



Project no. 224609

Project acronym: DEHEMS

Project title: Digital Environment Home Energy Management System

Instrument: <i>Please tick</i>	CA	STREP <input checked="" type="checkbox"/>	IP	NOE
--	----	---	----	-----

ICT - Information and Communication Technologies Theme

Deliverable reference number and title (as in Annex 1):

D5.4 Sub-System deliverable - Energy pattern analysis engine delivered

Due date of deliverable (as in Annex 1): 30th June 2009

Actual submission date: 23rd July 2009

Start date of project: 1st June 2008

Duration: 30 months

Organisation name of lead contractor for this deliverable: Manchester City Council

Revision [V1.0]

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	<input checked="" type="checkbox"/>
PP	Restricted to other programme participants (including the Commission Services)	<input type="checkbox"/>
RE	Restricted to a group specified by the consortium (including the Commission Services)	<input type="checkbox"/>
CO	Confidential, only for members of the consortium (including the Commission Services)	<input type="checkbox"/>



Document Control

Work Package Leader: Hildebrand Technology Limited

Document Owner: Hildebrand Technology Limited, Joshua Cooper

File Reference: Dehems_Deliverable_5.4.pdf

Date: 17th July 2009

Version: V2.0

Version Control Record

Version	Date	Author	Comments
V1.0	17/07/2009	Joshua Cooper	
V2.0	20/07/2009	Martine Tommis	Formatting and layout

Energy Pattern Analysis Engine

Table of Contents

Executive Summary	4
Background and motivations	5
Architecture	5
Approach	7
Data input.....	7
Feature Extraction.....	8
Classification.....	8
Measures	9
Dimensions	9
Engine Implementation	10
API.....	11
Data input.....	11
Function calls	11
Persistence	12
Experiments.....	12
Area 1. Normalizing data.....	12
Area 2. Detecting waste	12
Area 3. Detecting appliance level events	13
Area 4. Combined long term and short term measures for performance	13
Conclusions	13
Future work	13
Implications for other Work packages	14
References	14
Appendix 1. Sample RSOAP session	15

Executive Summary

The pattern analysis engine is required by the DEHEMS project to provide statistical calculations, feature extraction, time series analysis and general routines for measurement of the sensor data coming from the DEHEMS households. Systems that will use the pattern analysis engine are defined in the overall system architecture with the data warehouse component as a broker to pattern analysis functionality.

As much as possible, general solutions are offered such that the engine can be deployed into other applications in the future, including general sensor processing. Specific application requirements are however address with measurements that will be required for comparisons, normalization, trending and state identification Markov model described in Deliverable 5.1.

Two of the major issues in creating a pattern engine for DEHEMS is the amount of data that potentially be analyzed and the real-time nature of the system. The size of the data could be in the petabyte range over the duration of the project and in keeping with the design goals, all systems should be able to scale to service a community numbering into the millions. Traditional analytic software, such as MATLAB, SAS and SPSS perform analysis on data that is manually extracted from a set rather than as a runtime, dynamic analysis system. These challenges will be addressed in the solution presented.

Background and motivations

The pattern analysis engine supplies the environment for pattern recognition and characterization of data for comparative purposes and the detection of new patterns of usage that result from behaviour change.

The aim is to classify raw data against patterns based either on a priori knowledge or on statistical information extracted from the patterns. The patterns that DEHEMS is interested in are generally groups of measurements or observations from various sensors mapped and compared in a multidimensional space. Patterns are not rigidly matched but have an distance that will determine whether or not the observation space is classified as matching a pattern.

Some patterns are dynamic in that distance will be measured against other observation groups. For instance, a set of household observations may be treated as a pattern to be classified against.

It is the intention to discover new patterns in the data and therefore support for the exploration of data is required. Typically these patterns will be described as statistical correlations between observations; a cause and effect. The effect will be either a positive effect on lower energy consumption or a negative effect on higher energy consumption with the cause attributed to an action inferred from or captured as sensor input.

Architecture

The architecture of the overall system has the pattern analysis engine interfacing with the Data Warehouse element. The Data Warehouse provides provides dimensions and measures over the time series observations and therefore blends the knowledge based systems with the measurement systems. The data warehouse is also the public interface for the pattern analysis engine to be used by other subsystems.

This assumes that the Data Warehouse, or subsystems that it utilizes, has access to current statistics, historical data and aggregates on group levels of interest.

Deliverable 5.2 outlined the APIs that would be available to the pattern analysis engine. These take the form of RESTFUL URLs that return data in delimited column formats. The query for data specifies the sensor identifier, measure and time range. Likewise summary data can be fetched.

Similar APIs are available from the Data Warehouse engine to access household, appliance and appliance type identifiers. Measures or descriptive data may be held within the pattern analysis engine where storage may not be possible within the knowledge based systems.

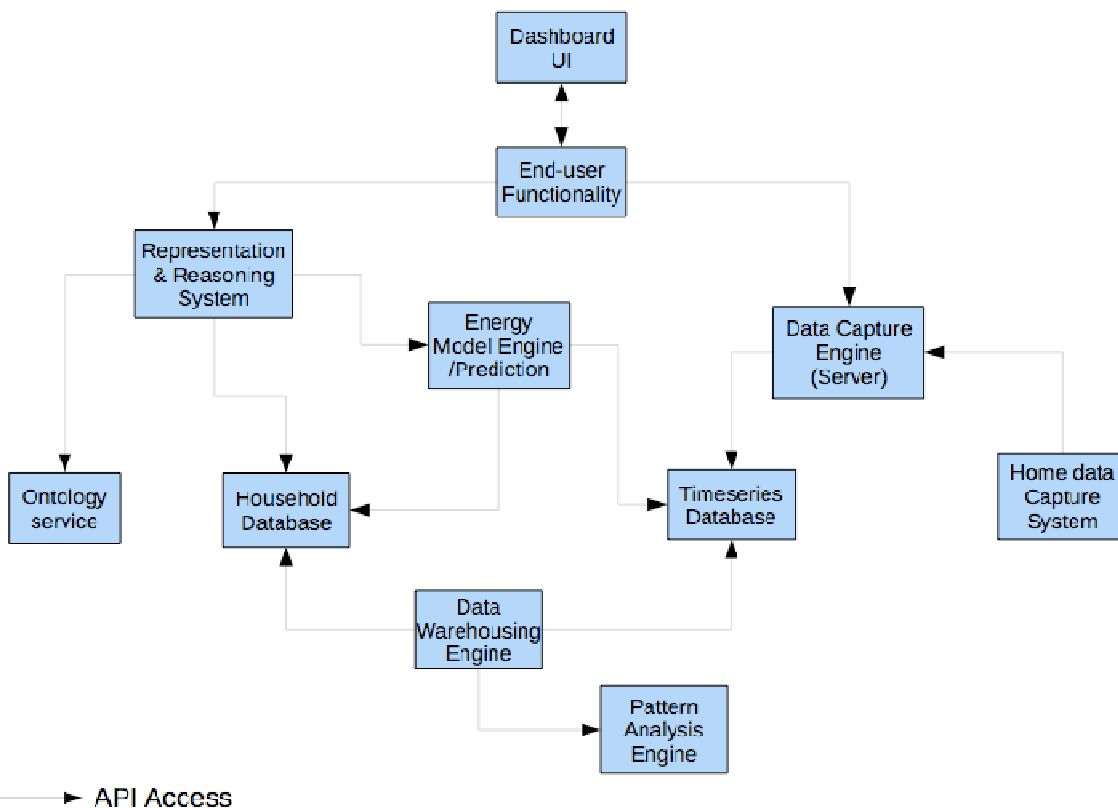


Figure 1. Draft architecture showing the pattern analysis engine.

Approach

There are common elements to pattern analysis systems that generally describe the requirements and approach. Figure 2. shows a functional flow model of the data and how a set of patterns are classified against. Each element will be taken in turn with a description of how they will be approached.

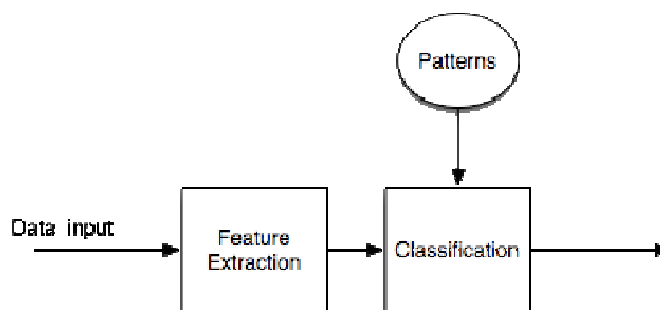


Figure 2. Functional elements of a pattern analysis engine

Data input

Many sensors form input into the DEHEMS system, in order for the pattern analysis engine to work effectively, data from observations needs to be available for feature extraction. In the DEHEMS system, data is temporal and some patterns emerge over long periods. Given that there are a large number of observations over time, the pattern engine must be efficient in overall memory and storage requirements. Specifically,

- data should not be stored separately from the main time series data as this will require infeasible amounts of disk space
-
- observations that are held in memory should be able to be accessed by the pattern analysis engine such that they are not recalculated or become stale

Data input is through the APIs provided by the data storage mechanisms. These are RESTFUL web services that are unique locations onto the data.

Feature Extraction

In some cases feature extraction is performed within the Data Warehouse element or within subsystems that the Data Warehouse brokers. This includes things like moving averages and temporal features that are best processed in a streamed manner.

In other cases, for instance spectral analysis of a portion of a time series, the pattern analysis engine performs feature extraction to deliver frequency information. Complex features over combined time series, such as correlation coefficients are also calculated from raw data and the descriptive dimensions from the knowledge base in the system. Feature extraction capability also works to detect new patterns of interest. These features that form the signature of an event or cause are then stored as a pattern for future classification. A scenario may be that an event pattern is detected that in turn defines a time range to extract features from.

Features can be considered measures or compound measures that essentially supply a scalar or multidimensional space to a classifier. In our approach there are no time series past this point; temporal elements of the data have been measured and expressed as models.

Classification

Classification works to compare features with patterns to then place the result into a category that can be used to supply knowledge within the system and to end users. The categories will be consistent with ontologies that describe the physical environments. As there are many classification techniques a focus on providing a set of tools rather than single methods has been adopted. Basic classification will be around:

- Positive impact on energy use
- Negative impact on energy use
- Good/Best energy user
- Bad/Worst energy user
- Event detection: start of use, end of use, change of state

Classifications must be able to use various techniques with a study on what is the most effect way to classify data in the model.

Measures

Measures are important for the feature extraction area and are the nomenclature used for online analytical processing. There are very few direct measures in the system as the sensor input is limited to watts for various power inputs and some direct event information like, lights are on or person in room. Derived measures, such as watt hours are calculated by integrating over time within the time series services of the system. Example measures that the pattern analysis engine is interested in:

- Energy use in Wh over various periods
- Power use in W at given instant
- Usage of device as a count
- Slope of power use (dP/dt)
- Slope of energy use (dE/dt)

Dimensions

Likewise, dimensions are a fundamental concept in analytical processing. They filter the data by descriptive groupings for example by area or for an appliance. Dimensions are often hierarchical such that drill down can occur; for instance total energy usage (the measure) UK-> England-> London->Hackney filters the measure by ever smaller geography.

Dimensional analysis can often limit comparison because people may want to compare themselves to others that are not identical. For instance I might want to compare energy usage from A. a 3 bedroom house with B. a 1 bedroom flat to know whether A is better than B in terms of efficiency behavior. Since there are large biases along the dimensions it is very difficult to make direct comparisons. We will use the term “normalising” the as one that allows for the comparison within a dimension regardless of its inherent bias. The engine will be able to support normalisation across a single or multiple dimension as a part of the classification step.

Engine Implementation

The engine utilizes R at its core. R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

The R software environment runs on a Linux server that is accessible via TCP/IP in the backend private address space of the system network. It is not publicly accessible directly, end user functionality must go through the data warehouse subsystem.

API

The application programming interface for R is through RSOAP a python based SOAP server bridging the core R functionality.

RSOAP provides an API which has been designed to be simple and flexible while supporting all common operations. At this time, the RSOAPManager class provides a single API call, `newServer()`, which starts a new R Session. Once `newServer` has been called, the R Session object, implemented by the RProcess class, provides 4 categories of methods: file access, code execution, variable manipulation, and session management. These API methods provide all of the necessary functionality to interact with an R process. Except for two calls of the code, execution and variable manipulation methods directly accept and return native data types and objects. The `eval` and `script` methods are provided for handling R operations which are difficult or impossible to directly represent using the client's native data types (such as model formulae) and to facilitate the execution of code blocks (a.k.a. 'scripts'). Graphical operations are not supported by the API.

Data input

Raw data is fetched from the persistent database through API calls found in deliverable 5.2. An example for URL fetching a household's daily data in a comma separated format is:

```
http://data.dehems.org/get_daily_data.html?mac=000E2EE0B827&average=1
```

This URL is called by R's `file` function. For example:

```
house1<-  
file(http://data.dehems.org/get_daily_data.html?mac=000E2EE0B827&average=1)
```

will load data into a `house1` matrix for manipulation. Common dimensions can be loaded from the data warehouse for days of week, seasons and house types.

Function calls

Function calls are executed by a SOAP client that is able to execute R commands through a simple server call. Examples of this are found in Appendix 1.

Persistence

RMySQL is an extension that brokers database activity to a MySQL database. Results can be stored in a local MySQL database in order to reduce execution memory on long running analysis.

Experiments

From the environment that has been established a few pattern modules have been identified and experiments will be set up to test hypothesis areas for exploration.

Area 1. Normalizing data

A system for comparison between households will rely on normalising the data for dimensions that are highly biased. The main comparison of interest is on the behavioral elements of usage therefore eliminating bias introduced from size of dwelling and number of occupants.

The first set of hypothesis are around understanding the contribution of size of dwelling on electricity use and on then on the thermal characteristics of the house. Occupancy is thought to be a major contributor to biased data as more people tend to be active in a house, whereas dwelling size may only have marginal impact on anything but heat and lighting. Also the size and occupancy of the home is also related as typically the larger the home the more people that may live in it.

Area 2. Detecting waste

Many human users looking at sensor data in watts gravitate towards peaks in usage. Most peaks are due to single usage of appliances such as kettles or heat producing appliances. These peaks are often sharp and have comparatively little energy under the peak. The main contributors to energy consumption are usually hidden in the baseline of the power plot or occur as plateaus that are not easily distinguishable.

The plateau or constant use energy is where savings and usually waste occurs. This area of investigation will quantify waste and isolate the contribution due to long running appliances. Indications of waste can also be aided by the comparative measures and understanding if the energy used has been put to “good” use.

Area 3. Detecting appliance level events

The ability to detect an appliance turning on, off or going into a different state is useful when determining the waste profile and quantifying the contribution of behavior to energy use. Detection will be done through a combination of direct measurement in some cases and inference in others.

Signal processing techniques for systems identification will be used along with experimental data on directly measured contribution.

Area 4. Combined long term and short term measures for performance

A key indicator will be devised such that short term performance and long term performance can be measured as a single value. This should motivate decision making and give a rank to how a household is performing against any other household.

Conclusions

R is a flexible and powerful environment for pattern analysis and discovery. It offers a programmatic richness and access to many scientific computation libraries. The analysis can be programmed such that it is available to the larger system via SOAP or used by researchers to explore data through an interactive command line interface.

Future work

The pattern analysis engine is an analytical environment for experimentation and triggered processing of sensor signals into patterns and events. Once feature extraction and classification methods have been proven, they will become more useful within the main data processing stream of the system.

R can be used as a library by C and C++, but it is not multithreaded and not optimized for stream processing of data. Therefore, while R is a good environment for experimental analytics, it lacks the performance characteristics required for mass scale (millions of households) implementation. We are under the opinion that flexibility for trying various computation techniques and optimal performance are not possible. Once analytics have been proven, custom implementation will be required for scalable performance gains.

Much of the work in this environment will be around selecting the best techniques for analysis and understanding what features of the sensor data hold the most value. Limited (because of resource constraints) access to this environment should be made available to external researchers so that other experiments can be conducted with direct access to real, live data.

Implications for other Work packages

WP3 requires measurements and analysis for both display and decision support. This environment will supply WP3 with its requirements. Likewise there is a dependency on WP3 to supply concepts into the analysis, such as time ranges, house and appliance categories and geographical references.

References

1. R - A statistical computing project - <http://www.r-project.org/>

Appendix 1. Sample RSOAP session

```
import SOAPpy

def unwrap(object ):
    "Unwrap SOAPpy objects to get 'raw' python objects"

    if isinstance( object, SOAPpy.SOAP.structType ):
        return object._asdict
    elif isinstance( object, SOAPpy.SOAP.arrayType ):
        return object.data
    else:
        return object

print "## Contact the RSOAPManager and ask for a new R Session\n"
mgr = SOAPpy.SOAPProxy("http://pattern.dehems.org:9980")
server_url = mgr.newServer()
print "Result: %s " % server_url
print "\n"

print "## Connect to the new session\n"
server = SOAPpy.SOAPProxy( server_url )
print "Result: %s " % server
print "\n"

print "\n"
print "## Request 10 random values\n"
x= unwrap(server.call("rnorm",10))
print "Result: %s " % x
print "\n"

print "## Request 10 random values with given means\n"
y = unwrap(server.call("rnorm",10,x))
print "Result: %s " % y
print "\n"

print "## Fit a linear model\n"
server.putObject("x",x)
server.putObject("y",y)
lm = unwrap(server.script("reg <- lm( y ~ x ) \n summary(reg)"))
print "Result: %s" % lm
print "\n"

print "## Extract a the p-values\n"
p_value = unwrap(server.eval("coef(summary(reg))[,4]"))
print "Result: %s" % p_value
print "\n"

print "## Close session ###\n"
server.quit()
print "\n"
```