Deliverable **4.3**

Algorithms and architectural requirements

Lead Partner:	UNIVERSITY OF SKOVDE
Authors:	UNIVERSITY OF SKOVDE
Date:	October 2014
Revision:	V0.1
Dissemination Level	PUBLIC

With the support of

Project Acronym: HELICOPTER
Project full title: HEalthy LIfe support through COmPrehensive Tracking of individual and Environmental Behaviors
AAL project number: AAL-2012-5-150

Algorithms and architectural requirements of information fusion methods

There are two part of the requirements: (i) the technical requirements associated with interfacing with software components that can run the models employed in HELICOPTER and (ii) the requirements on the data fed into the software components as well as data that is used for data-generated models.

Technical requirements

There are two parts: (i) the hypothesis testing of diagnostic suspicions is based on Bayesian Belief Networks that uses an open-source tool Genie for development and SMILE for execution and (ii) a Java packages for anomaly detection (one is based on the incremental local outlier detection method and the other is based on random forest classification).

The technical requirements of evaluation of Bayesian belief networks is to interface with SMILE, either from Java or C++. The API provided is employed to set and clear evidence as well as evaluate unknown variables based on the evidence. See appendix for an example of usage.

Concerning the Java packages, the APIs are documented in the appendix.

Data requirements

There are a number of requirements on the data fed into the reasoning:

1. Each data representing a information from an information source such as a sensor must be timestamped.
2. Each data should be anchored to a providee with a certainty. This certainty may change over time. This certainty should be computed based on anomaly detection based either on incremental local outlier detection or random forest classification. For verification and validation purposes, it is desirable to keep the history of changes to the certainty.
3. Depending on the assumptions in the reasoning, data may need to be transformed. The following operations may need to be performed:
    a. Low-pass filtering by, for example, employ a moving average window for the recent past of the history of sensor data that is useful for making decisions in the present. For example, if a person flushes the toilet twice in a few minutes, then this may indicate that there is something wrong with the toilet and should be ignored.
    b. To compute the frequency of events over a period of time that is meaningful to the decision making in the present. For example, what is the frequency of flushing the toilet associated with

c. To consider the trend over a period of time that is useful for making decision in the present. For example, how has the average movement speed changed over the last month?
d. To perform a spectographic analysis via, for example, fast Fourier transformation if data can, for example, be of bursty nature and we want to analyse the various components of the signal. This is useful if we have a continuous signal that we measure, where changes are of bursty nature.

4. This preprocessed aggregated data can be used by the hypothesis testing either directly or indirectly via anomaly detection. For example, if we check the frequency of toilet flushings over a moving window of 24 hours with respect to time of day, then a normal model for anomaly detection based on these two data can be used to detect when the frequency is abnormal in a context. It can be useful to add other contextual data that may improve the ability to detect anomalies (e.g., power consumption).

Sensor data and proposed aggregation functions

In the following table, the connection between the evidence indicators and information sources outlines in the Domain Model (D4.1) is detailed out with respect to what we monitor (state or events), if frequency should be computed, if moving average should be applied, the duration of the time window that the processing (frequency or moving average) should be applied to[1], the update frequency and if anomaly detection should be applied. For example, at row 14, the movement speed indicator is based on the MUSA as well as electronic gates and reads the state (movement speed), where we employ a moving average over the last hour computed every 5 minutes. This is fed into anomaly detection and, if there is an anomaly, the value is compared to the expected value; if the value is less than expected, then this is an indication of decreased movability.

#	Evidence indicator	Sensors	Event, state	Frequency	Moving average	Past time window	Update frequency	Anomaly detection
1	Accelerometer	MUSA	State	N	Y	1h	5min	Y
2	Age	Configuration, dialogue	State	N	N	-	-	N
3	Catheter	Configuration, dialogue	State	N	N	-	-	N
4	Diabetes	Configuration	State	N	N	-	-	N
5	Diuresis frequency	Dialogue, PIR, toilet flushing	Event	Y	Y	24h	1h	Y

[1] These are initial estimates. HELICOPTER should be configurable so that when the duration of the past time window has been tested, then we can refine them. This will initially be done in task T4.4, where we refine the setting by the help of simulations.

6	Fever	Dialogue	State	N	N	-	-	N
7	Food intake	Dialogue, food diary	State	N	N	-	-	N
8	Food intake: confirmed	Dialogue, food diary	State	N	N	-	-	N
9	Food quality	Dialogue, food diary	State	N	N	-	-	N
10	Gender	Configuration	State	N	N	-	-	N
11	Hypertension							
12	Insuline or insuline stimulating medicine	Dialogue	State	N	N	-	-	N
13	Laying	MUSA, Dialogue, Pressue sensor	Event, state	Y	Y	24h	1h	Y
14	Moving speed	MUSA, Electronic gate	State	Y	Y	1h	5m	Y
15	Open fridge at night	Dialogue	State	N	N	-	-	N
16	Physical activity	MUSA, Dialogue, electronic gate	Event	Y	Y	24h	1h	Y
17	Renal failure							
18	Self monitoring not requested	Configuration, dialogue	State	N	N	-	-	N
19	Sitting	MUSA, Dialogue, pressure sensor	Event, state	Y	Y	24h	1h	Y
20	Soft drink intake	Dialogue	State	N	N	-	-	N
21	Weight	Scale, dialogue	State	N	Y	7days	24h	Y

The context of anomaly detection can consist of different aspects. For example, time of day, week day and month can be used to account for variations in the weeks and months due to different routines and typical weather type. This can be complemented with weather as well as power consumption data. In HELICOPTER, we mainly employ time of day, week day and month for anomaly detection. Since the algorithm is local and incremental, the normal model will be continously updated as the providees live their daily lives.

If anomaly detection is employed, then if there is an anomaly the value is compared to the normal value. If the value is increased or decreased, then the value of the evidence indicator is set to increased or decreased respectively.

Appendix - Use of Bayesian Belief Networks

For tutorials and API documents as well as installation instructions, access the Genie/SMILE home page at https://dslpitt.org/genie/.

Briefly, an application must load the network from the file systems. Once loaded into an object, the hypothesis testing is performed by setting evidence of the evidence indicators and then read the hypothesis nodes. The value of the hypothesis is compared to the output dictionary for that hypothesis; if the value is above a threshold, then there is a risk, otherwise not. In some cases, there may be multiple thresholds, indicating different levels of risk.

In the following table, an example of loading the Bayesian belief network representing the reasoning of hyperglycemia is presented.

```
import smile.Network;
import smile.SMILEException;

Network network=new Network();

network.readFile(PATH);

network.clearAllEvidence(); // clears all evidence
network.clearEvidence("BodyWeight"); // clears the evidence of node with id
"BodyWeight"
network.setEvidence("BodyWeight","normal_or_decrease"); // sets evidence

double hypothesisValues[]=network.getNodeValue("Hyperglycemia");
for (int i=0; i<hypothesisValues.length; ++i) {
   System.out.println("Value of hypothesis \"Hyperglycemia\", outcome
\""+network.getOutcomeId("Hyperglycemia",i)+"\" = "+hypothesis[i]);
}

if(hypothesisValues[n]>RISK) { // n is the index of the risk value
   // Warn
}
```

Appendix - Local incremental outlier detection

The java implementation of the local incremental outlier detection is under development. It will look something like this:

```
MultiDimensionalPointType mdpt=new MultiDimensionalPointType();
mdpt.addParameter(new Parameter("HourOfDay",Int.class));
mdpt.addParameter(new Parameter("WeekDay",Int.class));
mdpt.addParameter(new Parameter("Weight",Double.class));

DataStream datastream=new DataStream(2,mdpt);

Vector<Euclid> data={13,1,53.7};
MultiDimensionalPoint mdp=new MultiDimensionalPoint(mdpt,data);
datastream.insertDataRecord(mdp);
if (datastream.anomaly()) {
  // there is an anomaly
}
```