A Framework for User-Assisted Knowledge Acquisition from Sensor Data

Kenneth Conroy, B.Sc., M.Sc.

A Dissertation submitted in fulfilment of the
requirements for the award of
Doctor of Philosophy (Ph.D.)

to

Dublin City University

Faculty of Engineering and Computing, School of Computing

Supervisor: Dr. Mark Roantree

August 2013

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and 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.: 58117016 Date: 21/08/2013

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Mark Roantree for his excellent supervision and for all his guidance and advice over the course of my PhD. I would like to thank Clarity and Science Foundation Ireland who provided the funding for this research.

I would also like to thank the members of the Interoperable Systems Group (ISG) and Clarity who I've worked with for their support and advice during my time at DCU. A special thanks to Greg May, who was essential in gathering valuable sensor data and interesting case studies.

I would also like to thank my friends and family for all of their support and encouragement.

Finally, I would like to thank my fiancée Doireann for her constant love, support and understanding.

This dissertation is dedicated to Doireann and my parents.

Contents

List of Figures

List of Tables

Abstract

The availability of a new generation of accurate, low-cost sensors to scientists has resulted in widespread deployment of these sensors in a variety of environments. Data generated by these devices are often in a raw, proprietary or unstructured format. As a result, it is difficult for scientists to analyse or query across various biological and physiological sensor data values. There exists both a physical-digital divide between sensor data with related real-world conditions, and a knowledge divide between the information needs of domain specialists. A key challenge is to bridge these divisions in order to allow scientists to make better decisions based on the sensed information. The goal of this research is to show that low level data collection resources such as sensors can be used for high level query expressions and knowledge extraction, without the need for expensive human-based operations. To achieve this goal, it was necessary to deliver a generic approach to enriching raw sensor data, providing information services to enable the end user to acquire knowledge from low-level sensor data and defining an integration framework for sensor data and related contextual information. As a result, key research questions of interpreting heterogeneous sensor data, enriching sensor data with contextual information, and integrating sensor data with related participant and environmental information, are tackled over the course of this dissertation.

Chapter 1

Introduction

1.1 Sensors and Sensor Networks

Advances in manufacturing technologies in recent times has led to the proliferation of low cost, portable and reliable sensors. These devices sense a wide variety of environmental, physiological and contextual factors and have been deployed in a variety of situations across multiple domains. The result is the ability to document multiple factors relating to some activity. Sensor deployments may consist of independent devices using different protocols to gather data for later off-line analysis. There are several advantages to the emergence of these sensors. They include the flexibility to deploy in inhospitable situations, replacing expensive solutions with a cheaper alternative and the gathering of massive volumes of potentially interesting information. Modern Wireless Sensor Networks (WSN) consist of spatially distributed autonomous sensors and have been deployed in a variety of scenarios such as ambient assisted living [37], environmental monitoring [76] and energy management applications [44]. This network of sensors can allow wireless communication between each node and may include real-time analysis of data using tools such as Infoshere Streams [28]. The research presented in this dissertation is more focussed on off-the-shelf sensors which do not com-

municate directly with each other, and are instead analysed post deployment with an aim to improving techniques for discovering new knowledge. The key advantage of sensors is the automated sensing of information, sometimes for data which is not feasible for a human observer to acquire (such as an accelerometer sensing at 30 Hertz), or information which could be identified but would require long-term observation (such as identifying when a human is running).

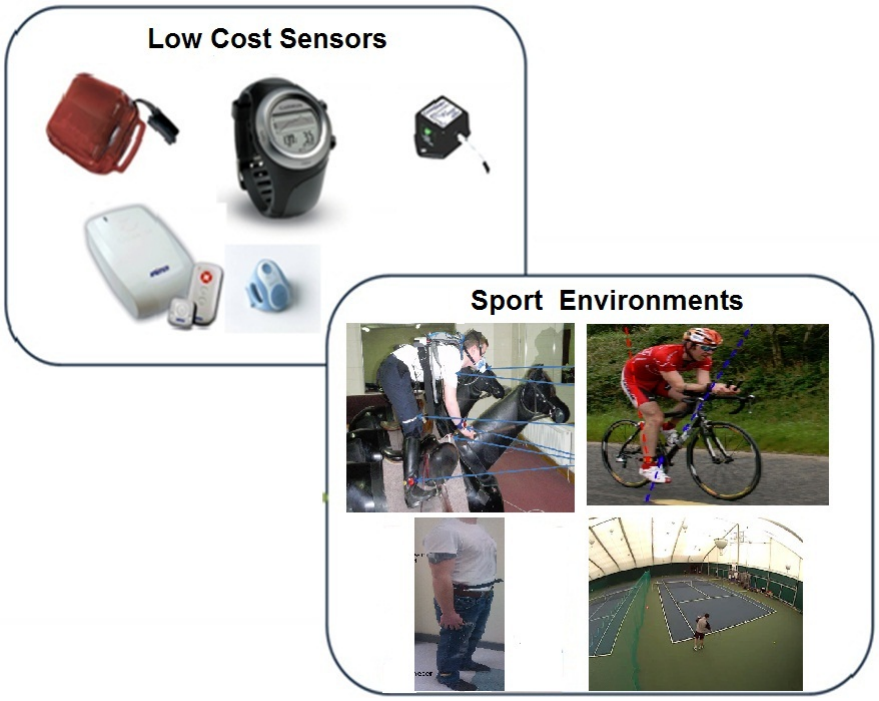

Figure 1.1: Diverse Sensors and Sporting Scenarios

The inexpensive nature of sensors has also contributed to their wide emergence. Replacing expensive lab equipment with simple sensors means more monitoring can take place and thus, improves efficiency. Large scale deployments - such as fitting out a house with Ambient Assisted Living (AAL) based sensors, or weather monitoring - are now feasible. New sensor technologies have also resulted in greater flexibility in monitoring certain situations. More experiments can be performed *in the field*, such as monitoring

the health of employees while they works, or observing a cyclist as he/she climbs a mountain, rather than trying to duplicate a real world environment in the laboratory. The combination of these advantages provided by the new generation of sensors has resulted in an explosion in their use in real-world scenarios.

1.2 Research Problem

As a consequence of the wider usage of sensors and their deployments, large volumes of data are being generated which scientists or other interested parties wish to analyse. This requires harvesting, normalisation, synchronisation, management of high volumes of data, and a mechanism for free form queries.

Typically, non-IT scientists deploying off-the-shelf sensors make use of proprietary software such as SPSS Statistics [29] or Microsoft Excel [16] for analysing sensor streams. This allows them to identify some patterns (by graphing data), providing the data streams are individually aligned, normalised and synchronised. This is routinely performed in cases where there is a very small (less than 10) number of sensors deployed, each of which is understood by the scientist working with the data. When another scientist or end user wishes to interact or query the sensor data, they must familiarise themselves with the data and its capabilities. There is a lack of clarity between what has been detected previously by the initial knowledge worker. For instance, the knowledge worker may have introduced a new column aggregating information from several other columns (in Excel). This new column is not clearly defined in a way an end user can understand. More advanced systems [42] [66] [53] make use of pre-defined ontological reasoning between sensors and their output and environment. However, such approaches are generally geared towards a specific domain and a defined set of sensors and environment. They are not scalable and are not widely

applicable. Every time a new form of analysis of experiment is devised, it requires somebody to write a new piece of software. This is not a feasible solution to domain experts and scientists who wish to express their information needs in a high-level format. For example, there is a query expressed by a sports scientist: *"Estimate the total energy expenditure of a Jockey n in training session x"*. This query, specified in English is not straightforward as it requires the analysis of the available sensor for that deployment, an accelerometer - with data such as that shown in Example 1.1.

Example 1.1 *Sample Extract of GT3X data (in csv format):*

x	y	z	lux	incline
488	198	47	1	1
467	186	49	2	1
494	155	80	1	1
479	272	22	2	1
192	187	24	1	1
492	163	45	2	1

This device (*GT3X accelerometer*) records movement in space, not physiological characteristics so the scientists wishes to combine data entries from this sensor with additional context not present in the data. Thus, there is a significant gap between the sensor data gathered in this deployment, and the real-world requirements of those wishing to utilise this data to answer queries such as this total energy expenditure estimation.

This research involved close collaboration with sport and health physiologists, who wish to analyse human actions and physiological attributes of people as they engage in sport and other activities (such as the personal health of an employee at work). There has also been collaboration with coaches who wish to segment training sessions or matches in order to automate the discovery of certain events such as a player serve in tennis and detecting energy expenditure during various activities. At a basic level, it is the use of sensors which can potentially generate information of interest,

which can then be analysed to detect more complex information or context. For instance, in the domain of a worker in a Search and Rescue helicopter, a requirement is to detect when that worker is *resting*. Determining this information requires analysing the position of the worker and their physiological properties of heart rate and respiration rate, details which are monitored by separate sensors. If properly understood and analysed, the sensor information can be used by scientists to identify this *resting* state and influence future experimental deployments.

The calculation of the total energy expenditure introduced earlier involves several iterations of calculation. For instance, another concept in the sports science domain: *work energy* must be determined first using accelerometer data before the detection of total energy expenditure can take place. Thus, any generic solution must include both read and write capabilities and allow the scientist to iteratively impose context on sensor data which can later be queried using a standard format. To provide the necessary functional requirements (later identified in chapter 2), a framework comprising a series of components which allow the interpretation, integration, updating and analysis of data is presented in this work.

There are two types of users wishing to query and interact with the data generated by sensors. A *knowledge worker* is a scientist who understands the sensor data and the application domain. They wish to analyse the data and use it to discover interesting, higher-level information. A second type of user is a scientist or other interested party who wishes to query the sensor data at a high level using simple means. These read-only queries do not require detailed knowledge of the sensor data in its initial form. It is the role of the knowledge worker to enable these queries by discovering the high-level information.

An ideal solution would also assist the scientist in identifying complex algorithms. For example, if a number of sensors are being used to estimate the

energy expenditure of a subject, some variable may be included as part of the calculation. Generic functions may improve the results by altering this variable (providing it is not a verified value). The scientists would like this feedback to help them make decisions and finalise algorithms.

1.3 Aims and Objectives

The hypothesis put forward in this research is that by using a framework to structure and contextualise low level data acquisition tools such as sensors, this information can be used for high level query expressions and knowledge extraction using basic user-interactions rather than expensive human-based operations.

The *Semantic Gap* is the gap between scientist information needs (or query requirements) and the data output from sensors. It is not possible to apply high level or free form queries on a set of semi-structured sensor data and therefore bridging the semantic gap is the key to verifying this hypothesis.

1.3.1 Research Questions

There are four main research questions which must be answered to bridge this gap between user needs and the raw data collected by sensors.

1. Is it possible to enable sensors such that their output can be queried at a high level?

2. How is it possible to facilitate the data transformation necessary to facilitate highly complex queries?

3. How can one enable the user to provide the missing semantics necessary to support higher level queries?

4. Can all of this be provided in a framework that allows heterogeneous sensor devices (sensors of varying types and configurations) used in

heterogeneous domains to meet the needs of different user types?

The first question illustrates the need for a framework to manage data output from sensors. How can one interpret the information based on the activities and participants involved? How does one define location or orientation of different sensors? In what format can one store the information gathered? What is an acceptable level of granularity for the domain specialists? Any solution will have to be adaptable to the diverse potential deployment environments.

The second research question addresses the need to facilitate data transformations to meet complex requirements. These requirements include detecting events, actions and specific body or equipment properties. How can one allow the user to define event detection algorithms?

The third research question to be answered is how to leverage user interaction to provide yet more information. How is user input interpreted, and what is the process for acquiring this input? In which cases are algorithms or sensor data related to one another?

The fourth research question asks how to ensure that these processes are generic? In other words, different sensors can be deployed in different environments without reprogramming. How can one ensure it will work across domains, and ensure it is customizable?

1.3.2 Goals

In order to meet these challenges, a framework is proposed for the automatic transformation of raw sensor data to embed user semantics, and through a small level of user interaction, provide a mechanism by which semantics can be embedded in queries and interpreted by the system to extract results from data. In order to deliver this overall contribution, it is necessary to deliver a degree of novelty in specific parts of the framework, addressing the research questions.

1. Implement a metadata driven process to structure sensor data output.

2. Implement a generic mechanism to provide extensibility for new levels of context and understanding of the data.

3. Support the scientist in discovering new information by combining multiple sources of sensor data.

4. Automate a process of discovering new information by combining user-defined metadata without any need for human interaction.

Achieving these goals will involve addressing issues such as context provision, semantic enablement, integration and user assisted updating and querying. The result should be a framework to allow user-interaction for the definition of queries of differing complexities, which when applied to multiple sources of both sensed and non-sensed data, detect events or actions and assist the user with improving their algorithms.

Discussions with knowledge workers and users from a variety of domains have identified a set of functional requirements for the solution. Related work will be evaluated against this criteria in chapter 2, as will the framework described in this research (in chapter 8). A detailed discussion of the different case studies and the reasoning behind the following functions is provided in chapter 3.

- Setup: Allow on-the-fly setup of new sensors, subjects and deployments pertaining to a sensor deployment
- Query: Query the sensed data using different criteria
- On-the-fly Context: Integrate new information discovered as context within the sensor dataset
- KW Interaction: Allow non-IT knowledge workers to query and add new context or knowledge
- Incremental Enrichment: Allow the incremental construction of queries and context based on previously discovered context/knowledge.

1.3.3 Scope and Contributions

The framework defined by this research is to provide the necessary functionality for a wide range of off-the-shelf sensor devices when deployed in a wide range of environments. There is an assumption that those wishing to lead the deployment of these sensors (such as scientists) understand their function and how they can meet their individual requirements using the data output from these sensors. Once necessary transformations of data are performed by such a knowledge worker, the data format itself describes the information contained and thus, those who are less knowledgeable of the data and the deployment description will be able to query for their own requirements.

As will be discussed throughout this dissertation, meeting the requirements of a diverse set of users, while maintaining support for heterogeneous sensor devices and domains led to a framework which provides the following contributions to the state of the art.

1. Provision of a generic framework (AREA) for the semantic enrichment of low-level sensor data

2. A generic, rule-based mechanism for allowing a knowledge worker to semantically enrich the data with new terms with appropriate data values. (both user and system driven)

3. Provision of an environment that facilitates free-form queries

1.4 Thesis Structure

This dissertation will demonstrate how the goals outlined in this chapter have been achieved, and by doing so, new levels of functionality have been provided to different types of end users who deploy sensor technology. The remainder of this thesis is structured as follows:

Chapter 2

This chapter presents a detailed analysis of the related research in the fields of contextual enrichment, context definition and efforts to manage and combine data sources. Each approach is evaluated against the functional requirements identified in section 1.3.2.

Chapter 3

In order to rigorously test the framework presented in this dissertation, a number of end-users in a number of different domains were engaged. They were asked to specify their requirements which are described in chapter 3. In each case, an overview of the sensors deployed, the deployment environment and how these users currently assess the performance and behaviours of subjects during these activities in an effort to improve their capabilities is provided. In explaining the differences and commonalities involved for different domains, this chapter aims to show how a data analysis framework requires both a combination of automated system operations and user interaction.

Chapter 4

The AREA framework, with the goal of managing both the information needs of the knowledge workers while facilitating the heterogeneous environments posed by the different case studies is detailed in chapter 4. Each component, tasked with providing structure, integrating and applying context to sensor data is described. Chapter 4 also outlines the user controlled transformation and automated activity analysis functionality which are fully described in chapters 5 and 6.

Chapter 5

As one of the core components in the framework, the evaluation of AREAs user controlled transformation functionality needs to be described in more detail. This part of the process exploits user-defined *update* and *extension queries* to the dataset which facilitates the incorporation of context and interesting events and observations. Chapter 5 describes how these queries are applied on scenario datasets, thus altering both content and structure, and enabling subsequent querying for complex information using simple methods.

Chapter 6

Chapter 6 details the approach to automated activity analysis. The generic functions used to analyse the data and detect potentially interesting information are discussed. Chapter 6 also describes the automated mechanism devised to analyse the metadata (generated previously by the user) using techniques such as clustering to potentially improve algorithms proposed by the scientists.

Chapter 7

In this chapter each of the case studies are revisited in order to evaluate the functionality and flexibility of the AREA framework.

Chapter 8

In chapter 8, the AREA framework is analysed and evaluated against the functional requirements identified in this chapter (section 1.3.2). Chapter 8 also includes an analysis of the time taken to enrich sensor data and to determine if this time is reasonable.

Chapter 9

The final chapter presents the conclusions and reassess the hypothesis and research questions. Limitations of the AREA framework are discussed as is future work which could be undertaken to broaden the overall impact of this work and extend AREA with new functionality.

Chapter 2

Related Research

2.1 Overview

The aim of this chapter is to evaluate related work against the list of functional requirements identified in chapter 1. In each case, a description of the approach taken is provided along with a critical evaluation with respect to the support for these requirements. The approaches are structured into the following different sections. Research based on ontological approaches to interpreting sensor data is presented in section 2.2. These solutions rely on a hierarchical structure to determine context, and in some cases allow the discovery of higher-level knowledge [55] [66] [6]. Section 2.3 describes a number of solutions developed for context-aware environments, representing a large area of active research [78] [72]. Section 2.4 then describes efforts in the object-role modeling of context, focusing on the Context Modeling Language (CML) approach [23]. Following this, section 2.5.2 details a number of efforts to representing sensor data in a number of domain-specific applications. Section 2.6 presents the standardization efforts of bodies such as the Open Geospatial Consortium to sensor data, in particular aspects of Sensor Web Enablement.

What is required is a framework which allows a user to adapt their analy-

sis, customise their approach and fine tune algorithms of analysis of sensor data. An approach which requires a new system to be built for each and every domain or application is not suitable. Instead, a system which is in place before the user arrives with their information needs and deployment characteristics will satisfy the requirements of dynamic applications. Section 2.7 summarizes related work and includes a comparison of the functionality of each system with respect to the requirements of a complete solution.

2.2 Ontological Approaches

As summarised in chapter 1, context can be any information used to characterise the situation of service users, which includes information about users, their environment or their tasks [14]. Ontologies can enable applications to interpret contexts based on semantics. An ontology's hierarchical structure lets developers reuse domain ontologies in describing context and build a practical context model without starting from scratch. Because contexts described in ontologies have explicit semantic representations, semantic web tools such as federated query, reasoning and knowledge bases can support context interpretation [65]. There have been a number of ontology based approaches to dealing with multiple sensor data files. The first analysed is the Context Ontology Language (CoOL).

2.2.1 CoOL: Context Ontology Language

CoOL is derived from an *Aspect-Scale-Concept* model, and is used to enable context-awareness and contextual interoperability [55]. Specifically, it is designed as a generic mechanism for modelling the context of entities such as person, place or objects and the states they can reach through a changing environment. The context modelling approach aims to close the *formality gap*, that is the lack of a well designed model to describe contextual facts and

relationships. To do this, CoOL uses ontologies to describe contextual facts and interrelationships. The core concepts of context modelling are Aspect, Scale and Concept. An Aspect is a set of one or more scales, a Scale is an unordered set of objects defining a valid range of context information and a Concept represents context information as content with some metadata. A useful function of CoOL is the ability to scale measurements automatically (such as conversion of nautical miles to kilometres). The formality achieved using ontologies ensures good support for automated interpretation capabilities of an implementation of the system. It can be used with a service model, and by extension, make service interactions on that model context-aware. The Aspect-Scale-Concept model is shown in Figure 2.1 (taken from [55]).

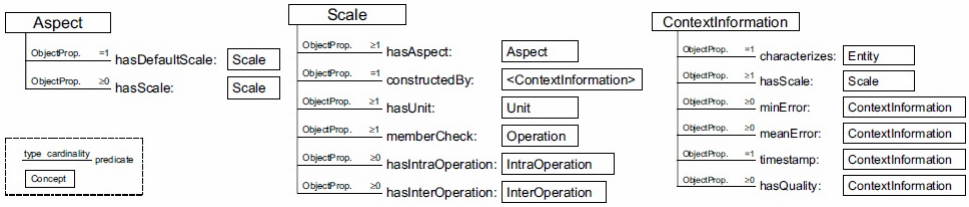

Figure 2.1: Aspect-Scale-Concept model for CoOL [55]

Critical Evaluation

The focus of CoOL is on formal semantics and automatic services rather than enabling a user to customise transformations of the data. There is no mechanism for altering or updating context either through interaction with the user or through some automated process. Any alteration to context requires programming which goes beyond the capabilities of a typical user. It is also targeted more at context-aware systems where dynamically changing requirements are less of a concern, as are adaptation capabilities to suit changing environments or domains. Thus, while a useful tool for developers, it cannot adapt to requirements or domains which are not initially defined.

2.2.2 CONtext ONtology (CONON)

CONON is a Web Ontology Language (OWL) encoded context ontology for modelling context in pervasive computing environments [66]. The application domain is a smart phone which can alter its behaviours (ringtone, call forwarding, etc.) based on the preferences of the user. General context is provided in an upper context ontology level, and extensibility is provided by adding domain-specific context hierarchically [66]. While acknowledging the difficulty in providing support for any contextual information, entities such as location, user, activity and computational are used as core context within the upper level ontology. This provides the high-level ontology from which individual domain ontologies can be extended.

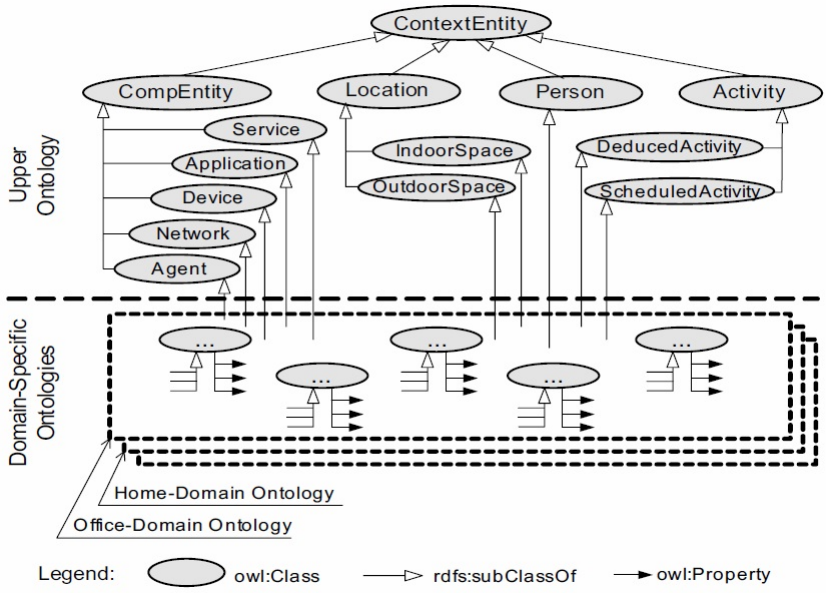

Figure 2.2: Partial Definition of CONON Ontology [66]

Figure 2.2 (taken from [66]) shows the upper and domain-specific ontologies in CONON. The upper ontology is a high-level ontology capturing general features of the entities. The domain-specific ontology are different environments where the smartphone might be used - such as home or office. A

different set of rules can apply based on these different locations. A key aspect of their system is context reasoning which acts to check the consistency of context and to deduce high-level implicit context from low-level explicit context [66]. This means low-level sensor data can be used to derive rules which correspond to some high-level information. CONON achieves this using first order predicates, where a (`subject, verb, object`) triple defines context information. The context reasoning can be either ontology reasoning using description logic, or user-defined reasoning using first-order logic.

Example 2.1 *Situation: Cooking [66]*

```
(?u locatedIn Kitchen) /\ (ElectricOven locatedIn Kitchen) /\
(ElectricOven status ON) => (?u situation COOKING)
```

Example 2.1 shows the user defined context reasoning corresponding to a *situation* of *cooking*. In this example, if the user is `located in` the Kitchen, the oven is also `located in` the Kitchen, and the oven is turned on (`status ON`), a situation of *cooking* is identified. Based on this situation, the phone may have its ring volume increased, or be set to ignore incoming calls if these rules are specified by the developer.

Critical Evaluation

CONON is aimed at a location-based smart phone domain where all aspects of a system are continuously in communication with each other, in (close to) real time. As a result, it is designed to react to events as they happen in real time. Much of the focus is on performance rather than functionality ensuring results are acquired in close to real time. While the upper ontology is a useful baseline for context modelling, the system would need to be reprogrammed considerably in order to allow the user to define context. The approach to context reasoning is suitable for many context-aware applications, but for detailed offline requirements, such as allowing customisation

of algorithms or updating data with new context, CONON is not applicable. While the iterative definition of more complex context reasoning is possible, CONON cannot perform iterative transformations of the data on which further context reasoning is based. For example, a query which requires the calculation of energy expenditure based on three input parameters cannot be implemented using CONON. Some simple read-only queries are possible, but would require further programming and development of a user interface.

2.2.3 SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications

The Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) describes an OWL-based modular vocabulary for intelligent agents and their beliefs, desires and intentions as well as time, space, events, user profiles, actions and policies [6]. The prototype application includes a smart meeting room environment, which utilises bluetooth devices to monitor location of participants, light sensors, and schedule and meeting information to determine when a presentation can begin. Another application is a smartphone (PDA) which interacts with other devices to discover information to ensure a schedule or readline is met. The ontology consists of two distinct ontologies: Core and Extension. Core attempts to define a generic vocabulary to different pervasive computing applications such as *Person*, *Policies* and *Events*. A Policy is a set of rules that are specified by a user or entity to restrict or guide the execution of actions [6]. This is a mechanism for describing how low-level system behaviours can be adjusted using high-level rules. In SOUPA, rules can be set to permit or forbid the execution of actions. Domain specific or application information is stored in the the *extension ontology*.

The goal of SOUPA is to present a shared ontology which can be used by developers to reduce the requirement of defining an ontology from its basic

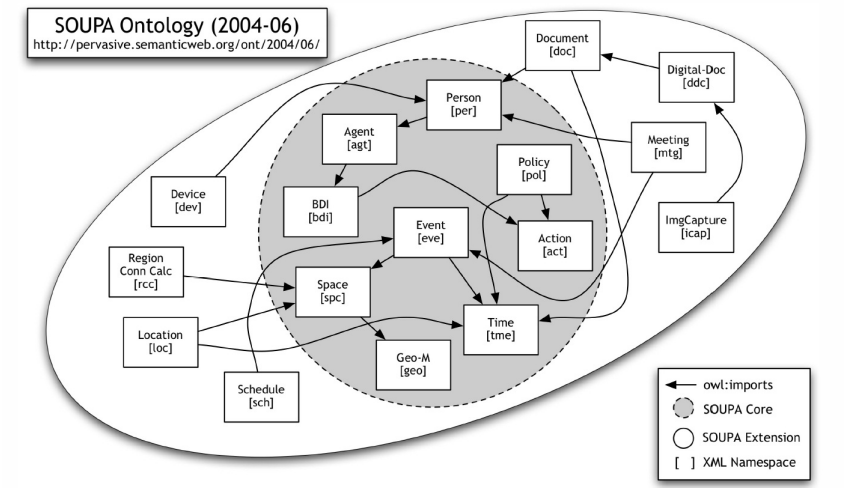

Figure 2.3: The SOUPA Ontology [6]

elements, and instead focus their efforts on the implementation details of their application. In other words, SOUPA provides the basics for ontology building. SOUPA provides a good starting point for an application developer in the context-aware domain, from which application specific context can be extended. Figure 2.3 (taken from [6]) shows the SOUPA ontology with core and extension ontologies clearly illustrated.

Critical Evaluation

For an end user, the functionality to incrementally add context to the model is not provided by SOUPA. Their approach is suitable for defining context for different domains from the beginning of development, controlled by an end user. New domains requires re-working the system from the Core base, adding the required context for that domain. If SOUPA was extended for use in sporting applications, each sporting term and state would need to be pre-defined as part of a SOUPA extension. There is no support for allowing the user to customise their approach by adding their own terms. Results from previously identified knowledge cannot effect the development of rules, and there is little support for complex event definition, such as mathemat-

ical equations involving multiple data elements as parameters. Complex events can only be identified if the developer or programmer, as opposed to a knowledge worker, knows from the beginning of development which conditions correspond to an event for every information need. As a result, it is not suited to dynamic deployment environments where the information needs of the user evolve over time. SOUPA is appropriate as a tool for developers wishing to define an ontology for a domain or set of domains where all relevant information is known from the initial set of available data.

2.2.4 OntoSensor

The research in [42] and [43] describes the approach to building OntoSensor, a prototype sensor knowledge repository compatible with evolving sensor web infrastructure. They propose OntoSensor as a component of the support knowledge base for military applications - specifically those which make use of heterogeneous data sources to counteract enemy tactics and strategies. These data sources include high-level information obtained by satellite imagery. OntoSensor includes definitions of concepts and properties adopted in part from SensorML [48], the Web Ontology Language (OWL) [39] and extensions to IEEE Suggested Upper Merged Ontology (SUMO) [56]. Sensor ontology's are used to establish a terminology for sensors, their properties, capabilities and services. The authors have implemented an ontology with basic querying capabilities using Protege 2000 [40] and Prolog.
In [1], the authors describe a semantic model for heterogeneous sensor data representation. Figure 2.4 (taken from [42]) shows an OntoSensor hierarchy. The researchers aim to establish a terminology for sensors, properties, capabilities and services. A sensor data ontology is created based on the Sensor web Enablement (SWE) and SensorML data component models. Semantic relationships and operational constraints are deployed in a uniform structure to describe the sensor data. OntoSensor has a number of advantages,

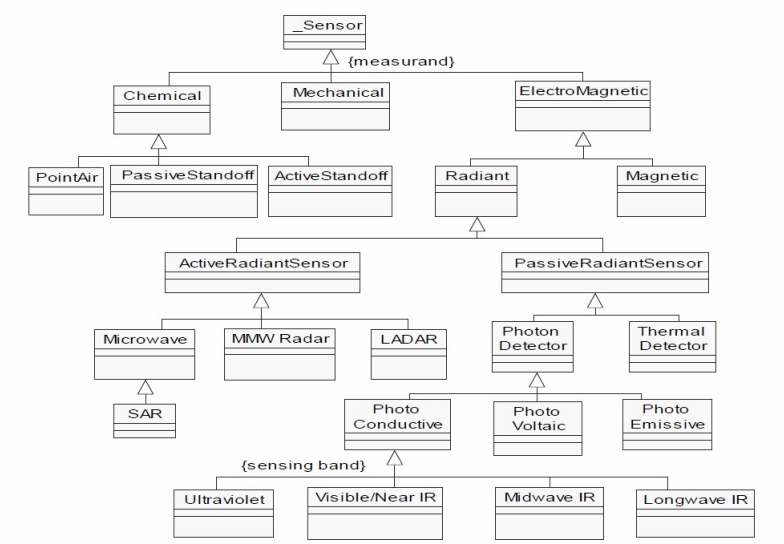

Figure 2.4: OntoSensor Sensor Hierarchy [42]

including self-descriptive metadata embedded in the descriptions, which can be used in various sensor discovery and reasoning applications. OntoSensor illustrates a semantic approach to sensor description and provides an extensive knowledge model [1]. This model allows machines to process and interpret the emerging semantics to create intelligent sensor network applications.

Critical Evaluation

Despite providing a semantic approach to sensor description, OntoSensor has no facility for an end user, or knowledge worker to update existing context using an iterative process. A knowledge worker (for each domain) must be involved from the beginning of development in order to ensure the initial information needs can be achieved. As a result, it is not sufficient to meet the goal of allowing generic or flexible deployment's context to be updated by a knowledge worker where necessary during analysis. The ultimate result is the development of different systems for different domains and requirements.

2.3 Context-Aware Systems

2.3.1 CXMS

In [78], CXMS is introduced as a base framework with tools for designing context managing applications. This framework integrates user modelling and context modelling, combining personalisation and contextualisation. The target domains are ubiquitous environments - thus location forms a core part of determining context. An example application is an advertisement board in a train station which reacts to sensed properties such as time, noise levels and train times in order to decide what to display. To address the issues of strong dependence of domain and a lack of standard interfaces or representation of context, CXMS provides a general framework where context knowledge acquired, domain inference and adaptive methods for personalisation and contextualisation can be combined. This enables the user to define application information sources, sensors, parameters and rules for adaptive behaviours. The user can therefore define when and how a system is to react to new information such as behaviours.
Context-aware systems can be developed and maintained by CXMS, with complex technical details hidden from developers and end users. CXMS's toolkit has a layered approach to defining a context-aware system. Simple sets of rules set by the end user can lead to complex output following development.

Critical Evaluation

While this framework addresses many of the problems involved with domain dependant system development, it still requires the input of domain specific or end user defined rules at the development stage, as opposed to reacting to the changing information needs of the users. What CXMS provides is a structured approach to the development of a context-aware system. It is

not designed to be modified once a system has been developed (apart from maintaining and altering an existing context-aware system). Thus, a new source of information or sensor data cannot be easily integrated by the user throughout its use. Similarly, the adapting analysis of the user cannot be met without re-engineering a new system. It is also heavily dependant on location based context (due to its focus on ubiquitous environments).

2.3.2 AlarmNet

Ambient Assisted Living (AAL) is a topical application area in the context-aware domain. AAL is a term used to describe technologies which help to extend the time where older people can live in their home environment. AlarmNet [72] is a context-aware wireless sensor network for the AAL domain and residential monitoring applications. It is a monitoring system for AAL environments, relying on sensors such as temperature, dust, light and motion, to improve the health of residents. It provides a 2-way flow of data and analysis between front and back ends to enable context-aware protocols to suit individual living patterns. The AlarmNet architecture is illustrated in Figure 2.5 (taken from [72]). AlarmNet integrates environmental, physiological and activity sensors in a scalable heterogeneous architecture. The SenQ Query protocol [75] provides AlarmNet with real time access to data and in-network processing. Activity information is fed back into the system to identify patterns and aid with power management and dynamic privacy policies. The key contributions of AlarmNet is an extensive heterogeneous network middleware for widescale deployment, integrating devices, backend systems, online analysis and user interfaces. In the Ambient Assisted Living domain, the benefits are power management, real-time communications, extensibility and learning patterns.

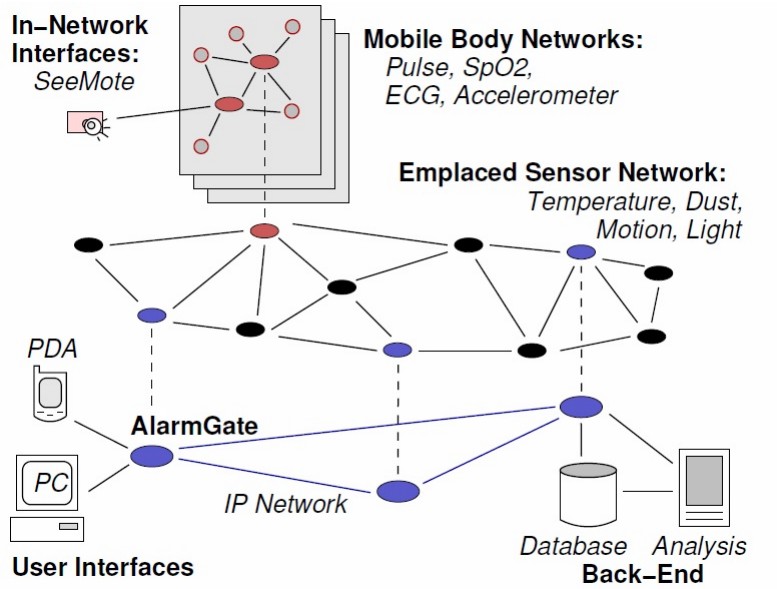

Figure 2.5: The AlarmNet Architecture [72]

Critical Evaluation

Due to its application domain, context gathered is subject to privacy constraints, and updating the kind of data to be observed or identified is a complex process. AlarmNet provides a basic User Interface for interacting with and setting parameters to basic functions. However, this is limited to basic settings, such as setting alarms or reminders and does not provide a mechanism for defining concepts or complex events. The user interface may be suitable for some scientists as it provides basic functionality, but is unsuitable for many knowledge workers who wish to extend the data with new information or context based on lower-level sources. In addition, AlarmNet does not allow for mathematical functions or specialised notation or text mark-up based on conditions.

2.3.3 User Interface Context Model

The research in [68] [69] presents a system which supports an end user interface for describing context dependant services in the field of Ambient

Assisted Living (AAL). In particular, a smart home which can have its functionality extended using a user interface. A scenario is introduced where a user wishes to add reminders to their AAL services and also requests to turn on the light when the room they are in is dark while the user is not sleeping. Due to the target demographic of those using the system, it had to be usable without context modelling expertise. To achieve this, they devised a layered context model. As with most context-aware applications, AAL is dependant on a smart space. The three layered context model is illustrated in Figure 2.6 (taken from [68]). The model includes context entities, dimensions, attributes and relations. In the *User Interface Layer*, alteration of predefined situation descriptions is permitted, using a taxonomy of terms. This model does have advantages of usability, specifically by non-scientists in the field of AAL.

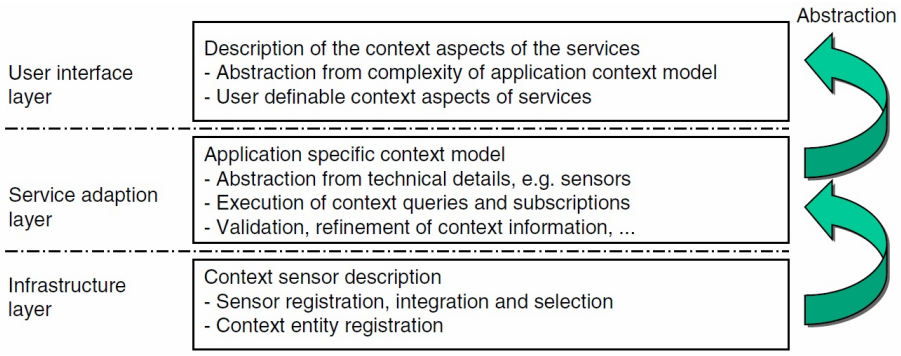

Figure 2.6: Three layered context model [68]

Critical Evaluation

This system allows corrections and alterations but this is only useful where the entire context model has been defined in advance. The target user is not assumed to have scientific knowledge, or knowledge about the data so there is also no mechanism for adding context dynamically. It serves more as an expert system rather than a decision support system that interacts with the

user.

2.3.4 SCAFOS

In [34], [33] the authors present SCAFOS, a model-based approach for distributed heterogeneous environments. The application domain is real-time context-aware environment, where users are monitored and, for instance, reminded to do tasks if in the same room as someone required to perform those tasks. They propose a *deductive* component as a bridge between the application layer abstraction and sensor data [33]. This reasons with low-level sensor data to deduce high-level abstract knowledge, which can then be used by the application layer. As with previous systems designed for context-aware environments, the focus is on linking location with users. Arguments such as `UserAtPosition, UserInRegion` are abstracted using first-order logic evaluated as true or false. Based on these observed states, more abstract information can be derived by using combinations of assertations.

Critical Evaluation

The notation used to derive abstract, high-level information is not easily understood. While its functionality can be used in the application layer, a new system must be built for each domain. The context-aware focus means it is highly focused on location and smart space environments and a real-time data analysis. There is no support to alter existing context data and thus, customisation of information needs is limited to the initial setup.

2.4 Object Role Modelling

2.4.1 CML: Context Modelling Language

The Context Modelling Language (CML) is an object role modelling approach to query processing and reasoning [23]. Context-aware applications, such as a commications tool which suggests contact information for someone most appropriate based on aspects such as priority, topic and list of people available [24] [25]. CML is based on the *Object-Role Modelling (ORM)* conceptual modelling method and provides a graphical notation designed to support the software engineer in analysing and formally specifying the context requirements of a context-aware application [3]. CML supports the evaluation of simple assertions. This includes the ability to support querying over uncertain information using three-valued logic (triples). A grammar for high-level abstraction of context in real world conditions is defined in [23]. This context is denoted as a *Situation* by Henricksen [23], defined using predicate logic. Expressions are either equalities, inequalities or assertions and high-level context can be combined to form more complex logical expressions. A CML Situation for identifying if a channel can be used is shown below in Example 2.2 (taken from [24]). This situation is taken from a communications application. *CanUseChannel* is satisfied for a person and channel when all devices required to use that channel are located near to the person and the person is permitted to use those devices.

Example 2.2 *CML Situation*

```
CanUseChannel(person, channel)
: \forall  device  RequiresDevice[channel, device]
LocatedNear[person, device]  PermittedT oUse[person,device]
```

The graphical notation (as shown in Figure 2.7 [24]) supports the analysis and design of context requirements of a context-aware application. The grammar for high-level context abstractions is another advantage in cases

where CML provides the necessary reasoning model, and CML also provides support for evaluating imperfect and historical data.

The research presented by Henricksen in [26] focusses on addressing the need to improve ontological context-aware systems and their general inability to combine ontology concepts with context modelling and reasoning. In a hybrid approach, they attempt to map their previously defined context model (CML) to the interoperable ontology-based reasoning approaches. In doing so, they aim to eliminate the shortcomings of ontological approaches, in particular addressing the immature standards and tools available for supporting ontology-based reasoning and the problems associated with reasoning over ontology languages such as Object Query Language (OQL) and the Semantic Web Rule Language (SWRL) [46]. Another weakness identified by Henricksen is the lack of support for reasoning over imperfect information. To combine the advantages of their CML approach with ontologies, their hybrid solution is presented in [26].

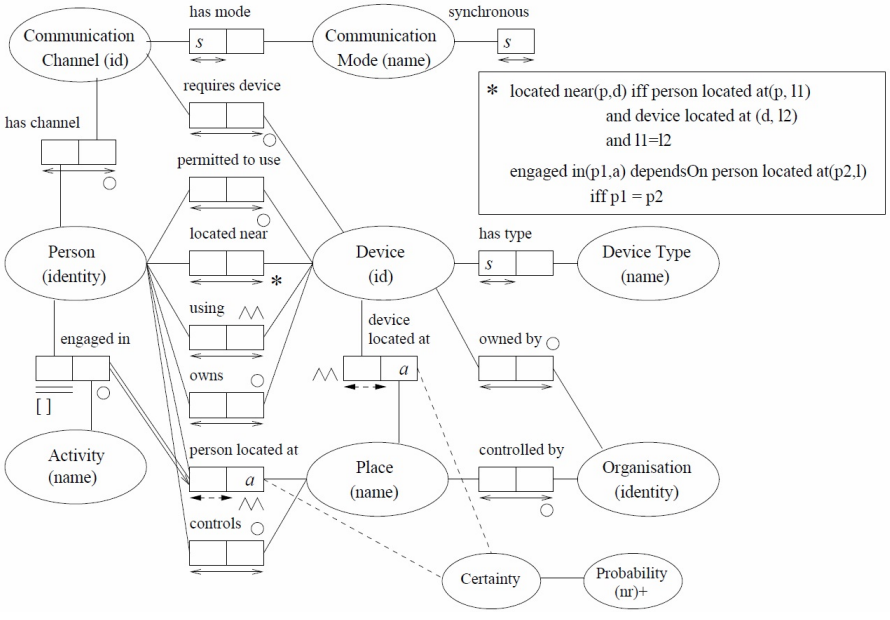

Figure 2.7: Modelling context for communication channel using CML [24]

They examined the possibility of converting their CML Situation definitions (such as *CanUseChannel* in Example 2.2) to an SWRL rule. As previously explained, these *Situations* derive additional context from context previously defined. The resulting SWRL rule is much more complex, not user friendly and according to Henricksen, provides no real benefit to the system compared to the CML notation. As a result, they dismiss using SWRL rules for notating situation abstractions. Instead, the ontological aspect of the hybrid system is more focused on reasoning about context models - checking for errors, inconsistencies and supporting interoperation with relationships.

Critical Evaluation

The focus of CML is in context modelling and reasoning. While context modelling is expressive and can be applied in any domain, it is in the hands of the developer and not the knowledge worker. This is crucial as many projects and applications will involve both changing information needs and non-IT users. Incremental updates are limited to combined high-level assertations. CML is focused on providing solutions to specific domains or applications rather than a flexible, user-driven solution. An improvement to CML proposed in [26], and takes a hybrid approach which is now discussed. For context-aware applications, the context reasoning aspect (the key aspect of the hybrid approach) is useful as the required functionality is built-in. However, as knowledge workers require a framework for the gradual development of context, the verbose and non user-friendly aspect of using SWRL is unsuitable. The proposal for supporting interoperation through relationships is planned, but not developed or specified. In the cases examined in this research (chapter 3), a knowledge worker knows the type of information they wish to discover but not necessarily the approach required to achieve this. Feedback from previous queries or analysis of multiple context may be required. Finally, the CML approach does not permit the addition or

extension of context by users.

2.5 Domain Specific Approaches

2.5.1 GeoAIDA

In [5], the authors describe the GeoAIDA system, which aims to provide intelligent, concise and flexible control of *scene* interpretation by using semantic scene descriptions. The system's target domain is geographic and mapping applications. It aims to allow the structural definition of scenes, generate hierarchical thematic maps and analyse multiple sensors. They include external objects related to the analysis and also include previous knowledge of geographic information systems. They allow for the integration of different image processing operators and aim to have a clear and concise structure for the system. A key aspect of GeoAIDA is the semantic network, which is knowledge provided to the system in order to help interpret the input data. The semantic network contains nodes and edges where nodes represent objects and edges represent relations between objects. During analysis, the generic sensor network generates an instance network. Analysis takes place in two steps, a bottom up step and a top down step. An overview of the system is shown in Figure 2.8 (taken from [5]), where the top-down analysis is model based and generates a network of hypothesis based on the semantic network. The bottom-up step groups these hypothesis into verifications and falsifications.

Critical Evaluation

While its deployment is generic for any *scene*, GeoAIDA uses existing image processing operations and is thus, limited to the image processing domain. There is limited interaction between the knowledge worker and the system is essentially a solution specifically for the image processing domain. Thus, the

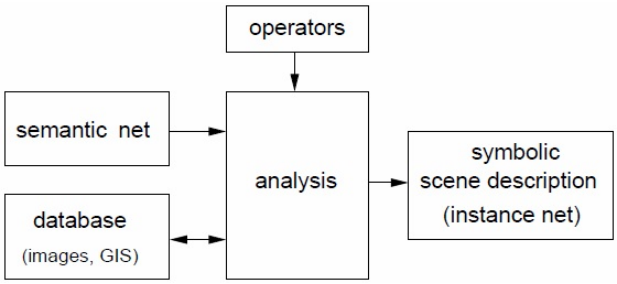

Figure 2.8: GeoAIDA System Components [5]

two main issues with GeoAIDA are its limitation to the geographic/mapping domain and the semantic network is pre-defined and cannot be changed by the knowledge worker. There is also no facility for incrementally improving or building hypothesis or algorithms, even within the mapping domain.

2.5.2 Other Domain Specific Context

In [77], the authors represent context with varying granularity with a tuple consisting of an Resource Description Framework (RDF) [41] triple defining the relationship, a lifespan and a *conditional confidence value*. The main aim was to reduce uncertainty in context integration. The method used to achieve this was to combining multiple sources of information and using *Bayes theorem* in a data-mining approach to calculate conditional confidence values. In [32], the authors present a simple approach to determining the context in which two accelerometers are present in an ubiquitous environment. The research is based on modelling the mechanical characteristics of the human body to determine if two low-cost accelerometers are carried by the same person. In [45], a framework for real-time context provision in ubiquitous sensing environments is presented. This metadata-based approach structures context in a relational database. The research was applied in an ubiquitous environment where context is defined as zones, where each zone is a subset of a smart space. The event or feature detection approaches detailed illustrate only a small subset of the research in the field of domain

specific event or action detection.

Critical Evaluation

Due to the diversity of initial requirements, domain based systems are not sufficiently abstract for changing requirements. This also applies to the ontology approaches where the ontology is initially hard-coded to work for a specific number of instances. These approaches are highly specialised, dependant on specific hardware setups and not suited to changes in domain or algorithms. They also do not provide a facility for end users to define the features of interest for a certain deployment. The wide variety of requirements and domains required by knowledge workers exclude a domain-based approach. Extending a solution to be able to utilise any sensor, any environment and any domain or event provides a key strength of flexibility for the knowledge worker. It allows them to adjust algorithms based on their ever changing needs, driven by their scientific advances based on previous results.

2.6 Standardization

A number of standardization bodies have devised standards for the representation and monmitoring of sensor data, primarily with the aim of enabling interoperability across the sensor web. The Sensor Web Enablement (SWE) group is a working group of the Open Geospatial Consortium (OGC) [57] and is responsible for specifying interoperable interfaces and metadata encodings for the integration of heterogeneous sensor data. The Sensor Interface Standards (IEEE 1451) [35] have been defined by IEEE, involving some collaboration with the OGC. They provide low level physical sensor interfaces. The Emergency Interoperability Consortium of OASIS have also provided standards including the Common Alerting Protocol (CAP) [9] and

the Emergency Data Exchange Language (EDXL) [17]. ISO have defined the Sensor Data Model for Impages and Gridded Data (ISO 19130) [15]. This section describes some of the details of these standards along with a query processor which can run on the nodes of a sensor network.

2.6.1 Sensor Web Enablement

OGC members have developed and tested a suite of 7 candidate specifications which act as a framework to web accessible sensor networks and archived sensor data that can be discovered, accessed and controlled using open standard protocols and interfaces [4]. Each standard tackles a particular aspect of managing sensor networks, such as the descriptions of the sensors which are defined by the Sensor Model Language (SensorML) [49]. XML is a key part of the infrastructure, with XML Schemas defined using SWE standards, used to publish formal descriptions of a sensors capabilities, location and interfaces. Sensor Web Enablement has been applied in a number of applications [7], [8]. A real world deployment of the SWE technology, in parallel with an existing approach built around a Java Messaging Service (JMS) [30] middleware, is performed in order to evaluate the effectiveness of the SWE approach.

The SWE approach has a number of advantages. The SensorML, Observations and Measurements (O&M) and Sensor Observation Service (SOS) provide a flexible mechanism for describing and querying sensor data. SensorML provides a standard model to describing sensors. O&M can then be used to enable sensor data. This is a useful technology for building shared sensor information spaces [7]. In [8] the authors note SWE can enable sensor data to be virtualised, providing a common, self-describing data format and access protocol. In addition, the emergence of domain-agnostic toolkits indicate that the overhead in creating new applications based on SWE can be lowered.

A number of weaknesses of the SWE approach were described in [7]. Problems centred on the immaturity of the standards, such as the Sensor Alert Service (SAS) lacking a stable implementation. In addition, the SOS does not directly provide support for streaming data to clients. Additional middleware must be designed to facilitate this. The sensors are designed for a static geographic location, and problems associated with context arise when the sensor is dynamic. The SOS interface also provides no means of access control, so sensitive data must be encrypted, with key management procedures applied externally to the system.
As the focus on this work is a functioning system in which the knowledge worker can drive the contextual enrichment process, a verbose representation of the sensor data does not offer substantial benefits. A simplified, streamlined description of sensors provides the user a quick setup for new sensors in different domains, which can be easily incorporated into new deployments when required.

TinyDB

TinyDB [60] is a distributed query processor that runs on each of the nodes of a sensor network. It allows the functionality of a traditional database, such as the ability to select, join, project and aggregate data. As it is based on each node it prioritises efficiency over functionality, ensuring results can be obtained in real time. TinyDB uses a sensor networks smart sensors to customise sampling rates among other properties to increase accuracy [36]. The approach in this dissertation is not focused on maintaining real-time query capabilities. Instead, the incremental addition and analysis of context will provide the necessary knowledge for subsequent querying by users.

2.7 Summary

Tabe 2.1 summarises the approaches described in this chapter, showing support for each of the functional requirements first described in chapter 1. In each case, a tick represents support, or partial support for that functional requirement.

- Setup: Allow on-the-fly setup of new sensors, subjects and deployments pertaining to a sensor deployment
- Query: Query the sensed data using different criteria
- On-the-fly Context: Integrate new information discovered as context within the sensor dataset
- Knowledge Worker (KW) Interaction: Allow non-IT knowledge workers to query and add new context or knowledge
- Incremental Enrichment: Allow the incremental construction of queries and context based on previously discovered context/knowledge.

Approach	**Setup**	**Query**	**On-the-fly Context**	**KW Interaction**	**Incremental Enr.**
CoOL		✔	✔		
CONON		✔			
SOUPA		✔			
OntoSensor		✔			
CXMS		✔			✔
AlarmNet					
UI Context Model					
SCAFOS		✔			✔
CML		✔			✔
GeoAIDA					

Table 2.1: An Analysis of Functional Requirements

Analysis of related work has concluded there is no existing system suited for the deployment of multiple sensors, on multiple participants, in multiple environments with the aim of providing a framework for a user to define domain specific context and subsequently meet their information needs.

Chapter 3

Problem Description

This chapter illustrates a number of evaluation sites provided by sport and health physiologists. In each case, there is a description of the activity being monitored, requirements of scientists in this domain, how they currently assess performance and behaviours during these activities and the suite of sensors deployed to discover information of interest. The purpose and format of each sensor is detailed, along with the size of the datasets collected in each domain. Both domain-specific queries and requirements common to all deployments are examined. In explaining the differences and commonalities involved for different domains, this chapter shows how a data analysis framework requires both a combination of automated system operations and user interaction.

Section 3.1 explains how the case studies were chosen and introduces some sensor specific terminology. Sections 3.2, 3.3, 3.4 and 3.5 detail the requirements and current approaches to meeting these requirements in the tennis, cycling, horse-racing and search and rescue domains respectively. Section 3.6 describes the overall issues and approach to solving the user requirements, explaining both the differences and commonalities across deployments. Section 3.7 presents a summary of this chapter.

3.1 Overview

In this chapter, a number of case studies and sensor technologies are described, chosen to illustrate the difficulties involved with typical heterogeneous sensor deployments. These deployments, in the sporting and health domains, and including a smart space environment, are representative of the nature of such sensor usage and the sensors are similarly varied in terms of output format, sampling rates and data properties recorded. [54]

The criteria for selecting these sensors were the availability of real-world problems along with a knowledge worker from which domain knowledge could be obtained. In addition, these sensors provide a representative example of the problems associated with heterogeneous devices [2]. In section 3.6 these problems are summarised along with the commonalities and similarities across typical sensor deployments.

As will be shown in this chapter, there are many areas in which scientists can deploy sensors in an effort to extract better levels of knowledge. Extensive data sets can be collected, which coupled with the diverse requirements of the domains, result in a situation where specialised applications are then required to extract the necessary information. In each case, this chapter will describe the problems which arise when sensors are used to gather information. In addition, the gap between end user requirements and the data generated by these devices will be discussed.

3.1.1 Sensor Description and Usage

The purpose of this chapter is to demonstrate the scope of this research as it plays a part in the experiments which follow in chapter 7. The evaluation sites were quite different and the sensors used varied a great deal in output. This section introduces the sensors used within the experiments and explains the terminology relating to these sensors. Measurements required by the scientists include sensing the position and movement of a subject.

This is achieved for movement using accelerometer sensors: GT3X, GT3X+ and GCDC. For position, GPS sensors (Garmin 405, Garmin Geko) can be used outdoors. Indoor deployments can make use of the indoor localisation sensing device, Ubisense. Physiological data is also required, for this a Sensewear armband sensor can be deployed, as can the Cosmed metabolic sensing system. Finally, the scientists wish to measure the power transmitted to a bicycle during a cycling event. This can be measured with a PowerTap sensor mounted on the wheel. The remainder of this section will detail each sensor involved in the evaluation sites.

ActiLife GT3X and GT3X+ Accelerometers

Figure 3.1: A GT3X accelerometer and its deployment on an ankle

The *ActiLife GT3X* [21] and *GT3X+* [22] are tri-axial (x,y,z) accelerometers which monitor human activity or movement. The GT3X samples at 30 Hertz (Hz) and the GT3X+ can be set at up to 100Hz prior to deployment. The sampling rate may be reduced by scientists deploying the devices to save memory space on the device, which may be necessary for longer deployments. This means each GT3X/GT3X+ deployment may have slightly different output based on initial configuration. All records are assigned a timestamp measured in milliseconds. Each device includes an inclinometer sensor which measures *device orientation information* such as

standing/sitting/lying down. The GT3X+ includes a lux sensor, which is a measure of ambient light. Before the scientist uses a GT3X/GT3X+ accelerometer in any domain, they can configure whether or not to measure inclinometer and lux. Both accelerometers are lightweight, look identical and are strapped to a body. Figure 3.1 shows a GT3X accelerometer positioned on the ankle. A sample of data recorded is shown in Example 3.1 in `.csv` format.

Example 3.1 *Sample Extract of GT3X data:*

x	y	z	lux	incline
488	198	47	1	1
467	186	49	2	1
494	155	80	1	1
479	272	22	2	1
192	187	24	1	1
492	163	45	2	1

The Ubisense Localisation Device

Ubisense is an indoor localisation sensing device. It tracks the 3-D position of Ubisense Tags (*Ubitags*) which are worn by subjects as they move around a *smart space*. Position is tracked in real time by *Ubisense* sensors, which are placed around the smart space and data is output in XML to a server. The information recorded is: (`UbitagID, x, y, z, timestamp`) where the (x, y, z) are coordinates in space. Timing is recorded in milliseconds and the sampling rate is variable (based on the movement of the device). Figure 3.2 shows a *Ubisense* sensor and two *Ubitags*.

Bodymedia Sensewear Armband

The *Sensewear* armband device has a number of sensors, such as a dual-axial accelerometer, heat-flux and temperature sensors which gather information, including an estimate of energy expenditure and physical activity. These

Figure 3.2: A Ubisense sensor (left) and its Ubitags (right)

armbands are worn by subjects on their upper arm. They are suitable for real-world experiments due to their portability, but are not considered the most reliable solution for measuring energy expenditure. Laboratory based sensors are superior in terms of accuracy but cannot be readily deployed in a real scenario. 20 different measurements are output by the *Sensewear* armband, time is measured using real-time timestamps (date-HH:MM:SS) and samples are taken at 0.25Hz. Figure 3.3 shows the device and its positioning on a subject.

Figure 3.3: A Sensewear sensor

CycleOps PowerTap power sensor

The *PowerTap* sensor is a power measuring device placed within the rear hub (center part of the wheel) of a bicycle. *PowerTap* provides an accurate

(within 2.5%) measure of power applied to a bike by a cyclist. The sampling rate is variable, and sensor entries are timestamped using minute (and fractions of minutes) units. Measurements gathered include `watts` (power), `speed` and `cadence`. Figure 3.4 shows a PowerTap hub, and example 3.2 shows some sample output in `.csv` format.

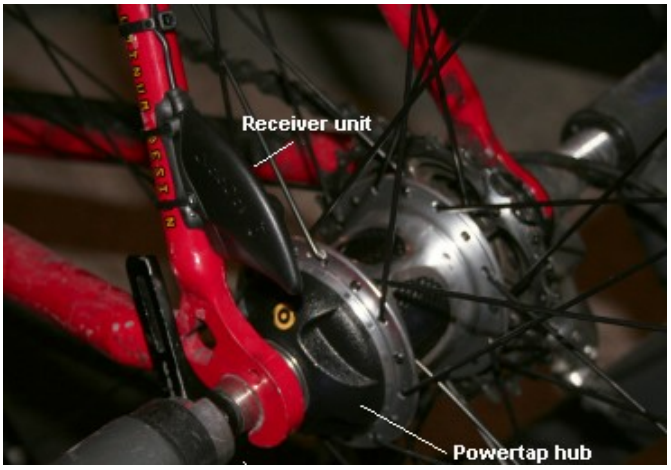

Figure 3.4: PowerTap Hub

Example 3.2 *Sample Extract of PowerTap data (in csv format):*

Time	Speed	Power	Distance	Cadence
22.893	14.67	185	34	99
22.935	14.67	184	36	99
22.977	17	164	38	99
23.019	17	162	40	98

Gulf Coast Data Concepts (GCDC) X6-2A Accelerometer

The X6-2A is a tri-axial accelerometer with a sample rate of 320Hz. Movement of each axis is monitored in `x, y, z` fields, each marked with a time (in seconds). This accelerometer is lightweight and is used to monitor precise movement of the body it is placed. This accelerometer is preferred where extremely precise measurements of movement are required.

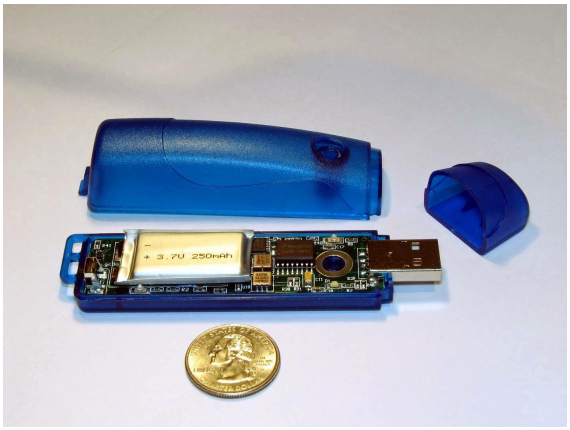

Figure 3.5: The GCDC Accelerometer

Garmin 405 and Geko GPS devices

Two Garmin devices were used in monitoring outdoor activities. The *Garmin Forerunner 405* has a variable sample rate and measures `lattitude`, `longitude` and `altitude`. It also has an integrated heart rate monitor which can be disabled if not required. The device outputs data in the XML compliant GPS eXchange format (`.gpx`), and timestamps are in real time format (Date-HH:MM:SS). The *Garmin Geko* records the same data as the *405* (excluding the heart rate) and has a set sample rate of 0.1Hz. Both devices are deployed by the scientists, the *Garmin Geko* is used where it can be attached to a bicycle, and the *Garmin 405* is worn by the subject in cases where the a heart rate monitor is to be paired with it.

Cosmed k4b2 Metabolic Sensor

The *Cosmed k4b2* [13] is considered the most accurate semi-portable sensor for measuring energy expenditure available at present. The breathing of the wearer is analysed during an activity, and data such as `VO2, Heart Rate` and `O2` (oxygen) are recorded. The data is sampled by the sensor at a variable sample rate (a sample every 3-6 seconds) and timestamps are in HH:MM:SS format. The Cosmed sensor is worn over the nose and mouth

Figure 3.6: The Garmin 405 (left) and Garmin Geko (right)

and has a bulky appearance as can be seen in Figure 3.7. This means it is not possible to deploy in many instances where weight or visibility is a priority to ensure best performance.

3.2 Sensing Movement and Position in Tennis

The sport of tennis takes place on a court (grass, hard-court or clay-based), in an indoors or outdoors environment and can be individual or team (doubles) based. Tennis is not set with fixed time boundaries and the game is driven forward by the scoring of points. The scoring system can be complex, with games, sets and matches making up the structure of the sport. The role of *serving* alternates between players at defined point-based intervals, and the players switch sides at regular set-based intervals. Serving players stand in a defined space behind the baseline prior to each serve, a full list of tennis rules are published by the International Tennis Federation (ITF), the governing body of tennis [59]. Tracking the players movement can help the scientists to deduce these events. As a result of these differences and rules, each tennis activity may have different and diverse requirements. Thus any solution must be flexible to adapt to the changing environment and requirements.

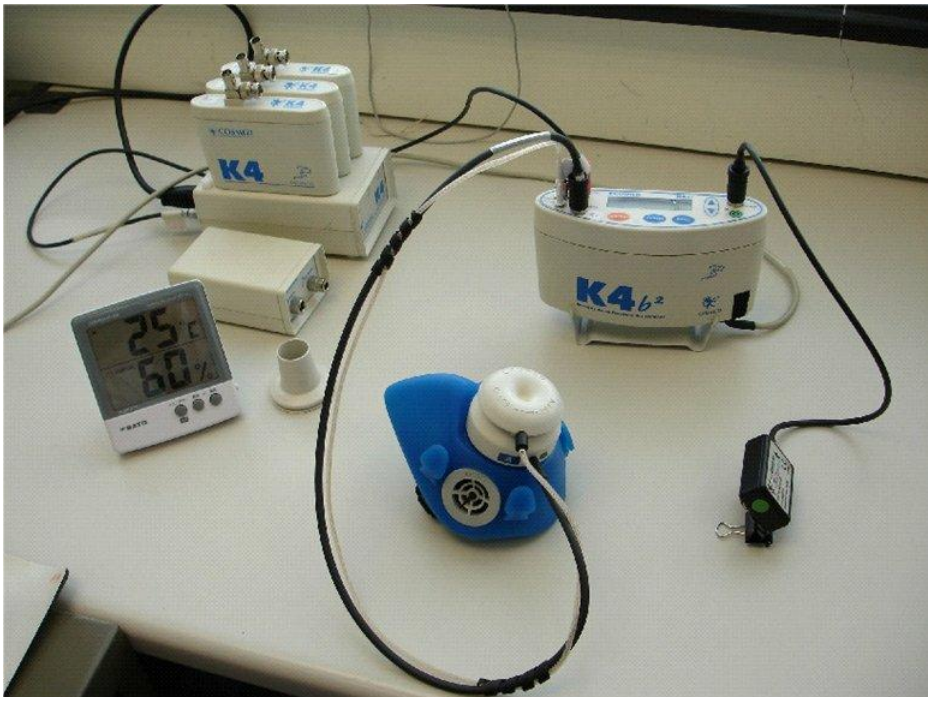

Figure 3.7: Cosmed k4b2 Metabolic Sensor

3.2.1 Background and Requirements

Sports scientists and tennis coaches each have a range of requirements when it comes to analysing and querying behaviours during training and competition matches. Table 3.1 provides a list of the queries the coaches wish to execute. For a coach, the key requirement is the analysis and segmentation of important occurrences, or events throughout a period of play. These include breaking each match into its game, set and serve boundaries and detecting who has scored points. For instance, *query 4* requires the identification of the length of a *rally* prior to a point being scored. The scientists require that all occurrences are identifiable by a time stamp, to be later retrieved when necessary.

Currently, this analysis is carried out manually, checking video recordings for serves, points and game breakdown. Records are then stored in a spreadsheet format and graphed in an attempt to discover interesting information.

	Query
1	Return all serves for Player N in Game N
2	Return the time for each point scored in Game N
3	What part of the court is Player N in at game time T?
4	What is the duration of the rally prior to the point at 05:02 being scored?
5	What is the average duration of a rally resulting in a point scored in Game N?
6	How many points were scored in Game N?
7	What is the average duration of a rally in games where Player N is taking part?
8	Which player covers most ground during Game N?

Table 3.1: Tennis Coach Queries

This is a highly user-intensive process which the scientists wish to avoid by automating using sensor information. The goal is to automate this process as much as possible.

3.2.2 Information Collected

In an effort to automate the process for gathering player and match data, scientists installed a Ubisense [64] system on an indoor tennis court. This Ubisense setup consists of a number of static sensors located around the court which triangulate a signal omitted from each of two Ubitag receivers, carried by each player as they move around the court. Ubisense tracks each player throughout both matches and training sessions. This setup is shown in Figure 3.8 where each player is wearing a *Ubitag* receiver. Additionally, cameras were set at multiple angles around the court to record each match. A summary of the data gathered from the Ubisense sensor setup from deployments is shown in Table 3.2. To aid with the manipulation of sensor data, additional 'basic' context was recorded for each deployment by the scientists. In this domain, this information is a record of the 19 players taking part describing their age, weight, height etc. (*anthropometric information*) and a record for each of 15 deployments describes information such as the start and end time. This is useful information to later allow the scientist

Figure 3.8: Court Ubisense Setup

to focus queries based on individual properties, or to query across sets of similar players and is gathered for all deployment.

Name	**Purpose**	**Frequency**	**Files**	**Data Pts.**	**Total Size**	**Max Size**
Deployment info.	Context	N/A	15	75	5kB	3kB
Subject info.	Context	N/A	19	114	20kB	3kB
Ubisense	Location	var.	36	845,000	14MB	1.2MB

Table 3.2: Tennis Data Gathered

Table 3.2 displays the name of each sensor or information type and its purpose (what it is used to detect). The *Frequency* field refers to the sample rate in Hertz. Ubisense has a variable (var.) sample rate, and sample rate does not apply to simple context information. The *Files* column shows how many individual records for each sensor have been gathered and the set of deployment and subject anthropometric data documented. *Data Pts.* refers to the total number of data points (or entries) for each sensor. *Total Size* is the size of the dataset gathered for that sensor, and *Max Size* is the size of

the single biggest file in that set. In this case, the sensed data consists of 36 individual Ubisense sensor data documents, with 845,000 data points and a total size of 14MB.

While the sensor deployment automates the recording of movement within a spatial environment, interpreting its semantics remains a problem. Using the camera feed as a ground truth, locations of players can be mapped to *zones* using a manual analysis and annotation process. This is not a substantial improvement on the previous method of real-time or video based manual analysis. Relating the basic context recorded to the sensor data recorded is another problem which remains, as it remains external to the automated data collection of sensor values. At a domain level, identifying how to detect certain occurrences using the data available is also an issue, as there is no mechanism (other than spreadsheet and graph analysis) to gain feedback from experimental algorithms using multiple data sources. Basic context information such as a subjects age, weight, or deployment location remains outside the sensor dataset and has to be added manually where required.

For instance, basic Ubitag output values (x,y,z) cannot be easily mapped to the game structure (Game/Set/Point boundaries) as required in *query 1* from Table 3.1. More complex queries such as *query 5* require the detection of a rally and point-score, which requires human analysis. To overcome these issues it is necessary to provide a system to represent data in a canonical form, integrate context and sensor data and a method of allowing a scientist to specify and add new customised levels of context.

3.3 Sensor Deployments in the Cycling Domain

Cycling has many activity formats, such as track sessions, endurance road races and off-road mountain bike races. Events can be individual or team based, and training is usually a combination of fitness improvements and devising race strategies. Scientists may have a number of deployments they

wish to use as evidence for a number of sport science based experiments with the aim of discovering and improving existing detection algorithms. They also want to segment and classify interesting occurrences throughout these activities.

3.3.1 Background and Requirements

In the cycling domain, context necessary for the analysis of sensor data are the different properties (height, weight, etc) of the participants, as well as some environmental factors, such as terrain information. The most advanced methods of measuring direct and indirect calorimetry in the cycling domain are laboratory setups with expensive specialised mechanisms such as a bicycle power meter such as Wattbike [67]. In the field, non-laboratory based experimental analysis is typically performed by devices such as the *Bodymedia SenseWear* sensor which estimates the energy expenditure of the wearer. Other sensors are often used to discover basic information such as location (GPS), heart rate (HRM), speed (GPS) and distance monitors (GPS / Accelerometers). Scientists use manual analysis of these sources using 3rd party software along with related contextual information to try to cross reference with *gold standard* (i.e. best available) solutions currently available with an aim to replicate this accuracy using different sources of data. The advantage of this is that portable, low cost sensors can replace expensive lab-based sensors.

Requirements also includes the calculation of *Total Energy Expenditure (TEE)*, which is the overall aim of scientists in this field. The goal is to detect and measure energy used using direct or indirect calorimetry (measuring physical changes). TEE is the sum of *Resting Metabolic Rate (RMR)*(energy expended at rest), *Thermic Effect of Food (TEF)* (energy expended processing food) and energy expended by *Physical Activity (PA)*. Each of the algorithms for measuring energy-related information terms apply across any

human based deployment, yet calculation may depend on an individuals characteristics, or may have to take into account external factors affecting accuracy. In order to calculate TEE using basic sensors, scientists require the ability to first define and later alter the algorithms for RMR, TEF and PA. In addition, they are required to detect domain context such as climbs, climb categories, sleeping and sleep quality. A more specific list of requirements is shown in Table 3.3.

Queries	
1	Find total amount of time spent above 250W (Power-measuring)
2	Find total amount of time spent above 165BPM (Heart rate)
3	Calculate average heart rate where power above 200W and cyclist_type='endurance'
4	Find the total amount of time where pedal pivot = 'pivot_range_1'
5	Find 'best intervals' for highest '1minute' heart rate and return values for distance covered.
6	Find the average performance factors (Power/Heart Rate/Speed) for each gradient of type = 'hill'
7	Find the average Power value when pedal vector magnitude = 'peak'
8	Find the average speed when braking activity = 'none'
9	Find all occurrences where gradient_profile = 'flat' and cycle cadence = 'high'
10	Find all occurrences where effort intensity = "peak"

Table 3.3: Cycling Requirements

Query 7 requires the average power value where *pedal vector magnitude ='peak'*. To answer this query, some definition for detecting a *peak* vector magnitude is required. Information used as input to algorithms will often be a mix of sensor generated data and background context information.

3.3.2 Information Collected

In a series of experiments, a set of sensors were attached to a group of cyclists in a number of scenarios. The list of the sensors used is shown in Table 3.4 where two different types of accelerometer were deployed: Actilife

GT3X accelerometers positioned on the cyclist, and GCDC accelerometers positioned on the bicycle frame. The Garmin Geko GPS device was also placed on the bicycle. The SenseWear and PowerTap sensors were used to estimate energy expenditure and power respectively.

Physiological data (e.g. heart rate) was gathered by the scientists using sensors, and contextual data was entered manually. Sensor data was gathered from three separate deployments; two training sessions for one participant in a mountainous and on-road environment. The third was gathered as four cyclists took part in the Race Around Ireland [58] endurance event. Figure 3.9 shows the typical deployment, with the accelerometers located on the cyclist and bike, the GPS located on the bike, a heart rate monitor on the cyclist, and PowerTap located on the bike hub to measure power.

Table 3.4 shows the data recorded by the sensors during the cycling deployments. The dataset includes some cyclist specific information which are key properties relevant to cycling, including the cyclist type. Finally each of the sensor data outputs are listed under their sensor type. The table columns are the same as those in Table 3.2, described in Section 3.2.2.

Name	**Purpose**	**Hz**	**Files**	**Data Pts.**	**Total Size**	**Max Size**
Deployment info.	Context	N/A	3	24	40kB	2kB
Garmin GPS	Location	0.4	2	30,000	589kB	922kB
GCDC Accelerometer	Measure position	100	8	800,000	16MB	2MB
GT3X Accelerometer	Measure position	30	8	7,800,000	200MB	80MB
Heart Rate Monitor	Measure heart rate	0.4	1	675	322kB	322kB
PowerTap	Measure Power	1	10	161,000	1.6MB	314kB
Subject info.	Context	N/A	5	30	10kB	2kB
Video Camera	Ground truth record	N/A	1	N/A	1GB	1GB

Table 3.4: Sensors Used in Cycling Tests

The data format and sample rates differ across sensors deployed. Finding a good (or optimal) algorithm for detecting energy expenditure may require many iterations which is very time consuming for scientists due to the large

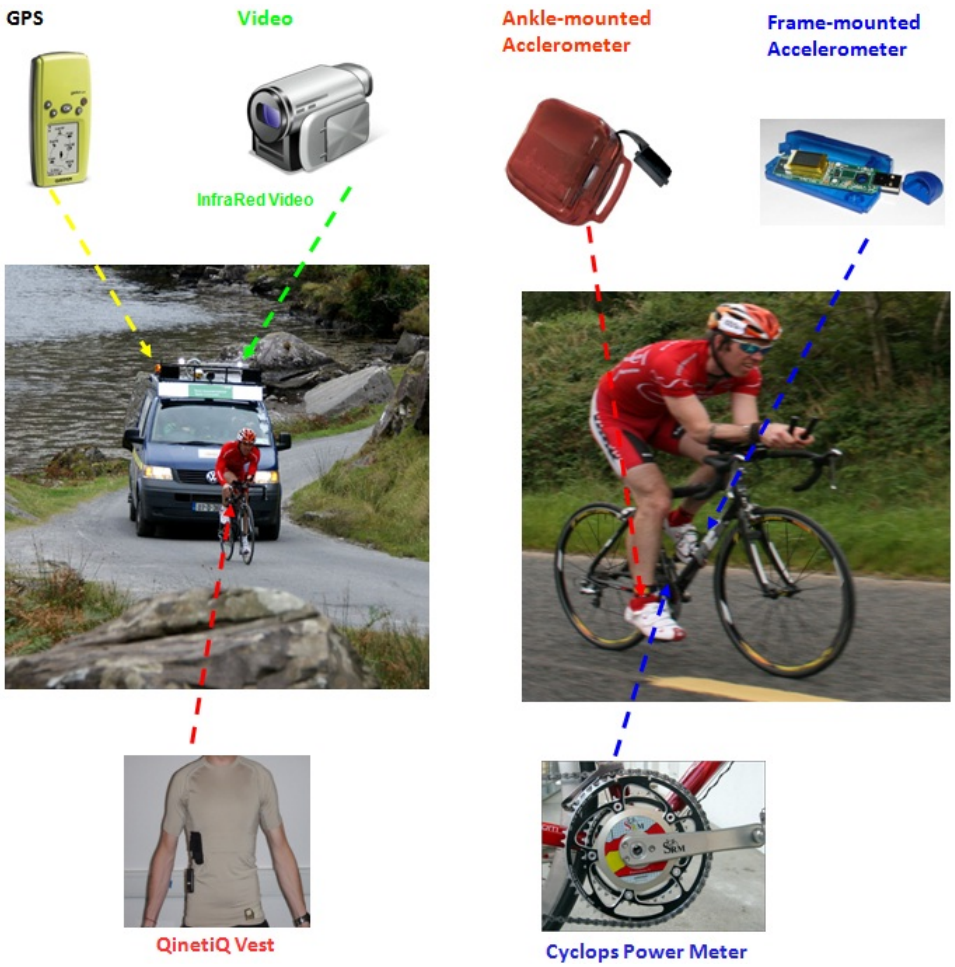

Figure 3.9: Cycling Deployment: Race Around Ireland

volume and varied nature of the sensor data gathered. The data in its initial format can be input to potential algorithms but building up iterative strategies (where one result feeds into another algorithm) remains a problem. Context such as the cyclist property information has to be manually added to relevant datasets, which can become an issue where algorithms require this information for personalised results, or when comparisons across multiple participants are required. For instance, *query 4* is aimed at endurance cyclists only.

3.4 Using Sensors in Horse-Racing

In horse-racing, the majority of performance related research has been focussed on the training and breeding of the horse. Recently, more focus has been put on those who control the horse, the jockeys. In preparation for races, extensive training is undertaken by jockeys to meet the needs of a particular race or horse. There are many factors which may affect the performance the jockies (as they ride the horse) such as *the going*, which is the condition of the track (firm, soft, heavy etc.) or the energy levels of the jockey and horse. Scientists wish to analyse the effects of these factors and experiment with new algorithms for detecting the energy expenditure of jockeys. In doing so, they aim to replace expensive, bulky or laboratory based techniques with affordable, lightweight and portable sensors.

3.4.1 Background and Requirements

As with cycling, a key requirement of scientists in horse-racing is to generate algorithms for estimating *Total Energy Expenditure (TEE)* using sensors. Another requirement is to determine the energy expenditure values, both those calculated through new algorithms and the measurements provided by the SenseWear armband (or the more precise measurements obtained using the Cosmed k2b4 sensor), by assigning *stages of the activity* with an *intensity* metric. Scientists also require querying abilities over a defined time period. They want to analyse the performance of jockeys in a simulated indoor setting, and compare this with performance outdoors in various environments with different weather conditions, and in race situations with various horses. They wish to perform standard data analysis algorithms to detect standard deviation, median, or peaks and troughs of values across time bounds.

Table 3.5 shows some of these queries expressed during these analyses. There are some events that underpin their understanding of the data. For instance,

Queries	
1	What is the maximum heart rate of jockey N while the horse gait was fast-cantering?
2	Which trainee jockey aged under 16 had the highest energy expenditure for training race N?
3	Calculate the Total Energy Expenditure for apprentice Jockey 2 for a training session.
4	How accurate (%) is the current algorithm for Total Energy Expenditure relative to the Cosmed values?
5	Identify all instances of a jockey whipping the horse / simulator.
6	Calculate the TEE value for all jockeys aged N, while on the simulator.

Table 3.5: Sample Horse Racing Queries

horse gait classification is required to detect a canter in *query 1*. This feature is not directly sensed or tracked by a sensor and must be identified using a separate algorithm. The requirements are varied, from domain algorithms for detecting specific horse-racing activities to standard analysis of human physiological factors applicable to any domain where a human is being monitored.

Currently, the 'gold standard' method for measuring the key requirement - Total Energy Expenditure - is to use a Cosmed K4 b2 metabolic system on a jockey. This can be performed during a training deployment on a horse or simulator. The measurement of breathing during an activity provides a good measure of energy expenditure for that individual. However, this system is bulky and difficult to deploy as it impairs vision and adds weight to the jockey. As a result it is not suitable for large scale analysis in outdoor environments or during races. Scientists also measure other metrics of a jockey during training, such as heart rate and various accelerometer values. This information is currently interpreted using spreadsheets to compare multiple sets of information. Requirements based on a jockeys individual properties must be manually entered into each relevant set of data.

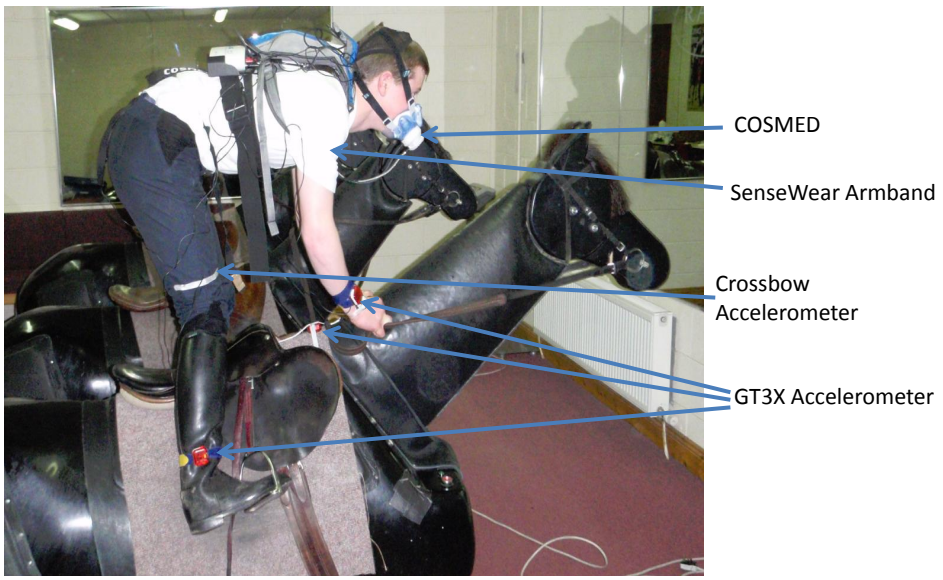

Figure 3.10: Jockey Riding the Simulator

3.4.2 Information Collected

In an effort to provide a more complete set of information and more choice to the scientist, a number of sensor experiments in the horse-racing domain were undertaken. The sensor deployments in the horse-racing domain were extensive, including 11 subjects, each equipped with a variety of sensing devices during both indoor and outdoor training sessions. Indoor sessions took place on the simulator, and each participant was equipped with a Cosmed, SenseWear and 6 GT3X accelerometers, five of which were located around the body and one on the saddle of the horse. Outdoor sessions included a GPS system for location. Figure 3.10 illustrates the typical setup of the deployment on the indoor simulator. The deployment was performed on a set of 11 trainee and apprentice jockeys using the sensors in table 3.6.
Table 3.6 shows a summary of data gathered during these experiments. As with the other domains, activity information, such as start / end times were manually recorded for each of the 22 training sessions performed. The jockey information was gathered and included domain-specific information such as

the qualification (trainee / apprentice) of the jockey and other human based properties (height, weight, BMI etc.). The sensor data gathered amounted to 2.1GB, with the Cosmed sensor generating the most content.

Name	**Purpose**	**Hz**	**Files**	**Data Pts.**	**Total Size**	**Max Size**
Cosmed k4b2	Breath analysis	10	22	4,000,000	20MB	1MB
Deployment info.	Context	N/A	22	110	38kB	3kB
Garmin GPS	Location	0.4	11	14,300	2MB	300kB
GT3X Accelerometer	Measuring position	30	110	240,000,000	231MB	13MB
Sensewear	Energy Expenditure	0.3	1	16,000	1.4MB	1.4MB
Subject info.	Context	N/A	11	66	20kB	2kB

Table 3.6: Horse Racing Data Gathered

There are a number of problems which remain after gathering this dataset. Normalisation and integration must be undertaken to deal with different formats and sample rates. Often external tools are required to analyse multiple data sources. Outliers must be manually identified. Interpreting or visualising what some information, such as the coordinates from an accelerometer takes considerable time as the values are not at a high (abstract) level. A problem with the horse-racing environment is that it involves a considerable external force (that of the horse) which makes things more complex when trying to measure the activity of the jockey using movement-based sensors. For instance, the total energy expenditure is the product of both jockey and horse energy expenditure when measured solely on movement. This makes the TEE algorithms *Physical Activity (PA)* metric different for a cyclist than for a running athlete. Basic context information such as a jockeys properties are recorded externally to sensor data which leads to hard-coding algorithms requiring this information. As a result, adapting or modifying these algorithms may involve significant effort from the scientist.

3.5 Using Sensors in Search and Rescue

The Search and Rescue (SAR) base of the Irish Coast Guard serves as a launching point for helicopter based rescues of those in difficulty in the sea or in mountainous/remote areas. The members are often on call either on site or at home, awaiting a rescue mission. It is a physically demanding job, and maintaining high fitness levels is a key requirement of those working in this field. There are limits in place for the amount of time the employees may work, and hours they can fly, based on energy and fatigue levels. Thus, scientists wish to study and evaluate these limits using energy expenditure algorithms in order to improve operations.

3.5.1 Background and Requirements

Health scientists monitor the behaviour of these workers as they conduct their work, either on call (at home or on site) or while engaging in a rescue mission. The key goal is to analyse their work and downtime behaviours in order to identify ways in which their work-life could be balanced and energy levels optimised. For instance, it may be advantageous to be on call while resting on site rather than at home. Table 3.7 shows the queries required by the scientists. As with the sporting domains, energy expenditure was again a key feature the scientists wish to measure using accelerometers. A requirement for the SAR domain was to measure the *Sleep Quantity* and *Sleep Quality* of the participants (*Query 3)*. This can be estimated physically by measuring other events, such as time spent moving, and intensity of motion over a given time span. Together, these calculations can be used as part of a sleep quality algorithm. Discovering energy expenditure intensity information across user-defined time boundaries is also a requirement in the SAR domain.

Queries	
1	Calculate Total Energy Expenditure for each employee using direct calorimetry
2	What proportion of a specified day is spent at an energy intensity level of moderate or higher?
3	Which employees have a good quality of sleep?
4	What is the relationship between sleep quality and energy use on the day before and after sleep?
5	What alterations to the Physical Activity algorithm is required to compensate for external factors?
6	Compare algorithms for energy to SenseWear measurements, what is the average error?

Table 3.7: Requirements in the Search and Rescue Domain

3.5.2 Information Collected

Scientists placed a set of sensors on each SAR worker. Two ActiLife GT3X+ accelerometers were used: one positioned on the right side of the participants waist and the second located on the right ankle. Each accelerometer was aligned to face forward. A SenseWear armband with a sample rate set to 15Hz, but the output set to 1 minute intervals was also placed on each worker. These are worn on the upper arm of all subjects (participants) at all times (24hrs/day), for a period of 24 or 48 hours set for each deployment. They also wore a SenseCam, which records an image based on movement or change of scenery, during the day. The SenseCam is used as a ground truth rather than as a parameter for the formation of algorithms. External information about each of the participants was also recorded, including age, weight, mass, height and their dominant side (right/left handedness).

Name	Purpose	Hz	Files	Data Pts.	Total Size	Max Size
Deployment info.	Context	N/A	44	220	80kB	2kB
GT3X+ Accelerometer	Measuring Position	100	88	900,000,000	1.8GB	90MB
SenseWear	Energy Expenditure	0.3	44	1,440,000	9MB	600kB
Subject info.	Context	N/A	11	66	20kB	2kB

Table 3.8: Search and Rescue Data Gathered

The data set gathered in the SAR deployment is shown in Table 3.8. Throughout deployment, the workers go about their day as usual, engaging in rescue missions where required and resting on base or at home while on call. The devices are also worn while sleeping. Initially, the scientists decided to aggregate data at a sample rate of 100Hz. Over the deployment, the two GT3X+ Accelerometers, SenseWear armband and SenseCam were deployed on 11 different workers a total of 30 times, for an average of 24 hours each time. The standard setup was a 24 hour work day deployment followed by a 24 hour rest day.

While the range of sensors deployed in the SAR domain is not as extensive as the cycling and horse-racing domain, challenges still exist in allowing the scientist to achieve their requirements by analysing the dataset. It remains a challenge to allow the application of automated data analysis and iterative improvement of detection algorithms using the scientists knowledge of the domain. The SenseWear armband provides an estimation for energy expenditure, on which algorithms utilising the GT3X+ could at first be modelled. This information forms part of the evidence the scientist will use to improve their event detection capabilities. For instance, a correlation between an accelerometers `(x,y,z)` values and the Sensewear output could allow for an initial algorithm to be defined. The different sample rate and output format (in terms of units) requires changes to previous TEE and PA algorithms, but the broad algorithm can remain the same. This illustrates the benefit of a system which would reuse algorithms, such as energy expenditure in different scenarios, while allowing key parameters to be included or altered.

3.6 Groundwork Approach

3.6.1 Overall Issues and Approach

Across each of the case studies described, there are a number of problems to be addressed.

- Sensor data is low level
- Context must be integrated
- There is no method for defining or locating patterns or events of interest
- Data is heterogeneous and must be integrated
- Data is not suited to a high-level query language

The approach taken in this research is to close the gap between low-level sensor data and a high-level query language, by first examining commonalities across the different domains and then determine where they differ. Doing so will identify where the system requires no user input and where the system needs some level of user interaction to provide domain knowledge.

3.6.2 Commonalities Across Multiple Deployments

Table 3.9 lists the number of sensors used in each deployment. As can be seen there was some overlap for some of the sensors deployed in multiple scenarios. Each number represents an individual sensor data output file. This section describes the commonalities across the multiple sensor deployments.

Sensor Data is Raw: How a user interacts with sensor data remains an issue. In some cases, the sensor data is raw or unstructured. If more than two types of sensor are involved in the deployment, as is the case in all but the tennis deployments, then multiple sensor data outputs

Sensor	Cycling	Horse-Racing	Search and Rescue	Tennis
GT3X / GT3x+	8	110	88	-
Ubisense	-	-	-	36
SenseWear	-	1	44	-
GCDC	15	-	-	-
PowerTap	10	-	-	-
Cosmed	-	22	-	-
Garmin Geko GPS	2	-	-	-
Garmin 405 GPS	2	11	-	-

Table 3.9: Sensors Deployed in Each Domain

must be analysed independently and manually cross checked by a scientist if required. Where many sensors are deployed, keeping track of this information manually is a considerable problem for the scientists who wish to use the data. To address these issues, sensor data must be *enabled* by representing it in canonical format. In chapter 4, the approach to providing a standard structure for sensor data is described as is the single process which enables all sensor data.

Context is Missing: Multiple datasets, each available for analysis in isolation are often missing key semantic information. Information which may only be known to the scientist at the time of deployment is often embedded in manual records and this information may be required multiple times in order to detect or identify some requirements (as part of an algorithm). Semantics relating to the deployment environment, such as the sensor configuration, in addition to the technical details of the data output are sometimes not included in the sensor data. Those involved as *participants (subjects)* and the *activity* information is known by the scientists at deployment time, but utilising this semantic knowledge remains an issue as there is a gap between this information and the sensor data. Chapter 4, presents the enablement process of providing a standard mechanism for the scientists to

input basic context.

Requirements are Complex: The evaluation sites provide an extensive set of requirements and queries for multiple sensor outputs. Some of these queries apply in multiple deployment instances (e.g. measuring distance based on time and speed), while others are specific to suit a certain deployment (e.g. classifying a horses *gait*). The requirements of the scientists are dynamic, changing with the introductions of new sensors, new environments and new subjects. Results from some algorithms may lead to the discovery of new information to be used as part of the construction of a new algorithm. As a result, a generic method of applying context to sensor data is provided to meet the varied requirements of the users. To achieve this, *extension* profiles are defined which provide a standard mechanism for user driven contextual enrichment in chapter 5.

3.6.3 Differences Across Multiple Deployments

The differences across the multiple sensor deployments are summarised in this section.

Structural issues: There are structural issues associated with the different output format of the various deployed sensor devises. These differences include different sample rates (e.g. 100Hz for a GT3X+ accelerometer to 1/3Hz for a SenseWear armband), or inconsistent sample rates, such as for Ubisense where samples are taken only when movement is detected rather than at a set period. Others, such as the Actilife GT3X+ accelerometer, can have their sampling rates changed during configuration pre-deployment, or can aggregate values in a reduced sample rate post-deployment. The scientist decides on a plan for analysis using various individuals, sensors and scenarios. Thus it is necessary to

facilitate the specification of these entities. This is enablement and is described in chapter 4.

Units of measurements: Timing is often measured by timestamps based on time and date (Garmin GPS), others offset time from a master time (stored as a header, e.g. GT3X accelerometer), or measure time in terms of fractions of a minute or second. Following deployment it is the scientist who must synchronise and normalise the sensor data. This is an issue to the scientist as it is time consuming, and requires considerable effort. Chapter 4, describes how the timing protocol for each sensor is defined, and now this information is normalised to a common standard. The user has the ability to further alter the data if they wish to use a different time format.

Semantic issues: This relates both to the ambiguous terminology in sensor data files as well as ambiguous identification of the context in which the sensor was deployed. An accelerometer located on a wrist has an output different to one located on an ankle. Defining the context in which the data has been gathered helps the scientist to later analyse and interpret their datasets. However, there remains issues with clearly defining and formalising this sensor information and automating the integration process. Chapter 4 describes how location and other basic context can be encoded as part of the enablement process.

Domains and Queries are Different: To discover knowledge such as classifying a *horse gait*, multiple sensors are required. Data gathered from sensors can be manually examined, using a ground truth, to try to discover patterns in the data corresponding to this kind of knowledge. Applying algorithms to detect this information across multiple instances requires using third party software such as a spreadsheet to track changes in different data sources and changes must be made in

multiple locations to improve existing algorithms. Once this information is discovered, there is no standard way of querying the results, or to use this knowledge to further enrich, or discover new information. Thus, there is a substantial semantic gap between the sensor data, context and the users complex requirements for each deployment. For example, it is difficult to ascertain the metric of *physical activity*, as required by Query 5 in Table 3.5, using the sensor (GT3X) required, as its output are limited to simple `(x,y,x)` values. Thus, it is necessary to facilitate user driven extensions of the schema which drives the contextual enrichment. The approach to achieving this is detailed in chapter 5.

3.7 Summary

This chapter has described the range of both sensor types and the domains in which they were deployed. The requirements of each of these deployments have driven this research. Clearly a solution which meets the specific requirements of a single scientist, user or domain will not suffice.

To tackle these issues, a generic framework for transforming sensor data from its initial format to an intelligent format which can be queried using a simple user interface and a standard query language is presented. To achieve this in each of the domains listed as deployments, the scientists must be enabled to be the driver of this process, implementing algorithms and updating context directly, enriching the sensor data sufficiently to achieve the requirements and expand the knowledge available to discover previously unknown information. In doing so, any dataset from any activity can be monitored, interpreted, analysed and enriched by a *knowledge worker*, substantially improving their ability to discover new knowledge in any domain. This led to the development of the AREA Framework, as explained in the next chapter.

Chapter 4

The AREA Framework

The previous chapter presented multiple evaluation sites, assessed the similarities and differences across user requirements and the data gathered from these deployments. In doing so, the need for an abstract method of transforming sensor and context information to a format where a high level query language could be used to integrate the data was identified. The challenge is to provide a framework that manages both the information needs of the knowledge workers while also facilitating the heterogeneous environments posed by the different case studies. A key goal is to deliver a method to allow the analysis of activities with an aim to observe the past and subsequently, adapt future behaviour.

Section 4.1, introduces the AREA architecture. Section 4.2 illustrates the steps required to describe the sensors and their environment. Section 4.3 describes the process of providing structure to the sensor data. Section 4.4 explains how sensor data is integrated with related context. Sections 4.5 and 4.6 present a brief overview of the application of user-defined enrichment and the automated system-defined enrichment processes respectively, which are covered in more detail in chapters 5 and 6. AREA is summarised in Section 4.7.

4.1 Overview

The core contribution of this dissertation is the Activity Retrieval for Enrichment and Analysis (AREA) framework. The motivation behind AREA is to allow scientists to structure, interpret, query and analyse heterogeneous sensor data using a simple set of tools. As described in chapter 3, the data output from multiple sensors presents a number of issues, relating to the requirements of the scientists deploying sensors. The challenge is to provide a generic mechanism for modelling sensors and other concepts (subject and deployments) and their relationships which can be understood and used by knowledge workers wishing to analyse this data. The broad aims are: integrating multiple sources of information, enabling a knowledge worker to design and improve techniques for discovering new knowledge (using user-controlled `query` profiles), allowing the sensor data to be queried using a standard query language and deploying *automated* activity analysis to discover new information (using `dataset` profiles).

There are three types of profile. Enablement profiles include sensor, subject and deployment profiles. User-controlled enrichment is performed by applying `update` and `extension query` profiles. Finally, metadata analysis allows system-defined enrichment to take place using an approach where AREA generates another profile: the `dataset` profile.

A high-level illustration of the approach taken is shown in Figure 4.1. What the AREA framework provides is a generic process for bridging the semantic gap between user requirements and the basic sensor output. While there will always be some level of user interaction, AREA provides the necessary structure and tools to allow sensor output and semantics to be integrated irrespective of the deployment environment, subject or sensors. As illustrated, there are sensors deployed in multiple instances, each measuring different subjects. To represent this context are the following concepts: the `sensors`, `deployments` and `subjects` (or participants) involved. Each con-

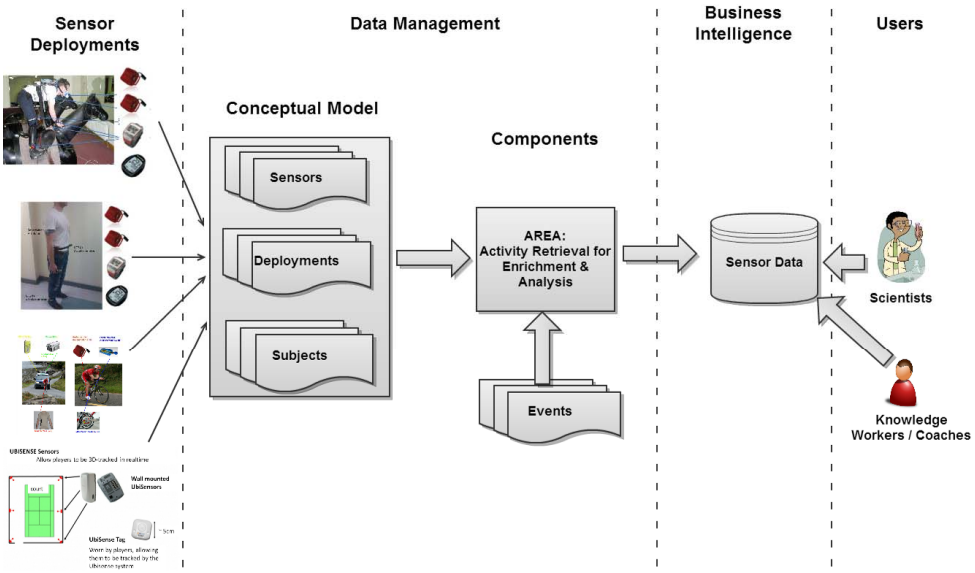

Figure 4.1: The AREA System Architecture (High level)

cept is modelled as a profile. This context information is recorded by a user prior to a deployment. Having a standard mechanism for describing the {sensor, subject, deployment} triple allows a generic transformation of sensor data to take place following any deployment. Later stages allow a knowledge worker to extend the dataset structure. These structural updates are then available to all user types (as defined in chapter 1).

An overview of the operating architecture is shown in Figure 4.2, which shows the flow of sensor data and user interaction throughout AREA. Each process is numbered for ease of discussion. The approach ensures that once the required context (profiles) are defined during context initialisation (section 4.2), any sensor data can be interpreted by a scientist without the need for manual annotation for each experiment.

4.1.1 Design Criteria

Following on from the analysis of the information needs of different knowledge workers and the technical restraints of the sensors used in the previous

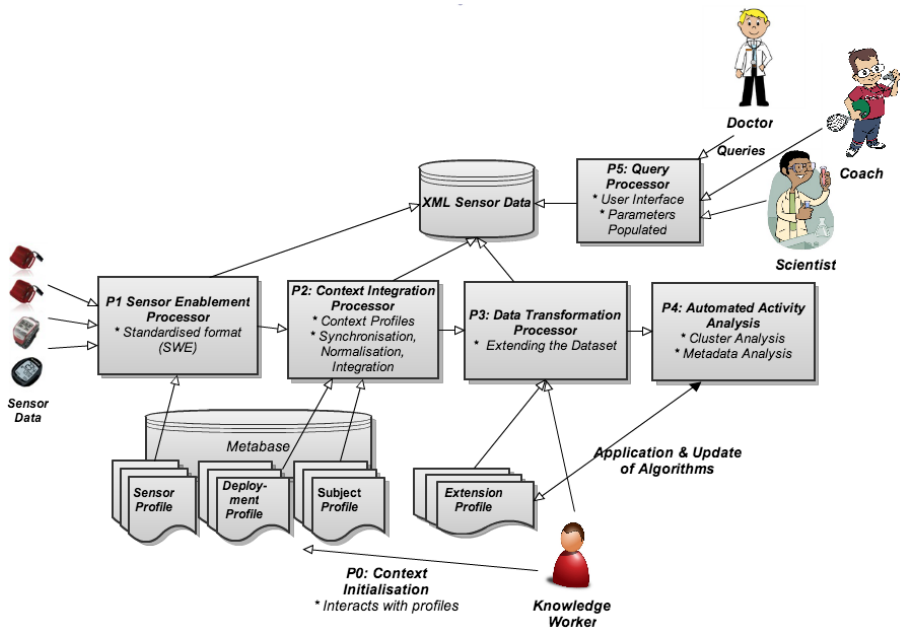

Figure 4.2: The AREA Operating Architecture

chapter, there is a need for a complete framework to structure, interpret, query and analyse these sensor sources. The AREA framework takes these four broad requirements as input to the overall design. The solution must be modular, allowing additional sensors or participants when necessary. In addition, queries must be simple to express to an average user. Due to the different domain requirements, a knowledge worker is always going to be part of the solution. The challenge is in enabling this worker to impose context on the sensor data, thus allowing simple querying for complex information needs. Therefore, a solution must be extensible in terms of context. Table 4.1 shows the design criteria for AREA, which when complete will meet the functional requirments first introduced in section 1.3.2.

4.1.2 Chapter Roadmap

The AREA framework, defined by this dissertation, is presented in this chapter. Each of the processors involved in bridging the semantic gap are

Criteria	Description
Simple Setup	Definition of sensors, subjects and deployments
Interoperable format	Structured data format to allow querying
Simple Querying	User can query data directly
On-the-fly Update	Allow the user to change structure and content of the dataset
Automated Analysis	Analysis of data and user-defined context to suggest improvements

Table 4.1: Design Criteria for AREA

labelled P0 - P5 in Figure 4.2. In this chapter, section 4.2 describes context initialisation (P0). Section 4.3 details the process of sensor enablement (P1) and section 4.4 the process of context integration (P2). The data transformation process (P3) and automated activity analysis (P4) are covered in sections 4.5 and 4.6 respectively with further discussion in chapters 5 and 6. Finally, query formulation (P5) is detailed later in chapter 7.
The chapter will proceed with a full description of the basic processes and finish with an overview of the complex components which form the basis of subsequent chapters.

4.2 Context Initialisation (P0)

Before processing of sensor data takes place, it is necessary that the system has some knowledge of the activity taking place. Thus, an initialisation of basic information about the hardware setup (sensors), deployments and subjects is specified and stored in the metabase as context data. Context initialisation generates an instance of `sensor, deployment` and `subject` profiles for each scenario or activity. These profiles have the role of providing relevant structural and contextual information to a deployment of sensor devices. Profiles are stored in the metabase, to be later used as query parameters or to be used by knowledge workers to update with additional context if necessary. They are also necessary in the process of sensor enablement (P1) and context integration (P2).

It is necessary to identify information required to describe any sensor, deployment or subject in order to provide generic descriptions of these concepts. In the case of sensors, it is assumed that the scientist identifies each sensor with a `name`, `id` and `type`. To interpret sensor readings it is necessary to model the timing protocol for each sensor. As sensors record a vast range of properties, any framework must be able to interpret sensor data in terms of its output. Thus, an expandable list of sensor fields are defined in terms of their format and scale. This solution will provide a framework to define sensor output, deployment environments, and subject properties, which can later be merged as necessary using a wrapper class - which encapsulates the underlying subject and deployment information with sensor data.

4.2.1 Sensor Profile

A `Sensor` profile is an XML template formally describing the output from a sensor. The `sensor` profile defines the sensors involved in the deployment in terms of its key properties: what it is recording, in what measuring unit, what these values represent. Once a sensor profile is defined it can be used as the basis for describing that type of sensor in any environment. These profiles are used to perform the necessary data operations in the sensor enablement stage, providing the basic structure to sensor data. Definition 4.1 shows the XML Schema for a `sensor` profile.

Definition 4.1 *Sensor Profile Schema*

```
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <xs:element name="value"><xs:simpleType><xs:restriction base="xs:string"/>
3   </xs:simpleType></xs:element>
4   <xs:element name="type"><xs:simpleType><xs:restriction base="xs:string"/>
5   </xs:simpleType></xs:element>
6   <xs:element name="timing"><xs:complexType><xs:sequence>
7       <xs:element ref="format"/>
8       <xs:element ref="sampleRate"/>
9       <xs:element ref="startTime"/>
10  </xs:sequence></xs:complexType>
```

```
11   </xs:element>
12   <xs:element name="startTime">
13       <xs:simpleType><xs:restriction base="xs:dateTime"/></xs:simpleType>
14   </xs:element><xs:element name="scale">
15       <xs:simpleType><xs:restriction base="xs:string"/></xs:simpleType>
16   </xs:element>
17   <xs:element name="sampleRate"><xs:simpleType><xs:restriction base="xs:string"/>
18   </xs:simpleType></xs:element>
19   <xs:element name="name"><xs:simpleType><xs:restriction base="xs:string"/>
20   </xs:simpleType></xs:element>
21   <xs:element name="id"><xs:simpleType><xs:restriction base="xs:int"/>
22   </xs:simpleType></xs:element>
23   <xs:element name="format"><xs:simpleType><xs:restriction base="xs:string"/>
24   </xs:simpleType></xs:element>
25   <xs:element name="fields"><xs:complexType>
26   <xs:sequence><xs:element ref="field" maxOccurs="unbounded"/>
27   </xs:sequence></xs:complexType>
28   </xs:element>
29   <xs:element name="field"><xs:complexType><xs:sequence>
30           <xs:element ref="name"/>
31           <xs:element ref="scale"/>
32           <xs:element ref="value"/>
33   </xs:sequence></xs:complexType></xs:element>
34   <xs:element name="SensorProfile"><xs:complexType><xs:sequence>
35       <xs:element ref="id"/>
36       <xs:element ref="name"/>
37       <xs:element ref="type"/>
38       <xs:element ref="timing"/>
39       <xs:element ref="fields"/>
40   </xs:sequence></xs:complexType>
41</xs:element>
42</xs:schema>
```

The sensor profile has five main elements used to describe a sensors output. They are shown in Definition 4.1. The sub-elements of the `<SensorProfile>` and `<timing>` and at least one `field` element are present in every sensor profile definition. In XML schema terminology the default *number of occurences* for an element is 1. If an element and its sub-elements occur multiple times, `MaxOccurs` can be set to *unbounded* as is the case for the

`field` element on line 26 of Example 4.1. Similarly, setting `MinOccurs = 0` defines an optional element. A brief description of each of the key elements of the sensor profile is shown below:

- id: An integer input representing an identification value for that sensor (line 21, 35)
- name: The name of the sensor (e.g. Garmin 405) (line 36, 19)
- type: Specifies the type of sensor. Out deployments include types: accelerometer, GPS, Ubisense, physiological, metabolic, powermeter (line 37)
- timing: Is an element to describe the timing protocol of the sensor and contains 3 sub-elements: (lines 6-10, 38)
  - format: Specifies the unit used to represent time. e.g ms (milliseconds)
  - sampleRate: Specifies the sample rate set on the sensor. e.g. 30Hz (Hertz)
  - startTime: Specifies the real-time that the sensor was deployed, using XML Schemas *dateTime* datatype.
- fields: Consists of one or more `field` elements, which are individual sensor records. Each `field` is defined by sub-elements: (lines 25, 29-33)
  - name: Specifies the name of the field being recorded (data field)
  - scale: Specifies the format of the value recorded. e.g. cm, kg, g
  - value: Specifies the expected type of value for this field. e.g. integer, text

In Example 4.1, the `sensor profile` for a GT3X is displayed. Different usages are presented by changing the <`startTime`> element. The example shows the compulsory elements defined: `id` is assigned '1', the profile is named `gt3x+` and is an accelerometer <`type`>. Timing details are set, with 30Hz milliseconds starting at 1.30pm on 11th January specified (line 6-9). The sensor specific information is then defined, with three <`field`> elements listed named (`x`,`y` and `z`), corresponding to a force in 3 dimensional space, each of which have a scale of 'g' (g-force) as a double data input (line 11-37).

Example 4.1 *Sensor Profile for ActiLife GT3X+ Accelerometer (30Hz)*

```
1<sensorProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNames
2paceSchemaLocation="file:///C:/NewWorkspace/csv2xml/ch7/sensorProfile.xsd">
3    <id>2</id>
4    <name>gt3x+</name>
```

```
    <type>accelerometer</type>
    <timing>
        <format>ms</format>
        <sampleRate>30</sampleRate>
        <startTime>2010-01-11T13:30:00</startTime>
    </timing>
    <fields>
        <field>
            <name>x</name>
            <scale>g</scale>
            <value>int</value>
        </field>
        <field>
            <name>y</name>
            <scale>g</scale>
            <value>int</value>
        </field>
        <field>
            <name>z</name>
            <scale>g</scale>
            <value>int</value>
        </field>
        <field>
            <name>lux</name>
            <scale/>
            <value>int</value>
        </field>
        <field>
            <name>incline</name>
            <scale/>
            <value>int</value>
        </field>
    </fields>
</sensorProfile>
```

4.2.2 Subject Profile

A `subject` profile is an XML template, defined by an XML schema as shown in Definition 4.2, of key features of a subject involved in a sensor deployment. This includes properties such as age, weight, height and Body Mass Index

(BMI). Participant properties specified in the `subject` profiles can be reused for multiple deployments where that participant was the subject of interest. Definition 4.2 shows the structure of a `subject` profile.

Definition 4.2 *Subject Profile*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="value" type="xs:string"/>
  <xs:element name="subject">
      <xs:complexType><xs:sequence>
              <xs:element ref="id"/>
              <xs:element ref="name"/>
              <xs:element ref="activity"/>
              <xs:element ref="properties"/>
      </xs:sequence></xs:complexType>
  </xs:element>
  <xs:element name="scale" type="xs:string"/>
  <xs:element name="property">
      <xs:complexType><xs:sequence>
          <xs:element ref="name"/>
          <xs:element ref="scale"/>
          <xs:element ref="value"/>
      </xs:sequence></xs:complexType>
  </xs:element>
  <xs:element name="properties">
      <xs:complexType><xs:sequence>
              <xs:element ref="property" maxOccurs="unbounded"/>
      </xs:sequence></xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="id" type="xs:integer"/>
  <xs:element name="activity" type="xs:string"/>
</xs:schema>
```

The subject profile has 4 main elements, all of which are compulsory. There can be multiple `<property>` elements, which are extensible properties allowing new characteristics to be added for each subject. For instance, *anthropometric* data such as height, weight, body mass index, etc. may be required when querying for sensor data relating to a certain physiological

grouping, or where these metrics feed into personalised algorithms. The key elements are now described:

- id: An integer input represents an identification value for that profile (line 5)
- name: A text representation that names the subject profile (line 6)
- activity: Specifies the activity type the subject engages in (horse-racing, athletics etc) (line 7)
- properties: One or more `property` elements representing properties of the subject. Its sub-elements are: (lines 8, 12-23)
  - name: Specifies the name of the property
  - scale: Specifies the format of the value recorded
  - value: Specifies a value for this property

Example 4.2 shows an instance of a `subject` profile for a trainee jockey. In this case, the standard compulsory elements: `<id>, <name>` and `<activity>` are specified as *'1'*, *'James'* and *'horse-racing'* respectively. Domain-specific information can be captured in the sensor profile by using property elements. There are also five jockey-specific features as individual `<property>` elements. Each of these assign a name for an element, a value for this element and a scale for that value. For instance, the first `<property>` specifies a qualification element which is defined as a text value *'trainee'*. Similarly height and weight are assigned *145cm* and *50kg* respectively.

Example 4.2 *Subject Profile for a trainee Jockey*

```
1 <subject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespa
2 ceSchemaLocation="file:///C:/NewWorkspace/csv2xml/ch7/subject.xsd">
3    <id>1</id>
4    <name>James</name>
5    <activityType>horse-racing</activityType>
6    <properties>
7        <property>
8            <name>qualification</name>
9            <scale>text</scale>
10           <value>trainee</value>
11       </property>
12       <property>
13           <name>sex</name>
```

```
14            <scale>text</scale>
15            <value>male</value>
16        </property>
17        <property>
18            <name>height</name>
19            <scale>cm</scale>
20            <value>145</value>
21        </property>
22        <property>
23            <name>weight</name>
24            <scale>kg</scale>
25            <value>50</value>
26        </property>
27        <property>
28            <name>vo2max</name>
29            <scale>mlkg</scale>
30            <value>57</value>
31        </property>
32    </properties>
33</subject>
```

4.2.3 Deployment Profile

A `deployment` profile is a formal XML representation defining interesting features and properties of an *activity* and its environment such as location, or the different sub-activities of some sport. Standard information recorded in a `deployment` includes start/end times/dates, deployment ID and location details. Specific information pertaining to an activity or deployment is defined within the `<properties>` element. Definition 4.3 illustrates the structure of the `deployment` profile.

Definition 4.3 *Deployment Profile*

```
1<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2    <xs:element name="value" type="xs:string"/>
3    <xs:element name="startdatetime" type="xs:dateTime"/>
4    <xs:element name="scale" type="xs:string"/>
5    <xs:element name="property">
6        <xs:complexType><xs:sequence>
```

```
7               <xs:element ref="name"/>
8               <xs:element ref="scale"/>
9               <xs:element ref="value"/>
10      </xs:sequence></xs:complexType>
11  </xs:element>
12  <xs:element name="properties">
13      <xs:complexType><xs:sequence>
14              <xs:element ref="property" maxOccurs="unbounded"/>
15      </xs:sequence></xs:complexType>
16  </xs:element>
17  <xs:element name="name" type="xs:string"/>
18  <xs:element name="location" type="xs:string"/>
19  <xs:element name="id" type="xs:integer"/>
20  <xs:element name="enddatetime" type="xs:dateTime"/>
21  <xs:element name="deployment">
22      <xs:complexType><xs:sequence>
23              <xs:element ref="id"/>
24              <xs:element ref="name"/>
25              <xs:element ref="location"/>
26              <xs:element ref="startdatetime"/>
27              <xs:element ref="enddatetime"/>
28              <xs:element ref="properties"/>
29      </xs:sequence></xs:complexType>
30  </xs:element>
31</xs:schema>
```

A deployment profile has 6 primary elements, of which 5 are compulsory. There can be multiple deployment `<property>` elements. As with the `fields` in the sensor profile description, the `property` elements are unbounded, and so multiple properties corresponding to a deployment can be present.

- id: An integer input represents an identification value for that profile (line 23).
- location: Basic information about location of deployment. Can be address or indoor/outdoor (line 25).
- name: A text representation that names the deployment profile (line 24).
- startdatetime: A start date and time is defined using the *dateTime* datatype for XML schema (line 3).
- enddatetime: An end date and time is defined (line 20).
- properties: One or more properties about the deployment (line 12-16). Sub-elements:

- name: Specifies the name of the property.
- scale: Specifies the format of the value recorded. For example: kg.
- value: Specifies a value for this property. For example: 10.5.

Example 4.3 shows an instance of a deployment profile for an off-road cycle race. Each of the standard elements are specified, with the start and end times 19:30 and 09:30 on the 5th/6th January (`startdatetime, enddatetime`). The domain specific properties of interest are the terrain and cycle_type details which are named in the `<name>` sub-element of `<property>` and have their values in the `<value>` element, specified by the format in `scale`. This structure allows any number of properties to be defined by the end user.

Example 4.3 *Deployment Profile for an off-road cycle*

```
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespa
ceSchemaLocation="file:///C:/NewWorkspace/csv2xml/ch7/deployment.xsd">
    <id>1</id>
    <location>outdoors</location>
    <name>cycling</name>
    <startdatetime>05-12-2010T19:30:00</startdatatime>
    <enddatetime>06-12-2010T09:30:00</enddatatime>
    <properties>
        <property>
            <name>terrain</name>
            <scale>text</scale>
            <value>offroad</value>
        </property>
        <property>
            <name>cycle_type</name>
            <scale>text</scale>
            <value>endurance</value>
        </property>
    </properties>
</deployment>
```

Example 4.4 shows a typical deployment profile for a search and rescue worker based deployment. In this case the base is identified as *Dublin*, and

the workers are on call for a period of 1 day, in a dynamically changing location.

Example 4.4 *Deployment Profile for a Search and Rescue worker*

```
<deploymentProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespa
ceSchemaLocation="file:///C:/NewWorkspace/csv2xml/ch7/deployment.xsd">
    <id>1</id>
    <name>SAR7</name>
    <location>dynamic</location>
    <startdatetime>2011-10-25T13:00:00</startdatetime>
    <enddatetime>2011-10-26T13:00:00</enddatetime>
    <properties>
        <property>
            <name>oncall</name>
            <scale>text<scale/>
            <value>true</value>
        </property>
        <property>
            <name>base</name>
            <scale>text<scale/>
            <value>Dublin</value>
        </property>
    </properties>
</deployment>
```

Ultimately, the end user is enabled to create their own schema, which represents the interesting features of a deployment. These features can later form part of a condition criteria or be used as a parameter to query for other information.

4.3 Sensor Enablement (P1)

As discussed in chapter 1, a key goal of this research is to provide services to close the semantic gap between high level, complex information needs and low level sensor data. As a first step a standard structured format is applied to the initial sensed data to enable the interpretation and analysis

of this data. The goal of sensor enablement is to provide structure to sensor data so that it can be interpreted and queried. At a high level, the process transforms data using the properties specified in the profile from its initial format (exported from sensors in csv) to XML. Sensor enablement results in the transformation of sensor data to the Sensor Web Enablement (SWE) compatible Extensible Markup Language. This section explains how the Sensor Enablement Processor (SEP) overcomes the problems involved with dealing with multiple heterogeneous sensors and shows how all sensor data is converted to a standard format to support interoperability. Data conforming to a standard is easier to interpret, modify and aids with normalisation.
Following context initialisation, the scientist has described a sensor's basic output. Sensor enablement puts in place the mechanism to automatically impose a structured data format on raw data. This is performed using a metadata driven approach, where `sensor` profiles defined during context initialisation are used to describe the sensor data. A sample `sensor` profile for an ActiLife GT3X Accelerometer was shown earlier in example 4.1.

4.3.1 Imposing a Standard Format on Sensor Data

AREA achieves the goal of providing structure to sensor data using XML terminology. This is a two-step process:

Normalise sensor data by time Each sensors output is converted to a `.csv` file representation where each row represents a set of values for a specific time interval. Each column represents some sensed data point.

Map sensor entries to generate XML An AREA system process uses the sensor profile and mapping rules from the repository to map each row of the `.csv` file to a corresponding position in XML. A `time` element is generated for each row of entries if not explicitly *sensed* in the raw sensor data.

The process of imposing this format is illustrated in Figure 4.3. The first step is a simple reorganisation of sensor data from its initial text based data (comma or tab delimited) to the `.csv` format. The `csv` organises the data in rows of entries, which can be subsequently mapped to XML. The system then begins mapping the *basic header* information - the `id`, `name`, `type`, `timing`, `scale` and `value` elements from the sensor profile to XML. There follows a 1-to-1 mapping of every entry on each row of the csv to a corresponding named element in the XML document. The `scale` and `value` details are omitted from each sensor entry and are instead recorded in an `entryproperties` element in order to avoid unnecessary duplication. In some cases, the time is not directly recorded by a sensor, but is inferred based on its sample rate and other entries. In these situations, a time will be assigned for each entry later, during normalisation, and is not performed during sensor enablement.

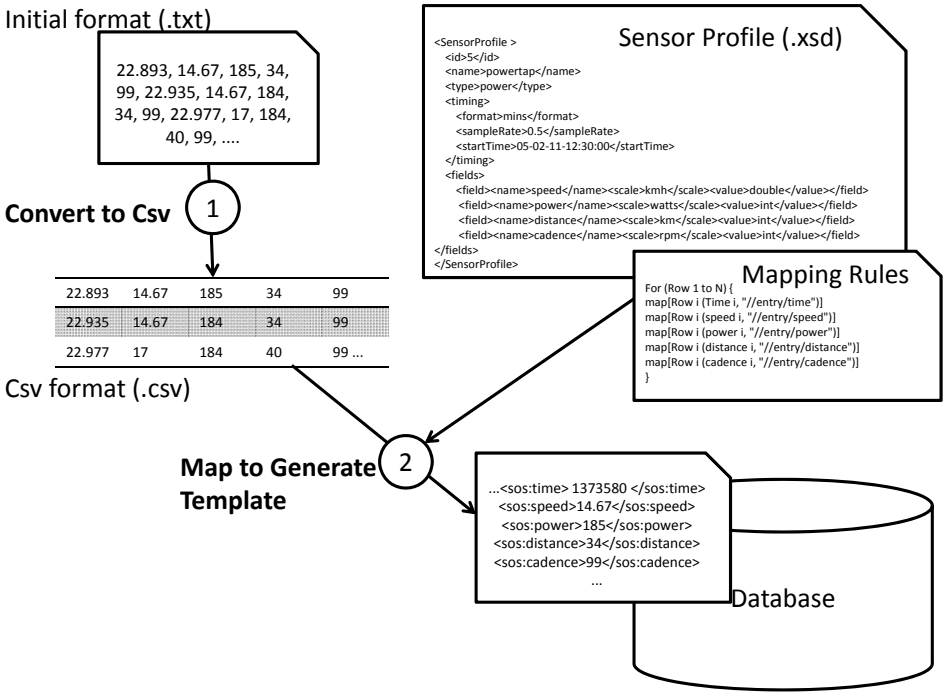

Figure 4.3: Sensor Enablement

Thus, the problem has been reduced to a simple engineering function where sensor data is to be mapped to XML using profiles and mapping rules defined during context initialisation.

Example 4.5 *Sample Extract of PowerTap data (in csv format):*

```
Time            Speed       Power       Distance    Cadence
22.893          14.67       185         34          99
22.935          14.67       184         34          99
22.977          17          164         40          99
23.019          17          162         40          98
```

To further illustrate the process, a sample set of data from a *Powertap* sensor is shown in its (post step 1) `.csv` format in Example 4.5. The sensor profile for this sensor is shown in Example 4.6.

Example 4.6 *Sensor Profile for a Powertap*

```
<SensorProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/sensorProfile.xsd">
   <id>5</id>
   <name>powertap</name>
   <type>power</type>
   <timing>
       <format>mins</format>
       <sampleRate>0.5</sampleRate>
       <startTime>05-02-11-12:30:00</startTime>
   </timing>
   <fields>
       <field>
           <name>speed</name>
           <scale>kmh</scale>
           <value>double</value>
       </field>
       <field>
           <name>power</name>
           <scale>watts</scale>
           <value>int</value>
       </field>
       <field>
           <name>distance</name>
```

```
24            <scale>km</scale>
25            <value>int</value>
26        </field>
27        <field>
28            <name>cadence</name>
29            <scale>rpm</scale>
30            <value>int</value>
31        </field>
32    </fields>
33</SensorProfile>
```

This sensor profile is then used to populate the mapping rules shown in Definition 4.4. Finally, the sensor data itself is mapped to the new elements. The mapping function maps `csv` rows in sequence. In this example, the mapping pairs are straightforward: `(Time, //entry/time)`, `(speed, //entry/speed)`, `(power, //entry/power)`, `(distance, //entry/distance)`, `(cadence, //entry/cadence)`.

Definition 4.4 *Pseudocode for mapping PowerTap values to XML*

```
For (Row 1 to N)
{
    map[Row i(Time i, "//entry/time")]
    map[Row i(speed i, "//entry/speed")]
    map[Row i(power i, "//entry/power")]
    map[Row i(distance i, "//entry/distance")]
    map[Row i(cadence i, "//entry/cadence")]
}
```

In Example 4.7, sensor data conforms to the Sensor Web Enablement (SWE) [51] standard. In particular, the Sensor Observations Service (SOS) [50] component is used in the transformation process (on archived data, rather than real-time streams). SOS is used to request, filter, and retrieve observations and sensor system information from the data repositories and this, it provides new levels of interoperability with other SWE based networks.

Example 4.7 *PowerTap Output after Sensor Enablement*

```
<sos:Sensor>
   <sos:id>5</sos:id>
   <sos:name>powertap</sos:name>
   <sos:type>power</sos:type>
   <sos:timing>
       <sos:format>mins</sos:format>
       <sos:sampleRate>0.5</sos:sampleRate>
       <sos:startTime>2011-02-05T12:30:00</sos:startTime>
   </sos:timing>
   <sos:fields>
       <sos:entrypropterties>
           <sos:speed><scale>kmh</scale></sos:speed>
           <sos:power><scale>watts</scale></sos:power>
           <sos:distance><scale>km</scale></sos:distance>
           <sos:cadence><scale>rpm</scale></sos:cadence>
       </sos:entrypropterties>
       <sos:entry>
           <sos:time>22.893</sos:time>
           <sos:speed>14.6</sos:speed>
           <sos:power>185</sos:power>
           <sos:distance>34</sos:distance>
           <sos:cadence>99</sos:cadence>
       </sos:entry>
       ...
   </sos:fields>
</sos:Sensor>
```

4.4 Context Integration (P2)

The role of the Context Integration Processor (CIP) is to apply initial contextual information to the sensor data. Typically, this is the deployment environment details and information about the subjects (participants) being monitored. The context added is basic knowledge specified in the profiles during context initialisation and is necessary to facilitate queries. The input to the CIP is the sensor enabled data output from sensor enablement. This section explains how profile information is merged with sensor data

and shows the relationships between these concepts and the sensor data.
As explained in Section 4.2, the deployment and subject profiles describe the environmental information and participant properties associated with a sensor deployment and are specified during context initialisation by scientists. The integration of this information with sensor data, is a three step process:

1. Time is normalised on all datasets to a millisecond (ms) offset from the deployment start time.

2. Sensor data output from Sensor Enablement is paired with both a `subject` (participant) and a `deployment` (activity) profile.

3. AREA transforms the sensor data and encodes contextual information using rules from the `subject` and `deployment` profiles and the XQuery Update Facility.

4.4.1 Normalisation and Synchronisation Steps

Normalisation rules can be deployed to provide uniform structure to the multiple sensor outputs. For instance, a function can duplicate sensor readings, providing an estimation for a higher-sampling sensor's timestamp. A heart rate value may only be recorded every second by a monitor. A number of functions have been devised for normalising sensor data such as a function which averages the pre and post entries to provide a new intermediate entry (used for heart rate normalisation). These averages might not always be appropriate, such as when the data is in text format, or where a non-integer representation is not valid. In cases where duplication is not required, elements are simply omitted, saving space and improving efficiency. Other normalisation functions convert numbers from arbitrary proprietary ranges to a more manageable percentage representation. For the purposes of demonstrating the AREA framework, a number of built in functions syn-

chronise and normalise the required information to allow deployment based querying capabilities.

4.4.2 Pairing Sensor Output with Subjects

As a first step to contextual enrichment, it is necessary to pair the relevant `subject` profile to the corresponding sensor output. Each subject is described by the basic subject profile information (as they are human) and also relevant domain-specific information, such as a job role or cyclist type (Example 4.2). If one was to add a new user, for instance an elderly person in an ambient assisted living environment, information relevant to that profile would extend the subject profile, such as a *medical condition* field. As AREA adapts a schema-less (XML) approach, this is simply a matter of adding a new element to the subject profile.

Example 4.8 *The Subject Profile (Cyclist)*

```
<SubjectProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/subject.xsd">
    <id>2</id>
    <name>cyclist02</name>
    <activityType>Cyclist</activityType>
    <properties>
        <property>
            <name>BMI</name>
            <scale>kgm</scale>
            <value>22</value>
        </property>
        <property>
            <name>vo2max</name>
            <scale>mlkg</scale>
            <value>60</value>
        </property>
        <property>
            <name>weight</name>
            <scale>kg</scale>
            <value>80</value>
        </property>
```

```
22        <property>
23            <name>age</name>
24            <scale>years</scale>
25            <value>21</value>
26        </property>
27        <property>
28            <name>cyclist_type</name>
29            <scale>text</scale>
30            <value>endurance</value>
31        </property>
32    </properties>
33</SubjectProfile>
```

To use contextual information relating to deployment scenarios, `deployment` profiles are paired with a group of sensor outputs and one or more subjects (participants). Pairing the relevant data to the `deployment` profile provides the information needed ahead of transforming the sensor data to include this context. This step will allow the next step of merging context with sensor data to allow queries required by the end users. The `subject` and `deployment` profiles for the *Powertap* data output from sensor enablement (Example 4.7) are shown in Examples 4.8 and 4.9.

Example 4.9 *The Deployment Profile (Cycling training)*

```
1<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
2Location="file:///C:/NewWorkspace/csv2xml/ch7/deployment.xsd">
3    <id>3</id>
4    <location>outdoors</location>
5    <name>cycling</name>
6    <startdatetime>05-12-2010-19:30:00</startdatatime>
7    <enddatetime>06-12-2010-09:30:00</enddatatime>
8    <properties>
9        <property>
10           <name>terrain</name>
11           <scale>text</scale>
12           <value>mountainous</value>
13       </property>
14       <property>
15           <name>cycle_type</name>
```

```
16            <scale>text</scale>
17            <value>fitness</value>
18        </property>
19    </properties>
20</deployment>
```

4.4.3 Merging Sensors and Subjects with a Deployment

The integration of context with sensor data is performed using an *equi-join* of a triple {`sensors, subjects, deployment`}. In each `scenario`, a deployment can have multiple subject participants and each participant can wear multiple heterogeneous sensors. To show the impact of context integration, a sample enriched data segment is presented in example 4.10. The transformation embeds the subject and deployment profiles in the scenario. The sensor data is then listed as a series of entries output from the sensor enablement process. The transformation is performed using a combination of Java and the XQuery Update Facility [73]. The result is a single XML document nesting the `sensors, subject, deployment` details. The steps taken to integrate the sensors is as follows:

1. A scenario is created.
2. Each subject involved with a deployment and the deployment details are added to the scenario.
3. The sensor data for each subject are appended to their entry in the scenario.

An example of sensor data following the merge is shown in Example 4.10. This shows the PowerTap sensor output merged with a `deployment` profile and the relevant `subject` profile. In this example, the scenario (`id=3`) in the cycling domain corresponds to a period of time from 19:30 on 5th December 2010 to 9:30 the following day. The subject involved (`cyclist02`) is added to the deployment, and a profile for the *Powertap* sensor is added to that

subject. At this point, the scenario contains all the relevant information for the activity: deployment, subjects and sensors in one XML document.

Example 4.10 *Sensor Data following Context Integration*

```
<sos:scenario xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/AREA.xsd">
  <sos:id>3</sos:id>
  <sos:location>outdoors</sos:location>
  <sos:name>cycling</sos:name>
  <sos:startdatetime>05-12-2010T19:30:00</sos:startdatatime>
  <sos:enddatetime>06-12-2010T09:30:00</sos:enddatatime>
  <sos:SubjectProfile>
      <sos:id>2</sos:id>
      <sos:name>cyclist02</sos:name>
      <sos:activityType>Cyclist</sos:activityType>
      <sos:properties>
          <sos:property>
            <sos:name>BMI</sos:name><sos:scale>kgm</sos:scale><sos:value>22</sos:value>
          </sos:property>
          <sos:property>
            <sos:name>vo2max</sos:name><sos:scale>mlkg</sos:scale>
            <sos:value>60</sos:value>
          </sos:property>
          ....
      </sos:properties>
  </sos:SubjectProfile>
  <sos:SensorDevice>
      <sos:id>5</sos:id>
      <sos:name>powertap</sos:name>
      <sos:type>power</sos:type>
      <sos:timing>
          <sos:format>mins</sos:format>
          <sos:sampleRate>0.5</sos:sampleRate>
          <sos:startTime>05-12-2010T19:30:00</sos:startTime>
      </sos:timing>
      <sos:fields>
          <sos:entrypropterties>
              <sos:speed><scale>kmh</scale></sos:speed>
              <sos:power><scale>watts</scale></sos:power>
              <sos:distance><scale>km</scale></sos:distance>
              <sos:cadence><scale>rpm</scale></sos:cadence>
```

```
38              </sos:entrypropterties>
39          <sos:entry>
40              <sos:time>22.893</sos:time>
41              <sos:speed>14.6</sos:speed>
42              <sos:power>185</sos:power>
43              <sos:distance>34</sos:distance>
44              <sos:cadence>99</sos:cadence>
45          </sos:entry>
46          ...
47      </sos:fields>
48      </sos:SensorDevice>
49</sos:scenario>
```

The application of context is a necessary step to add a layer of semantics to sensor data to allow more complex analysis. Following the application of basic contextual enrichment, the query potential of the sensor data has improved. More focused queries on the data output from the CIP can now be made directly by the scientist based on the deployment activity, the participants involved and comparisons across multiple information sources (both sensor data and profile information). For example, XQuery can now be used to find the highest power output from `Powertap` where the cyclists BMI is greater than 21 and location is outdoors.

4.5 Transforming Sensor Data and Metadata (P3)

At this stage, the queries available to the end user are still limited, and updates to the data cannot be defined without knowledge of XQuery or some programming language. Further enrichment will be driven by a knowledge worker based on profiles and the schema. More complex information specific to a domain cannot yet be easily defined by a knowledge worker. In order to facilitate interaction with the end user, the *Data Transformation Processor (DTP)* had been defined.

The goal of the user-controlled transformation process is to allow the knowl-

edge worker to encode knowledge at an abstract level using underlying sensor data or external context as evidence supporting the existence of this knowledge. This will allow other users - both knowledge workers and end users to query for specific information using simple methods (a standard query language) to obtain this potentially complex information, such as the computation of energy expenditure using multiple sensor fields from multiple sources. Information obtained can then be *reused* to discover new knowledge. This section presents a brief introduction to the data transformation part of the AREA framework, which is covered in detail in Chapter 5.

The information added by the scientist can be specialised or domain specific information, observable by their expertise in the field. There is a two stage process to data transformation. First, an `update` or `extension query` is defined by a knowledge worker and stored in AREA's repository. The second stage is the application of these `queries` on a data set. During this process, sensor data or its metadata description (schema) are altered, based on the `query` definition.

The application of these user-defined algorithms results in semantic enrichment of the underlying XML documents. Querying the results provides the ability to improve algorithms and the identification of information which can be used subsequently by the scientist to alter strategies or behaviours of those being monitored. The next section describes the system-defined enrichment process called automated activity analysis which further enhance the functionality of the AREA framework.

4.6 Automated Activity Analysis (P4)

The goal of the *Automated Activity Analysis (AAA)* processor is to allow the system to use metadata from the AREA repository to discover new information without the intervention of the scientist. The knowledge workers often do not appreciate the full extent of what information can be provided

by a sensor deployment and a system driven approach allows AREA to suggest new levels of context. This section briefly introduces and describes the key functions of automated activity analysis.

The design of the framework allows for statistical analysis of sensor data and metadata. There are three modules in AAA: baseline, cluster and metadata analysis. In baseline analysis, AREA automatically generates a statistical analysis of a single context elements entries - that is all records of a specific entry. In doing so, a new instance of a new profile is generated, called a `dataset` profile and is stored in AREA's repository. Cluster analysis makes use of `dataset` profiles to populate key properties of a clustering algorithm to allow the detection of clusters of similar data. Thus, AREA can exploit information within the data to potentially improve techniques to discover more information. In other words, cluster analysis may result in improved classification of similar data generated by previous `update query` profiles.

In the third module of AAA, metadata analysis, AREA again exploits the metadata generated by baseline analysis. The values within the update element of an `update/extenstion query` profile have been classified by baseline analysis. AREA uses this classification information to experiment with the parameters used to generate or discover information. For instance, where a mutable variable forms part of an algorithm to detect energy expenditure, AREA can deploy several similar values in parallel in an attempt to discover the most effective value to use, or input potential future values to predict the effect this would have on other data. Chapter 6 fully describes the AAA processor.

4.7 Summary

This chapter discussed the processes and components that form the AREA framework. It began by explaining the context initialisation stage and how this is applied to the data as part of sensor enablement and then described

the process of context integration. The chapter proceeded with a brief overview of how the user-controlled `query` profile instances interact with context and sensor data to acquire more knowledge from the data gathered in a deployment. Then, an overview of the functionality of AREA's AAA processor was provided. The AREA framework provides the knowledge worker with the ability to meet their information needs while maintaining a generic method of interpretation and processing of heterogeneous sensor deployments. In the next chapter the functionality and specification of the user-controlled data transformation process is described.

Chapter 5

User-Controlled Transformations

The previous chapter described the architecture of the AREA framework and briefly introduced the Data Transformation Processor (DTP). Following the Sensor Enablement and Context Integration processes structure and context has been imposed on the sensor data. Each individual sensor output has been assigned the relevant subject (wearing that sensor) and the deployment (environment in which it was deployed). In order to properly query the information available following an experiment, a number of transformations are required. There is a requirement for a mechanism to allow a knowledge worker to incorporate new context and interesting observations and apply new knowledge in order to extend the dataset. This user-controlled data transformation is now presented in this chapter.

In section 5.1, the constructs involved in defining knowledge are defined. In section 5.2, the `update` and `extension query profiles` are introduced, which represent the user-defined instruction to extend the dataset. Section 5.3 describes how update and extension queries are applied on a dataset and schema respectively using transformations to XQuery. Finally, in section 5.4 a summary of the user-controlled transformation process is presented.

5.1 Generating Context in AREA

In this section, the key constructs created by extending the dataset: `context` elements are described. Following sensor enablement, context integration and the preliminary data transformation steps (i.e. synchronisation and normalisation) of the AREA framework, sensor data has become enriched and associated with a `sensor`, `subject` and `deployment`. The result is an XML document for each experiment which has an associated schema description specifying the format and content of the sensor dataset. The elements within these documents are `context` elements which can be queried using XQuery expressions. The `context` elements can be used as parameters to `conditions` and `updates` or as part of an XQuery expression when extending the dataset further.

System-Defined Context Elements

Basic `context` elements are created and populated as a result of Sensor Enablement (i.e. the imposition of structure) and Contextual Enrichment processors (as described in Chapter 4), based on information in the `sensor`, `subject` and `deployment` profiles. These are *system-defined* elements and allow for the querying and retrieval of enabled sensor data based on deployment or subject characteristics. The `context` elements for a *gt3x* sensor following sensor enablement are the entry name elements (e.g. `<x>, <y>`). Example 5.1 shows a segment from a *gt3x accelerometer* output which has been enabled. Following Context Integration, additional elements such as `<id>` and `<startdatetime>` are added to the dataset by the system during integration (using the mappings discussed in section 4.4) and become new `context` elements.

Example 5.1 *Single Entry XML and Raw Data*

```
<Entry>                          Raw data: 87, 225, 103, 2, 0, 1, ...
    <x>87</x>
    <y>225</y>
    <z>103</z>
    <steps>2</steps>
    <lux>0</lux>
    <incline>1</incline>
    <timestamp>4000</timestamp>
</Entry>
```

User-Defined Context Elements

User-defined `context` elements are XML elements defined by a knowledge worker using information from other elements. These elements extend the structure of the sensor data and are created as the result of updates defined in a new construct, the `extension query profile`. The next section will show how context elements (both *system* and *user-defined*) are used as input parameters to `conditions` and `updates`, and how new user defined elements are defined and created or populated by an end user.

User-defined context elements allow the user to impose concise information into the sensor data and therefore provide new constructs to query for new knowledge. Consider the requirement of detecting a *"serve"* event in tennis, using the location coordinates of two players (provided by *Ubisense* sensors). This cannot be queried by the user without some pre-processing of what a serve means with respect to two moving players. It is `query profiles` which are *defined* and *applied* on sensor data to detect this intermediate knowledge and allow for the detection of serves. In doing so, a new context element is defined and encoded in the sensor data. Another example of a user-defined `context` element is the <`speed_class`> element, which is populated with a classification metric such as *"low", "moderate"* or *"high"* based on the <`speed`> system element. The result of adding this context element to the sensor data is shown in example 5.2. The new context element

(<speed_class>) has been encoded as a child of the <Entry> element.

Example 5.2

```
<Entry>
   <speed>24.6</speed>
   <speed_class>moderate</speed_class>
   ...
</Entry>
...
```

To explain how the user defines and populates context elements, the next section presents the process of defining `read, update` and `extension` query profiles.

5.2 Using Profiles to Query and Extend the Dataset

In this section, the structure of a `query` profile is described. The goal of the `query` profile is to allow the user to both query for information and to update and extend the dataset, to facilitate future queries of complex requirements using simple query techniques. An `extension query` allows a knowledge worker extend the AREA schema and an `update query` allows the definition and and application of new context to sensors by specifying conditions that lead to an `update`.

There are three operations which the end users wish to perform:

1. Query for sensor data.
2. Update the value of a context element.
3. Update AREA's schema to allow changes to the structure of the datasets.

An Event-Condition-Action based format allows the user to perform each operation by defining query profiles. These will enable the end user to query, update and extend sensor datasets in a structured approach. The elements are defined by the (partial) XML schema template in definition 5.1. The full schema can be found in Appendix B.

Definition 5.1 *Query Profile (abridged)*

```
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2     <xs:element name="updateQuery">
3         <xs:complexType>
4             <xs:sequence>
5                 <xs:element ref="name"/>
6                 <xs:element ref="doc"/>
7                 <xs:element ref="target"/>
8                 <xs:element ref="condition"/>
9                 <xs:element ref="update"/>
10            </xs:sequence>
11        </xs:complexType>
12    </xs:element>
13    <xs:element name="update">
14        <xs:complexType>
15            <xs:sequence>
16                <xs:element ref="address"/>
17                <xs:element ref="content"/>
18            </xs:sequence>
19        </xs:complexType>
20    </xs:element>
21 </xs:schema>
```

There are 5 main sub-elements of the `updateQuery` element in this profile, each playing a part in querying and updating of sensor data. Similarly, the `extensionQuery` and `readQuery` elements share many of these elements, as can be seen in Appendix B. They are described below.

1. name: A unique name for this specification (line 5).
2. doc: The location where this query is to be applied (line 6).
3. target: XPath expression pointing to the location of the result required (read queries) (line 7).
4. condition: An optional condition element before an update can be triggered (line 8).
5. update: The update element is also optional and has sub-elements `address` and `content` (line 9, 13-20). These specify the update instructions.
    - address: An XPath expression for the context element to be updated or populated.
    - content: A value or math equation for the value to be placed in the context element.

To illustrate the `query` profiles, example 5.3 from the cycling domain is presented. In cycling, the sport scientists defined a cyclists heart rate value

of over 180, to correspond to *high* intensity. As a result they have defined a profile which defines a *high* intensity metric based on all occurrences where the heart rate (HR) is above 180. The <`update`> element provides the instruction to update the node <`intensity`> at the path specified and populate it with the content *"high"*. An `update` profile is invoked to populate `<intensity>` with the content *"high"* where the condition is satisfied. The result of *applying* this `query` profile on the sensor data is the creation of an `<intensity>` context element, populated with *"high"* at all instances where HR > 180. For clarity, for this and all further query profile examples, the parent node `query` is omitted.

Example 5.3 *Update Profile: User Searching for High Intensity Heart Rate*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
ocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
   <name>heartIntensity</name>
   <doc>s3.xml</doc>
   <target>/sensorProfile[name="garmin405"]/Entry</target>
   <condition>
       /sensorProfile[name="garmin405"]/Entry/HR > 180
   </condition>
   <update>
        <address>/sensorProfile[name="garmin405"]/Entry/intensity</address>
        <content>"high"</content>
   </update>
</updateQuery>
```

This section has provided a brief overview of the structure and layout of the `query` profile. The process of *applying* update and extension profiles and thus, triggering queries or updates is discussed further in Section 5.3.

5.2.1 Specifying Conditions

Definition 5.1 briefly described the `condition` element of a query profile. Definition 5.2 described the allowable content and syntax of this element. The condition is specified by the user (via a user interface) using the grammar shown in Definition 5.2. This allows the user to make

use of boolean operators (`booleanOps`) to bind a number of smaller conditions (`check`). These in turn can consist of equivalence checks (using `relationalOps`) and numbers (`number`) or `context` elements as parameters. The `conditionstatement` token provides the user with the ability to target updates or discover information based on specific criteria.

Definition 5.2 *E-BNF for conditionstatement*

```
conditionstatement = check [{booleanOps check}]
check = [NOT] number relationalOps number
relationalOps =  >= | != | == | <= | > | <
booleanOps = AND | OR
number = [sign] {digit} [decimalPt {digit}] | ContextElement
decimalPt = .
digit = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
ContextElement = text
```

Example 5.4 shows a sample `conditionstatement` for identifying the occurrence of a *whipping* instance in the domain of horse-racing (a jockey whipping a horse during a race). In this example, the `context` elements evaluated are a mix of system (`<x>`) and user-defined (`<gait>`) elements. The `condition` checks if the `gt3x accelerometer` on the left-hand wrist (`LHWrist`) has all three axes (`x,y,z`) value greater than `MAXVALUE` (the upper threshold for the axis value) and that a `gait context` element has the text value *"fast-canter"*. This is a real world requirement necessary for scientists to identify a fast cantering horse using accelerometers. The `conditionstatement` is defined using the grammar provided by the E-BNF specification.

Example 5.4 *Conditionstatement to identify a "whipping" instance for a jockey*

```
<condition>
    //sensorProfile[location="LHwrist"]/Entry/x > 'MAXVALUE' AND
    //sensorProfile[location="LHwrist"]/Entry/y > 'MAXVALUE' AND
    //sensorProfile[location="LHwrist"]/Entry/z > 'MAXVALUE' AND
    //sensorProfile[location="saddle"]/Entry/gait = 'fast-canter'
</condition>
```

The `<condition>` element's `conditionstatement` token allows the knowledge worker to specify the relevant criteria for identifying an event or occurrence. It is the role of the `<update>` element to specify what happens following identification. These elements are described now.

5.2.2 Generating the Metadata

The `<update>` element of the query profile specify modifications to and generation of sensor data and metadata. In Example 5.5, the `<address>` element contains the `xpathexpression`: *//gt3xaccelerometer[location=LHWrist]/entry/gait*. This XPath expression is the address of the node targeted for update. Every `<update>` must have an `<address>` associated with it. In the example shown, an `<address>` element contains an XPath expression to a context element `whip`. The result is to be stored as a new element in the dataset from a *gt3x accelerometer* located on the left wrist, all of which is specified by the `address`.

Example 5.5 *Update Element for a Whip Classification Query*

```
<update>
  <address>//gt3xaccelerometer[location=LHWrist]/entry/whip</address>
  <content>"leftwhip"</content>
</update>
```

In this example, the address refers to a *gt3x Accelerometer* located on the left hand wrist (`LHWrist`) and forms part of the query profile for the classification of a whipping instance, shown later in Example 5.7. The target node (`<whip>`) will be the one affected by the `query` profile. Depending on which write operation is taking place (an update or an insert), the data within an existing element is altered or data is altered and metadata is extended. If its the later operation, the new context element (`<whip>` for example 5.5) is added as an *optional* element to the XML schema. In other words, the schema update adds an entry for `whip` with a *MinOccurs = 0*. After this,

an update is invoked to make changes to the sensor data. The alteration of the data is specified in the `<content>` element discussed in the next section.

5.2.3 Content Transformation

It is the role of the `<content>` element to specify the update to take place on the node specified by the `<address>` element. As was shown in Definition 5.1, the `<content>` element contains the token `transformstatement`. The Extended BNF (E-BNF) grammar for this is shown in Definition 5.3.

Definition 5.3 *E-BNF for transformstatement*

```
transformstatement = part [{(sign) part}]
part = num [{(operator) num}]
num = number | ( transform )
number = [sign] {digit} [decimalPt {digit}]
operator = / | *
sign = - | +
decimalPt = .
digit = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

This grammar for transforming sensor data allows for mathematical calculations and the input of text values. As with the `conditionstatement` the transforms can be made of smaller calculations and combined. It allows the scientist to utilise a number of system defined functions to aid with querying for more advanced information. These AREA functions are shown in Table 5.1, with their purpose and expected input. `SUM` and `COUNT` are implementations of the `fn:sum` and `fn:count` functionality of XQuery. `RANGE` allows querying of data between two timestamps. `PEAKS` allows the discovery of signal peaks in a single-data value mapping. The output for each is XML. Example 5.6 shows a definition called *accelMetric*. This is a complete `update query` profile for the calculation of *acceleration metric*. In the cycling domain it was found that by taking 20% of the mutiplication of `x, y, z` values for an accelerometer gave an estimation for the amount of acceleration activity. The definition of this `query` is shown here:

Name	Description	Input
SUM(dataset)	Accumulation of adding values	1 element dataset
COUNT(dataset)	Number of dataset entries	1 element dataset
RANGE(dataset), time1, time2	Range of dataset for int seconds	1 element dataset, dateTime
PEAKS(dataset, threshold)	Identifies Peaks in data	1 element dataset, double
DURATION(dataset, time)	Specifies window of time	1 element dataset, double

Table 5.1: AREA Transformation Functions

Example 5.6 *Update Query: Calculate acceleration metric*

```
<updateQuery>
   <name>accelMetric</name>
   <doc>subj01_30Jan11</doc>
   <target>/GT3XAccelerometer[location=LHWrist]/entry/accelMetric</target>
   <update>
        <address>/GT3XAccelerometer[location=LHWrist]/entry/accelMetric
        </address>
        <content>
        ((/GT3XAccelerometer[location=LHWrist]/entry/x *
        /GT3XAccelerometer[location=LHWrist]/entry/y *
        /GT3XAccelerometer[location=LHWrist]/entry/z) / 5)
        </content>
   </update>
</updateQuery>
```

The `<doc>` element specifies the location of the document, corresponding to an experiment, to be queried. This is an XML representation of the sensors, nested within a subject and a deployment environment. The `<target>` element identifies the node on which the query will be based. In the case of read queries, this is the node from where the results will be retrieved. For write queries, it is the root node for subsequent node locations. When *applied*, this `update query` will update the node (`<accelMetric>`) with the transformation specified in the `<content>` element. This transform computes `((x*y*z)/5)` to generate a single value for `accelmetric` (per entry). The `query` profile is converted to XQuery using the AREA transform process defined in Section 5.3.

A `query` profile is shown in Example 5.7. This example shows an `update query` profile for the definition of whipping horse by a jockey (using the left arm). The criteria for defining this `update query` is the three axes of an accelerometer on the left wrist reaching their maximum value while a `<gait>` context element on the saddle accelerometer has the content *fast-canter*. It is assumed that `<gait>` (line 7) has formed part of an earlier `extension query`. This node will have its value set to *"leftWhip"* (line 11) after the transformation process in cases where the condition is satisfied.

Example 5.7 *Query Profile to detect a left-handed horse-racing whip*

```
1 <updateQuery>
2   <name>Left-Handed-Whip</name>
3   <doc>jockeyDM_22Jun11</doc>
4   <target>/GT3XAccelerometer[location=LHWrist]</target>
5   <condition>
6       //entry/x > 'MAXVALUE' AND //Entry/y > 'MAXVALUE' AND //Entry/z > 'MAXVALUE'
7       AND //GT3XAccelerometer[location=saddle]/Entry/gait = 'fast-canter'
8   </condition>
9   <update>
10       <address>/Entry/whip</address>
11       <content>leftWhip</content>
12  </update>
13 </updateQuery>
```

To allow the application of the update query in example 5.7, an extension query shown in example 5.8 must first be applied. This will update the AREA schema to allow the `<whip>` node to be present in subsequent scenario datasets.

Example 5.8 *Extension Profile to extend the schema with a whip node*

```
<extensionQuery>
   <name>Whip</name>
   <doc>AREA</doc>
   <target>//GT3XAccelerometer/Entry</target>
   <update>
        <address>/GT3XAccelerometer/Entry/whip</address>
   </update>
</extensionQuery>
```

Example 5.9 shows an `query` profile with instructions to convert a measurement value from miles to kilometres. This `update query` simply populates a node called `kmcovered` in the scenario dataset with the kilometre value equivalent to the mile-based (`miles`)) original. The result is two entries measuring distance, one in kilometres, the other in miles.

Example 5.9 *Query Profile to Convert Miles to Kilometres*

```
<updateQuery>
   <name>milesToKm</name>
   <doc>cycle2_01Feb11</doc>
   <target>//entry/kmcovered</target>
   <update>
        <address>//entry/kmcovered</address>
        <content>//entry/miles * 1.609344</content>
   </update>
</updateQuery>
```

The extension query to facilitate the structural change of adding the `kmcovered` node is shown in example 5.10.

Example 5.10 *Extension Profile to extend the schema with a whip node*

```
<extensionQuery>
   <name>Whip</name>
   <doc>AREA</doc>
   <target>//Powertap/Entry</target>
   <update>
        <address>/GT3XAccelerometer/entry/kmcovered</address>
   </update>
</extensionQuery>
```

To illustrate the affect, a small sample of sensor data altered by these two query profiles is shown in example 5.11.

Example 5.11 *Data Before and After Applying the milesToKm Extension*

```
...
<entry>
  ...
  <miles>10.1</miles>
</entry>
<entry>
  ...
  <miles>10.3</miles>
</entry>
<entry>
  ...
  <miles>10.5</miles>
</entry>
...
```

```
...
<entry>
  ...
  <miles>10.1</miles>
  <kmcovered>16.254</kmcovered>
</entry>
<entry>
  <miles>10.3</miles>
  <kmcovered>16.57</kmcovered>
</entry>
<entry>
   <miles>10.5</miles>
   <kmcovered>16.898</kmcovered>
...
```

This section has shown how `query` profiles specify the update and extension instructions. It has introduced `update` and `extension` profiles and explained how the end user specifies `conditions` and `updates` using defined grammars. Upon definition, the profiles are stored as metadata in the metabase similar to triggers in relational database systems. When a knowledge worker *applies* a query, sensor metadata is created (in the case of extension queries). This section has also illustrated the effect this has on a dataset through a series of examples. It is during this *application* stage that the `context` elements are created and populated or simply returned (in the case of read queries). As with the forming of definitions, it is the end user who specifies the `query` required in specific deployments. The system uses XQuery and AREA functions implemented in Java to update the individual sensor data sources and the profiles for those datasets. This enabling these new concepts or `context` elements to form part of later queries or for the formation of new `query` profiles. The definition and *application* of profiles are specified using the query

definition utility of AREA and future `query` profiles can be defined based on the results obtained from this application. This query formulation is examined in detail in chapter 7, for each of the deployments described in chapter 3.

When a `query` profile is *applied* on a dataset, the individual `<doc>, <target>, <condition>, <update>` element values of the definition are mapped as parameters to a single AREA update function. Where the condition is satisfied, the system checks the `xpathexpression` address specified for the target node. If this does not exist, it is first defined within a <`update`> element and created as a child of the preceding node in the AREA schema using XQuery. The `transformstatement` is evaluated using Java, and an XQuery extension (XQuery Update Facility) inserts the result into the target element as a child element or replaces the value in an existing element within the scenario dataset. The mapping from AREA's XML representation to XQuery is fully specified in the next section.

5.3 Transforming AREA Commands

5.3.1 Introduction

This section will show how the high-level AREA profiles and commands have a direct mapping to XQuery expressions. It is easiest to show this by classifying the different user operations into read, data update and metadata update expressions (extensions). AREA deliberately keeps its interface with the end user as simple as possible and abstracts the logic and syntax of XQuery from non-IT users. There are three commands required by the end user. These are outlined below.

1. A pure retrieval query.
2. Update the content of existing sensor data. (Data update)
3. Update the structure of sensor data. (Metadata update)

This section is structured as follows. Section 5.3.2, examines the process of read queries, those which do not alter any data or metadata. Section 5.3.3 describes how data updates are performed, and section 5.3.4 details how the schema can be extended.

5.3.2 Command 1: Read Queries

The first case to examine is the query for basic sensor data. This is the basic requirement of the end user: to find information recorded by sensors for offline analysis. There is no change in data or metadata where read queries are applied and simple relevant results are returned to the user. Some of these queries are conditional, i.e. they require the satisfaction of some condition to discover the required information. For instance, a simple query may be to return all instances of heart rate, while a conditional query may require the heart rate to be above a certain threshold in order to be returned.

Example 5.12 *Find heart rate values greater than 190*

```
<readQuery>
   <name>findHighHR</name>
   <doc>cycling01_30Jan11</doc>
   <target>/sensorProfile[name="garmin405"]/Entry</target>
   <condition>
       /sensorProfile[name="garmin405"]//Entry/HR > 190
   </condition>
</readQuery>
```

Example 5.12 presents a sample read query. Assume there is a requirement from the cycling domain, which seeks heart rate data for all instances where a cyclists heart rate exceeds 190bpm (beats per minute). The XQuery expression for read queries is shown in Definition 5.4. The generic expression requires 3 tokens (`P1, P2, P3`) from the query profile specified in Definition 5.5.

The mapping pairs are straightforward: `(doc, P1)`, `(target, P2)`, `(condition, P3)`. Thus, the tokens P1, P2, P3 in the XQuery expression are replaced with the values from `<doc>`, `<target>` and `<condition>`. Each query is stored in the AREA repository upon definition. The XML format facilitates future query application on the same scenario dataset following any changes. Similarly, the queries can be performed on other documents.

Definition 5.4 *AREA: Simple Retrieval Query (XQuery Template)*

```
for $a in doc("P1")P2
where P3
return $a
```

Definition 5.5 *AREA: Read Query Function*

```
<readQuery>
    <name>NAME</name>
    <doc>P1</doc>
    <target>red{P2}</target>
    <condition>P3</condition>
</readQuery>
```

All the relevant information for querying the sensor deployment in Example 5.12 (`cycling01_30Jan11.xml`) is provided within the query profile. In order to detect the heart rate values as required in Example 5.12, the query is expressed based on the mapping of the tokens. In other words, the XML contents are converted to the relavent XQuery, as can be seen in example 5.13.

Example 5.13 *Find heart rate values greater than 190 (XQuery)*

```
for $a in doc("cycling01_30Jan11.xml")//HR
where //HR > 190
return $a
```

5.3.3 Command 2: Update Queries

Update queries facilitate changes to the dataset based on some criteria. AREA facilitates the application of updates using one of two functions.

In the first, existing nodes or elements within a scenario dataset are updated with new context. The update function is presented in definition 5.6, taking the parameters from the `update query` (definition 5.7), as labelled P1-P5. Again the mapping is straightforward: `(doc, P1)`, `(target, P2)`, `(condition, P3)`, `(address, P4)`, `(content, P5)`. Thus, the tokens P1, P2, P3, P4, P5 in the XQuery expression are replaced with the values from `<doc>`, `<target>`, `<condition>`, `<address>` and `<content>`. The update query profile is stored in AREA's repository to be applied when necessary. As the mapping is fixed, it allows the user to periodically update entire datasets of information using the same query profiles. A set of specialised update queries can be *applied* on a scenario, reducing the requirement of building new queries.

Definition 5.6 *AREA: Update Function (XQuery Template)*

```
for $a in doc("P1")P2
where P3
return replace value at node P5 with P4
```

Definition 5.7 *AREA Update Query Function*

```
<updateQuery>
   <name>NAME</name>
   <doc>P1</doc>
   <target>P2</target>
   <condition>P3</condition>
   <update>
       <address>P4</address>
       <content>P5</content>
   </update>
</updateQuery>
```

To illustrate the mapping, example 5.14 shows a *"lateral"* pedal pivot in cycling. This update is required by sports scientists in order to discover the position at which most power is exerted upon the pedal by the cyclist. Based on `(x,y)` coordinates of an accelerometer located on the cyclists ankle, it is similar to the elements in a read query, with an additional element

<update> and its sub-elements as described by Example 5.1. The goal is to populate a <pedal_pivot> element with the content *"lateral"*.

Example 5.14 *Update Query Profile to define a lateral pivot point*

```
<updateQuery>
   <name>identifyLateralPivot</name>
   <doc>cyclist01_30Jun11</doc>
   <target>//gt3xaccelerometer[location=ankle]</target>
   <condition>
       //gt3xaccelerometer[location=ankle]//x < 200
       AND //gt3xaccelerometer[location=ankle]//x > 120
       AND //gt3xaccelerometer[location=ankle]//y > 10
       AND //gt3xaccelerometer[location=ankle]//y < 200
   </condition>
   <update>
        <address>//gt3xaccelerometer[location=ankle]/entry/pedal_pivot</address>
        <content>"lateral"</content>
   </update>
</updateQuery>
```

The XQuery equivalent for example 5.14 is shown in example 5.15.

Example 5.15 *XQuery for defining a lateral pivot point*

```
for $a in doc("cyclist01\_30Jun11.xml")//gt3xaccelerometer[location=ankle]
where //gt3xaccelerometer[location=ankle]//x < 200
       AND //gt3xaccelerometer[location=ankle]//x > 120
       AND //gt3xaccelerometer[location=ankle]//y > 10
       AND //gt3xaccelerometer[location=ankle]//y < 200
return replace value of node //gt3xaccelerometer[location=ankle]/entry/pedal_pivot
with "lateral"
```

In this example, the tokens P4 <address>: "//gt3xaccelerometer[location= ankle]/entry/pedal_pivot" and P5: <content>: "lateral" are mapped to an XQuery Update function (*replace value of node pedal_pivot with lateral*). Should the scientist wish to apply this update on another scenario's dataset, the <doc> value can be changed to reflect this and the update query applied as before.

The second AREA update function inserts new nodes populated with data into a scenario's dataset. The schema is assumed to have been extended to reflect these new nodes (using the approach described in the next section). This update uses a different XQuery expression to the update of existing data. The XQuery expression for these updates is of the form: *insert node* $< / >$ *after xpathexpression*. The template for the insert update function is shown in definition 5.8. The input parameters are the same as those in definition 5.7.

Definition 5.8 *AREA: Insert Update Function (XQuery Template)*

```
for $a in doc("P1")P2
where P3
return insert node <P4>P5<P4> after P4.parent
```

As with the previous update, the mapping is straightforward: `(doc, P1)`, `(target, P2)`, `(condition, P3)`, `(address, P4)`, `(content, P5)`. Thus, the tokens P1, P2, P3, P4, P5 in the XQuery expression are replaced with the values from `<doc>`, `<target>`, `<condition>`, `<address>` and `<content>`. The AREA insert function (definition 5.9) differs from the update function in that a `new node` is inserted with its data at the `address` specified.
The mappings are now illustrated through a sample update query: example 5.16. This shows the structure of an update query to populate a node `intensity` with text based on a heart rate value.

Example 5.16 *Identifying a High Intensity*

```
<updateQuery>
   <name>highheartintensity</name>
   <doc>jockey01_12May11</doc>
   <target>/HRM/entry</target>
   <condition>
       //HR > 180
   </condition>
   <update>
        <address>//entry/intensity</address>
```

```
        <content>high</content>
    </update>
</updateQuery>
```

The scenario dataset update for example 5.16 is mapped to the XQuery function in example 5.17.

Example 5.17 *High Intensity Scenario Update*

```
for $a in doc("jockey01_12May11.xml")//entry
where //HR > 180
return insert node <intensity>high</intensity> after //entry
```

5.3.4 Command 3: Metadata Update Queries

There will be situations where the scientist will require new knowledge to be added to the schema, requiring a schema extension or metadata update. In effect, a new node is created in the XML metadata tree. In example 5.16, the schema has an `<intensity>` node created at the address specified. AREA defines this node as an *optional element*, thereby not enforcing its requirement for each parent node in the datasets. Example 5.18 shows the XQuery representation of the edits required to the XML schema for the high intensity extension query (example 5.16). The target document is the AREA schema (`.xsd` format).

Definition 5.9 *AREA: Metadata Update Function (XQuery Template)*

```
insert node P3 as last into doc("P1.xsd")/P2
```

Definition 5.10 *AREA Extension Query Function*

```
<extensionQuery>
    <name>NAME</name>
    <doc>P1</doc>
    <target>P2</target>
    <update>
        <address>P3</address>
    </update>
</extensionQuery>
```

To illustrate the application of this function, the XQuery expression for the extension of the AREA schema in example 5.16 is shown in example 5.18.

Example 5.18 *Intensity AREA Schema Update*

```
insert node  <intensity> as last into doc("AREA.xsd")//HRM/entry
```

In example 5.18, the schema has a new node (`intensity`) inserted at the location specified (`//entry`). This will facilitate structural changes to scenario datasets, i.e. allow the user to apply different levels of intensity using AREA update queries.

5.4 Summary

The user-controlled transformation process defined in this chapter allows the user to interact and alter the sensor data gathered. Two types of users are envisaged. A knowledge worker can now apply progressively more complex context to the data, and scientists or technicians can now query this data using simpler methods. The knowledge worker adds this information by defining conditions and updates within new `update` and `extension query` profiles, which are then applied on relevant datasets. Chapter 7 will show how a simple user interface allows the expression of queries, and easy definition of profiles. AREA provides a structured method of applying data and metadata updates and thus, provides the ability to iteratively define and improve user-defined algorithms to achieve the information needs of the user. The schema representation of the data is continually updated in AREA by the knowledge worker to reflect the extensions, and the resulting dataset is queryable using XQuery or simple user interface. All query profiles are stored in AREA's repository, allowing subsequent updating of sensor data and metadata using simple means. However, to make use of automated techniques would improve productivity and lead to the discovery of new information not previously known by the deploying scientist. In the

next chapter, the approach to automated activity analysis is detailed. Its inclusion allows the user to take advantage of automated algorithms and metadata to potentially improve the results obtained.

Chapter 6

Automated Activity Analysis

In the previous chapter, the AREA framework, which facilitates the definition and application of knowledge on a sensor dataset was described. The query profiles can be applied where necessary to further enrich data with extracted information. This process allows later analysis of complex data using simple structured queries. However, all of this assumes the user knows precisely what to search for. In some cases, the user does not appreciate the potential benefits from the vast amounts of sensor data gathered. In this chapter, a toolkit for AREA is presented, which will aid the user in identifying information not known in advance.

Section 6.1, provides a broad overview of the approach to automated analysis, and the potential benefits for the end user. Section 6.2 presents the methodology for analysing and classifying data returned from queries or updates. Section 6.3 describes the process of applying clustering algorithms to detect patterns in the sensor data using metadata from the AREA repository. Section 6.4, explains how the results from baseline analysis can influence the user and show how techniques can be altered to discover new or more accurate information. Section 6.5 presents a summary of the functionality of the AREA framework's automated analysis capabilities.

6.1 Broad Strategy to Automating Analysis

The process of closing the semantic gap between information needs and heterogeneous sensor data began with defining three constructs: sensor, subject and deployment profiles to describe sensors, participants and sensor deployments respectively. For each sensor deployment, the relevant profiles have been merged. The end user can subsequently interact with the sensor data through a fourth construct, the `query` profile to update and extend the data set. This facilitates the application of users' own techniques and algorithms to identify, detect and impose context. This suits a user who knows precisely what the information needs are, and the process to acquire this information. However, due to the complexity of sensors output and the vast quantity of information gathered, there is often some characteristics or properties of a deployment not noticed or understood by the user. In many cases, the user does not fully understand the potential benefits of analysing the sensor data to discover this information. This chapter presents a toolkit which automatically analyses previously defined profiles and sensor data in an effort to discover interesting information, patterns or trends.
Automated Activity Analysis (AAA) is made up of 3 modules: `baseline`, `cluster` and `metadata analysis`. The goal is to use existing profiles as leverage to discovering new information automatically. Each profile type is re-used as input to one or more of these modules.

Baseline Analysis

`Baseline analysis` is the process of analysing the origin and content of a deployment's context elements. This is necessary to provide the system with a better understanding of valid data ranges and to facilitate the subsequent application of clustering and metadata analysis techniques. Baseline analysis involves the execution of a number of generic AREA functions to gather statistical information about the data within the context element. Analysis

includes a function which interprets the source for the context elements content (if not sensed directly), such as the update process undertaken to create this content. The results obtained by executing these functions are stored in a new construct, a fifth profile, called a `dataset` profile. This chapter will show how this construct's content will then feed into the cluster and metadata analysis modules.

Cluster Analysis

Chapter 5 described AREA's approach to allowing users to define classifications using query profiles. In some cases, initial classifications may not be entirely accurate, and may be more of an estimation. In addition, some patterns in sensor data may be missed altogether by the end user. This chapter shows how `cluster analysis` is exploited by AREA without the need for user interaction. This chapter will demonstrate how metadata (from the `dataset` profile) is used to populate key properties of a clustering algorithm to allow the detection of clusters of similar data.

Metadata Analysis

To prompt the user with potentially better techniques (i.e. Better `query` profile definitions) there is an additional module of `metadata analysis` provided in section 6.4. This makes use of information from baseline analysis to apply variations of the respective query profile updates in an effort to improve the results obtained. This is advantageous where an end user does not have a set technique for discovering some information, and is instead trialling different variables or algorithms. In other words, the `update query` profile definitions are experimental. AREA automatically applies multiple versions of an update using different values for those variables which mutable. For example, the scientist may wish to replicate the value for *energy expenditure* as output from a specialised sensor, by using different sensors

(accelerometers) and inputing some scaling factor as part of an `update` query to discover this second energy expenditure value. As this is not well-defined, and may differ from deployment to deployment or with different athletes, applying multiple *similar* versions for detecting this value can result in better efficiency for the user.

6.2 Baseline Analysis of Activity Data

As a first step to automated analysis, a method for analysing individual datasets to determine their basic properties is provided. In this context, an *individual* dataset is a set of all entries for a single context element - either system or user-defined. The analysis includes numerical operations, and operations to measure or count text content throughout a dataset. In this section AREA's approach to analysing the source of the information (the queries which led to their creation) is desribed, as is construct generated as a result: the `dataset` profile. This information will provide the system with additional knowledge about a dataset, such as unique counts and statistical information (median, mean etc.), to continue analysing for complex features (by using the subsequent cluster and metadata analysis modules).

6.2.1 Analysing Sets of Sensor and Transformation Data

Setting the target for analysis is simply a case of mapping to one of two types of data set: one consisting of all values for a sensed field (e.g. all heart rate values for some subject) or one from values created by some *transformation* (equation). The content of the target node is specified by an XPath expression. It is at this stage that a set of operations are performed based on the type of data (numeric, text), provided by the sensor profile of the device recording the data, or inferred from the input of parameters to discover the properties of that dataset. The results achieved are auto-

matically generated by AREA in an instance of a new metadata construct: the `dataset` profile. To describe this process, an example: the heart rate of a cyclist recorded during a training exercise over a mountainous terrain is introduced. Example 6.1 shows a sample set of heart rate values obtained by an XPath expression for heart rate. To demonstrate the result of applying this process, this sample represents a subset from a larger dataset.

Example 6.1 *Subset of Heart Rate Sensor Data*

```
<HR>99</HR>
<HR>100</HR>
<HR>104</HR>
<HR>109</HR>
<HR>114</HR>
<HR>120</HR>
<HR>122</HR>
<HR>126</HR>
<HR>128</HR>
<HR>131</HR>
<HR>134</HR>
<HR>134</HR>
<HR>136</HR>
<HR>140</HR>
<HR>141</HR>
<HR>142</HR>
<HR>146</HR>
<HR>151</HR>
<HR>153</HR>
<HR>153</HR>
<HR>152</HR>
<HR>151</HR>
<HR>150</HR>
```

6.2.2 Evaluating the Set of Data Obtained

Once the set of data to be analysed has been identified, the generic functions are executed to summarise the properties for this data point in the deployment. In example 6.1, the heart rate monitor (Garmin 405) records data as numerical integers. Thus, AREA executes numerical operations on

the data to discover key properties such as median, mean and standard deviation. The operations performed and their definitions are shown in table 6.1, along with the results for each when applied to the data in example 6.1. The `count` and `countunique` operators will be executed in cases of text-based baseline analysis. In cases where the data being analysed is non-numerical, such as text descriptions, only valid operations are performed: `count, countunique, updateConstruct` and `conditionConstruct` operations.

Operation	**Description**	**Result**
Maximum	The largest number (numerical)	153
Minimum	The smallest number (numerical)	99
Mean	The average of a set of numbers (numerical)	132
Standard Deviation	Variation from the mean (numerical)	17.65
Median	The middle point of a set of numbers (numerical)	134
Count	The amount of values in a set	22
Countunique	The amount of unique values expected in a set	N/A
UpdateConstruct	Parameters involved in (multiple) update transformations	N/A
ConditionConstruct	Parameters involved in (multiple) conditions	N/A

Table 6.1: Baseline Analysis Operations

A description of each of the operations performed is provided below:

- Maximum: This will calculate the largest value of a numerical set of data.
- Minimum: This will calculate the smallest value of a numerical set of data.
- Mean: This will calculate the average value of a numerical set of data.
- Standard Deviation: This calculates a value representing the variation of the data from the average.
- Median: This calculates the numerical value separating the higher half of a set of data from the lower half.
- Count: This counts the amount of entries in a set of data.
- Countunique: This counts the amount of potential content updates that were applied on this set of data.
- UpdateConstruct: This operation analyses the updates used to generate the current set of data and categorises the parameters involved.
- ConditionConstruct: This operation analyses the conditions used prior to the generation

of the current set of data and categorises the parameters involved

The results describe the content of the data for that dataset. This metadata includes information such as `countunique` which will feed into automated algorithms, as will be described later in the chapter. The process of analysing the results is necessary to allow the user to understand the variability and decide which parameters to use in future queries, or to be used or altered in new or existing query profiles. This process feeds into both clustering algorithms and into incrementally improving previous update and extension queries as discussed in section 6.4.

To facilitate this, AREA collates the results of analysis in the `dataset profile`. Definition 6.1 shows the structure for an dataset profile. Simple elements have been omitted here but the full version can be found in Appendix B. These serve as evidence both for tailoring algorithms to suit a particular deployment and as input to clustering algorithms. As with profiles, the structure of a dataset profile is defined using XML schema. Many of the individual elements are *optional* as they apply only to numerical data. For example, the median element will not be present for a dataset profile of a text-based context element data. The `updateConstruct` and `conditionConstruct` elements are only available when update or extension query profile(s) have created the context element being analysed. They contain the results of analysing the input parameters (condition and update) to these profiles. Both the condition and update elements have their input parameters categorised into one of three classifications: variables, fields and complex. It is explained later in this chapter how this classification feeds into incremental updates and clustering algorithms.

Definition 6.1 *Dataset Profile Schema (abridged)*

```
1 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
2   <xs:element name="updateConstruct"><xs:complexType><xs:sequence>
3           <xs:element ref="variables"/>
```

```
4           <xs:element ref="fields"/>
5           <xs:element ref="complex"/>
6   </xs:sequence></xs:complexType></xs:element>
7   <xs:element name="property"><xs:complexType><xs:sequence>
8           <xs:element ref="name"/>
9           <xs:element ref="value"/>
10  </xs:sequence></xs:complexType></xs:element>
11  <xs:element name="properties"><xs:complexType><xs:sequence>
12          <xs:element ref="property" maxOccurs="unbounded"/>
13  </xs:sequence></xs:complexType></xs:element>
14  <xs:element name="variables"><xs:complexType><xs:sequence>
15          <xs:element ref="name"/>
16          <xs:element ref="value"/>
17          <xs:element ref="fixed"/>
18  </xs:sequence></xs:complexType></xs:element>
19  <xs:element name="conditionConstruct"><xs:complexType><xs:sequence>
20          <xs:element ref="variables"/>
21          <xs:element ref="fields"/>
22          <xs:element ref="complex"/>
23  </xs:sequence></xs:complexType></xs:element>
24  <xs:element name="complex"><xs:complexType><xs:sequence>
25          <xs:element ref="name"/>
26  </xs:sequence></xs:complexType></xs:element>
27  <xs:element name="datasetprofile"><xs:complexType><xs:sequence>
28          <xs:element ref="id"/>
29          <xs:element ref="name"/>
30          <xs:element ref="deployment"/>
31          <xs:element ref="properties"/>
32          <xs:element ref="updateConstruct"  minOccurs="0"/>
33          <xs:element ref="conditionConstruct"  minOccurs="0"/>
34  </xs:sequence></xs:complexType></xs:element>
35</xs:schema>
```

The key elements of the `dataset` profile are described below:

- id: An identification value for an instance of a dataset profile. (line 28)
- name: A name for an instance of a dataset profile (line 29).
- deployment: The deployment in which this dataset was recorded (line 30).
- properties: One or more properties about the data collected during this dataset (line 31, 7-10). e.g.: Each of the operations in Table 6.1. Sub-elements:
  - name: Specifies the name of the property. e.g.: Median
  - value: Specifies a result for this property. For example: 10.5.

- updateConstruct: Each part of the update construct of the update(s) which created these data points are listed and categorized (line 32, 2-6) Sub-elements:
  - variables: Numerical values, either constant (fixed) or changeable. e.g.: 3.5
  - fields: Raw data fields. For example: Heart Rate.
  - complex: A data point previously defined by a query profile. For example: Intensity.
- conditionConstruct: Each part of the conditions of the updates(s) which created these data points are listed and categorized (line 33, 19-23). Sub-elements: (same as updateConstruct)

The `updateConstruct` and `conditionConstruct` elements further categorise the parameters of the updates and conditions respectively by naming each field and complex element, and assigning a value to the variables. The variables are also explicitly marked as mutable or immutable. A sample dataset instance for the heart rate example (Table 6.1) is shown in Example 6.2. Each named property and value are on lines 7, 9, 11, and 13. This is automatically generated by AREA.

Example 6.2 *Dataset Profile for Heart Rate Deployment*

```
1<datasetprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
2chemaLocation="datasetprofile.xsd">
3   <id>1</id>
4   <name>HeartRate</name>
5   <deployment>cyclingMar31</deployment>
6   <properties>
7       <property><name>mean</name><value>132</value>
8       </property>
9       <property><name>maximum</name><value>153</value>
10      </property>
11      <property><name>minimum</name><value>99</value>
12      </property>
13      <property><name>standarddev</name><value>17.65</value>
14      </property>
15      <property><name>median</name><value>134</value>
16      </property>
17      <property><name>count</name><value>22</value>
18      </property>
19  </properties>
20</datasetprofile>
```

Due to the dynamic nature of scenario datasets, this information allows for the analysis of both the data itself and the metadata which created this data (if applicable). The `dataset profile` generated is treated as yet more metadata in a similar manner to dataguides [20] because the `dataset` can be accessed and queried by other AREA processes. It will be demonstrated later how this metadata is used by the other 2 modules (cluster and metadata analysis) of the AAA processor. The analysis of data aids with classification and may lead to improved enrichment of data using query profiles. For instance, an intensity metric tailored for a certain deployment may be defined based on *relative intensity* for that situation. This might be a method of discovering when the most effort was applied during a deployment. Analysis of the range of heart rate, respiration rate and other metrics will provide us with the range of values recorded, and by using these a profile for intensity can be derived. This might be an improvement on previously user-defined query profiles for classifying intensity. This provides flexibility to AREA while maintaining simplicity. The next section explains how data analysis technques are further utilised to detect potentially interesting patterns in the sensor data and how metadata guides the configuration of these techniques.

6.3 Cluster Analysis

AREA employs clustering as a means of detecting patterns in sensor data. This can benefit the user in that it provides an automated means of discovering groups of similar data within a data set. This may be used to improve current approaches, or to experiment with different data sources to identify interesting information such as a group of athletes with similar `work energy` relative to `age` values. A clustering process groups together objects which are similar to each other and dissimilar to objects in other clusters [31]. In this section, the clustering process is described to demonstrate how it can be used with `dataset` profile instances to discover information such as the

discovery of different *intensity* levels of a group of athletes.

K-means is an exclusive clustering algorithm [70], meaning each point is associated with only one cluster. K-means clustering was chosen as it allows the configuration of k. Metadata can thus be used as input to populate the algorithms prior to execution. In addition, k can be set by the end user when required. *'k'* represents the number of clusters and in AREA, its initial value is derived from the `dataset profile` in cases where the algorithm is to replace or improve a classification. The `conditionConstruct` and `countunique` elements from the dataset profile contains the information for setting up the clusting process. Clustering can have multiple dimensions, with different distance algorithms used depending on the amount of dimensions being analysed. If more than 2 dimensions are involved in the algorithm, multi-dimensional points are plotted (but not graphically represented) and the distance formula is altered. To demonstrate the operation and effectiveness of clustering, specifically k-means clustering, an example from cycling i presented. For simplicity two dimensions are considered, recorded by the same device - heart rate and respiration rate. This allows the graphing of data on two axes.

Example 6.3 *Dataset Profile Instance Resulting from Data Analysis of the Intensity Context Element*

```
<datasetprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
SchemaLocation="datasetprofile.xsd">
  <id>7</id>
  <name>intensity</name>
  <deployment>wickMts</deployment>
  <properties>
      <property>
          <name>count</name>
          <value>730</value>
      </property>
      <property>
          <name>countunique</name>
          <value>3</value>
```

```
14        </property>
15    </properties>
16    <conditionConstruct>
17        <fields>
18            <name>heartrate</name>
19            <name>respirationrate</name>
20        </fields>
21    </conditionConstruct>
22</datasetprofile>
```

This dataset profile (example 6.3) is automatically generated as a result of baseline analysis of the `intensity` element. The content of the `intensity` elements has been created previously by 3 update query profiles (using the process described in chapter 5), each classifying intensity as *"low", "moderate"* or *"high"*. The `value` of *countunique*, generated during `baseline analysis`, is *3* (line 13) as there are 3 potential values across all instances of `intensity`: (*low, moderate, high*). The content of the `conditionConstruct` shows two fields named *heartrate* and *respirationrate* (line 18-19). As explained earlier, the `conditionConstruct` lists the elements involved in prior update queries to identify where update transformations are to occur. Thus, the heart and respiration values from a sensor were used in the `condition` elements from each of the 3 user `query` profiles defining the intensity classification. In some cases, such as when a younger athlete is being monitored, or a less strenuous activity is being performed, the distribution of intensity may be different. To resolve this, the scientist can make use of AREA's automated cluster analysis to improve the intensity classification for that dataset. Subsequently, for a new batch of perhaps under age children playing sport, the system can suggest new centroids based on those obtained from a similar set of data. Alternatively, AREA can prompt the user with cluster analysis in cases where the `countunique` value does not match the actual number of unique values within a set of data. This could occur if (for instance) boundaries are set too low when trying to detect *low* intensity.

The aim is to group these values into a set of clusters. These values correspond to the real-world physiological characteristics of the subjects at the time of data collection. In this case, k can be set to 3 (representing the *countunique* `value`) for the purpose of discovering three clusters within the data set. The `conditionConstruct` specifies that the heart and respiration rates are used to evaluate intensity. Thus, these data points are plotted on a graph and the k-means algorithm will assign each point to a cluster. Initially, 3 (k) points *(heartrate, respirationrate)* are randomly selected as initial centre points (centroids). Table 6.2 shows a list of heart and respiration rates over a time period and their distance from the 3 centroids. A centroid is a point (i.e. a heart and respiration rate pair) that is arbitrarily selected by AREA. Each combination is assigned an initial cluster, corresponding to the closest centroid. The first two columns represent the data analysis provided *heartrate* and *respirationrate*. The next three ($k = 3$) columns list the `C1 (146,28), C2 (141,27), C3 (122,26)` labels representing the three initial centroids, and their (heart rate, respiration rate) values. The final column lists which cluster the HeartRate, Respiration rate is initially closest to.

The calculation of clusters on this data involves identifying some initial centre point for the cluster and calculating the distance of every other point from this and other centroids. The k-means algorithm re-calculates the closest centroid multiple times until they are stable, i.e. the centroid does not change. The clustering process results in detecting similar sets of data, which in this example may provide a more accurate estimation of intensity. AREA's implementation of the k-means clustering algorithm is shown in Algorithm 1.

HeartRate	RespirationRate	C1 (146,28)	C2 (141,27)	C3 (122,26)	Cluster
109	24	37.2	32.1	13.2	C3
114	24	32.2	27.2	8.2	C3
120	24	26.3	21.2	2.8	C3
122	26	24.1	19	0	C3
126	25	20.2	15.1	4.1	C3
128	27	18	13	6.1	C3
131	27	15	10	9.1	C3
134	28	12	7.1	12.2	C2
134	27	12	7	12	C2
138	28	8	3.2	16.1	C2
140	28	6	1.4	18.1	C2
141	27	5.1	0	19	C2
142	26	4.5	1.4	20	C1
146	26	0	5.1	24.1	C1

Table 6.2: Two sets of data and their distance from 3 initial centroids

Algorithm 1 AREA Implementation of k-means Clustering

1. The Dataset profile for the concept (context element) to be analysed is generated automatically. From the Dataset profile, the *countunique* value is assigned to k and the `conditionConstruct` fields are plotted for analysis.
2. k points are arbitrarily (randomly) selected as initial centroids for the clusters.
3. Each other point is assigned to its nearest centroid's cluster.
4. The centroid for each cluster is re-calculated.
5. Steps 4 and 5 are repeated until the centroids are stable.

High rates for both heart and respiration may imply a high level of intensity, while low values for each may be low intensity. Figure 6.1 shows the points from example 6.2, three of which are selected as initial centroids randomly by AREA (`C1 (146,28), C2 (141,27), C3 (122,26)`). The distance between two points [71] ($d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) is calculated for each value to each centroid, with the value assigned to the closest centroid. This

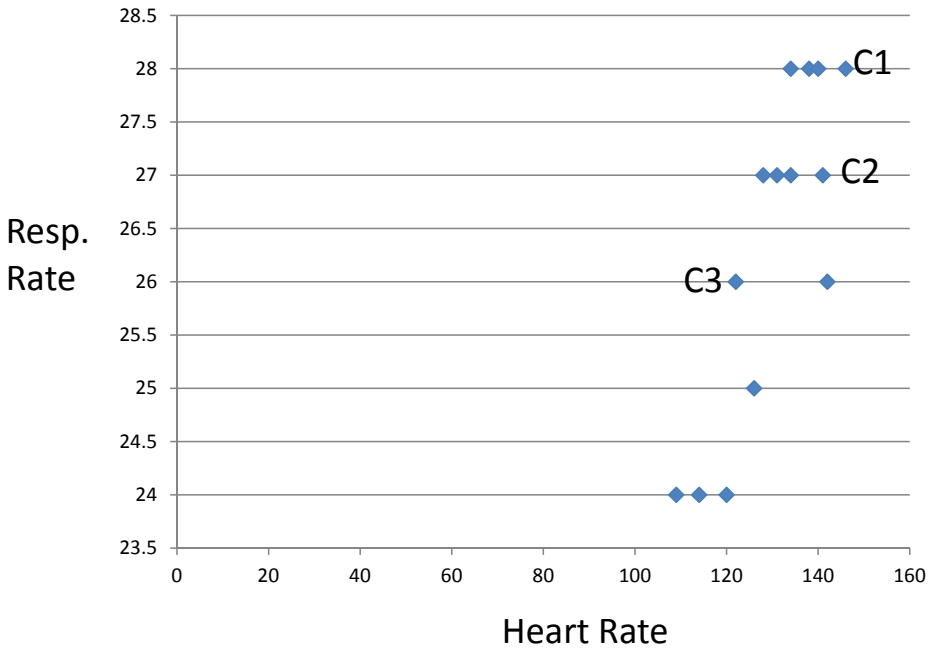

Figure 6.1: Initial mapping of heart and respiration rates

means each point will be assigned to its most relevant (closest) cluster. Take for instance the first point from table 6.2 (heartrate = 109, respirationrate = 24). As can be seen in Table 6.2, this point is a distance of *37.2* from C1, *32.1* from C2, and *13.2* from C3. Thus, it is assigned to C3, the closest centroid. As the process continues, and the clusters evolve with multiple iterations, the end result is a new machine generated classification of the `intensity` metric. These results can then be compared with those achieved using the original approach using update queries. Subsequently, the results may be used to alter these queries, or simply used in future deployments where the activity or subject involved are similar to those in this example. This approach allows the identification of new clusters of potentially interesting information within the data. It provides measures of data relative to that specific instance of deployment to provide new possibilities. This could be used to alter the *low / moderate / high* boundaries of the intensity algorithm in cases where the data set being analysed is in a sport like golf rather

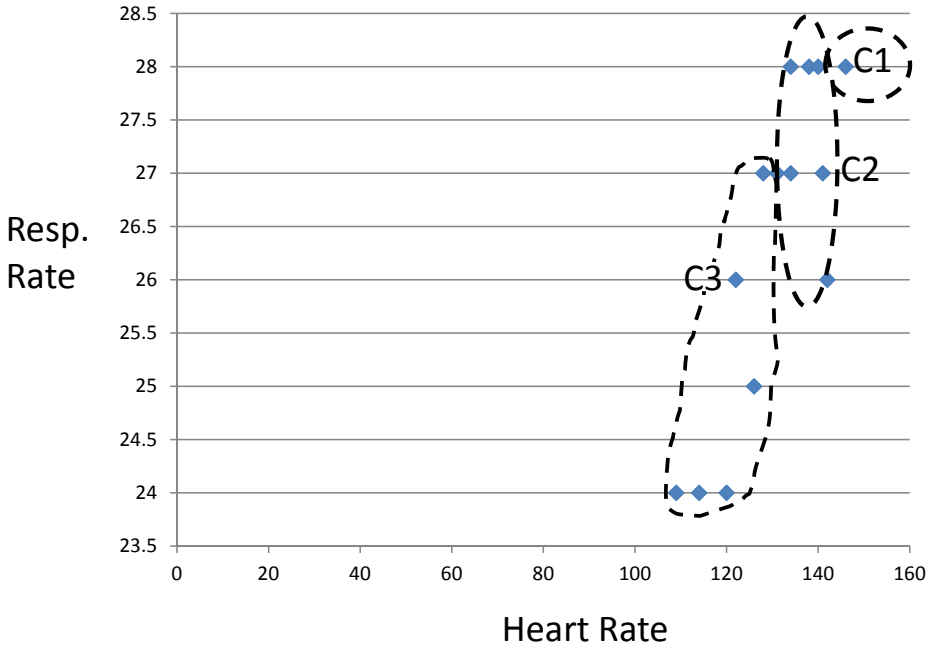

Figure 6.2: Mapping the first three clusters

than the sport of cycling for which the original technique was defined (and which has a higher level of general intensity). Likewise, different subjects ages affects the result of heart and respiration rates. These differences are not always uniform across all subjects with similar ages so cluster analysis can help define the boundaries for individuals.

In the experiments (chapter 7) a real-world example of how cluster analysis has been utilised is presented. IT shows an example where the knowledge worker has defined an inaccurate method of classifying the *intensity of effort* of a cyclist during an event. The update profiles defined in an effort to detect *peak* intensity has led to no instances in the data labelled as *peak*. It is also demonstrated how AREA's cluster analysis utilised the repository information to build a new classification, and how displaying this information through a portal allowed the user to modify their original update queries.

6.4 Metadata Analysis

The query profile may include an update instruction and in some cases, this update is an equation making use of sensor data, context data, mathematical constants or arbitrary variables. These different values used to generate data are documented and categorised by a `dataset` profile following baseline analysis. The goal of metadata analysis is to provide another means of detecting information for the user by focussing on some of the structural (metadata) aspects of the AREA repository.

The metadata analysis module analyses the transformations described in the `<update>` elements of the user defined `query` profiles. The goal is to experiment with different values for variable parameters of a transform equation. This makes use of the `updateConstruct` metadata of the dataset profile to calculate multiple versions of the equation, each with altered values for the variable parameter. Subtle changes may lead to better results and where this is the case, the end user can maintain the altered value for the parameter for future update operations. Take for instance the detection of a *serve* in a smart space environment. The algorithms in this domain (as described in section 3.2) had to use position and time constraints to build up more complex query capabilities. As part of this process, a zones used to serve must be defined. As the position from which a player serves is not fixed, this position can vary between players. Thus, it is necessary to try different coordinates of the court to refine the most appropriate location for a serving zone. As this is a manually intensive process, a metadata analysis process was implemented to suggest provisional boundaries.

6.4.1 Interpreting the Update Construct

The `updateConstruct` is generated (as part of the `dataset Profile`) when baseline analysis is carried out on a context element. As explained in Section 6.2, baseline analysis categorises the updates used to generate a dataset into

one of three categories: *fields, complex* and *variables*. For *variables*, metadata analysis generates similar values to those within the update transform, generates different output to present to the user. The remainder of this section will explain how AREA's approach uses the metadata from prior system-based analysis, apply and re-evaluate algorithms to improve their overall techniques.

6.4.2 Adapting Values of Variables

The metadata analysis of the *fields* category of update parameters is now described. There are two types of variables within equations, those which cannot be altered while keeping their meaning (such as π), and those which form part of a user-defined algorithm (such as a scaling value of '0.000019'). The AREA `metadata analysis` generates n (=2 in the example presented) additional values for each "scaling" variable within the `updateConstruct` of the `dataset` profile.

Example 6.4 *Calculating work energy metric using GT3X accelerometer*

```
x * y * z * 0.000019 * WEIGHT
```

Example 6.5 *UpdateConstruct for Example 6.4*

```
<updateProfile>
    <fields>
        <name>x</name>
        <name>y</name>
        <name>z</name>
        <name>WEIGHT</name>
    </fields>
    <var>
        <variable>
            <name><name/>
            <value>0.000019</value>
            <fixed>false</fixed>
        <variable>
     </var>
 </updateProfile>
```

In example 6.5, the `updateConstruct` for the equations in Example 6.4 is shown. There are 4 *fields*: one subject profile field (weight) (line 6); three sensor readings (x,y,z) (line 3-5); and one *variable*: '0.000019'(line 11). If two additional variable *values* are to be applied, AREA generates these 1% above and below the current value. In this case the new variables are `0.00001881` and `0.00001919`. Each of the adjusted algorithms are run again with new parameter values. Thus, 3 transformations are applied simultaneously. The result is three data set results, such as those shown in table 6.3, where simpler figures have been used to illustrate the process. The first (original) uses the initial value (2), the other two use the newly generated values. If the end user changes to one of the newer variables, a query profile for that instance can be created. There is also an option to change the (generic) initial update query which lead to the initial results in order to use the best update identified on other scenarios. The data from the new, improved value for the variable is propagated through the data set for which it is valid.

HeartRate	**RespirationRate**	**2**	**2.02**	**1.98**
91	24	7.58	7.65	7.5
93	24	7.75	7.82	7.67
95	24	7.91	8	7.83
95	25	7.6	7.67	7.52
97	26	7.46	7.53	7.38
100	26	7.69	7.76	7.61

Table 6.3: The results of three slightly altered algorithms

The metadata generated in previous stages of enrichment has lead to the identification of these elements that are important for metdata analysis (i.e. mutable), and these suggestions allow incremental updating of query profiles at a specialised level or at a generalised higher level. Chapter 7 details the efforts undertaken to detect a serve zone using this approach. In brief, it was possible to adapt the coordinates representing the serve zone after multiple provisional zones were presented to the user via the portal. As a result, the

condition on which the initial update query was based was updated by the user to reflect the best zone discovered using metadata analysis.

6.5 Summary

This chapter has described AREAs approach to automated activity analysis. The chapter began by describing the generic baseline analysis of sensor data and classification of the data generated by query profiles. Following this, it detailed how the data gathered during this process can drive an automated analysis of multiple data points to discover unknown patterns or improve previous classifications using a clustering approach. It then described how AREA can automatically guide new techniques for discovering data by altering individual parts of update queries. The next chapter revisits the evaluation sites from chapter 3 to demonstrate how the AREA framework can be used to meet the information needs of the users, including an evaluation of AAA.

Chapter 7

Experimental Analysis

Having developed an infrastructure for managing different sensors in different deployments, the use cases from chapter 3 are now revisted. This means satisfying the real world requirements of the specialists, and in particular meeting the sport scientists requirements regarding event detection, event definition and integration of data sources. To evaluate the system, queries from each domain are implemented using AREA to determine if the necessary functional requirements have been met. The following questions are relevant to the case studies described in this chapter.

- Is the process for providing and enriching context successful in transforming sensor data to enable queries?
- Can the Automated Activity Analysis processor aid the user in identifying information not known in advance?

7.1 Strategy for Query Enablement

As discussed in Chapter 4, there are three *categories* of profiles: enablement, user, and system defined profiles. The enablement profiles are the sensor, subject and deployment profiles and are defined for each deployment envi-

ronment by a knowledge worker familiar with this data. User-defined query profiles are *also* defined by a knowledge worker who understands the meaning behind different sensor and context information. They can be applied to query, update or extend the scenario dataset and schema with additional context. This section demonstrates the application of these queries across a number of domains with an aim to illustrate the strategy for enabling subsequent scientists to query for their information needs, without extensive knowledge of the underlying data.

To evaluate the functionality and flexibility of the AREA framework some of the queries presented in Chapter 3 are revisted. It is demonstrated how a knowledge worker can apply query profiles where necessary to define context. This context can later be queried using simple XQuery expressions or the user interface provided by AREA, allowing scientists to access potentially complex knowledge using basic commands. This chapter begins by stepping through the process of defining and applying update and extension queries in a number of different environments. It will be shown that the strategy to achieve these query requirements is the same in each case, utilising the definition of query profiles and applying them on a dataset. This process is carried out by a knowledge worker, a scientist who understands the data and applies operations on this data to achieve their information needs. The query profiles can be defined by this user to iteratively update and improve the dataset to be later queried by a second type of user, a scientist less familiar with the raw data source.

7.1.1 Horse-Racing

Calculating the Total Energy Expenditure of a Jockey

To begin, *Query 3* from Table 3.5 is examined: *Calculate the total energy expenditure for Jockey CG during training session S15CG (Simulator).* This query will be expressed as shown in Example 7.1 following the application

of relevant query profiles by the knowledge worker. This query cannot be expressed without first defining the `totalenergy` context element and then, populating it with the relevant information.

Example 7.1 *Query 3 expressed in XQuery*

```
for $a in doc("S15CG.xml")
//subject[name="CG"]/sensorProfile[location="LHWrist"]/totalenergy
return $a
```

The subject profile instance for jockey *CG* is shown in Example 7.2, the deployment used for evaluation is described by the profile in Example 7.3 with the sensors individually described by their profiles in Appendix C. Following sensor enablement and context integration, the scenario dataset is ready for basic queries. At this stage the required context element `totalenergy` does not exist. The knowledge worker has a technique for discovering the total energy expenditure value, using sensor data with additional context. This technique uses a combination of information from previously validated algorithms (in the sport science domain) and sensor data. The steps required to discover this information is now desribed, illustrating how each of these steps take place. Following this process, the new context element is created and populated, ready to be queried using the expression in Example 7.1. Each step uses the same strategy - A new update query profile is defined and then applied on the dataset. An extension query profile may need to be applied on the AREA schema prior to the update query.

Example 7.2 *Subject Profile for Jockey CG*

```
<subject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/subjectProfile.xsd">
    <id>15</id>
    <name>CG</name>
    <activity>horse-racing</activity>
    <properties>
        <property>
            <name>gender</name><scale></scale><value>male</value>
        </property>
        <property>
            <name>age</name><scale>years</scale><value>16</value>
        </property>
        <property>
            <name>height</name><scale>cm</scale><value>165</value>
        </property>
        <property>
            <name>weight</name><scale>kg</scale><value>59</value>
        </property>
    </properties>
</subject>
```

Example 7.3 *Deployment Profile for Simulator Deployment*

```
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/deploymentProfile.xsd">
    <id>1</id>
    <name>SIM</name>
    <location>indoors</location>
    <startdatetime>2011-03-31T19:32:00</startdatetime>
    <enddatetime>2011-03-31T19:52:20</enddatetime>
    <properties>
        <property>
            <name>session</name><scale/><value>training</value>
        </property>
        <property>
            <name>simulatorID</name><scale/><value>1</value>
        </property>
    </properties>
</deployment>
```

While these XML profiles may look relatively complex for the non-IT scientist, a simple user interface can be used to create them, as will be shown in section 8.1. To facilitate the query in Example 7.1, the following steps are performed by the knowledge worker. These steps will provide the dataset with all the parameters necessary to discover the required information - total energy expenditure.

1. Define a query profile to insert the resting metabolic rate (RMR) for this subject in this deployment.

2. Define a query profile to insert the thermic effect of food (TEF) for this subject in this deployment.

3. Define a query profile to insert a scaling factor (`scalingfactor`) for the subsequent steps.

4. Define a query profile to create and populate the *work energy* for each Entry.

5. Define a query profile to sum all work energy values and add this value to RMR and TEF to result in a single value for `totalenergy`.

Following the application of the 5 updates, the query in Example 7.1 can be expressed to return the valid result.

Example 7.4 *Update Query: Defining the resting metabolic rate (RMR) value (Step 1)*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
    <name>RMRvalue</name>
    <doc>S15CG.xml</doc>
    <target>//sensorProfile[location="LHwrist"]/RMR</target>
    <condition></condition>
    <update>
        <address>//sensorProfile[location="LHwrist"]/RMR</address>
        <content>34</content>
```

```
    </update>
</updateQuery>
```

Example 7.5 *Update Query: Defining the thermic effect of food value (Step 2)*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
    <name>TEFvalue</name>
    <doc>S15CG.xml</doc>
    <target>//sensorProfile[location="LHwrist"]/TEF</target>
    <condition></condition>
    <update>
        <address>//sensorProfile[location="LHwrist"]/TEF</address>
        <content>34</content>
    </update>
</updateQuery>
```

Example 7.6 *Update Query: Defining a scaling factor (Step 3)*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
Location="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
    <name>scalingvalue</name>
    <doc>S15CG.xml</doc>
    <target>//sensorProfile[location="LHwrist"]/scalingfactor</target>
    <condition></condition>
    <update>
        <address>//sensorProfile[location="LHwrist"]/scalingfactor</address>
        <content>0.000019</content>
    </update>
</updateQuery>
```

Each of the updates in example 7.4, 7.5 and 7.6 can then be applied on a dataset, providing relevant extension profiles have been applied on the AREA schema (such as example 7.7 for the RMR element in example 7.4).

Example 7.7 *Extension Profile to extend the schema with an RMR node*

```
<extensionQuery>
    <name>RMR-gt3x</name>
    <doc>AREA</doc>
```

```
    <target>//sensorProfile[name=gt3x]/Entry</target>
    <update>
         <address>//sensorProfile[name=gt3x]/Entry/RMR</address>
    </update>
</extensionQuery>
```

The information in these examples 7.4, 7.5 and 7.6 has been gathered by the knowledge worker and input using these query profiles as they are not input during initialisation (as part of a sensor, subject or deployment profile). In some cases it may be suitable to input values such as these as part of a deployment profile - where a value applies to all subjects or sensors involved in the deployment. Example 7.8 shows the technique for calculating the work energy value for each entry of a gt3x accelerometer, using the axes of the accelerometer, the weight of the jockey and the `scalingfactor` from example 7.6 (line 10-13). The result is stored in a new element called `workenergy` (line 8), within each existing `Entry` elements of that sensor. This is a more generic query profile using data previously enriched by other profiles to identify the requirements.

Example 7.8 *Update Query: Calculating Work Energy Using GT3X in Horse-Racing (Step 4)*

```
1<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchema
2Location="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3    <name>workEnergyCalc</name>
4    <doc>S15CG.xml</doc>
5    <target>//sensorProfile[location="LHwrist"]/fields/Entry</target>
6    <condition></condition>
7    <update>
8        <address>//sensorProfile[location="LHwrist"]/fields/Entry/workenergy
9        </address>
10       <content>//sensorProfile[location="LHwrist"]/fields/Entry/x *
11       //sensorProfile[location="LHwrist"]/fields/Entry/y *
12       //sensorProfile[location="LHwrist"]/fields/Entry/z *
13       scalingfactor * subjectProfile//property[name=weight]/value</content>
14   </update>
15</updateQuery>
```

The final step to allowing the XQuery to be expressed is to calculate the total energy expenditure of the subject during that deployment. Example 7.9 shows the technique for discovering this value, which adds the data from step 1 and 2 (TEF and RMR) to the total sum of work energy values discovered in step 4 (line 9-10). The context element `totalenergy` is created as a child node of the `sensorProfile` element (line 8), and is populated upon application of this update. Following the application of these five steps in sequence, the query in Example 7.1 can be expressed to return the result for this scenario. The results for this query are shown later in Table 8.4.

Example 7.9 *Update Query: Calculating Total Energy Expenditure (Step 5)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
2 SchemaLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>totalEnergyExp</name>
4     <doc>S15CG.xml</doc>
5     <target>//sensorProfile[location="LHwrist"]/fields/Entry/workenergy</target>
6     <condition></condition>
7     <update>
8         <address>//sensorProfile[location="LHwrist"]/totalenergy</address>
9         <content>SUM(//sensorProfile[location="LHwrist"]/fields/Entry/workenergy)
10        + RMR + TEF</content>
11    </update>
12 </updateQuery>
```

Identifying instances where a Jockey uses a whip

Query 6: Identify all instances of a jockey whipping the horse or simulator. To demonstrate how this query is enabled, data gathered from jockey CG is again used. The XQuery to express this query is shown in example 7.10. As with the query in example 7.1, the key element required to run this query (`whip`) is missing from the initial dataset. The knowledge worker is looking to discover when a whip occurs using the sensors available in the horse-racing deployment. Scientists discovered that the accelerometer located on

the wrist of the jockey hits its maximum value (`MAXVALUE`) on all three axes each time a whip took place. [11] As a result the knowledge worker defined the update query shown in example 7.11 to discover a whip on the left-hand side, and a corresponding *righthandwhip* (Appendix D).

Example 7.10 *Query 6 expressed in XQuery*

```
for $a in doc("S15CG.xml")//subject[name="CG"]/sensorProfile[location="wrist"]
where $a/whip="leftwhip" or $a/whip="rightwhip"
return $a/time
```

Example 7.11 *Update Query to Identify a Left-Handed Whip*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
<updateQuery>
    <name>lefthandedwhip</name>
    <doc>S15CG.xml</doc>
    <target>//sensorProfile[location="LHwrist"]/fields/Entry</target>
    <condition>//sensorProfile[location="LHwrist"]/fields/Entry/x > MAXVALUE AND
     //sensorProfile[location="LHwrist"]/fields/Entry/y > MAXVALUE AND
      //sensorProfile[location="LHwrist"]/fields/Entry/z > MAXVALUE</condition>
    <update>
        <address>//sensorProfile[location="LHwrist"]/fields/Entry/whip</address>
        <content>"leftwhip"</content>
    </update>
</updateQuery>
```

After the new context element `<whip>` ((example 7.11, *righthandwhip*) was added to the AREA schema and the update query populated this element, scientists analysed the data and discovered a number of *false-positives* were being detected early in the session. As a jockey can only *whip* the horse when close to top speed, the initial update was altered to include a condition that both horse and simulator must be fast-cantering or galloping at the time of the whip for it to be valid. To do so, a period of *fast-cantering* was identified by the knowledge worker, and applied on the data using the update query in example 7.12. This uses a time constraint to define when the simulator

was in a *fast-canter* gait (lines 6-7, 9-10). No galloping was performed in this deployment as it is not permitted for trainee jockeys (which included subject CG).

Example 7.12 *Query Profile: Identifying where the horse is fast-cantering*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>fastcantergait</name>
4     <doc>S15CG.xml</doc>
5     <target>//sensorProfile[location="LHwrist"]/fields/Entry</target>
6     <condition>//sensorProfile[location="LHwrist"]/fields/Entry/time >= 720000 AND
7     //sensorProfile[location="LHwrist"]/fields/Entry/time < 960000</condition>
8     <update>
9         <address>//sensorProfile[location="LHwrist"]/fields/Entry/gait</address>
10        <content>"fast-canter"</content>
11    </update>
12 </updateQuery>
```

Now with the dataset extended to include a *fast-cantering* `gait`, for which the AREA schema has also been extended, the knowledge worker can re-define the query profiles for discovering left and right hand whips. The left-handed whip update is shown in Example 7.13. In this improved query profile, the condition includes a constraint that the gait is a *fast-canter* at the time the `MAXVALUE` threshold being reached. Thus, a combination of detecting a *fast-canter* (and galloping in the case of professionals), and the improved whip detection technique will enable the accurate identification of whipping instances in the dataset, which can then be queried using the XQuery expression in Example 7.10.

Example 7.13 *Update Query: Left-Handed Whip Version 2 (Improved)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>lefthandedwhip2</name>
4     <doc>S15CG.xml</doc>
5     <target>//sensorProfile[location="LHwrist"]/fields/Entry</target>
6     <condition>//sensorProfile[location="LHwrist"]/fields/Entry/x > MAXVALUE AND
```

```
7     //sensorProfile[location="LHwrist"]/fields/Entry/y > MAXVALUE AND
8      //sensorProfile[location="LHwrist"]/fields/Entry/z > MAXVALUE AND
9       //sensorProfile[location="saddle"]/fields/Entry/gait = 'fast-canter'
10    </condition>
11    <update>
12        <address>/y/sensorProfile[location="LHwrist"]/fields/Entry/whip</address>
13        <content>"leftwhip"</content>
14    </update>
15</updateQuery>
```

This strategy uses the same approach as the previous query, but the steps taken were decided by the knowledge worker as they got feedback from previous updates. In other words, they were able to adapt their understanding of the data. Error checking the initial *lefthandedwhip* update showed the weakness of the initial approach. This section has shown how AREA can allow on-the-fly changes to the steps to enabling the queries and improving the results obtained.

7.1.2 Search and Rescue

Classifying Quality of Sleep

The next query examined is *Query 3* from Table 3.7: *Does subject N have a good quality of sleep on Day x?*. This will be expressed in XQuery as shown in Example 7.14 following the application of required query profiles by the knowledge worker. This XQuery cannot be expressed without first defining the `sleepQualityClass` context element and populating it with a value representing the quality of sleep for that deployment.

Example 7.14 *Query 3 expressed in XQuery*

```
for $a in doc("007ES1.xml")/deployment/sleepqualityclass
return $a
```

This query is focussed on a single deployment of one SAR worker identified as *'007ES'*. This employee has been monitored using two *gt3x+* accelerometers

and a *Sensewear* armband for two consecutive days. The days alternated between working (on call) while at home and while resting on the base. The rescue worker is described by his subject profile shown in Example 7.15 and the deployment by the profile shown in Example 7.16. The sensor profiles for the gt3x+ and Sensewear sensors deployed are shown in Appendix A.

Example 7.15 *The Subject Profile for Subject 007ES*

```
<subject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/subjectProfile.xsd">
    <id>7</id>
    <name>ES</name>
    <activity>SAR</activity>
    <properties>
        <property>
            <name>BMI</name>
            <scale>bmi</scale>
            <value>22</value>
        </property>
        <property>
            <name>age</name>
            <scale>years</scale>
            <value>35</value>
        </property>
        <property>
            <name>gender</name>
            <scale>text</scale>
            <value>male</value>
        </property>
        <property>
            <name>height</name>
            <scale>cm</scale>
            <value>175</value>
        </property>
        <property>
            <name>weight</name>
            <scale>kg</scale>
            <value>85</value>
        </property>
    </properties>
</subject>
```

Example 7.16 *The Deployment Profile for Deployment SAR7*

```
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/deploymentProfile.xsd">
    <id>1</id>
    <name>SAR7</name>
    <location>dynamic</location>
    <startdatetime>2011-10-25T13:00:00</startdatetime>
    <enddatetime>2011-10-26T13:00:00</enddatetime>
    <properties>
        <property>
            <name>oncall</name><scale/><value>true</value>
        </property>
        <property>
            <name>base</name><scale/><value>dublin</value>
        </property>
    </properties>
</deployment>
```

The scientists approach to classifying sleep quality is the following four step process:

1. Define a query profile to identify when the subject is sleeping.
2. Define a query profile to discover when the subject is prone - i.e. Not moving.
3. Define a query profile to calculate the sleep quality metric, defined as the proportion of sleeping time spent prone.
4. Define a query profile to classify the sleep quality metric, specifically identifying *good* sleep quality.

Each of these four steps are defined by the query profiles shown in examples 7.17, 7.18, 7.19 and 7.20.

Example 7.17 *Updating the Scenario with a Sleeping Element. (Step 1)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>sleepPeriod</name>
4     <doc>007ES1.xml</doc>
5     <target>//sensorProfile[location="waist"]/Entry</target>
6     <condition>//sensorProfile[location="waist"]/Entry/time > 1400000</condition>
7     <update>
8         <address>//sensorProfile[location="waist"]/Entry/sleeping</address>
9         <content>"true"</content>
10    </update>
11 </updateQuery>
```

Before step 1 is applied, the AREA schema is extended with a new element `sleeping` as a child of the `Entry` node. The update profile then creates a new context element (`sleeping` in the scenario dataset for each entry satisfied by the condition, and populates them with '*true*' (line 8-9).

Example 7.18 *Updating the Scenario with a Prone Element. (Step 2)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>proneSleep</name>
4     <doc>007ES1.xml</doc>
5     <target>//sensorProfile[location="waist"]/Entry</target>
6     <condition>//sensorProfile[location="waist"]/Entry/x == 0 AND
7     //sensorProfile[location="waist"]/Entry/y == 0 AND
8     //sensorProfile[location="waist"]/Entry/z  == 0</condition>
9     <update>
10        <address>//sensorProfile[location="waist"]/Entry/prone</address>
11        <content>"true"</content>
12    </update>
13 </updateQuery>
```

Step 2 extends the scenario with a new element `prone` as a child of the `sensorProfile` node.The update then creates a new context element (`prone` in the scenario dataset for each entry satisfied by the condition `(x,y,z = 0)` (line 6-8), and populates them with '*true*' (line 11).

Example 7.19 *Calculating the Sleep Quality Metric (Step 3)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>sleepQualityCalc</name>
4     <doc>007ES1.xml</doc>
5     <target>//sensorProfile[location="waist"]/Entry</target>
6     <condition></condition>
7     <update>
8         <address>/deployment/sleepquality</address>
9         <content>COUNT(//sensorProfile[location="waist"]/Entry/movement == "prone"))
10        / COUNT(//sensorProfile[location="waist"]/Entry/sleeping == "true"))
11        </content>
12    </update>
13 </updateQuery>
```

In step 3, the sleep quality metric is calculated (line 9-10) and stored in a single new context element (`sleepqualitymetric`) in the scenario data. The schema is also updated with this new element prior to application. The value inserted into this element is the proportion of time spent prone while sleeping. The result is a percentage value representing sleep quality.

Example 7.20 *Classifying a Good Quality of Sleep (Step 4)*

```
1 <updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
2 maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
3     <name>goodSleep</name>
4     <doc>007ES1.xml</doc>
5     <target>/deployment/sleepquality</target>
6     <condition>/deployment/sleepquality >= 90</condition>
7     <update>
8         <address>/deployment/sleepqualityclass</address>
9         <content>good</content>
10    </update>
11 </updateQuery>
```

Finally in step 4, the knowledge worker classifies a sleep as *"good"* when the movement is less than 10% of the sleeping period. The condition (line 6) checks the scenario's sleepquality value to determine if this is a good quality

of sleep. The AREA schema is updated with the optional `sleepqualityclass` element using a previous extension query. When each of these four updates are applied on the dataset, the query in Example 7.14 can be expressed to find the required information.

7.1.3 Cycling

Detecting Effort Intensity and Cluster Analysis

To evaluate cluster analysis the concept of `effort Intensity` is reintroduced. This user-defined concept describes the intensity of a cyclist during a race or training session. The scientists wish to analyse periods of peak effort intensity and subsequently analyse the time at which these occur. The knowledge worker will define three update queries to identify poor, moderate and peak effort. Once these changes are applied on the data, the XQuery to discover this information is shown in example 7.21. This corresponds to query 10 from table 3.3.

Example 7.21 *Query 10 expressed in XQuery*

```
for $a in doc("GM01.xml")/deployment/Entry/effortIntensity
return $a
```

A simple, one step process is required to return these results. The query profile for peak effort is shown in example 7.22. This was defined for the data gathered during the *Wicklow mountains* data set. It makes use of the available data values, namely the speed and respiration rate of the cyclist. In the `peakEffort` query, if the heart rate is greater than 180 and the repiration rate greater than 35 (line 5-6), the context element `<effortIntensity>` is updated with the text *"peak"* (line 8-9). Similar profiles exist for poor and moderate effort intensity.

Example 7.22 *Update Query to Identify Peak Effort Intensity*

```
<updateQuery>
    <name>peakEffort</name>
    <doc>wickMts1.xml</doc>
    <target>//Entry</target>
    <condition>//sensorProfile[name=heartrate]/Entry/HR > 180 AND
        //sensorProfile[name=heartrate]/Entry/RR > 35
    <update>
        <address>//Entry/effortIntensity</address>
        <content>peak</content>
    </update>
</updateQuery>
```

Upon applying this update to the scenario (following a metadata update of the schema), no instances of *peak* effort as defined by the user were discovered. However, after analysing the results, and comparing to a ground truth (video), it was observed that some instances were being identified as moderate efforts which were in fact periods of sustained effort. The reason behind this was that the values chosen by the knowledge worker to define peak intensity were not accurate; they were not designed for situations where the cyclists natural heart and respiration values may be different depending on environmental factors such as weather or types of terrain. To discover more appropriate values for this situation, better values must be discovered and applied as boundaries and a new classification required.

Example 7.23 *Dataset Profile Instance for Effort Intensity*

```
<datasetprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
SchemaLocation="datasetprofile.xsd">
  <id>22</id>
  <name>effortIntensity</name>
  <deployment>wickMts</deployment>
  <properties>
      <property>
          <name>count</name>
          <value>3600</value>
      </property>
      <property>
```

```
12            <name>countunique</name>
13            <value>3</value>
14        </property>
15    </properties>
16    <conditionConstruct>
17        <fields>
18            <name>respirationRate</name>
19            <name>heartRate</name>
20        </fields>
21    </conditionConstruct>
22</datasetprofile>
```

Thus, cluster analysis was performed on the dataset, using the approach detailed in chapter 6. This used the *uniquecount* from `baseline analysis` (example 7.23) to populate $k = 3$ (line 13), and the *respiration* and *heart rate* fields are identified from the `conditionConstruct` (line 18,19). The k-means clustering algorithm was then applied using the scenario data, and three new centroids were presented to the user through the portal. This provided an updated classification of the `effort intensity` context. Further analysis of the results from cluster analysis facilitated the user in discovering new boundaries to use as criteria for detecting peak effort intensity.
This new classification is tailored for this data set, but may also be applicable for other subjects with similar characteristics or with similar deployment scenarios. Alternatively, the user can decide to use additional information in future deployments, utilising subject profile information (if available) to further personalise the update query.

7.1.4 Tennis

Detecting a Serve and Metadata Analysis

Evaluation included the metadata analysis module of Automated Activity Analysis (AAA). In this module, AREA automatically analyses the `updateConstruct` derived from a context element during `baseline analysis`.

When defining different *events* in tennis, each event has to be derived from Ubisense data only, which is location based. To facilitate this, it was necessary to partition the court into different zones, on which further updates could be built [52] [12]. One such event identified as a requirement by the user was to identify when a player was serving. Following the identification of serves, the XQuery shown in example 7.24 will return the relevant results for query 1 from table 3.1.

Example 7.24 *Query 1 Expressed in XQuery*

```
for $a in doc("KV01.xml")/deployment/Entry/serve
return $a
```

Initial efforts detected serves based on the serving player being in a *special serving* zone, while the opposing player was in a corresponding *receiving zone* on the opposite side of the court. The approach to classifying serving events is the following 3 step process.

1. Define a query profile to segment the court into zonal boundaries
2. Define a query profile to discover when a player is in a special serve zone
3. Define a query profile for defining which combination of player positions correspond to a serve event taking place

The update query applied on the scenario to define the serving zone (on the left of the court) is shown in example 7.25.

Example 7.25 *Update Query to Identify Upper Left Serve Zone (Step 2)*

```
<updateQuery>
    <name>UpperleftServeZone</name>
    <doc>tennis10Jan.xml</doc>
    <target>//Entry</target>
    <condition>//Entry/x < 4.15 AND
```

```
6         //Entry/x > 3.15 AND
7         //Entry/y < 6.99 AND
8         //Entry/y < 5.49
9     <update>
10        <address>//Entry/serveZone</address>
11        <content>UpperleftServeZone</content>
12    </update>
13</updateQuery>
```

	Serves
Detected	74.7%
Missed	2.2%
Two-detected as 1	18.3%
False-Positives	4.8%

Table 7.1: Detecting a Serve with Initial (v1) Boundaries [12]

When applied in combination with the other query profiles it is possible to discover serves using this technique. Using this approach, 82% of the players serves were initially discovered [52]. It also detected some *false positives*, i.e. instances which were labelled as serves which were not serves. To improve the technique, tweaking of the initial coordinate boundaries for the zones was required. The aim was to improve the detection of serves without introducing too many *false positives*. As this is a manually intensive process, metadata analysis was performed instead, where the variable parameters had their values altered slightly, for a specified number of iterations. Example 7.26 shows the `updateConstruct`

Example 7.26 *UpdateConstruct for* UpperleftServeZone *7.25*

```
1   <updateConstruct>
2       ...
3       <var>
4           <variable>
5               <name>x1<name/>
6               <value>3.15</value>
7               <fixed>false</fixed>
```

```
            <variable>
            <variable>
                <name>x2<name/>
                <value>4.15</value>
                <fixed>false</fixed>
            <variable>
            <variable>
                <name>y1<name/>
                <value>5.49</value>
                <fixed>false</fixed>
            <variable>
            <variable>
                <name>y2<name/>
                <value>6.99</value>
                <fixed>false</fixed>
            <variable>
        </var>
    </updateConstruct>
```

The results for the scenario being analysed provided a new breakdown of serves detected. The user used the portal to examine the results, and picked the best boundaries based on cross-checking with a ground truth.

Example 7.27 *Update Query to Identify Upper Left Serve Zone (version 2)*

```
<updateQuery>
    <name>UpperleftServeZonev2</name>
    <doc>tennis10Jan.xml</doc>
    <target>//Entry</target>
    <condition>//Entry/x < 4 AND
        //Entry/x > 3.18 AND
        //Entry/y < 6.5 AND
        //Entry/y < 5.45
    <update>
        <address>//Entry/serveZone</address>
        <content>UpperleftServeZone</content>
    </update>
</updateQuery>
```

Following metadata analysis, the best boundary for the left serve was altered in the `updateQuery` 7.27, to be applied in subsequent queries prior to applying updates to discover serves. Using this approach it was possible to detect 97% of serves, and the amount of *false positives* was reduced to 9 instances as can be seen in table 7.2 [12]. Table 7.2 also lists the detection of point and game boundaries as the algorithms to detect these follow on from serve detections.

	Serves	Points	Games
Total	72	44	6
Detected	70(97%)	43(98%)	6(100%)
Missed	2(3%)	1(2%)	0
False-Positives	9(12.5%)	0	0

Table 7.2: Detecting a Serve with Modified (v2) Boundaries [12]

More focus was on increasing the number of serve detections rather than reducing the false positives as serves play an important part in detecting subsequent boundaries such as points and games.

7.2 Summary

This chapter evaluated AREA in terms of its functionality and efficiency. It was demonstrated how each of the queries introduced in chapter 3 can be resolved by extending the dataset. This chapter also explained how the user interface provides the necessary function of allowing the user to specify items of interest and command the system to locate them. The benefit of automated analysis in the real world was then deonstrated. AREA's approach to defining query profiles allows their application in multiple deployments and domains and the use of profiles allows a simple setup for a new sensor deployment, ensuring genericity. It was also demonstrated that the time taken to provide the necessary structure and semantics is within acceptable timeframes for user interaction.

Chapter 8

Evaluation

In this chapter, the AREA framework is analysed and evaluated against the functional requirements described in chapter 1. These requirements: simple setup, querying, on-the-fly context, knowledge worker interaction and incremental enrichment are discussed in section 8.1. Section 8.2 provides an overview of the setup required for AREA to be deployed in additional domains. Finally, section 8.3 details a number of complete datasets taken from each of the evaluation sites, and measures time to enrich and query information needs using AREA.

8.1 Meeting the Functional Requirements

8.1.1 Setup

The implementation of the context initialisation as described in section 4.2 allows the simple setup of diverse deployments. The initial set of sensor enablement and context integration information, the sensor, subject and deployment profile instances are defined by a knowledge worker prior to any usage of the sensor and thus, before any data being generated. This information is gathered using the user interface representation shown in Figures 8.1, 8.2 and 8.3. In each case the number of properties field allows

the user to specify how many properties or fields are to be input as context to describe the subject, deployment, or the sensor data output. This is necessary as it is not know in advance the quantity or type of attributes used to describe these concepts and is the first step of the knowledge worker.

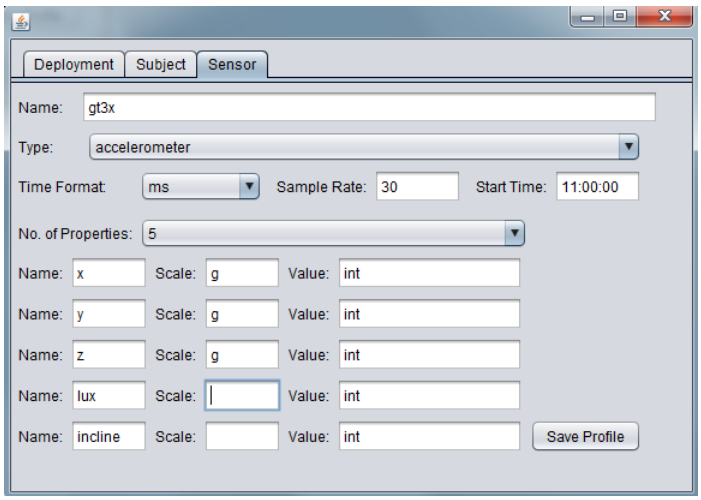

Figure 8.1: Defining a GT3X Accelerometer Profile Instance

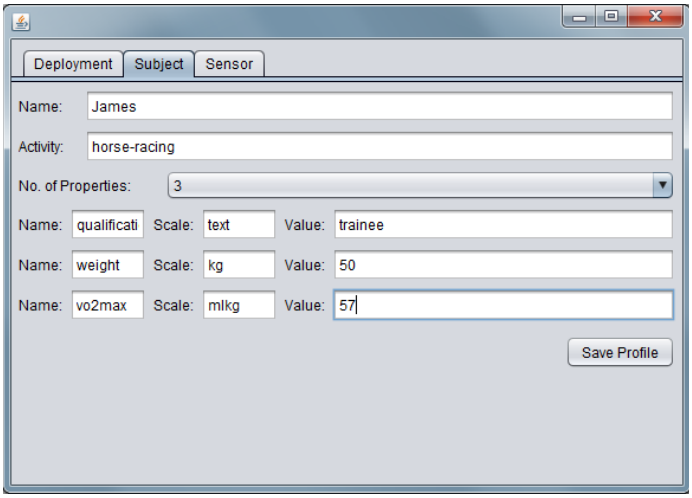

Figure 8.2: Defining a Subject Profile Instance for a Jockey

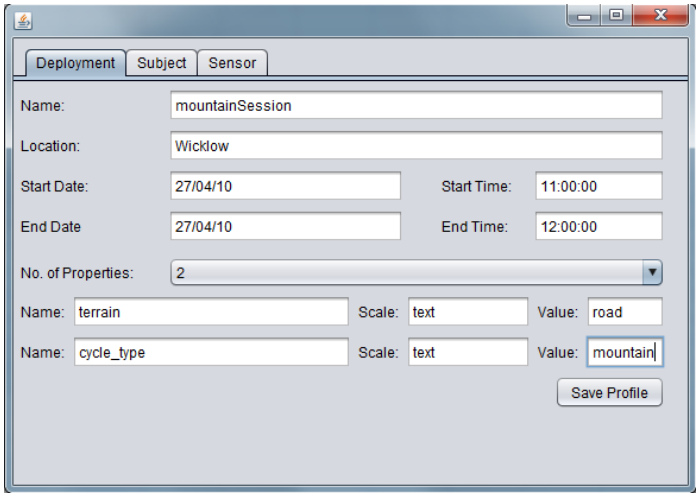

Figure 8.3: Defining a Cycling Deployment Instance

8.1.2 Querying

The functional requirements included the ability to query the sensed data based on different criteria. Following setup, read queries such as those shown in section 5.3.2 are now possible query profiles. The querying capabilities of AREA are extended as the data is enriched with new information based on previously sensed values. For instance, the identification of a "fast-canter" event in horse-racing can be encoded within the data and returned based on a query for that higher level information (rather than build a complex algorithm to discover fast cantering at query runtime).

Once information is enabled and integrated by AREA's *Sensor Enablement Processor (SEP)* and *Context Integration Processor (CIP)*, basic queries can be expressed as the data is structured and can be interpreted. To enable more complex queries, it is left to the user to continue adding context where necessary using a query profile interface.

8.1.3 On-the-fly Context

Chapter 7 demonstrated how individual queries required by the end user are made achievable by extending and transforming the dataset. This has been

achieved with the user-controlled *Data Transformation Processor (DTP)* using a strategy of defining query profiles, which when applied add new data or metadata (user context) to the scenario dataset and the AREA schema. A step by step process is employed until the information needs can be met using simple queries. The query profiles are defined using a user interface provided by AREA. This simplifies the process of selecting node values, defining when and where updates are applied, and allows for the submission of simplified queries. Thus, the DTP extends the scenario dataset as far as required by the user. As the AREA schema is extended, different datasets can be updated. This results in a schema that can be consulted by the user to verify the structure of data as it evolves through iterations of extension and update query applications.

The Data Transformation Processor provides the knowledge worker the ability to discover new contextual information, and applies it on the dataset. This is demonstrated by the examples in figures 8.4 and 8.5.

8.1.4 Knowledge Worker Interaction

The solution AREA provides to on-the-fly context provision remains a user-driven process and is designed for the knowledge worker to drive the process. Figure 8.4 shows the user interface for defining the update query to extend the scenario with a *sleeping* element (corresponding to example 7.17). The fields populate an instance of a query profile once submitted. The profiles are then stored in the AREA repository. It is left to another option on the user interface, invoked using the *"Apply Update"* tab, shown in figure 8.5, to apply this query profile.

These two options allow the definition and application of each of the updates described in section 7.1 and Appendix A, and any future updates as required. As the scientist may not be familiar with the notation for path expressions, each node can be selected from a drop-down list provided by AREA, based

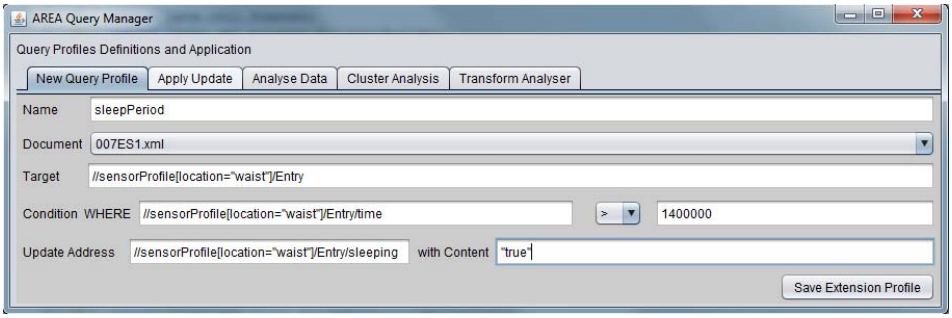

Figure 8.4: User Interface: Defining an Update Query Profile

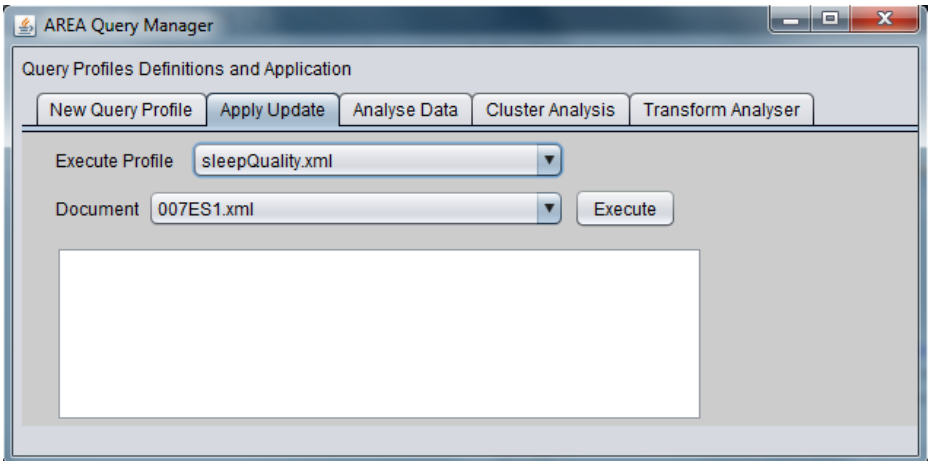

Figure 8.5: User Interface: Applying a Query Profile

on all available nodes for the current scenario. AREA expands this node selection into a full path.

8.1.5 Incremental Enrichment

The DTP is designed to allow the definition of further context based on context already discovered. This allows the knowledge worker to alter the dataset incrementally as required to discover increasingly more complex information based on information identified by previous query profiles. The ability to apply pre-defined query profiles from the AREA repository means reusability is a simple process. For instance, in the tennis domain it is necessary to categorise the court into different zones in order to later express the

queries in table 3.1. This is because all subsequent techniques of discovering information are based on location. Therefore, when the user encounters a new set of Ubisense data integrated with a scenario dataset, they can simply run an update to apply each of the update and extension query profiles necessary to provide the zone context. Afterwards, the user can apply further query profiles as required, or build new query profiles based on this context.

8.2 Evaluating AREA Across Domains

Chapter 7 has shown how the knowledge worker can define and apply various extension and update queries on a dataset in order to meet complex requirements. Each of the examples involved a human being monitored as they performed in some sport or during work or rest. At a basic level, a new sensor being deployed in any of these environments requires only the definition of a new sensor profile to be created. Additional deployments can also be introduced with a new `deployment` profile instance.
A `subject` profile has been designed as a human-based construct. However, sensors deployed on objects can easily be managed by AREA. This can be achieved with a slight alteration of the content of a subject profile instance. When presented with a new domain or deployment environment, the following steps ensure the ability to manage and interpret the sensor data, and related context.

- Sensor profile - Defined for each sensor deployed to describe the sensor data output. Can be reused if the same sensor type is deployed in multiple deployments. For example, a temperature sensor can be defined with a sample rate of 1 second, monitoring temperature in degrees Celsius while deployed on a vehicle. This is defined by a knowledge worker using the sensor profile interface in figure 8.1.

- Subject profile - Defined for each subject involved in a deployment

to describe key features of the participant. This can be a person or any object scientists wish to monitor. For instance, it could defined to describe the key features of a car, with properties such as make / model, engine size and top speed. Once defined for a subject it can be reused in any deployment where the properties are still valid. AREA allows flexibility in the terms used to describe the subject.

- Deployment profile - Defined for each deployment instance, it describes the time and location and a number of properties to describe the deployment. A new domain, such as a motor race might include properties such as race class, or weather conditions.

- Query profile - Defined as required by the knowledge worker to further enrich the dataset with relevant information, using evidence from sensor values, context elements and user input. In a motor racing domain, this might include defining periods of the race when a safety car was deployed.

This genericity is possible due to the XML representation of data. The extensibility of XML allows the addition of new nodes where necessary, or new datasets corresponding to different subjects, sensors or deployments.
As some of the algorithms are defined for human-based analysis, such as the definition of energy expenditure using an accelerometer, these can be applied in multiple domains. Providing the path to the nodes which form the update or conditions of the queries are the same, the same query can be applied without alteration. There may be slight alterations required if (for instance) the accelerometer is located on a different part of the body. In these cases, an existing query profile can simply by modified by the user prior to its application on the new dataset.

8.3 Measuring Efficiency of Data Transformation

This section presents an evaluation of AREA framework's efficiency in order to determine whether the time taken to transform sensor data from its initial format to a format extended with contextual information is reasonable. As the goal of this work is not based on optimisation, and the data is to be queried following deployment (rather than in real time), reasonable times are sufficient. By reasonable, it is assumed that the user requires this information within seconds.

Sensor	**Size of Initial Data (kB)**	**Time to Enrich (s)**	**Size of XML**
gt3x+ (2)	477	0.586	3,041kB
sensewear (2)	336	0.215	1,720kB
SAR ES	**813**	**0.801**	**5,761kB**
GCDC	25,100	12.607	103MB
PowerTap	173	0.1	481kB
gt3x (2)	213	0.232	1,986kB
Cyclist GM	**25,486**	**12.939**	**106MB**
gt3x (5 indoor)	404	0.487	3,095kB
PowerTap	84	0.076	357kB
Cosmed	127	0.086	453kB
GPS	159	N/A	159kB
gt3x (5 outdoor)	428	0.6	3.8MB
Cosmed (outdoor)	82	0.061	295kB
Jockey CG	**1,284**	**1.31**	**5.1MB**
Ubisense (36)	14,000	8.241	49MB
Tennis VAR	**14,000**	**8.241**	**49MB**

Table 8.1: Conversion from Csv to XML

Table 8.1 presents a list of sensors from a deployment in each of the evaluation sites: horse-racing, cycling, search and rescue and tennis. The table details the name of the sensor, the initial data size, the time taken by AREA to convert this to XML, and the total size of that sensors data in that deployment. The total amount of values for each of the datasets examined is also recorded. The data from the horse-racing domain has both indoor and

outdoor data listed in the table. As can be seen, the cycling deployment took almost 13 seconds to enrich, primarily because of the large GCDC data sources. The tennis dataset consists of all *Ubisense* information recorded in many scenarios. The SAR, cycling and horse-racing data is from the deployment details for one specific participant.

Domain	**Dataset 1**	**Dataset 2**	**Dataset 3**
SAR	ES, 5.8MB	S2, 5.9MB	S3, 5.8MB
Cycling	GM, 106MB	P1, 26MB	M1, 21MB
Horse Racing	CG, 5.1MB	J2, 5.3MB	J3, 4.7MB
Tennis	KV, 2MB	B1, 2MB	KV2, 2MB

Table 8.2: Dataset properties for each domain

The times displayed in table 8.1 do not include time taken for the user to create profiles in order to enrich the dataset. This will depend on how familiar the user is with the system, but one should bear in mind that once the enrichment has been specified (profiles), it is not necessary to do so again. However, three datasets in table 8.3 and the time taken (in ms) to apply each of the query profiles (from section 7.1) to these datasets are presented. The time to update the dataset (update queries) is measured, rather than the schema update (extension queries). This is because a schema update only ever involves changing one line of code on a small `.xsd` file. Queries were expressed using the eXist 2.0 [18] database on an Intel core 2 Due processor 3Ghz with 4GB RAM running Windows 7 32bit.

Table 8.2 shows the datasets used for each of the domains queried in the remaining experiments. In each case a label representing the subject and the size of the dataset being queried is displayed.

The queries resulting in the most updates - *workenergy*, *sleeping* and *pronesleep* took the longest to execute, at between 11 and 15 seconds per dataset. This was because these profiles involved transformations on almost every `Entry` element within the data set. Simpler updates, such as *RMR* and *TEF* in-

Profile	Dataset 1 (ms)	Dataset 2 (ms)	Dataset 3 (ms)
RMR	504	505	499
TEF	571	597	553
scalingfactor	611	512	503
workenergy	12,165	12,712	12,690
totalenergy	1,391	1,450	1,511
lefthandedwhip	1,971	1,318	723
fastcantergait	2,349	2,415	1,890
lefthandedwhipv2	2,010	1,516	890
sleeping	14,395	15,300	14,933
proneSleep	11,391	11,410	11,001
sleepQualityCalc	1,515	1,568	1,581
goodSleep	631	793	709

Table 8.3: Time to Apply Queries

volved the update of only one node following some calculation. Thus, their execution times were shorter, taking less than a second to execute.

Finally, an evaluation of the time taken to query for complex information needs, after the application of the query profiles is presented. Following the relevant data transformations, each query has been reduced to a simple XQuery expression and thus, query times are significantly lower than previous enablement and enrichment times. In each case, the query is expressed on three datasets. For each, the time taken to execute is shown, along with the result obtained. As can be seen, the times are all below 30ms.

Query	Dataset 1	Dataset 2	Dataset 3
Total energy exp. (SAR, cycle, jockey)	7ms, 2,336kcal	8ms, 637kcal	7ms, 295kcal
Horse whip (Jockey)	12ms, 29 whips	13ms, 27 whips	13ms, 15 whips
Good sleep quality (SAR)	11ms, Yes (92%)	10ms, Yes (93%)	11ms, Yes (92%)
Detect serve (Tennis)	28ms, 87 serves	26ms, 72 serves	25ms, 91 serves

Table 8.4: Time and Result of Read Queries

Chapter 9

Conclusions

This concluding chapter summarises this work and reassesses the hypothesis and research questions. It also describes future work which could be undertaken to broaden the overall impact of this work and extend the AREA framework with new functionality. A summary of the limitations of this work is also presented.

This chapter is structured as follows: Section 9.1.2 presents a summary of this dissertation and reasseses the research questions and contributions. Section 9.2 describes the limitations of AREA and concludes the dissertation with a discussion of areas of future research.

9.1 Summary and Reassessment

9.1.1 Thesis Summary

Chapter 1 introduced the recent emergence of low cost, portable and reliable sensing devices. It discussed how scientists often deploy these sensors in varied situations across multiple domains. Using multiple heterogeneous sensors facilitates the gathering of large volumes of data with little human interaction. However, when it comes to analysing these data sources, a number of challenges arise. The key issue is the *semantic gap*, i.e. the gap

between the information needs of the scientist and the sensor data in its initial format. This research aimed to put in place the structures to bridge this gap using a limited amount of user-interaction.

Chapter 2 analysed the approach of 11 different systems. In each case, their approach was evaluated based on their ability to enable a user to adapt their analysis, customise their approach and fine tune algorithms or analysis. A system which is built for each and every domain is not appropriate. Instead, what is wanted is a general system where the user arrives with their information needs and deployment characteristics. It was concluded that no existing solution provides these key objectives.

Chapter 3 introduced four real world sensor deployments provided by sport and health physiologists. In each case, the sensors involved, user requirements and data gathered were examined. This provided a suitably broad set of requirements and issues on which to base out design.

The Activity Retrieval for Enrichment and Analysis (AREA) framework was then described. Using AREA, a *knowledge worker* can drive the enrichment process, from initial low-level data to XML. This allows the user to define new sensors, subjects and deployment details for some activity scenario. AREA processes this information, integrates the context and provides an interface for the user to interact with the data.

Chapter 5 presented the user controlled data transformation process within AREA. This allows a knowledge worker to interact with existing data, and use it as evidence to derive higher levels of abstraction. This is achieved through a system of query profiles which can retrieve data, update data and alter data structure. Each of the query profiles are stored in AREA's repository upon definition for later *application*. These facilitate the knowledge worker in applying all necessary context to a dataset to allow future queries by a non-IT user, using simple means.

The *Automated Activity Analysis* (AAA) processor was then presented in

chapter 6, to provide an alternative method to analysing data. Instead of user-interaction, three system-defined modules aid with improving the results obtained from user-controlled query profiles. In the first module, *baseline analysis* performs an initial statistical analysis of a specific set of sensor or context data which is stored in AREA's metadata repository as a new *dataset profile*. *Cluster analysis*, implements a basic *k-means clustering* algorithm using input from the dataset profile - specifically the number of unique values within a set of data, and the classification details from the *conditionConstruct*. This allows an automated re-classification of sensor data into new bands. A third module *metadata analysis*, uses the *updateConstruct* from the dataset profile to determine alternative updates for query profiles. It allows experimentation with new values in place of mutable variables, replace classification values with those identified by cluster analysis, and anticipate the results of future data. Together these modules provide added value to the knowledge worker by automatically experimenting with data and metadata.

Chapter 7 revisited the case studies from chapter 3, and demonstrated the performance of AREA meeting user requirements in each domain. Chapter 8 evaluated AREA based on the functional requirmeents identified in chapter 1, and efficiency - how the solution can be delivered efficiently in terms of usage and performance. Although optimisation and performance were not a key priority in this research, chapter 8 demonstrated that the time taken to enable and enrich sensor data is within normal working boundaries.

Finally, this chapter addressed the research questions and contributions first described in chapter 1, and summarised the limitations of this research.

9.1.2 Reassessing the Aims and Objectives

The hypothesis put forward in this research was that by using a framework to structure and contextualise low level data acquisition tools such as sensors,

this information can be used for high level query expressions and knowledge extraction using basic user-interactions rather than expensive human-based operations.

In section 1.3 it was stated that closing the semantic gap between sensor data and user requirements was the key to verifying this hypothesis. This section revisits each of the research questions answered during this research.

RQ1: Is it possible to enable sensors such that their output can be queried at a high level?

A layered set of processes can allow the enablement of sensor data. To provide this, AREA standardises the definition of key information - subjects, deployments and sensors with everything represented in XML. The Context Initialisation and Sensor Enablement processors implement these processes. The representation of data in XML format allows interoperability and the ability to query using a standard query language. The inclusion of a user interface allows the high level querying capabilities on sensor data as posed by this research question.

RQ2: How is it possible to facilitate the data transformation necessary to facilitate highly complex queries?

The XML representation and schema description of AREA also facilitates the transformation of sensor information. The Data Transformation Processor as described in chapter 5, provides the ability to transform the data by changing the content and structure of data gathered from a sensor deployment. This allows the context built on lower-level context, and the incremental improvement of querying capabilities to subsequent users. Users are now able to

RQ3: How can one enable the user to provide the missing semantics necessary to support higher level queries?

The third research question was addressed by the Automated Activity Analysis (AAA) processor. As described in chapter 6, utilising information gathered from the knowledge worker (in the form of query profiles) can lead to the discovery of new knowledge. The clustering and baseline analysis of sensor data can aid the user with reclassifying their own algorithms and assist with improving the discovery of higher-level information. Results obtained during this analysis are presented to the user in the form of a portal. Should these improvements lead to better techniques for event definition or detection, the changes are reflected in the data and are queryable following analysis.

RQ4: Can all of this be provided in a framework that allows heterogeneous sensor devices (sensors of varying types and configurations) used in heterogeneous domains to meet the needs of different user types?

AREA has demonstrated an implementation of the functional requirement across a variety of sensors in four distinct case studies. While not an exhaustive solution, there is extensive support for new sensors, subjects and deployments in other domains which require management. The flexibility of XML, the simplicity of setup and the support for knowledge worker and automated identification of occurrences or events of interest provides the framework necessary to allow heterogeneous sensors and environments. The framework enables the knowledge worker to enrich the data with the necessary context, and subsequent users are facilitated with read querying capabilities without the need to fully understand the underlying data gathered.

Each of the goals identified in section 1.3.2 have been achieved while an-

swering these research questions. Meeting these goals and answering these questions while designing the AREA framework has shown support for the hypothesis. The semantic gap - the gap between user information needs and the data output from sensors - has been bridged by implementing the AREA framework. As a result it is possible to apply high-level or free-form queries on a set of semi-structured sensor data enriched by a knowledge worker using these tools.

9.2 Future Research

9.2.1 Limitations

The AREA framework is not an exhaustive solution for providing the ability of data transformations. Functionality could be extended further by providing more complex mathematical equations to update the data. This may be advantageous when dealing with more scientific or mathematical user types who wish to perform extensive data analysis using complex equations.
Another limitation of this work is the lack of a presentation layer on which usability tests could be carried out. Collaboration with knowledge workers allowed the gathering of query requirements and allowed them to formulate their queries which were then implemented. A usability study on a fully functional presentation layer would allow testing of AREA on a large sample of users to discover where improvements could be made if necessary.

9.2.2 Further Research

Minimising user workload in user-controlled transformations

When creating filters for classifications, there can be a great deal of repetition and one area of future research might seek to automatically generate more update queries. Consider a requirement to subdivide a smart space into a chess board. This would require 64 different transformations based

on coordinates. It would be advantageous for AREA to incorporate a *smart portal* which identifies the similarities in the data being input. Upon doing so, an algorithm could then start to suggest the next expected coordinates, or auto-generate groups of update queries. Such functionality can apply for many classifications where the data boundaries are consistent in their differences.

Automatic Transformations

As part of metadata analysis, AREA suggests multiple different values for parameters in an update. The results obtained by using these additional values are then presented to the user via a portal. At this time the user picks the best parameter set for the dataset scenario. Another area of future research could investigate if AREA could integrate a machine learning algorithm, which learns from the value set selected by the user. If the same value set is selected for a specific kind of dataset, then future, similar datasets can have the selected value input automatically. This could be further improved if the value set can refine its boundaries based on user selections. For example, if the user chooses 4 modifications say to add 0.25, 0.5, 0.75 and 1 and always choose 0.75, then the system will use more fine grained selections close to 0.75, e.g. 0.74 and 0.76.

A Presentation Layer

This thesis did not focus on visualisation, digital dashboard or any form of customised portal. Instead it was decided to focus on functionality and a generic framework for diverse sensor networks. The work presented in this dissertation could have greater impact if future research explores how these three areas could provide a presentation layer for AREA. This might include the ability to visualise the schema trees of different scenarios or graph the results of cluster analysis to aid the user with identifying valuable

information.

Appendix A

Publications arising from this work

A number of publications resulted from the research undertaken for this dissertation. Two (RCIS 2009 and DMSN 2009) were based on early prototypes based on tennis, utilising the Ubisense localisation sensors to define and identify events such as serves and game boundaries. This work was then improved using a modular approach to context provision, removing the requirement for location in detecting events (DEXA 2010). Further research on user-assisted data transformation and automated analysis techniques in two new domains then resulted in publications in cycling (BNCOD 2011) and horse-racing (DaWaK 2011) applications meeting requirements for knowledge workers.

1. **Knowledge acquisition from sensor data in an equine environment.**

   Kenneth Conroy, Gregory May, Mark Roantree, Giles Warrington, Sarah Cullen and Adrian McGoldrick

   In: 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2011), Toulouse, France, August 29-September

2,2011, pp.432-444., Springer Proceedings.

2. **Expanding sensor networks to automate knowledge acquisition.**

   Kenneth Conroy, Gregory May, Mark Roantree and Giles Warrington

   In: Advances in Databases - 28th British National Conference on Databases, (BNCOD 2011), Manchester, UK, July 12-14, 2011, pp. 97-107., Springer Proceedings.

3. **Enrichment of raw sensor data to enable high-level queries.**

   Kenneth Conroy and Mark Roantree

   In: 21st international conference on Database and expert systems applications, (DEXA 2010), Bilbao, Spain, August 30 - September 3, 2010, pp. 462-469., Springer Proceedings.

4. **Extracting tennis statistics from wireless sensing environments.**

   Adel Shaeib, Kenneth Conroy and Mark Roantree

   In: 6th International Workshop on Data Management for Sensor Networks, (DMSN 2009), Lyon, France, August 24, 2009, ACM Proceedings.

5. **Querying XML data streams from wireless sensor networks: An evaluation of query engines.**

   Martin F. O'Connor, Kenneth Conroy, Mark Roantreem, Alan F. Smeaton and Niall Moyna

   In: Third IEEE International Conference on Research Challenges in Information Science, (RCIS 2009), Fes, Morocco, 22-24 April 2009, pp. 23-30, IEEE Proceedings.

Appendix B

XML Schema

Definition B.1 *Query Profile Schema*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="updateQuery">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="name"/>
                <xs:element ref="doc"/>
                <xs:element ref="target"/>
                <xs:element ref="condition"/>
                <xs:element ref="update"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="update">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="address"/>
                <xs:element ref="content"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="target">
        <xs:complexType/>
    </xs:element>
    <xs:element name="readQuery">
        <xs:complexType>
            <xs:sequence>
```

```
            <xs:element ref="name"/>
            <xs:element ref="doc"/>
            <xs:element ref="target"/>
            <xs:element ref="condition"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="queryProfile">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="readQuery"/>
            <xs:element ref="updateQuery"/>
            <xs:element ref="extensionQuery"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="name">
    <xs:complexType/>
</xs:element>
<xs:element name="extensionQuery">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="doc"/>
            <xs:element ref="target"/>
            <xs:element ref="condition"/>
            <xs:element ref="update"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="doc">
    <xs:complexType/>
</xs:element>
<xs:element name="content">
    <xs:complexType/>
</xs:element>
<xs:element name="condition">
    <xs:complexType/>
</xs:element>
<xs:element name="address">
    <xs:complexType/>
</xs:element>
```

```
</xs:schema>
```

Definition B.2 *Dataset Profile Schema*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="value"><xs:complexType/>
    </xs:element>
    <xs:element name="updateConstruct">
        <xs:complexType><xs:sequence>
            <xs:element ref="variables"/>
            <xs:element ref="fields"/>
            <xs:element ref="complex"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="property">
        <xs:complexType><xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="value"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="properties">
        <xs:complexType><xs:sequence>
            <xs:element ref="property" maxOccurs="unbounded"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="name"><xs:complexType/></xs:element>
    <xs:element name="id"><xs:complexType/></xs:element>
    <xs:element name="fixed"><xs:complexType/></xs:element>
    <xs:element name="deployment"><xs:complexType/></xs:element>
    <xs:element name="fields">
        <xs:complexType><xs:sequence>
            <xs:element ref="name"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="variables">
        <xs:complexType><xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="value"/>
            <xs:element ref="fixed"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="conditionConstruct">
```

```
        <xs:complexType><xs:sequence>
            <xs:element ref="variables"/>
            <xs:element ref="fields"/>
            <xs:element ref="complex"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="complex">
        <xs:complexType><xs:sequence>
            <xs:element ref="name"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="datasetprofile">
        <xs:complexType><xs:sequence>
            <xs:element ref="id"/>
            <xs:element ref="name"/>
            <xs:element ref="deployment"/>
            <xs:element ref="properties"/>
            <xs:element ref="updateConstruct"  minOccurs="0"/>
            <xs:element ref="conditionConstruct"  minOccurs="0"/>
        </xs:sequence></xs:complexType>
    </xs:element>
</xs:schema>
```

Definition B.3 *Scenario XML Schema*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="value">
        <xs:complexType/>
    </xs:element>
    <xs:element name="type">
        <xs:complexType/>
    </xs:element>
    <xs:element name="timing">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="format"/>
                <xs:element ref="sampleRate"/>
                <xs:element ref="startTime"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="subject">
```

```
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="id"/>
            <xs:element ref="name"/>
            <xs:element ref="activity"/>
            <xs:element ref="properties"/>
            <xs:element ref="sensor" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="startdatetime">
    <xs:complexType/>
</xs:element>
<xs:element name="startTime">
    <xs:complexType/>
</xs:element>
<xs:element name="sensor">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="id"/>
            <xs:element ref="name"/>
            <xs:element ref="type"/>
            <xs:element ref="timing"/>
            <xs:element ref="fields"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="scenario">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="id"/>
            <xs:element ref="name"/>
            <xs:element ref="location"/>
            <xs:element ref="startdatetime"/>
            <xs:element ref="enddatetime"/>
            <xs:element ref="properties"/>
            <xs:element ref="subject"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="scale">
    <xs:complexType/>
```

```
</xs:element>
<xs:element name="sampleRate">
    <xs:complexType/>
</xs:element>
<xs:element name="property">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="scale"/>
            <xs:element ref="value"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="properties">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="property"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="name">
    <xs:complexType/>
</xs:element>
<xs:element name="location">
    <xs:complexType/>
</xs:element>
<xs:element name="id">
    <xs:complexType/>
</xs:element>
<xs:element name="format">
    <xs:complexType/>
</xs:element>
<xs:element name="fields">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="field"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="field">
    <xs:complexType>
        <xs:sequence>
```

```
                <xs:element ref="name"/>
                <xs:element ref="scale"/>
                <xs:element ref="value"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="enddatetime">
        <xs:complexType/>
    </xs:element>
    <xs:element name="activity">
        <xs:complexType/>
    </xs:element>
</xs:schema>
```

Appendix C

Enablement Profiles

Definition C.1 *Sensor Profile: GT3X Accelerometer*

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/sensorProfile.xsd">
    <id>2</id>
    <name>gt3x+</name>
    <type>accelerometer</type>
    <timing>
        <format>ms</format>
        <sampleRate>30</sampleRate>
        <startTime/>
    </timing>
    <fields>
        <field>
            <name>x</name>
            <scale>count</scale>
            <value>int</value>
        </field>
        <field>
            <name>y</name>
            <scale>count</scale>
            <value>int</value>
        </field>
        <field>
            <name>z</name>
            <scale>count</scale>
            <value>int</value>
```

```
        </field>
        <field>
            <name>lux</name>
            <scale/>
            <value>int</value>
        </field>
        <field>
            <name>incline</name>
            <scale/>
            <value>int</value>
        </field>
    </fields>
</SensorProfile>
```

Definition C.2 *Sensor Profile: GT3X+ Accelerometer*

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/sensorProfile.xsd">
    <id>2</id>
    <name>gt3x+</name>
    <type>accelerometer</type>
    <timing>
        <format>ms</format>
        <sampleRate>100</sampleRate>
        <startTime/>
    </timing>
    <fields>
        <field>
            <name>x</name>
            <scale>count</scale>
            <value>int</value>
        </field>
        <field>
            <name>y</name>
            <scale>count</scale>
            <value>int</value>
        </field>
        <field>
            <name>z</name>
            <scale>count</scale>
```

```
            <value>int</value>
        </field>
        <field>
            <name>lux</name>
            <scale/>
            <value>int</value>
        </field>
        <field>
            <name>incline</name>
            <scale/>
            <value>int</value>
        </field>
    </fields>
</SensorProfile>
```

Definition C.3 *Sensor Profile: Sensewear armband*

```
<?xml version="1.0" encoding="UTF-8"?>
<SensorProfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/sensorProfile.xsd">
    <id>3</id>
    <name>sensewear</name>
    <type>physiological</type>
    <timing>
        <format>realtime</format>
        <sampleRate>0.0166</sampleRate>
        <startTime/>
    </timing>
    <fields>
        <field>
            <name>time</name>
            <scale>realtime</scale>
            <value>string</value>
        </field>
        <field>
            <name>transverseAccelP</name>
            <scale/>
            <value/>
        </field>
        <field>
            <name>longAccelP</name>
            <scale/>
```

```
    <value/>
</field>
<field>
    <name>heatFluxA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>skinTempA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>transverseAccelA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>longAccelA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>nearbodyTempA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>transverseAccelM</name>
    <scale/>
    <value/>
</field>
<field>
    <name>longAccelM</name>
    <scale/>
    <value/>
</field>
<field>
    <name>stepCounter</name>
    <scale/>
    <value/>
</field>
```

```
<field>
    <name>GSRA</name>
    <scale/>
    <value/>
</field>
<field>
    <name>lyingdown</name>
    <scale/>
    <value/>
</field>
<field>
    <name>sleep</name>
    <scale/>
    <value/>
</field>
<field>
    <name>physicalActivity</name>
    <scale/>
    <value/>
</field>
<field>
    <name>energyExp</name>
    <scale/>
    <value/>
</field>
<field>
    <name>sedentary</name>
    <scale/>
    <value/>
</field>
<field>
    <name>moderate</name>
    <scale/>
    <value/>
</field>
<field>
    <name>vigorous</name>
    <scale/>
    <value/>
</field>
<field>
    <name>vVigorous</name>
```

```
            <scale/>
            <value/>
        </field>
        <field>
            <name>METs</name>
            <scale/>
            <value/>
        </field>
    </fields>
</SensorProfile>
```

Appendix D

Query Profile Instances

Example D.1 *Update Query: Right-Handed Whip V2*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
    <name>righthandwhip</name>
    <doc>S15CG.xml</doc>
    <target>//sensorProfile[location="RHwrist"]/fields/Entry</target>
    <condition>//sensorProfile[location="RHwrist"]/fields/Entry/x > MAXVALUE AND
     //sensorProfile[location="RHwrist"]/fields/Entry/y > MAXVALUE AND
      //sensorProfile[location="RHwrist"]/fields/Entry/z > MAXVALUE AND
       //sensorProfile[location="saddle"]/fields/Entry/gait = 'fast-canter'
    </condition>
    <update>
         <address>/y/sensorProfile[location="RHwrist"]/fields/Entry/whip</address>
         <content>"rightwhip"</content>
    </update>
</updateQuery>
```

Example D.2 *Update Query: High Work Energy*

```
<updateQuery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche
maLocation="file:///C:/NewWorkspace/csv2xml/ch7/QueryProfile.xsd">
    <name>workEnergyHigh</name>
    <doc>007ES1.xml</doc>
    <target>///sensorProfile[location="waist"]/fields/Entry/workenergy</target>
    <condition>///sensorProfile[location="waist"]/fields/Entry
    /workenergy >= 7200</condition>
    <update>
```

```
        <address>///sensorProfile[location="waist"]/fields/Entry/workenergyIntensity
        </address>
        <content>high</content>
    </update>
</updateQuery>
```

Bibliography

[1] Barnaghi, P.M., Meissner, S., Presser, M. and Moessner, K. 2009. Sense and sens'ability: Semantic data modelling for sensor networks. *Proceedings of the ICT Mobile Summit.*

[2] Beigl, M., Krohn A., Zimmer, T. and Decker C. 2004. Typical Sensors needed in Ubiquitous and Pervasive Computing. *International Conference on Networked Sensing Systems (INSS).*

[3] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A. and Riboni, D. 2010. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing.*

[4] Botts, M., Percivall, G., Reed, C., and Davidson, J. 2008. OGC sensor web enablement: Overview and high level architecture. *GeoSensor networks.*

[5] Buckner, J., Pahl M., Stahlhut O. and Liedtke, C. E. 2001. geoAIDA - A Knowledge Based Automatic Image Data Analyser for Remote Sensing Data. *International ICSC Symposium on Advances in Intelligent Data Analysis.*

[6] Chen, H., Perich, F., Finin, T., Joshi, A. 2004. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. *International Conference on Mobile and Ubiquitous Systems: Networking and Services.*

[7] Churcher, G., Bilchev, G., Foley, J., Gedge, R. and Mizutani, T. 2008. Experiences applying sensor web enablement to a practical telecare application. *International Symposium on Wireless Pervasive Computing (ISWPC).*

[8] Churcher, G. E. and Foley, J. 2009. Applying and extending sensor web enablement to a telecare sensor network architecture. *International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE).*

[9] Common Alerting Protocol. 2010. *Common Alerting Protocol (CAP).* Available from http://docs.oasis-open.org/emergency/cap /v1.2/CAP-v1.2-os.html [Accessed 7th July 2013]

[10] Conroy, K., May, G., Roantree, M. and Warrington, G. 2011. Expanding Sensor Networks to Automate Knowledge Acquisition. *British National Conference on Databases (BNCOD).*

[11] Conroy, K., May, G., Roantree M., Warrington G., Cullen, S. and McGoldrick A. 2011. Knowledge acquisition from sensor data in an equine environment. *Data Warehousing and Knowledge Discovery.*

[12] Conroy, K. and Roantree, M. 2010. Enrichment of raw sensor data to enable high-level queries. *DEXA'10: Proceedings of the 21st international conference on Database and expert systems applications.*

[13] Cosmed. 2013. *Cosmed K4b2.* Available from http://www.cosmed.com /en/products/cardio-pulmonary-exercise-testing/k4-b2-mobile-cpet [Accessed 7th July 2013]

[14] Dey, A. K. 2001. Understanding and Using Context. *Personal and Ubiquitous Computing*

[15] Di, L., Kresse, W. and Kobler, B. 2004. The Current Status and Future Plan of the ISO 19130 Project. *International Society for Photogrammetry and Remote Sensing (ISPRS).*

[16] Diamond, D. and Hanratty, V. C. A. 1997. *Spreadsheet Applications in Chemistry Using Microsoft Excel.* Wiley.

[17] Emergency Data Exchange Language. 2006. *Emergency Data Exchange Language (EDXL)*, Available from http://docs.oasis-open.org /emergency/edxl-de/v1.0/EDXL-DE_Spec_v1.0.pdf [Accessed 6th January 2013]

[18] eXist. 2013. *eXist XML database*, Available from http://exist-db.org/exist/index.xml [Accessed 7th July 2013]

[19] Golden Cheetah. 2013. *Golden Cheetah*, Available from http://goldencheetah.org/download.html [Accessed 7th July 2013]

[20] Goldman, R. and Widom, J. 1997. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. Available from http://ilpubs.stanford.edu:8090/264/

[21] GT3X Accelerometer. 2013. Available from http://support.theactigraph.com/product/GT3X-device [Accessed 6th January 2013]

[22] GT3X+ Accelerometer. 2013. Available from http://support.theactigraph.com/product/GT3Xplus-device [Accessed 6th January 2013]

[23] Henricksen, K., Indulska, J. and McFadden, T. 2005. Modelling Context Information with ORM. *On the Move to Meaningful Internet Systems (OTM).*

[24] Henricksen, K. and Indulska, J. 2006. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing.*

[25] Henricksen, K., Indulska, J. and Rakotonirainy, A. 2002. Modeling context information in pervasive computing systems. *Pervasive Computing*

[26] Henricksen, K., Livingstone, S. and Indulska, J. 2004. Towards a hybrid approach to context modelling, reasoning and interoperation. *1st International Workshop on Advanced Context Modelling, Reasoning and Management.*

[27] Huang, Y-P., Chiou, C-L. and Sandnes, F.E. 2009. An intelligent strategy for the automatic detection of highlights in tennis video recordings. *Expert Systems with Applications (ESWA).*

[28] IBM InfoSphere streams. 2013. Available from http://www-01.ibm.com /software/data/infosphere/streams/ [Accessed 6th January 2013]

[29] IBM SPSS Statistics. 2013. Available from http://www-01.ibm.com /software/analytics/spss/ [Accessed 6th January 2013]

[30] Java Message Service. 2011. Available from http://www.oracle.com /technetwork/java/jms/index.html [Accessed 6th January 2013]

[31] Jiawei, H., and Kamber, M. 2006. *Data mining: concepts and techniques.* Morgan Kaufmann.

[32] Jonathan L., Blake H. and Gaetano B. 2004. Are You with Me? using accelerometers to determine if two devices are carried by the same person. *Proceedings of Second International Conference on Pervasive Computing (Pervasive).*

[33] Katsiri, E. and Mycroft, A. 2003. Knowledge representation and scalable abstract reasoning for sentient computing using first-order logic.

Proceedings of Challenges and Novel Applications for Automatic Reasoning (CADE-19).

[34] Katsiri, E., Bacon, J. and Mycroft, A. 2008. Scafos: Linking sensor data to context-aware applications using abstract events. *International Journal of Pervasive Computing and Communications.*

[35] Lee, K., IEEE 1451: A Standard in Support of Smart Transducer Networking, Instrumentation and Measurement Technology Conference. 2000. *Instrumentation and Measurement Technology Conference (IMTC).*

[36] Madden, S. Franklin, M., Hellerstein, J. and Hong, W. 2005. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Transactions on Database Systems.*

[37] Mileo, A., Merico, D. and Bisiani, R. 2008. Wireless sensor networks supporting context-aware reasoning in assisted living. *International conference on Pervasive Technologies Related to Assistive Environments.*

[38] O'Connor, M. F., Conroy, K., Roantree, M., Smeaton, A. F. and Moyna, N. M. 2009. Querying XML data streams from wireless sensor networks: An evaluation of query engines. *Research Challenges in Information Science, RCIS.*

[39] OWL: Web Ontology Language Semantics and Abstract Syntax. 2013. Online resource http://www.w3.org/TR/owl-semantics/ [Accessed 6th January 2013]

[40] Protege 2000. 2013. Available from http://protege.stanford.edu/ [Accessed 6th January 2013]

[41] Resource Description Framework (RDF). 2013. Available from http://www.w3.org/RDF/ [Accessed 6th January 2013]

[42] Russomanno, D.J., Kothari, C. and Thomas, O. 2005. Sensor Ontologies: from shallow to deep models. *Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory (SSST).*

[43] Russomanno, D. J., Kothari, C. and Thomas, O. 2005. Building a Sensor Ontology: A practical approach leveraging ISO and OGC models. *International Conference on Artificial Intelligence.*

[44] Schoofs, A., Ruzzelli, A. G. and O'Hare, G. M. 2010. Appliance activity monitoring using wireless sensors. *International Conference on Information Processing in Sensor Networks.*

[45] Shaeib, A., Cappellari, P. and Roantree, M. 2010. A framework for real-time context provision in ubiquitous sensing environments. *IEEE Symposium on Computers and Communications (ISCC).*

[46] Semantic Web Rule Language. 2013. Available from http://www.w3.org/Submission/SWRL/ [Accessed 6th January 2013]

[47] SenseWear System (BodyMedia). 2013. Available from http://sensewear.bodymedia.com/ [Accessed 6th January 2013]

[48] SensorML. 2013. Available from http://www.opengeospatial.org /standards/sensorml [Accessed 6th January 2013]

[49] Sensor Model Language (SensorML). 2011. Available from http://www.opengeospatial.org/standards/sensorml [Accessed 6th January 2013]

[50] Sensor Observation Service. 2011. Available from http://www.open geospatial.org
/standards/sos [Accessed 6th January 2013]

[51] Sensor Web Enablement. 2011. Available from http://www.open geospatial.org/projects/groups/sensorweb [Accessed 6th January 2013]

[52] Shaeib, A., Conroy, K. and Roantree, M. Extracting tennis statistics from wireless sensing environments. *International Workshop on Data Management for Sensor Networks (DMSN).*

[53] Sheth, A. P., Henson, C. A. and Sahoo, S. S. 2008. Semantic Sensor Web. *IEEE Internet Computing Vol. 12.*

[54] Smyth, B. 2009. The sensor web : bringing information to life. *ERCIM News, (76): 3-3, Special theme: the sensor web*

[55] Strang, T., Popien, C. L., Frank, K. 2003. CoOL: A Context Ontology Language to enable Contextual Interoperability. *International Conference on Distributed Applications and Interoperable Systems (DAIS2003).*

[56] Suggested Upper Merged Ontology (SUMO). 2013. Available from http://suo.ieee.org/SUO/SUMO/index.html [Accessed 6th January 2013]

[57] The Open Geospatial Consortium (OGC). Available from http://www.opengeospatial.org/ [Accessed 6th January 2013]

[58] The Race Around Ireland. 2010. Available from http://www.race aroundireland.com/ [Accessed 6th January 2013]

[59] The Rules of Tennis. 2013. Available from http://www.itftennis.com /shared/medialibrary/pdf/original/IO_38810_original.PDF [Accessed 6th January 2013]

[60] TinyDB. 2013. Available from http://telegraph.cs.berkeley.edu /tinydb/ [Accessed 6th January 2013]

[61] The Turf Club. 2013. Available from http://www.turfclub.ie /site/ [Accessed 6th January 2013]

[62] Tien, M., Lin, Y. and Wu, J. 2009. Sports wizard: sports video browsing based on semantic concepts and game structure. *International conference on Multimedia (MM).*

[63] TrainingPeaks WKO+. 2013. Available from http://home.trainingpeaks.com/products-desktop/wko.aspx [Accessed 6th January 2013]

[64] Ubisense. 2013. Available from http://www.ubisense.net/en/ [Accessed 6th January 2013]

[65] Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S. and Zhang, D. 2004. Semantic Space: An Infrastructure for Smart Spaces. *IEEE Pervasive Computing, Vol 3 Num 3*

[66] Wang, X.H., Zhang, D.Q., Gu, T. and Pung, H.K. 2004. Ontology based context modeling and reasoning using OWL. *Pervasive Computing and Communications Workshops*

[67] Wattbike. 2013. Available from http://wattbike.com/uk/ [Accessed 6th January 2013]

[68] Wojciechowski, M. and Xiong, J. 2008. A user interface level context model for ambient assisted living. *Smart Homes and Health Telematics*

[69] Wojciechowski, M. and Xiong, J. 2008. On Context Modeling in Ambient Assisted Living. *International Workshop Modeling and Reasoning in Context (MRC).*

[70] Wolfram Mathworld: k-means clustering. 2013. Available from http://mathworld.wolfram.com/K-MeansClusteringAlgorithm.html [Accessed 6th January 2013]

[71] Wolfram Mathworld. 2013. Available from http://mathworld.wolfram.com/Distance.html [Accessed 6th January 2013]

[72] Wood, A., Virone, G., Doan, T., Cao, Q., Selavo, L., Wu, Y., Fang, L., He, Z., Lin, S. and Stankovic, J. 2006. ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring. *Technical Report CS-2006-11, Department of Computer Science, University of Virginia.*

[73] XQuery Update Facility. 2011. Available from http://www.w3.org /TR/xquery-update-10/ [Accessed 6th January 2013]

[74] XQuery. 2013. Available from http://www.w3.org/TR/xquery/ [Accessed 6th January 2013]

[75] Wood, A. D., Selavo, L. and Stankovic, J. A. 2008. SenQ: An extensible query system for streaming data in heterogeneous interactive wireless sensor networks. *Distributed Computing in Sensor Systems (DCOSS).*

[76] Yang, J., Zhang, C., Li, X. Huang, Y., Fu, S. and Acevedo, M. F. 2010. Integration of wireless sensor networks in environmental monitoring cyber infrastructure. *Wireless Networks Vol. 16 Issue 4.*

[77] Ye, J., McKeever, S., Coyle, L., Neely, S. and Dobson, S. 2008. Resolving uncertainty in context integration and abstraction: context integration and abstraction. in *International conference on Pervasive services (ICPS).*

[78] Zimmermann, A., Specht, M. and Lorenz, A. 2005. Personalization and Context Management. *User Modeling and User-Adapted Interaction*