Building a Context Rich Interface to Low Level Sensor Data

Jie Shi

Bachelor of Science in Computer Science

A Dissertation submitted in fulfilment of the

requirements for the degree of M.Sc. in

Computer Science

to the

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisor: Mark Roantree

July 2012

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Science is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ ID No.: 55138098 Date: September 10, 2012

Abstract

Sensor networks play an important role in our modern information society. These networks are used for a variety of activities in different domains, including traffic monitoring, environmental analysis, transport and personal health. In general, systems generate data in their own format with little or no associated semantics. As a result, data must be managed individually and significant human effort is required to analyze data and develop ad-hoc applications for different end-user requirements. The research presented here proposes a holistic and comprehensive approach to significantly reduce the human effort in analyzing networks of sensors. The goal is to facilitate any form of sensor network, enabling users to combine related semantics with sensor data, and facilitate the end-user transformation of data necessary to provide more complex query expressions, and thus meet the analytical requirements.

Acknowledgements

I would like to thank all those people who made this dissertation possible. In particular, I wish to express my sincere gratitude to my supervisor Dr. Mark Roantree for his patience, guidance, effort, encouragement and excellent advices throughout the MSc project. Without Mark, this dissertation would not have been possible.

Thanks also to Enterprise Ireland who supplied the funding for my research and to Dublin City University for the structures and support they provided.

Thanks to my colleagues from the Interoperable Systems Group for sharing their experiences and knowledge during the time of my study. Especially to Ken for helping me improve my writing skills and thrashing out various ideas.

Finally, I would like to express my deepest gratitude to my family for all their support and encouragement during my MSc study.

Contents

Chapter 1

Introduction

Sensor networks are increasingly used to monitor and support various processes and functions. The sensors provide a way to automate the monitoring process, as they perform simple and specific tasks. What these networks offer is the continuous generation of information. However, these networks generate vast amounts of information that is not structured and is very difficult to use. Due to the fact that there are no standard data storage methods or mechanism for query answering or knowledge extraction, and therefore, the sensed data must be handled on case by case basis, which requires considerable human effort to analyze data or develop ad-hoc introduction of the applications required for each situation. The sensors provide information that is treated, filtered, interpreted, and stored in a useful way of infrastructure for users. By providing an appropriate data management layer, the data can be controlled and thus, transformed into knowledge, providing input into all forms of decision making through an efficient query answering process.

1.1 Background Study

Information innovation has had a significant impact on our lives. For many of us, we interact with two very separate worlds: the real world in which we stay and the online world of the web. Web technology has changed the way we learn, work, and play in our real world.

The catalyst for this change is based on a new generation of cheap, reliable, and efficient sensor technologies where these technologies keep sensing diverse events in the real world. They include web-connected sensors and sensor systems of all types: flood gauges, air pollution monitors, stress gauges on bridges, mobile heart monitors, webcams, satellite-borne earth imaging devices and countless other sensors and sensor systems. In short, the combination of sensors, software, and the Internet will enable new types of information services across a wide range of sectors from health and the environment to education, retail, and entertainment.

A recent overview of the Sensor Web [15] highlighted the growth of sensor networks and described research in areas such as sensor development, toolkits and standards, security, ubiquitous sensing systems and wearable sensors. Many of these topics focused on bridging the physical-digital divide and discussed research into areas ranging from environmental monitoring, testing in large scale engine development, detection of hazardous gases in emergency disasters, and personal and wearable sensors. The deployment of sensors, as a means of automatic extraction of data from a wide variety of sources, is commonplace today. A recent survey [14] of applications and test environments using sensor technology provided details of sensor deployments in areas such as monitoring in the automotive industries, security, fire safety, environmental monitoring and health.

The Sensor Web aims to simplify access to sensor resources, similar what the World Wide Web does for accessing the documents. It relies on new information and communication standards for structuring sensor information and its exchange. The Open Geospatial Consortium (OGC)[25] sets up these new standards, which refers to web accessible sensor networks and archived sensor data that can be discovered, accessed and, where applicable, controlled using open standard protocols and interfaces (APIs). The OGC provides a framework for building exchange standards and service interfaces for accessing sensor data and contextual information, and thus, enables real time integration of heterogeneous

sensor data on the Internet. The research was undertaken through global collaborations of experts, often in the context of coordinated engineering activities (the OGC Web Service Initiatives) accelerates the development of interoperability standards and protocols. Within Open Geospatial Consortium (OGC) standards, the SANY (Sensors Anywhere) [15] project works towards 'plug and measure' in sensor networks for environmental monitoring. The architecture specifies all kinds of fixed and moving sensors, and it allows seamless sharing of information between sensor networks. SANY provides a quick and cost-efficient way to reuse data from sensor and data sources that are currently incompatible. Data sources include live sensor data and archived data in the databases.

Sensor networks are increasingly finding their way into our living environment, where they perform a variety of tasks like surveillance, safety or resource monitoring. Progress in standardization and communication protocols has made it possible to interact and exchange data in a generic fashion. Researchers expect that the sensor network performance will become more robust when information from multiple sources is integrated. In addition, they believe that sensor networks could become smarter for at least two reasons: sensors that produce highly reliable output can be used to provide guidance of other sensors within the network, and correlations among sensed events could lead the automatic propagation of semantic information across sensors.

Sensors and sensing technology are everywhere. People often deploy sensors to address issues like networking, calibration, sensor fusion and sensor event detection. The general trend is towards networking sensors into the Sensor Web, and there is another way of using them such as they can be used in small groupings on standalone devices that gather information, and report back live sensor readings. This wearable sensor technology [15] records sensor readings of a wearer's daily life, the focus of this work is on detecting events of interest to the wearer from a multi-sensor standalone device. In the field of personal health (pHealth) sensor networks, wearable sensors are used to demonstrate both levels of

the health performance and well-being. Discrete sensors are now commonly used to assess physiological responses during individual and team sports. Measuring heart rate and breathing to assess the physiological stress during sports activities is widely accepted within the sporting community. Wireless devices such as the wireless Polar heart rate monitor and the team breathing sensor is used in this study to monitor the performance of the participant during team sports. Relative heart rate can be estimated by treating and manipulating the output of sensors, and it is a common measure of intensity in any sport.

1.2 Motivation

Before coaches can introduce a new training system, it is necessary to validate this strategy through a large study. Traditionally in intermittent type sports, aerobic fitness is enhanced through endurance training. This type of training requires long time periods which could alternatively be spent on enhancing other essential sports specific skills. Recent research [17] and [4] have demonstrated that short sprint interval training lasting, typically 10 to 15 minutes a session, can convey the same aerobic fitness as these endurance type sessions. The challenge in performing the necessary tests using sensor devices is with managing the very high volumes of data that will be generated. Often, exercise physiologists will spend days analysing spreadsheets simply to arrive at basic calculations such as average heart rates over specific periods of time, or comparisons of heart rates for selected athletes over different training sessions. They lack the ability to do proper information analysis on these high volumes of data. In other words, there is a significant gap between high level user requirements and the data generated by sensors, and when this data is generated in high volumes, it provides a significant barrier to knowledge extraction. Thus, the motivation behind this research is to provide an information management layer that provides exercise physiologists with the ability for meaningful analysis through a flexible query interface.

1.3 Approach and Contribution

The challenge for us is to close the semantic gap between end user requirements and raw datasets. We do this by providing a number of processors and algorithms to transform the data to a format that could be queried using a high level query language. Therefore, the overall contribution of this research is to provide a mechanism which will automatically enrich and transform sensor data upto a point, and then through user interaction provide new levels of semantic enrichment.

As part of our approach, we aim to achieve these goals:

- A rule based paradigm with which we can associate sensor data with the context information in which these data has been generated, which is covered in Chapter 4.
- An efficient sensor outlier removal algorithm, which is provided in Chapter 5.
- A set of data functionalities or transformations to enable the end user express complex queries against low level sensor data, which is covered in Chapter 5.

The contribution presented in this dissertation is two-fold. Firstly, we provide a framework for the contextual enrichment of sensor data that allows the user to specify how to merge context and raw sensor data in a rule based approach. Secondly, we provide a set of data transformation primitives to extract levels of knowledge required for the more complex user queries. We evaluate our work using a series of queries and provide a detailed discussion of the results. In this evaluation, both user requirements (queries) and datasets are provided by the exercise physiologists.

1.4 Summary and Dissertation Structure

The structure of this dissertation is as follows: We continue in Chapter 2 by giving an analysis of related research in this field; in Chapter 3, a real world case study and a system overview is then provided; Chapter 4 presents the enrichment of enabled sensor data using a rule-based approach; in Chapter 5, we introduce our outlier removal algorithm together with

the data transformation primitives; in Chapter 6, we present our evaluation with a detailed analysis of experiments; and finally, we provide conclusions and discuss future work in Chapter 7.

Chapter 2

Related Research

The Related Research Chapter will mainly focus on four topics: Section 2.1 involves research on context modeling and management in designing a context-aware system architecture; Section 2.2 discusses the relevant technologies on Body Sensor Networks; Section 2.3 discusses research on the rule based contextual enrichment of raw sensor data; Section 2.4 discusses the research efforts on sensor outlier removal algorithms.

2.1 Context Modeling Architecture

Authors in [47] propose context management as a new approach to the design of context-aware systems in ubiquitous computing that combines personalization and contextualization. They present a framework that integrates user modeling and context modeling and this base framework is designed for context-aware applications. They developed a component-based architecture for hosting several components on different abstraction levels and providing functionalities like database access and knowledge exchange between these components. We can illustrate the four main layers of the framework:

1. **Sensor Layer**

   The Sensor Layer consists of tools that receive incoming data, perform cleaning and fusion of sensor values. It serves as an information collector, where each context-

adaptive system relies on a network of sensors placed in the physical environment and delivering an image of the current situation that the user is acting in.

2. **Semantic Layer**

   The Semantic Layer defines the context model of a context-adaptive system. The context model then captures the current situation that the user is acting in. Each context attribute on the Semantic Layer is connected with one or multiple sensors, and in turn sensors deliver information to one or more attributes, creating across linked network between sensors and attributes. Context attributes receive sensor values and map them onto semantically more enriched values.

3. **Control Layer**

   On the basis of the context model and data provided by the Semantic Layer, the Control Layer decides what actions should be taken if a particular condition in the model has been matched.

4. **Indicator/Actuator Layer**

   Finally, The Indicator/Actuator Layer handles the connection back to the domain by mapping the operations triggered in the Control Layer to real world actions.

These four components form the basis of a rule system that controls the desired behavior of the target application. This rule system is a set of hierarchically ordered rules, where each rule comprises a precondition part and an action part. If the precondition of a rule is matched, the rule is triggered and all related actions are executed. Preconditions consist of only boolean expressions while actions involve changes in context attribute values, and the selection of content on different devices. This research work gives us an overview of a functional layer architecture that supports sensor data management, context abstraction, and the control of actuator output.

This four layered context-aware framework is similar to our own design, where we also have the concepts such as the Sensor Layer where we enable sensors with affiliated tem-

plates, the Semantic Layer where we associate context with the raw sensor data, the Control Layer where we can apply necessary data transformations to enable complex queries.

Issues and limitations. This research work gives us an overview of a functional layer architecture that supports sensor data management, context abstraction, and the control of actuator output. However, one of the biggest drawbacks is that, the rule system they present is in a relative low level hence only very limited queries can be proposed and answered. In contrast, our system enables a wide range of queries over raw sensor data and transforms data if necessary to answer more complex queries.

The research in [40] also focuses on context modeling and context management. They suggest that an efficient context model is a key factor in designing context-aware services. They establish a context awareness enabling architecture which provides the user with a uniform way to access context information, thus hiding the complexity of context management. It also offers streamlined mechanisms for massive distribution and synchronization of personal repositories across multiple data sources. Under this structure, a component called the Context Broker answers context requests and serves as the access point to the context data. When a user requests context from the Context Broker, it decides and retrieves the context from the most appropriate context source among multiple sources providers. There is also an Inference Engine working under Context Broker, which infers additional and indirectly observable context information. It contributes to minimise user interactions with the pervasive computing system and plays the important role of estimating the activity of a person given some lower level sensor data. The raw data is tracked by the Sensor Manager, and eventually the Context Store retrieves context from the Sensor Manager (raw data) and the Inference Engine (contextual data), processes this data, stores it to the appropriate repositories and updates the context databases.

Their framework acts as an agent based architecture supporting context-aware computing

in intelligent spaces populated with intelligent systems that provide pervasive computing services to users. While our framework does not provide the user such services as live data collecting nor live querying, we store and retrieve the associated context information to and from the appropriate context repositories.

Issues and limitations. The biggest problem they experience is that the cost of communication and maintenance is sufficiently high in such large-scale distributed systems, and it lacks the flexibility to include any kind of domain specific ontologies on behalf of the end-users. Moreover, due to lack of a well-designed data aggregation/transformation layer, even for answering a simple set of queries defined by end users will result very complex contextual inputs. In contrast, our system architecture is quite context flexible and adaptive on any domain which benefits from our template driven approach, and we allow the end user to express complex query over sensor data with the help of a proper designed data transformation component.

2.1.1 Summary.

These research projects present a context management architecture, which is adequate for large scale ambient systems that have a strong relation to heterogeneous and distributed networks. Their architecture supports context refinement, uniform access to context data by users, as well as federation of context repositories. They offer a standardised mechanism for discovering and accessing sensor data which enables contextual information to be reused in potentially new and novel ways, and also enables contextual data to be utilized, providing a common, self-describing context ontology.

2.2 Relevant Technologies

In both [7] and [8], the authors examine how Sensor Web enablement services work in healthcare sensor networks. Their research follows the standards of the Sensor Web Enablement (SWE) from the Open Geospatial Consortium (OGC) [25] as a possible compo-

nent for the virtualization of sensors and to build a generic sensor network architecture with reusable components for various sensor networks. They offer a standardised protocol for discovering and accessing sensor data which enables data to be reused in potentially new and novel ways, and also enables sensor data to be enriched, providing a common, self-describing data format and access protocol. Both approaches enrich multiple sources of sensor data into a data model that represents different sensors and data as a series of observations. They focus on simple live sensor data from multiple sources, obtaining query results on the basis of a series of simple rules applied to the streams.

The research in [1] also focuses on personal health sensor networks. They are developing a wearable light device capable of measuring specific vital signs of the elderly, detecting falls and location, and communicating this information automatically in real-time. Their system has been designed with the aim of reducing the reliance on a central station, thus leading to minimizing power consumption by communicating only when it is strictly necessary. In contrast with [8], the full set of standards was not adopted since it proves inefficient when taking into account the sensor protocols used in their system. Also, their systems are only processing the sensors which are only in a Body Area Network, unlike the OGC Sensor Networks which are geographically distributed, measured and observed. Nevertheless, the concepts such as measurement and observation initiated by the OGC have been partly adopted by them to manage and extract data from a sensor device.

2.2.1 Summary.

Both approaches discuss the relevant technologies on Body Sensor Networks and follow the concepts and standards of the Sensor Web Enablement (SWE) which has also been adopted in our approach. However, they do not adopt the full set of standards in comparing with our standard OGC template which can be reused for various sensor networks.

2.3 Rule Based Contextual Enrichment

In [44], the authors process and query streams of raw data as it arrives from sensors. Their approach enriches the raw data into "semantic streams" and processes the streams as they are generated. The "semantic streams" permits the user to issue queries over semantic values directly, without addressing which data or operations are to be used. Their programming model consists of two fundamental elements: event streams and inference units. Event streams represent a set of events, where each event represents a real world object, such as a person or a car and has its own properties such as the time or location, speed, identity etc. Inference units are the processes that operate on event streams. They refer to the semantic information about the world from incoming events and either generate new event streams or add the information to existing events as new properties. So as a stream flows from sensors and through different inference units, its events acquire new semantic properties. In order to automatically compose sensors and inference units, they use a markup language to encode a logical description of how they fit together. Furthermore, each inference unit must be fully specified in terms of its input streams and output streams and any required relationships between them. The "semantic streams" defines a set of logical predicates that can be used to declare sensor and inference units as following:

Definition 2.1. *An example of logical predicates*

```
sensor( <sensor type>, <region> )
inference( <inference type>, <needs>, <creates> )
needs( <stream1>, <stream2>, ... )
creates( <stream1>, <stream2>, ... )
stream( <identifier> )
isa( <identifier>, <event type> )
property( <identifier>, <value>, <property name> )
```

The `sensor` predicate defines the type and location of each sensor. The `inference`, `needs`, and `creates` predicates describe an inference unit in terms of the event streams that it needs and creates. The `stream`, `isa`, and `property` predicates describe an event

stream and the type and properties of its events. A query is simply a first-order logic description of the event streams and properties desired by the user. For example, a simple query could be: `stream(X), isa(X,vehicle)`, which has a simple meaning that the query would be true if the event stream X is a vehicle.

Overall, there are two notable features in this research work. First, the declarative programming they proposed is easier to understand than low-level, distributed programming and allows users to query high level information from sensor networks. Second, the framework allows multiple users to task and retask the network concurrently, optimizing for reuse of services between applications and automatically resolving resource conflicts. Together, their framework allows non-technical users to quickly extract semantic information from raw sensor data using a set of logical predicates. Similar to this work, we developed a set of Event-Condition-Action rule templates for enriching contextual information, which also allows non-IT users to efficiently adding semantics to raw sensor data. Moreover, we use XQuery as our stardard query language to make all queries more flexible and manageble for the user.

Issues and limitations. There are several issues/limitations involved in this approach as listed below:

- Queries are at a relatively low level.
- No open query language is supported.
- Finally, this work is still theoretical and has yet to provide experiments or an indication of query performance.

In [12] and [11] , the authors present an approach to address contextual synthesis of sensor networks in the sports domain. The context synthesis is to generate new knowledge, as a result of a reasoning process applied to context information that is already present in their system. In their research, they utilize a context engine to retrieve context information on

mobile applications. The context engine provides mechanisms to retrieve, model, synthesize, and distribute context information in a distributed, mobile environment. They use the term *context query* as a request for context information. The context query uses an operation to produce the expected context information. Operators can be considered as functions that take an input or list of inputs, perform an operation, and produce an output. Furthermore, they split context queries into two categories: simple and complex context queries. These two query types are distinguished depending on whether they contain an operator or not. Simple context queries obtain context information directly from the repository, i.e. the database, without using operators; while the second category contains an operator, whose inputs determine the required context information. These are called complex context queries. The context engine contains a context mapping component that issues local and remote context queries. Communication between the context mapping component and the context repository is based on queries.

The authors define context synthesizing as a process of generating new knowledge. This context synthesis requires some operator based rules to drive the process. In their distributed context retrieval, context synthesis begins with an interpretation of a context query to retrieve a description and implementation of the operator. The basic context query is represented by an expression of operators, arguments, and a context quantifier. The description of the retrieved operator specifies the operation performed by this operator, the required input arguments, and the output returned by this operation. The output of the operation is then sent to the mobile application as a result of the context query.

In summary, they provide context synthesis to the user which offers two main advantages: firstly, an operator provides the user with a functional approach to context data simplifying context synthesis and programming of context-aware systems in general. Secondly, the context engine allow the user to apply operators dynamically based on description of input and output types. Similar to their work, our main research area is also based on analyzing

sensor data in the sports domain. However, they put their research effort in a distributed, mobile environment which has its own proprietary data format, while our aim is to provide a more stardard platform where the user has a high level, stardard query language to express queries.

Issues and limitations. Sensor data, generated from a single event, are kept in their proprietary raw format. Queries are developed as ad-hoc programs over the proprietary data format, but they are not SQL-like, and instead all object-oriented, containing context operators which perform synthesising operations. Context operators can in turn use other simpler operators to execute smaller tasks and to reuse existing functionality. The biggest problems involved in this approach are: rules are still at a relatively low level and this research does not support any open query language but requires precomposed built-in queries. Every time a query needs to be added or edited, there is the need for a system expert to apply the changes.

2.3.1 Summary.

Overall, the approaches presented lack the ability to handle complex queries over raw sensor data and rules for enriching data are at a relatively low and abstract level. Moreover, no open query language is supported among them. I will provide an Event-Condition-Action (ECA) like paradigm which enables users to specify context enrichment using open query language as XQuery, which will be shown in Chapter 4.

2.4 Sensor Outlier Removal

Palpanas et al. [30] have proposed an in-network approach for distributed online deviation detection for streaming data. Their interests is in finding those values that deviate significantly from the norm. Their detection mechanism can be used to identify faulty sensors, and to filter spurious reports from different sensors. However, this approach depends on the existence of more powerful and sophisticated sensors (high capacity sensors) to perform

the filtering process and to manage groups of low capacity sensors. Their more recent work [39] propose a general and flexible data distribution approximation framework that does not require a priori knowledge about the input distribution. They use an online filtering scheme for sensor networks. The goal is to identify, among all the sensor readings in a sliding window, those values that have very few near neighbors.

In the sensor network setting, they require that each sensor maintains a model for the distribution of values it generates. Since they are not interested in the entire history of the values produced by the sensors, it suffices to consider the values in a sliding window W of size N. Thus, T holds only the values in this sliding window, i.e., the N most recent values. At each point in time, they are interested in approximating the distribution of the data values within the sliding window. The techniques proposed operate efficiently in an online fashion. Moreover, they distribute the computation effort among the nodes in the network, thus better exploiting the available resources and cutting down on the communication and processing costs. Their approach initially estimates the sensor data distributions, then computes the density of the data space around each value, and therefore determines which values need to be cleaned. The key point is that every sensor keeps a sliding window of the historical data and estimates the data distribution to detect the outliers.

Our outlier removal method is similar in that it uses a sliding window W of size N to detect and calibrate the actual outlier. The differece is that our method sets the initial valid range/bound for the sliding window rather than keeping it on the entire historical data. In this way, less memory footprint is needed while running this method against large volume of sensor data.

Issues and limitations. Their approach initially estimates the sensor data distributions, computes the density of the data space around each value and therefore, determines which values need to be cleaned. The key point is every sensor keeps a sliding window of the his-

torical data and estimates the data distribution to detect the outliers. This method, however, consumes a lot of memory and may not find all outliers.

A similar approach for outlier detection in streaming data is described by Kenji et al. [21]. They focus on the issue of online unsupervised outlier detection. Their approach introduces SmartSifter as a probabilistic model to represent an underlying mechanism of data-generation. The model takes a hierarchical structure, and every time a datum is input, it employs an on-line learning algorithm. SmartSifter give a score to each datum on the basis of the learned model, measuring how large the model has changed after learning. Thus a higher score indicates a higher possibility that the datum is an outlier. In contrast to Palpanas's work [30], their method does not operate on sliding windows, but rather on the entire history of the data values, using an exponential forgetting factor for discounting the effect of the older values.

Their approach detects outlier based on a way of smart learning the data sources. In contrast, our methodology is based on parameterized variables, which is flexble to the user and adaptive to different sensors.

Issues and limitations. Unlike Palpanas's work [30], SmartSifter does not operate on sliding windows, but rather on the entire history of the data values, using an exponential forgetting factor for discounting the effect of the older values. Furthermore, the above approach is not geared towards a distributed environment, such as a sensor network.

2.4.1 Summary.

Both of the approaches need to compare all values with their near neighbors as well as the historical data. Our approach identifies candidate outliers only after the valid range is selected. It requires a small memory footprint and facilitates the calibration of actual outliers in a deterministic manner, detailed outlier removal algorithm will be listed in Chapter 5.

2.5 Conclusion

From the research discussed in this chapter, we can clearly see that there are many approaches for modeling, enriching, cleaning and querying raw sensor data, but due to the lack of a proper management layer, either queries are at a relatively low level or they do not support any standard query language. Section 2.1 looked at the ideas in designing a context-aware system architecture similar to that described in Chapter 3; Section 2.2 discussed the issues in adopting the concepts and standards of the Sensor Web Enablement which is also covered in Chapter 3; Section 2.3 described research issues on rule based contextual enrichment of raw sensor data which is discussed further in Chapter 4; Section 2.4 involved examining issues on sensor outlier removal algorithms which we will provide in Chapter 5. Overall, the approach presented in this research will provide a mechanism that uses a rule-based context enrichment ontology which supports open query languages such as XQuery, and moreover, provides an efficient sensor outlier removal algorithm to detect and calibrate all necessary outliers. The next chapter will provide an overview of how our system works.

Chapter 3

System Architecture

Chapter 3 illustrates the processes our system uses to structure, enrich, transform and query sensor data in real world scenario. In Section 3.1, a real world case study with user queries is presented; in Section 3.2, we give an overview of our approach in answering these real world queries; in Section 3.3, we describe a template driven approach to enable raw sensor data for basic queries; in Section 3.4, we describe the enrichment of enabled sensor data with the context information in which has been generated; and finally in Section 3.5, we discuss future knowledge and transformations we can extract from the sensed data.

3.1 Real World Case Study

This research is a result of a collaboration with exercise physiologists. The goal of the exercise physiologists is to collect data using a set of sensors and ensure that participants complete the whole process of the activity. The challenge for us was to allow the user to query raw datasets in a flexible manner. In order to achieve the goal, we need to provide a number of processors and algorithms to transform the data to a format that could be queried using a high level query language, which will be introduced in the following sections.

3.1.1 HIIT Study

The treatment of individuals with CardioVascular Disease (CVD) has evolved considerably in the last 50 years [6]. Individuals with cardiovascular disease are now encouraged to undertake a rehabilitation programme that offers a multi-faceted and multidisciplinary approach to optimize cardiovascular risk reduction, promote adoption and adherence to healthy behaviours, enhance emotional well-being, reduce disability, and encourage an active lifestyle [5]. Traditionally, individuals with CVD are encouraged to undertake continuous, moderate to vigorous intensity exercise as part of their cardiac rehabilitation programme. A growing body of evidence has shown that High-Intensity Interval Training (HIIT) can be an effective alternative to traditional continuous exercise interventions. High-intensity interval training involves a series of repeated bouts of exercise performed between 70 - 100% of maximal effort, and alternated with periods of active or passive recovery. The duration of the high-intensity intervals can range from 10 seconds to 5 minutes. When compared to continuous endurance exercise, interval training has been shown to elicit similar or even superior physiological adaptations, in both healthy and diseased populations.

The majority of studies that have examined the effect of interval training have used intervals of 3 minutes in duration. Our collaborations examined the cardiovascular responses to short duration High-Intensity Interval Training (HIIT) in individuals with documented cardiovascular disease (CVD). Twenty-five men and women undertook a HIIT session that involved a 10 minute warm-up, and 3 blocks of 8 high-intensity intervals, followed by a 5 minute cool down. Each interval consisted of 45 seconds of treadmill exercise at 90 - 100% of their maximal effort, interspersed with 15 seconds of passive recovery, with details illustrates in Figure 3.1. Heart rate was continuously monitored during each HIIT session using a heart rate telemetry system (Polar Team2 Pro) [32]. To date, the majority of studies evaluating the cardiovascular responses to HIIT have provided summary statistics, such as average HR, maximal HR and minimal HR. The goal of the exercise physiologists is ensure that participants complete the whole process of the activity. This is important because

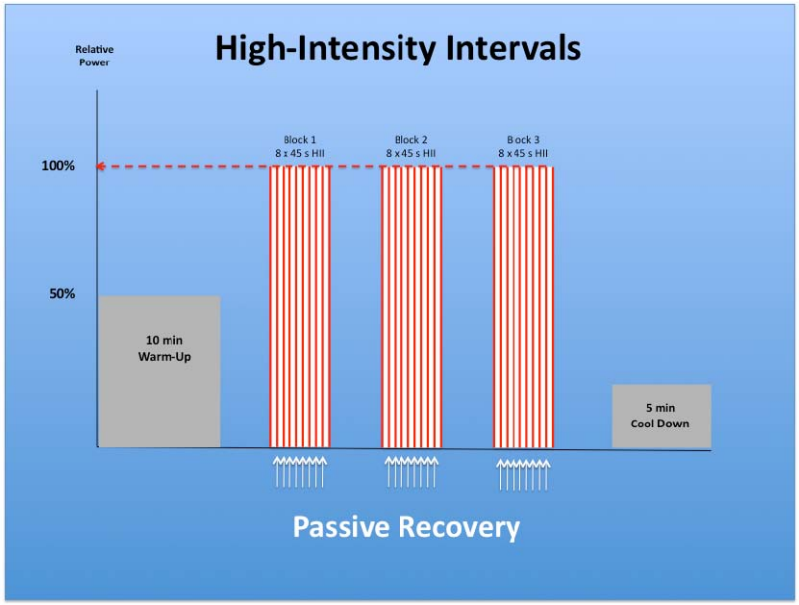

Figure 3.1: High-Intensity Intervals

it allows healthcare professionals to better monitor the acute cardiovascular responses to HIIT and design and implement individually tailored exercise programmes. Howerver, data produced by the heart rate telemetry system was at a very low level and a high level query language is not available for this type of data.

Aim To compare the effects of 4 weeks of traditional community-based cardiac rehabilitation programme and an individualized 4 week high-intensity intermittent exercise programme.

Specific Aim:

- To evaluate the effect of a 4 week high-intensity intermittent exercise programme.

3.1.2 Queries

Requirements of the exercise physiologists are shown in Table 3.1. None of these queries were possible to be expressed on the original datasets, because the original sensor data does not have any contextual information like participant name or state information related to

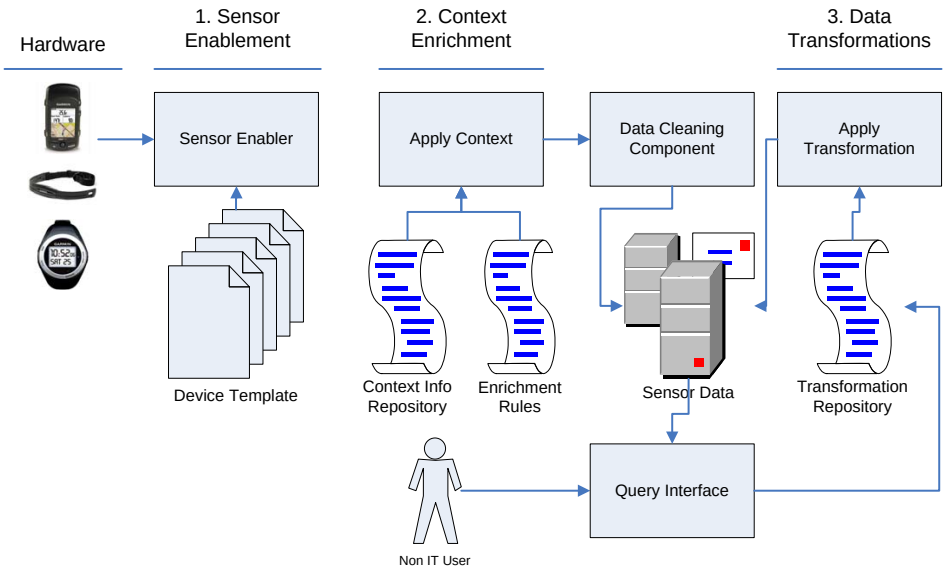

Figure 3.2: System Architecture

it in order to answer these queries. We will show our efforts to provide a proper query interface for the exercise physiologists in Chapter 4 and Chapter 5.

Table 3.1: Queries in English.

	Query
1	Return all participants names who belong to a specific group.
2	Return all participants names who wear a heart rate monitor in a specific activity.
3	Return all the values for each sensor reading of a participant during any specific state.
4	Calculate the minimal/maximum/average value for each sensor reading during any specific state.
5	Calculate the minimal/maximum/average value for each sensor reading in a specific group.
6	Return the participant name who has minimal/maximum value in a specific group.
7	Identify the incidence where each sensor reading of a participant was from N% to M% of its maximal value.
8	Identify any N second incidence where each sensor reading was at M% of its maximal value or above.
9	Identify the N minute rolling average for each sensor reading of a participant during any specific state.
10	Identify the earliest drop point/time reading of a participant where the slope between current sensor reading and next sensorreading is negative for each interval.

3.2 System Architecture

The motivation for our system is to allow users to specify queries at a high level, and to structure, enrich, transform and query sensor data. The architecture is depicted in Figure

3.2 as consisting of 5 major components, starting from the sensor devices which have various types like heartrate monitors, breathing monitor, etc. Sensor data is then collected by the *Sensor Enablement* component, which is responsible for structuring and mapping sensed data from their native, raw, format to a standard format. The *Context Enrichment* component merges the structured sensed data with the context information in which the data has been generated. The *Data Cleaning* component then detects and calibrates the outliers. At this point, sensor data is made available to the *Data Transformation*, and the *Query Interface* components.

Due to the fact that sensor data now has all the the context information such as participant name and state name associated with it, users can then access enriched sensed data with simple queries like 'List all the values for each sensor reading of one participant during any specific state' through the query interface. The full XQuery Expressions are listed in Example 3.1 that return all the values for a particular participant grouped by state name. However, as some like 'Calculate the minimal/maximum/average value for each sensor reading during any specific state' needs the aggregation information like minimal/maximum/average value from enriched sensed data before it can be specified, it is necessary to transform or further enrich the data in order to answer these complex queries.

Example 3.1. *Sample Query*

```
for $c := collection('HIIT')
$s in distinct-values($c//sos:user[text()='Ann']/sos:sensorData/sos:sections
/sos:section/sos:measurement/@state)
let $value := $c//sos:user[text()='Ann']
/sos:sensorData/sos:sections/sos:section/sos:measurement[@state = $s]
/sos:reading[sos:key[text()='HeartRate']]/sos:raw-value
order by $s
return <value state="{$s}">{for $v in $value return $v}</value>
```

3.3 Sensor Enablement

This component is responsible for parsing and converting raw sensor data generated from the hardware devices into a structured standard format. Raw data is a series of sensor readings plus some device specific parameters. Example 3.2 shows an example of raw data sensed by a heart rate monitor for HIIT Study. First, the sensor outputted some configuration data followed by a series of sensed heart rate values (delimited by the keywords `Params` and `HRData`). From this sample output data, it is clear that while being machine readable, sensor data cannot be manipulated without the development of a specific application.

Example 3.2. *Raw Sensor Stream for HIIT Study*

```
[Params]
Device=S1,
Version=106,
Monitor=2,
SMode=00000000,
Date=20110316,
StartTime=12:11:53.0,
Length=01:04:34.3,
Interval=1,
Upper1=0,
Timer1=0:00:00.0,
ActiveLimit=0,
StartDelay=0,
... (other configuration parameters)
[HRData]
71,
71,
72,
72,
74,
... (remaining heart rate values)
```

In our approach, we have chosen to transform raw data into a standard sensor format, known as the Sensor Web Enablement (SWE) initiative by the Open Geospatial Consortium (OGC)[25]. The Sensor Web Enablement group is a working group of the Open Geospatial

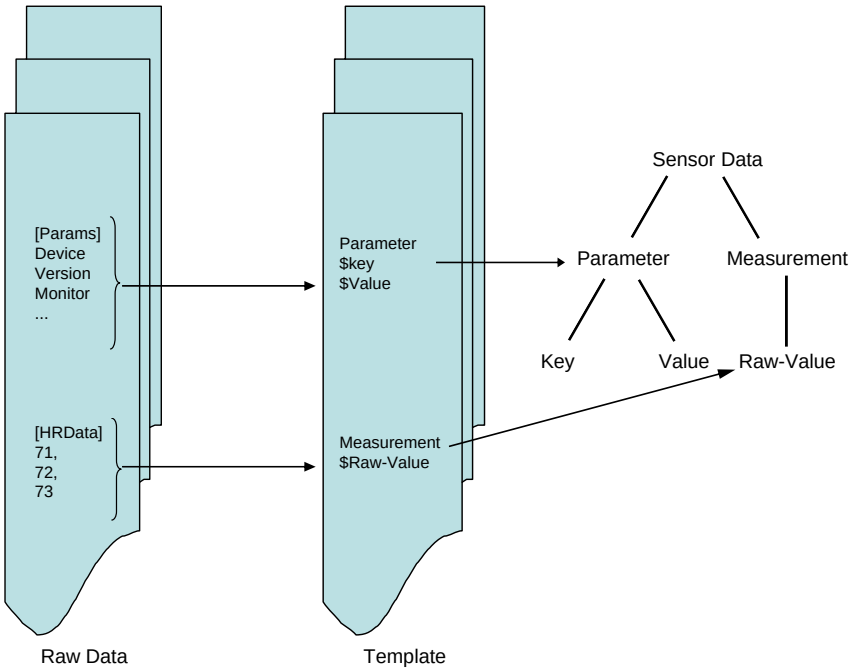

Figure 3.3: Convert Raw Data

Consortium, which refers to web accessible sensor networks and archived sensor data that can be discovered, accessed and, where applicable, controlled using open standard protocols and interfaces (APIs). The main contribution of the SWE is the specification of a number of standards which are responsible for specific aspects of managing sensor networks. For example, the Sensor Observation Service (SOS) is responsible for providing actual sensor readings encoded in a standard XML format which we are using for our stardard XML output. Concepts such as measurement, observation and process adopted in SWE initiative by the OCG have inspired us to manage and utilize data from different types of sensor devices. Moreover, we have developed a template approach on top of the OGC standard to ensure that no requirement of system modification but providing new XML templates when new sensors are developed.

The template driven approach converts the raw value into standard OGC XML tags in our real case scenario, which is depicted in Figure 3.3. For instance, in our real case scenario,

the template converts the paramenter `Version` into the complex element `<sos:paramenter>` composed of two sub-elements `<sos:key>Version</sos:key>` and `<sos:value>106</sos:value>`. Similarly, the heart rate data is actually converted into the complex element `<sos:measurement>` with its sub-element `<sos:raw-value>`. Example 3.3 shows a full enriched XML/OGC version of raw HIIT Dataset in Example 3.2. Note that adopting the OGC standard, facilitates the interoperability and integration of the sensed data with other OGC compliant data. Within the template a user can define functions to perform time and data format transformation or timestamps generation.

This is a relatively simple process but it plays an important role as it protects the system from changing in its envrionment, by providing Sensor Enablement we can make a number of basic assumptions regarding sensor data. Firstly, we provide generic structural markups for any sensor devices. Secondly, we convert the raw sensor data into XML format so that basic queries can be easily applied to the sensor data.

Example 3.3. *Enriched HIIT Dataset Now in OGC format*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetResult xmlns:sos="http://www.opengis.net/sos/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  service="SOS" version="1.0.0">
  <sos:ObservationTemplateId>urn:DCU:HeartRate</sos:ObservationTemplateId>
  <sos:sensorData>
<sos:device>HRM</sos:device><sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime><sos:interval>1000</sos:interval>
<sos:sections>
<sos:section name="Params">
<sos:parameter><sos:key>Version</sos:key><sos:value>106</sos:value>
</sos:parameter>
...  (other parameter elements)
</sos:section>
<sos:section name="HRData">
<sos:measurement offset="0" time="1300277513000">
<sos:reading ordinal="">
<sos:key>HeartRate</sos:key><sos:raw-value>71</sos:raw-value>
</sos:reading>
```

```
</sos:measurement>
... (measurement element repeats)
</sos:sensorData>
</sos:GetResult>
```

3.4 Context Enrichment

The Context Enrichment component has the role of integrating the structured sensor data with context information. Before the integration, although already queriable by means of XQuery expressions, sensed data provides only basic device related information from which meaningful knowledge extraction has not yet been provided. For instance, the output from a sensor measuring HR values worn by an athlete, return those values associated with a specific device identification (or code) rather than associated with the athlete, while for an high level user it is intuitive to pose a query using the athlete's name. The association with the athlete, as well as other information such as which team the athlete is with or what training session the sensed data belongs to, etc., is implicit in the context, i.e. known from those who deployed the sensors for each activity to monitor. As a result, a trainer who wants to know simple information such as "what is the performance level of a (specific) athlete," must first verify which sensor the athlete is wearing, then specify the query. Clearly, with more complex queries involving, for instance, information about teams or specific parts of an activity, require a significant amount of work in order to retrieve the data of interest. Such work is usually manual, complex and error prone. Thus, it can greatly benefit from explicitly including context knowledge in the sensed data. In order to integrate context information with sensed data, exercise physiologists must provide clear and sufficient context information by creating and populating a context repository with the required data. It is then necessary to define enriching rules to specify how to enrich (or merge) sensor data with context information. This is described in detail in Chapter 4.

3.5 Data Transformation

Although the sensor data is structured and semantically enriched, we also need to facilitate end users in specifying some complex queries like 'Identify any N second incidence where each sensor reading was at M% of its maximal value or above.'. Thus, the data transformation component is introduced to perform modification on the database in terms of data aggregations. Data aggregations is desirable when certain queries cannot be expressed. End user queries can be rather complex, requiring the processing of aggregated data, performing sorting and comparison on several attributes, etc. In this case, the user may want to precompute specific data (for instance used by several queries) so that queries can take advantage of such precomputations both in terms of expressions complexity and performance. This is described in detail in Chapter 5. Our contribution here will be transforming sensor data necessarily to make queries more manageable for exercise physiologists.

3.6 Summary

In this chapter, we went through all the components in our system architecture. The *Case Study* section demonstrated with a real world scenario into where we applied our system. The *Context Enrichment* component associates sensor data with the context information in which data has been generated. The *Data Transformation* component offers a set of predefined, yet customizable and extensible data transformation primitives, and also provides data functionalities to answer more complex user queries.

Chapter 4

Sensor Dataset Enrichment

This chapter presents the enrichment of enabled sensor data with additional semantics such as participant details, experiment details and state information in which has been generated. In Section 4.1 we present a brief overview of the integration process; in Section 4.2 we present an Event-Condition-Action approach for integrating sensor data with contextual information; in Section 4.3, we then provide detailed analysis of the Rule Based Enrichment.

4.1 Process Overview

The goal of sensor dataset enrichment is to provide sufficient semantics to enable end users to express basic queries using a high level, standard query language. There are many ways to achieve this type of enrichment but in general, a metabase which holds information about the context of the activity or experiment (people, objects, activity details, time, etc.) is required and then sensor data and context must be merged to provide a more enriched information source.

At an abstract level, a sensor will be deployed in a specific **Context** and in assocation with a specific **Activity**. A **Context** will always have one or more of *Participant/Team* (people or group of people who get invovled in the activity), *State* (the phase of the activity during which the sensor value was generated, or is relevant to), and *Timing* (information on the time

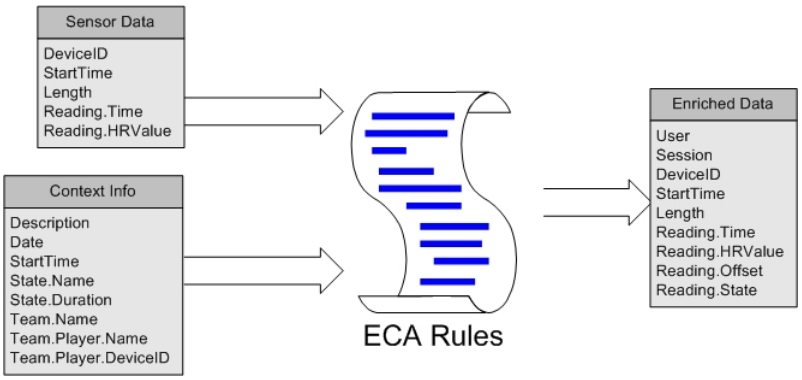

Figure 4.1: The Integration Process

at which an activity commenced and the interval at which sensor readings are generated). Fig 4.1 illustrates the overview of the integration process. The Integration Process has 4 components: **Sensor Data** in Fig 4.1 is the XML/OGC version of raw sensor data, which has the attributes as DeviceID, StartTime, Sensor Reading etc; **ECA Rules** show how context data integrate into the structure enriched data using our Event-Condition-Action rules, here we have ECA Rules work for adding in player information, adding in team information and also the most important one: adding in the *State* and *Timing* information; **Context Info** are the real values from External Data Sources, for instances: Warmup, Run and Recovery is the real values for the attribute *State*; **Enriched Data** are the context enriched data after ECA Rules has been applied to. Example 4.1 shows the external context information that relates to Example 3.3.

Example 4.1. *The External Context Infomation*

```
<?xml version="1.0" encoding="UTF-8"?>
<context>
<exercise_desc>HIIT running test</exercise_desc>
<date>16/03/2011</date>
<start_time>12:21:19</start_time>
<states><state><name>warmup</name><duration>10:00</duration></state>
<state><name>run</name><duration>30:00</duration></state>
<state><name>recovery</name><duration>35:00</duration></state></states>
<teams><team><name>Training Day 1</name>
<deviceID>S1</deviceID><deviceID>S2</deviceID><deviceID>S3</deviceID>
<participants><participant><name>Ann</name><deviceID>S1</deviceID>
```

```
<participant><name>Bryne</name><deviceID>S2</deviceID>
<participant><name>Paul</name><deviceID>S3</deviceID></participant></participants>
</team></teams>
</context>
```

In our approach we enable users to specify context enrichment in an Event-Condition-Action (ECA) like paradigm. Once defined, ECA-like rules are transformed into standard XQuery expressions, that can be applied to the input data and applied by any XML engine. We will describe the transformation process along with the rule template in the following section.

4.2 ECA Rules

Our ECA rules have a rather intuitive syntax. We use two pre-defined variables to refer to the input sources: *sensor* refers to the raw sensor data, while *context* refers to the context data. To navigate XML nested elements we adopt the . (dot), borrowing the notation from object modelling. Because the deploying scenario varies often, we focus on the provision of a method that enables non-expert users to specify how to enrich sensor data with context information.
An Event-Condition-Action rule template is shown in Definition 4.1.

Definition 4.1. *Event-Condition-Action rule template*

On	:	`<Event>`
When	:	`(<Condition>)+`
Do	:	`(<Action>)+`
Default	:	`(<Action>)?`

Each rule has four main sections: **On**, **When**, **Do** and **Default**.

- The **On** section specifies the event name to which this rule must be applied. The name can be specific name, such as “HIIT”, or if the rule is rather generic and can

be applied to any event, as in this case, then the keyword **any** can be used. While transforming into standard XQuery expressions, the keyword **On** is transformed into **let** clause in XQuery, the event name is transformed into the **collection** name in the database.

- The **When** section specifies the condition under which the rule must be executed. Generally, this is a condition on either the raw sensor data, the context data, or possibly both. The condition can be either one or more. In standard XQuery expressions, the keyword **When** is transformed into **if** clause, the condition stays as the same but each element is expressed using its XPath.

- The **Do** section specifies which action to perform as a manipulation of the input data. Same as the conditon section, the action can be one or more depending on the requirement. The keyword **Do** is transformed into **do insert** clause in standard XQuery expressions and the attribute of latter element is assigned to former element.

- Finally, the **Default** section specifies the default action to perform and it is optional. In standard XQuery expressions, the keyword **Default** is transformed into **else** clause.

Detailed example of how the ECA rule works will be described in the following section.

4.3 Rule Based Enrichment

There are three types of ECA rules that has been used in our integration process.

1. **Participant Mapping Rule**

   The Participant Mapping Rule integrates the basic participant information, which includes participant name, age, gender, DOB, address, phone number, and ect.

2. **Group Mapping Rule**

   The Group Mapping Rule adds the basic group information, which includes group name, location, number of people in the group, and ect.

3. **State Mapping Rule**

   The State Mapping Rule adds state information to sensed data to identify in which session of the training a participant is. For example: First Half, Half Time, Second Half are the states of a Gaelic Footbal game.

As the dynamics of sensor monitoring will see changing sensors, people or objects and the context in which data is delivered, this rule based enrichment provides a generic approach to avoid having to develop software wrappers for every experiment. It also provides a mechanism whereby end users (often non-computing users) can easily define how this merging of context and raw data takes place. Thus, our focus here is on the provision of a method that enables non-expert users to specify how to enrich sensor data with context information. Furthermore, with the additional Participant/Group/State information added on, it is now possible for users to express the query like 'Return all the values for each sensor reading of one participant during any specific state' on the enriched sensor datasets, a example is already provided in Section 3.2. In here we will walk you through each rule step by step.

4.3.1 Participant Mapping Rule

The purpose of the participant mapping rule is to add participant information, such as name and the training group the participant is with, into the XML data by mapping the device IDs from the raw sensor data with that from the context. Let us consider first the rule that adds personal information about the participants. For the sake of clarity in the presentation, let us assume that the only information we need to add is the participant's name (including additional information is straightforward).

Example 4.2. *Participant Mapping Rule*

On : `HIIT`

When : `context.participant.deviceID = sensor.deviceID`

Do : `sensor.user = context.participant.name`

Example 4.2 specifies that for the sensor data related to the activity "HIIT", if the device ID from the raw sensor data matches the device ID specified in the context for some participant then the participant's name is added to the sensed data. The transformation algorithm works as follows:

- The keyword **On** is transformed into **let** clause in XQuery, with the **collection** name "HIIT".
- The keyword **When** is transformed into **if** clause, and each element is expressed using its XPath.
- The keyword **Do** is transformed into **do insert** clause and the value of participant name is assigned to the name of *user* under sensor data itself.
- The keyword **Default** is not used here but will be shown late in Example 4.8.

Full XQuery expressions is listed in Example 4.3.

Example 4.3. *Participant Mapping Rule in XQuery*

```
let $c := collection("HIIT")
if $c/context//participant/deviceID = $c//sos:sensorData/sos:deviceID
do insert $c//sos:sensorData/<sos:user>$x/name</sos:user>
```

Applying this rule, the sensed data is updated as shown in Example 4.4 by adding the participant' names wrapped in XML tags.

Example 4.4. *An Enriched Sensor Stream with Player Name*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
< sos:user > Ann </ sos:user >
<sos:device>HRM</sos:device>
<sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime>
<sos:interval>1000</sos:interval></sos:sensorData>
</sos:sensorData>
```

4.3.2 Group Mapping Rule

Similarly, we can map the information of which group a participant was part of in a training session. In this rule shown in Example 4.5, if the device ID from the sensor data matches a device worn by an participant belonging to a certain group, then the group name is added to the sensor data. The group information is useful when the user wants to express queries over a group of people. At a high level, this rule is fairly similar to the preceding, which has the same behaviour in the generation of the XQuery implementation.

Example 4.5. *Group Mapping Rule*

On	:	`HIIT`
When	:	`context.team.deviceID = sensor.deviceID`
Do	:	`sensor.session = context.team.name`

Similar to the Participant Mapping Rule, the same transformation logic applies in this case. This rule inserts the XML element *session* with value "Training Day 1" to the sensor data and Example 4.6 lists the Full XQuery expressions.

Example 4.6. *Group Mapping Rule in XQuery*

```
let $c := collection("HIIT")
if $c//sos:sensorData/sos:deviceID = $c/context/team//deviceID
do insert $c//sos:sensorData/<sos:session>$c/context/team/name</sos:session>
```

After applying the rule, the default value of element *session* in Example 4.7 has been modified with the team name provided.

Example 4.7. *An Enriched Sensor Stream with Team Name*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
   <sos:user>Ann</sos:user>
   < sos:session > Training Day 1 </ sos:session >
<sos:device>HRM</sos:device>
<sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime>
<sos:interval>1000</sos:interval>
</sos:sensorData>
```

4.3.3 State Mapping Rule

The purpose of the State Mapping Rule is to add *state* (the phase of the activity during which the sensor value was generated, or is relevant to) information to sensed data to identify in which session of the training an participant is. For example the high-intensity interval training (HIIT) test is organized in a warmup session, followed by consecutive sessions of running, then a final recovery session. The state mapping rules associate heart rate values with the session (state) in which they have been generated according to the static definition of the test. So there are 3 major states in the HIIT test: the beginning of the test is marked as *warmup* which is 10 minutes in length; the next 20 minutes is denominated as *run* state; the last 5 minutes is marked as *recovery* for participants to cool down.

The states associations can be implemented as a sequence of ECA rules, one for each condition. Additionally, we introduce two new symbols in our condition part, the "()" symbol means the attribute of an element and the "[n]" symbol means the number n value of an element. Let us see the Example 4.8 for the rule that implement the above described enrichment.

Example 4.8. *State Mapping Rule*

```
On      :   HIIT

When    :   sensor.time >= context.start_time and
            sensor.time < (context.start_time + context.duration[1])
Do      :   sensor.measuremnt(state) = context.state.name[1]

When    :   sensor.time >= (context.start_time + context.duration[1]) and
            sensor.time < (context.start_time + context.duration[2])
Do      :   sensor.measuremnt(state) = context.state.name[2]

When    :   sensor.time >= (context.start_time + context.duration[2]) and
            sensor.time < (context.start_time + context.duration[3])
Do      :   sensor.measuremnt(state) = context.state.name[3]

Default :   sensor.measuremnt(state) = "rest"
```

This rule looks more complex than formal ones, but the transformation logic remains the same except that the **Default** section is used in this rule to specify the default action.

- In this case **Default** is transformed into **else** clause with an action by setting the default value of sensor state.

The XQuery equivalent is shown below in Example 4.9. The state information will be inserted as an attribute into the *measurement* element.

Example 4.9. *State Mapping Rule in XQuery*

```
let $c := collection("HIIT")
if ($c//sensorData//time >= $c/context/start_time and
     $c//sensorData//time < ($c/context/start_time + $c/context/duration[1]))
do insert attribute $c//sensorData@state '$c/context/state/name[1]'
if($c//sensorData//time >= ($c/context/start_time + $c/context/duration[1]) and
     $c//sensorData//time < ($c/context/start_time + $c/context/duration[2]))
do insert attribute $c//sensorData@state '$c/context/state/name[2]'
if ($c//sensorData//time >= ($c/context/start_time + $c/context/duration[2]) and
     $c//sensorData//time < ($c/context/start_time + $c/context/duration[3]))
do insert attribute state '$c/context/state/name[3]'
else(do insert attribute $c//sensorData@state 'rest' )
```

Applying the State Mapping rule, the sensed data is updated by inserting the state information in Example 4.10.

Example 4.10. *An Enriched Sensor Stream with State Infomation*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
   <sos:user>Ann</sos:user>
   <sos:session>Training Day 1</sos:session>
<sos:device>HRM</sos:device>
<sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime>
<sos:interval>1000</sos:interval>
<sos:section name="HRData">
<sos:measurement state="rest" offset="0" time="1300277513000">
<sos:reading ordinal="">
<sos:key>HeartRate</sos:key>
```

```
<sos:raw-value>71</sos:raw-value>
</sos:reading>
</sos:measurement>
... (measurement element repeats)
</sos:sensorData>
```

4.4 Summary

In this chapter, we introduced the ECA rule based context enrichment which facilitates a wide range of basic queries by adding additional context information to the sensed data. At this point, we have already associated the participant, group and state information with sensor data. Therefore, we need to extract further knowledge from the sensed data using transformation functions. The next chapter will provide a detailed description of sensor outlier removal algorithm, followed by several data transformation functions which facilitate the advanced queries proposed by exercise physiologists in Section 3.1.

Chapter 5

Sensor Dataset Transformation

This chapter presents the levels of knowledge and transformations we can extract from the sensed data. In Section 5.1 we provide an algorithm to detect and calibrate outliers; in Section 5.2 we present a wide range of data transformations in order to handle more complex analyses requested by exercise physiologists.

5.1 Outlier Removal Algorithm

In almost all sensor networks, the network will generate outlier values which are clearly outside the normal acceptable range. Before any queries or transformations of data can progress, it is necessary to detect and calibrate these outliers. We present a method that operates on XML sensor output and can be parametrised by exercise physiologists. There are four primary steps: Set Valid Range; Identify Candidate Outliers; Identify Actual Outliers; and Calibrate Outlier.

Step 1. Set Valid Range, MinValue and MaxValue In heart rate (HR) monitoring, the general rule of thumb for Max HR is $HR_{Max} - age$ (and generally HR_{Max} = 220). And we also need to introduce a variable HR_{Min} to set the minimal heart rate value (and usually HR_{Min} = 50). Variable HR_{Limit} represents the boundary of heart rate range, it could be either $HR_{Max} - age$ or HR_{Min}. While this is generally applicable, we need to be as

flexible as possible as we are dealing with an age group that may have wide differences. Here we define the following function to identify the upper and lower bounds of the Max HR range and the Min HR range, which contain valid candidate outliers.

Definition 5.1. *Outlier Removal Algorithm*

$$f_{ValidRange}(hr, HR_{Limit}, variance) = \begin{cases} \textbf{\textit{FALSE}} & \textit{if } (hr > HR_{Limit} * (1 + variance)) \\ \textbf{\textit{FALSE}} & \textit{if } (hr < HR_{Limit} * (1 - variance/2)) \\ \textbf{\textit{TRUE}} & \textit{Otherwise} \end{cases}$$

As HR_{Limit} consists of two boundaries: the highest boundary $HR_{Max} - age$ and the lowest boundary HR_{Min}, we need to set two valid ranges using this algorithm. In the case of Max HR experiment we have age, so the probable max HR is $ProbableMax = HR_{Max} - age$. Then we add a 10% variance ($variance = 10\%$) so that anything upto $ProbableMax + 10\%$ is a possible valid MaxHR. Anything above is an outlier (automatic). Thus: $MaxValue = ProbableMax + 10\%$; $MinValue = ProbableMax - 5\%$. These two upper and lower bounds set up Max HR valid range.

The same logic applies to Min HR experiment, in this case, $MaxValue = HR_{Min} + 10\%$ is the upper bound of Min HR valid range and $MinValue = HR_{Min} - 5\%$ is the lower bound. Anything below $HR_{Min} - 5\%$ is also treated automatically as an outlier. Moreover, anything between $ProbableMax - 5\%$ and $HR_{Min} + 10\%$ cannot be an outlier, which is important because there are many variances we need to check as the heart rate increases swiftly in the early stages of activity.

Step 2. Identify Candidate Outliers Read through the stream of heart rates from start to finish. We examine all heart rates which are within the above two defined valid ranges: $ProbableMax - 5\%$ upto $ProbableMax + 10\%$ and $HR_{Min} - 5\%$ upto $HR_{Min} + 10\%$. Anything within these two ranges is a *Candidate Outlier*.

Step 3. Identify Actual Outlier Read the 5 heart rates before and 5 heart rates after the candidate outlier. Calculate the mean of these 10 values ($Mean\_Compare$) If the candidate outlier is outside 5% of $Mean\_Compare$, where 5% is a variance parameter, it is deemed an Actual Outlier.

Step 4. Calibrate Outlier Once deemed an Actual Outlier, it will then be replaced with the $Mean\_Compare$ calculated in Step 3.

5.1.1 Algorithm Evaluation

We evaluate our work using datasets from High-Intensity Interval Training (HIIT) study described in Section 3.1. Table 5.1 shows the statistics of all the outliers detected and calibrated in the HIIT datasets. As the average age group in HIIT study is 50 we have $age = 50$ and thus, $ProbableMax = HR_{Max} - age = 170$. So the Max HR valid range is between $ProbableMax - 5\% = 161$ and $ProbableMax + 10\% = 187$. Meanwhile, the Min HR valid range is between $HR_{Min} - 5\% = 47$ and $HR_{Min} + 10\% = 55$. Anything between these two valid ranges is a *Candidate Outlier*. 60 files were fed through this outlier removal process and we detected 573 candidate outliers, but only 11 of them were outside 5% of $Mean\_Compare$ and deemed an actual outlier. Calibrated outliers all comes from the the Max HR valid range which means outliers mainly occur when a participant nearly reaches some threshold close to her maximal performance. There are still 74 outliers (automatic) caused by erroneous error input, all of them are zero readings from heart rate monitor.

Table 5.1: Sensor Outlier Statistics

Dataset	No. of Files	Candidate Outliers	Outliers Calibrated	Outliers (automatic)
HIIT	60	573	11	74

5.1.2 Summary

The broad goal of this algorithm is to monitor a range of consecutive heart rate measurements and observe if there is a sudden jump before an immediate return to within, or close

to, the average of the current set of values. Using the variance parameters, we showed this to be very effective with one exception. Heart rate values during activities tend to start low, build fairly rapidly and then hit some threshold close to the participant's maximal performance. Our algorithm works well once the threshold figure has been reached. What we did here is to identify, among all the sensor readings in a sliding window, those values that have very few near neighbors. Outliers are often caused by measurement error, and failing to remove them will probably result in wrong or misleading measurement on athelet for exercise physiologists.

5.2 Data Transformation

As our target user is non-IT, our method is deliberately simple. We have a series of low-level data transformation primitives that calculate for peaks, troughs, distances between peaks and troughs, search rolling averages and also calculate slopes. The end user adds semantics by applying names to this new knowledge and associating this knowledge with different data streams. The benefit is to have a repository from which new queries can retrieve basic transformations to build new ones, achieving reuse and simplifying the engineer's task.

Basic contextual enrichment facilitates a range of basic queries but cannot handle more complex ones. Given the simple nature of sensor data, which is often single or multiple values generated at standard intervals, it is possible to predefine a series of operations to generate new knowledge. A set of data transformation primitives are used to run a standard set of analyses to which end users can add their own semantics. The primitives we currently provide are the following:

- fn:Minimal - Calculate the minimal sensor readings for a participant during the activity;
- fn:Average - Calculate the average sensor reading of a participant during the activity;

- fn:Max - Calculate the maximum sensor readings for a participant during the activity;
- fn:As % Range - Give a percentage range of a specific sensor readings (normally the maximal value) for a participant during the activity;
- fn:SlidingWindow(N) - Give the N-minute sliding window for each sensor reading of a participant over the activity;
- fn:Slope($time_1$, $value_1$)&($time_2$, $value_2$) - Calculate the slope between two sensor readings ($value_1$ and $value_2$) based on the time ($time_1$ and $time_2$), and

$$Slope = \frac{value_2 - value_1}{time_2 - time_1}$$

In our experimental setting, exercise physiologists were interested in answering the queries provided in Section 3.1. We choose a few samples of them here to give a general overview of how Data Transformation works in answering those queries.

Table 5.2: Sample Queries in English.

	Query
1	Calculate the average value for each sensor reading of a participant during any specific state.
2	Identify the incidence where each sensor reading of a participant was from N% to M% of its maximal value.
3	Identify the N minute rolling average for each sensor reading of a participant during any specific state.
4	Identify the earliest drop point/time of a participant where the slope between current sensor reading and next sensor reading is negative for each interval.

These queries can be answered by selecting one of the built-in functions for data transformation or by composing a few of them in a sequence. We have an XQuery expression template in Example 5.1, which contains various parameters needed for data transformations. The *$event* specifies name of the event, similarly, *$participant*, *$group* and *$state* specify the participant, group and state information. *$value* and *$time* specify the sensor reading and its corresponding time, *$total* specifies the total number of sensor readings.

Example 5.1. *XQuery Transformation Template*

```
for $c := collection('$event')
let $participant := $c//sos:user/text() $group := $c//sos:session/text()
$state := distinct-values($c//sos:measurement/@state)
$value := $c//sos:user[text()='$participant']/sos:sensorData/sos:sections
/sos:section/sos:measurement/sos:reading[@state = $state]/sos:raw-value
$time := $c//sos:user[text()='$participant']/sos:sensorData/sos:sections
/sos:section/sos:measurement/@time
$total := fn:count($c//sos:user[text()='$participant']/sos:sensorData/sos:sections
/sos:section/sos:measurement/sos:reading/sos:raw-value)
return do insert
```

Each data transformation primitive also has its XQuery implementation which will get populated into the XQuery transformation template:

- fn:Minimal: 'let $min := fn:min($value)';
- fn:Average: 'let $avg := fn:avg($value)';
- fn:Max: 'let $max := fn:max($value)';
- fn:As % Range: 'for $range in (0 to 1) let $percentage := 0';
- fn:SlidingWindow: 'for sliding window $window in (0 to $total) let $length := 0';
- fn:Slope: 'let $slope := ($value - following-sibling::$value) / ($time - following-sibling::$time)'.

We choose High-Intensity Interval Training (HIIT) study as an example to show the necessary data transformation process needed in answering the queries in Table 5.2. In relation to these 4 queries, we will introduce the following 4 corresponding transformations: average transformation, peak transformation, rolling average transformation and slope transformation.

5.2.1 Average Transformation

For example to answer the first query *Calculate the average value for each sensor reading of a participant during any specific state*, only the fn:Average primitive is used to get the

average value during any specific state. So the fn:Average XQuery implementation is added into the transformation template and thus, populated with HIIT data. The full XQuery expression is shown in Example 5.2. .

Example 5.2. *Average Transformation in XQuery*

```
for $c := collection('HIIT')
let $participant := $c//sos:user/[text()='Ann']
$group := $c//sos:session/[text()='Training Day 1']
$state := distinct-values($c//sos:measurement/@state)
$value := $c//sos:user[text()='Ann']/sos:sensorData/sos:sections
/sos:section/sos:measurement/sos:reading[@state = $state]/sos:raw-value
$time := $c//sos:user[text()='Ann']/sos:sensorData/sos:sections
/sos:section/sos:measurement/@time
$total := fn:count($c//sos:user[text()='Ann']/sos:sensorData/sos:sections
/sos:section/sos:measurement/sos:reading/sos:raw-value)
let $avg := fn:avg($value)
return do insert <avg state="{$s}">$avg</avg>
```

After applying the average data transformation, the sensor data from HIIT study is then updated into Example 5.3.

Example 5.3. *HIIT Sensor Data Updated With Average Transformation*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
<sos:user>Ann</sos:user>
<sos:session>Training Day 1</sos:session>
<sos:sensorData candidate="candidate">
<sos:device>HRM</sos:device>
    <sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime>
<sos:interval>1000</sos:interval>
< avg state="warmup" > 101.67 </ avg >
< avg state="run" > 119.21 </ avg >
< avg state="recovery" > 97.73 </ avg >
</sos:sensorData>
```

5.2.2 Peak Transformation

However, further transformations are necessary in order to answer more complex queries like the second query *Identify the incidence where each sensor reading of a participant was from N% to M% of its maximal value*. First the fn:Max is used to get the maximum value; then based on that maximum value, the fn:As % Range 90% to 100% will be triggered. Same transformation logic applies here as these two transformation primitives have been added into the transformation template. The full XQuery expressions is listed in Example 5.4:

Example 5.4. *Peak Transformation in XQuery*

```
for $c := collection('HIIT')
...same criteria apply
let $max := fn:max($value)
let $percentage := 0.9
return do insert <sos:peakHR>$max</sos:peakHR>
then do insert <sos:peakHR90>$max*$percentage</sos:peakHR90>
```

After applying the peak data transformation, the HIIT sensor data is then updated into Example 5.5.

Example 5.5. *HIIT Sensor Data Updated With Peak Transformation*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
<sos:user>Ann</sos:user>
<sos:session>Training Day 1</sos:session>
<sos:sensorData candidate="candidate">
<sos:device>HRM</sos:device>
    <sos:deviceID>S1</sos:deviceID>
<sos:startTime>1300277513000</sos:startTime>
<sos:interval>1000</sos:interval>
< sos:peakHR > 141 </ sos:peakHR >
< sos:peakHR90 > 127 </ sos:peakHR90 >
</sos:sensorData>
```

5.2.3 Rolling Average Transformation

Moreover, for answering the third Query *Identify the N minute rolling average for each sensor reading of a participant during any specific state*, the fn:Average is triggered, then the fn:SlidingWindow(N) will provide a one-minute sliding window for the fn:Average to calculate the 1 minute rolling averages. Same transformation logic applies and the full XQuery expression for this transformation is listed in Example 5.6:

Example 5.6. *Rolling Average Transformation in XQuery*

```
for $c in collection("HIIT")
...same criteria apply
let $avg := fn:avg($value)
for sliding window $window in (0 to $total) let $length := 60000
return do insert <sos:average><sos:time>$length</sos:time>
<sos:value>$avg[@time=$length in $window]</sos:value></sos:average>
```

After applying the rolling average data transformation, the HIIT sensor data is then updated into Example 5.7.

Example 5.7. *HIIT Sensor Data Updated With Rolling Average Transformation*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
<sos:section name="HRData">
<sos:measurement offset="0" state="warmup" time="1209826519000">
<sos:reading ordinal="">
<sos:key>HeartRate</sos:key>
<sos:raw-value>71</sos:raw-value>
< sos:average >
    < sos:time > 60000 </ sos:time >
< sos:value > 82.36 < sos:value >
</ sos:average >
... (measurement element repeats)
</sos:sensorData>
```

5.2.4 Slope Transformation

The slope of a heart rate curve tells the exercise physiologists by how much their heart rate is increasing or decreasing. In order to answer the last Query *Identify the earliest drop point/-*

time of a participant where slope between current sensor reading and next sensor reading is negative for each interval, only the fn:Slope primitive is used to calculate the value of slope for each sensor reading. So the fn:Slope XQuery implementation is added into the transformation template and again, populated with HIIT data. Again, the full XQuery expression is listed in Example 5.8:

Example 5.8. *Slope Transformation in XQuery*

```
for $c := collection('HIIT')
...same criteria apply
let $slope := ($value - following-sibling::$value)
/($time - following-sibling::$time)
return do insert <sos:slope>$slope</sos:slope>
```

After applying the slope data transformation, the HIIT sensor data is then updated into Example 5.9.

Example 5.9. *HIIT Sensor Data Updated With Slope Transformation*

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:sensorData>
<sos:section name="HRData">
<sos:measurement offset="0" state="warmup" time="1209826519000">
<sos:reading ordinal="">
<sos:key>HeartRate</sos:key>
<sos:raw-value>71</sos:raw-value>
< sos:slope > 0 </ sos:slope >
... (measurement element repeats)
</sos:sensorData>
```

5.3 Summary

In this chapter, a detailed description of outlier removal algorithm was introduced, followed by a deep analysis of several sensor data transformations. At this point, queries in Section 3.1 are now made available to exercise physiologists. In the next chapter, we will validate our work based on the evaluation of experiment results.

Chapter 6

Evaluation

Chapter 6 evaluates the experiments performed to validate our approach. In Section 6.1, we give an overview of our evaluation process; in Section 6.2, we give a general description of the system setup where the experiments run on; in Section 6.3, we describe the sensor datasets and queries we have for the experiments; and finally in Section 6.4, we analyse the experiment result and its impact on the exercise physiologists' work.

6.1 Evaluation Overview

In this section, we evaluate our work in an incremental manner. As the exercise physiologists cannot express any query on the raw sensor datasets, we need to close the semantic gap between end user requirements and raw datasets, and the best way is to provide a number of processors and algorithms to transform the data to a format that could be queried using a high level query language. To do this, we implemented a template driven approach to convert raw sensor data into XML files. We also implemented the Event-Condition-Action (ECA) rule approach, which has three types of integration rules, to enrich the XML files with additional context information such as participant details, experiment details and state information which has been generated. Then we implemented the outlier removal algorithms to detect and calibrate all the necessary outliers. As the goal is to facilitate end

user to express more complex queries, we implemented a wide range of data transformations in order to handle more complex analyses requested by exercise physiologists. In the following sections, we will provide a description of our experimental setup, followed by an introdution of the detailed experiment, and then we proceed to the evaluation and a discussion of the experiment results.

6.2 System Description

After the enrichment and transformation processes, data is stored in the MonetDB XQuery server [26] where it is then calibrated until ready for user queries. There is one MonetDB database server deployed for this experiment, which acts as the local database server storing all source XML data. All experiments ran on an Intel Core 2 Duo processor PC with 4GB of RAM running Windows XP professional. The system was implemented using the Sun Java Virtual Machine version 1.6 and standard XQuery 1.0; the MonetDB XQuery server is version 4.30.0. All experiments were executed four times with the average of the last three runs times recorded. The first run was treated as a cold run and thus ignored. Table 6.1 lists all four sensor datasets we have, each of their sizes in raw format, the number of files in each dataset, and their total sizes in the MonetDB after the enrichment and transformation process.

Table 6.1: Sensor Datasets Statistics

Dataset	Size of Original Dataset	No. of Files	Total Size in Database
The HIIT datasets	1.1 MB	60	39 MB
The Bangsbo datasets	2.5 MB	147	100 MB
The breathing datasets	12.8 MB	52	162 MB
The referee datasets	211 MB	35	837 MB

6.3 Experiments

The queries listed in Section 3.1 are real world requirements provided from exercise physiologists. These queries are not expressible on the original data, they can only be executed

once sensed data is integrated with its context and transformations to expose data of interest have been applied. Here we state the various datasets we have for our research evaluation:

- The HIIT datasets: sensor data generated from a set of HeartRate sensor devices worn by High-Intensity Interval Training (HIIT) group.
- The Bangsbo endurance datasets: sensor data collected from HeartRate sensor devices and provided by the Dublin Gaelic football team for their Bangsbo endurance tests.
- The breathing datasets: sensor data generated from the breathing monitor supported by Sprint type Interval Training (SIT) group.
- The referee datasets: sensor data consisting of GPS, accelerometer and HeartRate sensor outputs supported by professional GAA referees.

Table 6.2 provides the full details for each process in terms of: Single Data Input Size; Result Size for Enrichment Output; the Execution Time for Enrichment Process; Result Size for Transformation Output and the Execution Time for Transformation Process. For example, in the HIIT datasets, a single raw sensor data file is only 18 KB in size. And after the enrichment process, it has been converted into a single XML file with associated semantics which becomes 627 KB in size and has been stored into the MonetDB XQuery server. It takes about 10 seconds for the whole enrichment process. The transformation process will enlarge the file upto 704 KB in five and half seconds, having performed all possible transformations and is ready to be queried through relatively simple XQuery expressions by end user.

Table 6.2: Process Details

Dataset	Input Size	Enrichment Output	Execution Time	Transformation Output	Execution Time
HIIT dataset	18 KB	627 KB	10s 141ms	704 KB	5s 563 ms
Bangsbo dataset	20 KB	652 KB	11s 372ms	779 KB	6s 141 ms
Breathing dataset	233 KB	3016 KB	32s 609ms	3725 KB	19s 78 ms
Referee dataset	6023 KB	24257 KB	1m 05s 361ms	29384 KB	52s 394 ms

6.4 Analysis

Table 6.3: Sample Queries in English.

	Query
1	Return all participants names who belong to a specific group.
2	Return all participants names who wear a heart rate monitor in a specific activity.
3	Return all the values for each sensor reading of a participant during any specific state.
4	Calculate the average value for each sensor reading of a participant during any specific state.
5	Calculate the maximum value for each sensor reading in a specific group.
6	Identify the incidence where each sensor reading of a participant was from N% to M% of its maximal value.
7	Identify the N minute rolling average for each sensor reading of a participant during any specific state.
8	Identify the earliest drop point/time of a participant where the slope between current sensor reading and next sensor reading is negative for each interval.

In this section we use the High-Intensity Interval Training (HIIT) dataset as our main experiment to validate our approach and its impact on the exercise physiologists work. Table 6.3 shows the sample queries in English. None of these queries were possible to be expressed over the original datasets. Query 1 to query 3 can only be answered through the MonetDB query interface after the sensor enrichment has been processed. When all the transformations proposed in Section 5.2 have been applied to the sensor datasets, it is then possible to express query 4 to query 8 over the transformed datasets. As a result, we divide these queries into two parts: queries handled by enrichment; and queries handled by transformation. Each query's XQuery implementation and result will be provided and analysed. The execution time is there for illustration purpose only.

Queries Handled by Enrichment Results and execution time are provided in Table 6.4. Each XQuery implementation corresponds to each query in English.

- XQuery 1. This answers Query 1 'Return all participants names who belong to a specific group'. The result lists all possible names in a training session in less than 200 ms with the reason being that they query a small group of sensor data files.
- XQuery 2. The query 'Return all participants names who wear a heart rate monitor in a specific activity' returns all possible participant's name using a heart rate monitor

in HIIT study. The execution time is twice as long as the previous query due to the fact that the whole dataset is queried in order to get the result.

- Query 3. The query 'Return all the values for each sensor reading of one participant during any specific state' is answered in approximately a second, because it needs to list all possible sensor readings group by each state for a single participant.

Table 6.4: XQuery Expressions for Queries Handled by Enrichment

#	Query as XQuery expression	Results	Execution Time
1	for $c := collection('HIIT') let $participant := $c//sos:user/text() $group := $c//sos:session/text() where $group = 'Training Day 1' return <name group="$group">$participant</name>	<name group="Training Day 1"> Ann </name> ...	197 ms
2	for $c := collection('HIIT') let $participant := $c//sos:user/text() $sensor := $c//sos:reading/sos:key/text() where $sensor = 'HeartRate' return <name sensor="$sensor">$participant</name>	<name sensor="HeartRate"> Ann </name> ...	381 ms
3	for $c := collection('HIIT') , $s in distinct-values($c//sos:user [text()='Ann']/sos:sensorData/sos:sections/sos:section /sos:measurement/@state) let $value := $c//sos:user[text()='Ann'] /sos:sensorData/sos:sections/sos:section/sos:measurement[@state = $s] /sos:reading[sos:key[text()='HeartRate']]/sos:raw-value order by $s return <avg state="$s"> for $v in $value return $v </avg>	<value state="warmup"> 78 </value> <value state="warmup"> 79 ...	998 ms

Queries Handled by Transformation Results and execution time are reported in Table 6.5 for each XQuery implementation handled by data transformation.

- XQuery 4. The query expression calculates the average value for each sensor reading of a participant grouped by each state in less than half a second. The results are in XML format and can be viewed in any text editor.

- XQuery 5. It answers the Query 'Calculate the maximum value for each sensor reading in a specific group', which is similar to the previous query. It works quite efficiently due to the fact that it returns only one single result.

- XQuery 6. The query answers the Query 'Identify the incidence where each sensor reading of a participant was from N% to M% of its maximal value'. It executes even quicker since it only queries a single sensor data file in 241 ms.

- XQuery 7. This query expression returns the one minute rolling averages for each sensor reading during any state, it is the slowest query at 769 ms because it needs to retrieve all possible averages calculated in a one minute rolling window for a single participant.

- XQuery 8. This returns the earliest time where a sensor reading starts falling. The execution time is less than 200ms with the reason that it only queries one sensor data file and retrieves one single result.

Table 6.5: XQuery Expressions for Queries Handled by Transformation

#	Query as XQuery expression	Results	Execution Time
4	for $c := collection('HIIT') , $s in distinct-values($c//sos:user [text()='Ann']/sos:sensorData/sos:sections/sos:section /sos:measurement/@state) return <avg state="$s"> $c//avg/text() </avg>	<avg state="warmup"> 101.67 </avg> ...	494 ms
5	for $c := collection('HIIT') let $participant := $c//sos:user/text() $group := $c//sos:session/text() where $participant = 'Ann' and $group = 'Training Day 1' return <max group="$group"> fn:max($c//sos:peakHR) </max>	<max group="Training Day 1"> 177 </max>	316 ms
6	for $c := collection('HIIT') let $max := $c//sos:peakHR90/text() return <count> fn:count($csos:user[text()='Ann']/sos:sensorData /sos:sections/sos:section/sos:measurement/sos:reading [sos:key[text()='HeartRate']]/sos:raw-value[text()>$max]))</count>	<count> 733 </count>	241 ms
7	for $c := collection('HIIT') let $participant := $c//sos:user/text() where $participant = 'Ann' return <rollingAvg> $c//sos:averages/sos:average[sos:time[text()='60000']]/ sos:value[text()]</rollingAvg>	<rollingAvg> 82.36 </rollingAvg> ...	769 ms
8	for $c := collection('HIIT') let $participant := $c//sos:user/text() $slope := $c//sos:slope/text() where $participant = 'Ann' and $slope ¿ 0 return <time participant="$participant"> $c//sos:measurement/@time </time>	<time participant="Ann"> 12:19:16.0 </time> ...	182 ms

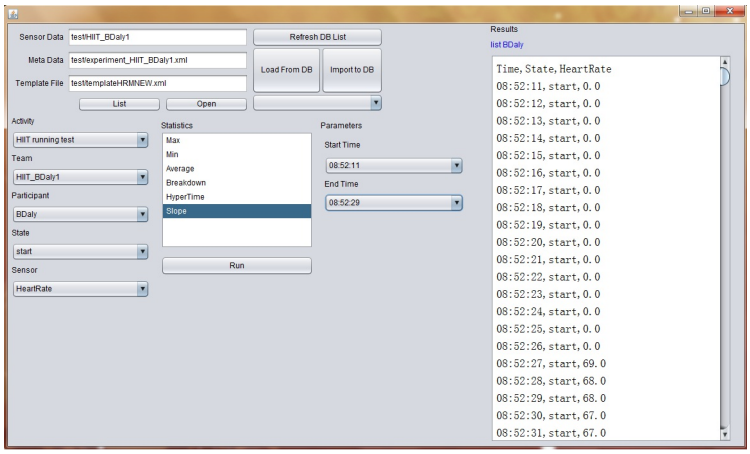

Figure 6.1: User Interface

Let us emphasise the following two facts. Queries are executed on the sensor data obtained after the above discussed enrichment and transformation process: without such enrichments or transformations these queries are either difficult or impossible to express on the original dataset. Queries are implemented as standard XQuery expressions: if needed, they can be modified and customised using a simple text editor thus the system implementation does not need to be altered.

6.5 Summary and End User Feedback

In order to determine how best to deliver a system for pHealth sensors, I regularly consulted with exercise physiologists from the School of Health and Human Performance. Through a series of interviews, I learnt of their requirements; using three physiologists work areas in different scenarios. I continued to work with them as I developed the system, and I had them evaluate my work. The results obtained were accurate to the extent that previously unattainable information was included in their PhDs. The collabrative effort in the production of several publications [10] [36] [37] throughout my research period. Figure 6.1 shows the basic user interface I developed for them to query the sensor data.

The results for all queries in this chapter can also be stored in the database and thus, it is possible for exercise physiologists to propose new queries to access new information. For

example, they can query some average or maximum sensor reading across the groups and even across the different studies. They could also express queries across different sensors. Thus, queries become more flexible and extendable after our enrichment and transformation process.

The Exercise physiologists demonstrated a favourable opinion of the prototype. While avoiding the manual error prone processes, they have been able to extract knowledge and perform the analysis of interest quickly. They also realised the possibility of having access to a wider information and analysis capacity than was previously available. From those experiments, exercise physiologists were able to determine the intensity at which each subject trained. These measures can be used to prescribe various exercise protocols which can be used to further enhance endurance capacity. Analysing heart rates in such a capacity also enables exercise physiologists to identify training targets and optimal training zones for each subject which is essential when prescribing an individual exercise program. The queries posed during this study enabled physiologists to determine how hard each subject trained and whether the level of training was adequate to induce the training adaptations required.

Chapter 7

Conclusions and Future Work

In this dissertation, we presented a system that allows users to specify queries at a high level, and to structure, enrich, transform and query sensor data. In this final chapter, we review the work presented and following that, we discuss areas where future work can be explored.

7.1 Thesis Summary

This work is the result of a collaboration between exercise physiologists and computer scientists. Sensors often generate large volumes of data, making analysis very difficult. There is no standard format for query output, no method of merging results from different tests or integrating results across groups, and a full query interface to datasets does not exist. In this dissertation, we presented a framework for capturing low level sensor data and through a number of enrichment processes to transform data to a position where high level query expressions were possible. We also demonstrated in our evaluation that these query expressions captured the precise needs of the domain experts. The end result is a process whereby the exercise physiologists can run different test configurations and adapt their behaviour between tests as a result of the fully automated query process.

The initial challenge for us was to close the semantic gap between end user requirements and raw datasets, by providing a number of processors and algorithms to transform the data to a format that could be queried using a high level query language. We outline our achievements as they show that our objectives were reached:

1. A rule based paradigm with which we can associate sensor data with the context information in which these data has been generated.

2. An efficient sensor outlier removal algorithm was developed.

3. Finally, a set of data functionalities or transformations to enable the end user express complex queries against low level sensor data, were provided.

A methodology was set to complete each goal in an incremental manner through the entire dissertation.

In Chapter 1, an introduction to the Sensor Web [15] was presented and existing efforts of personal health (pHealth) sensor networks were briefly discussed. It was concluded that, there is a significant gap between high level user requirements and the data generated by sensors. When this data is generated in high volumes, there is a significant barrier to knowledge extraction. As a result, we needed to provide a mechanism to automatically enrich and transform sensor data up to a point, and then through user interaction provide new levels of semantic enrichment.

In Chapter 2, an analysis of related research in this field was given. We discussed three main research areas: context modeling architecture, rule based contextual enrichment, and sensor outlier removal.

In Chapter 3, we described a real world case study, and provided an overview of a system which structures, enriches, transforms and queries sensor data in this real world scenario.

In Chapter 4, we presented the enrichment of enabled sensor data using a rule-based approach, an Event-Condition-Action rule template was introduced in this chapter, followed by three types of context enrichment rules.

In Chapter 5, we introduced our outlier removal algorithm in detecting and calibrating the necessary outliers, we also presented a set of data transformation primitives that transform sensor data necessarily to make queries more manageable for exercise physiologists.
In Chapter 6, we evaluated the experiments to validate our approach and its impact on the exercise physiologists work, and they demonstrated a favourable opinion on the prototype.

7.2 Future Work

Based on what was learnt while delivering this research, we believe that there is a number of interesting research areas to be explored. We separate these into short term goals which can be achieved relatively quick and long term goals, which require more long term research efforts.

7.2.1 Short Term Research Goals

We think the following goals are relative easy to complete from a short term perspective and therefore, they will be our primary research objects.

Incremental Data Transformation Primitives Current research in this area is focused on expanding the set of data transformation primitives to allow users more control in terms of these transformation operations. The current set of functions operate on the values that are generated by a single sensor. More complex queries require the fusion of sensor data with different analyses required across these new datasets.

Compact Query Language and Query Analyzer A set of compact query expressions/language could be provided where the query analyzer will first analyze the query expressions and then perform the appropriate data transformations where needed. The compact query language must provide a simple way to exploit well-understood relational semantics, and queries where performing relatively complex tasks should be easy and compact to write. The compact query language will also have sufficient constructs to capture a wide

variety of sensor data, and it should also be kept as simple as possible without sacrificing too much expressiveness. The query analyzer translates the Compact Query Language into well-formed XQuery expressions and perform data transformations where necessary.

7.2.2 Long Term Research Goals

The long term research goals of our research are to provide a more comprehensive query interface which benefits both advanced and unskilled users.

Visual Query Interface XQuery was designed to meet the needs of skilled database programmers. Its inherent complexity makes the new language unsuitable for unskilled users. The visual query interface allows non-IT users to query sensor data by simply dragging and dropping visual parameters without the need to learn complex query language like XQuery, then these visual annotations will be translated to the compact query language by a set of well-designed algorithms. The visual query interface itself must sufficiently support the sensor data model and all of its constructs, also is capable of querying and presenting all forms of sensor data.

Bibliography

[1] Artur, R., Angelo, M., Jose, C., et al. Innovations in Health Care Services: The CAALYX System. *International Journal of Health Geographics*, 2010.

[2] Augurusa, E., Braga, D., Campi A, Ceri, S. Design and implementation of a graphical interface to XQuery. *Proceedings ACM symposium on Applied computing*, 2003.

[3] Braga, D., Campi, A. A Graphical Environment to Query XML Data with XQuery. *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, 2003.

[4] Burgomaster, K., Howarth, K. and Gibala, M. Similar metabolic adaptations during exercise after low volume sprint interval and traditional endurance training in humans. *Journal of Physiology*, 586: 151-160, 2008.

[5] Balady, G.J., Williams, M.A., Ades, P.A., Bittner, V., Comoss, P., Foody, J.A.M., Franklin, B., Sanderson, B., and Southard, D. Core components of cardiac rehabilitation/secondary prevention programs: 2007 Update. *Circulation*, pp. 2675-2682, 2007.

[6] Certo, C.M. History of Cardiac Rehabilitation. *PHYS THER*, pp 1793-1795, 1985.

[7] Churcher, G., Foley, J., Bilchev, G., et al. Experiences applying Sensor Web Enablement to a practical Telecare application. *ISWPC*, 2008.

[8] Churcher, G., and Foley, J. Applying and Extending Sensor Web Enablement to a Telecare Sensor network Architecture. *ICST*, 2009.

[9] Camous, F., McCann, D., and Roantree, M. Capturing Personal Health Data from Wearable Sensors, *International Symposium on Applications and the Internet* (SAINT), IEEE Computer Society Press, pp. 153-156, 2008.

[10] Cappellari, P., Shi, J., Roantree, M., Tobin, C., and Moyna, N. Enabling Knowledge Extraction from Low Level Sensor Data. *Proceedings of the 22nd International Conference on Database and Expert Systems Applications (DEXA 2011)*, pp. 411-419, 2011.

[11] Devlic, A., Klintskog, E. Context retrieval and distribution in a mobile distributed environment. *Proceedings of the Third Workshop on Context Awareness for Proactive Systems (CAPS 2007)*, 2007.

[12] Devlic, A., Koziuk, M., and Horsman, W. Synthesizing Context for a Sports Domain on a Mobile Device. *EuroSSC*, LNCS vol. 5279, pp. 206-219, 2008.

[13] Evangelista-Filha, I. M. R., Laender, A. H. F., and Silva, A. S. Querying Semistructured Data By Example: The QSByE Interface. *Proceedings of the International Workshop on Information Integration on the Web* (WIIW'2001), April 9-11, pp. 156-163, Itaipava, Rio de Janeiro 2001.

[14] ERCIM News, Vol.79, (http://ercim-news.ercim.eu/en76), January 2009.

[15] ERCIM News Special theme: The Sensor Web. ERCIM News vol. 76, Janaury 2009.

[16] Fabin, J., Sylvie, R., Vincent, D., and Michel, C. IDEE: An Integrated and Interactive Data Exploration Environment Used for Ontology Design. *EKAW*, 2006.

[17] Gibala, MJ and Little, PJ. Short-term sprint interval versus traditional endurance training: similar initial adaptations in human skeletal muscle and exercise performance, Journal of Physiology 901-911, 2006.

[18] Gu, T., Pung, H.K., and Zhang, D. A Service-Oriented Middleware for Building context Aware Services. *Journal of Network and Computer Applications*, 28(1), pp. 1-18, Elsevier, 2005.

[19] Grust, T. Accelerating XPath Location Steps. *SIGMOD*, pp.109-120, 2002.

[20] Keane, S and Reilly, T. Analysis of work rates in Gaelic football. Australian Journal of Sports Science 100-102, 1993.

[21] Kenji, Y., Jun, I.T., Graham, J.W., and Peter, M. On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, pp. 275-300, 2004.

[22] Kawashima, H., Hirota, Y., Satake, S., and Imai, M. Met: A Real World Oriented Metadata Management System for Semantic Sensor Networks. *DMSN*, 2006.

[23] Kotidis, Y. Processing Proximity Queries in Sensor Networks. *DMSN*, pp. 1-6, 2006.

[24] Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. *OSDI*, pp. 131-146, 2002.

[25] Mike, B., George, P., Carl, R., and John, D. OGC Sensor Web Enablement: Overview and High Level Architecture, 2006.

[26] MonetDB - open source XML database. `http://monetdb.cwi.nl/`, 2008.

[27] Munroe, K.D., Papakonstantinou, Y.: BBQ: A Visual Interface for Integrated Browsing and Querying of XML. *VDB*, 2000.

[28] McCann, D. and Roantree, M. A Query Service for Raw Sensor Data. In *EuroSSC*, LNCS vol. 5741, Springer, pp. 38-50, 2009.

[29] O'Connor, G.T., Buring, J.E., Yusuf, S., Goldhaber, S.Z., Olmstead, E.M., Paffenbarger, R.S., and Hennekens, C.H. An overview of randomized trials of rehabilitation with exercise after myocardial infarction. *Circulation*, pp. 234-244, 1989.

[30] Palpanas, T., Papadopoulos, D., Kalogeraki, V., and Gunopulos, D. Distributed deviation detection in sensor networks. *SIGMOD*, vol. 32, no. 4, pp. 77-82, 2003.

[31] Papakonstantinou, Y., Petropoulos, M., Vassalos, V. QURSED: Querying and Reporting Semistructured Data. *ACM International Conference on Management of Data* (SIGMOD), 2002.

[32] Polar, `http://www.polar.fi`, 2008.

[33] Reilly, T., Energetics of high intensity exercise (soccer) with particular reference to fatigue. In *Journal of Sports Sciences*, vol. 15, pp. 257-263, 1997.

[34] Ramakrishnan, R., Gehrke, J. Query-By-Example (QBE), *Database Management Systesms*, Third Edition 1999.

[35] Roantree, M., McCann, D., and Moyna, N. Integrating Sensor Streams in pHealth Networks. *Proceedings of 14th International Conference on Parallel and Distributed Systems* (ICPADS 2008), IEEE Press, pp.320-327, 2008.

[36] Roantree, M., Shi, J., Cappellari, P., and Martin, F. O'Connor. Data Transformation and Query Management in Personal Health Sensor Networks. *Journal of Network and Computer Applications special issue: Intelligent Algorithms for Data-Centric Sensor Networks*, 2010.

[37] Roantree, M., Whelan, M., Shi, J., and Moyna, N. Using Sensor Networks to Measure Intensity in Sporting Activities. *Proceedings of QShine 2009*, LNICST Vol. 22, pp 598-612, Springer 2009.

[38] Shaeib, A., Cappellari, P. and Roantree, M. A Framework for Real-Time Context Provision in Ubiquitous Sensing Environments, *SISS 2010*, IEEE Press, 2010.

[39] Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., and Gunopulos, D. Online outlier detection in sensor data using non-parametric models. *VLDB*, pp 187-198, 2006.

[40] Strimpakou, M., Roussaki, I., Pils, C., Angermann, M., Robertson, P., and Anagnostou, M.. Context modeling and management in ambient-aware pervasive environments. *LoCA 2005*, LNCS 3479, pp 2-15, 2005.

[41] Stylus Studio Website, `http://www.stylusstudio.com`.

[42] Tang, Z., Kadiyska, Y., Li, H, Suciu, D and Brinkley, J.F. Dynamic XML Based Exchange of Relational Data: Application to the Human Brain Project. *Proceedings of American Medical Informatics Association Fall Symposium*, pp 649-653, 2003.

[43] W3C, `http://www.w3.org/`.

[44] Whitehouse, K., Zhao, F., and Liu, J. Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data. *EWSN*, LNCS 3868, pp 5-20, 2006.

[45] XQuery 1.0, `http://www.w3.org/TR/xpath-functions/`, 2010.

[46] Yao, Y., and Gehrke, J. Query processing for sensor networks. *CIDR*, Jan. 2003.

[47] Zimmermann, A., Lorenz, A., and Specht, M. Personalization and Context Management. *Journal of User Modeling and User-Adapted Interaction*, Volume 15, Numbers 3-4, pp 275-302, 2005.