, improved reliability and accuracy is possible. This is further supported by the ability to retrain the classifiers on an ongoing basis.

A survey of agent-based intelligent decision support systems (IDSS) to support clinical management and research is presented in [42]. It identifies a number of IDSS from the literature available and compares them based on features specific to IDSS functionality (the types of decisions supported, the reasoning for developing the system, decision time sensitivity), the data layer (data source, distributed capabilities, real-time support) and the agent-specific features (whether the multi-agent system was open or closed, the agent

structure and the coordination mechanism used).

1.2.5 Knowledge Discovery

Knowledge discovery (also referred to as Knowledge Discovery in Databases (KDD)) is a broad field of study, involving research into the automated identification of novel patterns in data that are potentially useful and understandable. KDD differs from related fields, such as data mining and machine learning, in that it is concerned with the entire process from data acquisition to the eventual feedback of novel discoveries to the domain from which the data originated. Data mining and machine learning are considered stages within the process, used either for the classification of targeted entities within a data set or the training of a model to perform that classification.

In [7] efforts undertaken to integrate expert knowledge with data-driven techniques in order to improve operational protocols are presented. The difficulties faced by diagnosing a critically ill patient are highlighted. The large number of variables, potentially in excess of 200, are too great to develop a systematic response to. This is seen as a motivating factor for the utilisation of a DSS. The stated goal of decision support is "to supply the best recommendation under all circumstances" [7]. However, the task of developing the knowledge base is seen as an impediment to the development of DSS, particularly due to the cost in time.

It is proposed that a combination of statistics, machine learning and knowledge discovery should be used to develop the guidelines associated with DSS. Abstracted data and a Support Vector Machine (SVM) are combined with an expert knowledge base in order to assist clinicians in their decision making process. A combination of time series analysis and machine learning methodology is used to assist the development of decision support algorithms in a critical care setting. Time series data of high dimensionality is abstracted using the *Phase Space Model*. The SVM is used for learning state action rules and, combined with first order logic, as a representation of medical knowledge for action effect behaviour. The approach supports ongoing improvement and greatly reduces the cost of forming operational procedures and their validation.

In [8] a system for knowledge construction is presented. This is achieved through a human-machine collaborative approach to the exploration of time series data. It advocates annotations as a means for human interaction with the machine, allowing them to take the form of informative labels or segment delineations. These annotations act as a learning cycle for machine-based methodology, allowing clinicians to train the machine as to which segments should be flagged. The system is based on a multi-agent paradigm and introduces

a number of different agents; one for finding interesting segments, a second for classification and a third for defining relationships between time-stamped symbols. The feasibility, performance and usability of the system are then evaluated with an ethos of structural coupling (continual and influential mutual contributions to the definitive structure of the knowledge base at each level of the process), differentiating it from previous efforts. It is proposed that the use of a feeback loop from relationship to segment detection could lead to a performance increase.

In [43] a platform for the unsupervised data mining of time series data in an attempt to discover local patterns is presented. The design of an abstraction process to support discrete representation of global trends in multi-dimensional time series and support multi-level learning is proposed. An algorithm that aids in the mining of noisy, multidimensional data across a number of data transition phases is presented.

There is a large overlap between KDD and DSS. Both utilise aspects of monitoring and analysis, data mining, machine learning, pattern recognition and statistical methodology where appropriate. It could be argued the differentiating factor between the two is that DSS's application of information, and knowledge, is primarily focused on the assistance of data analysis, as opposed to KDD's primary focus which is knowledge generation. However, the two are not mutually exclusive and often are simultaneously integrated into systems.

One such example is the Artemis project [18, 19]. This framework involves the acquisition of large volumes of streaming physiological data from the ICU and the application of clinical rules to these data streams in real time. Clinical rules are derived from specifications for existing clinical guidelines, or specified anecdotally by a clinician. Alternatively, rules can be proposed through a set of data mining conditions in physiological data streams, laboratory results, and observations of patients. The framework supports the acquisition, real-time processing, storage and data mining of physiological data. The framework is used in the ICU to detect changes that might serve as onset predictors for selected conditions.

1.2.6 Complex Software Systems

An IE consists of an arbitrarily large number of interacting components. These components can be loosely organised and operate free of laws that govern their behaviour, individual components can be goal-driven and can change over time. They can also be influenced by the behaviour of their subcomponents. As such, an IE is considered a complex software system. This realisation has influenced many of the subsequent design and implementation decisions.

Components involved in the process of monitoring and analysing data have a many-to-many relationship. This process can comprise of multiple sources of data, each of which

can contain a multitude of data formats requiring a multitude of analysis techniques; the supporting of which may necessitate the facilitation of expert analysis.

Complex software systems and how to manage complexity are discussed by Jennings [44]. He states that complexity is a natural characteristic of systems with large numbers of interacting components and that software engineering plays a significant role in providing a means to handle this complexity. Identifying the complexity of an IE and its role within the process of monitoring and analysing data is a step towards identifying a means of handling that complexity. A number of characteristics relating to complex software systems are identified:

i) Complex systems and hierarchy:

   - Systems are made up of interacting subsystems; each of which has its own hierarchy that can be broken down into smaller subsystems (until the lowest form of components is arrived at).
   - The organisational relationships between subsystems can take forms that are readily recognisable (*e.g.* peer-to-peer, client/server) but these can also change over time.

ii) Defining that which is considered primitive within a system is an arbitrary decision and is dependent on the observer.

iii) "Hierarchic systems evolve more quickly than non-hierarchic ones of comparable size (that is, complex systems will evolve from simple systems more rapidly if there are clearly identifiable stable intermediate forms than if there are not)." [44]

iv) Interactions between subsystems and those within a subsystem can be differentiated based on the frequency and predictability of the interaction. Interactions within a subsystem occur more frequently and are more predictable than those that occur between subsystems. For this reason, complex systems are considered "nearly decomposable" and their subsystems are treated as independent entities even though they interact with one another. Jennings [44] highlights the benefits of a complex system as well as the limitations of a less complex, yet well understood design.

Based on these characteristics, a complex system in its simplest form, as described by Jennings [44], is depicted in Figure 1.1. The "frequent interaction" connections are representative of interactions within subsystems, the "infrequent interaction" connections are representative of interaction between subsystems and the hierarchial nature of the system is represented by the "related-to" connections.

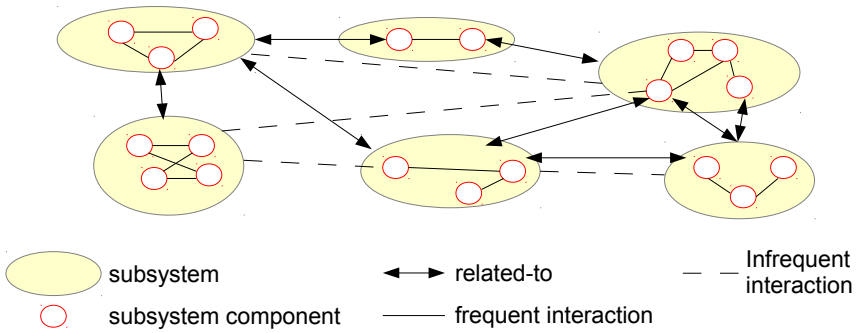

Figure 1.1: Jenning's view of a canonical complex system.

A number of tools, devised to manage complexity, are compatible with this view of complex systems. Decomposition: the breaking down of a problem, or system, into smaller manageable ones. Abstraction: defining the level (of complexity) at which interaction with the system takes place and suppressing complexity outside of this level. Decomposition and abstraction assist with managing the complexity of a system by limiting the scope of its design. Organisation: how relationships between components are managed. This assists with complexity by enabling organised relationships between simple components to be created, resulting in the creation of higher-level functionality.

Complex Systems Integrated With Expert Analysis

Financial markets are an example of a complex system and can be viewed as a non-linear dynamic system that is continually evolving [9]. The associated domain knowledge is both complex and dynamic. This results in the selection of an optimal investment being a difficult problem for an investor [9]. Irrespective of these difficulties, significant research efforts have been undertaken to further the understanding of stock market fluctuations, to develop predictive capabilities and formulate associated trading strategies.

Financial time series data is characterised by high-frequency multi-polynomial components, nonlinearities and discontinuities [45]. As such, the forecasting of future market prices and the prediction of their movements is difficult [45]. This is an example of a domain where time series data requires expert analysis. The mining of financial time series data is challenging due to both the characteristics that embody the market and the volatility of data being monitored. Endeavors to further our understanding of the nature of this data and its complexities have been met with some success, as evidenced by the increasing level of stocks and future trades that employ intelligent systems to assist/make decisions.

1.2.7 Agent Frameworks

Agent theory is concerned with the application of individual agents to collectively solve problems. It typically incorporates a multi-agent system, as described in [46], where a collection of loosely-coupled problem solver entities cooperate to achieve desired objectives that are beyond the individual capabilities or knowledge of each entity. An agent-oriented approach aligns with that of a complex system as a number of their characteristic traits overlap.

A significant body of work exists on the application of Multi Agent Systems (MASs) to medical problem domains. Medical MASs have been developed in areas such as decision support, clinical knowledge tools, patient history tracking and monitoring solutions. One of the earlier examples of the intended use of intelligent agents for intensive care monitoring is presented in [47]. The agent operates on two levels; the lower level performs data reduction and abstraction tasks and the higher level applies various reasoning rules and reactive diagnosis skills against the data.

There are many benefits to be realised from an agent-oriented application. Software reusability and maintainability can be achieved through the use of an agent-based approach. Other advantages, such as the ability of the system to evolve over time, or pivot to change with user needs, can also be realised. A more comprehensive overview of agent approaches is presented in [48]. Agent theory influenced the IE's design because of the intentional stance and the ability to reason about the problem domain that characterise the agent approach also closely mimic the process of human interpretation. Furthermore, the independent nature of the various components envisioned in the IE lend themselves to implementation as agents. A number of generic agent frameworks have been developed

JADE [49] is a Java-based software framework to develop agent applications in compliance with the FIPA (Foundation for Intelligent Physical Agents) specifications for interoperable intelligent multi-agent systems. JADE provides a development environment that enables the implementation of agent platforms in a manner that is compliant with FIPA standards for interoperable intelligent multi-agent systems. Its goal is to simplify agent development while ensuring compliance with FIPA standards [50]. This is achieved by providing common functionality for the non-application specific tasks in an agent framework such as message transport, encoding, parsing and the agent lifecycle. In doing so, JADE enables simpler integration with external software providing a general model that can be specialised to implement both reactive and belief-desire-intent architectures.

The SPARK agent framework [51] is based on the belief-desire-intent model of rationality. SPARK offers a framework that is scalable, while maintaining a clean semantic underpinning, traditionally associated with formal agent frameworks. This allows SPARK's

procedural language to be well defined and the framework to support reasoning techniques for procedure validation. SPARK's ability to scale makes it applicable to a multitude of real world problems.

Agentcities [10] is an initiative to create a global, open, heterogeneous network of agent platforms and services to which any agent researcher can connect their agents. The intent is to enable the running of agents on a variety of different platforms, operated in different manners, implemented in different languages and owned by different organisations. This is achieved by providing agent components that can "be created through the flexible use of this inter-agent communication model and the semantic frameworks, shared ontologies, content languages and interaction protocols that support it." [10]. The main objective of AgentCities is to identify how heterogeneous resources and services can identify and co-ordinate with one another.

Agent Frameworks provide a practical means of embodying complex behaviour in a system. They incorporate a model of how functionality can evolve, demonstrate goal-directed behaviour and provide a mechanism for the integration of new techniques on an ongoing basis. The frameworks reviewed support many of the characteristics that are desired in an IE and as such will influence its design and development.

1.3 Research Question

This thesis explores the creation of a system that can meet the challenges, and requirements, outlined in Section 1.1. It describes the creation of a generic environment for managing the lifecycle of data, from its production, through to its appropriate interpretation in a given context, so that it can be effectively exploited. A system capable of performing these functions is referred to here as an *Interpreting Environment* (IE). No existing system takes the generic approach proposed here.

Thus, the novelty in the IE lies in its generic approach, in the use of transparent, hierarchically organised workflows, the incorporation of appropriate feedback techniques, which may be used to drive the system to convergence, and the dynamic creation of an audit trail supporting data provenance.

A prime motivation for the creation of the IE was to make it difficult to use data inappropriately within a field of study. Appropriateness is a relative term and should be defined and accepted within each application domain. The generation of transparent workflows enables the IE map the flow of data through the system. This is intended to enable enhanced peer review with regard to how data is processed within a system.

Once defined, only valid workflows should be supported by the IE and any attempt, albeit inadvertent, at an inappropriate workflow construction should be transparent and

easily discoverable. This phenomenon is recognised by Keogh [52], and with the proliferation of data and the explosion of specialisations, it is anticipated that this problem will become more prevalent. The IE will address this by enabling its components to log data such that a verifiable audit trail could be generated.

To illustrate the practical advantages of the IE, a challenging application domain requiring all of the subtleties of interaction alluded to in Section 1.1, is used as a running example to illustrate its efficacy. This application domain is described in Section 1.4.

1.4 Use Case: Neurophysiological Monitoring

Neurophysiological monitoring of brain function is required for many newborn babies. This involves the acquisition of EEG data, which is carried out by trained clinicians. Data is interpreted by neurophysiologists who are expert in the analysis of neonatal EEG. The analysis of EEG, and other physiological data streams captured in the NICU is essential for the effective diagnosis and subsequent treatment of neonates.

1.4.1 The Clinical Perspective

Hypoxic-ischaemic encephalopathy (HIE) and associated seizures are common neurological conditions that can cause lifelong disability in neonates. Neonates can experience seizure as a result of multiple central nervous system diseases. Seizures afflict approximately .003% of full-term neonates and approximately 5% of pre-term neonates [53] [54]. 20-40% of full-term neonates afflicted are subsequently handicapped. This can be as high as 75-88% for premature neonates [55] [56].

Seizures are notoriously difficult to detect, as the signs for clinical diagnosis may be subtle or absent. Currently, the monitoring of a multi-channel EEG is the gold standard for identifying neonatal seizures. This requires the neonate to be continuously monitored for long periods of time [57].

In the case of HIE, the monitoring of EEG is used to form a prognosis [58]. Neuroprotective treatments, such as therapeutic hypothermia, are available and are becoming the standard of care [59]. However, treatment is time-critical and must commence within six hours of birth. So a delay in EEG analysis results in a missed opportunity to improve patient outcomes and has a significant impact on long-term outcomes.

1.4.2 The Challenge For Neurophysiological Monitoring

Monitoring of a multi-channel EEG is the most effective means of detecting seizure activity accurately. Ideally, an experienced on-site neurophysiologist will be available to analyse the

EEG and issue a clinical report. However, in reality, NICUs are widely dispersed and the vast majority have no access to the required expertise [60] [61]. The problem is further frustrated by a lack of standardised data collection procedures and the prevalence of stand-alone *ad hoc* systems [60] [62].

There is a divide between our medical, scientific and technical advances and their application to neurophysiological monitoring. As this divide grows, health systems, their personnel and the infrastructure to support them will come under increasing pressure. This divide is analogous to the "know-do gap" [63], whereby known solutions are not being applied even though the potential to do so exists. The challenge for neurophysiological monitoring of brain function lies in identifying the root causes for this divide and devising a solution to overcome them.

The Current Process

In the case of neurophysiology, monitoring refers to the observation of electrical activity in the brain. This is achieved by having trained technologists place a number of electrodes on a patient's head. The electrical activity is observed as an analog signal, converted to a digital format and persisted to an EEG monitoring machine. EEG data consists of a series of two-dimensional, high-frequency, time-value pairs with each time-value pair corresponding to a reading from a specific electrode (see Section 5.2).

Ideally, the neurophysiologist is physically present in the NICU and interacts directly with the EEG monitoring machine connected to the patient. The patient's data is accessed from the EEG monitoring machine locally, as depicted in Figure 1.2. The data is interpreted by visualising it as a series of time series values, drawn in a left to right fashion, across the monitor of the EEG monitoring machine. A montage (the arrangement of the electrodes placed on the head combined with the form of monitoring (referential or bipolar)) is applied to the EEG data to more clearly display the brain activity. Analysis is carried out by a neurophysiologist reviewing this visualisation. As a result, the neurophysiologist formulates a diagnosis and prescribes a treatment.

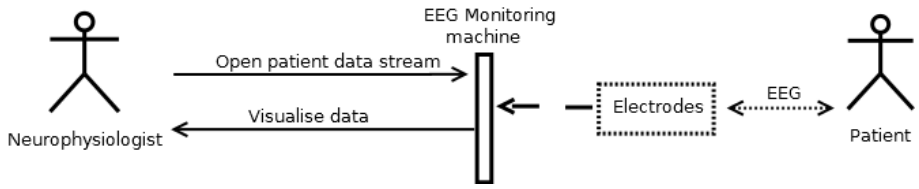

Figure 1.2: A neurophysiologist's workflow.

Where a neurophysiologist is not available, possible courses of action include: forego analysis; move the patient to another location where expertise is available; request that

a neurophysiologist travel to the acquisition location; or provide a neurophysiologist at a remote location with some means of viewing the recording data remotely. All of these occur in practice. The first two options are less desirable as they may impact negatively on patient outcome. Bringing an expert on-site may introduce a delay in diagnosis and the time spent travelling results in a loss of productivity for the neurophysiologist. The last option is the most desirable, provided that the process of transferring EEG data between locations does not introduce delays that could result in a missed opportunity for treatment.

In many cases, data transfer is achieved by stopping the acquisition device and sending the resulting EEG data by courier or electronic transfer to the remote location for analysis. This technique introduces a significant delay as no analysis is performed until acquisition has stopped and the data transfer has been completed. Another technique that has become increasingly common is to allow the off-site expert to interact with the acquisition software directly using a remote desktop session. This eliminates the time delay, but is not without drawbacks: a low-latency network connection is required; difficulties may arise with hospital firewalls; confidential patient information may be visible on screen, possibly in breach of hospital policy or data protection regulations; and inadvertent interference with the acquisition process is possible.

Issues With The Current Process

Diagnosis of neonates and their subsequent treatment is a time-critical process with a limited window of opportunity. This results in a definitive time limit in which diagnosis and treatment can occur. This time limit puts considerable pressure on the operation of the NICU. As such, the *timeliness of data analysis* is a major issue for the NICU.

Currently the NICU workflow requires a neurophysiologist to be physically present to diagnose a patient. In reality, NICUs are geographically dispersed and the majority have little or no access to expertise. This localised workflow is overly restrictive and results in the suboptimal treatment of patients as well as the suboptimal operation of the NICU.

A workflow is influenced by both the personnel and the infrastructure available. Current NICU infrastructure could be described as an intermeshing web of closed platforms. This leads to inter-operability and communication barriers whenever a change is proposed to the NICU workflow. These barriers stem from the prevalence of proprietary hardware and software utilised for the monitoring and analysis of EEG; these systems also use proprietary file formats, and communication protocols, further frustrating the problem and resulting in vendor lock-in.

The analysis of physiological data requires expert knowledge in that type of physiological data. Just as a neurophysiologist is required to analyse an EEG, a cardiologist is required

to analyse an electrocardiogram (ECG). This expertise infers a significant investment, both monetary and in terms of time, on the part of the expert. The finite nature of this expertise can make its offering prohibitively expensive to an individual NICU.

The issue is further frustrated by the localised nature of the workflow, making it impossible to effectively pool expertise and share associated costs. Furthermore, even where cost is not an issue, demand for expertise can often outstrip the supply [64]. This precarious *access to expertise* negatively impacts NICU services.

The *availability of expertise* is also an issue for the NICU. Naturally, the time of birth of a neonate is unpredictable. As diagnosis and treatment is time-critical, in order to maximise patient outcome, the required expertise needs to be available on a twenty-four-hour basis. However, even NICUs that have access to EEG monitoring and the required expertise on a continuous basis are often unable to provide an "out-of-hours service" [64].

While these issues are presented within the context of the use case, they are not unique to neurophysiological monitoring. Each issue can be extrapolated to a broad range of domains that require the monitoring and analysis of data. Consequently, it is envisaged that the applicability of any solution identified will be far reaching and benefit each of the domains that experience these issues.

1.5 The NICU As An Interpreting Environment

It is clear that the monitoring and analysis of neurophysiological data is an integral part of the NICU's operation. The diagnosis and treatment of neonates necessitates data acquisition, analysis and subsequent action. In this case, these actions are the application of appropriate treatments. As such, an IE for the monitoring and analysis of neurophysiological data that mitigates the issues identified in Section 1.4 would be highly advantageous.

As EEG data is digitised, data interchange between neurophysiological instruments and other computer and information networks can be facilitated. Teleneurophysiology capabilities have been developing over the past two decades, and detailed remote assessments of brain function by a neurophysiologist are now possible. This has resulted in the development of a number of proof-of-concept and limited commercial endeavours being undertaken (see Section 1.2.3 for more detail). However, none of these has managed to address the problem in a comprehensive manner. The reasons for this are many and varied:

1. *Timeliness of data analysis*: Realising real-time access to data in the NICU is difficult due to the proprietary, and closed, nature of the software used for the monitoring and acquisition of data.

2. *Over specification*: Current systems focus on clinicians' immediate needs, resulting

in the development of niche monitoring solutions that are difficult to maintain and are not capable of supporting a broad range of functionality that would simplify and assist clinicians with their work. Clinical experts should not be expected to perceive the value of general purpose solutions.

3. *NICU infrastructure*: Inter-operability issues are prevalent due to a use of proprietary file formats, communication protocols and the prevalence of closed platforms within the NICU.

4. *Localised workflow*: Access to expertise due to geographic constraints. Workflows, and the systems that support them, should optimise access to expertise, even if only local use is intended.

5. *Availability of expertise*: There is a finite level of expertise available. How can a system be utilised such that an expert's time is maximised? Improvements to collaborative functionality, automated solutions to reduce an expert's workload and standardising the practices involved could all contribute to increasing the availability of expertise.

6. *Capturing of expertise*: The ability to capture expert knowledge in a reusable state is critical to deriving long-term solutions. It facilitates research into advanced automated solutions.

The complexity of this application domain will draw upon all of the characteristics associated with the IE as set out in Section 1.1. Thus, near real-time acquisition helps to solve the problem of providing a clinical diagnosis by making data available to a pool of non-local neurophysiologists. The unique perspectives of neurophysiologists will be accommodated by providing appropriate interfaces to the environment. Over-specification will be overcome by providing these interfaces in the form of modular components and analysis techniques, thereby removing any associated infrastructural and workflow constraints. The ability of the environment to dynamically accommodate expert knowledge, and its association with raw data, enables it to be captured at the level at which it is being generated. Furthermore, the availability of this knowledge provides the means to construct, preserve and maintain an expert knowledge base, gathering the necessary data for future growth and the development of advanced, automated solutions.

1.6 Summary

This chapter outlined the motivation for creating a generic IE, illustrated some concepts and characteristics of that environment, discussed related work based on those characteristics,

described a complex application domain to which it would be of benefit and described how an IE would be mapped to that application domain.

1.7 Dissertation Overview

The remainder of this dissertation is organised as follows: Chapter 2 details the structure of the IE and the roles comprised within it. Chapter 3 discusses the workflow of the environment and the flow of data through its components. Chapter 4 details the implementation of the IE, highlights the design and engineering decisions taken, and introduces an IE Toolbox to assist the development of generic IEs. Chapter 5 details the application of the IE to a specific application domain resulting in the creation of the Physiological Data Server. In Chapter 6, an evaluation of the Physiological Data Server is carried out. Finally, Chapter 7 details future avenues of work and conclusions.

Chapter 2

Structure Of The Interpreting Environment

The use case in Section 1.4 presented a series of issues that affect the monitoring and analysis of neurophysiological data. These issues are not unique to neurophysiological monitoring, but can be extrapolated to many domains requiring expert analysis of data in a timely manner. In this chapter we view the process more generally. This allows for a more comprehensive understanding of the issues involved in creating a generic IE.

Initially, the monitoring and analysis of data is discussed. Thereafter, the roles that constitute an IE are identified. The relationships between these roles and how they relate to the challenges and issues that afflict a monitoring and analysis system are discussed. The use case is explored such that a switching of perspective from the general to the specific, and *vice versa*, is facilitated. This allows a greater appreciation for what a generic solution should entail, it assists in identifying the issues that arise and highlights the level at which they should be addressed.

2.1 Monitoring And Analysis

Monitoring

Monitoring is the observation of the state of an entity. Carrying out an observation requires both the means to access and record the state being observed. The frequency of monitoring can vary. An entity can be monitored:

1. As a single point observation *e.g.* an MRI scan to perform a diagnosis.
2. Periodically *e.g.* a quarterly review of expenditure.
3. Continuously *e.g.* meteorological data being accessed in real time.

Analysis

Analysis of data is a process that transforms raw data into information. This process can range from inspection by an expert in the data domain to the application of sophisticated algorithms designed to uncover trends and relationships. The information resulting from the process of analysis may subsequently be interpreted by another expert, but this time in the application domain rather than the data domain. Similarly, algorithms can analyse data in the application domain and result in informed actions being taken based on that information.

Cost Of Monitoring And Analysis

Monitoring and analysis typically has associated costs in relation to the accessing, processing and storage of data. These costs often have an impact on the decision of how frequently monitoring takes place. For example, the limiting factor in an MRI scan could be one of access to data resulting from the availability of the patient to be scanned. In the case of quarterly reviews, the limiting factor could be the cost of processing data, resulting from the financial cost associated with accountants who process that data. While in the case of meteorological data, the limiting factor could be the cost associated with the storage of the data generated.

Other factors determining the frequency of monitoring and analysis can include the amount of information that can be derived from the data. For example, in trend detection there is little point in monitoring at a higher frequency than the data is changing, or for the purposes of real-time analysis, there is little point in monitoring at a frequency that generates more data than it is possible to process. Moreover, monitoring does not occur in isolation and invariably impacts on the system being monitored. If monitoring is performed intrusively it may alter the behaviour of the system being monitored. For these reasons, it is important to carefully identify which system entities should be monitored and the degree to which each should be monitored.

2.2 An Interpreting Environment

An IE provides an abstract perspective from which systems for the monitoring and analysis of data can be viewed. This perspective makes the inner workings of a system more readily approachable thereby allowing for an improved understanding of their inherent complexity and by providing a standardised perspective from which to view systems of a comparable nature. Moreover, it provides a means of managing and organising this complexity by enabling the decomposition of a system into multiple levels of abstraction. Each level of

abstraction is defined by the roles that embody the processes of monitoring and analysis.

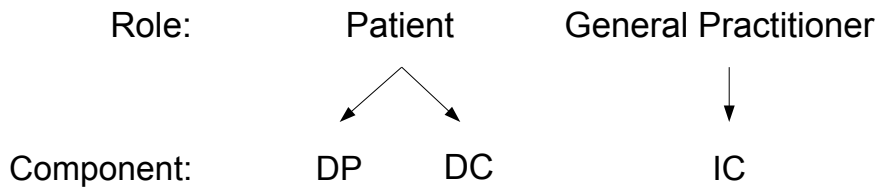

Figure 2.1: The mapping of roles between a patient and general practitioner from the perspective of an Interpreting Environment (IE).

In Chapter 1 an IE was defined as a collection of DPs, ICs and DCs together with a description of their relationships. Each of these components is present in a non-trivial IE. From the perspective of an application domain in which data is produced, interpreted and consumed, each action can be associated with an entity adopting a particular role within that domain. Thus, for example, a patient may act in the role of the DP when they undergo tests and describe symptoms to a general practitioner. The general practitioner may adopt the role of the IC as they interpret data received from patients to form a diagnosis and the patient may adopt a second role, that of the DC, as they consume and act on that diagnosis.

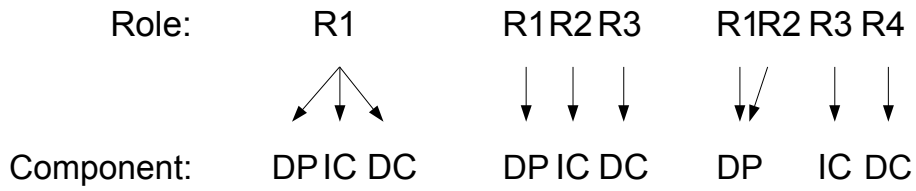

Figure 2.2: Examples of the different mappings between an IE's roles and an IE's components.

The mapping of roles to an IE's components may be one-to-one, many-to-many or one-to-many. In addition, as seen later, a component may be associated with a number of entities collaborating to fulfill a particular role in such a way that that collaboration can be recursively mapped onto the three main components of the IE. This recursive definition can give rise to a hierarchical description of the application domain and each level in that hierarchy represents an abstraction of the level below as depicted in Figure 2.4.

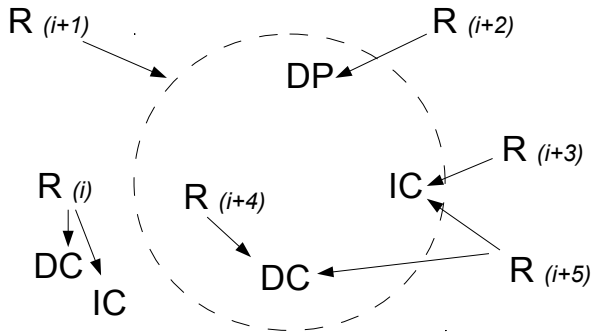

Figure 2.3: Multiple components being mapped to multiple roles. R(i+1) contains components that are mapped to alternate roles.

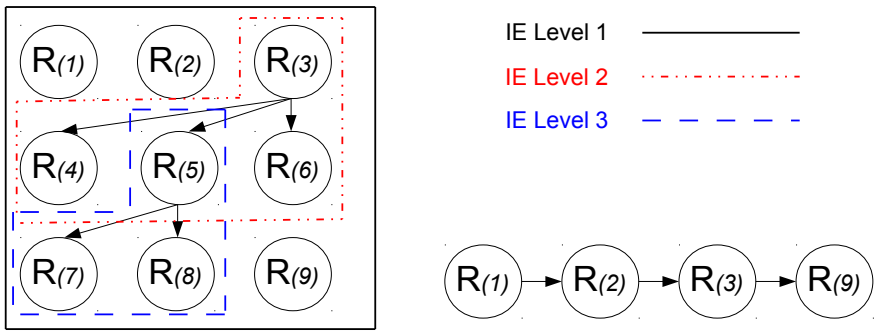

Figure 2.4: An IE depicted with multiple levels of abstraction and its transformation to a single network of DPs, ICs and DCs.

The power of the IE comes from being able to handle each level of abstraction as a single network of DPs, ICs and DCs, thus hiding information appropriately to reduce complexity at that level.

2.2.1 The Process Of Mapping Roles To Components

The process of mapping roles to components is carried out by the domain expert when attempting to capture the workflow in the application domain. In this process DPs, ICs, and DCs are identified, each of which may in turn be refined into other DPs, ICs and DCs. The decisions taken in creating the hierarchy will be informed by the expert knowledge about how the workflow should best reflect the goals and correctness of the specific application.

For example, two alternate mapping processes are depicted in Figure 2.5. The first two diagrams depict different approaches to the mapping process for the treatment of a patient in the NICU[1]. In both instances, a neonate is mapped to the role of a DP, as the producer of data, the NICU is mapped to the role of the IC, as it carries out the interpretation of that data. This is where the similarities end.

In the first instance (see Map#1 in Figure 2.5), the neonate, the neonate's parent and the clinician are mapped to the role of DC and so are consuming the interpretation generated by the NICU. While a domain expert may create this hierarchy, it is not ideal. For instance, the parent, and the neonate, may not be capable of comprehending the interpretation produced by a NICU (In this instance a clinical report issued by the NICU relating to the patient state). As such, an alternate hierarchy that enables interpretation by a non-expert may be better suited.

Instead, a domain expert may desire a hierarchy involving a series of IC such that interpretation of data can be derived and delivered at the appropriate level. In the second instance (see Map#2 in Figure 2.5), this process is broken down into multiple levels of abstraction. This enables an improved hierarchy to be realised. The neonate and NICU maintain their original roles, but they are mapped to a DP at a lower level of abstraction (level 2 of the abstraction process). This enables the clinician to act as an IC and provide an interpretation that is more suitable for a parent (resulting in a lower level of abstraction again). Subsequently, the parent and neonate repeat this process enabling the two higher levels of abstraction to act as a DP and the parent to provide interpretation for the neonate. In this manner, the domain expert can create hierarchies that map components at a particular level of abstraction, simplifying its representation and more accurately reflecting the goals and correctness of the specific application desired.

This process is a means to maintain a record of the components in an IE, the role they play and the relationships between them. In doing so, the provenance of data production, interpretation and consumption can be provided. This record can then be interrogated by interested parties, enabling a greater level of transparency throughout the entire process.

Transparency introduces a number of potential benefits. For example, certified IE instances, whose workflows have been approved by the community, could be digitally signed thus enabling these peer-reviewed workflows to be used correctly and with a degree of confidence. Thus enabling reusable components to be created and to be used as elements in software engineering methodologies.

[1] In the context of this workflow the NICU is responsible for the interpretation of data acquired from the neonate and outputting data relating to the patient state. The clinician is considered outside the scope of the NICU.

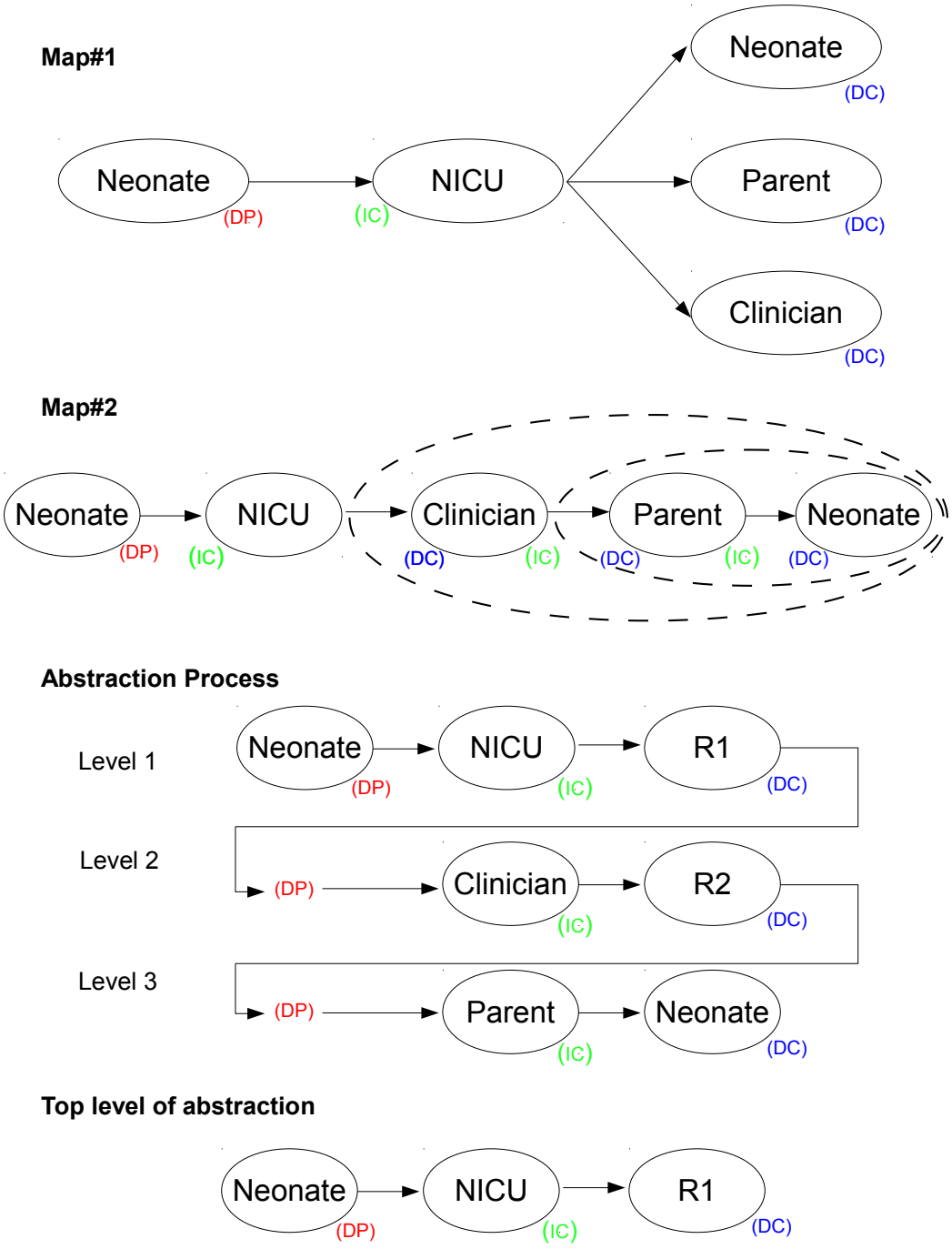

Figure 2.5: The NICU mapping process and the rearrangement of components to form a hierarchy.

2.2.2 Feedback

Feedback is defined as using the output of a process as an input to an earlier process. Feedback loops are used for checking system up-time in a wide array of systems such as control systems, telecommunications, internet protocols etc. Integrating feedback loops are one way in which expert knowledge can be collected and expert knowledge repositories formed. Integrating them with an IE can be a complex challenge.

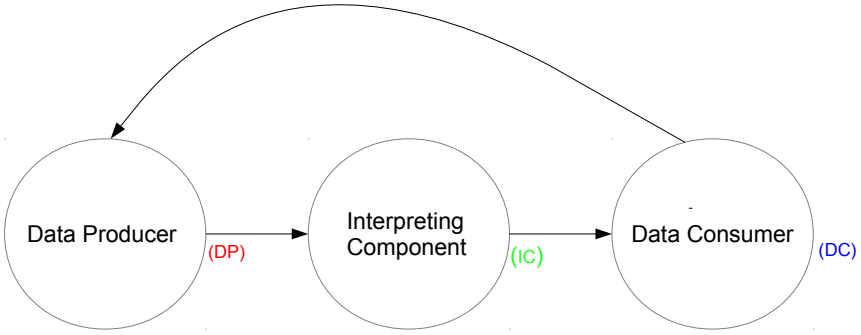

Figure 2.6: The IE incorporating a feedback loop.

Figure 2.6 depicts an IE that contains a feedback loop. The IC is continually processing data being generated by the DP. The IC's output is being fed to the DC, until the desired result is output from the IC. The DC receives the interpretation and if it is in the desired state exits the IE, otherwise the loop continues and an iteration of the interpretation process occurs.

To simplify the management of feedback loops in the IE they can be viewed as a dynamic recursive invocation of the workflow. Thus each recursive step can once again be viewed as a simple produce, interpret, consume graph. In effect, this recursion creates a dynamic hierarchy (in contrast to the static hierarchy defined by the domain expert) whose lowest level of abstraction is the base case of the recursion and is achieved when desired convergence is reached.

2.2.3 Convergence Within An IE

Convergence is a desired state into which you wish to dynamically transition and once achieved, in which you wish to remain. Convergence to a particular goal state is often the desired outcome from an IE and hence a mechanism to support it is needed.

Workflows incorporating feedback may be used to model feedback control systems, resulting in the state of a component in an IE being driven to convergence. Through the use of explicit feedback loops in a workflow within an IE, the components whose state is being modified is explicit and transparent. Depending on the application domain, manipulating

that state may or may not be appropriate.

Inappropriate Use Of Convergence

The scientific method is the process of observing a phenomenon, formulating a hypothesis to explain that phenomenon, designing and running an experiment in a controlled environment to objectively evaluate this hypothesis. Scientific rigour requires that these controlled experiments can be reproduced by others and it is an effective means of identifying biased, incorrect and fraudulent research findings. Figure 2.7 and Figure 2.8 depict the IE's application of convergence to facilitate the scientific method.

In Figure 2.7 the IE is being used to implement the scientific method in an incorrect manner. A phenomenon is occurring in a controlled environment (the IE) resulting in the production of data (DP), this data is being used to evaluate a hypothesis (IC), this interpretation results in an observation as to the validity of that hypothesis and a subsequent action being performed based on that observation (DC). In this instance, the hypothesis (IC) is static as the output from the observation (DC) is being fed into the phenomenon (DP) as an input such that the data is being acted on but the hypothesis is not. From the perspective of the scientific method, this is an invalid use of convergence as the data is being modified to suit the theory.

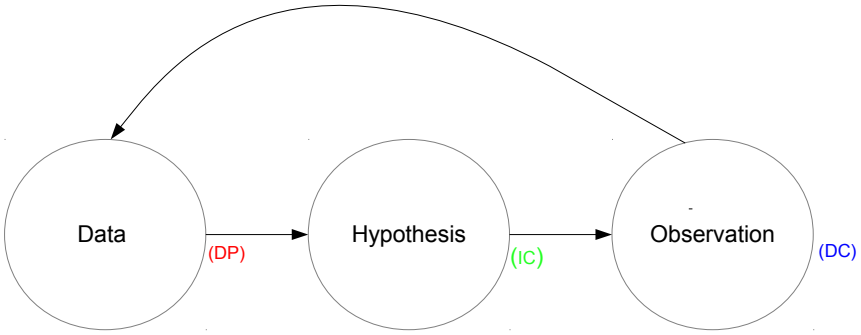

Figure 2.7: The resultant state (DC) observed is converging the data (DP) to a desired state that complies with the hypothesis (IC). This is an invalid use of convergence, as the resultant observation is being used to alter the data acquired. In effect, the data is being altered to suit the theory.

In Figure 2.8 the IE is being used to implement the scientific method in a correct manner. A phenomenon is occurring in a controlled environment (the IE) resulting in the production of data (DP), this data is being used to evaluate a hypothesis (IC), this interpretation results in an observation as to the validity of that hypothesis and a subsequent action being performed based on that observation (DC). However, in this instance the hypothesis is not static, the observation is being used to alter and refine the hypothesis. From the perspective

of the scientific method, this is valid as the hypothesis is being modified to suit the data.

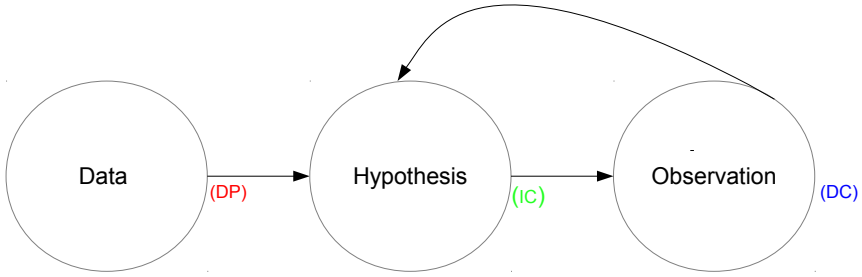

Figure 2.8: The resultant state (DC) observed is converging the hypothesis (IC) to a desired state, such that it complies with the data being generated. This is a valid use of convergence as the resultant observation is being used to alter the hypothesis.

Appropriate Use Of Convergence

In Figure 2.9 a neurophysiologist is continually interpreting data being produced by the neonate, which results in an action being carried out by the NICU. The NICU is responsible for interpreting the neurophysiologist's prognosis and carrying out any associated actions, in this case, some form of treatment. Due to the feedback mechanism within the IE, this subsequently affects the data being produced by the neonate and the interpretation by the neurophysiologist. This results in a change to the prescribed course of treatment consumed by the NICU. The neurophysiologist desires the patient data to converge on a healthy patient state, and is continually altering their interpretation based on the data being interpreted. As the IE is converging to a healthy patient state this is a valid use of convergence within this application domain.

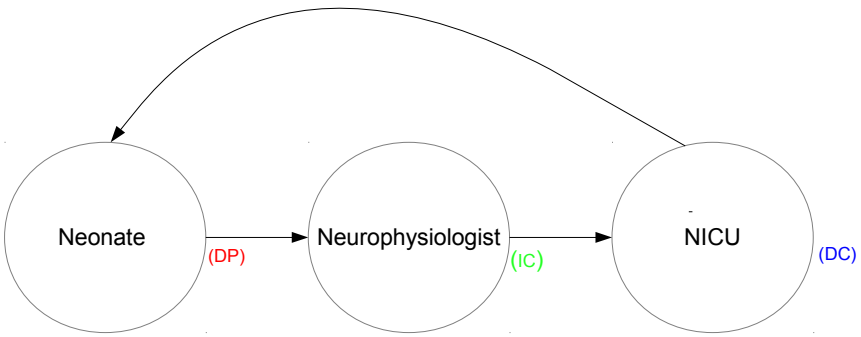

Figure 2.9: The neurophysiologist is converging on a healthy patient state and so it is a valid use.

Similarly, in Figure 2.10 a neurophysiologist is continually interpreting data being produced by the neonate, which results in an action being carried out by the NICU. In this case, an altering of the neurophysiologist's state. Due to the feedback mechanism within the IE, this subsequently effects the interpretation by the neurophysiologist, but has no effect on the data being produced by the neonate. It may result in a change to the prescribed course of treatment consumed by the NICU, but this change will not impact the patient. The neurophysiologist desires the patient data to converge on a healthy patient state but alterations to the interpretation of data have no effect on the neonate. As such, the IE is not converging to a healthy patient state, rather it is converging a neurophysiologist's interpretation to a particular state. This is an invalid use of convergence within this application domain.

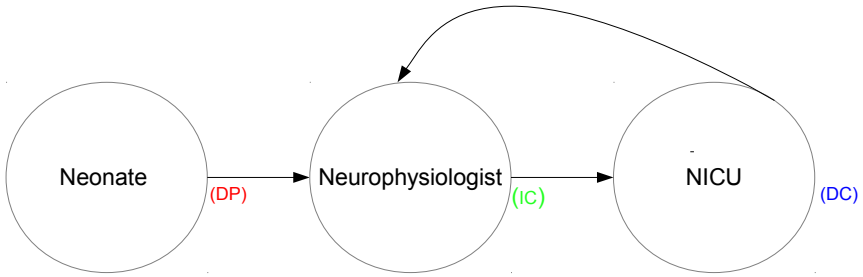

Figure 2.10: The neurophysiologist is converging the desired results to match the state dictated by the NICU.

Overview

In both cases, the benefit of the IE is its capacity to visualise the workflow of the system in a transparent manner. This exposes its relationships to the relevant community, allowing them to identify whether or not a process is being carried out in the correct manner. The validity of a workflow using convergence is specific to an application domain, for instance, the valid and invalid examples given for the instances above are opposites of one another. The application-specific nature of workflows further emphasises the need for transparent analysis of such workflows.

2.2.4 The Use Case

The use case (see Section 1.4) has three stakeholders from which the process of monitoring and analysis is considered. The neonate; who is, or is suspected to be, in need of treatment, the neurophysiologist; who is an expert in neonatal EEG and possesses the knowledge that

enables analysis of the neonate's brain activity, and the NICU; which is responsible for interpreting any prescribed treatment received from the neurophysiologist and acting on it, administering treatment as a result if necessary. The stakeholders and the roles they play are highlighted in Figure 2.11.

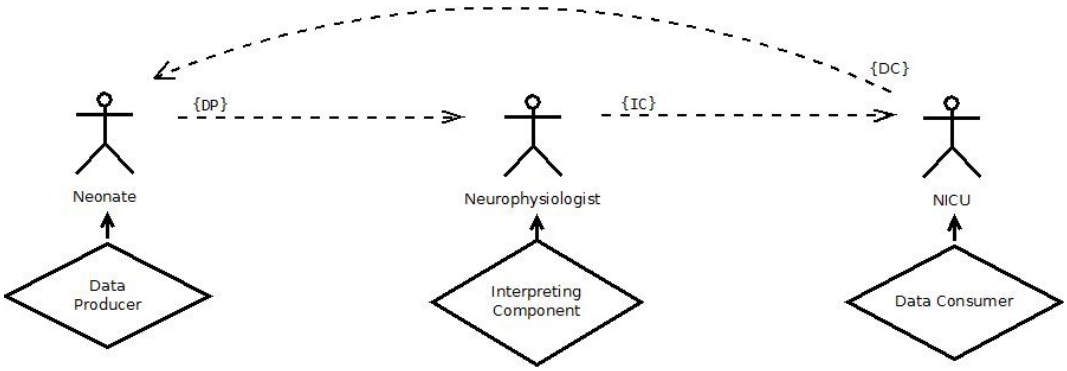

Figure 2.11: The stakeholders that encompass the process of neurophysiological monitoring: The neonate, the neurophysiologist and the NICU.

2.3 Summary

This chapter outlined the structure of an IE. It discussed the process through which a system is decomposed into multiple levels of abstraction and the mapping of roles to entities within a domain based on the production, interpretation and consumption of data. The role of a domain expert in facilitating the mapping process is also highlighted. The concept of feedback and the resulting dynamic hierarchies that ensue are discussed, along with the appropriate/inappropriate application of convergence within an IE.

Chapter 3

Interpreting Environment Workflow

Having looked at the structure of the IE, this chapter looks at the flow of data through the workflows in that environment. As seen in Chapter 2, every workflow can be abstracted into a simple network of producers, interpreters, consumers (possibly with feedback loops) by hiding sub-workflows in component containers at the appropriate level of abstraction. This drastically reduces the complexity of analysing workflows within the IE. Requiring us to only look at each of the three components (DP, IC and DC).

3.1 The Data Producer

In practice, DPs can emit data at different frequencies from single point measurements to high-frequency data streams. The greater the frequency, the bigger the challenge, since the data will have to be either processed at the rate of emission or stored for later processing.

Figure 3.1 depicts data being produced at a rate greater than the IC can process it. A buffer is used to cache the excess data. This enables the IE to continue operating as excess data is stored in a manner that is accessible to the IC. The relationship between production and processing introduces a constraint to the IE's workflow as the quantity of data the buffer can handle is finite and if this limit is exceeded an overflow will ensue. It should be noted that buffering can be introduced for reasons other than attenuating the frequency of production and the frequency of interpretation (see Section 3.2.1).

3.1.1 Time Series Data

Time series (TS) data is a sequence of observations taken at discrete (not necessarily uniform) time intervals. An observation is one, or more, data points that denote the state of

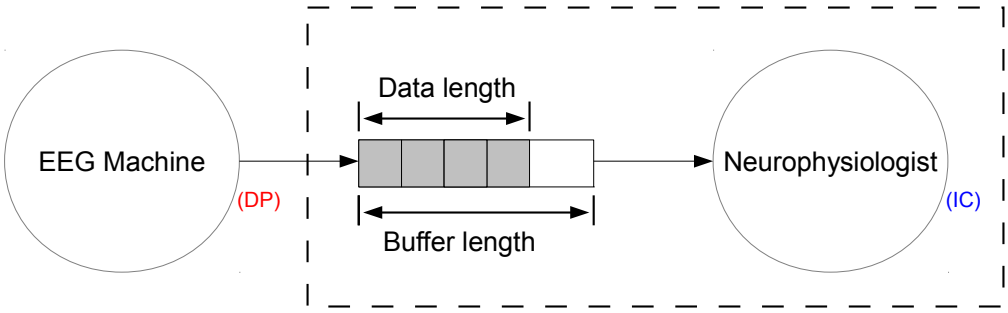

Figure 3.1: This workflow depicts data awaiting processing by the Interpreting Component being persisted to a buffer.

an entity. Therefore, TS data can be defined as:

$$TS = x_{t1}, x_{t2}, x_{t3}, ..., x_{tn} \tag{3.1}$$

where each observation x occurs at time t.

TS data is used to monitor the state of an entity over time and its numeric and continuous nature makes it particularly suitable for monitoring and analysis purposes [65]. As such, it is utilised in a wide range of domains. For example, the monitoring of stock prices in financial markets, the monitoring of meteorological data for weather forecasts and the monitoring of web traffic.

Figure 3.2 illustrates the visualisation of TS data in the form of a line chart. This chart depicts the value of the US dollar plotted against that of the euro from the year 2000 to 2013. The X-axis is representative of time in years and the Y-axis is representative of the value of the US dollar. To calculate the value of the Euro in US dollars at a point in time, simply select the year on the X-axis and move up until you intersect the plotted line, then move across to the Y-axis and you have the Euro's value in US dollars at that point in time.

Time Series Analysis

TS analysis is the inference of meaning from TS data. One of the most common forms of analysis is the visual interpretation of a TS data plot (as depicted in Figure 3.2). There are a large number of analysis techniques for the interpretation of TS data in existence, ranging from statistical techniques such as regression analysis, moving average and seasonality, to artificial intelligence (AI)-based approaches, such as support vector machines and neural networks.

The known applicability of an analysis technique influences whether or not it is used in

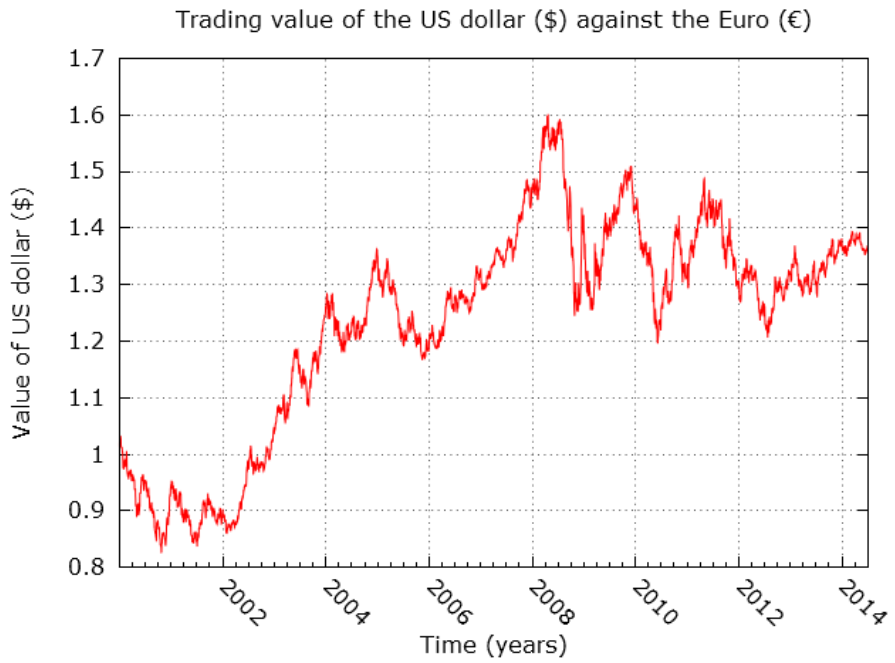

Figure 3.2: A plot of time series data showing the exchange rate of the US dollar against the Euro.

a domain. Other factors that influence analysis include:

i) The quantity of TS data available: this is related to the number of data points in each observation, the frequency at which these observations take place and the time interval concerned.

ii) The complexity of the interpretation process: the transformation of TS data into temporal information, which is carried out to assist, or enable, the inference of meaning from the TS data. The primary intent is the structuring, or presentation, of TS data in a more intelligible format for the entity intended to consume that interpretation.

iii) The knowledge to be exposed from the TS data: This is concerned with the identification of patterns, trends or shapes that add value to the analysis of the data; typically in the form of an objective analysis.

TS is widely used as a means of representing data specific to a domain in an abstracted form. It allows for the analysis and comparison of multiple sources of data, which can be used as the basis for decision-making. Consequently, TS data is a promising candidate for representing knowledge in an accessible manner, allowing for domain-specific information to be discovered, and for key features to emerge through the use of abstract analysis techniques. As such, TS is considered a suitable choice for data representation within the IE.

Analysis techniques for TS data are well understood, it is an understanding of their application and uniformity of approach that is lacking. Libraries containing functionality to analyse data are continually developed for specific data formats in specific domains. This specificity limits both the utility of the method and the impact of the associated results.

Embodying analysis techniques in an IC provides a simple means of highlighting the relationship between data produced and the consumption of that interpretation by abstracting the interpretation process. It encourages the development of generic interfaces for these techniques, thereby preventing over-specification and promoting the idea that functionality should be leveraged across multiple separate domains.

3.1.2 Temporal Abstraction

Temporal abstraction (TA) is the creation of interval-based abstractions from a combination of time-stamped data and interactions that originate outside the scope of that data stream. TA is typically used in a clinical setting, and can be considered analysis of TS data and contextual information that originates outside the scope of the data. It has been previously defined as the detection of relevant patterns in data over time [66]. In effect, it is the combination of TS data and contextual information relating to an event as evidenced by the data.

TA uses a domain-specific ontology to encapsulate the terms, concepts and relevant relations, and a method called the *knowledge-based temporal-abstraction* (KBTA) [67]. It is postulated by Shahar [68] that the explicit declaration of knowledge requirements has benefits in the form of increased reusability, easier acquisition, reduced maintenance and the potential for increased sharing between domains. It has been used extensively for the process of acquiring and maintaining domain-specific, temporally abstracted knowledge in a clinical setting.

It is quite common that multiple types of data are continually being generated in the diagnosis and treatment of patients. This data can be in multiple alternate formats including physiological time-stamped data, records of routine check-ups or treatment plans underway. This data is used in the identification of trends/patterns within a patient context and the specific domain in which it occurs.

The temporal model incorporates *time intervals* and *time-stamped points*. A time interval is a pair of TS values (t_{start}, t_{end}) representing the interval's start and end time. A *time-stamped point*, T, is a 0-length interval represented by the same TS value (T_{point}, T_{point}). An action or process that originates outside the data stream is considered an event for which a description shall be attached.

One of the strengths of TA is its ability to take disjointed data sources and identify

relationships from them. A perceived weakness is its divergence from TS analysis and its reliance on domain-specific tools and functionality. As such, a number of beneficial approaches provided by TS analysis are isolated from TA, and vice versa, due to the manner in which they are realised. This was a motivating factor in the development of a generic workflow within the IE.

3.2 The Interpreting Component

The IC is where data is processed to produce information. This can be done using one of the variety of approaches described in Chapter 1. Each of these approaches requires specific architectural components to support its operation. For example, digital signal processors may be required to implement data transformations in a timely fashion or a data source may be required to train a support vector machine.

In addition to these automated techniques, an IC may be realised by a human with expert domain knowledge. Incorporating the human into the interpretation process brings with it unique challenges. The idealised expert will always give the same interpretation to the same input data. However, humans can be far from ideal and their analysis may vary depending on factors outside the IE.

A workflow may rely on peer review and community validation for acceptance. Variances between experts' interpretations highlights the value of collaborative environments for achieving consensus. In this respect, an IE with a human IC is the most challenging. In the context of the use case, the IC is the neurophysiologist and the DP is the EEG machine connected to a neonate. Sitting in front of the machine, the neurophysiologist can process the data at the rate of production and deliver an interpretation, which may be used for subsequent treatment. If in this scenario, data is not stored, consequently its interpretation cannot be validated. In practice, it is common for the interpretation to be stored separately to the raw data and hence the link between them may be difficult to recover.

A workflow can be created to address these issues. In Figure 3.3 the IC contains a sub-workflow in which a number of neurophysiologists have access to the raw data from the EEG machine and an appropriate buffer. They can asynchronously navigate that data and annotate it with their observations. Once annotated, these observations become available to other neurophysiologists and thus a collaborative environment is created using this feedback mechanism. A secondary advantage of this workflow is that the observations and ultimately the consensus interpretation is stored with the raw data thus preserving this valuable link. As part of the workflow it can be seen that one neurophysiologist is charged with communicating the interpretation to the DC. The transparency of this arrangement helps to eliminate miscommunications that could occur in a human-centric network, that

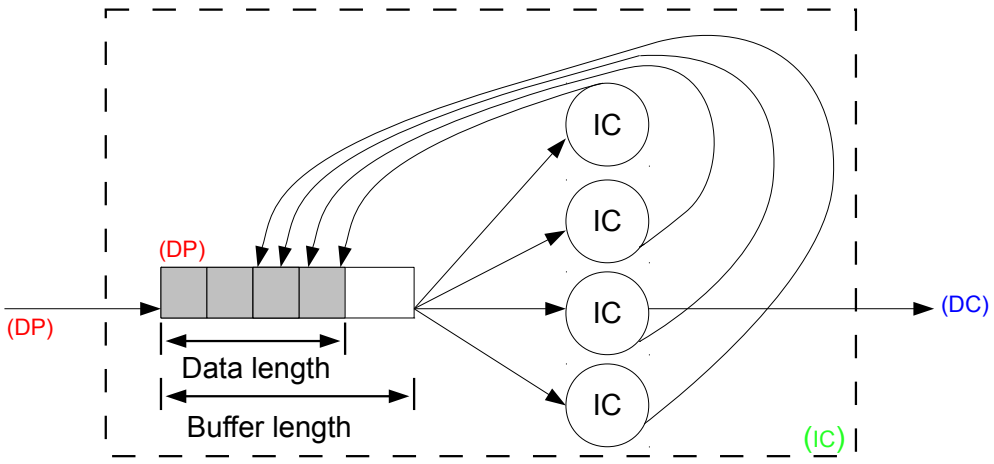

Figure 3.3: An IC with a sub-workflow comprised of multiple ICs, interpreting data asynchronously.

is, only one neurophysiologist speaks authoritatively to the DC on behalf of the group. Analysing this network more closely, it can be seen that the buffer acts as a DP and a DC.

3.2.1 Capturing Expert Knowledge

Expert knowledge can be used for the development of data analysis techniques, be it training or validation data for a machine learning algorithm or as contributions to a knowledge base that a decision support system depends on. The workflow described in Section 3.2, and depicted in Figure 3.3, is producing expert knowledge by storing the link between the experts' consensual interpretation and the raw data.

As such, the IC is facilitating the capturing of expert knowledge. This expert knowledge can be used to develop improved tools and techniques that assist with the interpretation of data. Automated solutions, improved interpretation techniques and advanced visualisations are just some of the solutions that could be enabled from this workflow. The mechanism used to facilitate this process is the annotation.

In Section 3.2.3 a review of technical work relating to annotations is presented. This is followed by a discussion on annotations in the context of the use case (see Section 3.2.4) and finally annotations in the IE (see Section 3.2.5).

3.2.2 Interpretation In The Context Of The Use Case

The interpretation process requires the neurophysiologist to review EEG data acquired from a neonate. This is achieved by viewing screenfuls of EEG data, comprising multiple channels of EEG data over a small time scale. For example, Figure 3.4 depicts a screenful of EEG data that is comprised of 8 channels over a 10-second period. The neurophysiologist must review

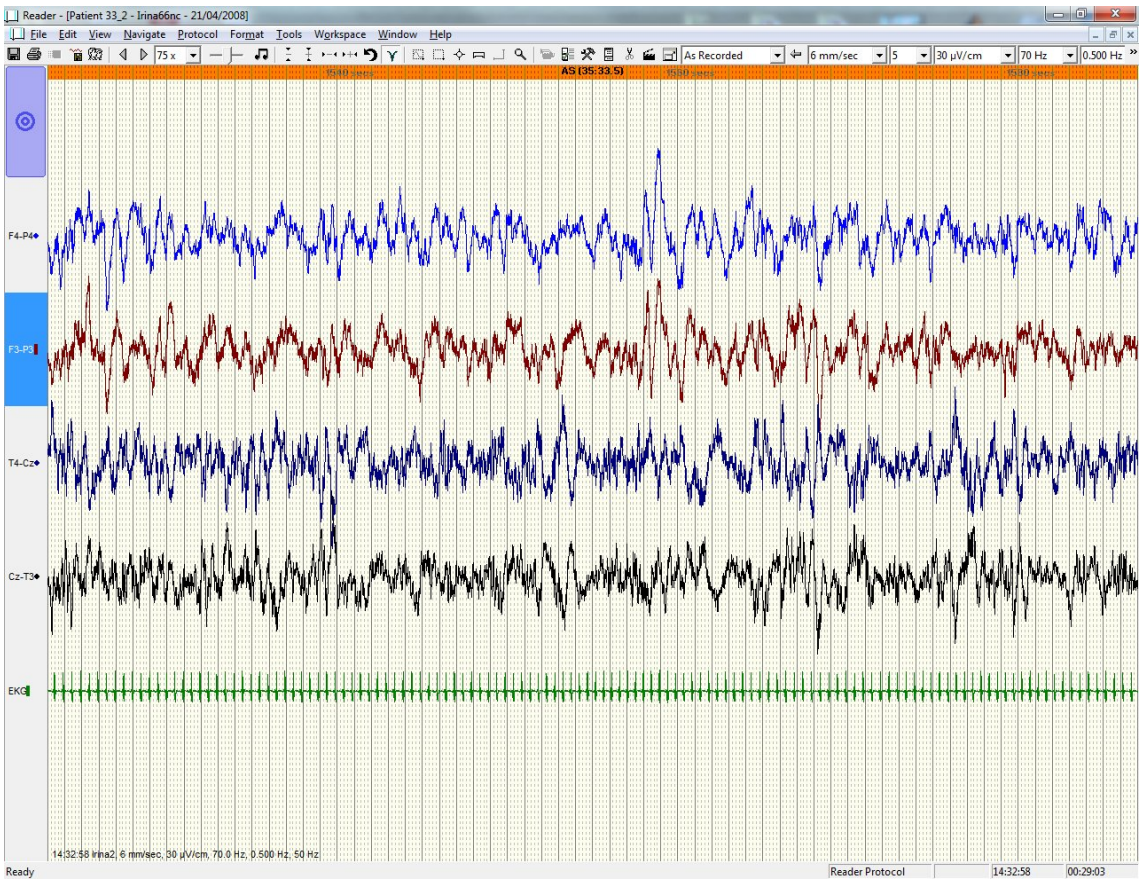

Figure 3.4: A visualisation of EEG data recorded from a healthy term neonate.

this data and identify any waveforms, trends or shapes indicative of a neonate's condition, *e.g.* healthy, unhealthy, recovering, worsening etc. Furthermore, the neurophysiologist must be capable of differentiating between EEG data and artefact or noise, as these may affect interpretation. Adding to the complexity is the fact that analysis occurs across multiple channels of data and spans multiple screenfuls of data.

3.2.3 Annotation Of Data

The annotation of data, be it paper-based written material or digital data, represents an engagement with the material [69]. This engagement results in the creation of *metadata* that serves a purpose desired by the individual creating the annotation. The defining characteristic of an annotation is its intended purpose. For example, the act of highlighting a sentence might serve as a signal for future attention or as a memory aid.

It is important to note that this engagement is both user-defined and user-specified. It has the capacity to evolve over time and be adapted to different demands and desires, as long as there are appropriate interfaces to the data. Traditionally, annotations were

paper-based. These took many forms, such as highlighting, underlining, comments, color coding, short notation, calculations, references to other material, writing in the top/bottom margins of documents and keeping pages of separate notes for a document.

The reasoning behind paper annotations is examined in [70]. The motivations for the different types of annotative marks made on paper, as well as the perceived benefits for each kind are discussed. A comparison is then made to Web 2.0 technologies and a prototype for annotating web pages in a paper-like fashion is presented. This provides a foundation from which to consider design guidelines for an annotation system that could assist with the integration and capturing of expert knowledge.

A Digital Annotation System

Digital annotations are a natural evolution of the traditional paper-based approach. The design and implementation of digital annotation systems that mimic the functionality of their paper-based counterparts in medical and technical fields are presented in [71], [72], [73].

A survey of the most desirable functionality for the digital annotation of data, in [74], found that functionality available in traditional paper-based annotation schemes, such as marking up documents, writing in margins and writing at the top of documents is not as desired as more advanced features, such as the ability to annotate images, support for multiple document formats and the ability to search annotations. This is indicative of the higher expectations users have of digital annotations. Functionality already available to them on paper is taken for granted. As a result, when designing an annotation system for the IE it was considered a priority to support the functions already provided by paper-based systems, as these would be expected by users, in addition to identifying advanced features which would be of value.

An advantage of digital annotations is their ability to implement functionality that paper-based schemes cannot support. Examples include:

1. Support for comment-sharing in documents.
2. Compact display of indexed annotations and relationship discovery.
3. Collaborative sharing of comments with selective viewing permissions [74].
4. Support for editing of own and other users' annotations [75].
5. The ability to search a collection of documents based on keywords within annotations.
6. Annotation of TS data and searching based on similarity to annotated segments.
7. Extraction of meta-data that, combined with further processing, allows for the creation of further valuable data, such as training sets for machine learning techniques.

8. The ability to circumvent the rigidity of electronic health records by allowing the creation of data not envisaged and accommodated by the original document schema architects.

The collaboration between different disciplines in healthcare and the complexities involved in implementing efficient information exchange is discussed in [72]. Difficulties with written and electronic medical forms, the rigid structure imposed by forms, and how this constrains practices are outlined. A model for communication via forms that support collaboration by accommodating different levels of interpretation is presented.

The addition of annotations to electronic health records (EHR) as a means to promote collaboration, coordination and awareness is presented in [71]. The loss of created knowledge due to the strict structure of the EHR is discussed, along with a proposed solution: the use of annotations to allow documents to overcome this rigidity and provide support for non-envisaged functionality.

A tablet-PC-based software tool for collaborating on TS data is presented in [75]. It provides functionality to allow the researcher to browse datasets, create, view and edit annotations (both their own and those created by others). The use of annotated data to create feature-specific datasets for the training of diagnostic algorithms is proposed.

3.2.4 Annnotations In The Context Of The Use Case

Paper-based EEG monitors allowed for manual annotation through the addition of handwritten notes indicating acquisition events and features of interest in signals. More modern, software-based monitors support the addition of simple textual annotations both during acquisition and subsequent analysis. Some systems, such as the Natus Olympic Brainz Monitor [76], allow for annotations to be viewed and added from a remote review station.

The goal of the annotation system in an IE is to extend this functionality to create a common medium for the interchange of signal meta-data between all actors (human and software) in the IE. Humans use the annotation system as an aid for analysis, collaboration with colleagues, identification of features of interest and for training purposes. Software agents provide annotations that can assist humans with their analysis. Furthermore, software agents can make use of annotations provided by humans, both as a resource to be searched and as a source of training data. The resulting meta-data is therefore of benefit to all actors in the system. The following sections present a number of interesting use cases which discuss the annotation systems' goals in more detail.

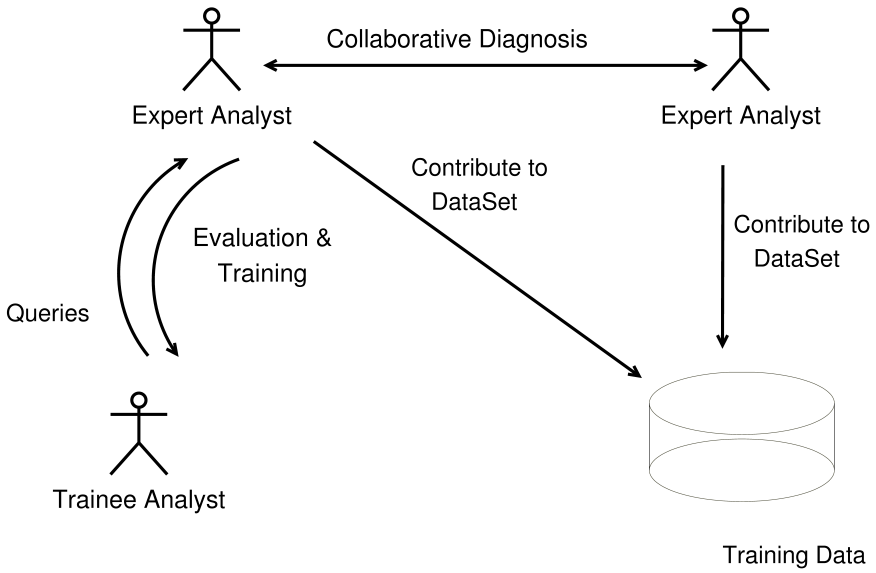

Figure 3.5: Annotations facilitating cooperation between the human actors.

Expert-to-Expert Collaboration Via Annotation

While reviewing a TS data stream, an expert analyst identifies a feature of interest. The expert analyst creates a public annotation, incorporating the data, and continues reviewing the data stream. When the annotation is saved, the information entered, such as the waveform type, duration, start-time, end-time, and affected data feeds, is stored in a relational database.

The result is a collection of annotations that can be independently verified by another expert analyst. Furthermore, if expert analysts do not agree on a waveform classification, then a discussion takes place via the addition of comments. The expert analyst and his/her peers can also express their level of confidence in the classification. Eventually, a consensus is reached on whether or not the waveform fits the classification assigned to it. In this manner, the annotation system supports a multitude of expert perspectives simultaneously.

Expert-to-Analyst Training Via Annotation

A trainee analyst is reviewing a TS data stream. The trainee is uncertain about the classification a particular waveform should be given. The trainee analyst annotates the data with a query, stating their own views and asking for guidance on how to distinguish between the different types of waveform classifications.

An expert analyst sees the query and responds by distinguishing between the waveform classifications and annotating the annotation. The annotations evolve into a thread-based

discussion related to the type of data being analysed. This aids both the transfer, and generation, of knowledge and assists in overcoming the geographic constraints on expert knowledge.

Automating Expert Analysis

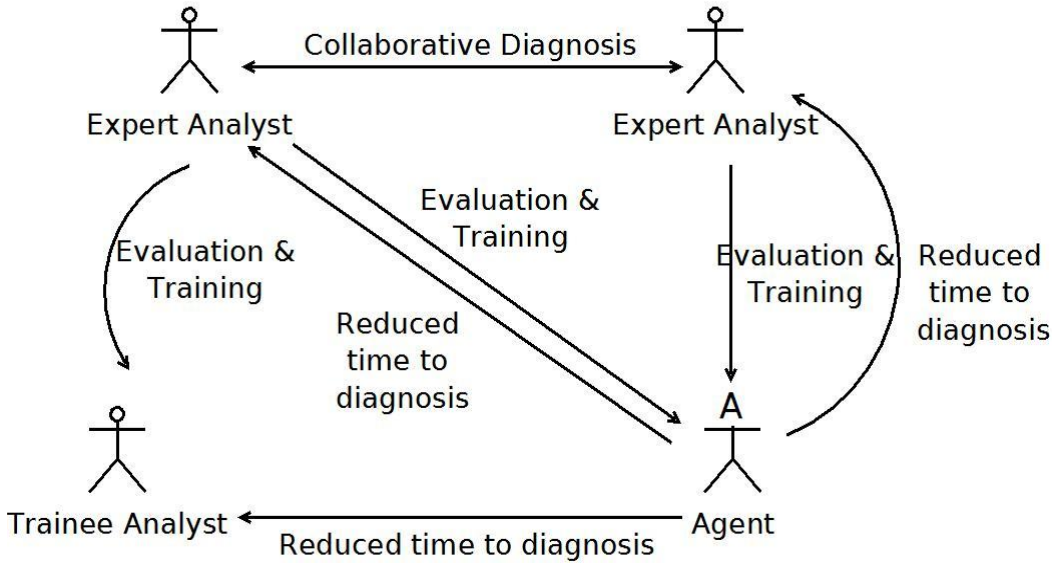

Figure 3.6: Annotations facilitating cooperation between agent-based and human actors.

Reasonably accurate seizure detection algorithms are available (see Section 6.5), but are not considered sufficiently accurate for diagnostic purposes. However, if the output of these algorithms is presented to an expert analyst in the form of annotations, these segments of interest have the potential to assist diagnosis.

Consider the case of an expert analyst examining a neonatal EEG data stream in order to determine whether the patient is suffering from seizures. The expert analyst can scan EEGs visually at two 10-second pages per second [32]. The interpretation of 24 hours of EEG recording therefore requires 72 minutes of analysis time per patient. If the expert analyst, on opening a data stream, can view a list of annotations that have been generated by the seizure detection algorithm, then he/she can jump immediately to these segments and confirm whether or not seizures are present. This plays to the strengths of both types of analysis: the ability of algorithms to rapidly process large volumes of data, and the ability of humans to definitively categorise waveforms.

Complementing the objective of assisting human interpretation, the annotation system could assist with interpretation by software agents. As the system is used over an extended period of time, a large number of waveforms are classified by expert analysts. This expert knowledge would become a valuable source of data for the development of new ICs.

Annotated signal segments can be extracted to use as training and evaluation datasets for machine learning algorithms. These segments could also form the basis of ICs used to compare new data against a database of known waveforms of a particular type.

3.2.5 Annotations Within The Physiological Data Server

The annotative functionality chosen for the Physiological Data Server (PDS) (For more information on PDS see Chapter 5) is a combination of the features provided by traditional paper-based systems, with the collaborative and knowledge-sharing capabilities facilitated by digital annotations. This functionality is then extended to include meta-annotations in the form of comments and confidence measures. Top-level annotations fall into two categories: *events* and *waveform classifications*. Events indicate occurrences at a more abstract level than that of the waveform.

- *Waveform classifications* - An annotation denoting the observation of a type of waveform. Representative of the categorisation, or classification, of data segments by an expert or software process. They are considered bottom-up as they are identified directly from the data.
- *Event type* - An annotation denoting the observation of an event. Event types are external observations. They originate from a more abstract level than that of the waveform and are considered top-down in that they originate from a context outside the data.

Annotations within an IE are denoted by a start-offset and end-offset in milliseconds relative to the beginning of a data stream, incorporating one or more feeds of data from that stream. Meta-annotations, in the form of comments (general free-form text markup of data), can be added to any top-level annotation. However, confidence measures (such as the Likert scale [77]) can only be applied to waveform classifications.

In the context of the NICU use case, annotations can originate from a number of sources:

1. *Acquisition location*: Modern EEG acquisition software typically annotates the output file being produced with annotations indicating events of interest during acquisition, such as when an impedance test is performed or a loose lead is detected. These can be supplemented by notes entered manually by staff either at the bedside or at a review station. Examples include notes indicating that the patient has been moved or highlighting features of interest such as suspected seizures or signal artifacts.
2. *Remote reviewers*: Analysts at remote locations are free to create new top-level annotations of any type. They can also add comments to any top-level annotations.

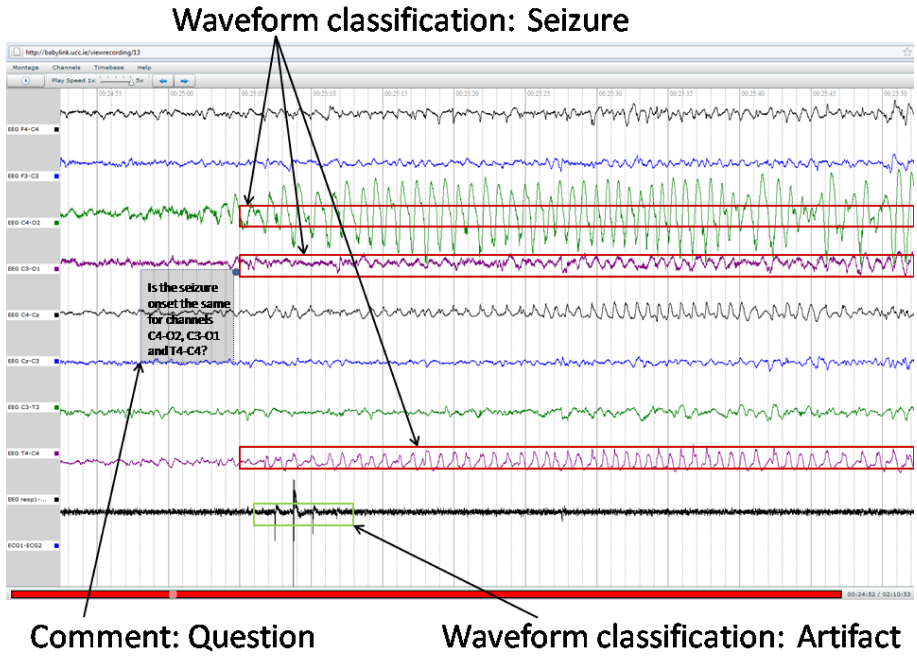

Figure 3.7: An annotated EEG being reviewed in the web-based viewer.

As noted above, reviewers can only add confidence measures to existing waveform classifications.

Realising Feedback Through Annotations

Annotations were chosen as the mechanism to realise feedback, as described in Section 2.2.2, within the PDS. In order to provide annotative functionality, a means of visualising and interacting with TS data was required. To that end a web-based viewer capable of visualising and annotating TS data was developed and is described in more detail in Section 5.2.

Operation And Visibility

The simplest annotation use case is as a means for users to leave notes for themselves. This is useful for memory aiding, indications for further analysis, *etc.* The system supports private annotations for this purpose. A new annotation is created by selecting an option from the drop-down menu at the top of the application window. A mouse-click is then used to indicate the annotation starting point. Once the starting point has been selected, a new, empty, annotation appears. The on-screen representation consists of an annotation summary component and two arrows indicating the start and end recording offsets. The start and end arrows can be dragged to change the recording segment referred to by the annotation. The annotation summary consists of four sub-components indicating the

annotation type, description, visibility (public/private) and affected channels. These can be specified by clicking on the respective subcomponents. The annotation summary also contains a button that saves the newly created annotation to the server. Alternatively, a drop-down menu option is available to save all newly created annotations.

A more advanced use case is when an analyst shares annotations with peers. When the annotation is shared, all other users viewing the recording in question are notified as soon as the annotation is saved via a server-push messaging system. The other users can then add meta-annotations by rating or commenting on the annotation. The rating system uses a Likert scale [77] to enable reviewers to provide a measure of their confidence in the accuracy of the annotations. The available options are "strongly disagree", "disagree", "neutral", "agree" and "strongly agree". Users may also comment on the annotation. The result is a thread of ratings and discussion pertaining to the annotation, facilitating consensus diagnosis or training instruction.

The annotation summary components for public annotations contain a label indicating the number of associated ratings and comments. If any of these items have not yet been viewed, then the label text is rendered in a different colour to draw the user's attention to the new content. Hovering the mouse pointer over an annotation summary component causes it to expand and to display the ratings and comments below. Clicking on an annotation summary component brings up a more detailed user interface that allows new comments and ratings to be added. Any comments or ratings added to an annotation are immediately saved to the server automatically.

Users can navigate quickly from one annotation to the next using a pane at the bottom of the viewing window that displays the annotations in the vicinity of the currently visible recording segment. Annotations currently visible on-screen are displayed in the center, with those preceding and following shown on the left and right, respectively. A complete list of all annotations attached to the data stream can be viewed via a drop-down menu option. This view contains visibility settings for the various annotation categories and allows annotations to be grouped.

The creation of a large body of data that has been annotated by experts is also of benefit to users. Domain experts can identify users whose work is relevant to them and view all of that expert user's annotations by visiting the expert user's profile page on the system. This feature encourages knowledge-sharing and collaboration by providing a platform from which trainee analysts can learn from expert analysts, and likewise expert analysts can monitor the progress of trainees.

Table 3.1: Agent capability to create annotations.

Case	Event	*FOI*	Comment	Question	Meta
Acq. software	✓	✗	✗	✗	✗
Staff at acq. site	✓	✓	✓	✓	✓
Remote Reviewers	✓	✓	✓	✓	✓

Benefits Arising From The Annotative Ability

The means to quantify and capture expert knowledge is facilitated through annotative functionality. Annotations are created by analysts as they review the data, resulting in an electronic record of that interaction. This facilitates the categorisation of data which subsequently enables features of interest (FOI) within the data to be analysed collectively. It is envisioned that this will result in a large corpus of expert-annotated data being accumulated, which will have a much greater value than raw data alone.

Furthermore, the provisioning of functionality for annotations will provide a rich medium of communication that will facilitate collaborative analysis and form the basis of a co-operative environment for data interpretation. The introduction of functionality for human-to-human collaboration will allow for improved patient care by facilitating more accurate and timely diagnosis. Furthermore, the annotations added by expert analysts will provide a valuable resource for the creation and evaluation of ICs.

The combination of functionality for the processing of data as it is acquired by the system, and the automated annotation of that data, would assist in the identification of segments of interest. These informative, and automated, annotations are seen as highly beneficial.

3.3 The Data Consumer

The DC uses information from the IE to determine the appropriate action. This can be anything from simply invoking an actuator, to the initiation of a complex process designed to achieve specific goals. In the context of the use case, the NICU is the entity that acts as the DC (see Section 2.2.4). The NICU is responsible for interpreting any prescribed treatment received from the neurophysiologist and acting on it, administering treatment as a result if necessary. The personnel, within the NICU, are charged with caring for the neonate in question, and so acts on the information received from neurophysiologists (IC) to deliver the appropriate care. In treating the neonate, the NICU's actions may be viewed as a feedback loop in our network, the exercising of which results in a change in the state of the neonate. This change is reflected in the data emitted by that DP and is used to drive a

subsequent invocation of the workflow. This process repeats until there is no further need to apply treatment. In terms of the abstract workflow the DP converges to a desired goal state, this is interpreted by the IC and relayed to the DC, who reacts accordingly.

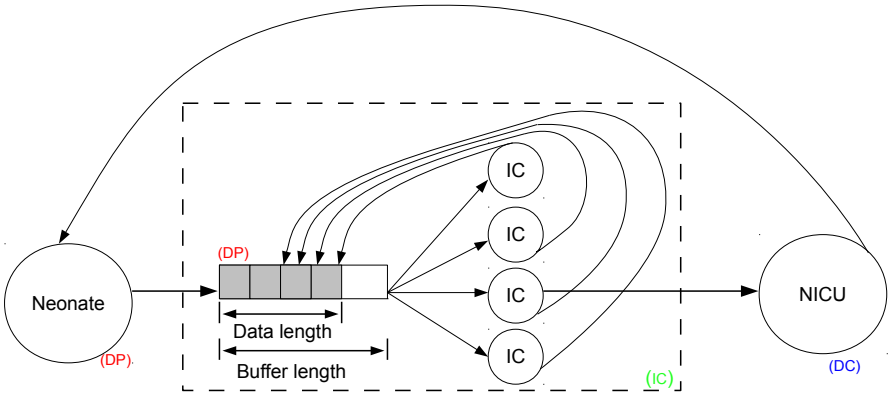

Figure 3.8: The complete workflow including the DC with a feedback loop to the DP.

The complete use case workflow is illustrated in Figure 3.8. It is interesting to note the differences between the two feedback loops in this workflow. In the first, the feedback information is captured and stored and may subsequently be inspected. In the second, the feedback loop does not result in information being stored explicitly, rather, in that case the state change is represented by the subsequent behaviour of the data-producing component. This means that the change in state in the DP is not inspectable unless the history of those changes are stored.

It can be seen from Figure 3.8 that the history of state changes in the EEG can be stored in the buffer. However, to preserve the link between change in state and the action performed to precipitate that change, it would be necessary to propagate that action at the start of each workflow invocation. Thus, the action is communicated over the feedback loop concatenated, and concatenated to the subsequent data stream emitting from the DP. In this manner, the provenance of the new data can be recorded.

Supporting features like provenance and audit trails, requires the addition of some "memory component" to the system. The location and number of these memory components will depend on the application workflow.

3.4 Overview Of The Workflow

Within an IE there are constraints that affect the frequency at which data is produced, data is interpreted and data is consumed. While these constraints are specific to each instance of an IE, there are a number of generic requirements arising from the workflow, which may need to be addressed. Some of these include:

i) Data generation and storage: Often the quantity of data being generated is greater than that which can be processed. This results in an interest in methodology for separating data into what needs to be processed and what does not. As well as what needs to be stored and what can be discarded.

ii) Supported data: TS data was identified as the data format that the IE would support. It is robust enough to support monitoring in a generic manner, such that single-point, periodic or continuous monitoring, and analysis, can be accommodated. TS provides a sufficiently fine-grained measurement and is readily compatible with a broad range of domains.

iii) Real-time processing: Processing data in real time is typically associated with critical systems. For instance, an aircraft's life support system is of a critical nature. Its designers ensured it processes environmental data in real time (air pressure, oxygen levels etc.), as any delay would jeopardise the safety of passengers on board. A real-time traffic control system for a city is also of a critical nature, as any delay could cause a potentially fatal road accident and/or result in less than optimal use of transport infrastructure, which incurs an associated cost. In both cases, the production, interpretation and consumption of data is occurring at a defined pace, which necessitates real-time analysis, adding to the complexity of that system.

iv) Storage: If data is stored then it can be processed retrospectively. This allows for a trade-off between the storage and processing of data. For instance, when compute cycles are costly it may prove more economic to store data for a period of time and process this data when the cost of compute cycles has reduced. In some instances the data may never be processed, but its storage is necessitated as a means of provenance. For example, for Service-Level-Agreements in cloud configuration, metrics for services provided are often persisted and only analysed at the request of a customer.

These outline some of the technical requirements facing a generic IE. These challenges directly relate to the storage of data, the maintenance of provenance and the need for historical records of interaction. Sometimes it may be necessary to augment the system with components that are specific to the application domain. An example of this will be given in Chapter 5 when discussing the PDS.

3.5 Summary

This chapter outlined requirements arising from the realisation of a generic IE, identified TS data as a suitable data format for that environment, and discussed the means for integrating and capturing expert knowledge. Two approaches to facilitate expert knowledge were reviewed: TA and annotations. Annotations are considered more appropriate due to their direct interaction with the data and their ability to support bottom-up observations, identifying features directly from the data, and top-down observations, originating from a more abstract context outside the scope of the data. An understanding of how the PDS's annotation system would operate was also presented with a series of use cases to highlight the benefits of such a system.

Chapter 4

Implementing The Interpreting Environment

In Chapter 3 the IE workflow was presented and in effect it is possible to view the workflow description as a "blueprint" describing a specific application supported by the IE. It will be seen that this "blueprint", in conjunction with a collection of re-useable software components, called the *IE toolbox*, can be used to construct a specific application.

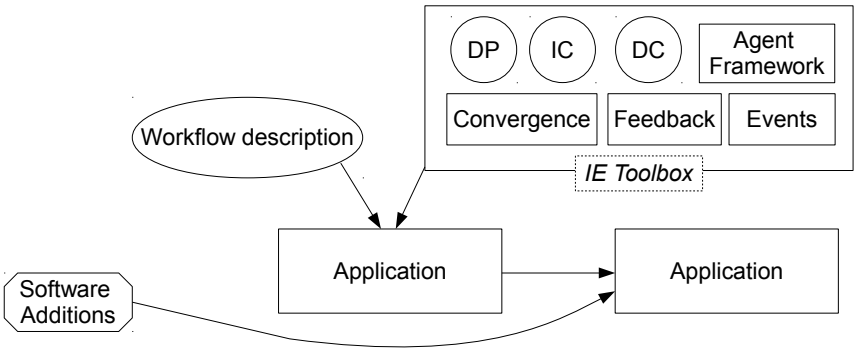

Figure 4.1: An abstract view of an application being constructed from a workflow description and the IE toolbox.

This chapter explores the IE toolbox, which can be used by an application engineer to simplify the construction of an environment for capturing, interpreting and consuming TS data. Employing this toolbox will also help standardise the construction of these environments, thus instilling confidence in the construction process. The toolbox is constructed to support the IE components and their interactions as depicted in Figure 4.1.

The rest of this chapter is organised as follows; the design philosophy of the IE is

discussed in Section 4.1, this is followed by a high-level overview of the IE Toolbox's architectural components in Section 4.2, after which, each component is discussed in more detail in subsequent sections. Finally, a number of distributed IE architectures are given, to illustrate how the partitioning of IEs among various geographically distributed sites results in exposing high-level characteristics, such as the creation of centres of expertise, which can be exploited in specific application domains.

4.1 Design Philosophy

4.1.1 Design Principles

Best practice software engineering design principles were employed in the creation of the IE toolbox. These include:

Separation of Concerns (SoC) - This involves maintaining a clear distinction between the implementation of each functional component such that no component is implicitly dependent on another. This approach allows for improved maintainability and robustness. The IE was primarily developed using an Object-Oriented (OO) approach and the architectural design incorporated design patterns as appropriate. For example, the Model View Controller was used where outward-facing visualisations were required by stakeholders.

Modular Design [78] - Modularisation of the code base is a complementary approach to SoC. It results in a clear separation between components and provides improved maintainability and increased reusability. It enables individual components to operate independently, and to be reused.

Portability - There was a desire to ensure sufficient abstraction between the IE's application logic and the underlying operating system (OS). In the context of the use case, the majority of the software designed and developed for a clinical setting runs on the Microsoft Windows OS [79]. By providing a portable solution, support for the environment across a broad range of OSs was maximised.

Operability - Monitoring and analysis systems support critical infrastructure. Ensuring they operate as expected in production is important. To that end a combination of Unit Testing (software testing method whereby source code is broken down into small units and tested in isolation to enure they operate as expected) and dependency injection (a design pattern that decouples software components through the implementation of the dependency inversion principle) were employed throughout the development process.

4.1.2 Technologies Employed

Java - The IE was developed using the Java programming language [80]. This enabled an agnostic approach to the underlying OS and nicely aligned with the aforementioned design principles. Java's OO nature assists with the modularisation of the code-base. Java's inter-operability across OSs makes it a widely adopted language for applications. For instance, the Physiological Data Server was required to run on both Linux and MS Windows. The portable nature of the IE components provided the required support.

Representational state transfer (REST) - This was chosen as the most suitable architectural style for components to communicate with one another within the IE. REST [81] assists in defining the architectural constraints between the system's primary components, by using existing standards provided by the World Wide Web. It not only enables the IE components to be easily distributed, but it also enables the integration of distributed IEs, thus, allowing for the construction of rich application environments.

Jetty [82] is an embedded HTTP server that is used for requesting and serving data by the IE. It enables ubiquitous access as HTTP is the communication protocol over which the World Wide Web operates. The server defines the connection socket from which the web portal is accessed, where requests should be addressed to and specifies the format of RESTful requests and responses.

Extensible Markup Language (XML) - XML provides a markup language in which the configuration of individual components can be defined. It was decided to use XML because it is human readable and works well with **RESTful** services. Due to the overheads associated with data transfer, it was decided not to encode data in an XML format but rather to use a more compact binary representation for data storage and transmission.

Binary attachment with web services - Compact binary representation for data storage and transmission. The original design of the system called for these interfaces to be implemented using Simple Object Access Protocol (SOAP). Although SOAP is widely supported across development platforms, it was found that this support did not extend to SOAP extensions for the efficient transmission of binary data (SOAP with Attachments and Message Transmission Optimization Mechanism). As, for reasons of efficiency, the environment involves the transfer of significant amounts of data in binary form, simple HTTP requests, which are supported by practically every platform, were used instead.

4.2 The IE Toolbox

The three components (DP, IC, DC) encapsulate the core functionality for the production, interpretation and consumption of data. An Agent Framework (AF) is used to facilitate the extension of this functionality and is the means through which scalable and elastic resources can be accessed.

The modular design of the IE enables a loose coupling of components from which the construction of sophisticated solutions can be realised. Transparency between components allows for SoC between communication, data transfer and processing functionality in the environment. This is intended to support a higher level of maintainability and stability [83].

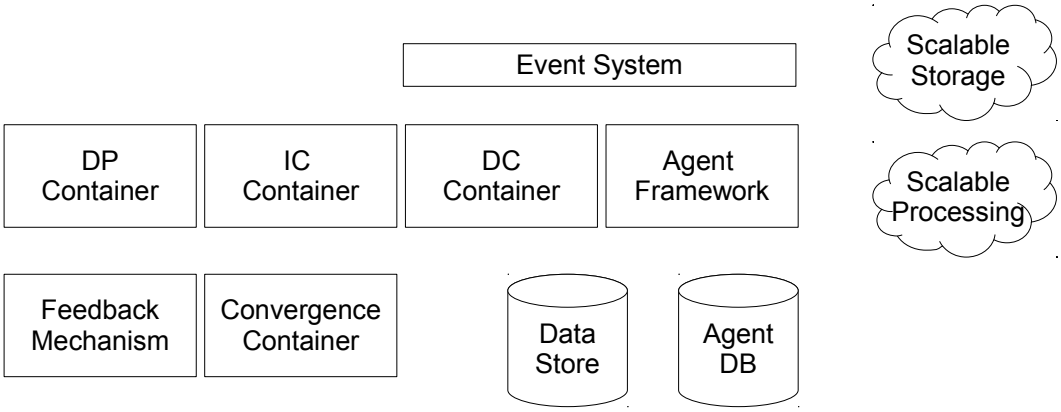

Figure 4.2: Components of the IE Toolbox.

The IE Toolbox can be used by application developers to create instances of the IE representing domain-specific applications. Using these components results in the rapid realisation of standardised environments due to component reuse. The components included in the IE Toolbox include:

- The Event System: It provides communications between components (see Section 4.3.1)
- The Agent Framework: It provides a mechanism for incorporating new computational components as the system evolves. This, for example, allows for the periodic upgrading of the system with more sophisticated ICs, without impacting on the rest of the implementation (see Section 4.3.2).
- DP, IC and DC containers: These provide standardised interfaces with the remainder of the IE. Specific DPs, ICs and DCs may require the provisioning, by the application designer, of adapter components mapping the specific DP, IC and DC interfaces to the standardised DP, IC and DC container interfaces.

- The Data Store: It provides the IE with a "memory component" which can be used to record state and state transitions, audit and feedback information (see Section 4.5).
- The Feedback Mechanism: It allows for the incorporation of state information into the next iteration or instance of the workflow. Each workflow iteration or instance is precipitated by the arrival of feedback information. In essence, feedback resembles a workflow acting as a DP. In practice, feedback information will be recorded in the Data Store and this record will also be used to support data provenance. An application's specific component used to determine when the workflow has converged to a specific state may also be accommodated by the provision of a convergence container.

4.3 The IE Architecture

The components of the IE Toolbox can be combined into an architecture that can be used to realise workflows. Elements of this architecture are now described.

4.3.1 The Event System

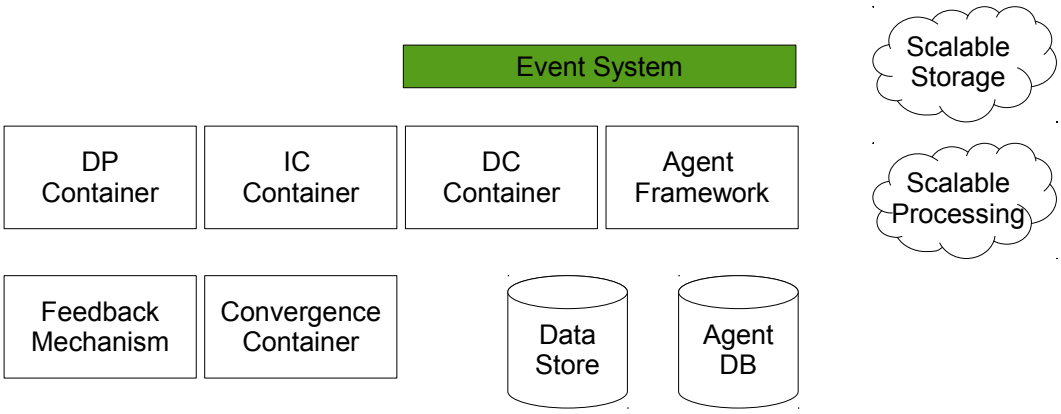

Figure 4.3: The event system facilitates communication within the IE.

Event-driven programming (EDP) is a paradigm where the control flow is determined by the occurrence of events. These events can originate from user actions (mouse clicks, key presses etc.), sensor input (arrival of new data, the breaking of threshold values) or from communication with other programs. In the context of the IE, events are used to notify other components of an occurrence that may be of interest. This involves the defining of an event and specifying when to create an event of that type. For instance, when a new entity is created an `EntityCreatedEvent` is generated.

EDP provides flexibility by enabling programs to react differently based on the events that occur. It provides for a publish/subscribe model in which users can define events of

interest. If these events occur, notification of the occurrence is sent to those parts of the system that are subscribed. Events may be used as system-wide notifications. They may refer to the arrival of new data, the deletion of data or the change of state of a component within the IE. In principle every component in the IE Toolbox is capable of generating events to reflect its change in state. Furthermore, every component is capable of subscribing to receive notification of an events occurrence.

Judicious use of these events by the application designer is important to ensure that faithful conformance to the workflow description is maintained. It is easy to subvert the explicit workflow description by listening for and reacting to global events in an unstructured manner.

Even though any component of the IE can subscribe to arbitrary system events and use these events to alter its subsequent behaviour, unintended system-wide consequences will not result provided the workflow blueprint is not violated.

4.3.2 The Agent Framework

The Agent Framework (AF) operates as a standalone module within the IE. This separates the acquisition, storage and direct analysis of data from the subsequent processing of that data. This separation allows access to data to be facilitated as part of a monitoring and analysis workflow, while simultaneously enabling the evolution of the environment through the implementation of new agents. The AF allows a user-configurable set of agents to perform analysis on data as they are streamed through the system.

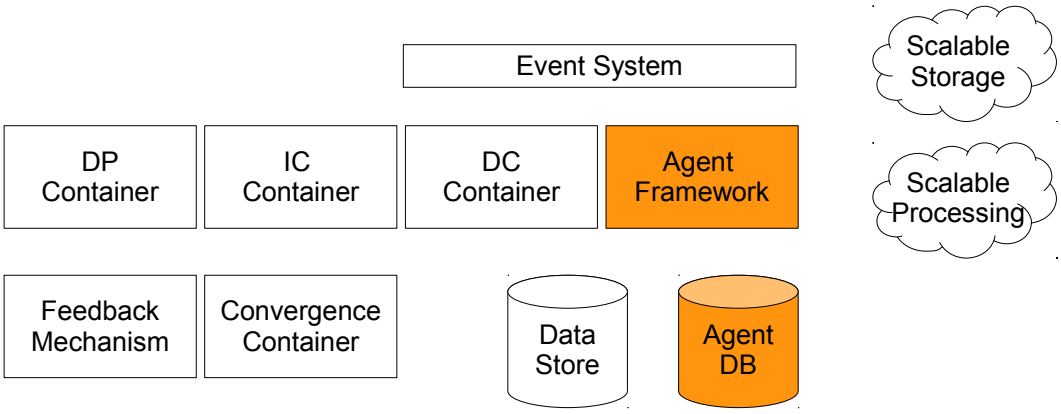

Figure 4.4: The components comprising the Agent Framework.

Agent Philosophy

Agent theory supports many of the characteristics that comprise a complex system. Adopting this design paradigm provides the IE with a means to model how its functionality should evolve, to demonstrate its goal-directed behaviour and to act as a mechanism for the integration of multiple analysis techniques. It is envisaged that individual ICs will be realised through their implementation as agents and will embody a variety of analysis techniques such as those described in Chapter 1. This will enable aspects of a multi-agent system, as described in [46], to be implemented, thus providing the means for a collection of loosely coupled problem-solver entities to cooperate and achieve a desired objective. This is important as the objective can be beyond the individual capabilities of a single technique, but is made possible through that cooperation.

The intentional stance views agents as rational acting entities, with each agent's actions predicted based on their beliefs, intentions, desires etc. [46] The ability to reason about a problem domain based on these characteristics closely mimics the interpretation process. This was another reason for adopting an agent-based approach to the analysis of data.

The AF enhances the IE by enabling a user-configurable set of agents to perform analysis on data, as they are being acquired by the system. The following design requirements were identified:

1. **Configurability**: The set of available agents should be user-configurable.

2. **Intentional stance**: The framework should be capable of determining which of the available agents, if any, are applicable to a particular container (DP, IC or DC).

3. **Persistence**: Agent state will be recoverable in the event of a server shutdown/restart, due to a fault.

4. **Error handling**: Errors generated by agents should be identified, gracefully extracted from the system, and reported to the user.

5. **Security**: Although access control to data available to the framework is outside the scope of the agent framework, agents should not be permitted to perform actions that could compromise system security. This is enforced by undertaking the code review process for agent development, and decoupling the storage of data from the AF, such that the AF's agents may read data, but may not overwrite it. This lead to the design decision that the agent database be segregated from the Data Store.

6. **Data integrity**: Ensuring data integrity requires the enforcement of global constraints. The actions of individual agents may not violate these constraints.

7. **Code reuse**: Code duplication in agent implementations should be avoided by providing well tested libraries to facilitate tasks that agents are expected to commonly perform. This will assist in a repository of utility code to emerge over time, thereby reducing the workload required for subsequent agent implementations.

8. **Resource management**: Agents may be created dynamically depending on the intended application. Memory will be managed by storing only active agents in memory. Inactive agents will have their state written to persistent storage to be reconstructed when required.

9. **Stream processing**: Data should be processed as they arrive at the appropriate container. New data segments should be detected when they are appended to a container and any agents attached to that container should be notified.

Agent Framework Hierarchy

The AF hierarchy is depicted in Figure 4.5. The agent manager is responsible for providing the agents' execution environment. This involves: managing the set of configured agent factories; creating agent instances as needed; managing agent persistence; subscribing to system-wide events and forwarding events of interest to the appropriate agent factories and executing the processing tasks generated by agents.

Agent factories are used to instantiate agent instances on behalf of the agent manager. Agent instances are persisted and reconstructed as required using the Java Persistence API. Event forwarding is performed by determining the set of agents to which a system event applies to, then reconstructing those agents from the persistent store and forwarding the event to them. A task executor, with a configurable number of threads, is used to perform signal processing on behalf of agents.

Agent Factory

The *Factory method* [84] is the design pattern used for agent implementation. This enables agent-related objects to be defined as an interface while allowing subclasses to decide on which class to instantiate. The intent is to provide an abstraction between the related or dependent objects and the concrete classes that implement them. This enables independent components within the AF to articulate the conditions under which something should happen, without having to concern itself with exactly how that implementation is realised.

This results in agents maintaining control of how, when and why they are instantiated as well as providing a separation between instantiation and implementation. Each factory

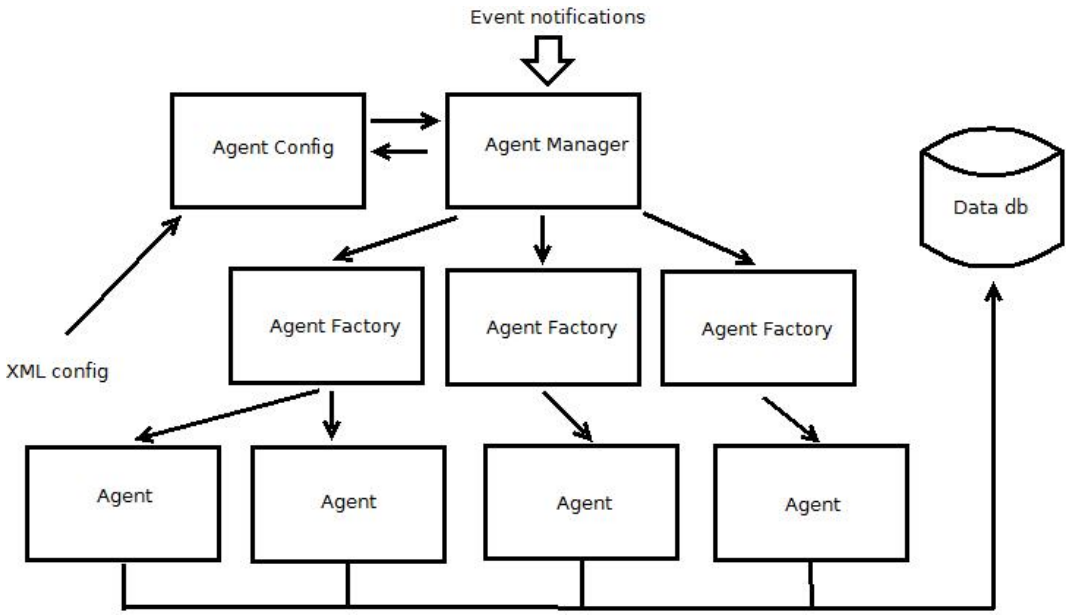

Figure 4.5: Agent hierarchial overview.

facilitates abstraction of the constructor from individual agent classes. It enables the polymorphic creation of agents, as opposed to specifying individual polymorphic behaviour for each agent individually.

Factories contain permutations of constraints, the general case includes a combination of events to listen for, a specific data type, and the presence of certain characteristics within that data type. This is a simple, yet powerful, means of packaging domain-specific knowledge while ensuring that both the knowledge, and the underlying implementation of processes, can continue to evolve and be further developed without affecting the overall platform.

Each factory results in the specification of domain-specific knowledge. This will facilitate the encapsulation of semantic knowledge of a domain, simplifying the process of organising, identifying and arranging associations within that knowledge base. Collecting and disseminating this domain-specific knowledge in a transparent manner can have multiple benefits. For instance, it could be used to assist in the identification of prejudice or bias within a system's knowledge base. This is considered further in Chapter 7.

Deploying an agent to an instance of the IE is then a case of packaging a factory and the associated underlying agents into a JAR file. This enables updated domain knowledge, and underlying techniques employed, to be rolled out in a manageable way. It is hoped that this will assist the evolution of a rich collection of domain-specific functionality.

Agent Database

The AF uses a relational database to store meta-data about the agents in addition to the agents themselves. The AF can be notified of any modification to the Data Store (such as the arrival of new recording data) by subscribing to the events system, as can any of the agents within the IE. The agent database data model is generated upon start-up and results in the creation of the "`agentDb`" and its associated tables. Each type of agent factory and agent can specify their own unique data model enabling greater configurability and extensibility of the IE's architecture. This is achieved by utilising a combination of Object-relational mapping [85] and the Java persistence API [86].

Similarly to the Data Store, the agent database is implemented using HyperSQL. The database statically defines the driver, protocol, database name and table name as well as the connection URL and associated log files. An instance of the database "`agentDb`" with the "`AGENTS`" table is created upon invoking the constructor. It is responsible for creating, shutting down and connecting to the "`agentDb`". Methods are provided for inserting and updating agents, checking if an agent exists by its ID, checking those agents, which may be associated with a container and methods for the serialisation/deserialisation of an agent's state.

Agent Development

Agent implementations are organised into packages, with each package containing an implementation of the `AgentFactory` interface. The agent factory is responsible for determining the set of agents in the package that are applicable to a given recording, instantiating the agent instances, and returning them to the agent manager.

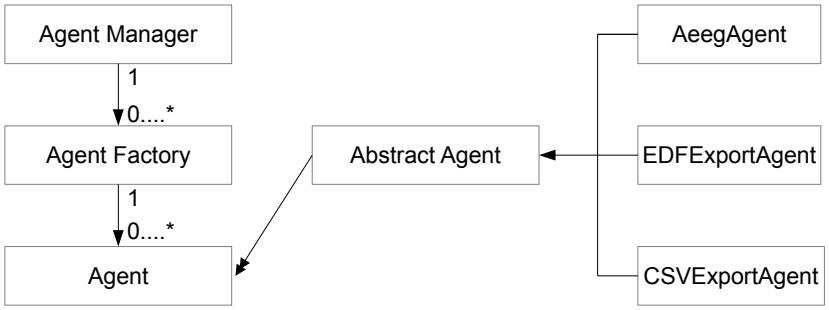

Figure 4.6: UML diagram representing the relationship between the `AgentManager`, `AgentFactory` and `Agent` instances.

Functionality common to all agents, such as task generation and error handling, is defined in the `Agent` interface. The `AbstractAgent` class provides a concrete implementation of shared properties (such as data source and data stream duration), and is designed to be subclassed as needed. Implementations of common functionality for the agents supported by the framework is provided by the availability of utility code (see Section 4.7), as depicted in Figure 4.6.

The AF should be made aware of any agent state that must be preserved between task invocations. For example, the implementation of the aEEG agent, as described in Section 6.4.2, requires that filter variables and 2 seconds of buffered input be preserved between tasks. State persistence is achieved by adding Java Persistence API annotations to the relevant sections of code.

Once agent development and testing is complete, the agent package is deployed by packaging the required class into a JAR file, which is placed in a specific location in the IE. Upon a system restart, the package will be detected automatically. Any new database tables required by the agents will be created and an instance of the agent factory will be instantiated.

Agent Perspective

Agents perceive their environment through the events system, and they effect changes to their environment through modifications to the agent database and the addition of data to the Data Store. These database changes in turn trigger further events that can be acted upon by both the AF and other components in the IE. For example, from Chapter 5, the creation of a new annotation by an agent would result in an event that would be detected by the Physiological Data Server, causing an annotation to appear on the screens of any analysts viewing the relevant recording.

Each event (`EntityCreatedEvent, EntityChangedEvent, EntityDeletedEvent`) has an appropriate handler within the AF. When an event is thrown the agent manager notifies each of the agent factories. Each factory encapsulates the constraints relating to the agent's domain-specific requirements. If the requirements are met, an instance of the agent is instantiated.

For example, if an entity is altered, an `EntityChangedEvent` is thrown. This results in the agent manager notifying all relevant agents. This is achieved by taking the entity ID and querying the agent database for all relevant agents. These agents are loaded into memory and each agent updates the parameters relevant to them.

Task Execution

`agentsWithTasksPending` and `agentsWithUnprocessedData` queues are used for managing the execution of tasks by the AF. `agentsWithTasksPending` contains the ID numbers of agents that have a submitted task for execution and `agentsWithUnprocessedData` contains the ID numbers of agents that have outstanding data waiting to be processed.

An agent task can be generated when an event is thrown (as described in Section 4.3.2), or when an agent task completes its execution and there is enough unprocessed data to create another, as depicted in Figure 4.7. Initially, the details of the data source are updated, then the `agentsWithTasksPending` is checked to ensure there are no outstanding tasks awaiting completion for this agent. If there are, these are added to the `agentsWithUnprocessedData`.

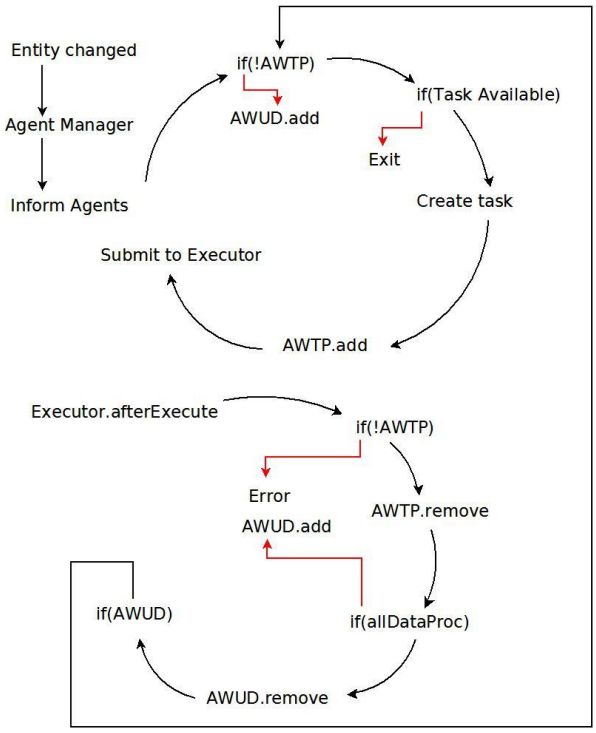

Figure 4.7: The agent processing queue in operation.

Next, the agent is queried to see if it has enough data available to create a task. This is dependent on the agent's implementation as they process time intervals of varying length. The agent task is then created, which updates the amount of agent data processed. A check is made against the amount of data available and if it equals zero it is subsequently removed from `agentsWithUnprocessedData`. Then the agent is added to the `agentsWithTasksPending` queue and submitted to the executor. The executor executes the submitted tasks as a `Runnable`.

4.4 IE Container Interfaces

The container interfaces provide standard interfaces to the IE and specific instances are adapted to these containers by the application designer. The standard interfaces include the following:

The standard interface for the DP, IC and DC are effectively read/write communication channels, which components can create and to which components can attach.

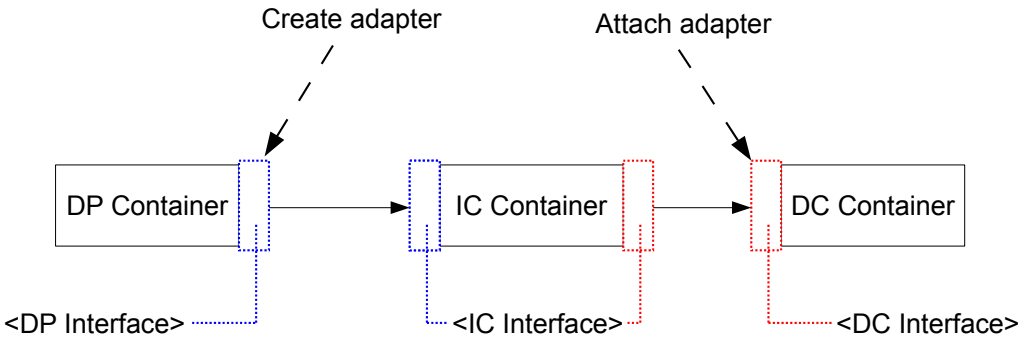

Figure 4.8: DP, IC and DC Containers.

The convergence container takes two or more states as input and communicates "the distance" between these states as its output. In the simplest case, the state container will output a true/false value depending on whether or not its input states are equal.

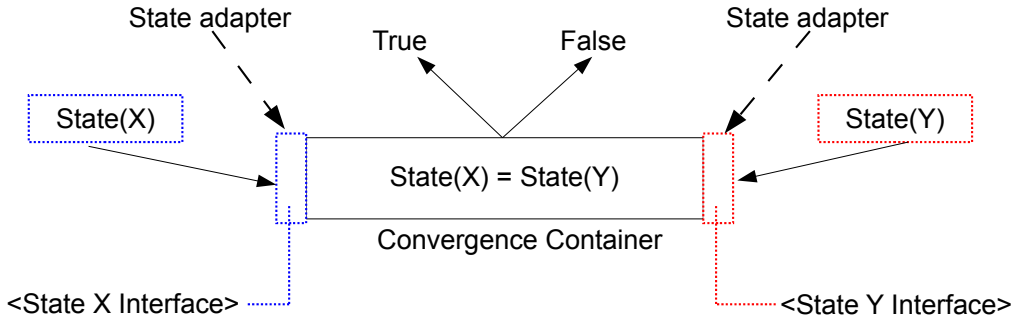

Figure 4.9: Convergence Container.

4.5 Data Store

The Data Store is the component that facilitates the provision of a "memory component", the need for which was highlighted in Section 3.3. It is encapsulated by the `Database.java` class, which statically defines the database driver, database protocol, database name, the connection URL and specifies log files associated with the Data Store.

JDBC is a Java database connectivity technology that defines how a client may access a database and provides the means for updating/querying data within the Data Store. The Data Store is decoupled from any specific implementation of a relational database through its use of JDBC. This decoupling is further supported by the use of Hibernate, which implements the Java Persistence API, and provides object relationship mapping. This enables all entities within the IE to be persisted to tables in the relational database.

It is up to the specific application domain designer to specify how data will be structured, the type of database technology that will be used and the layout of that database etc. As increasing numbers of storage solutions are specified, it is envisaged that the IE will be capable of providing a base for comparison between different storage techniques and practices for application-specific domains.

4.6 Configuring The IE

Configuration of the IE is encapsulated by a Java object when running and persisted to disk as an XML file (`IEConfig.xml`) when deactivated. The format of the XML configuration file is shown in Listing 4.1.

Listing 4.1: XML configuration file for the IE.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<agentConfigModel>
    <configFile>IE\conf\agentFrameworkConfig.xml</configFile>
    <debugModeEnabled>false</debugModeEnabled>
    <httpsEnabled>false</httpsEnabled>
    <httpEnabled>true</httpEnabled>
    <httpPort>8080</httpPort>
    <instanceName>Interpreting Environment Name</instanceName>
    <googleMapsAPI></googleMapsAPI>
    <jmxSupportModeEnabled>false</jmxSupportModeEnabled>
    <jmxPort>-1</jmxPort>
    <jmxUsername></jmxUsername>
    <jmxPassword></jmxPassword>
```

```
    <uuidModeEnabled>true</uuidModeEnabled>
    <uuid></uuid>
    <databaseName>dbtemp0</databaseName>
    <databaseNumber>0</databaseNumber>
    <csvGenerationEnabled>false</csvGenerationEnabled>
    <agentFrameworkEnabled>true</agentFrameworkEnabled>
    <agentDatabaseName>AgentDb</agentDatabaseName>
    <agentTypes>
    <agentType>
            <factoryClassName>AgentFactory#1</factoryClassName>
        </agentType>
      <agentType>
            <factoryClassName>AgentFactory#2</factoryClassName>
        </agentType>
    </agentTypes>
</agentConfigModel>
```

4.7 Utility Code

The design of the IE advocates the creation of utility code. Utility code refers to abstracted functionality that is not limited to a specific task, rather its potential for reuse in multiple components is maximised. The ongoing development of utility code while developing solutions to specific tasks is envisaged. This is intended to encourage the development of a standardised and maintainable code-base and discourage the implementation of idiosyncratic solutions that re-implement existing functionality.

There are a number of existing classes that that are examples of utility code. They assist with generic tasks related to the monitoring and analysis of data:

`epochGenerator`: Specifies an interval of TS data for which a sliding window is generated. It enables both the length (time period) and the increment between window instances to be defined.

`signalProcessingTask`: An abstract means of specifying a thread that should be processed. It is used by agents to encompass the work that should be carried out.

`SignalStorage`: An interface to the Data Store's fast file-based storage system. It links data sources to the data files on disk (see Section 5.1.3). It also specifies how the IE reacts to the creation, changing and deletion of such data.

`Annotation` simplifies the task of creating a list of signal annotations. Persistence of the resulting annotations is handled by the class. Other classes need only add annotations to the list once they have been identified and instantiated. Helper methods are provided to assist with the extension of previously created annotations.

`DerivedRecording` simplifies the task of creating recordings that are derivations of existing ones. The creation of a derived recording is handled by this class. Helper methods are provided to easily append sample values to the derived signals.

4.8 Distributed IE Architectures

The design of the IE, with its distinct loosely coupled components, makes it an ideal candidate for distributed implementation. Thus, the DP can be implemented at site 1, the IE at site 2 and the DC at site 3, where sites 1, 2 and 3 are geographically distributed as depicted in Figure 4.10. This would be seen to be the case, for example, in the Physiological Data Server described in Chapter 5. The distributed nature of these components is often mandated by the physical application being implemented and often poses additional engineering challenges including the provisioning of secure communication channels and the need to react in a timely manner in a real-time data-processing environment.

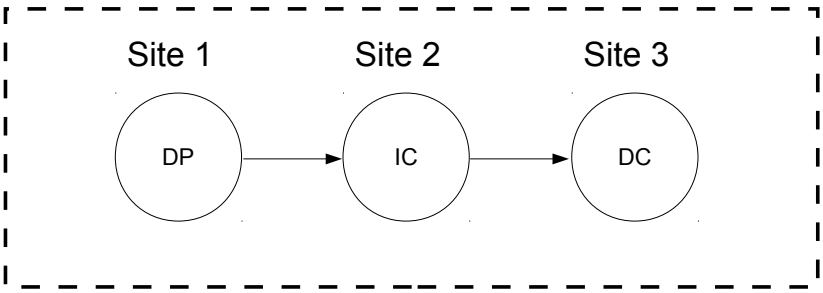

Figure 4.10: The IE enabling remote production, interpretation and consumption of data.

In addition to considering the distribution of a single component, it is natural to consider how multiple components of a particular type can be distributed for effective processing. For example, consider co-locating many ICs at a single site into which data is streamed from multiple distributed DPs and information is streamed to multiple DCS as depicted in Figure 4.11. This model effectively co-locates the IC, thus providing a centre of expertise servicing a distributed community.

Yet another architectural model can be realised by the creation of multiple distributed IEs as depicted in Figure 4.12 to produce a network of collaborating peers.

It will be shown in Chapter 5 how these distributed models map naturally onto specific application instances and how those mappings can be exploited to specific advantage.

4.9 Summary

This chapter described the implementation of an IE, introduced the IE Toolbox, a collection of abstract re-useable functionality for the development of application-specific IEs and presented a number of possible distributed IE architectures, each displaying interesting characteristics derived for specific groupings of functionality.

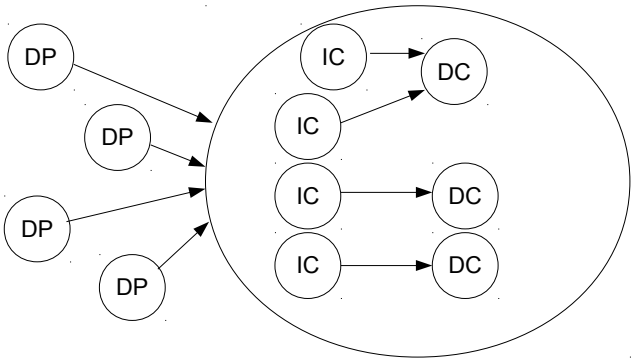

Figure 4.11: The IE as a distributed expertise centre.

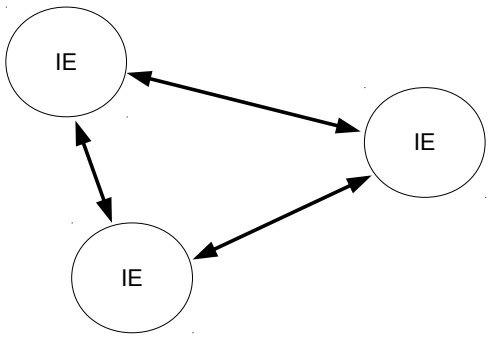

Figure 4.12: The IE facilitating collaboration between distributed experts.

Chapter 5

Realising A Physiological Data Server

The Physiological Data Server (PDS) was constructed based on the workflow description depicted in Figure 5.1[1]. This workflow description acts as a blueprint for the implementation of the application. Combined with the abstract methodology provided by the IE Toolbox, the complexity of the task was greatly reduced.

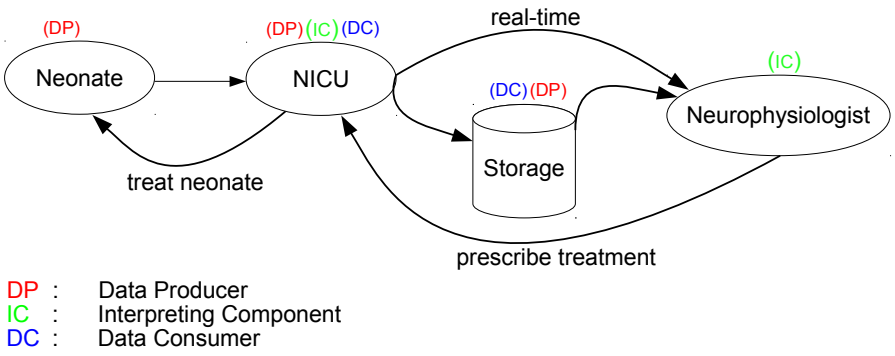

Figure 5.1: The workflow of the Physiological Data Server.

The neonate in the system adopts the roles of DP, producing data for the NICU and receiving treatment from the same. The NICU, in turn, adopts each of the roles in the IE, as it carries out an initial interpretation (IC), produces the data for the workflow's storage element (DP) and consumes interpretations of data output from the neurophysiologist (IC). The neurophysiologist adopts the role of an IC, analysing EEG data and providing an interpretation, in the form of a prescribed treatment for the neonate, is administered by the NICU. The dataflow incorporates feedback and so is continually driving the system towards

[1]The neurophysiologist (IC) is considered outside the scope of the NICU. The NICU (DC) consumes the neurophysiologists interpretation in the form of a diagnosis and acts to treat the neonate (DP)

a state of convergence, in this case the desire for a healthy patient state.

5.1 The Physiological Data Server

The PDS consists of four components: an upload application, a server, the Data Store and a viewing application, as depicted in Figure 5.2. The upload application is responsible for transferring data from the acquisition location to the server. The server provides the interfaces necessary for both human and software interaction with the data. The Data Store is a repository of all recorded data. The viewing application is used to analyse individual data records. Each of these components is discussed in more detail in the following sections.

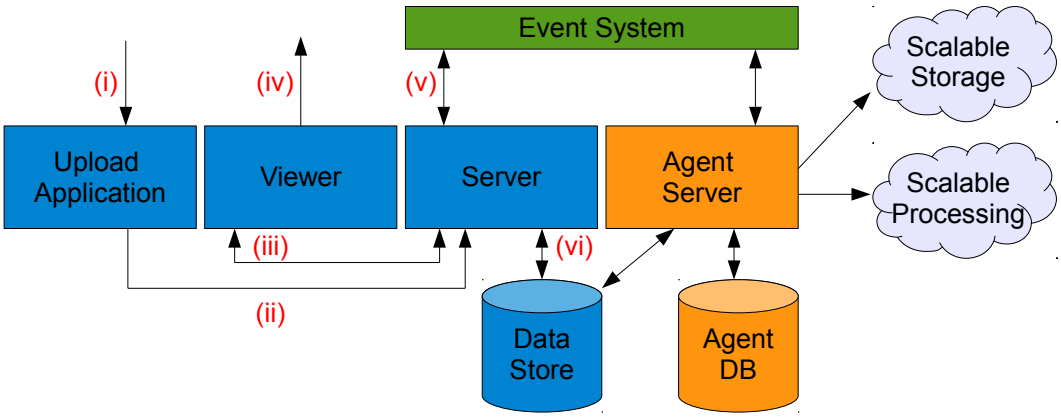

Figure 5.2: Architectural components specific to the Physiological Data Server.

5.1.1 The Upload Application

The upload application (UA) is responsible for the monitoring and acquisition of EEG data in real time. As such, the UA is acting in the role of the DP. The (i) in Figure 5.2 refers to the incoming flow of data to the UA, which is subsequently transferred to the server (ii). This results in a number of functional requirements:

i) The UA requires read access to the data being generated.

ii) The UA must be capable of identifying data that currently exists, detecting any modifications to this data and reacting accordingly when new data is generated. This necessitates the maintaining of a record of all previously acquired data.

iii) The UA must have a means of packaging and transferring data to the server.

iv) Finally, the UA must be capable of carrying out these tasks in near real time. The process is subject to a time constraint that complies with the operational procedures necessitated by the interpretation of data.

The UA's configuration is encapsulated by a Java object when running and persisted to disk as an XML file (`autouploadConfig.xml`) when deactivated. The format of the XML configuration file is shown in Listing 5.1.

Listing 5.1: XML configuration file for the UA

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AutoUploadConfigModel>
    <debugMode>false</debugMode>
    <serverIP>localhost</serverIP>
  <serverPort>8080</serverPort>
  <location>CUMH</location>
    <clientDataDirectory>C:\workspace\testdata</clientDataDirectory>
    <numConcurrentUploads>1</numConcurrentUploads>
    <uploadInterval>1</uploadInterval>
  <ignoreDirectories>
    <ignoreDirectory>backup</ignoreDirectory>
  </ignoreDirectories>
    <fileTypes>
    <fileType>
      <name>Nicolet EEG</name>
      <extension>e</extension>
    </fileType>
    <fileType>
      <name>European Data Format</name>
      <extension>edf</extension>
    </fileType>
    </fileTypes>
</AutoUploadConfigModel>
```

`serverIP` and `serverPort` tags provide the address of the server to which all data is forwarded.

`location` is the unique name of the data source.

`clientDataDirectory` denotes the top-level directory where data is stored.

`numConcurrentUploads` specifies the number of files that can be concurrently uploaded to the server. It has a default value of 1.

`uploadInterval` is the time interval that the UA waits before rechecking the `clientDataDirectory` for new data entries. It is specified in minutes and has a default value of 3 minutes.

`ignoredirectories` specifies the directories that are ignored when checking for newly created, or modified, data. The specified address is relative to the upload directory and directory names are case sensitive, *e.g.* "TEMP" is not the equivalent of "temp". Furthermore, specified directory names are valid for all subdirectories, not just at the top level, so any occurrence within the sub-directory tree structure will be ignored.

`filetypes` contain two nested elements that are used to specify both the name of a supported file type and its associated extension. If a file within the `clientDataDirectory` is not one of the supported file types, it will neither be detected by the UA nor transferred to the server.

Operation Of The UA

The operation of the UA is depicted in Figure 5.3. It is initiated by running a batch script (Microsoft Windows) or a shell script (Linux), depending on the underlying OS. Upon start-up the `clientDataDirectory` is queried to see if new data files have been generated or if existing data files are being extended. If so, this is detected and the new data is uploaded to the server. If not, the daemon for querying the `clientDataDirectory` sleeps for the specified `uploadInterval` before retrying.

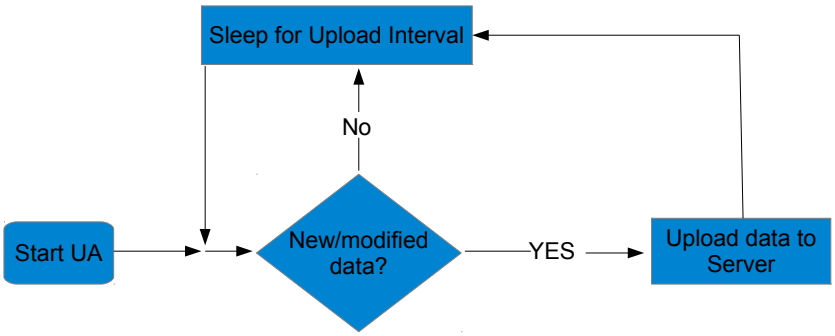

Figure 5.3: A flowchart depicting the operation of the Upload Application.

A record of all files that match supported file types, as specified in `autouploadConfig.xml`, are maintained by the UA. This facilitates the detection of new, and extended, files. Newly created files are detected by querying the set of existing file names, while an extension of a file is detected by comparing the file's name, access time and size to a previously recorded entry.

Transferring The Data

The UA uses HTTP as the communication protocol for transferring data to the server. There are a number of benefits associated with a HTTP-based transfer protocol. It is widely supported through the World Wide Web (WWW), can avail of secure connectivity and transfer of data through HTTPS (layering HTTP on top of the SSL/TLS protocol) and avoids issues with firewalls by operating on port 80, which is typically open in systems that have Internet access. Data is transferred to the server using a series of web requests with each `HTTP POST` request having a binary attachment that contains the data.

5.1.2 Server

The server is the intermediary between the components that comprise the PDS. As such, it it is fulfilling multiple roles, it acts as the primary DC and shares the role of IC with the Viewer. As depicted in Figure 5.2 it is responsible for the receipt and acknowledgment of data from the UA (ii), the persistence of that data to the Data Store (vi), the serving of data to the viewing application (iii) and the pushing of notices to interested parties via the event system (v).

The server enables experts to browse available data by providing a web interface from which they can select the data they are interested in (for more detail see Section 5.2). The server adopts the role of IC as it facilitates access to both the data streams and experts' access to those data streams. The server adopts the role of DC as it is responsible for acting upon the receipt of an expert's interpretation. The server provides a large array of functionality that enhances the work of the application, aiding the monitoring and analysis of data, and contains methodology that could be reincorporated into an abstract IE.

Acquiring Data

The server uses a RESTful architecture for communication. The arrival of data starts when a request, containing an XML description of the data, is received by the server from the UA. The server creates a corresponding database entry and responds with a data ID. This ID is then used as a parameter in the series of requests containing data samples uploaded by the UA. On receipt of one of these requests, the server saves the attached data to its internal Data Store (see Section 5.1.3) and updates the database entry to reflect the fact that it has been extended. This process continues until the UA is shut down manually, or a predefined time period has elapsed in which no changes are detected.

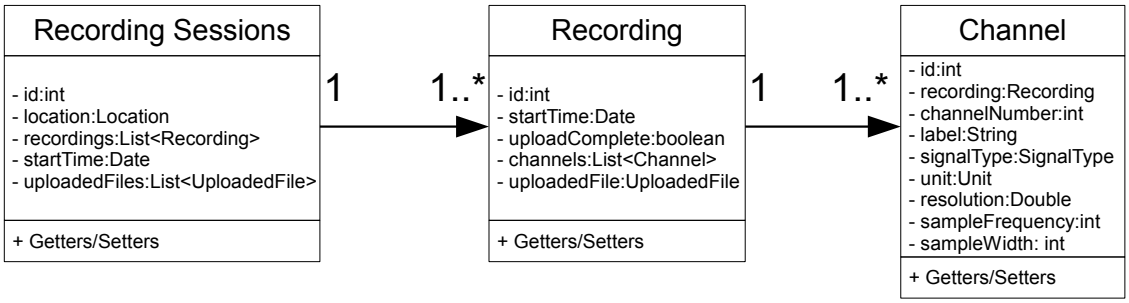

Figure 5.4: UML diagram of data structures.

5.1.3 Data Store

The Data Store is implemented using a combination of an embedded SQL database for storing meta-data and a collection of data files containing sample values. The database used by the server is implemented using HyperSQL [87], which is an open source relational database, implemented entirely in Java and available under the BSD License. HyperSQL was a fortuitous choice as it implements standards defined by Java, JDBC and SQL.

The use of standardised file formats such as EDF [88] and EBS [89] was considered when designing the Data Store, but none offered sufficient flexibility for planned future extensions to the environment's functionality. Instead a custom data structure was defined that enables an agnostic approach to the handling of TS data.

How Data Is Structured

The use case (see Section 1.4) influenced how data is organised and the data structure is depicted in Figure 5.4. While the data structure is specific to the application domain, as outlined by the use case, it is capable of handling TS data in a generic manner and so would be of benefit to other application domains that use TS data. Its main elements include:

- Channel: An individual series of TS values.
- Recordings: A collection of one or many channels of TS data from a DP.
- RecordingSessions: A top-level container encompassing multiple recordings from a single DP.

Meta-data stored in the database includes recording attributes such as the acquisition location, commencement time, and the properties of the individual channels, *e.g.* the frequency at which a channel was recorded. A data file containing sample values is maintained for each channel of each recording. Data files are used to avoid performance overheads incurred by storing individual sample values in the database, and allow arbitrary intervals of recording data to be retrieved efficiently.

5.2 The Viewer

The viewer facilitates the neurophysiologist's interpretation process and so adopts the role of an IC. It captures that interpretation by enabling the neurophysiologist to provide feedback in the form of annotations as described in Section 3.2.4.

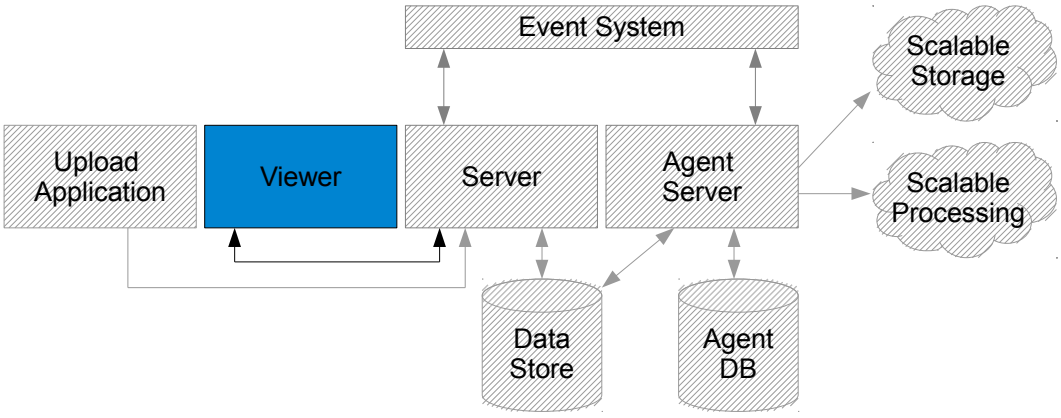

Figure 5.5: The components comprising the Viewer.

Although most TS data can be displayed effectively using a simple plot, the display of EEG tends to be more complex. The raw data is of limited use and typically a montage is applied in order to assist interpretation. A montage refers to the placement of electrodes on the human head. There are two types of montage used for clinical analysis, a referential montage, whereby there is a single reference electrode for all electrodes, or a bipolar montage where you have two electrodes per channel, thereby having a reference electrode for each channel.

The 10-20 system or International 10-20 system is a recognised standard for the placement of electrodes on the scalp for recording EEG data, as depicted in Figure 5.8. The 10-20 system is a standardised approach to the recording of EEG data; this standardisation enables the comparison of data acquired from different studies.

Experienced neurophysiologists may switch between several different montages when

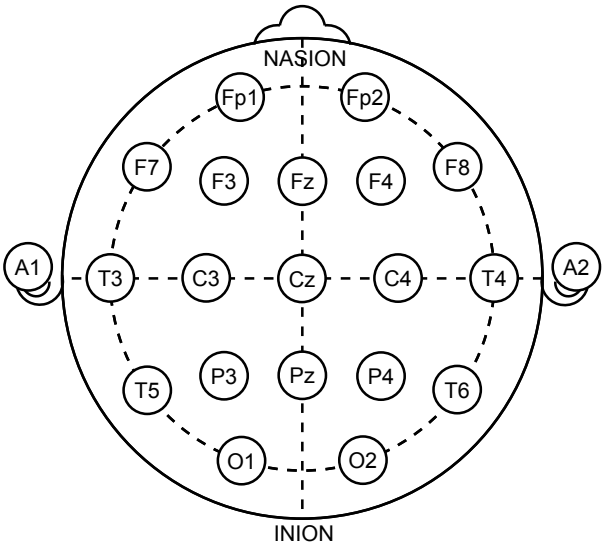

Figure 5.6: The electrode placements as set out by the International 10–20 system [2].

analysing a recording. Two viewing modes are typically used: a review mode that switches instantaneously between one screenful of data at the user's discretion, and a "playback" mode that simulates the acquisition process. Signal filters are essential to eliminate noise and highlight features of interest. Other features expected by experienced analysts include sensitivity adjustment and the rendering of signals in various colours determined by electrode location.

A plug-in technology capable of more advanced client-side applications was therefore required to implement the viewing application. During the evaluation process three such technologies were in widespread use, any of which would have been suitable: Flash, Java and Silverlight [90]. Flash was chosen for its ubiquity (available on 96% of browsers compared to 81% for Java and 35% for Silverlight[2]) and excellent graphics support. The use of a plug-in technology allows the application to be hosted on the platform and downloaded on demand, facilitating seamless software updates and eliminating the need for users to install software on their local machines.

Once the upload of an EEG recording to PDS has commenced, the recording may be viewed by users with access to the system. This process begins with the user selecting the recording to be viewed in the web application. This causes the viewing application to open in a popup window. Once the viewing application has finished loading, it commences operation by downloading the details of the recording being viewed, such as the acquisition location, commencement time and channel attributes. The signal browsing interface is then presented to the user (see Figure 5.8).

Next, the application begins to download signal data from the system. By default, the

[2]Statistics from statowl.com, Jul-Dec 2009.

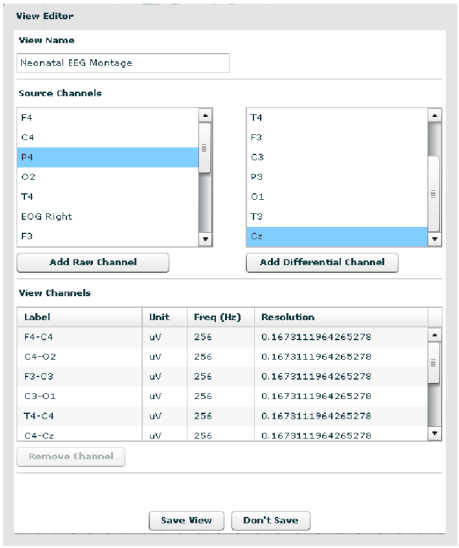

Figure 5.7: Editing a montage in the Viewer.

application will attempt to load all of the signal data in sequence in the background. A progress bar at the bottom of the screen indicates the current view position and the amount of data segments that have been loaded. For usability purposes, data is downloaded on demand, i.e. the recording segment currently in view receives priority. So, if the user moves the viewport to a recording segment that has not yet been downloaded, then the current request is interrupted and a new request is issued starting from the updated viewport location. If acquisition is ongoing, then new data will periodically arrive on the server. The viewing application detects the presence of new data by polling the server. If a change in the recording duration is detected, then the view is updated and the user is notified.

By default, a bipolar montage is applied to the electrode data. A menu is provided that allows the user to edit the default montage or create a new montage. As illustrated in Figure 5.7, side-by-side lists are used to select channel combinations, for instance, here the channel P4-Cz is about to be added. Signal rendering colours are applied automatically based on electrode placement, but the colours are also user-configurable. Three filters are available: a 0.5Hz high-pass filter to remove slow artefact, a 30Hz low-pass filter to remove high-frequency artefact and a 50Hz notch filter to remove artefacts caused by electrical power lines. Only the notch filter is enabled on all channels by default, but each can be enabled or disabled on a per-channel basis. An option is provided to disable the rendering of selected channels – a useful feature when artefacts caused by loose electrodes obscure the view of other channels.

Two viewing modes are provided: browsing and playback. In browsing mode, the user flips instantaneously forwards or backwards through screenfuls of data using buttons or keyboard shortcuts. The duration of the recording segment visible on screen can be adjusted

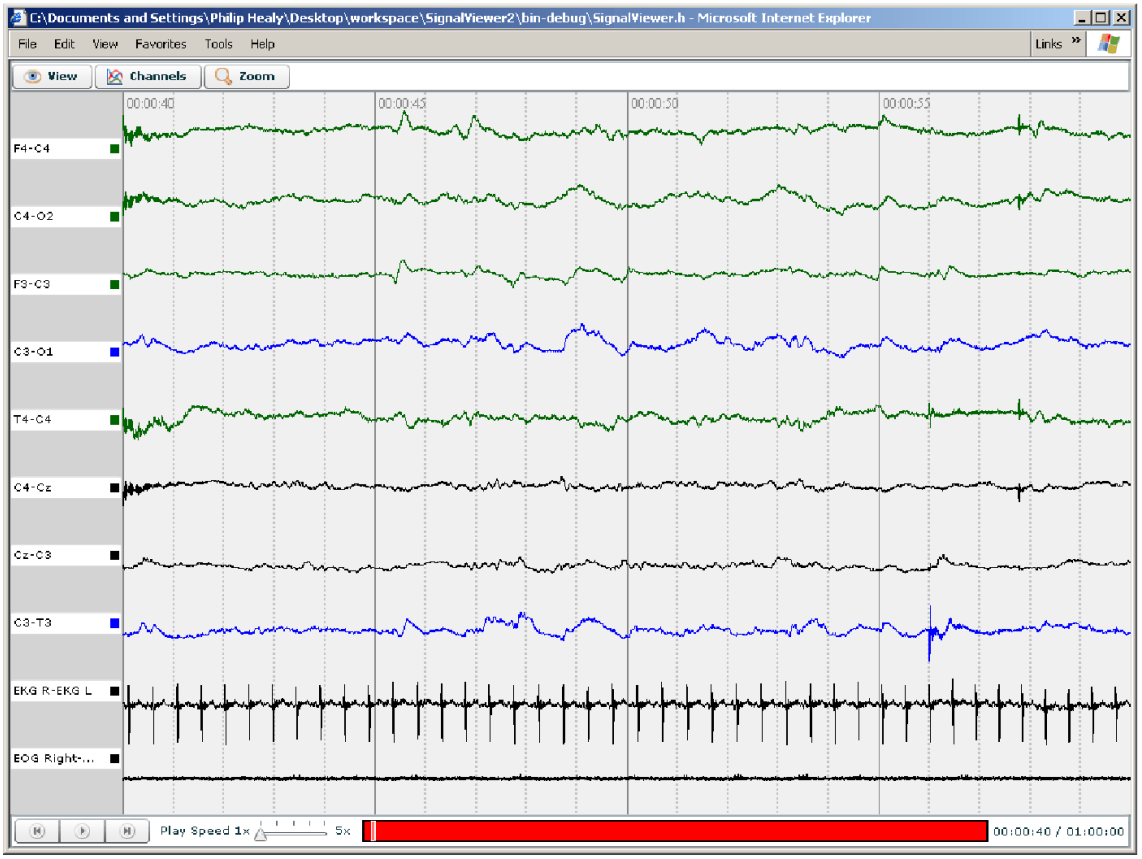

Figure 5.8: The Viewer displaying 8 channels of EEG data in a bipolar montage along with one channel each of EKG and EOG.

using a drop-down menu. An arbitrary recording offset can be selected by hovering the mouse over the bar at the bottom of the screen. When the mouse pointer is over the bar, a visual indication of the exact offset to be selected appears and changes dynamically as the pointer is moved left or right. Clicking the mouse moves the view to the selected offset. Playback mode is activated by clicking on the play/pause button. In playback mode, the next screenful of data is drawn gradually from left to right over the existing data on-screen. The update speed can be adjusted using a slider control, allowing data to be reviewed at a speed that suits the user. When the end of the recording is reached, new data will be displayed as soon as it arrives from the server, allowing the most up-to-date data to be continuously monitored.

Viewer Technologies

The use of REST and XML results in a solution that is free of proprietary technologies such as BlazeDS (server-based Java remote and web messaging technology that communicates with back-end and front-end Rich Internet Applications (RIA) developed for Adobe Flash (Adobe Flex or AIR)). Communication with the viewer is carried out via HTTP, REST and XML, thus, preventing vendor lock-in (restricted future development due to dependencies on a proprietary technology stack). For example, the development of a HTML5-based viewer for Android devices is a possibility and would readily integrate with the back-end of PDS.

5.3 Software-specific Additions

Application-specific requirements

While the IE is capable of facilitating a broad range of requirements, specific application domains often necessitate the fulfilment of application-specific functionality. In the case of the PDS, the use case identified a number of these requirements. A requirements document (see Appendix A) was formed based on consultations with neurophysiologists. This requirements document was also influenced by the use case description in Section 1.4 and a review of the related work (see Section 1.2).

5.3.1 Adobe Flash

Experts required a means to visualise data remotely. This resulted in the development of a browser-based application. The extensive level of custom viewing options required by experts for the interpretation of data required a sophisticated technology for its realisation (see Section 5.2). Adobe Flash [91] is a means to develop Rich Internet Applications

(applications capable of delivering functionality traditionally associated with a desktop application) with an excellent user experience to meet the requirements of the viewer for the IE. Furthermore, when the viewer was initially developed HTML5 was at an early stage and Adobe Flash was the dominant browser graphics rendering environment, being available on 96% of commonly installed web browsers [92].

5.4 Agent Framework Operation

Upon start-up, the agent manager reads its configuration file. An instance of each enabled agent factory is instantiated. Next, the agent manager determines if any agents have tasks outstanding that were prevented from executing by a fault such as an unexpected server shutdown. If so, the outstanding tasks are recreated and submitted to the task executor.

Once the upload of a new recording commences, an event is dispatched that is detected by the agent manager. The agent manager then informs the set of available agent factories, who instantiate the set of agents applicable to the recording. The resulting agents are then persisted to the database. Each agent queries the Data Store, in order to update the quantity of data available and to check if sufficient data is available to begin/resume processing. If so, the agent returns an executable task, which is submitted to the task executor.

Data is streamed from the acquisition location in discrete segments. The arrival of each segment triggers an event indicating that the data stream in question has been extended. The agent manager responds to these events by reconstructing the set of agents associated with that data stream using the persistence API, querying each agent for a task to be executed and scheduling the resulting tasks to the task executor. Although the set of agents associated with a datastream is frequently reconstructed, the caching facility provided minimises the resulting performance impact.

The completion of each task results in an update to the internal state of the agent. These changes are then reflected in the database through the persistence API. If the execution of an agent task has not been completed by the time a new segment of recording data has arrived, the agent is not reactivated until such time as the outstanding task has completed execution and the updated state of the agent has been written to the database (where the agent output is sequential and not parallel). This ensures that all tasks execute with an up-to-date representation of the agent's state.

In the event that an agent task results in an error, the agent manager flags the agent in question and does not solicit any new tasks from it. Information about the error (such as the amount of data that was successfully processed before the error occurred and the Java exception that was caught) is stored and made accessible via the web interface. This

error-handling scheme cannot detect situations where an agent task hangs indefinitely. The only means of preventing this currently are through testing and the code review process (see Section 5.5).

PDS is notified by the UA once signal acquisition has ceased and all outstanding data segments have been uploaded. This triggers an event within the IE, to which the agent manager responds by reconstructing the affected agents and notifying them in turn. This allows agents to perform any final processing that may be required.

5.5 Security

Security concerns for data protection are an important consideration when patient data is concerned. Steps have been taken to address the security concerns of tele-health systems, as outlined by ISO 27001 [93] and ISO 27799 [94] with regard to data sensitivity, data transfer and code review, in so far as feasible.

Data Sensitivity

PDS contains interfaces that interact directly with sensitive information. Care must be taken to properly manage such information. PDS addresses this in a number of ways. In order to preserve confidentiality, unnecessary data that could identify an individual is not acquired. For example, when being used in the NICU, the upload application does not read personal patient information (name, age, sex) from data files and the server has no means of storing it.

Data Transfer

Consideration was also given to the security of data transfer. To reduce the likelihood of malicious attacks and to prevent eavesdropping on data being transferred, the ability to enable HTTPS and communicate over an SSL connection is supported.

Code Review

Two possible solutions to the issues of security and data integrity considered are sandboxing and code review/JAR-signing. The sandboxing solution executes agents in a restricted environment using a dedicated Java security manager [95] to mediate access to system resources. The JAR-signing solution involves the use of a Java class loader that will refuse to load JARs that have not been signed using a key known only to the framework developers [96]. In order to have an agent JAR signed, the source code implementing the agent

package undergoes code review to determine if there any issues with security or data integrity. On reflection, it was decided that the JAR-signing solution is the most appropriate, as the use of a security manager imposes a performance overhead of 5-100% per resource access statement [97]. The JAR-signing restriction is only enabled on production builds of the remote monitoring server, in order to facilitate the development and testing of agent implementations, before they are submitted for review.

5.6 The Testbed

A testbed was deployed to the NICU in the Cork University Maternity Hospital in association with the Neonatal Brain Research Group [98] (NBRG) from October 2009 to March 2012. This enabled the operation of PDS to be tested in a real clinical environment on an ongoing basis. A process of requirement-gathering was undertaken with personnel working in the NBRG (the details of which are outlined in Appendix A). This was beneficial, as it assisted with the identification of issues affecting the design of PDS, such as infrastructural constraints and the technologies employed.

Many of the medical devices found in the NICU, such as heart rate monitors, typically contain a serial port that allows the data stream to be extracted easily during acquisition. However, many of the EEG machines currently in use do not have such a facility. Even if the data stream were to be extracted directly from the amplifier, this data is of little use without knowledge of the electrode placements and other settings that are typically configured in the acquisition software, and which enable interpretation.

Access to this information from within the acquisition software is possible if an appropriate plug-in API is supplied by the vendor of the acquisition software. A drawback of this approach is potential disruption of the acquisition process if bugs arise due to the interaction between the plug-in and acquisition software.

5.6.1 Initial Approach

Initially, the system's data flow contained five entities. These were the patient, the EEG monitoring machine, the UA, the server and the neurophysiologist, as depicted in Figure 5.9. The intention was to install the UA directly on the EEG monitoring machine. This would provide the UA with access to the data as it was being generated by the data source, enabling the upload application to run concurrently on the EEG monitoring machine and stream the EEG data to the server directly.

The acquisition software running on the EEG monitoring machine was proprietary and all data generated by the acquisition software was stored in a proprietary format. The software was capable of generating data in a non-proprietary format, the European Data

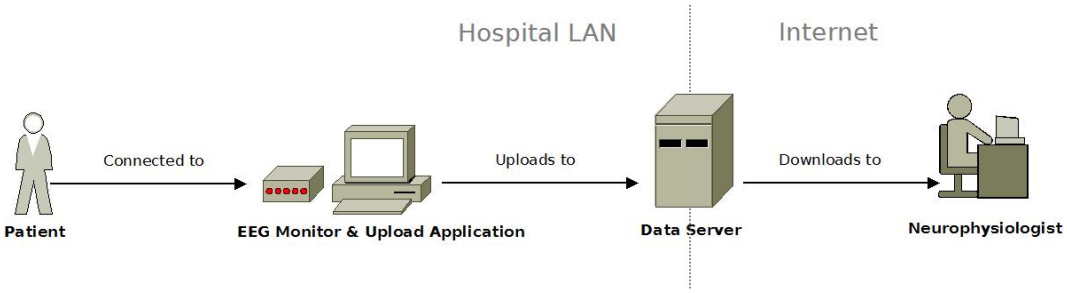

Figure 5.9: The original data flow envisaged for the system.

Format (EDF) [99]. However, EDF did not support the association of contextual information (in the form of text-based labels) with data streams and so, was largely unused by the clinicians.

A suite of dynamic link libraries (DLLs) were provided by the software vendor, enabling direct interaction with the acquisition software, and enabling the reading and writing of data in the proprietary format. A prototype of the UA was developed that transferred data as it was being acquired by the proprietary software. This demonstrated that the prototype could successfully acquire data in a simulated environment.

Upon deployment of the prototype to the EEG monitoring machine, the UA failed to acquire EEG data. The was due to the DLLs provided by the vendor of the acquisition software. The DLLs provided were a debug version. Registered in isolation, these debug-DLLs enabled the UA to acquire data. Registered in conjunction with the production-DLLs, they rendered the UA inoperable. As the production version of these DLLs was already registered, installing the UA would result in the prevention of critical operations being carried out in the NICU.

5.6.2 Consequence Of An Integrated Approach

As a result not only was the UA prevented from reading and transferring data to the PDS, but the EEG monitoring machine was prevented from acquiring data from patients, rendering it inoperable. It was decided that this method could only be made trustworthy through extensive testing in partnership with individual vendors of the acquisition software. Concerns relating to the validity of the medical device's CE mark [100] as a result of installing third-party software were also raised. As such, an alternative acquisition process to mitigate these issues was required.

5.6.3 A Revised Approach

A more "hands off" and non-invasive approach is to extract the required data from the EEG data file output by the acquisition software. An advantage of this method is that it can

be performed on a file server, eliminating the need to install any software on the machine used for acquisition, thereby ensuring the integrity of the machine and its CE mark. A limitation of this method is that the acquisition software must be capable of saving to a file server. Rather than saving the EEG data to the local machine, the acquisition software saves the data to a shared network drive on the intranet. This strategy provides a sandbox-like environment, isolating the production environment critical to the everyday operation of the NICU from untested code and configuration changes. However, this does not solve the issue of acquiring the data as it is being generated, a necessity for the analysis of data with temporal constraints.

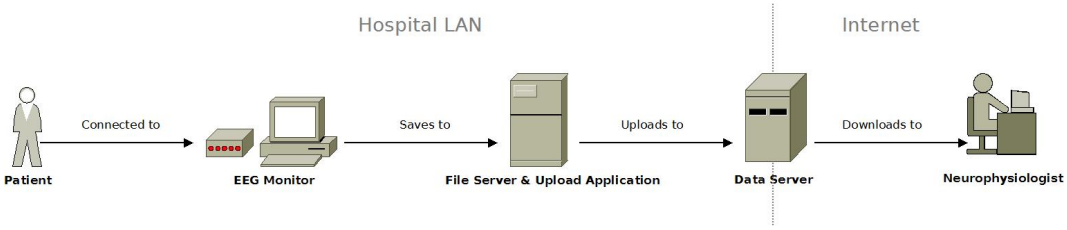

Figure 5.10: The revised data flow envisaged for the system.

Figure 5.10 illustrates the revised flow of data through the system. The UA, running on the file server, streams data from the EEG data file to the server. From there, it is streamed to the viewing application running on the neurophysiologist's PC within a browser. The file server is not accessible outside the hospital LAN because it contains sensitive patient information.

5.7 Streaming Data In Real Time

Streaming data necessitates the ability to access data in real time. This was a major challenge for PDS to address, as data generation was carried out by third-party software. As data generation was outside the scope of PDS's DPs, it required a general purpose solution for the accessing and streaming of data in real time.

This can require invasive measures, such as accessing a file while it is being modified by another process. This can be difficult from a software engineering perspective, especially when used in conjunction with critical infrastructure and processes. PDS provides a novel solution to overcoming this challenge, that enables the acquisition of data, while maintaining the integrity of the data and does so in a non-invasive manner.

The Challenge Of Real-Time Acquisition

The default behaviour of some OSs, such as Microsoft Windows, is to disallow concurrent access to a file by more than one process. If the OS enforces exclusive file access then the process that attempts to read from the shared file will be prevented from doing so and must wait indefinitely until the writing process closes the file. If the writing process opens and closes the file periodically then the reading process may access the file after the writing process closes it. However, the reading process may in turn lock the file, preventing the writing process from reopening it and potentially leading to undesirable behaviour. Although a processs locking behaviour can be changed programmatically, this approach requires access to the processs source code, and is not applicable as a general purpose solution. A similar situation arises when file access is being performed using a library for which the source is not available.

At present, the only means to implement concurrent access to locked files is to either (a) modify one or both processes to implement less restrictive locking behaviour or (b) run the processes on an OS that does not enforce exclusive locks using one of the techniques described above. If neither of these solutions is applicable, as in the context of the use case, then the only means available is to provide concurrent access to files in the presence of exclusive locks. This applies generally to any situation where file-locking behaviour prevents concurrent access to a file.

The Solution

Consider two processes: the first, the writer, is modifying a data file that runs continuously; the second, the reader, is invoked repeatedly, reading and processing any new data in the file upon invocation. Concurrent access to the shared file is prevented by the OS's file-locking mechanism. The PDS's solution is to create a third process, the monitor, that detects changes to the shared file using attributes, such as file length or access time, that can be determined without locking the file. When a change to the shared file is detected, the monitor performs a block-level copy of the shared file, bypassing the OSs file-locking functionality.

The reader then works on the newly created copy. This process is repeated until new data is no longer detected. This technique insulates the reader from the writer, preventing the locking behaviour of one from interfering with the operation of the other. The solution described here does not require any modifications to the applications/libraries themselves. However, the technique does involve the creation of copies of the file in potentially inconsistent states. The reading process must be robust enough to tolerate this.

This robustness is facilitated by the Volume Shadow Copy Service [101] (VSCS). The

VSCS is integrated into multiple versions of the Microsoft Windows OS. PDS re-purposes this service to enable real-time acquisition of data from a data source in a reliable and maintainable manner. VSCS enables a snapshot of data on disk to be taken and is typically used for backing up systems by making copies, or snapshots, of the files on disk. There are two important features of the VSCS: firstly, it ensures that data does not change while the backup is taking place, and secondly, it avoids the aforementioned problems with file locking. PDS uses Hobocopy [102] to incrementally invoke Shadow Copy on the data source, by creating a clone in a reliable manner that is compliant with Microsoft Windows operating procedures. These incremental copies are then read by the UA and any new data detected is uploaded to the Server as depicted in 5.11.

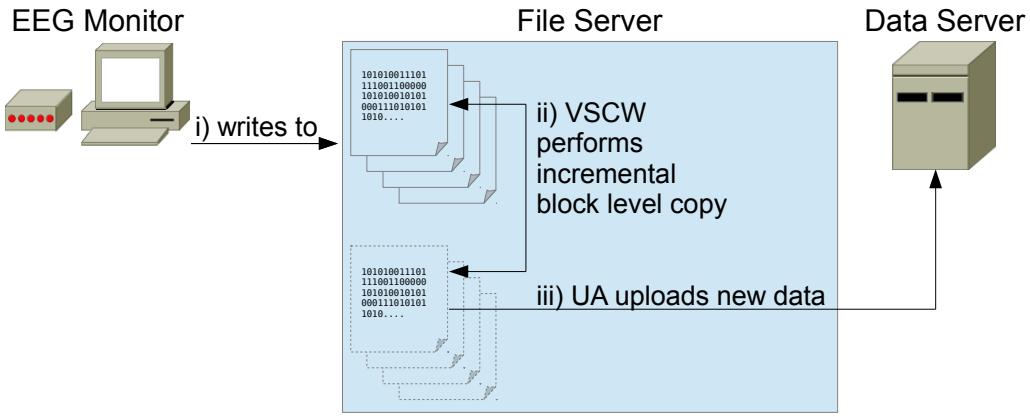

Figure 5.11: The process of streaming data within PDS.

An Alternative Approach

An alternative approach could be the use of Shim infrastructure [103], a technical solution for enabling application compatibility on Microsoft Windows. Shim infrastructure enables developers to hook into function calls made by particular versions of a running application. By linking API calls to alternative code it would be possible to access a data source as it was being generated by software running on a Microsoft Windows OS. In this manner, data could be accessed without interfering with the acquisition process and a copy of the data provided for streaming purposes. However, the scope of this solution is severely limited as it would require a specific instance of the solution for each software application generating the data, as well as an understanding of how that application interacted with the Microsoft Windows OS.

Deployment To The NICU

The real-time monitoring solution is directly integrated in to the UA. An instance of the UA is generated and deployed to a file server in the NICU. The UA itself is a collection of JAR files that encompass the functionality for acquiring, packaging and transferring the data to a server, the DLLs that enable the reading and writing of files in the proprietary format, and some third-party software for invoking snapshots. Also included is an XML configuration file for setting the relevant details and batch/shell scripts for running the application with the respective OS. The revised approach is depicted in Figure 5.10

The UA streams individual data files that are associated with a patient. This recording consists of multiple channels of EEG data. Each channel of the recording refers to a particular location on the human head where an electrode has been placed. A patient may have multiple recordings, *e.g.*, the recording goes on for a prolonged period of time, or different EEG monitoring machines are used to acquire the data. The recordings are then grouped into a recording session. This is how it is displayed in the viewer, as a series of recordings under one recording session. Also the UA can support multiple uploads to the server simultaneously.

5.8 Summary

This chapter used the concepts developed in Chapter 2, Chapter 3 and Chapter 4 to create the PDS. A workflow description, representing a blueprint for the specific application domain, was used to aid the software construction process. This process made use of the generic functionality offered by IE Toolbox.

In addition, it was necessary, due to specific application requirements, to create additional software components. These components predominantly relate to managing proprietary formats and interfacing with third-party libraries. As such, the functionality required could not be anticipated and provided for in a generic manner by the IE. It is likely that all applications of the kind described here will require some degree of customisation. However, it can also be seen that a large portion of the application code can be realised from the generic reusable components.

The fact that a TS application as complicated as the PDS could be rapidly constructed using mostly standardised components together with a transparent workflow description, provides strong evidence of the utility of the IE proposed here. The utility and fitness for purpose of the resulting application is explored further in Chapter 6.

Chapter 6

Evaluating The Physiological Data Server

In this chapter the Physiological Data Server's (PDS) capacity to produce, process and consume data is demonstrated. Throughout this demonstration an emphasis is placed on how PDS frees itself from the constraints of any particular IC, while at the same time facilitates the inclusion of specific analysis techniques that a particular domain may require.

6.1 Experimental Testbed

The hardware used for experimental work consisted of a Dell PowerEdge R720 Server with a mounted NFS datastore, which was directly connected through a 1Gbit/s network connection. Hardware specifications are detailed in Table 6.1. The server ran a 64-bit version of CentOS 6.6. Experiments were executed on virtual machines, with varying specifications, using the hypervisor KVM.

Manufacturer	Dell
Model	PowerEdge R720
CPU model	Intel Xeon CPU E5-2640 v2
Cores	8
Sockets	2
Total physical cores	16
Clockspeed	2.0 GHz
Memory	64GB
HDD	1TB over NFS
Hyperthreading	yes

Table 6.1: Hardware specifications of experimental testbed.

The virtual machines used, operated using a 64-bit version of Microsoft Windows 7

Professional SP1. Hardware configurations of VMs varied between experiments as shown in Table 6.2. The CPU configuration was set to host, to pass through all available CPU instructions. Due to the limitations of Windows 7, the socket count was set to 1. Otherwise the system would simulate a socket for each core, which does not work with Windows 7 Professional. The storage driver was set to virtIO with the storage format qcow2. This allowed the use of thin provisioning, which only grows the image file when the space is in use.

Size	Standard	Large
Cores	4	16
Sockets	1	1
Memory	4GB	16GB
Used in exp. work for	Sections 6.2, 6.3, 6.4.2, and 6.5.1	Section 6.7

Table 6.2: VM specifications used for carrying out experimental work.

Additional software installed on the VMs included the Java 7 runtime (version 1.7.0.71). The browser used to interact with the web interface of PDS was Google Chrome (version 41.0.2272.118m). Python scripts were used to automate the execution of the experiments and so Python (version 2.7.9) was also installed.

6.2 Generating Data for Experimental Purposes

To carry out a number of the experiments it was necessary to generate data on demand. This necessitated the creation of an arbitrary number of DPs in order to produce an arbitrary quantity of data on demand. This also necessitated the generated of data in real time. This generation must occur in a manner that periodically writes the data to a file such that the file-locking behaviour, as discussed in Section 5.7, occurs. Furthermore, the ability to generate multiple data feeds concurrently must be provided.

Data Generation

A custom instance of PDS was deployed to facilitate data generation. An upload application (UA) transmits a collection of pre-recorded EEG data files to this instance, in 60-second chunks, with a delay of 59,985 ms between each transmission. The delay period was selected based on empirical testing to identify which value most closely mimicked real-time generation. A DP was developed to assist with the process, the `EDFExportAgent` (for more information see Section 6.4.1).

Figure 6.1 depicts a 6-min segment of EEG data being uploaded by a UA at the throttled rate. Note how the data available on the custom instance (blue) and the data exported by

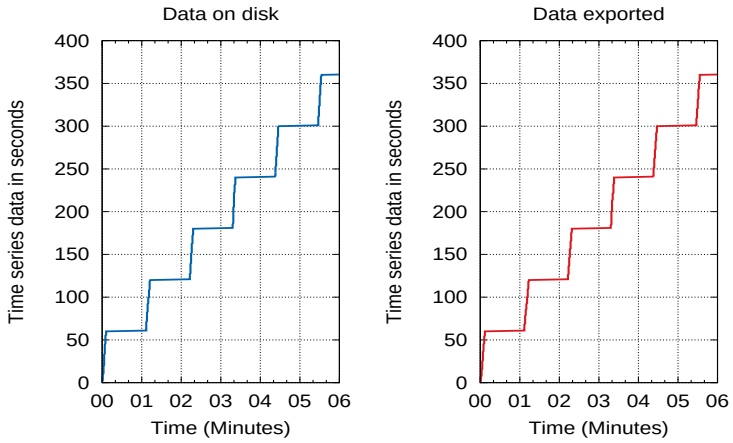

Figure 6.1: EEG data being uploaded to PDS.

the `EDFExportAgent` (red) are identical. This illustrates the PDS's capacity to export data without a degradation of service in the order of seconds. Figure 6.2 demonstrates the generation of data, over a longer period of time with no noticeable degradation in performance.

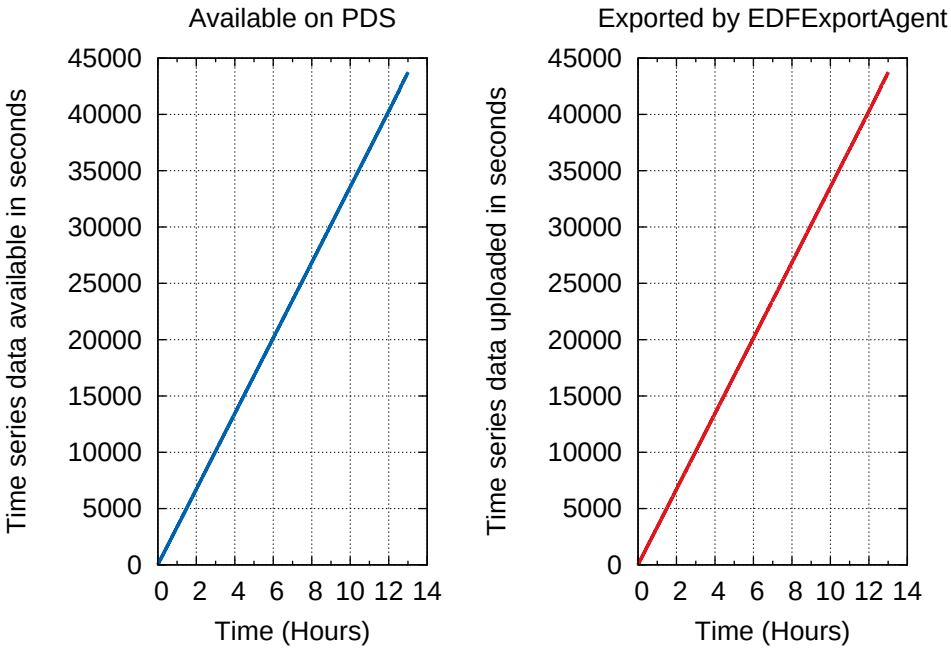

Figure 6.2: Approximately 12 hours of data, uploaded to PDS and exported by the `EDFExportAgent` in real time.

6.3 Streaming Data

The ability to stream data in real time is critical for timely analysis. A delay in data acquisition can have critical consequences, as in the case of neurophysiological monitoring (see Section 1.4). As such, real-time acquisition of data is seen as an essential requirement for a generic monitoring and analysis platform.

Experimental Procedure

To evaluate PDS's capacity to support real-time acquisition, a one hour EEG data file was acquired in a real-time setting. Each stage of the dataflow was monitored and associated metrics generated. UA1 and UA2 are both instances of an upload application (as described in Section 5.1.1). UA1 uploads one hour of EEG data to an instance of PDS. The PDS instance exports the data to disk using the `EDFExportAgent`. This recreates the file-locking behaviour that affects the acquisition of real-time data in the NICU (as described in Section 5.7). UA2 acquires and uploads the data in real time using PDS's non-invasive means of acquiring streaming data (as described in Section 5.7). There is a programmed delay of two minutes between the execution of UA1 and UA2 to ensure that the data has been uploaded and exported before acquisition is attempted by UA2.

Results

The data was acquired at a rate that exceeded real-time data generation and experienced no noticeable delay as can be seen from Figure 6.3.

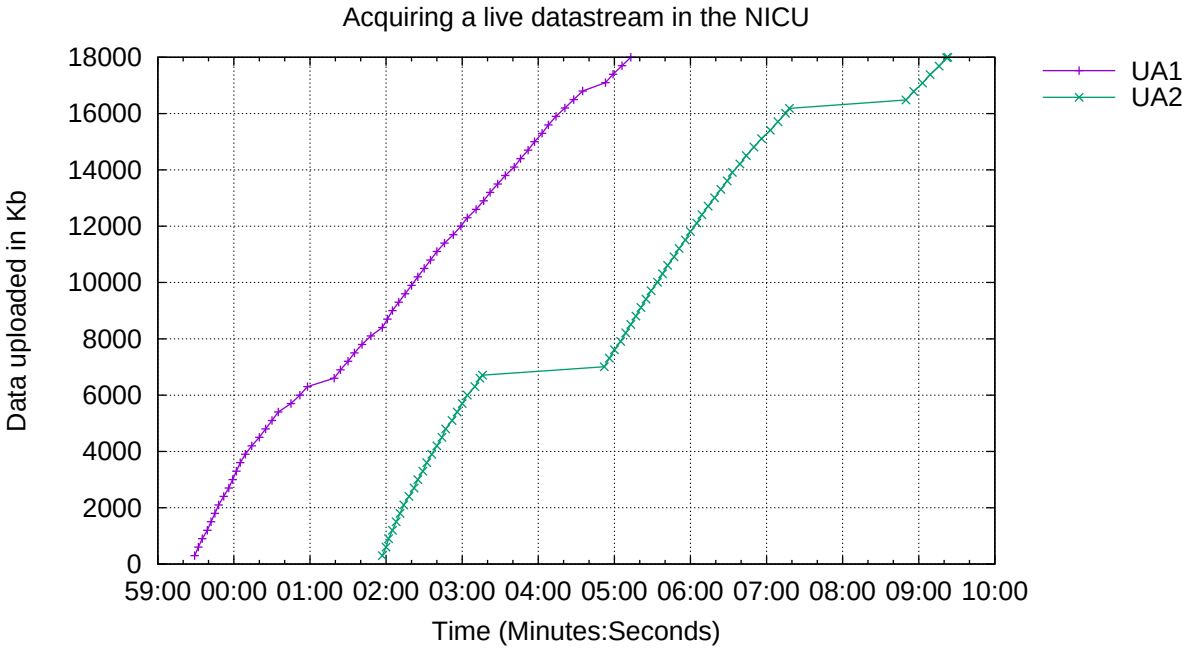

Figure 6.3: Real-time acquisition of one hour of data.

Discussion

Figure 6.3 demonstrates PDS's capacity to acquire data in real time and support the timely analysis of data. UA1 uploads an hour of data in approximately 6 minutes, this data is exported to disk as it arrives. UA2 periodically invokes a low-level copy of the data on disk and proceeds to upload it to a second instance of PDS. UA2's ability to access "locked data" without interfering with UA1's writing process is indicative of PDS's ability to acquire data in real time in a non-invasive manner.

6.4 Mapping Roles Via The Agent Framework

Individual agents are capable of adopting the roles associated with the workflow components and can embody functionality that enhances PDS. This provides a means of specifying extensible functionality tailored to the specific domain being analysed. To demonstrate this, a number of agents that assist with the production, processing and consumption of data have been implemented. The first acts in the role of a DP and assists with generation

of data (see Section 6.4.1), the second acts in the role of both a DP and an IC by broadening the number of experts (and processes) that can analyse the EEG data (see Section 6.4.2), and the third acts in the role of both an IC and DC, assisting experts with their workload by processing data streams as they are received (see Section 6.5).

6.4.1 The `EDFExportAgent` (DP)

The `EDFExportAgent` was developed to assist with simulating data generation. It demonstrates PDS's capacity for extension via the Agent Framework. As data is received by PDS an EntityCreated and an EntityChanged event are thrown. The `EDFExportAgent` is listening for both, and so, is notified when a recording is either created or extended. The `EDFExportAgent` reads this data from the Data Store and proceeds to write this data to a file on disk, extending it in 1-second chunks, in the EDF file format.

The manner in which data is written results in the periodic locking of the file, as data is written to disk as it arrives. This closely resembles the acquisition process as it occurs in the NICU. For example, the Nicolet One Monitor [104] was found to buffer approximately 8-12 seconds of data before writing to disk. In doing so, the `EDFExportAgent` recreates the file-locking behaviour experienced in the NICU.

This provides the ability to evaluate PDS at a user-defined rate of data generation. This is something the testbed, discussed in Section 5.6, could not have facilitated.

6.4.2 The `aEEGAgent` (IC)

Amplitude-integrated electroencephalogram (aEEG) is a derived signal from a reduced EEG and is used as a method for the continuous monitoring of a patient's brain. The method is based on filtered and compressed EEG that enables evaluation of long-term changes and trends in electrocortical background activity by relatively simple pattern recognition. It is an extension of the cerebral function monitor which was developed in the late 1960s to monitor adults in the intensive care unit [105].

In term infants (infants born at 37 to 42 weeks gestation) aEEG is an excellent method for evaluating cerebral function and cerebral recovery after hypoxic-ischemic insults such as perinatal asphyxia and apparent life-threatening events [105]. aEEG background activity within the first six hours of birth in term infants afflicted by hypoxic ischaemic encephalopathy has been shown to be predictive of later neurological outcome [64].

PDS's application to neurophysiological monitoring resulted in an agent for the derivation of aEEG being considered an appropriate analysis technique. As an IC, it enables the interpretation of raw EEG data, resulting in a derivation of new information and the production of new data, in effect the `aEEGAgent` is simultaneously carrying out the role of

a DP and a IC. As such, the `aEEGAgent` was developed and incorporated into the system. The aEEG approximation implemented by the agent uses a 9^{th} order digital infinite impulse response (IIR) least P-norm bandpass filter designed to pass energy in frequencies from 2Hz to 15Hz. The nonlinear mapping function is defined as follows:

$$F(\text{eeg}(t)) = \begin{cases} |\text{eeg}(t)| & |\text{eeg}(t)| \leq t_h \\ t_h \log\left(|\text{eeg}(t)| - [\log(t_h) - 1]\right) & |\text{eeg}(t)| > t_h \end{cases} \tag{6.1}$$

where log denotes the natural logarithm and the threshold, t_h, is chosen as 20μV. The nonlinear mapping is then averaged across 3s and scaled:

$$\text{aEEG}(t) = \frac{1}{3}\int_{t-1.5}^{t+1.5} 2F(\text{eeg}(\tau))d\tau. \tag{6.2}$$

The algorithm was initially implemented and tested using MATLAB [106]. As segments of recording data are acquired by PDS, the AF is notified via the event system resulting in the agent being instantiated. When the `aEEGAgent` data requirements are met, tasks are created and processing of the EEG data commences.

The MATLAB implementation was used to verify the agent's output. The persistent agent state preserved between task invocations consists of the buffered filter inputs and outputs, and the previous 2s of output from the nonlinear mapping function. Unit testing was used to ensure that the agent state was persisted and reconstructed correctly.

Evaluating The `aEEGAgent`

Experimental Procedure

1. The `aEEGAgent` was deployed to an instance of PDS.
2. EEG data was streamed to that instance of PDS for approximately six hours.
3. This data was acknowledged by the AF and an `aEEGAgent` was instantiated to generate an aEEG recording.
4. Metrics related to the `aEEGAgent` were monitored, including the arrival time of the EEG data and the time taken for the derivation of the aEEG data.
5. A clinical neurophysiologist reviewed the data derived by the `aEEGAgent`.
6. The clinical neurophysiologist compared this derivation to the output of the Matlab implementation.

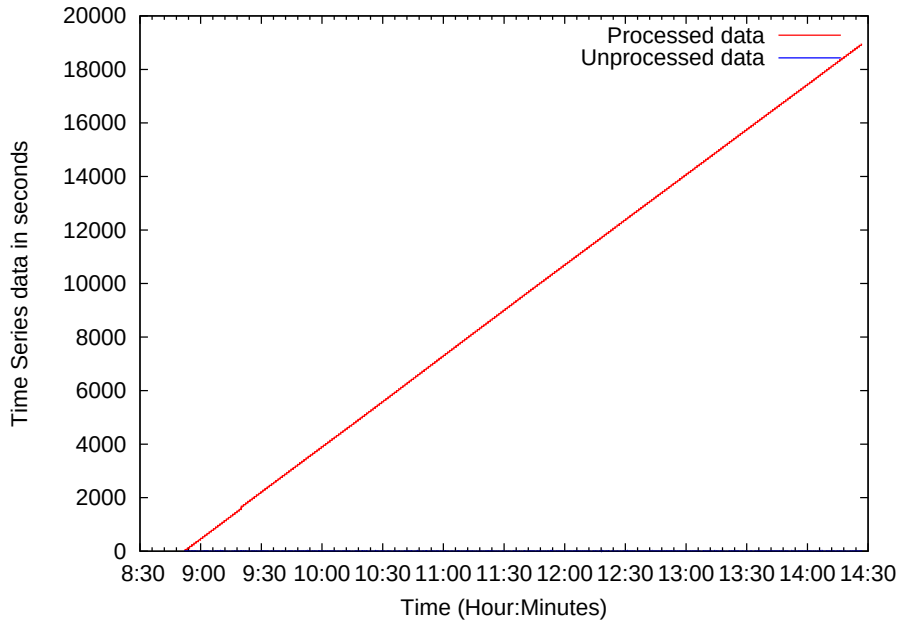

Figure 6.4: Available source data (EEG) graph against available derived data (aEEG) on the system.

Results

The processing of the EEG data by the `aEEGAgent` results in an aEEG recording being made available in real time. This recording can then be visualised in the web-based viewer and so is available to experts for analysis. Figure 6.6 compares a visualisation of the aEEG derived by the `aEEGAgent` on PDS and a screenshot of the same aEEG as derived by the Matlab implementation the agent was based on. The visual output of the aEEGAgent was reviewed by a clinical expert in neurophysiological monitoring and deemed sufficient for clinical diagnosis. As can be seen in Figure 6.6 there is little or no difference between the visualisations. Furthermore, Figure 6.4 indicates there is no delay in the availability of the aEEG visualisation due to processing requirements of the EEG.

Discussion

The `aEEGAgent` presents a number of desirable characteristics. It demonstrates PDS's capacity to integrate analysis techniques (ICs) that are beneficial to analysis, in this instance by broadening the scope of expertise that can interact with patient data. Similarly it provides an alternate data format that can be analysed by existing analysis techniques on the system. This demonstrates both the extensible nature of PDS and the reasoning for which that extension should be facilitated.

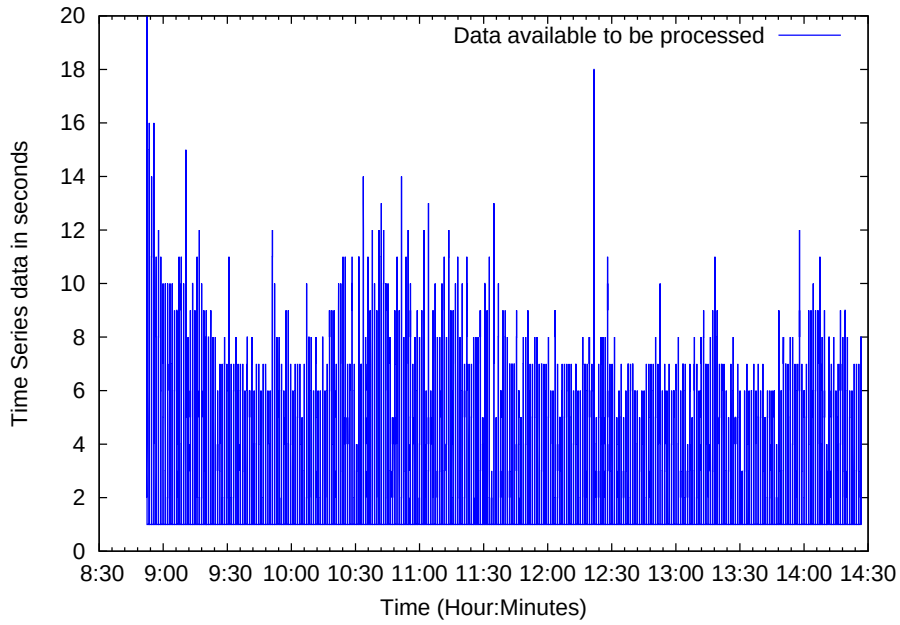

Figure 6.5: EEG data available awaiting processing by the `aEEGAgent`.

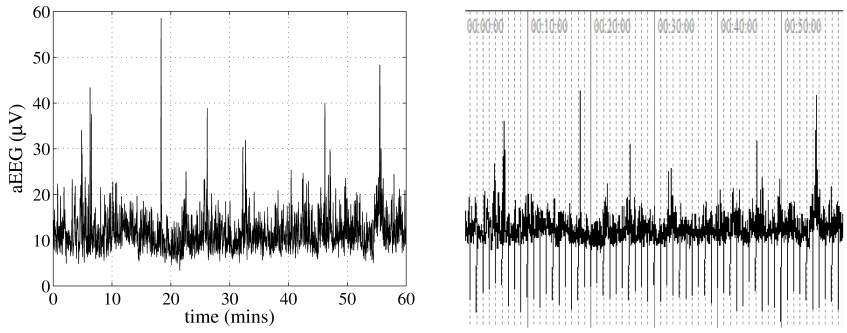

Figure 6.6: aEEG visualisation of two channels F4-C4 generated by MATLAB (on the left) and the `aEEGAgent` (on the right).

6.5 Assisting Interpretation

A large body of work exists on automated analysis techniques for the processing of EEG data. Typically, these techniques are used to identify features of interest (*e.g.* detect ailments indicative of an undesirable patient state), act as decision support aides or for knowledge discovery purposes.

Seizure detection algorithms are examples of analysis techniques. These algorithms use a variety of signal processing techniques, as well as AI and Machine Learning approaches, to process EEG with the intent of confirming the presence, or absence, of seizure activity. For example, in [107], a neural-network-based approach that operates in five stages: filtering, artifact detection, feature extraction (of both candidate and non-candidate data),

redundancy and relevance analysis is presented. In [108], the authors focus on the use of Discrete Wavelet Transformation as an improved time frequency representation for EEG in order to improve classification results.

In [109], three algorithms (Celka *et al.*, Gotman *et al.* and Liu *et al.*) for the automated analysis of neonatal EEG are evaluated. The algorithms employ a variety of classification techniques, such as modelling, complexity analysis, rhythmic discharge detection and auto-correlative functions. The algorithms were found to have sensitivities ranging from 42.9% to 66.1% and specificities ranging from 54% to 90.2%. It is interesting to note that the specificities and sensitivities differ from those originally published by each respective algorithm and opens the question as to how best to publish one's work such that it can be empirically re-implemented and tested.

Alternative approaches to traditional signal processing techniques have also been taken. For instance, in [110] it is stated that classifier based methodologies are too rigid and not suitable due to the large variance in neonates' frequency, morphology and topography. The method of detection proposed is to mimic a human expert. An algorithm that identifies two of the main characteristics that a human expert uses to detect seizure has been devised. The algorithm focuses on the background EEG and the recurrence of pattern in the signal.

To demonstrate PDS's capacity to assist experts analysis, a seizure detection-based analysis technique was implemented. The Gotman algorithm [111, 112, 113, 114] was chosen. As an algorithm for the automatic detection of seizure in the newborn it was considered an appropriate choice as an IC.

6.5.1 The `GotmanAgent` (IC)/(DC)

The Gotman algorithm is comprised of three methods for analysing EEG data: spectral analysis to find rhythmic discharges; spike detection to find groups of abnormal spikes that may not be rhythmic; and low-pass digitally filtered EEG to find very slow discharges.

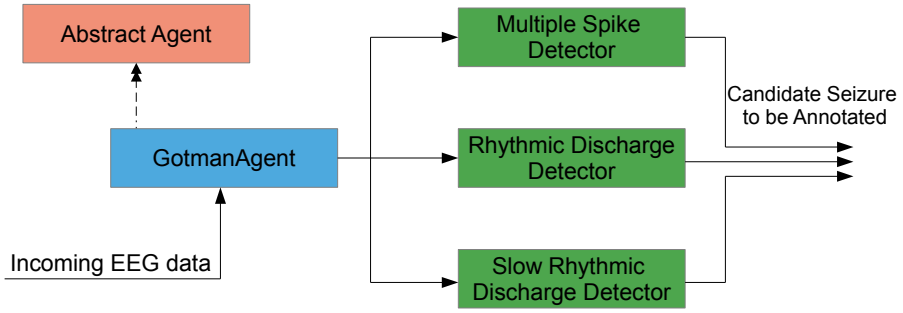

Figure 6.7: Components of the Gotman Agent.

Table 6.3: Values indicative of seizure in the Rhythmic Discharge Detector.

Dominant Frequency	WDSP	Power Ratio	Stability	PEC	PDC
0.5-1.5	≤ 0.6	3-4	< 3	< 0.8	> 0
1.5-10	≤ 0.6	2-4	< 3	< 0.8	> 0
1.5-10	≤ 1	4-80	< 3	< 0.8	> 0

Rhythmic Discharge Detector

This component involves the detection of rhythmic discharges from 0.5Hz - 10Hz. Spectral analysis is performed on 10-second time intervals of sequential EEG data, with a 2.5-second sliding window between time intervals. The frequency spectrum is computed using a fast fourier transform (FFT) with a length of 2048 points and a frequency ranging from 0Hz - 100Hz. Two prior time intervals, with a gap of 60 seconds, are used to provide background. The following are computed for each time interval: the dominant frequency, the width of the dominant spectral peak (WDSP), the power ratio, the stability of current time interval, poor electrode contact (PEC) and the patient disconnected indicator (PDC). If the dominant frequency, WDSP and power ratio for a given time interval are found to be within the specified range (see Table 6.3) for stability, poor electrode contact and patient disconnected indicator, it is considered a positive detection.

Multiple Spike Detector

This component uses a spike-detection algorithm developed for adult EEG [114]. The adult algorithm was modified due to a tendency towards longer spikes in neonatal EEG. An IIR filter is used to perform high-pass digital filtering of the EEG prior to spike detection to an order of 3 with a cut-off frequency of 2Hz. A detection is considered positive when 6 or more spikes are detected within a 10s time interval.

Slow Rhythmic Discharge Detector

The detection of slow rhythmic discharges uses an IIR filter to perform low-pass digital filtering, of order 2 with a cut-off frequency of 2Hz, of the EEG prior to it being processed by another algorithm also developed by Gotman [113] using the same time interval duration, gap and background duration as above.

Performance

The average seizure detection rate reported by Gotman has been found to be 69% with an average false detection rate of 2.3 per hour [112]. The algorithm can be decomposed into three independent functions for processing the EEG as depicted in Figure 6.7. Initially, each

process was developed and tested independently, with validation being performed using a pre-existing MATLAB implementation.

Evaluating the `GotmanAgent`

The algorithm as described in Section 6.5.1 was initially implemented and tested using MATLAB. As segments of recording data are acquired by PDS, the AF is notified via the event system resulting in the `GotmanAgent` being instantiated. When the `GotmanAgent`'s data requirements are met, tasks are created and processing of the EEG data commences. Upon detecting a seizure, the agent annotates the recording.

Upon reviewing the data, an expert is presented with a list of annotations. The expert can decide whether or not the `GotmanAgent` has classified the data correctly. This aids experts in analysing data by drawing their attention to classified sections. Naturally, the agent processing alone is not suitable for clinical detection due to the occurrence of false positives and false negatives, but it still is able to assist the expert in their analysis.

Discussion

This demonstrates one means in which PDS can support automated analysis of data; it also demonstrates its capacity to communicate the analysis results to an expert. In constructing re-useable, and extensible, ICs, PDS is contributing to an increased level of analysis; this in turn will assist with the uncovering of new knowledge as increased quantities of information are derived from raw data. It is envisaged that these knowledge repositories, and their classified data, will assist in advancing the understanding of the domain from which they originate.

6.6 Evaluating The Distributed Alternatives

PDS has proven to be a robust clinical tool within the field of clinical neurophysiology. This is demonstrated by the testbed, as discussed in Section 5.6, and through PDS's support of a clinical drugs trial, whereby it facilitated collaboration between distributed experts, as discussed in Section 6.6.2.

6.6.1 Supporting Neurophysiological Monitoring

A walkthrough of how clinicians use PDS and its remote reviewing capabilities are illustrated in Figures 6.8– 6.14.

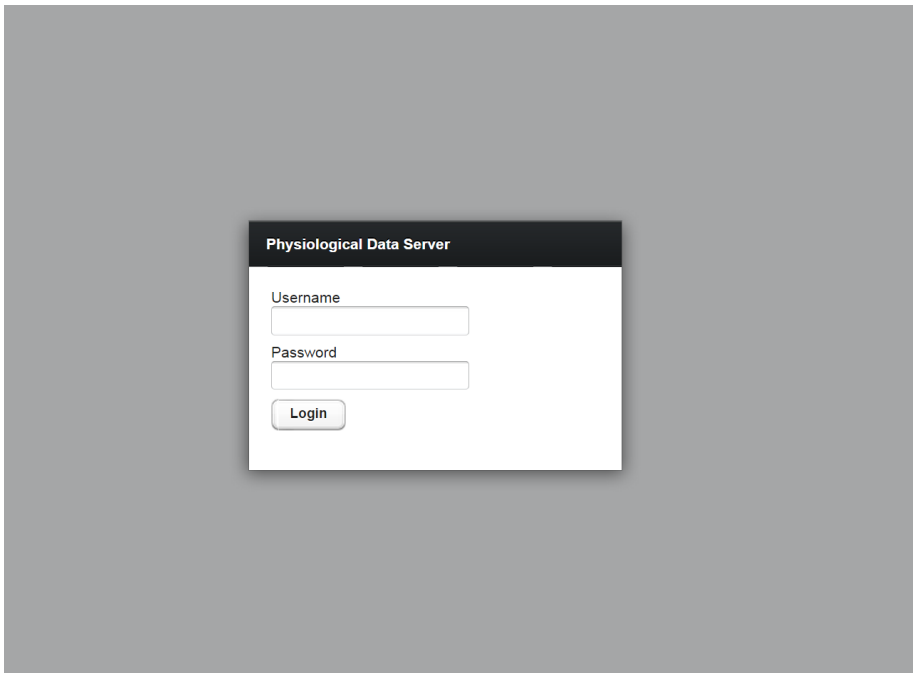

Figure 6.8: The login panel for secure user access.

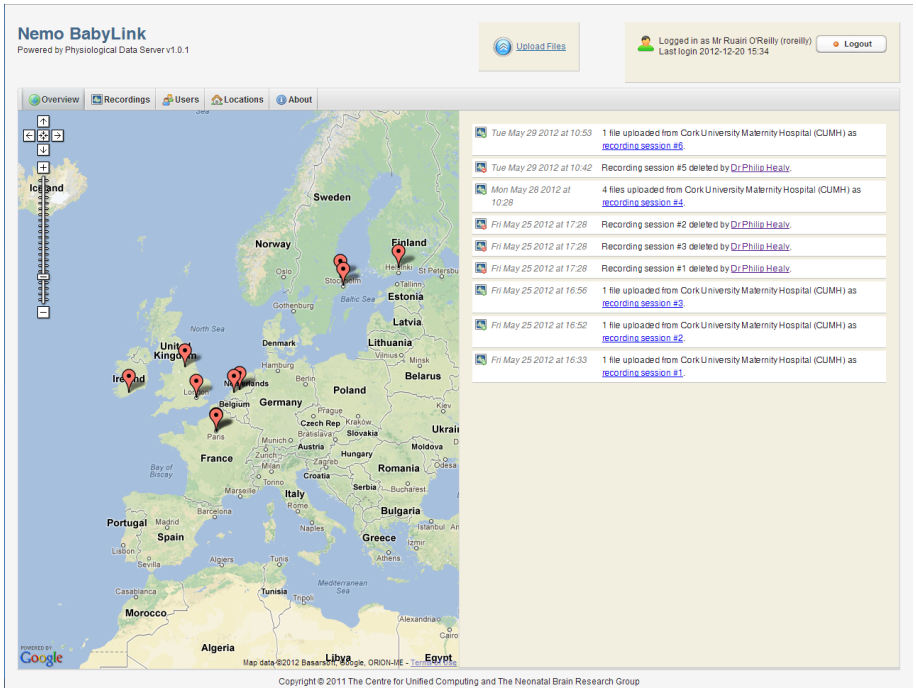

Figure 6.9: The view presented to users after logging in to PDS.

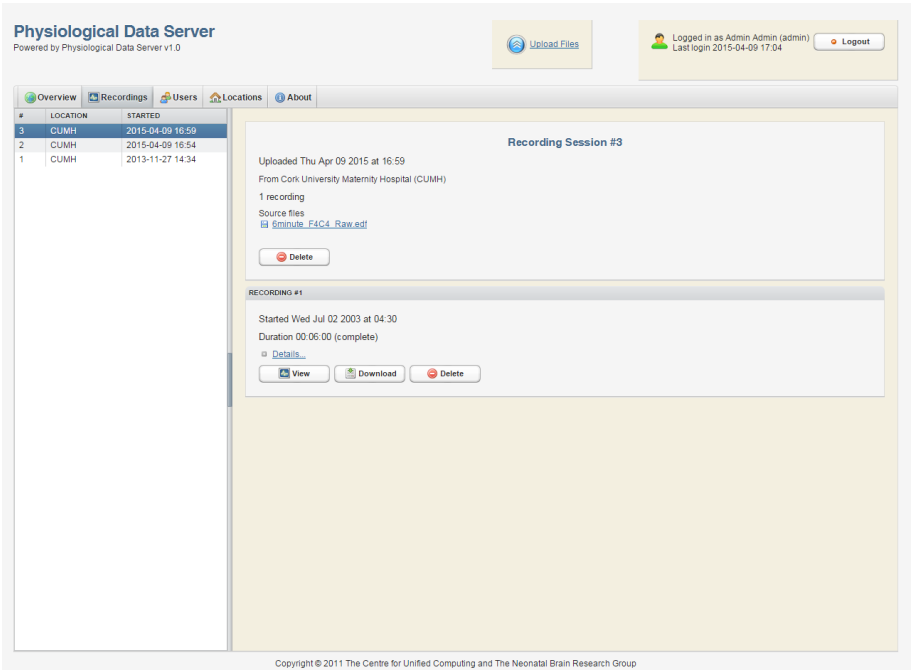

Figure 6.10: The recordings tab maintains a list of all EEG data on PDS and relevant meta-information.

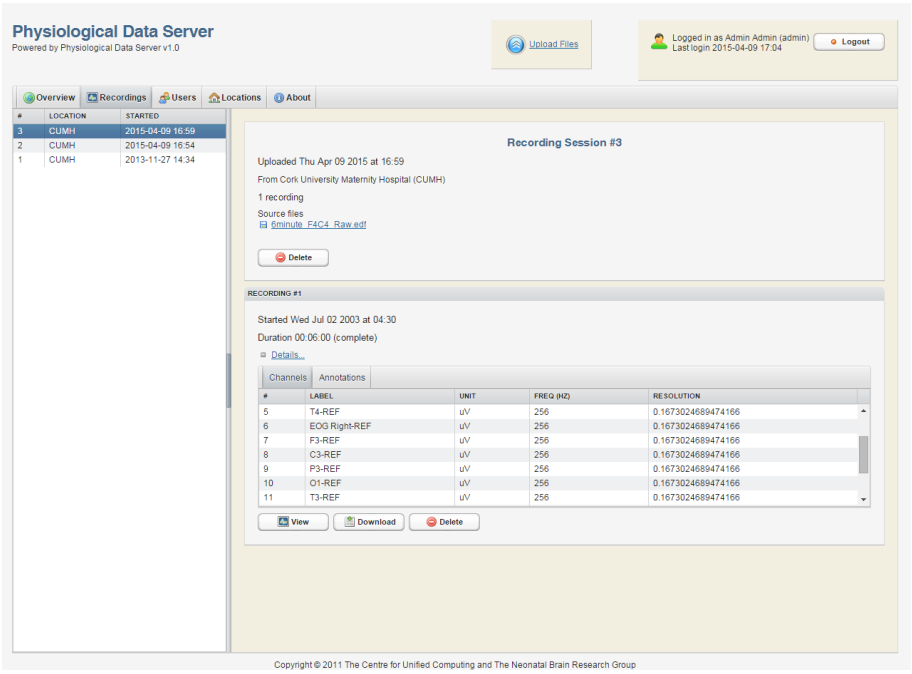

Figure 6.11: PDS enables you to view detailed information about a recording, such as channels present, frequency and annotations.

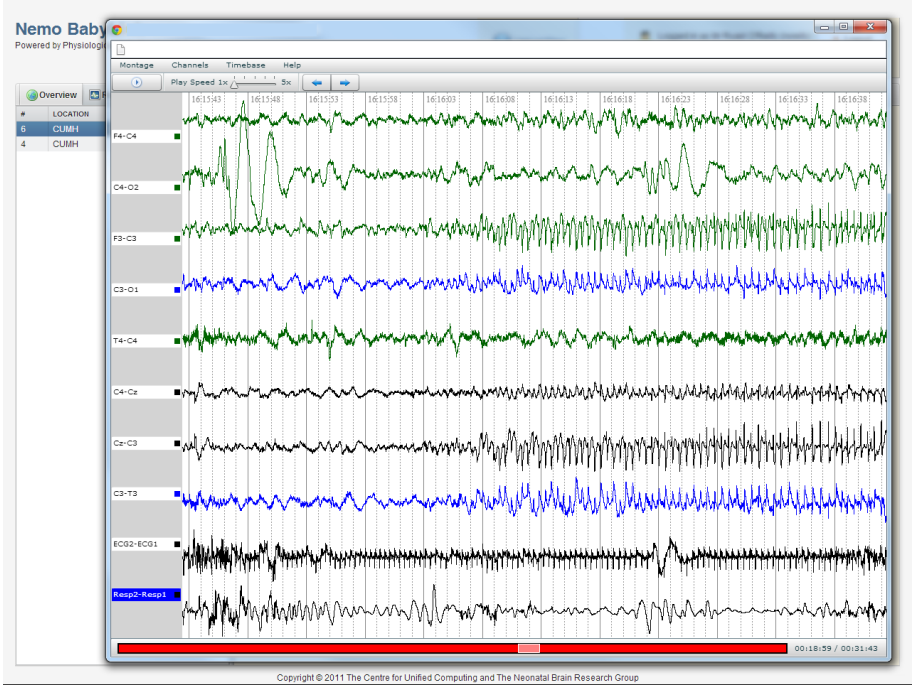

Figure 6.12: A user is able to interact with a recording via the Viewer. In this instance, 8 channels of EEG are being displayed, as well as a single channel depicting Respiratory function and ECG.

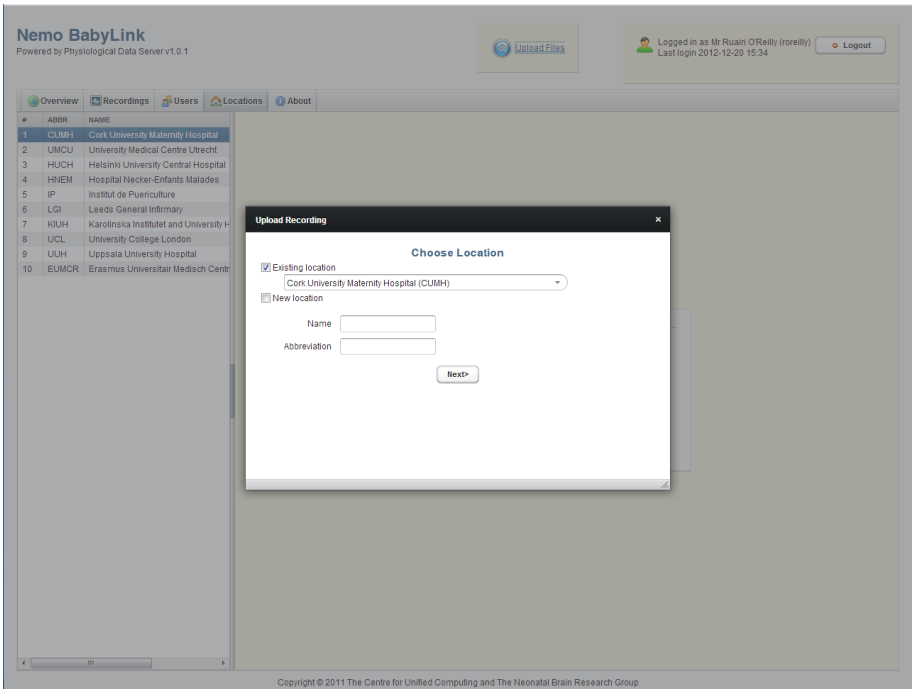

Figure 6.13: Adding a location. Used to associate geographic locations with the origin of data on PDS.

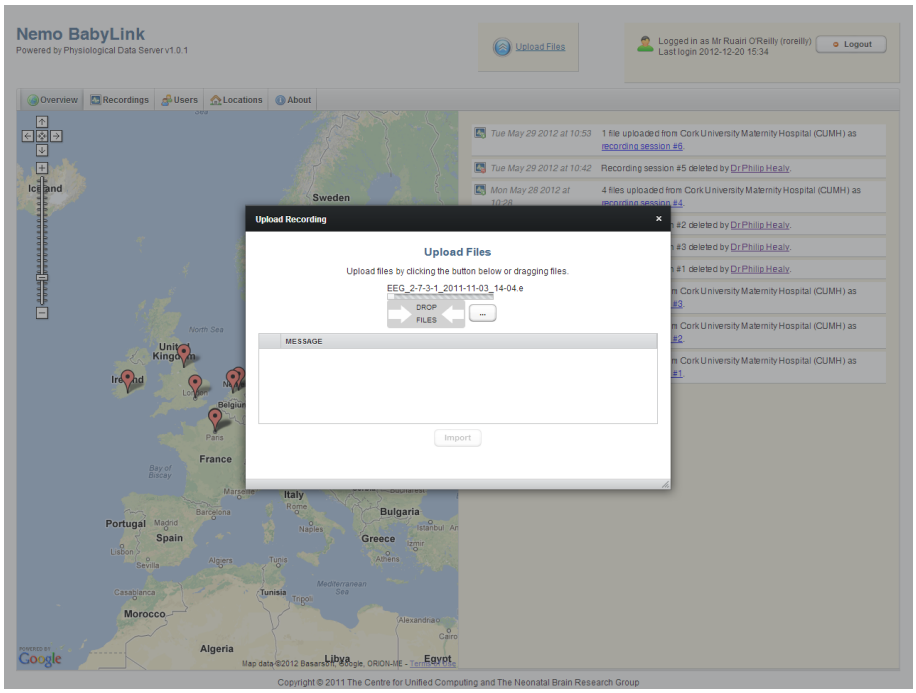

Figure 6.14: Manually uploading an EEG file to PDS.

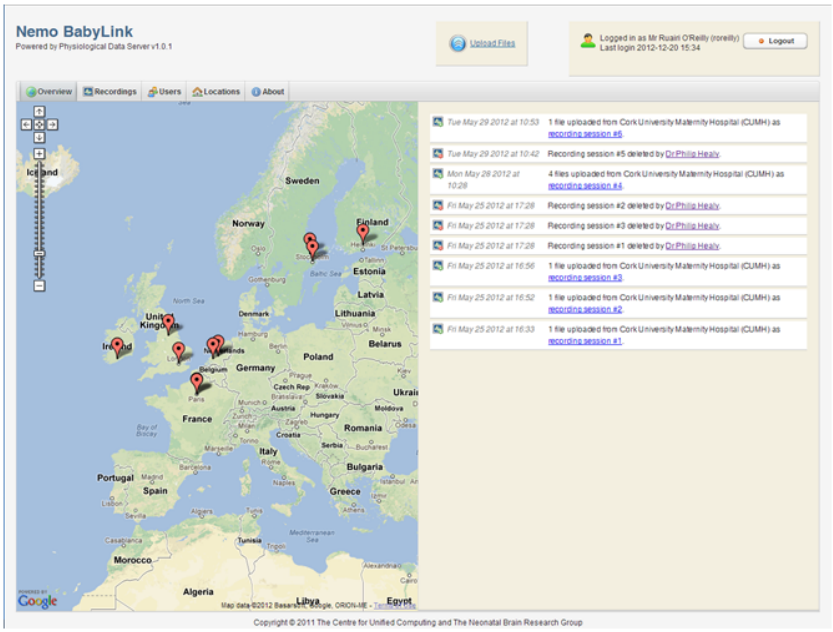

Figure 6.15: Overview screen of PDS's web application. Locations with EEGs available for review are indicated in the pane on the left. The pane on the right contains a summary of recent activity.

6.6.2 The NEMO project

Treatment of **NE**onatal seizures with **M**edication **O**ff-patent (NEMO) [115] is a study on the effects of bumetanide in the treatment of neonatal seizure. The study occurred at an international level with 14 participating institutions based in seven different EU member states and one partner in the USA (some of which are depicted in Figure 6.15). The goal is to develop a drug-based course of medical treatment for the seizures that affect the newborn in the form of anti-epileptic drugs.

The first stage of the study involved the evaluation of bumetanide, a drug that could potentially alleviate one of the main causes of seizure in neonates. The study was a first of its kind in that it was "the first time that an anti-epileptic drug (AED) specifically aimed at this age group will be evaluated in a large, adequately powered, randomised trial with EEG monitoring, recognised to be the "gold standard" method for seizure diagnosis in the newborn." [115].

In the context of neurophysiological monitoring, the challenge facing the project was the lack of expertise in neonatal EEG to allow the diagnosis of seizure and the administration of bumetanide to take place. The reasons for this occurring are similar to those discussed in Section 1.4 with this instance being an example of the problem at a larger scale.

PDS provided the means to remotely collect, collaborate and analyse the EEG data required for the study, thus enabling institutions without the necessary level of expertise to participate in the study thereby assisting in the evaluation of the drug.

Workflow Of PDS Within NEMO

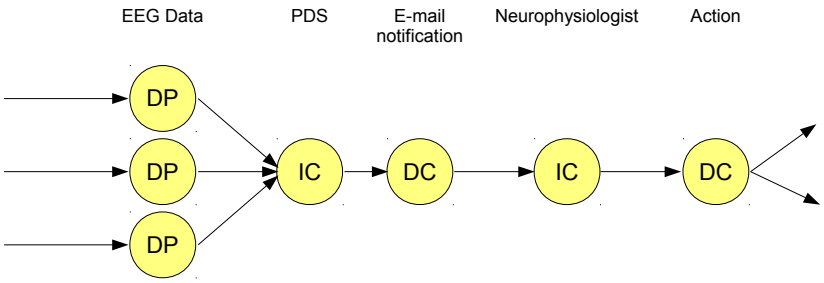

Figure 6.16: IE workflow within NEMO.

The typical use case of PDS was as follows. A patient is identified as having a suspected neurological condition. Acquisition of EEG data commences, with the resulting EEG data being saved to a file server. On the file server, the upload application is invoked upon the EEG data file. Any data present in the file is immediately uploaded to the data server. The

upload application then periodically determines whether additional data has been added to the file and if so, the new data is uploaded.

Once the transfer of data from the acquisition location has commenced, the data is queried to evaluate if it is of the correct type, in this case EEG. If so, the neurophysiologist at the remote location is contacted via e-mail. The neurophysiologist logs into the data server and selects the appropriate recording. The viewing application then opens in a popup window, providing the neurophysiologist with a continually updating view of the recording. The neurophysiologist analyses the EEG data and issues a clinical report containing his/her findings. The report is then uploaded to a separate web-based system for recruitment within clinical trials. Based on this report, the decision is taken whether or not to administer the bumetanide. The results of which are recorded (dosage, physiological ailments and patient outcome) thereby adding to the empirical evaluation of the drug as a form of treatment. These results are stored in a separate proprietary system that specialises in data entry for clinical trials and is managed by a private company.

The end result is an IE that is capable of providing a teleneurophysiology service for clinical trial recruitment. From a technical perspective, specific functionality is provided for each role in the review process: upload, review, administration *etc.* Access control and other security requirements were also rigidly enforced. The remote review feature of the system enables rapid in-browser remote review of uploaded EEG files, allowing candidate patients to be evaluated quickly. The web-based viewing application provides neurophysiologists with the features that they expect to have available for EEG interpretation.

6.6.3 Discussion

Feedback to date has been very positive, with users expressing satisfaction with functionality and usability of PDS when compared with more primitive methods of remote monitoring, such as those discussed in Section 1.2.3.

The testbed (see Section 5.6) and the NEMO study (see Section 6.6.2) provide real world applications of PDS. In order to fully demonstrate the benefits of PDS's distributed architecture, the workflow had to incorporate a greater number and variety of components.

6.7 National Data Store

To demonstrate PDS's distributed architecture and to highlight the benefits to neurophysiological monitoring, it was decided to simulate Ireland's NICU infrastructure. In total on the island of Ireland there are 35 NICUs, 26 in the Republic of Ireland [116] and 9 in Northern Ireland [62].

A National Data Store (NDS) was created to simulate a repository for all EEG data generated by Irish NICUs and demonstrate PDS's capacity to act as an expert knowledge repository at scale. In this manner, PDS is capable of facilitating the acquisition of data from multiple distributed NICUs (IE) and providing resources at scale by co-locating multiple components in a single location (see Section 6.8.1). This provides neurophysiologists with the means required to interact with the data, both locally and remotely.

6.7.1 Architectural Overview

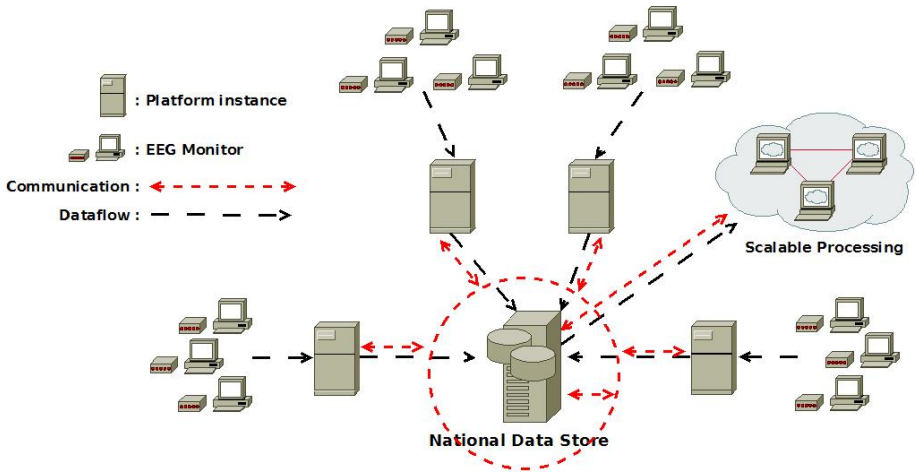

Figure 6.17: PDS depicted as a distributed network of interconnecting components.

Once a NICU is connected to the NDS, transmission of any pre-existing data commences and is continuously transferred as new data arrives at the NICU. Cross-contamination of data is avoided through the use of Universally Unique Identifiers (UUIDs) for identifying datasets belonging to individual NICUs. The NDS provides the functionality for the long-term storage of the data and acts as a gateway through which a proliferation of analysis techniques can be employed.

Experimental Procedure

Demonstrating the architecture involved the creation of multiple instances of PDS (8) with four data streams concurrently uploading data for approximately a 12hr period. This equates to 32 neonates being monitored at any given time. This figure exaggerates the number of neonates that are monitored in practice. However, it is an acceptable upper limit for what the system should support. Given that 16-data-streams are used for routine monitoring of a neonate with a sample rate of 256 samples per second per channel, this can lead to the continuous transfer, storage and processing of a large quantity of data.

Table 6.4: Expected data transfer between NICUs and the National Data Store.

Case	Web Requests transfer	No. of NICUs	No. of neonates	Per neonate
1 second	700kb	35	140	5kb
1 minute	41Mb	35	140	300kb
1 hour	2.460Gb	35	140	17.57Mb
1 day	59.06Gb	35	140	421.87Mb

A routine monitoring period for a neonate can be up to three days. If used for the monitoring of 140 neonates simultaneously this would amount to approximately 177Gb being transferred to the NDS (see Table 6.4). However, the experiment was not run at that scale. Log files detailing the state of the system were recorded for approximately a 12hr period. These were then parsed to derive performance metrics. With this data we can extrapolate how PDS handles large quantities of data, the effect on data integrity and identify correlations between the number of NICUs and the quality of service provided.

Results

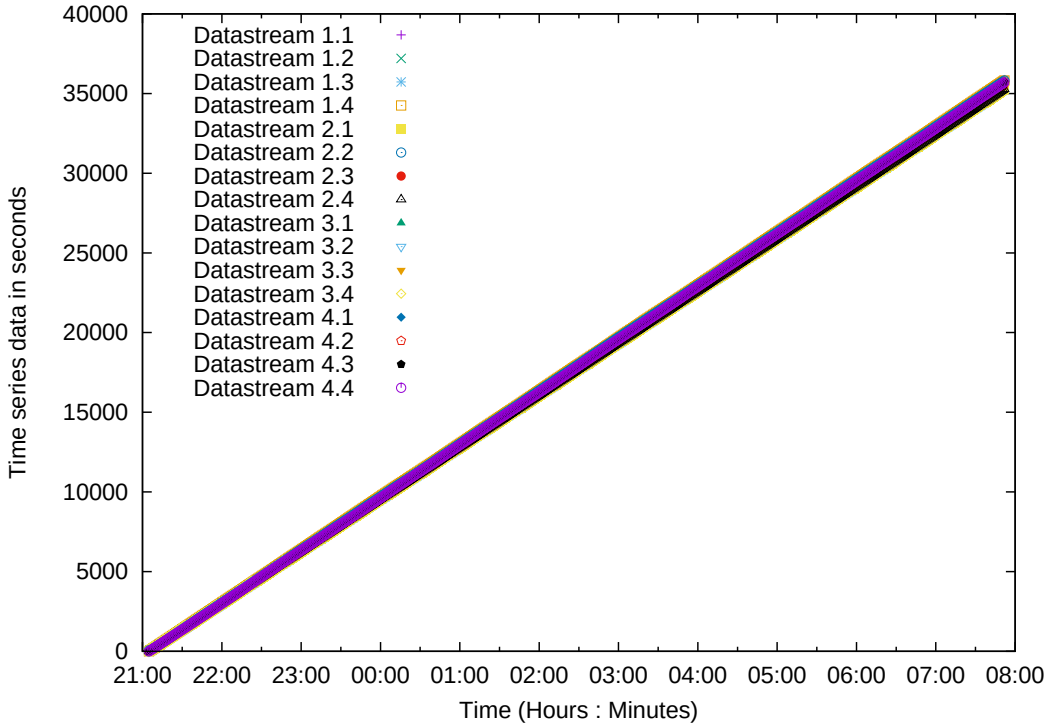

Figure 6.18: Real-time acquisition of multiple concurrent throttled data streams.

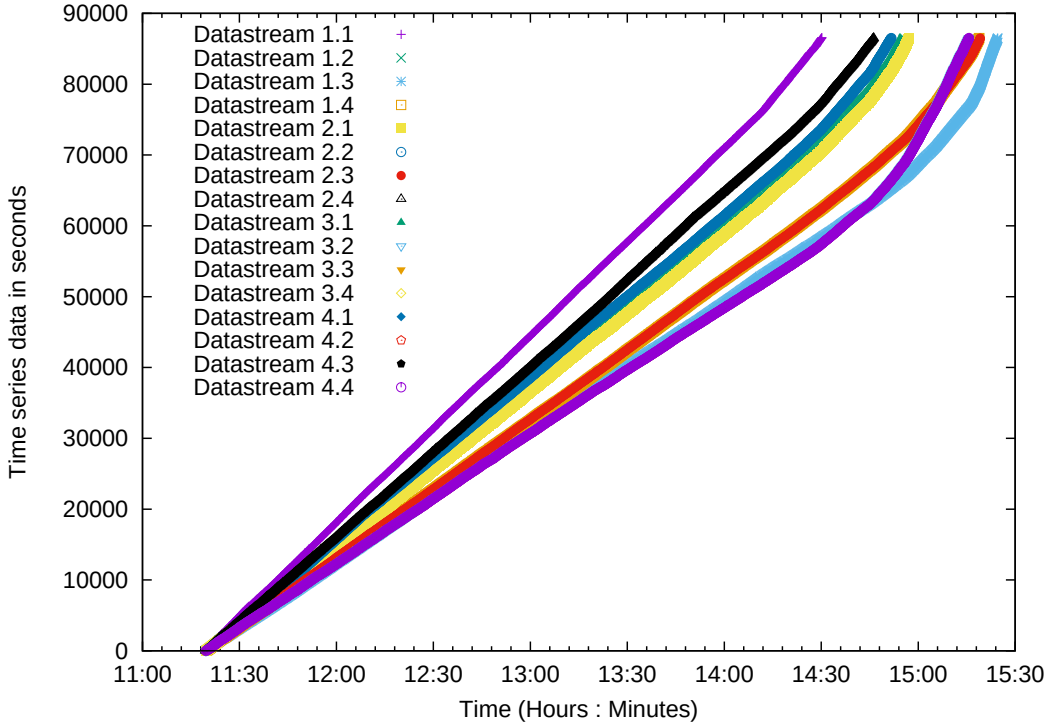

Figure 6.19: Real-time acquisition of multiple concurrent unthrottled data streams.

Discussion

The NDS was capable of handling multiple data streams from multiple instances of PDS concurrently. This resulted in 32 streams of data being written to the NDS at any given time. A selection of these streams are depicted in Figure 6.18 where data is transferred at a throttled rate to mimic real time. In Figure 6.19, data is transferred at the default rate. As can be seen from Figure 6.18, when data is transferred in real time there is no delay in acquisition. When the data is throttled at the default rate, as depicted in Figure 6.19, a slight degradation in the rate of acquisition is experienced over time, but is still approximately three times faster than the real-time rate of generation.

6.8 Integrating Cloud-based Services

One infrastructural advance PDS's distributed co-locating approach can avail of is the elasticity of the cloud. The cloud enables access to on-demand resources such as compute and storage, thereby enabling scalability at a level that would normally be outside the reach of a NICU. On-demand resources are applicable to a wide range of analysis techniques. One of those we consider is search based classification (SBC). SBC is an analysis technique that utilises the availability of pre-existing classified data, to classify newly acquired data. However, this can be a computationally expensive technique, particularly when dealing with large quantities of high-frequency data. This makes it an ideal candidate as a cloud-based Service. PDS is capable of integrating this form of analysis by using the Agent Framework to process queries on newly acquired data using an analysis technique called "ScrutiniseIt".

6.8.1 ScrutiniseIT

ScrutiniseIT is a cloud-based analysis technique for feature detection that leverages expert knowledge repositories and scalable computation. The *scan-and-scrutinise* based methodology, originally outlined in [117], consists of two aptly named phases "scan" and "scrutinise". The scan phase of the algorithm runs continually on data streams being acquired. A simple evaluation function is used to determine data segments that warrant closer examination. Upon detection of a potential feature of interest, that section of the stream is more closely scrutinised. The scrutinise phase determines if particular features of interest are present in the candidate signal. This is done within a specified confidence time interval. It incorporates the scalability benefits of cloud computing, enabling large volumes of data to be continuously analysed for the identification of features of interest. It relies on expert annotations to the data and an algorithm designed to search through existing data repositories.

The TS data stream is treated as a series of time intervals. Each time interval is

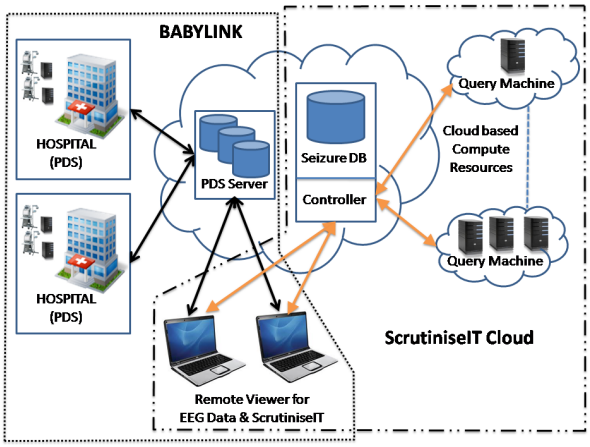

Figure 6.20: A high level view of PDS and ScrutiniseIT components in operation.

examined for potential matches to the database of known features. Once a pre-determined threshold has been reached, the potential for a match is noted, processing of that time interval ceases, and the process continues by advancing to the next time interval. The size of the time interval is defined by the Window Size (WS) parameter described below. A match occurs when a series of points observed match a similar series in the database.

ScrutiniseIT acts as a stand-alone component accessible by the individual instances of PDS via the AF. Through the tuning of different parameters, the algorithm can either accelerate its search through the TS data (*scanning*), or perform slower, more detailed, analysis (*scrutinising*). These parameters are:

Confidence: a percentage value indicating a minimum threshold that must be exceeded before a match with a feature of interest is recognised. The higher the confidence specified, the more exacting the match must be before the feature is reported.

Window Size (WS): ScrutiniseIT uses a dynamic sliding window to contextually analyse a candidate signal pattern. Larger windows are used during the scan phase when trying to identify regions of interest in the candidate signal. When such a region is identified, the window size contracts appropriately to scrutinise these regions in more detail in an attempt to report features for a given confidence.

Grain Size (GS): ScrutiniseIT operates by matching turning points in the incoming EEG with similar points in the database of known seizures. The grain size specifies the number of points required for a match to occur and is used to determine when an advance to the next time interval is triggered. The larger the grain size, the greater the accuracy of the resulting matches.

Threshold Deltas δt, δp : Matches are compared on each point at time +/- δt, and with accuracy +/- δp.

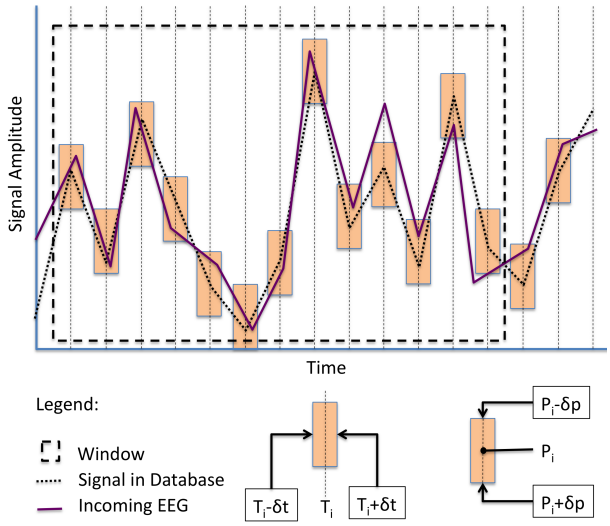

Figure 6.21: Signal matching in a ScrutiniseIT search.

Figure 6.21 illustrates the matching process in action. The dotted line shows a sample signal from the database. Each turning point in the signal has a known amplitude, which represents a point of interest. The window size being examined is encapsulated by the dashed rectangle. For each point of interest, the boundaries defined by the δt and δp points are shown as a shaded rectangle. The candidate signal is compared against the database of known signals. Although, in this case, the candidate signal exhibits a slightly different form; within the given window there are 11 matches out of 13 possible turning points exhibiting a confidence rating in the order of 84%. If a confidence rating less than 84% had been specified, a match would have been reported, if greater than 84% no match would have occurred. There is a trade-off between the time taken to process the signal and the search parameters such as window size (WS) and grain size (GS). The data shown in Table 6.5 illustrates this - the larger the window size, the faster the algorithm executes, but the less accurate the match with the points stored in the database. Similarly, the smaller the grain size, the faster the algorithm executes, but the less accurate the match.

Therefore, our initial results show that a larger grain size will yield more accurate matches, but at a cost of slower execution time. During our preliminary tests we have observed that for signals containing no known features, the number of matches is tiny.

Table 6.5: Results obtained from a number of searches altering WS and GS values. Speedup obtained is also shown.

Window Size	GS = 5 Time(s)	Matches	GS=10 Time(s)	Matches	GS=15 Time(s)	Matches	Speedup GS=5	GS=10	GS=15
0	3216	1840	3216	920	3216	613	1	1	1
256	612	176	1536	113	2208	108	5	2	1
512	476	65	898	60	1303	59	7	4	2
768	354	43	634	42	962	40	9	5	3
1024	266	33	511	31	677	32	12	6	5
1280	199	26	441	25	503	25	16	7	6
1536	183	21	329	23	420	21	18	10	8
1792	144	18	244	18	371	18	22	13	9
2048	137	16	268	17	381	16	23	12	8
2304	92	14	224	14	290	14	35	14	11
2560	79	13	216	14	273	13	41	15	12
2816	95	12	189	12	168	12	34	17	19
3072	76	11	178	10	241	11	42	18	13

Application To Neurophysiological monitoring

ScrutiniseIT affords clinicians the ability to collaboratively enhance one another's work. Clinicians annotate newly acquired data to highlight the occurrence of features of interest and this data becomes part of the annotated data repository. Subsequent processing of the data repository will be influenced by these contributions, thus altering the results of the system over time.

This service avails of a search-based approach to seizure detection that relies on comparison with a growing body of acquired data. It is not intended to compete with traditional signal processing techniques but rather to demonstrate the capabilities of utilising cloud-based services in conjunction with PDS. ScrutiniseIT enables PDS to provide an advanced seizure detection service on demand that would be beyond the scope of a local NICU. The processing of EEG can be computationally expensive, therefore availing of cloud infrastructure allows the required resources to be employed only when necessary.

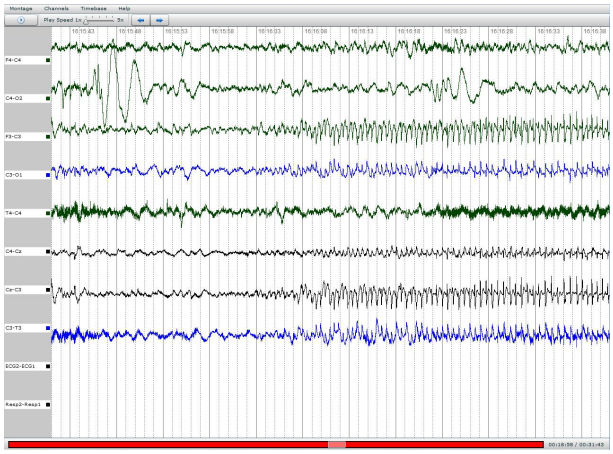

Figure 6.22: The view of EEG provided to a clinician through the viewer. Seizure is visible on channels F3-C3, Cz-C3, C3-01, C3-T3, C4-Cz, so predominantly on the left hemisphere of the brain.

Ideally, all data streamed from local instances of PDS would be processed by a service such as ScrutiniseIt. In doing so, all candidate EEG would be compared to a set of expert-annotated EEG data containing seizures. This would be beneficial in increasing the quantity of raw data being analysed and increasing the quantity of seizure data collected.

Results

Initial tests of ScrutiniseIT have been conducted with a limited database of known seizures. An excerpt of the results from these tests are listed in Table 6.5 and depicted in Figure 6.23. In Figure 6.23(A) total execution speedup obtained versus selected window size is shown.

Three sets of results are shown for different grain sizes. It can be seen that significant speedup can be achieved by selecting a smaller grain size, albeit at the cost of a reduction in the quality of matches. For the tests illustrated in this example, a confidence value of 70% was chosen.

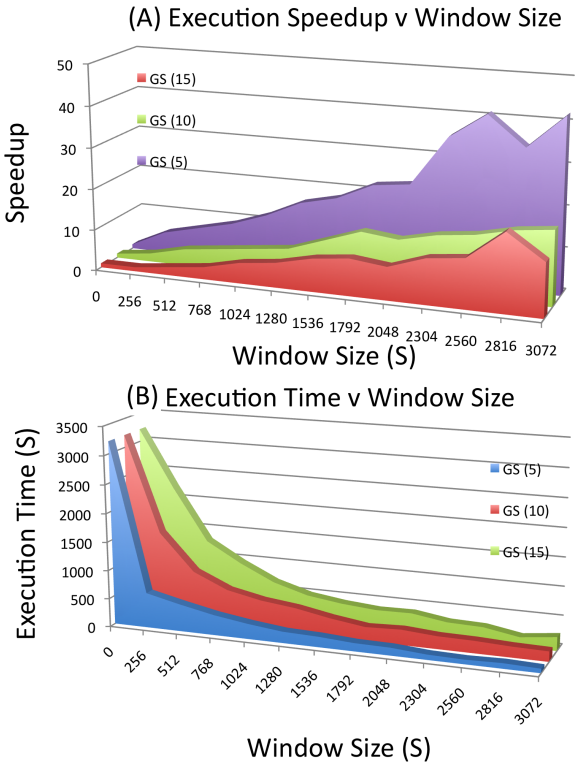

Figure 6.23: Preliminary results from ScrutiniseIT evaluation.

Figure 6.23 (B) illustrates that, as window size is increased, the total execution time is reduced. This is due to the reduced number of time intervals that are available for selection. If a significant number of contiguous matches are found, PDS begins the scanning process at the next time interval. So, larger window sizes result in faster, less accurate, searches.

Discussion

PDS's capacity to extend its functionality via components such as ScrutiniseIT enables access to levels of computation that would otherwise be unattainable. It is a symbiotic relationship in that PDS provides ScrutiniseIT with the data required (both EEG data and expert annotations) and ScrutiniseIT provides PDS with the computational means to carry out real-time seizure detection through the utilisation of cloud based resources. ScrutiniseIT demonstrates how a component that fulfills the role of an IE component can be developed and integrated with an IE's workflow.

Chapter 7

Conclusions

In [3] it is stated that "there exist glaring limitations in the datasets we investigate, the metrics we employ for evaluation, and the degree to which results are communicated back to their originating domains". These concerns were also articulated by Keogh [52] in relation to TS data mining. Keogh [52] surveyed a large body of work in that domain, re-implemented the techniques involved and carried out an exhaustive set of experiments. Subsequently, he found that the level of "improvement" offered by many of the published works (constituting new paradigms, algorithms and techniques for TS data mining) was of little, or no, utility due to the variance that would have been experienced by testing the work on a broader range of TS data sets, or altering minor unstated implementation details.

Keogh pointed to the lack of a standardised approach and the inability to accurately compare works, which, on the surface, would appear to be comparable. He recommended the following:

i) The design of experiments should be free of implementation bias.

ii) Transparency should be ensured by making code and data available where possible.

iii) The approach detailed should be tested on a wide array of datasets.

iv) The utility of a reported advance should be limited to that which can be demonstrated.

This insight by Keogh [52], pointing to a potential crisis in the evolution of scientific knowledge, motivated the design of the generic IE described here. This environment, composed of transparent workflows and standardised reusable components, attempts to make it difficult to inadvertently have the results of analysing TS datasets altered by opaque implementation details. In this manner, it was hoped that the transparency and standardisation, correctly identified by Keogh [52] as pre-requisites for the incremental building of a knowledge base, would be provided.

7.1 Summary

In this dissertation, platforms for monitoring and analysis of data are recognised as being composed of three components DP, IC and DC. This work showed how these components could be used in a workflow description that explicitly captured the specifics of an application design. These workflows incorporate the concept of feedback and hierarchical position and through this rich structure are capable of expressing complex and evolving monitoring systems.

This work discussed the creation of a generic toolbox and architecture for the IE, which could be used by application designers to rapidly produce standardised environments using the workflow description as a blueprint.

The thesis looked at creating an application instance of the IE, in the form of a Physiological Data Server (PDS) for the remote monitoring of neonatal EEG. This system was non-trivial and contained many implementation challenges including the acquisition of data from a secure environment, the anonymisation of that data to conform with data protection rules, the interfacing with proprietary standards and libraries, near real-time visualisation and interpretation of that data by human experts and the provision of a feedback mechanism, in the form of annotations, to allow those experts to drive the system to convergence.

This work resulted in the creation of a first-of-its-kind, real-time viewer for analysing TS data (as demonstrated by the use case), a non-invasive means of acquiring TS data in real time while maintaining the integrity of the data and the identification of an exemplar design paradigm for a monitoring and analysis platform. In the context of neurophysiological monitoring, the platform has been demonstrated to enable remote expert analysis, increase the number of experts who can analyse the data and assist experts with their analysis.

Individual components within IEs encompass significant engineering contributions, as well as pushing the boundaries of our theoretical understanding of how systems for the monitoring and analysis of data should be conceptualised, designed and implemented. The IE provides a generic architecture for the analysis of TS data. It has the capacity to be tailored to support specific requirements of domains that analyse TS data, as was demonstrated through its application to neurophysiological monitoring. Instances of this problem occur in multiple domains. The benefits demonstrated for EEG analysis could easily be applied to other forms of physiological and TS data.

Tests were carried out to demonstrate that PDS was performant and fit for purpose and this was further validated by positive feedback received from experts in the field. Finally, the success of PDS was demonstrated, by its use in the NEMO project, and in its supporting of the work of other international researchers. The design of a national centre for expertise in neonatal seizure detection was articulated as a natural extension of the PDS system.

7.2 Future Work

i) Completeness of the IE: Future work could investigate if the IE is complete in that it is capable of capturing all monitoring and analysis applications.

ii) Workflow descriptions created by application domain experts, faithfully implemented by application designers. In the future, it would be advantageous if the implementation could be formally checked against the workflow description.

iii) Explicit support of provenance: Currently the system records an evolution of state from which provenance information can be subsequently reconstructed. In the future, it is anticipated that the use of a notarised record, consisting of the application of data transformations to data, as it moves through the system, could provide that provenance information. A possible solution may be found by applying a combination of hashing techniques, third-party seeds and the Java Serialization API.

iv) Investigating convergence algorithms and appropriate techniques for driving systems to convergence may involve the creation of a system for measuring "distance" between system states, so convergence and divergence can be readily measured.

v) ScrutiniseIT Container.

vi) Incorporate Keogh's algorithms [118] into the platform.

vii) Load-balancing of components: How are roles allocated to the current resources, based on their availability and the level of priority? How is a level of priority defined?

viii) In a system where IC are human experts, the potential for subjective bias is high. Mitigating this bias through the socialisation of the expert input, thus replacing potential subjective bias with community consensus, is an important goal. Mechanisms to effectively achieve that goal need to be explored further.

7.2.1 Supporting Computer-aided Science

It would be exciting to see the IE facilitate the process of Knowledge Discovery. As the IE contains a monitoring and analysis solution with an Agent Framework, it already has the majority of components required to do so. Extending the IE's functionality to enable KD techniques process data, while providing an objective analysis (and record) of that technique, would be one means of assisting analysis even further.

Integration is a key component as it enables non-technical practitioners to combine their expertise with the benefits of KD. Objectivity is achieved by enforcing a scientific means

of inquiry. The intent is to provide a transparent, and reproducible, analysis of both the techniques, the experimental design and the subsequent empirical validation.

It is envisaged that the objective analysis will form the basis of a comparative analysis for KD techniques. Continually evaluating these techniques against an evolving set of statistical measures and benchmark thresholds is the primary means of ensuring objectivity. It enables the ongoing verification of an algorithm's suitability for its intended task.

Observation of experiments undertaken with the IE could enable the collection of meta-data. This meta-data forms the basis of performance metrics, providing a confidence measure in the techniques employed. This enables a confidence measure to be applied to an algorithm. This rating is built on metrics measured and observed in a controlled environment that is being validated through experimentation on an ongoing basis.

This comparative analysis can dictate the suitability of an algorithm for a domain. While being derived from observed performance metrics, and building upon the existing knowledge of KD techniques, it provides a means of disseminating novel information that can be readily accepted, and/or challenged, by the scientific community. The laboratory-like environment assists in the analysis of TS data while simultaneously enforcing scientific rigour.

An objective overview of an algorithm's performance and data-mining capabilities would be highly beneficial to the KD community. The identification of the most suitable algorithm for solving a problem is a non-trivial task, as the ability to reliably compare algorithms' suitability requires an expert understanding of the workings of each algorithm under consideration.

Environments for the analysis of TS data in its general sense have long been, and will continue to be, an important tool in many application domains, the most fundamental of which is knowledge discovery using the rigors of the scientific method. Heretofore, expert opinion would suggest that these environments are somewhat lacking and open to incorrect use [52, 119]. The work presented here attempts to meet this problem head-on and to provide the first steps for the creation of a much needed standardised, open, transparent and rigorous environment.

Appendix A

Requirements

Requirements using the word "shall" are mandatory deliverables and those using the word "should" are desirable deliverables. Each stakeholder (see Section 2.2.4) is considered a user of the system.

1. Monitoring: The monitoring of EEG shall infer that data can be acquired from multiple locations, *i.e.* not only acquired locally. *Monitoring* refers to the acquisition of data from one or many NICUs in order to carry out analysis.

   (a) (POINT 1) The platform shall remove geographic limitations that affect the monitoring of EEG in the NICU. This is achieved through the ubiquitous access provided by the internet.

   (POINT 2) The platform shall remove geographic limitations affecting expert analysis of EEG. An expert capable of connecting to the internet shall have access to all data available on the platform. Furthermore he/she shall be able to upload patient data directly for analysis. The limiting factor shall primarily be internet connectivity. (Secondary factors include bandwidth)

   (b) Experts shall have access to patient data acquired from one or many NICUs.

   (c) The system shall be capable of monitoring multiple patient data feeds from an individual NICU simultaneously.

   (d) No stakeholder shall be able to interact with software responsible for local data acquisition in the NICU. This is a "*separation of concerns*" and intended to prevent any unintended consequences.

   (e) All data acquired from the NICU shall be anonymised to prevent the disclosure of sensitive patient details. This is to prevent any breach of NICU policy, or violation of patients rights. Similarly as above, this is a "*separation of concerns*", thereby preventing unintended consequences.

(f) The system shall be capable of acquiring patient data in real time (bandwidth dependent).

(g) The expert shall be able to access the platform securely. They will only be able to review data for which they are authorised.

2. Analysis: this can refer to either the interpretation of data by an expert through visualisation, or processing of data by the system.

   (a) The expert shall be provided with a viewer to analyse patient data available on the system. The expert shall also be able to view details about the data (file size, number of channels, sampling rate, frequency etc.), upload data manually (in an accepted format) and download data directly to his/her local machine.

   (b) Multiple experts shall be able to analyse a data feed simultaneously.

   (c) EEG monitoring machines in the NICU allow for the capacity to directly annotate data on the monitor. The expert shall be capable of annotating data in the viewer. This is both to support quantification of expert knowledge and encourage increased collaboration between experts.

   (d) The expert shall be capable of analysing data in real time (bandwidth dependent).

3. Limited Expertise:

   (a) All users should experience a benefit or gain by adopting the system. For the NICU, this is achieved via access to expertise through remote monitoring. For the expert, it is achieved through increased utilisation of their time and increased efficiency in the analysis of data (remote monitoring, automated assistance). For the patient, it is achieved through increased availability of treatment and a reduced time to diagnosis.

   (b) The system should increase the relevance of analysts' expertise through the derivation of alternate formats from the original data feed where possible.

   (c) Accessibility: Increased accessibility to experts also increases the relevance of that expertise.

   (d) Collaboration: The system shall provide experts with a means of communicating to one another directly that mimics the localised work environment (Analysis of data on screen through annotations).

   (e) The system shall reduce an expert's workload through the provisioning of automated solutions. *e.g.* seizure and threshold detection algorithms, event notification, automated workflows.

(f) Annotated data reflecting the quantification of expert knowledge embodies an excellent training resource for both experts and the NICU.

4. Infrastructure: The technical infrastructure currently supporting the NICU is designed and optimised for a local workflow. It is postulated that an infrastructure developed on top of an architectural design that supports both future and evolving requirements would be better suited to the NICU. Limited expert knowledge, and the technology to overcome geographic constraints, have resulted in a localised workflow being adopted in the NICU. The platform architectural design shall provide the means by which a number of these issues can be mitigated.

   (a) Current workflows should not be impacted. The platform architecture shall support the current workflow of the expert and the NICU as much as possible. This is to encourage adoption of the system and reduce the impact experienced by users. This policy should be enforced as long it does not negatively impact the stakeholders.

   (b) The system shall avail of both distributed and scalable resources. This provides the NICU with a greater level of redundancy. The NICU shall have access to a pool of experts registered with the system and, in turn, this pool of experts will have access to the NICU's patient data.

   (c) The system shall enable concepts such as "economies of scale" to be applied. Thus, issues/tasks that would be prohibitively expensive for a smaller number of NICUs to address can be made affordable. This is necessary to improve our overall understanding of the field and is one of the major benefits of interconnecting stakeholders at scale.

   (d) The platform shall provide access to more advanced technological solutions than are available at a local level.

   (e) The platform shall provide access to functionality for the further processing of data (Agent Framework, Stream computing).

   (f) The system shall provide access to scalable just-in-time third-party services (ScrutiniseIT).

   (g) All stakeholders shall be confident in the integrity of the system. Monitoring relies on sensors for the acquisition of data, a transfer protocol for the transmission of data and a system for storing and serving the data.

   (h) The system shall store all data acquired.

This includes checks being implemented by the acquisition and transmission process, including the system storage and recall of data.

System Requirements

Three components that are considered necessary for the system to fulfill the above requirements are:

- (Data source) Client side application (CSA) - For acquiring patient data and transferring it to a centralised location.
- (Middleware) Server side application (SSA) - For registering experts and NICUs, enabling patient data to be acquired and access experts.
- (Process) Web-based viewer (WBV) - For experts to analyse patient data resulting in treatment (This comes from requirement 4a).

1a
1. EEG data acquired in the NICU shall be transferred to the SSA using HTTP requests.
2. Experts and NICUs shall be registered with the system prior to being able to access it.
3. An SSA shall have a publicly accessible web portal.
4. The only technical requirements for an expert to access the SSA are internet connectivity and a web browser.
5. The only technical requirements for the CSA to upload data to the SSA are internet connectivity and port 80 to be open.
6. Each expert shall have a unique username and password combination. The expert will use this to gain entry to the SSA.
7. Upon successful login, the expert shall be able to view a list of data files available on the SSA.
8. An expert shall be capable of uploading data to the SSA directly.

1b
1. The CSA shall be designed to acquire data locally and transmit it to remote locations.
2. The CSA's installation shall be required to enable real-time monitoring functionality.
3. The CSA shall run on a machine locally in the NICU. It shall have access to patient data as it is being recorded by EEG-monitoring machines.

4. The CSA shall achieve this by monitoring a file directory where data is being stored/created.
5. The CSA shall upload data (using predefined time segments) to the system as it arrives with a standard transfer protocol.
6. The SSA shall have a port open and acknowledge receipt of data from registered CSAs.
7. The SSA shall make the acquired data available to experts for remote analysis.

1c
1. Each CSA shall be associated with a single SSA.
2. The CSA shall be capable of monitoring, and uploading, multiple data feeds simultaneously.

1d
1. The CSA shall not interfere with the machine acquiring the data.
2. The CSA shall be installed on an alternate machine to the acquisition machine and have access to the local network.
3. The CSA shall not be capable of interacting with software running on the acquisition machine.
4. The CSA shall have read-only access to the directory where data is stored by the acquisition machine.

1e
1. The CSA shall not transmit confidential patient data.
2. The CSA shall have read-only access to the directory where data is stored by the acquisition machine.
3. The CSA shall detect when new data files are created.
4. The CSA shall detect when existing data files are modified.
5. The CSA shall read any newly available data.
6. The CSA shall package the data omitting all patient sensitive details.
7. The CSA shall begin uploading and transfer data to the SSA in pre-defined chunks.

4h
1. The CSA should employ appropriate security measures when transmitting data.
2. The SSA should enforce fine-grained access control.
3. The SSA should employ appropriate security measures when accepting/transmitting data.

Analysis

2a (a) The WBV shall visualise EEG data within a web browser.

(b) The WBV shall mimic that of a traditional viewer for EEG.

(c) The WBV shall support the features that an expert has become used to in the NICU.

(d) WBV features shall include: the ability to apply montages, and filters, to data, the ability to scale data feeds and the ability to select/deselect different feeds for viewing.

2b (a) The SSA shall support multiple users. Each user shall have a unique ID.

(b) Each data file available on the system shall have a unique ID.

(c) Multiple users shall have access to data feeds simultaneously.

(d) The SSA shall support synchronous requests for the same data feed.

2c (a) While viewing data in the WBV, an expert shall be able to create an annotation.

(b) The annotation shall be associated with data channels on the y-axis, and a start and end time on the x-axis.

(c) Multiple types of annotations can be supported.

2d (a) The transfer of EEG data shall not result in a delay in analysis.

Infrastructure that supports the solution:

4a (a) Currently an expert accesses EEG data from a local machine in the NICU. EEG data is visualised as a number of time series data feeds, drawn in a left to right fashion. Each data feed corresponds to an electrode (or combination of electrodes - depending on the montage).

(b) The system shall have a WBV to enable users to visualise TS data in a similar manner online.

4g (a) Integrity checks shall be undertaken for each stage of data workflow.

(b) The CSA should check to ensure the integrity of the data file is correct after reading.

(c) The transmission process should include a means to verify the data received by the SSA, is that which was sent.

(d) The SSA shall ensure integrity of all data preserved.

(e) The WBV shall ensure the data served by the system is identical to that received.

4h (a) The system's data module should provide a read-only, open and transparent interface for other modules to access data. (Infrastructure)

4c (a) The system shall employ a standardised workflow for data collection procedures.

4d (a) The system shall notify all interested parties of the creation/deletion/extension of data.

(b) The system should support long-term storage of data.

4f (a) The system should support scalable services.

Non-Functional requirements

1. The transfer of EEG data should not delay analysis being undertaken as it is a time-critical process that can impact patient outcome. (Product requirement)

2. Encouraging trust: potential points of failure within the system should be identified and a preemptive response implemented to ensure the integrity of the system is maintained. (Product requirement)

3. Neurophysiologist analysis of live data feeds: The system shall be capable of real-time acquisition of data to meet user expectations. Failure to do so would result in reduced adoption of the system. (Organisational requirement)

Bibliography

[1] EMC. Digital universe study. Technical report, EMC, April 2014.

[2] Wikimedia Commons. Electrode locations of international 10-20 system for eeg (electroencephalography) recording, accessed on 08-08-14. "http://commons.wikimedia.org/wiki/File:21_electrodes_of_International_10-20_system_for_EEG.svg", 2010.

[3] Kiri L Wagstaff. Machine learning that matters. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 529–536, 2012.

[4] F. De Turck, J. Decruyenaere, P. Thysebaert, S. Van Hoecke, B. Volckaert, C. Danneels, K. Colpaert, and G. De Moor. Design of a flexible platform for execution of medical decision support agents in the intensive care unit. *Computers in Biology and Medicine*, 37:97–112, 2007.

[5] Dimitris A. Kalogeropoulos, Ewart R. Carson, and Paul O Collinson. Towards knowledge-based systems in clinical practice: Development of an integrated clinical information and knowledge management support system. *Computer Methods and Programs in Biomedicine*, 72:65–80, 2003.

[6] Raymond Lister, George Bryan, and Mark Trac. The e-babies project: Integrated data monitoring and decision making in neo-natal intensive care. In *European Conference of Information Systems*, 2000.

[7] Katharina Morik, Michael Imboff, Peter Brockhausen, Thorsten Joachims, and Ursula Gather. Knowledge discovery and knowledge validation in intensive care. *Artificial Intelligence in Medicine*, 19(3):225 – 249, 2000. Knowledge-based Information Management in Intensive Care and Anaesthesia.

[8] Thomas Guyet, Catherine Garbay, and Michel Dojat. Knowledge construction from time series data using a collaborative exploration system. *J. of Biomedical Informatics*, 40:672–687, December 2007.

[9] Sheng-Tun Li and Shu-Ching Kuo. Knowledge discovery in financial investment for forecasting and trading strategy through wavelet-based som networks. *Expert Systems with Applications*, 34(2):935 – 951, 2008.

[10] Steven Willmott, Jonathan Dale, Bernard Burg, Patricia Charlton, and Paul O'Brien. Agentcities: A worldwide open agent network. *AgentLink Newsletter*, 8:13–15, November 2001.

[11] Emanuela Merelli, Giuliano Armano, Nicola Cannata, Flavio Corradini, Mark d'Inverno, Andreas Doms, Phillip Lord, Andrew Martin, Luciano Milanesi, Steffen Mller, Michael Schroeder, and Michael Luck. Agents in bioinformatics, computational and systems biology. *Briefings in Bioinformatics*, 8(1):45–59, 2006.

[12] Julien Balter, Annick Labarre-Vila, Danielle Zibelin, and Catherine Garbay. A knowledge-driven agent-centred framework for data mining in emg. *Comptes Rendus Biologies*, 325(4):375 – 382, 2002.

[13] A Aguilera, E Herrera, and A Subero. Medical coordination work based in agents. *Biomedical Engineering*, (Isbme):122–126, 2008.

[14] Monika R Henzinger Prabhakar Raghavan. Computing on data streams. In *External Memory Algorithms: DIMACS Workshop External Memory and Visualization, May 20-22, 1998*, volume 50, page 107. American Mathematical Soc., 1999.

[15] Stream computing, accessed on 13/08/2015. `http://www-01.ibm.com/software/data/infosphere/stream-computing/`.

[16] Daniel J Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB JournalThe International Journal on Very Large Data Bases*, 12(2):120–139, 2003.

[17] Daby Sow, Alain Biem, Marion Blount, Maria Ebling, and Olivier Verscheure. Body sensor data processing using stream computing. In *Proceedings of the international conference on Multimedia information retrieval*, pages 449–458. ACM, 2010.

[18] Carolyn Mcgregor, Daby Sow, Andrew James, Marion Blount, Maria Ebling, J Mikael Eklund, and Kathleen Smith. Collaborative research on an intensive care decision support system utilizing physiological data streams. In *Artificial Intelligence*, pages 1124–1126, 2009.

[19] Marion Blount, Carolyn Mcgregor, Maria R. Ebling, J. Mikael Eklund, Andrew G. James, Nathan Percival, Kathleen P. Smith, and Daby Sow. Real-time analysis for intensive care. *IEEE Engineering in Medicine and Biology Magazine*, 2010.

[20] Alejandro Buchmann and Boris Koldehofe. Complex event processing. *it-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 51(5):241–242, 2009.

[21] Fusheng Wang, Shaorong Liu, Peiya Liu, and Yijian Bai. Bridging physical and virtual worlds: Complex event processing for rfid data streams. In Yannis Ioannidis, MarcH. Scholl, JoachimW. Schmidt, Florian Matthes, Mike Hatzopoulos, Klemens Boehm, Alfons Kemper, Torsten Grust, and Christian Boehm, editors, *Advances in Database Technology - EDBT 2006*, volume 3896 of *Lecture Notes in Computer Science*, pages 588–607. Springer Berlin Heidelberg, 2006.

[22] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012.

[23] Charles D. Ray, Reginald G. Bickford, W. Grey Walter, and Antoine Remond. Experiences with telemetry of biomedical data by telephone, cable and satellite: Domestic and international. *Medical Electronics and Biological Engineering*, 1965.

[24] Donald R. Bennett and Reed M. Gardner. A model for the telephone transmission of six-channel electroencephalograms. *Electroencephalography and Clinical Neurophysiology*, 29:404–408, 1970.

[25] Reed M. Gardiner, Donald R. Bennet, and Richard B. Vorce. Eight channel data set for clinical EEG transmission over dial-up telephone network. *IEEE Transactions on Biomedical Engineering*, 1974.

[26] F. Vaz, O. Pacheco, and A.M. da Silva. A telemedicine application for eeg signal transmission. In *Engineering in Medicine and Biology Society, 1991. Vol.13: 1991., Proceedings of the Annual International Conference of the IEEE*, pages 466 –467, oct-3 nov 1991.

[27] M. L. Bykhovskii and É. M. Krishchyan. Transmission of medical information through telephone lines. *Biomedical Engineering*, 2:307–313, November 1968.

[28] André J.W. van der Kouwe and Richard C. Burgess. Neurointensive care unit system for continuous electrophysiological monitoring with remote web-based review. *IEEE Transactions on Information Technology in Biomedicine*, 7(2), June 2003.

[29] V. Nenov and J. Klopp. Remote analysis of physiological data from neurosurgical ICU patients. *Journal of the American Medical Informatics Association*, 1996.

[30] L. Grandinetti, D. Conforti, and L. De Luca. CAMD and TeleEEG: Software tools for telemedicine applications. *High-Performance Computing and Networking*, 1401, 1998.

[31] S. Tsuji, N. Akamatsu, Y. Murai, S. Tobimatsu, M. Kato, E.C. Jacobs, and H.O. Luders. Remote diagnosis of intractable epilepsy by bidirectional telemedicine system between Japan and USA. *Electroencephalography and Clinical Neurophysiology*, 103(1):60–60, July 1997.

[32] Jim Cameron David Holder and Colin Binnie. Tele-eeg in epilepsy: review and initial experience with software to enable eeg review over a telephone link. Technical report, University College London, 2003.

[33] D.S. Holder, R.H. Bayford, J. Fritschy, O. Gilad, H. Kaube, and C.D. Binnie. Development of generic software for analysis, archiving & internet dissemination of brain and systems physiological data. Technical report, University College London, 2004.

[34] J. Fritschy, M. De Lucia, and D.S. Holder. Web EEG reader for remote reporting and automatic detection of normal and abnormal patterns in EEG. *Clinical Neurophysiology*, 117, 2006.

[35] Carolyn McGregor, G. Bryan, J. Curry, and M. Tracy. The e-baby data warehouse: a case study. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2002.

[36] Carolyn McGregor, Jennifer Heath, and Ming Wei. A web services based framework for the transmission of physiological data for local and remote neonatal intensive care. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service on e-Technology, e-Commerce and e-Service (EEE'05)*, Hong Kong, China, 2005.

[37] Carolyn McGregor, Bruce Kneale, and Mark Tracy. On-demand virtual neonatal intensive care units supporting rural, remote and urban healthcare with Bush Babies Broadband. *Network and Computer Applications*, 103(1):60–60, July 1997.

[38] DL Hudson and ME Cohen. Use of intelligent agents in the diagnosis of cardiac disorders. *Computers in Cardiology*, 29:633–636, 2002.

[39] Chang-Shing Lee and Mei-Hui Wang. Ontology-based intelligent healthcare agent and its application to respiratory waveform recognition. *Expert Systems with Applications*, 33:606–619, 2007.

[40] Darren Foster and Carolyn McGregor. Design of an agent server for neonatal analysis and trend detection. *International Transactions on Systems Science and Applications*, 1:27–34, 2006.

[41] Horacio Gonzlez-Vlez, Mariola Mier, Margarida Juli-Sap, Theodoros N. Arvanitis, Juan M. Garca-Gmez, Montserrat Robles, Paul H. Lewis, Srinandan Dasmahapatra, David Dupplaw, Andrew Peet, Carles Ars, Bernardo Celda, Sabine Huffel, and Mag Lluch-Ariet. HealthAgents: distributed multi-agent brain tumor diagnosis and prognosis. *Applied Intelligence*, 30, 2009.

[42] Darren Foster, Carolyn McGregor, and Samir El-Masri. A survey of agent-based intelligent decision support systems to support clinical management and research. In *Multi-Agent Systems for Medicine, Computational Biology, and Bioinformatics*, 2005.

[43] Florence Duchne, Catherine Garbay, and Vincent Rialle. Learning recurrent behaviors from heterogeneous multivariate time-series. *Artificial Intelligence in Medicine*, 39(1):25 – 47, 2007.

[44] Nicholas R Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[45] Esmaeil Hadavandi, Hassan Shavandi, and Arash Ghanbari. Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. *Know.-Based Syst.*, 23(8):800–808, December 2010.

[46] Roberto A. Flores-Mendez. Towards a standardization of multi-agent system framework. *ACM Crossroads*, 5:18–24, 1999.

[47] Barbara Hayes-Roth, Serdar Uckun, Jan Eric Larsson, David Gaba, Juliana Barr, and Jane Chien. Guardian: A prototype intelligent agent for intensive-care monitoring. *Artificial Intelligence in Medicine*, 4:165–185, 1992.

[48] Davide Brugali and Katia Sycara. Towards agent oriented application frameworks. *ACM Computing Surveys*, 32, 2000.

[49] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE: A FIPA-compliant agent framework. In *Proceedings of the Fourth International Conference on the Practical Application Intelligent Agents and Multi-agent Technology (PAAM'99)*, pages 97–108, London, UK, April 1999.

[50] The foundation for intelligent physical agents, accessed on 07-08-15. http://www.fipa.org/.

[51] David Morley and Karen Myers. The spark agent framework. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pages 714–721, Washington, DC, USA, 2004. IEEE Computer Society.

[52] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov.*, 7(4):349–371, October 2003.

[53] M Lanska and DJ Lanska. Neonatal seizures in the united states: results of the national hospital discharge survey, 1980–1991. *Neuroepidemiology*, 15(3):117–125, 1996.

[54] Volpe JJ. Neonatal seizures. in: Neurology of the newborn. 4th edition:178–214, 2001.

[55] Hasan Tekgul, Kimberlee Gauvreau, Janet Soul, Lauren Murphy, Richard Robertson, Jane Stewart, Joseph Volpe, Blaise Bourgeois, and Adré J du Plessis. The current etiologic profile and neurodevelopmental outcome of seizures in term newborn infants. *Pediatrics*, 117(4):1270–1280, 2006.

[56] Mark S Scher, Kosaburo Aso, Marquita E Beggarly, Marie Y Hamid, Doris A Steppe, and Michael J Painter. Electrographic seizures in preterm and full-term neonates: clinical correlates, associated brain lesions, and risk for neurologic sequelae. *Pediatrics*, 91(1):128–134, 1993.

[57] D. M. Murray, G. B. Boylan, C. A. Ryan, B. P. Murphy, and S. Connolly. Defining the gap between electrographic seizure burden, clinical expression and staff recognition of neonatal seizures. *Archives of Disease in Childhood - Child Fetal Neonatal Edition*, 93(3):187–191, 2008.

[58] R.M. Pressler, G.B. Boylan, M. Morton, C.D. Binnie, and J.M. Rennie. Early serial EEG in hypoxic ischaemic encephalopathy. *Clinical Neurophysiology*, 112(1):31–37, 2001.

[59] A. J. Gunn. Cerebral hypothermia for prevention of brain injury following perinatal asphyxia. *Current Opinion in Pediatrics*, 12(2):111–115, April 2000.

[60] M. Fitzsimons, L. Ronan, K. Murphy, G. Browne, S. Connolly, J. McMenamin, and N. Delanty. Customer needs, expectations, and satisfaction with clinical neurophysiology services in Ireland: a case for tele-neurophysiology development. *Irish Medical Journal*, 97(7):208–211, July 2004.

[61] L. Ronan, K. Murphy, G. Browne, S. Connolly, J. McMenamin, B. Lynch, N. Delanty, and M. Fitzsimons. Needs analysis for tele-neurophysiology in the Irish north-western health board. *Irish Medical Journal*, 97(2):46–49, July 2004.

[62] J. Jenkins, F. Alderdice, and E. McCall. Improvement in neonatal intensive care in northern ireland through sharing of audit data. In *Quality and Safety in Health Care*, volume 3, pages 202–206, June 2005.

[63] Lee Jong Wook. New who report calls for a new and innovative approach to health systems research. http://www.who.int/mediacentre/news/releases/2004/pr78/en/, 2004.

[64] G. Boylan, L. Burgoyne, C. Moore, B. O'Flaherty, and J. Rennie. An international survey of EEG use in the neonatal intensive care unit. *Acta Paediatrica*, 99:1150–1155, August 2010.

[65] Tak chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164 – 181, 2011.

[66] Adam Stein, Mark A Musen, and Yuval Shahar. Knowledge acquisition for temporal abstraction. In *Proceedings of the AMIA Annual Fall Symposium*, page 204. American Medical Informatics Association, 1996.

[67] Yuval Shahar and Mark A Musen. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*, 8(3):267 – 298, 1996. Temporal Reasoning in Medicine.

[68] Yuval Shahar. *A Knowledge-Based Method for Temporal Abstraction of Clinical Data*. PhD thesis, Stanford University, October 1994.

[69] Catherine C. Marshall. Annotation: from paper books to the digital library. In *Proceedings of the second ACM international conference on Digital libraries*, DL '97, pages 131–140. ACM, 1997.

[70] Ricardo Kawase, Eelco Herder, and Wolfgang Nejdl. A comparison of paper-based and online annotations in the workplace. In Ulrike Cress, Vania Dimitrova, and Marcus Specht, editors, *Learning in the Synergy of Multiple Disciplines*, volume 5794

of *Lecture Notes in Computer Science*, pages 240–253. Springer Berlin / Heidelberg, 2009.

[71] Sandra Bringay, Catherine Barry, and Jean Charlet. Annotations for the collaboration of the health professionals. In *American Medical Informatic Associations Annual Symposium Proceedings*, pages 91–95, 2006.

[72] Nathalie Bricon-Souf, Sandra Bringay, Saliha Hamek, Francoise Anceaux, Catherine Barry, and Jean Charlet. Informal notes to support the asynchronous collaborative activities. *International Journal of Medical Informatics*, 76:342–348, Dec 2007.

[73] Yves Keraron, Alain Bernard, and Bruno Bachimont. Annotations to improve the using and the updating of digital technical publications. *Research in Engineering Design*, 20:157–170, 2009.

[74] ILIA A. OVSIANNIKOV, MICHAEL A. ARBIB, and THOMAS H. MCNEILL. Annotation technology. *International Journal of Human-Computer Studies*, 50(4):329 – 362, 1999.

[75] William B. S. Pressly Jr. Tspad: a tablet-pc based application for annotation and collaboration on time series data. In *ACM Southeast Regional Conference*, pages 166–171, 2008.

[76] Natus Medical Incorporated. Olympic brainz monitor. Pdf from online catalogue, august 2010.

[77] Likert scale wikipedia entry. `http://en.wikipedia.org/wiki/Likert_scale`. Accessed: 16/12/2014.

[78] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[79] Microsoft windows, accessed on 08-01-15. "http://windows.microsoft.com/en-us/windows/home".

[80] Java, accessed on 26-01-15. https://www.oracle.com/java/index.html.

[81] Roy T. Fielding. Representational state transfer (rest), accessed on 07-01-15. http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm, 2000.

[82] Jetty, accessed on 07-01-15. http://eclipse.org/jetty/about.php.

[83] Haitham S. Hamza. Separation of concerns for evolving systems: a stability-driven approach. *SIGSOFT Softw. Eng. Notes*, 30:1–5, May 2005.

[84] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[85] Object-relational mapping, accessed on 13/07/2015. `https://en.wikipedia.org/wiki/Object-relational_mapping`.

[86] Java persistence api. `https://en.wikipedia.org/wiki/Java_Persistence_API`. Accessed: 13/07/2015.

[87] Hypersql, accessed on 07-01-15. http://hsqldb.org/.

[88] B. Kemp, A. Varri, A.C. Rosa, K.D. Nielsen, and J. Gade. A simple format for exchange of digitized polygraphic recordings. *Electroencephalography and Clinical Neurophysiology*, 82:391–393, 1992.

[89] Gunther Hellmann, Markus Kuhn, Markus Prosch, and Manfred Spreng. Extensible biosignal (EBS) file format: simple method for EEG data exchange. *Electroencephalography and Clinical Neurophysiology*, 1996.

[90] Silverlight, accessed on 26-01-15. http://www.microsoft.com/silverlight/.

[91] Adobe flash player, accessed on 07-01-15. http://www.adobe.com/software/flash/about/.

[92] Philip Healy. Critical care telemonitoring: Babylink, 2008.

[93] Information technology – security techniques – information security management systems – requirements using iso/iec 27001, accessed on 07-01-15. http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54534, 2013.

[94] Health informatics – information security management in health using iso/iec 27002, accessed on 07-01-15. http://www.iso.org/iso/catalogue_detail?csnumber=41298, 2008.

[95] Almut Herzog and Nahid Shahmehri. Using the Java sandbox for resource control. In *Proceedings of the 7th Nordic Workshop on Secure IT Systems (NordSec)*, pages 135–147, November 2002.

[96] Scott Oaks. *Java Security (2nd Edition)*. O'Reilly, May 2002.

[97] Almut Herzog and Nahid Shahmehri. Performance of the Java security manager. *Computers & Security*, 24:192–207, May 2005.

[98] Neonatal brain research group (nbrg), accessed on 27-01-15. http://www.nbrg.ie/898p911ded0#&panel1-1.

[99] European data format, accessed on 07-01-15. http://www.edfplus.info/.

[100] Ce marking - basics and faqs, accessed on 07-01-15. http://ec.europa.eu/enterprise/policies/single-market-goods/cemarking/about-ce-marking/index_en.htm.

[101] Volume shadow copy service, accessed on 27-01-15. https://technet.microsoft.com/en-us/library/ee923636%28v=ws.10%29.aspx.

[102] Hobocopy, accessed on 07-01-15. http://candera.github.io/hobocopy/.

[103] Understanding shims, accessed on 27-01-15. https://technet.microsoft.com/en-us/library/dd837644%28v=ws.10%29.aspx.

[104] Nicoletone neurodiagnostic system, accessed on 07-01-15. http://www.natus.com/index.cfm?page=products_1&crid=693.

[105] L. Hellstrom-Westas, I. Rosen, L.S. de Vries, and G. Greisen. Amplitude-integrated EEG classification and interpretation in preterm and term infants. *NeoReviews*, 7(2):76–87, April 2006.

[106] Matlab, accessed on 07-01-15. http://uk.mathworks.com/products/matlab/.

[107] Ardalan Aarabi, Reinhard Grebe, and Fabrice Wallois. A multistage knowledge-based system for EEG seizure detection in newborn infants. Clinical Neurophysiology 1 December 2007 (volume 118 issue 12 Pages 2781-2797).

[108] A. Subasi. Epileptic seizure detection using dynamic wavelet network. Expert Systems with Applications, vol. 29, pp 343355, 2005.

[109] Stephen Faul, Geraldine Boylan, Sean Connolly, Liam Marnane, and Gordon Lightbody. An evaluation of automated neonatal seizure detection methods. *Clinical Neurophysiology*, 116:1533–1541, 2005.

[110] W. Deburchgraeve, P.J. Cherian, M. De Vos, R.M. Swarte, J.H. Blok, G.H. Visser, P. Govaert, and S. Van Huffel. Automated neonatal seizure detection mimicking a human observer reading eeg. *Clinical Neurophysiology*, 119(11):2447 – 2454, 2008.

[111] J. Zhang J. Gotman, D. Flanagan and B. Rosenblatt. Automatic seizure detection in the newborn: methods and initial evaluation. *Electroencephalography and clinical Neurophysiology*, 103:356–362, 1997.

[112] J. Gotman, D. Flanagan, B. Rosenblatt, A. Bye, and E.M. Mizrahi. Evaluation of an automatic seizure detection method for the newborn eeg. *Electroencephalography and clinical Neurophysiology*, 103:363–369, 1997.

[113] J. Gotman. Automatic seizure detection: improvements and evaluation. *Electroencephalography and clinical Neurophysiology*, 76:317–324, 1990.

[114] J. Gotman, J.R Ives, and P. Gloor. Automatic recognition of inter-ictal epileptic activity in prolonged eeg recordings. *Electroencephalography and Clinical Neurophysiology*, 46:510–520, 1979.

[115] Nemo: Treatment of neonatal seizures with medication off-patent: evaluation of efficacy and safety of bumetanide, accessed on 26-01-15. http://www.nemo-europe.com/en/about-nemo.php.

[116] E. Finan, T. Bolger, and SM. Gormally. Modes of death in neonatal intensive care units. *The Irish Medical Journal*, 99:106–108, 2006.

[117] Ruairi D O'Reilly, David Power, Philip D Healy, John P Morrison, and Geraldine B Boylan. Scrutiniseit: A search-based approach to eeg seizure detection. In *eTELEMED 2013, The Fifth International Conference on eHealth, Telemedicine, and Social Medicine*, pages 310–313, Nice, France, March 2013.

[118] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, August 2008.

[119] W.F. Tichy. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.