Data Transformation and Query Management in Personal Health Sensor Networks

Mark Roantree, Jie Shi, Paolo Cappellari, Martin F. O'Connor

Interoperable Systems Group, School of Computing, Dublin City University

{*mark,jshi,pcappellari,moconnor*}*@computing.dcu.ie*

Michael Whelan, Niall Moyna

School of Health and Human Performance, Dublin City University

{*michael.whelan,niall.moyna*}*@dcu.ie*

Abstract

Sensor technology has been exploited in many application areas ranging from climate monitoring, to traffic management, and healthcare. The role of these sensors is to monitor human beings, the environment or instrumentation and provide continuous streams of information regarding their status or well being. In the case study presented in this work, the network is provided by football teams with sensors generating continuous heart rate values during a number of different sporting activities. In wireless networks such as these, the requirement is for methods of data management and transformation in order to present data in a format suited to high level queries. In effect, what is required is a traditional database-style query interface where domain experts can continue to probe for the answers required in more specialised environments. The challenge arises from the gap that emerges between the low level sensor output and the high level user requirements of the domain experts. This paper describes a process to close this gap by automatically harvesting the raw sensor data and providing semantic enrichment through the addition of context data.

1. Introduction

Sensor networks are being used in increasing numbers to monitor and support various processes and activities. Sensors provide a means of automating monitoring processes as they conduct simple yet specific tasks and report their findings at either fixed or variable intervals. What is consistent is the continuous generation of information these networks provide. On the negative side,

[1]The research described in this paper has been jointly funded by Enterprise Ireland Grants TD-2007-201 and CFTD-2008-231.

these networks provide vast volumes of information that has no structure and little semantics and is thus, very difficult to exploit. In effect, there are no data storage methods and no mechanism for query answering or knowledge extraction. On the positive side, with an appropriate data management layer, this information can be transformed into knowledge, providing input into all forms of decision making through an efficient query answering process.

A recent overview of the Sensor Web [6] highlighted the growth of sensor networks and described research in areas such as sensor development, toolkits and standards, security, ubiquitous sensing systems and wearable sensors. Many of these topics focused on bridging the physical-digital divide and discussed research into areas ranging from environmental monitoring, testing in large scale engine development, detection of hazardous gases in emergency disasters, and personal and wearable sensors.

In the area of personal Health (pHealth) sensor networks, sensors are used to indicate both levels of health and well being and levels of human performance. Unobtrusive sensors are now commonly used to assess the physiological responses during individual and team sports. The measurement of heart rate to assess the physiological load during sporting activities is widely accepted within the sporting community [1, 20]. Ambulatory telemetric equipment such as the wireless Polar Team Heart monitor [19] used in this study has made it possible to innocuously monitor heart rates during team sports. Relative exercise intensity can be estimated [2] by processing and manipulating the output from the heart rate monitors as this is commonly used as a measure of exercise intensity during a game of soccer [1, 20]. This work describes a number of processes that have been developed to transform the low level output from pHealth sensor networks and provide an environment in which exercise scientists can express their needs in the form of complex query and analysis operations.

Paper Structure. The paper is structured as follows: in the remainder of this section, we discuss the background of this research, the research aims, and the contribution; in §2, we describe the hardware environment that is used to generate knowledge and our initial filtering process for removing sensed data that has no value; in §3, we present a process to identify and normalise abnormal sensor data; in §4, we describe how knowledge is extracted from this dataset and transformed for usage by a wider user base; in §5, we present an outline of our framework and metadata service that we used to manage the data transformations; in §6, we provide an overview of some of the query needs of users, how they are expressed in a standard query language, and the times needed to generate the results; in §7, we discuss some related research and finally, in §8, we provide conclusions.

1.1. Background and Motivation

Gaelic football [8, 21] is the most popular sport in Ireland. It is a hybrid of Rugby and Australian Rules football. This project assessed heart rate responses during small sided and regular Gaelic football games in young players. To achieve this, we created a wireless sensor network that has multiple configura-

Table 1: Heart Rate Training Zones

Perc.	Zone	Description	Typical Range
Rest to 60%	Resting	Walking Pace	RHR to 120
60%-70%	Recovery	Develops basic endurance and aerobic capacity.	120 to 140
70%-80%	Aerobic	Develops the cardiovascular system.	140 to 160
80%-90%	Anaerobic	Develops the lactic acid system.	160 to 180
90%-100%	Maximal	Training in this zone is optimal for development of players' aerobic capacity (but is possible only for short periods).	180 to 200

tions and requires a sophisticated data management layer to process, normalise and query the data streams.

Heart rate monitoring with telemetry equipment is commonly used to assess and monitor the physiological responses during team sports involving intermittent activity. A wireless heart rate monitor consists of a transmitter attached to a belt worn around the chest which transmits automatically to a receiver base station. This electromagnetic signal contains heart rate data and is generated for as long as the heart monitor is worn. In Table 1, we show the heart rate training zones, which range from Resting to Maximal. Coaches and exercise scientists seek to accurately determine the intensity at which each player is working while on the training field or where possible, in a competitive environment. To do this, a calculation of each player's maximum heart rate (MHR) and resting heart rate (RHR) is calculated. Later in this section, we will describe how this information was gathered for the set of experiments on which this paper is based.

1.2. Research Aims

The focus of this research is to monitor the effect of high intensity sporting activity on a group of 14 year old boys. The aim is to determine if the maximal heart rate is attained by all players while varying two parameters: the size of the playing area and the number of players per team. The challenge is how to capture and make sense of the large volumes of data that will be gathered during each experiment. This paper describes a collaboration between a team of computer scientists and exercise scientists. Written informal consent and assent were obtained from the parents/guardians and children respectively and the study was approved by the Research Ethics Committee at Dublin City University.

As heart rate data indicates the amount of time spent in different intensity zones (Table 1), sensors to detect heart rate during game time were used. A player's heart rate was measured every 5 seconds during the different activities. A major problem for coaches and sport scientists is how to effectively interrogate the large databases generated during match play. For example, there are approximately 10,800 heart rate values generated by a team of 15 players during a 60 minute game of Gaelic football. This presents a major challenge to exercise scientists as they seek to determine the intensity for each individual or across teams, per activity, at specified times or for specified intervals. One of the goals was to the develop a fast and accurate measurement of the amount of time spent in each training zone as this is an important factor in determining the primary energy source utilised during games and training. It also plays a vital role in allowing coaches to develop an individual physiological profile for each player and thus, personalised training programmes.

The requirement for an infrastructure to monitor and optimise players' performances led to the development of a wireless sensor network that was configured for each experiment as described in §2. While the network provided the participants and hardware, it was then necessary to develop the data management layer in order to process and calibrate data generated by the networks. The motivation was firstly, to provide a traditional query interface for the low level data generated by the wireless network and secondly, to process the data streams to enable an easier expression of queries. As such, the goal from the computer scientists' perspective was to develop a method to automatically harvest and transform or enrich low level sensor data.

1.3. Contribution

In earlier work [23], we described how we deployed the different sensor networks and created common wrapper algorithms to provide the necessary context to interpret different activities and groups. This process was automated to allow simple queries to be expressed using a standard query language. XQuery was the language of choice as our approach wrapped the sensor output in XML and stored the output directly into the MonetDB XML database. What was missing from this work was the transformation of sensor data so that it could be interpreted and manipulated by a wider target group. In this paper, we complete this transformation process to make knowledge extraction by the masses possible. Specifically, our contribution can be summarised as follows:

- The development of a new process for identifying and eliminating outlier values generated by the sensors;
- A process of automatically transforming data generated by sensors into a schema-based format and thus, providing an interface for a wider user group;
- The development of a metamodel and metadata service to make our research more applicable to a wider user base as it facilitates the customisation of the transformation process.

2. Managing Data in Sensor Networks

This section contains a description of the wireless sensor network configuration and the experimental scenarios in which it operates. We will also outline the filtering process by which we remove sensor readings that lie outside the activity periods to be analysed by the sports scientists.

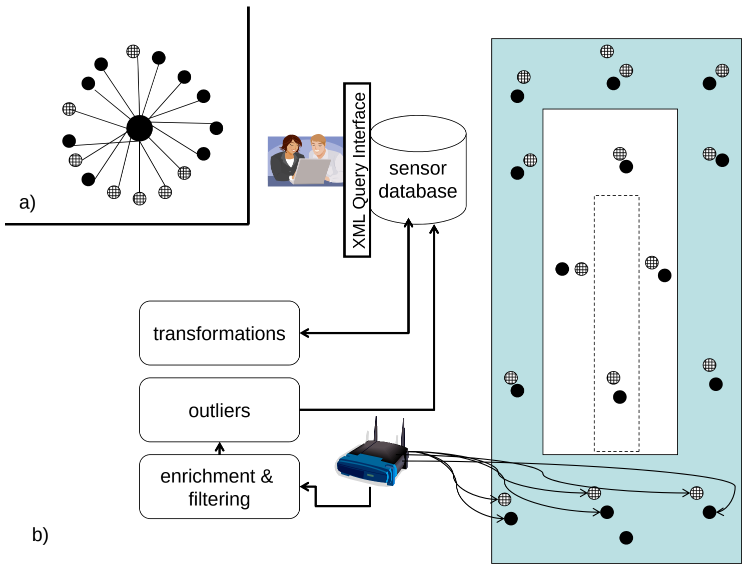


Figure 1: Network Configurations and Data Management Processes

2.1. Wireless Sensor Network Configuration

Four teams took part and the study involved participation in a series of: three 15-a-side games, played on a full size adult pitch under official adult rules; three 9-a-side games played on a modified size pitch (100m x 72m); and three other 9-a-side games in which the pitch size was further reduced in size (90m x 60m). Both 9-a-side games were played under official adult rules. A unique feature of the study is the fact that the surface area available to each player was identical in the 15-a-side games and the 9-a-side games played on the 100m x 72m pitch. Finally, an endurance test was used to determine players' maximum heart rates. Different configurations of the sensor network can be described as follows:

- 15-a-side games: Each team consists of one goalkeeper, six defenders, two midfielders and six attackers. The wireless sensor network will contain 30 nodes (2 * 15 players).
- 9-a-side games: Each team consists of one goalkeeper, three defenders, two midfielders and three attackers. The wireless sensor network will contain 18 nodes (2 * 9 players).
- Endurance test: The endurance test evaluates an individual's aerobic endurance fitness. Up to 50 participants take part in an activity which progresses with levels of increasing intensity until they drop out due to fatigue.

In figure 1(a), we illustrate the network topology used for the experiments. Sensed data is transmitted to the base station through the heart rate monitors, for every node (participant) in the network. The conceptual model for the sensor network is shown in figure 1(b), where we expect nodes to be in certain physical areas.

Each player straps on the wireless device shortly before the activity begins and removes them shortly after the activity ends. However, the devices are not synchronised and not all players strap on the devices at the same time. Furthermore, a game of Irish Gaelic football is played over two halves and sports scientists require heart rate values for the periods corresponding to the first and second half only. Extra sensor readings from either side of these periods are in effect *noise* and serve only as a corrupting influence on the integrity of results returned by queries processing the data. This requirement motivates the need for a filtering process whereby we *filter out* and eliminate all sensor readings that lie outside the first and second half periods. Before filtering can begin we need to structurally enrich the sensor stream into XML format.

Example 1. *An Enriched Sensor Stream*

```
<user>Kelly</user>
<session>080709</session>
<sessiontype>Under 14</sessiontype>
<sensorData candidate="candidate">
    <device>HRM</device>
    <startTime>1215627214000</startTime>
    <interval>5000</interval>
    <sections>
        <section name="Params">
            <parameter>
              <key>Version</key>
              <value>106</value>
            </parameter>
            ...  (parameter element repeats)
        </section>
        <section name="HRData">
            <measurement offset="0" state="" stateoffset="" time="1215627214000">
                <reading ordinal="">
                    <key>HeartRate</key>
                    <raw-value>91</raw-value>
                    <outlier-value>91</outlier-value>
                    <padded-value>91</padded-value>
                    <value>80</value>
                    <averages>
                        <average>
                            <time/>
                            <value/>
                        </average>
                    </averages>
                </reading>
            </measurement>
            ... (measurement element repeats)
```

2.2. Structural Enrichment

The purpose of the structural enrichment process is to convert the raw sensor data into XML format, providing both structure and meaningful semantics. The semi-structured XML format permits the use of a high level query language

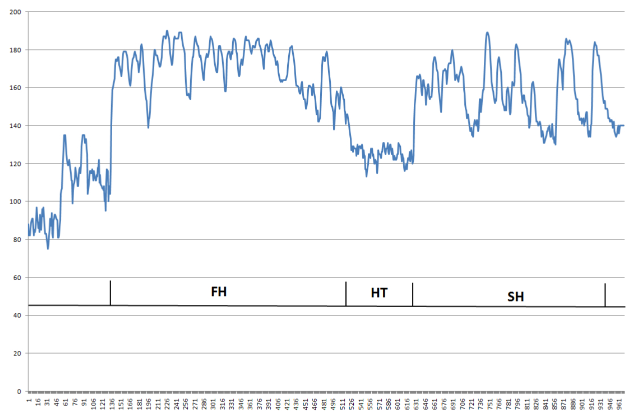


Figure 2: Player Sensor Data Stream

rather than requiring low level primitives to be written every time the user modifies or has a new query requirement. A template file (an XML schema document) facilitating the transformation of the raw data into XML is provided for each sensor device. We described an architecture where all sensor streams use an XML template to make themselves readable and queryable by the system in our previous work [13]. The benefit of this modular approach is that the system requires no modification when new sensors are added to the wireless network. The enriched XML file contains a header section detailing the user information, session parameters that describe the current experiment or activity, and sensor device ID information. The body of the XML file contains the readings recorded by the sensor device.

In Example 1, we present a small extract from an enriched sensor stream containing some header data, followed by a stream of heart rate values and time stamps. Meaningful queries (see Tables 3 and 6 for different queries expressed by the exercise scientists) are only made possible by the addition of the descriptive attributes from the sensors' template file. In addition to structural markup, semantics such as outlier information, rolling averages and athlete details are also included to enable more complex queries.

2.3. Filtering Process

The purpose of the filtering process is to remove all heart rate values that are not part of the core sporting activity as the inclusion of these values generates erroneous results. In the case of football matches, algorithms seek to identify the first or second half periods. A 15-a-side game consists of two halves, each with a duration of 30 minutes; whereas each half in a 9-a-side game has a duration of 15 minutes. The following description of the filtering process is for a 15-a-side game, however, it may be parametrised and applied to a 9-a-side game by changing the appropriate values.

When dealing with match activities, the exercise specialists requested the identification of two 30-minute periods of maximal activity immediately before and after the half-time interval. All other values should be eliminated for the purpose of these experiments. Thus, we defined three fixed functions. The functions `T(FH) = 1800` and `T(SH) = 1800` (30 * 60 seconds) correspond to the first and second halves respectively. The half-time interval (12 minutes) is defined as `T(HT) = 720`.

The filtering process comprises three principal algorithms. The first `detectHT` identifies the start and end times of the half-time interval. This algorithm is key to the entire filtering process as it is used to effectively divide and reduce the search space into two smaller sections. The algorithms `detechFH` and `detectSH` will process and identify the first and second half periods in the first and second sections respectively.

The `detectHT` algorithm begins processing from the sensor reading at point [`endOfStream - T(Init)`] in the sensor stream until the end of the stream and calculates all candidate Half Times by computing 12-minute rolling averages for every 5 second interval. `T(Init)` is calculated as the sum of the length of the second half period, `T(SH)` and the length of half time period, `T(HT)` and a surplus period, `T(Surplus)`. The purpose of the surplus period is to ensure we have processed a sufficient segment of the data stream. The algorithm identifies the smallest 12 minutes rolling average as the end of the half time period and designates the time associated with this reading as the end time of the half time period. The start of the half time period is obtained by subtracting the duration of half time (12 minutes). An illustration of a typical player heart rate stream over a full game is presented in Fig 2.

Algorithm 1: The `detectHT` algorithm

```
Input: An XML file corresponding to the sensor stream of one participant
Output: The end time of the half time period
Start from End(Stream)
Calculate all 12 minute average AVG($i$) for T(SH)+T(HT)+T(Surplus)
foreach AVG(i) do
    if $AVG(i) < SmallestAvg$ then
        $SmallestAvg$ = AVG($i$);
        // locate the smallest 12 minute rolling average.
Return Time($SmallestAvg$);
```

The `detectSH` algorithm reads from the end of half time plus `T(SH)` until the end of the sensor stream and generates each 30 minute rolling average of heart rate values. The algorithm identifies the largest 30 minute rolling average and designates the time associated with this reading as the end time of the second half period. The start time of the second half period is obtained by subtracting the duration of the second half (30 minutes). Although the end of half time and the start of the second half period should be the same time in theory, in our experiments this was often not the case. This anomaly is a positive finding because our algorithm is designed to return the 30 minute period of maximal activity after the half time period.

Algorithm 2: The `detectSH` algorithm

```
Input: An XML sensor stream of one participant after the half time period
Output: The end time of the second half period
Start from End(HT) + T(SH)
Calculate all 30 minute average AVG($i$) until end of stream
foreach $AVG(i)$ do
    if $AVG(i) > LargestAvg$ then
        $LargestAvg$ = AVG($i$);
        // locate the largest 30 minute rolling average.
Return Time($LargestAvg$);
```

A similar approach is adopted for the `detectFH` algorithm. The filter process for the endurance test is far more complex and shall be omitted from this paper due to space restrictions.

3. A Method for Outlier Removal

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

Step 1. Set Valid Range, MinValue and MaxValue. In heart rate (HR) monitoring, the general rule of thumb for Max HR is $HR_{Limit} - age$ (and generally $HR_{Limit} = 220$). While this is generally applicable, we need to be as flexible as possible as we are dealing with a specific age group and within that age group, there may be wide differences. Here we define the following function to identify the upper and lower bounds of the range containing valid candidate outliers.

$$f_{validRange}(hr, HR_{Limit}, age, variance) = \begin{cases} \textsc{false} & \text{if } (hr > (HR_{Limit} - age) * (1 + variance)) \\ \textsc{false} & \text{if } (hr < (HR_{Limit} - age) * (1 - variance/2)) \\ \textsc{true} & \text{Otherwise} \end{cases}$$

In the case of this experiment we have $age = 14$, thus the probable maximum HR is $ProbableMax = HR_{Limit} - age = 206$. Then we add a 10% variable ($variance = 10\%$) so that anything upto $ProbableMax + 10\%$ is a possible valid MaxHR. Anything above is an Outlier (automatic). Thus: $MaxValue = 206 + 10\% = 227$; $MinValue = 206 - 5\% = 196$. Anything below 196 cannot be an outlier which is important because there are many variances as the heart rate increases swiftly in the early stages of activity.

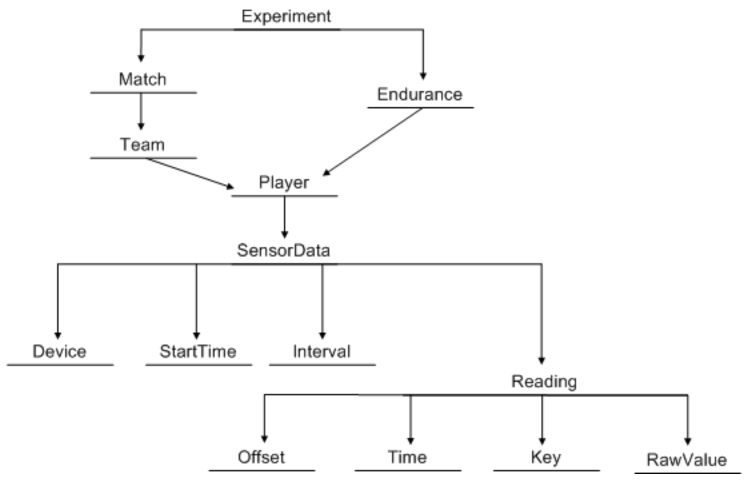


Figure 3: Basic Sensor Schema S1

Step 2. Identify Candidate Outliers. Read through the stream of heart rates from start to finish. We examine all HRs which are within the above defined valid range: anything within this range is a *Candidate Outlier*. In our case, everything within the range 196 to 227.

Step 3. Identify Actual Outlier. Read the 5 HRs before and 5 HRs after the candidate outlier. Calculate the mean of these 10 values (call this $Mean\_Compare$) If the Candidate Outlier is outside 1.5% of $Mean\_Compare$, where 1.5% is a variance parameter, it is deemed an Actual Outlier.

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

4. Data Transformation and Storage Model

In the two previous sections, we described how data is harvested from sensor networks and taken from a raw format and enriched both structurally and with some element of context to provide for basic user queries. We also described a generic process for outlier removal. In this section, we discuss the transformation processes necessary to make queries more manageable for domain specialists. In Fig. 3, we can see the XML schema for sensor data that has been structured, filtered and classified. This schema forms the input to all three transformation processes described here. In each case, the result will be an enlarged schema to facilitate knowledge based queries. At the end of this section, we provide details of the times required for processing the experiments used throughout this work.

4.1. Aggregate Reporting

In §2.2, we indicated that the sensor streams are structurally enriched as XML files. These XML files are rich in data (sensor readings) but are devoid of context. In particular, a single heart rate reading by itself does not convey useful information. However, when examined within a wider category of values, heart rate readings provide for invaluable comparative and performance related metrics.

When the filtering processing described in §2.3 has completed, we apply *state* information to each sensor reading and the state information is stored with every sensor reading in the XML file. A state refers to some interval in the sporting activity such as the first half period or second half period in a Gaelic football game. We then mine these XML files to produce aggregate information for each participant in a sporting activity. Depending on the type of activity, we can compute several layers of aggregated information and generate reports that can assist sport scientists to analyse and optimise the performance of their athletes. The following aggregate information may be obtained from a game of Gaelic football.

- PeakHR - The maximum heart rate for a participant during the game;
- HR as % of PeakHR - The heart rate reading as a percentage of the maximal heart rate for a participant during the game;
- The average heart rate of a participant over the entire game.

Table 2: Process Details.

Process	**Input**	**Total in MB**	**Execution Time**	**Result Size**	**Output**
Filter	Sensor Data	1.97 MB	4m 25s 141ms	145 MB	S1
Outlier	Schema S1	145 MB	1m 38s 372ms	276 MB	S2
Aggregate	Schema S2	276 MB	3m 05s 361ms	8.7 MB	S3
PeakHeartRate	Schema S2	276 MB	1m 02s 609ms	840 KB	S4
PeakInterval	Schema S2	276 MB	1m 15s 208ms	862 KB	S5

4.2. PeakHeartRate Transformation

This process performs the PeakHeartRate Transformation of the sensor data stream and is designed according to sports scientists' requirements. According to the user query as "The time length of each player performing at: 0% to 50% of his Peak Heart Rate, 50% to 60% of his Peak Heart Rate, 60% to 70% of his Peak Heart Rate, etc.," the process computes the Peak Heart Rate from each player data stream and calculates the time length of different Peak Heart Rate percentage ranges for each player.

The result is enriched into XML format as shown in *Example 1* and stored in MonetDB on top of previously enriched sensor streams, ready for sports scientists to query using XQuery.

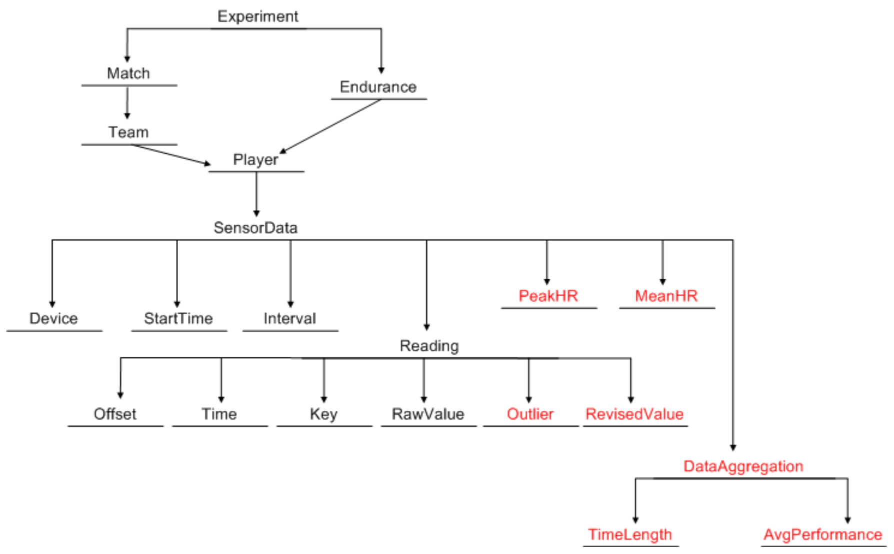


Figure 4: Transformed Schema S2

4.3. PeakInterval Transformation

Similar to the above process, the PeakInterval Transformation process is designed according to the user query as "The occurrence of each player performing at: 70% or above of his Peak Heart Rate lasting more than 10 seconds, 80% or above of his Peak Heart Rate lasting more than 10 seconds, 90% or above of his Peak Heart Rate lasting more than 10 seconds, 70% or above of his Peak Heart Rate lasting more than 20 seconds, etc.," then it computes the different PeakInterval levels for each player as required.

The transformation of data can now answer more complex queries from sports scientist as outlined in §6.

4.4. Process Evaluation

At this point, we reflect on the times required to process the raw sensor data and compute the various transformations. After enrichment, data is stored in the MonetDB XQuery server [15] where it is then calibrated until ready for user queries.

- **Filtering.** Harvesting and filtering takes four and half minutes to complete all sensor streams simply because it structurally enriches all the raw HR data into XML format with total outputs as large as 145 MB and stores them in MonetDB.

- **Outlier Detection and Removal.** This process is designed to calibrate outlier values, and takes approximately one minute and forty seconds for all of the data in this experiment.

- **Aggregate Reporting.** It takes a little longer to run the Aggregate Reporting Process, approximately 3 minutes, because it computes over several ranges and extracts the Activity Period for each individual player as well as for each team, then calculates and prints out each report for all sensor data, the total result size is about 8.7 MB.
- **PeakHeartRate and PeakInterval Transformations.** These two processes both require in excess of 1 minute to generate results. This is due to the fact that it must compute all the PeakHeartRate levels and intervals for the whole experiment.

Table 2 provides the full details for each process in terms of: Data Input; Total Size in MB; the Execution Time; Result Size and Data Output.

5. Metadata Service

The result of the transformations described in the previous section have effectively created a form of data warehouse where the same data subjects are used in different result materialisations. For domain specialists to understand the contents of each transformation and be able to express queries on these datasets, there must be a more abstract description of each dataset. In [9], authors build an ontology to provide context for location-based services such as the smart home. However, their ontology is tightly coupled with the smart home domain whereas our efforts focus on a more abstract sensor network. The role of the Metadata Service is to both manage these transformations and to facilitate configuration to ensure the correct linkage across the datasets. Furthermore, we must allow for a high level of genericity to make this service applicable across sensor networks. In this section, we provide only a brief overview of this service.

At an abstract level, a sensor will be deployed in a specific **Context** and in association with a specific **Activity**. A **Context** will always have one or more of *State* (the phase of the activity during which the sensor value was generated, or is relevant to), *Zone* (the location in which the sensor generated output), and *Timing* (information on the time at which an activity commenced and the interval at which sensor readings are generated). In prior work [24], we demonstrated how *Zone* information provided the necessary context for ubiquitous applications. In the work presented here, we focus more on the *State* (for example FH, HT, SH states are discussed earlier) and *Timing* information (heart rates are generated every 5 seconds). While **Context** is more often associated with an **Activity**, we associate **Context** with **Sensor** (in this case a HRM Sensor) as there may be multiple sensors with different contexts for later experiments (for example in future work, we correlate the readings from a heart rate monitor with one which monitors the levels of sweat produced by athletes).

5.1. Metadata Constructs

There are four primary objects that are consistent to all sensor networks: `Sensor`, `Subject`, `Activity` and `Context`. In our system, we include a fifth object, `Template`, that is used to insulate the system from new sensors being added

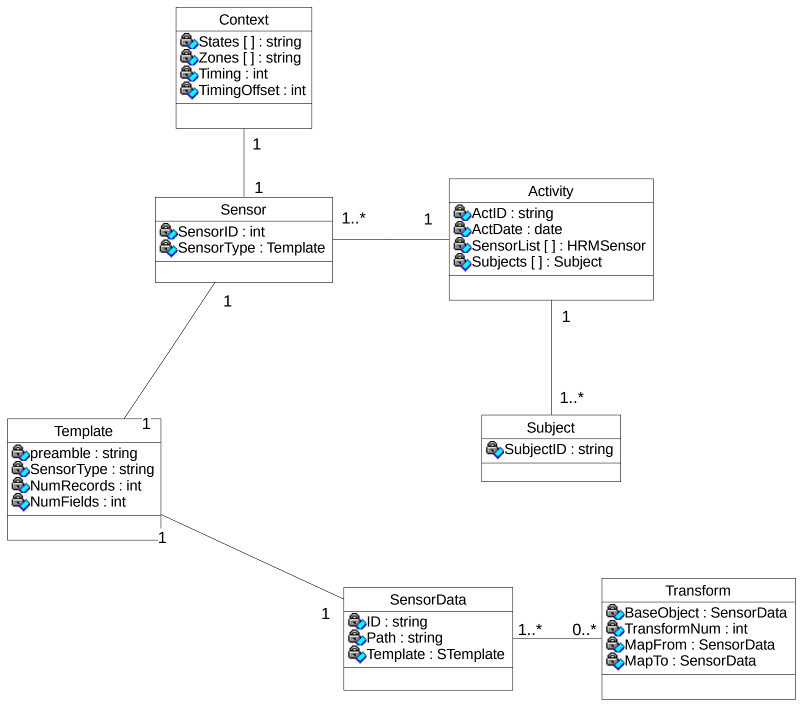


Figure 5: Metamodel Components

or a change in the way sensors create their output. This was presented in earlier work [13] where we described how XML template descriptions could be exploited through a generic interface to interpret any sensor device. When data generated by these networks is processed, we have two other objects: `SensorData` and (data) `Transform`. As will be shown in §6, transformations allow far more powerful queries to be expressed on the datasets. While the processors are part of the system (shown in Fig. 1), the mappings between the low level sensor readings and their transformed aggregations are captured by the metamodel.

- **Sensor.** The sensor produces output that must be interpreted and structurally enriched to a format suitable for queries. As a result, this sensor output is received by the system in the form of XML. As new sensors are added or modified, a template to interpret the sensed data is required, but without changing the system. A more detailed description of this process can be found in [22].

- **Subject.** Sensor(s) are generally associated with a single subject. This subject may be a piece of instrumentation, a person, or some aspect of the environment. This is necessary as the subject will provide a significant contribution to the contextual information required to make assumptions about, or interpret sensor data.

- **Context.** Sensor output has little meaning unless it can be used in a specific context which provides additional semantics to the data generated by the sensor. The metadata service allows end users to specify a number of different contextual parameters including States to be associated with all sensed data.

- **Activity.** Both `Subject` and `Activity` form the contextual input for interpreting sensor data. For example, a heart rate monitor will generate data in a uniform, easy to read, format. However, the context in which that sensor is used is crucial to query processing. In this work, `Activity` can be either 9-a-side matches, 15-a-side matches, or different forms of endurance testing.

- **Template.** Irrespective of the context in which a sensor may be used, the sensing device will generate its output in some prescribed format. The benefit of removing the logic that interprets sensor output from algorithms and describing the output in a template (managed by the metadata service) is that changes to sensor output do not require a system change.

- **SensorData.** When structurally enriched, the sensor stream is stored in XML format with an unique ID, XML Path, and mapping to its position in a Schema (the schema for this research project is shown in Fig. 3). This is the base form of the sensor data to which all transformations are mapped. Each SensorData instance may be used in any number of transformations.

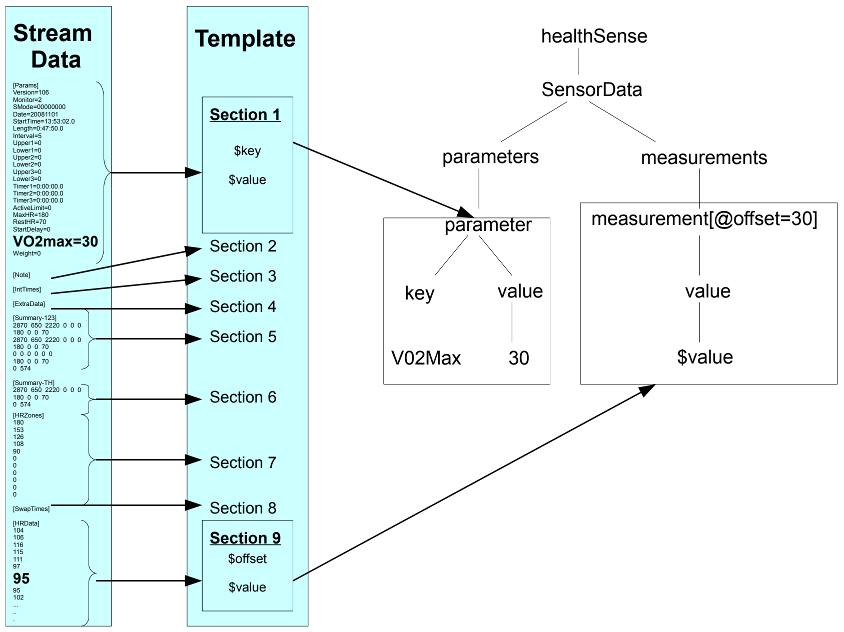
- **Transform.** Each transformation will comprise a series of SensorData instances. They may be part of the basic schema (see Fig. 3) or from an existing transformation (see Fig. 4).

5.2. Services

There are a number of services that are worthy of discussion in the context of this work.

- `Add New Sensor.` When domain users need to introduce a new sensor into an experiment or activity, they simply provide a template file and the system can read the sensor data. This will also create the new **Template** object necessary to interpret the sensor stream. In these experiments, it was necessary to add a Polar HRM sensor and the appropriate template and previously in [16], we provided details of how this process is performed. In Fig. 6, the relatively complex nature of the template for heart rate sensors is illustrated.

- `Add New Context.` In the work presented here, we provide a set of *States* {FH,HT,SH}, *TimingOffset* is set to 0, and *Timing* is set to 5. In many experiments, we use relative timing to indicate the start of an experiment but in some cases, we use the real time as indicated on the sensor. In the latter case, the *TimingOffset* parameter is used when we wish to start processing data on or after a specific time.



```
[Params]
Version=106
Monitor=2
SMode=00000000
Date=20081101
StartTime=13:53:02.0
Length=0:47:50.0
Interval=5
Upper1=0
Lower1=0
Upper2=0
Lower2=0
Upper3=0
Lower3=0
Timer1=0:00:00.0
Timer2=0:00:00.0
Timer3=0:00:00.0
ActiveLimit=0
MaxHR=180
RestHR=70
StartDelay=0
VO2max=30
Weight=0

[Note]

[IntTimes]

[ExtraData]

[Summary-123]
2870  650  2220  0  0  0
180  0  0  70
2870  650  2220  0  0  0
180  0  0  70
0  0  0  0  0  0
180  0  0  70
0  574

[Summary-TH]
2870  650  2220  0  0  0
180  0  0  70
0  574

[HRZones]
180
153
126
108
90
0
0
0
0
0
0

[SwapTimes]

[HRData]
104
106
116
115
111
97
95
95
102
...
..
.
```

Figure 6: Output Generated by Heart Rate Sensors

- `Apply Activity.` When a new sensor is introduced, the template file facilitates reading and processing the output. However, the activity provides the context that determines those values that are of no use, of high importance, that should map to a particular state, and so on. The role of the `Activity` object is to map each value to a particular state.

- `Add New Transformation.` There are three transformations used in this project, all transforming from the basic schema to provide enhanced versions of the schema. The `Transform` object is used to provide a mapping between the two schema versions.

6. Experiments and Results

In this section, we report on the experiments we developed to evaluate our system. The main validation came from the exercise scientists with a simply criteria: can the system automate contextual enrichment of sensor data and can it meet our information needs? These needs were expressed in the queries in Tables 3 and 5 with the remaining queries requiring only parameterized changes to the query set used in our experiments. Furthermore, as the sensor repository is stored in the MonetDB XML database which has a full XQuery interface, the query possibilities are broader than those listed in this paper. However, we created this experiment both to confirm that the overhead introduced by XML would not affect usability (in terms of query times) and to identify any queries which exceeded our internally chosen threshold of 1000ms. We now describe these experiments and provide details of query expressions and the times taken to compute the result sets.

The initial set of queries can be expressed before the development of the transformation layers and demonstrate the complex XQuery expressions required to satisfy relatively simple user requirements. The benefit of this approach is the provision of a standard query interface for low level or raw sensor data. After the development of the metadata service and transformation layer, it is possible to facilitate far more complex queries while reducing the complexity of the XQuery expressions.

Table 3: Standard Query Set and Response Times.

	Query	**Time**
1	Return all heart rate values for Player 1 during the first half	92 ms
2	Return the Avg HR of player 9 in first half	140 ms
3	Return the Max HR of player 9 in first half	113 ms
4	Return the Avg HR of Team CastleBlayney in same match	202 ms
5	Return Avg HR value for each CastleBlayney defender in selected game	593 ms
6	Return Max HR value for each CastleBlayney defender in selected game	566 ms

6.1. Standard Query Set

Table 3 shows a set of basic queries, in English, along with their execution time. These queries have been provided by domain experts and represent a real case scenario of the information health scientists are interested in retrieving from a game. In particular, these queries have been executed on the sensor data generated during a 15-a-side game. Sensors recorded data over a period of, roughly, 100 minutes, and were configured to read heart rate values at intervals of 5 seconds: in this configuration each sensor generated an average of 1200 readings. Response time for each query reported in Table 3 includes the time required to execute the query and return the result to the users.

In their English formulation, queries sound rather simple. However, it must be noticed that executing these queries on raw data streams is complex. Without the data management layer we provide, each query would be implemented in an ad-hoc program that: identify the stream (or the streams, possibly among many) from which to collect the values , enrich such values with context information (phase of the game, teams, etc.), implement the primitives to group sequences and compute aggregation functions (average, min, max, etc.). The provision of our data management layer free users from having to develop an ad-hoc program for each query, enabling the use of a standard query language (XQuery) to retrieve the data of interest.

Table 4 shows the queries from Table 3 in their XQuery formulation. While this syntax may still be not familiar to a non-IT specialist, the efforts required to specify queries in this format is dramatically lower than that of having to design and implement some ad-hoc program for each of the query of interest.

6.2. Evaluating Standard Queries

Table 4: XQuery Formulation for the Standard Query Set

	Query	Time
1	let $c := collection(’080517CS_15AS’) return $c/healthSense[user[text ()=’mbrenn’]]/sensorData/sections/section[@name=’HRData’]/measurement /reading[key[text()=’HeartRate’]]/value/text())	92 ms
2	let $c := collection(’080517CS_15AS’) return fn:avg($c/healthSense [user[text()=’cmcnal’]]/sensorData/sections/section[@name=’HRData’] /measurement[@offset>=3600000]/reading[key[text()=’HeartRate’]]/value/text())	140 ms
3	let $c := collection(’080517CS_15AS’) return fn:max($c/healthSense [user[text()=’cmcnal’]]/sensorData/sections/section[@name=’HRData’] /measurement[@offset>=3600000]/reading[key[text()=’HeartRate’]]/value/text())	113 ms
4	fn:avg(let $c := collection(’080517CS_15AS’) for $p in $c//healthSense, $q in $c//Players/Player where $p/user = $q/Name/Code order by $q/@Id return if $q/TeamId=”CB” then $p//measurement/reading[key[text ()=’HeartRate’]]/value/text() else())	202 ms
5	let $c := collection(’080517CS_15AS’) for $p in $c//healthSense, $q in $c//Players/Player,$r in $c//Players where $p/user =$q/Name/Code and $p/session = $r/Game/Id order by $q/@Id return if (($q/Position/Role=”DE”) and ($r/Game/Type=”15aside”) and ($q/TeamId=”CB”)) then fn:avg($p// measurement/reading[key[text()=’HeartRate’]]/value/text()) else())	593 ms
6	let $c := collection(’080517CS_15AS’) for $p in $c//healthSense, $q in $c//Players/Player,$r in $c//Players where $p/user =$q/Name/Code and $p/session = $r/Game/Id order by $q/@Id return if (($q/Position/Role=”DE”) and ($r/Game/Type=”15aside”) and ($q/TeamId=”CB”)) then fn:max($p// measurement/reading[key[text()=’HeartRate’]]/value/text()) else())	566 ms

Let us comment on the response times in Table 4. Query 1 and 3 are executed in about 100ms, while query 2 in 140ms. These three queries have rather fast execution time because they select values belonging to a single athlete, thus are marginally, if at all, affected by the size of the dataset.

Query 4 requires more time because it has to access the data of 15 athletes belonging to the same team, and calculate the average hear rate. Execution time for this query is likely to increase with the size of the dataset, as more match data is added. Currently, our dataset includes data from 200 games, 80 of which are from 15-a-side matches.

Results for the last two queries in Table 4 require slightly more than a half of a second to deliver their results, the reason being that they have to compute either averages or maximums for all athletes in a given team.

As a final comment, let us remark on two main advantages our data management layer provides: queries can specify a high level of detail, queries are exposed thus are easily editable (a simple text editor will suffice) and customizable.

6.3. Post Transformation Query Set

User queries are listed in Table 5 in plain English with their corresponding XQuery expressions shown in Table 6. A full description of the physiological value of these queries is outside the scope of this paper, but they are all based on the maximal heart rate achieved during each activity (PeakHR). The percentage of times that players were close to this value is of high interest to exercise scientists.

Queries are executed on the extended schema resulting from the transformation processes described in the previous sections: without such transformations these queries are either difficult or impossible to express on the original dataset. In other words, as it was the case for the standard query set, without the data management layer each query would essentially require a complex program, while in our system, thanks to the transformations, we can specify reasonably simple XQuery expressions to answer the users' queries.

Table 5: The List of Post Transformational Queries.

#	**Query in natural language**
1	*How many times was a specific player at 70% Max HR for more than 20s?*
2	*How many times was each player at 70% Max HR for more than 20s across all games?*
3	*How many players across all 9-a-side games were at 90% Max HR for 30 seconds?*
4	*How many players across all 15-a-side matches spent more than 10% of playing time above 80% of their Max HR?*
5	*What was the average time spent in seconds, per player, below 50% of their Max HR across all 15-a-side games?*

6.4. Evaluating Queries on Transformed Data

Execution times are reported in Table 6: times are average values measured over 10 runs on each query. By analysing the results, we can see all are within acceptable response times. The most complex case, query 5, must build results for roughly 300 players and generates the result in slightly over 100ms. All the other queries have execution times below 100ms. By analysing queries 1 and 2, it is interesting to note how establishing a connection with the database and building the resultset have an impact on the overall time. Query 1 performs 4 times faster than query 2. Query 1 and 2 both ask for occurrences of players performing at 70% Max HR for more than 20 seconds, where the difference is

Table 6: Full Query Expressions.

#	Query as XQuery expression	Time
1	doc('dataAggregation.xml')/healthSense/user[text()='edoher'] /sensorData/dataAggregations/peakInterval[avgPerformance=70 and timeLength>20]/timesAboveAvgPerf/text()	26 ms
2	doc('dataAggregation.xml')/healthSense/user/sensorData /dataAggregations/peakInterval[avgPerformance=70 and timeLength>20]/timesAboveAvgPerf/text()	97 ms
3	for $c in doc('dataAggregation.xml') return fn:count($c/healthSense [session[text()='9aside']]/user/sensorData/dataAggregations/ peakInterval[avgPerformance=90 and timeLength=30])	75 ms
4	for $c in doc('dataAggregation.xml') return fn:count($c/healthSense [session[text()='15aside']]/user/sensorData/dataAggregations /peakLength[gameLengthPercentage>10 and avgPerformance>80])	69 ms
5	for $c in doc('dataAggregation.xml') return fn:count($c/healthSense [session[text()='15aside']]/user/sensorData/dataAggregations /peakLength[avgPerformance<50]/gameLengthTime/text())	127 ms

that Query 1 retrieves the values for a specific (single) player in all the games and Query 2 retrieves the same information for each player (a total of, roughly 300) in all the games.

Before concluding this section, let us emphasise the fact that because queries are specified as standard XQuery expressions, they can be modified using a simple text editor for further interrogation, not requiring an alteration to any part of the system.

7. Related Research

In [5], the authors present an approach to address contextual synthesis of sensor networks in the sports domain. Sensor data, generated from a single sporting event, are kept in their proprietary raw format. Queries are developed as ad-hoc programs over the proprietary data format, they are not SQL-like, but instead they are object-oriented, containing context operators which perform synthesising operations. Context operators can in turn use other simpler operators to execute smaller tasks and to reuse existing functionality. While the experimental evaluation of the prototype shows fast response times for built-in queries, this work does not support any open query language but requires pre-composed built-in queries: every time a query needs to be added or edited there is the need for a system expert to apply the changes. Their work enables multiple applications or even different context-aware systems to use the same operators designed for a specific domain without being concerned about their implementation. In our approach, we expose data in a standard XML format which can be queried by the XQuery language and we enrich the data in order to answer complex queries.

In [26], the authors process and query streams of raw data as it arrives from the sensors. Their approach enriches the raw data into "semantic streams" and processes the streams as they are generated. The "semantic streams" permit the user to issue queries over semantic values directly without addressing which data or operations are to be used. This work is still theoretical and has yet to provide experiments or an indication of query performance. In [12], the authors also process sensor data in its raw format. They employ the concept of proximity queries where network nodes monitor and record "local" interesting events. They introduce proximity queries as a means of detecting interesting events that are observed by nodes in the network that are within certain distance of each other. While their results are positive in terms of cost, queries are still at a relatively low level (no common format for query expression), and it is difficult to see how this type of proximity network can be applied in general terms due to the complexity of the technologies involved.

There has been limited research efforts addressing outlier detection in the context of sensor networks. Palpanas et al. [18] have proposed an in-network approach for distributed online deviation detection for streaming data. They are only interested in finding those values that deviate significantly from the norm. Their detection mechanism can be used to identify faulty sensors, and to filter spurious reports from different sensors. However, this approach depends on the existence of more powerful and sophisticated sensors (high capacity sensors) to perform the outlier detection and to manage groups of low capacity sensors. Their more recent work [25] uses an online outlier detection scheme for sensor networks. Their approach initially estimates the sensor data distributions, then compute the density of the data space around each value, and therefore determine which values are outliers. The key point is every sensor keeps a sliding window of the historical data and estimates the data distribution to detect the outliers. This method, however, consumes a lot of memory and may not find all outliers. Our approach identifies candidate outliers only after the valid range is selected. It requires a small memory footprint and facilitates the calibration of actual outliers in a deterministic manner. A similar approach for outlier detection in streaming data is described by Kenji et al. [11]. In contrast to Palpanas's work [18], their method does not operate on sliding windows, but rather on the entire history of the data values, using an exponential forgetting factor for discounting the effect of the older values. Furthermore, the above approach is not geared towards a distributed environment, such as a sensor network.

Research efforts on early tiny database systems, such as TinyDB [14] and Cougar [27] have shown that a declarative approach can provide a powerful and easy to use interface for collecting data from static sensor networks. These early systems, however, have significant limitations. In particular, they require the end user to understand the operators running over the raw sensor data and interpret the meaning of the results. In our case, the automated transformations processes allow domain experts to express far simpler queries using the XQuery language.

In both [3] and [4], the authors examine how Sensor Web enablement services work in healthcare sensor networks. They offer a standardised protocol for discovering and accessing sensor data which enables data to be reused in potentially new and novel ways, and also enables sensor data to be vitalised, providing a common, self-describing data format and access protocol. Both approaches enrich multiple sources of sensor data into a data model that represents different sensors and data as a series of observations. The approaches focus on simple live sensor data from multiple sources, obtaining query results on the basis of a series of simple rules applied to the streams. However, our goal is to facilitate more complex queries by non-IT domain experts.

Authors in [7] implemented a working process integrating multiple databases into a data warehouse and in line with our strategy, a small part of the warehouse data is then extracted to answer user queries, depending on contextual needs. They are only concerned with the integration of experimental data from domain databases and the interactive manipulation of the heterogeneous data that have been integrated. Our approach differs in that we are using sensor networks and we enrich and make usable the low level data emerging from these networks.

8. Conclusions

Sensor networks provide both solutions and problems for experts across a range of different domains. In this paper, we focused on how exercise scientists developed a series of personal health sensor networks to monitor the physical effects of different sporting activities on young children. The benefit provided by the sensor network was in facilitating the automatic generation and harvesting of the heart rates of the individuals involved. However, the development of a sensor network does not in itself provide for any form of data storage methods, efficient query answering or the extraction or management of knowledge. Before the collaboration with data management specialists, data was captured in spreadsheets, one per sensor, with no clear understanding of what each heart value represented or how meaningful analysis could take place. In the work presented here, the goal was to provide a framework and services to enable the domain specials to generate the knowledge themselves, and thus, empower this wider user base with knowledge management capabilities.

We presented the data management layers that take the raw sensor data and process the data to a point where simple queries are expressed using a standard query language. The overall process is managed by a metadata service to provide the appropriate level of genericity. For more complex queries, we use the metadata service to associate processors with sensor data to perform different transformations. The metadata service tracks transformations and records the mappings between base values and the transformations in which these base values occur. System performance is demonstrated by publishing times for basic query expressions, transformation processes and the more complex queries that are executed on transformed data. As we cannot expect many domain specialists to be able to express their requirements using XQuery, our current focus is on creating a graphical interface that enables them to construct queries using a graphic of both schema and transformations. We are also continuing work on the Metadata Service to enable new processes to further improve the genericity of the system.

References

[1] Bangsbo J., The Physiology of Soccer - With special reference to Intense intermittent exercise, In *Acta Physioligica Scandinavica*, 151 (supp. 619), 1994.

[2] Bangsbo J. *Fitness Training in Football: A Scientific Approach*, HO+Storm, ISBN 87-983350-7-3, 1994.

[3] Churcher G., Foley J., Bilchev G., et al. Experiences applying Sensor Web Enablement to a practical Telecare application. *International Symposium on Wireless Pervasive Computing (ISWPC)*, 2008.

[4] Churcher, G., and Foley J. Applying and Extending Sensor Web Enablement to a Telecare Sensor network Architecture. *4th International ICST Conference on Communication System software and middleware(ICST)*, 2009.

[5] Devlic A., Koziuk M., and Horsman W. Synthesizing Context for a Sports Domain on a Mobile Device. *3rd European Conference on Smart Sensing and Context (EuroSSC)*, LNCS vol. 5279, pp. 206-219, 2008.

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

[7] Fabin J., Sylvie R., Vincent D., and Michel C. IDEE: An Integrated and Interactive Data Exploration Environment Used for Ontology Design. *15th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks (EKAW)*, pp. 256-271, 2006.

[8] The Gaelic Athletic Association, (`http://gaa.ie`), 2010.

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

[10] Grust T. Accelerating XPath Location Steps. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp.109-120, ACM Press, 2002.

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

[12] Kotidis Y. Processing Proximity Queries in Sensor Networks. *3rd International Workshop on Data Management for Sensor Networks (DMSN)*, pp. 1-6, 2006.

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

[14] Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. *5th Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 131-146, 2002.

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

[16] McCann D. and Roantree M. A Query Service for Raw Sensor Data. In *Proceedings of 4th European Conference on Smart Sensing and Context* (EuroSSC), LNCS vol. 5741, Springer, pp. 38-50, 2009.

[17] Marks G., and Roantree M. Metamodel-Based Optimisation of XPath Queries. *Proceedings of 26th BNCOD*, LNCS vol. 5588, Springer, 2009.

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

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

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

[21] Reilly T., and Doran D. Science and Gaelic Football: A Review. In *Journal of Sports Sciences*, vol. 19, pp 181-193, 2001.

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

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

[24] Shaeib A., Cappellari P. and Roantree M. A Framework for Real-Time Context Provision in Ubiquitous Sensing Environments, In Proceedings of 1st International Workshop on Semantic Interoperability for Smart Spaces (SISS 2010), IEEE Press, 2010.

[25] Subramaniam S., Palpanas T., Papadopoulos D., Kalogeraki V., and Gunopulos D. Online outlier detection in sensor data using non-parametric models. *Proceedings of the 32nd international conference on Very large data bases (VLDB)*, pp 187-198, 2006.

[26] Whitehouse K., Zhao F., and Liu J. Semantic Streams: a Framework for Composable Semantic Interpretation of Sensor Data. *3rd European Workshop on Wireless Sensor Networks (EWSN)*, LNCS 3868, pp 5-20, 2006.

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