INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 9(5):644-654, October, 2014.

A Knowledge-based Telemonitoring Platform for Application in Remote Healthcare

W. Zhang, K. Thurow, R. Stoll

Weiping Zhang*, **Kerstin Thurow**
University of Rostock
Celisca, Center for Life Science Automation, Rostock 18119, Germany
*Corresponding author: weiping.zhang@uni-rostock.de
kerstn.thurow@uni-rostock.de

Regina Stoll
University of Rostock
Institute for Preventive Medicine, University Medical Center, Rostock 18055, Germany
regina.stoll@uni-rostock.de

Abstract: Telemonitoring systems have been shown to greatly reduce medical costs while improving the quality of medical care. Today, the main factors restricting the development and popularization of Telemonitoring systems include scalability and compatibility. The challenge for the remote healthcare lies in the variety of heterogeneous medical sensors which need to be dynamically removed or added to the environment according to the health care needs. This paper presents the design for an ontology-based context model and related middleware that provides a reusable and extensible application platform for Remote Healthcare. We designed the ontology context model to describe physiological parameters, medical tasks and the patients personal profile. Developers may extend the ontology model by considering new requirements as needed.
Keywords: telemonitoring, ontology, knowledge-based System, Context-Aware application

1 Introduction

Increasing life expectancy, due to medical progress, and decreasing birth rates lead to a change in the population structure of Germany. As illustrated in Fig. 1, the percentage of people over the age of 65 is steadily increasing while the number of people under the age of 25 is declining [1]. This results in continuously rising costs for healthcare, especially for elderly people. One approach to this problem is the utilization of telemonitoring systems to enable a relocation of the therapy to the patients home, thus shortening the expensive periods of hospitalization.

A Medical Telemonitoring System can be defined as a technological means for sending remote physiological information and medical signals through a communication network to a monitoring center for analysis and diagnostics. The system generally include three components: a monitoring center, medical sensors, and a communication network connecting the two components.

However Telemonitoring systems are still in the early stages of development and currently face many challenges. Compared to a general monitoring environment, a remote healthcare environment has the following features: (1) Dynamistic: the remote healthcare environment usually requires a number of medical sensing devices and, unlike air conditioning which is fixed in an environment, some medical sensors need to be removed or added to the environment according to the health care needs; (2) Complexity: in the healthcare environment, a wide variety of medical terminology are used, and the relationships between them are very complex [2].

In order to solve these problems, we designed a Mobile Context and Ontology-based Reasoning/Feedback (MCOM) health-monitoring system which monitors a patients health status using a Smartphone. On the Smartphone, a Context-Aware Middleware is developed to connect

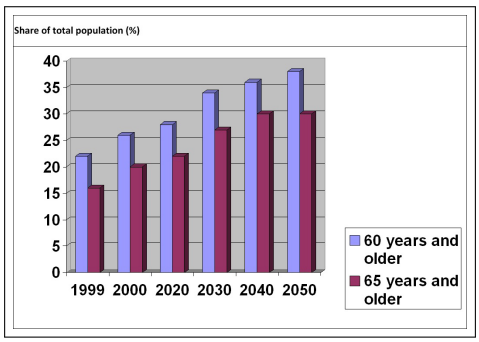

Figure 1: Projected Percentage of Elderly People in German Population

different types of medical sensors. In addition, MCOM uses an ontology-based context model to process and determine a patients health status.

2 Healthcare Context-Aware Middleware (HCAM)

2.1 Context-Aware Middleware

The first problem above mentioned can be solved by establishing a Context-Aware Middleware [3]. Context-Aware computing refers to a general class of mobile systems that can sense their physical environment, and adapt their behavior accordingly [4]. In order to extricate remote medical applications from tedious sensor data acquisition and management, this paper proposes a Healthcare Context-Aware Middleware (HCAM). With HCAM, the upper-layer applications would not be concerned about acquiring the context data, but only about the business logic itself.

As shown in Table.1, applications based on HCAM are divided into three layers. HCAM separates the collection of context information and the development of aware applications. It provides a Subscriber API (i.e. Subscriber Application Programming Interface [5]) for the developer of the upper-layer application; HCAM is then responsible for completing the data collection of various kinds of sensors.

Table.1 Healthcare Context-Aware Middleware	
Context aware applications	
Healthcare Context-Aware Middleware	Subscriber API
	Pub/Sub framework
	Provider API
	Data collection
All kinds of sensors	

2.2 Ontology Modeling

The second problem can be solved using the Ontology Context Model and Reasoning. Ontology can be defined as, an explicit and formal specification of a shared conceptualization [6] and provides a kind of shared vocabulary, including object type or concepts, their properties, and relationships existing in special fields. In this work, it is used to define information and relations, including sensor data, medical tasks, control strategies, and a patients personal domain knowledge. A medical task, like an Emergency Alert, can be triggered by Context Information reasoning based on the analysis of physiological parameters.

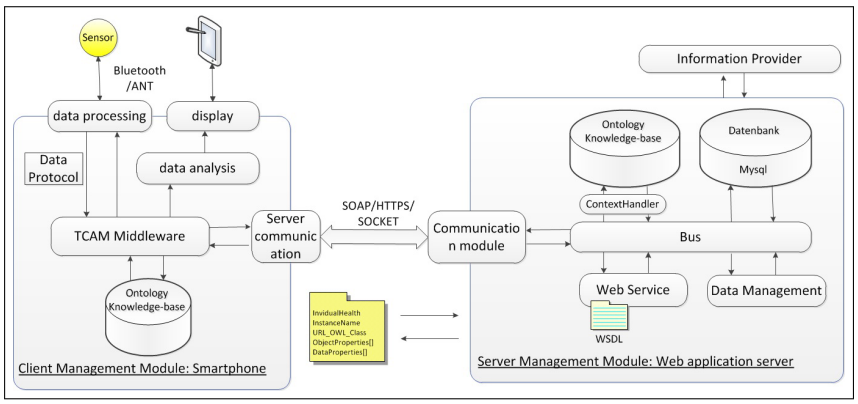

Figure 2: Telmonitoring System Architecture: Management Modules and Communication link

2.3 System structure

This work aims at describing the ontology-based context model and middleware that have been developed in order to ease the development of Context-Aware applications for remote healthcare [7]. The system architecture is shown in Fig.2. From a physical perspective, it is based on a C/S architecture which includes the sensor terminal, Smartphone client terminal, and remote server terminal. From a functional perspective, the whole system can be divided in to three modules: Client Management Modules (CMD), Server Management Modules (SMD), and communication mechanisms.

1. The core module at the Smartphone client is the HCAM Middleware. It is responsible for communicating with various sensors and performing data collection. Other core components include data processing, analysis, and display modules plus server communication modules. It should be noted that, unlike other Telemonitoring systems, the communication between client and server is bidirectional. In addition to sending data to a server, the client terminal can also synchronize information with the server. The design concept of the client terminal is to offload as much processing as possible to the server, so that the Smartphone client operates only as gateway to display and transfer data.

2. On the Server-side, the core component is the knowledge-base, based on ontology. Other components on Server include a communication module that is responsible for transmitting information between the server and the client terminal, a data management module, plus a web service module. These modules communicate through a task bus.

3. As shown in Fig.2, the communication between CMD and SMD is implemented through HTTP/HTTPS, SOAP [18] and SOCKET. HTTP is used in synchronous XML workflow description; SOAP is used to transmit ontology instance information; SOCKET connection is used in high real-time demand, for example, transmitting ECG (1024Byte/Sec) and Acceleration (200Byte/Sec) data.

2.4 HCAM Structure

The HCAM proposed in this paper is designed and developed based on the Android system for Mobile Phone. Its architecture is shown in Fig.3. According to the previously mentioned three-layer structure, HCAM include these three layers, i.e. context data acquisition, context data transmission and distribution, etc. HCAM first uses the context provider to collect the original context data provided by the sensors, then transmits the data to the upper layer applications via

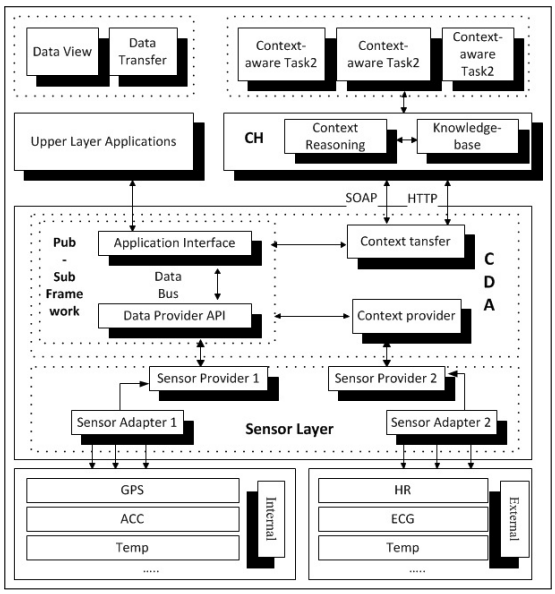

Figure 3: HCAM Architecture

the Pub/Sub framework. The Pub/Sub framework [8] completes the context data transmission and distribution. The following are the core components of the HCAM .

1. **Sensor Layer** All the original context of physiological data are derived from the sensor, which is the basis of any Context-Aware application. This paper divided sensors into two categories; internal sensors (located in the Smartphone) and external medical sensors.

2. **Data Provider** Data provider (Internal) 1 and data provider (External) 2 are shown in Fig. 3. A data provider usually exists in the form of a plug-in and is written using Provider API. Once the providers are completed, the Provider API will transmit the data to the upper sensing applications. During the actual realization process, data provider 1 and data collector 1 are usually combined and exist as many different system plug-ins. However, data provider 2 and the data package resolver, are combined and exist as an independent plug-in. Under these circumstances, the data provider can directly transmit the data, read by the data collector, and the data analyzed, by the data package resolver, to the upper sensing applications via Data-Bus [9].

3. **Context Discovery and Access (CDA)** CDA consists of the context provider, Pub/-Sub framework, and context transfer. The context provider serves as a context discovery protocol to connect different sensors. The Pub/Sub framework collects context information from the context provider, and the context transfer uploads the data to the central server

4. **Context Handler (CH)** CH reasoning is accomplished through the context instance from the client sensor data. The context information uses an inference engine based on conditions to conduct reasoning via OWL/RDF [10] modeling (Knowledge-base). Once the conditions are met, the context handler module will send information to the Context-Aware task and monitoring server.

3 Knowledge-base design

Before processing the Context Information, we first need to construct the Context Model. In an ubiquitous medical environment, a large number of medical devices and domain terms are used. With the advancement of technology, new domain terms can be joined very quickly. In addition, in the rescue and medical care process, the pathological conditions of a patient usually cannot be identified directly; medical professionals must filter and integrate information from a variety of different sources. Therefore, the context model permits not only adding the description of a new concept, but also supports the context reasoning.

By using ontologies and various reasoning methods, the system can process the sensor data gathered in a more parameters to create a method by which the engine can infer the patients health status [11].

The context knowledge-base has two parts. The first part stores the ontology and their instances in the system. The other part provides a reasoning interface and context discovery, i.e., how to inquire and to add modules. Ontology instances may be pre-defined in the profile or may be retrieved from the device and other Webservice. The pre-defined profile is dynamically imported into the system at the start of system operation. The context information also includes some fixed information from a specific time; for example, the patients blood type, emergency telephone number, clinical records, etc. Fig.4 shows a UML diagram of the context ontology specialized for the Patient Personal Domain. In the Patient Personal Domain, relevant context items include Measurements, Patient Profile, Device, and Task. These data can be used by the system to automatically deduce the patient health status and detect possible medical tasks.

The Class Patient, (see Fig. 4), constitutes the core of the ontology. Each defined patient will be configured as an instance of this Class. One patient personal profile instance is generated and associated for one particular patient, but this patient could have more than one patient profile instance associated to him [12]. A specialization of the Measurements Class is added for each specific physiological parameter which is gathered from a sensor (e.g., HeartRate Class). Measurement values are used to determine patient health status by comparing these values with a set of parameter ranges (MeasurementsRange Class) or results of the specialized Data-processing-algorithm (like the Stress-assessment algorithm in [13]). When a measured value falls outside the limits, a task (e.g., Alarm Task) of the corresponding level is then triggered. For Example, Class HeartRate (see Fig. 4) is a subclass of the Measurements Class ,and in this case, there are three alert rangeslow, medium, and high which can be modified by the Expert to adapt to patients' specific needs. In the case of HeartRate, the three ranges indicate HeartRate is too low, Normal or too high. And the model can be easily extended to include further levels. The hasCurrentValue attribute contains the most recently received sensor value from the mobile client.

This system uses OWL to construct the model and process it using Sample Semantic Web Rule Language [14]. The reason for using the context to construct the model is that, it can not only express the context information, but can also use the inference engine based on conditions to conduct inference to the subtle context to get higher level information [2] The context models have the following forms: (Subject, Predicate, Value)

- Subject belongs to S*: set of subject names. a patient, a place , a Smartphone, or an Emergency Task for example.
- Predicate belongs to P*: the set of Predicate names, the patients current body temperature for example.
- Value belongs to V*: the overall value of the Subject.

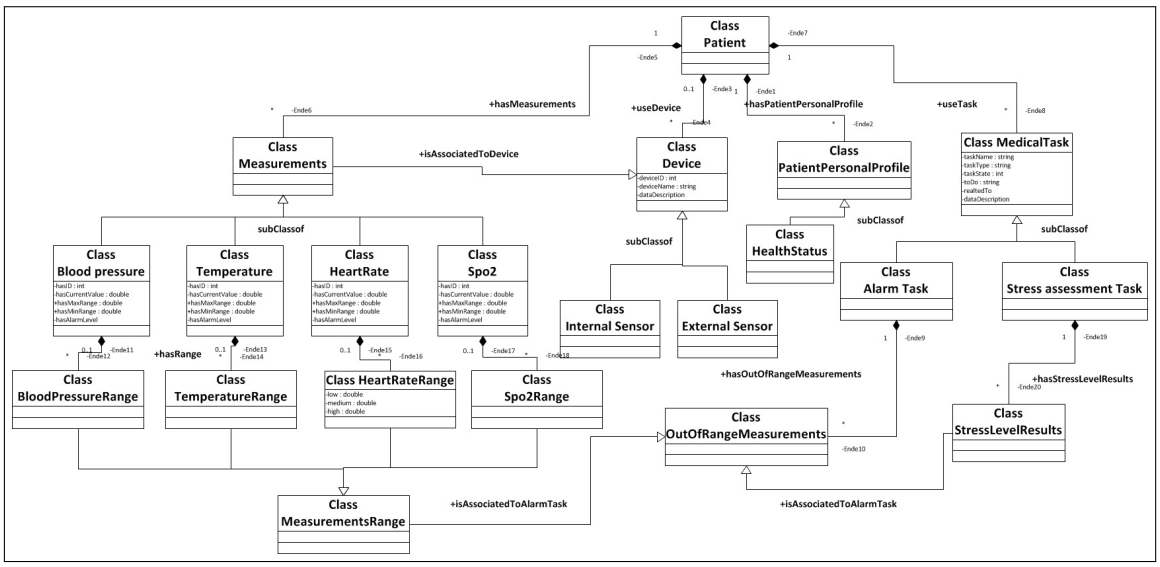

Figure 4: Patient Remote Healthcare Domain Ontology UML Representation

For example, we have a patient with ID 001, body temperature 37°C and use a PDA to connect to the monitoring center, then information would be recorded as follows:

- (PDA, online, true), (Patient001, Temperature, 37)

Table 2 shows a few example rules, which the system uses to predict the sensor-data-specific task and Table 3 shows basic example policy we defined for each Emergency Task level [7]. The HeartRateHigh rule assigns the Emergency Task level when the heart rate is abnormally high. The hasMinRange and hasMaxRange properties specify a normal heart rate in each instance of the HeartRate Class. If a patients heart rate is greater than the specified normal maximum, then the system sets the Heartrate alarm and starts the Emergency Task .

Table 2: rules for the systems alarm management	
RULE	DESCRIPTION
HeartRateHigh	(?patient rdf:type HeartRate) , (?par1 hasCurrentValue ?v1) , (?par1 hasMaxRange ?Max) greaterThan(?v1, ?Max) , − > (?taskstate hasAlarmLevel HES)
HeartRateLow	(?patient rdf:type HeartRate) , (?par1 hasCurrentValue ?v1) , (?par1 hasMinRange ?Min) , greaterThan(?v1, ?Min) , − > (?taskstate hasAlarmLevel LES)

Table 3: Alarm Task Notification Policy Examples	
Alarm TASK LEVEL	NOTIFICATION POLICIES
Low Alarm Task (LES)	sensor alarm sms to patient relative
Medium Alarm Task (MES)	sensor alarm sms , mail and call to relative
High AlarmTask (HES)	sensor alarm message to emergency operator call to relative

Unlike common ontology systems, in this paper the medical task is also included in the ontology; the task will invoke specific activities. There are two advantages in doing this: the first advantage is that the specific details of the task and the sub-actions are divided. Under these circumstances, the description of the task doesnt need to be changed when the device information, used by a sub-activity, changes. The other advantage is that the description of task by ontology, can make inference to the changes that should be made by the task; the inference processing depends on the task status, context information, and the custom rule. At the root node, is the design of a type of named task and a specific task level taking task. After inference is completed and the instances of task type exist, SOAP objects are sent to the task response components and then to the client terminal, including all task information. We use the following attributes to describe a task (see experiment):

1. TaskName: the name of the task;
2. TaskType: The type of task and function description information;
3. Todo: describes the current sub-activities of the task;
4. TaskState: indicates the current state of a task. There are three values to be chosen: -CLOSE. , -INIT and -OPEN and they respectively indicate the task has not been started, or the task needs to be initialized or is started for use.
5. RelatedTo: indicates which specific function in the ontology the task is related to, and what kind of device instances the task is related to.
6. DataDescription: data description describes the data type and format that needs to be collected by the task. The information is stored in the task profile and uploaded when the system is operational.

3.1 Simulation

In this section we will use an emergency assistance task as a practical example to describe the above described design pattern: how new devices are added to the ontology knowledge-base and medical treatment begins via conditional reasoning.

The specific scenario for an emergency assistance task is as follows: An anomaly has been detected for a patient suffering from heart disease or high blood pressure; based on system reasoning, emergency assistance is requested. According to this task profile, the response time of the task is one minute, meaning that medical personnel must respond within one minute, react to the patients condition and initiate corresponding measures. After the response, the tasks mode is set to CLOSED. If the patient's condition remains abnormal, the task is started again.

The first step is to add new devices: In the experiment we used the wearable multi-parameter sensor Equivital01 produced by (Hidalgo Ltd., Cambridge).

When a new device is connected to the HCAM sensor adapter, context information must be converted into ontology instances before being sent to the knowledge base. The HCAM context handler inquires whether classes with identical names already exist in the knowledge-base. If not, a new class is added with the new $< device >$. The following example shows the class of an Equivital device being added:

- $< owl : Classrdf : ID = Equital >$
  $< rdfs : subClassOfrdf : resource = *Device/ >$
  $< owl : disjointWithrdf : resource = */ >$
  $< /owl : Class >$

Then a new class is added which acts as subclass of Measurement. Below the classes HeartRate and BloodPressure are added for the Equital device.

- $< owl : Classrdf : about = *HeartRate >$
  $< rdfs : sublassOfrdf : resource = *Measurement/ >$
  $< owl : disjointWithrdf : resource = */ >$
  $< /owl : Class >$
  $< owl : Classrdf : about = *BloodPressure >$
  $< rdfs : sublassOfrdf : resource = *Measurement/ >$
  $< owl : disjointWithrdf : resource = */ >$
  $< /owl : Class >$

With the establishment of the above-mentioned ontology, it is ensured that the device description file and the ontology name match. When the HCAM context provider receives data from the device, it looks for the parameters in its own device files and device task files and creates an instance. The above-mentioned Equivital device might, for example, receive event messages from the HCAM detailing HeartRate-hasCurrentValue = 60(Beats/Minute), BloodPressure-hasCurrentValue = 120(systolic, mmHg). In this case the practical example would look as follows:

- $< HeartRaterdf : about = *EquitalHeartRate >$
  $< hasCurrentValuerdf : datatype = double > 60.0 < /hasCurrentValue >$
  $< /HeartRate >$
  $< BloodPressurerdf : about = *EquitalBloodPressure >$
  $< hasCurrentValuerdf : datatype = double > 120.0 < /hasCurrentValue >$
  $< BloodPressure >$
  $< Equitalrdf : about = *Equital001 >$
  $< measurementsrdf : resourcee = *EquitalHeartRate/ >$
  $< measurementsrdf : resource = *EquitalBloodPressure/ >$
  $< /Equital >$

After the context information is converted, the context handler will send the aforementioned information to the knowledge-base.

The second step is to execute the task.
When the middleware is executed, the system automatically generates an instance for every task class, which is saved in the knowledge-base. The important information in these task instances is as follows:

- $< MedicalTaskrdf : about = *EmergencyTask >$
  $< taskType > External < /taskType >$
  $< taskState > CLOSE < /taskState >$
  $< todordf : resource = *AlarmTask/ >$
  $< todordf : resource = *PrepareEmergencyOperator/ >$
  $< /MedicalTask >$

With the previous scenario in mind, we can define the specific conditions for the emergency task to be triggered: e.g. the Heart Rate of the patient drops below 40, or the blood pressure (systolic) exceeds 170. Reasoning may be carried out according to the rules of the language itself, for example:

- $(?a?p?b), (?prdf : subPropertyOf?q) - > (?a?q?b)$

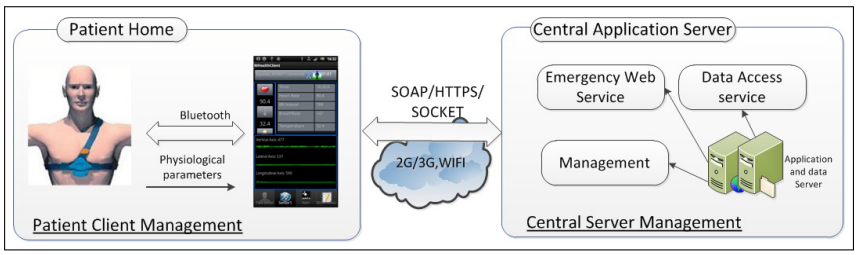

Figure 5: mHealth Prototype System Architecture

User defined rules may also be used in the reasoning process, e.g.:

- $(?patient, EquitalBloodPressure001, vl), GE(?vl, 170), (?patient, EquitalBloodPressure, vl), LE(?vl, 40) -> (?patient, healthStatus, danger)$
- $(?patient, healthStatus, danger), (?Patient, healthStatus, movementfail), (EmergencyTask, taskState, CLOSE) -> (EmergencyTask, taskState, OPEN)$

3.2 Deployment and Execution

The deployment configuration includes at least two Context Manager nodes as shown in Fig.5: We used a SonyEricsson-Xperia smartphone as Client Manager (CM) node, connected with the medical sensor module Equivital [15] via Bluetooth; and a Server Manager (SM) node, which is deployed on a Central Application Server, in the domain of the organization which is responsible for the integration and delivery of Emergency Web Task.

The Home Smartphone hosts the Client and communicates with the Central Application Server via 2G/3G/WiFi. The Central Application Server hosts the Server SM and other assistance and information task applications. These tasks include: a Emergency Task as Webservice, Data management, and a web application which offers web-based access to health operators via mobile device.

4 Results and Discussion

Fig.6 shows the sensor data for a patient. The graphic curve represents any changes in the data for a given day. Date and time can be changed easily, access a patients historical data; this data are stored in a central server database. The alarm is normally triggered automatically by context reasoning and, in some cases, the medical professionals can manually send a message to the client. Directly above the line chart in Fig.6, a message window is located. Medical professionals can send a message to the patient's Smartphone through it.

In this technical approach, the system structure improves the scalability and interactivity of telemonitoring systems; In our current work, we improve the reliability of the system by adding layers of redundancy, in case a particular system or function fails. In addition, we created a back-up system on the mobile client to save sensor data locally, in the event of a loss (or lack) of internet connection. Security issues can be divided into two parts: one is the security of local data transferred from the sensor to the mobile client; another is the security of remote data transferred from the mobile client to the central server. On the client side we implement the new 128-bit lightweight cryptographic algorithm Hummingbird-2 [16] cipher to encrypt the sensor data with java code. On the server side, we use Secure Socket Layer (SSL) to encrypt vital data and utilize 3P-AAA-SPs [17], Authorization and Accounting Task Providers to authenticate the client access.

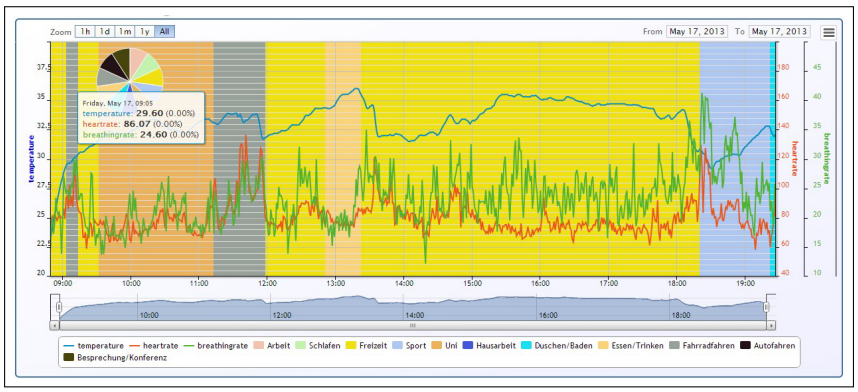

Figure 6: mHealth Prototype System Architecture

This paper designs an innovative Telemonitoring platform based on context and ontology technology. A Context-Aware Middleware is developed to support the scalability of the telemedical system; new sensor device and data formats can be integrated into the system conveniently. In the context reasoning module, ontologies are used to construct target medical tasks and circumstances. This paper, however, does not address the handling of context conflicts in the process of constructing ontologies and reasoning conditions. Further research will focus on the reliability and energy consumption problem of remote medical systems. Concerning the energy consumption problem, some new ultra low power transmission protocols, like BLE and ANT, have appeared that can greatly improve the sustainability of remote monitoring. Other challenging tasks will include the development of an intelligent error discovery and system recovery mechanism to maintain the stability of the system over time.

Bibliography

[1] Sitzungsberichte der Leibniz-Sozietat, Band 90 (2007); *Theoria cum praxi. Funf Jahre Leibniz-Institut fur interdisziplinare Studien e.V. (LIFIS)*, Trafo Verlag, Berlin, 217-228.

[2] Cheng J., Rao R. (2009); A Context-Aware middleware for pervasive healthcare. *Computer Applications and Software*, 50-53.

[3] Baldauf M., Dustdar S., Rosenberg F. (2007); A survey on Context-Aware systems. *International Journal of AdHoc and Ubiquitous Computing*, 2(4) : 263-277.

[4] Kuna M., Kolaric H., Bojic I., Kusek M., Jezic G. (2011); Android/OSGi-based Machine-to-Machine Context-Aware system. *Proc. of the 11th Int. Conf. on Telecommunications, ConTEL*, 95-102.

[5] Papapanagiotou I., Falkner M., Devetsikiotis M. (2012); Optimal functionality placement for multiplay task provider architectures. *IEEE Transactions on Network and Task Management*, 9(3): 359-372.

[6] Studer R., Benjamins V.R, Fensel D. (1998); Knowledge Engineering:principles and methods. *Data and Knowledge Engineering*, 25:161-197.

[7] Paganelli F., Giuli D. (2007); An ontology-based context model for home health monitoring and alerting in chronic patient care networks. *Proc. 21st Int. Conf. on Advanced Information Networking and Applications Workshops/Symposia*, 838-845.

[8] Wang N., Yang Z., Yang Y.(2011); Based on event-driven and task-oriented architecture business activity monitoring design and implementation. *Int. Conf. on System Science, Engineering Design and Manufacturing Informatization, ICSEM*, 241-245.

[9] Xiong L., Niu J., Zhang J., Chen F., Shen P.(2011); Context Sensing Middleware Based on Mobile Devices. *Journal of Chinese Computer Systems*, 1170-1174.

[10] Armas R., Cuenca G., Horrocks I.(2012); MORe: Modular combination of OWL reasoners for ontology classification. *Lecture Notes in Computer Science*, 1-16.

[11] Benlamri R., Docksteader L. (2010); MORF: A mobile health-monitoring platform, *IT Professional*, 12(3):18-25.

[12] Lasierra N, Alesanco A, Garcia J. (2010); An ontology approach to manage individual patient profiles in home-based telemonitoring scenarios. *Proc. of the IEEE/EMBS Region 8 Int. Conf. on IT Applications in Biomedicine, ITAB*, 1-4.

[13] Kumar M., Weippert M., Vilbrandt R., Kreuzfeld S., Stoll R. (2007); Fuzzy evaluation of heart rate signals for mental stress assessment. *IEEE Trans Fuzzy Syst*, 15:791-808.

[14] Horrocks I. et al. (2004); SWRL: A Semantic Web Rule Language Combining OWL and RuleML, World Wide Web Consortium (W3C), www.w3.org/Submission/SWRL.

[15] Neubert S., Arndt D., Thurow K., Stoll R. (2012); Mobile real-time data acquisition system for application in preventive medicine, *Telemedicine and e-Health*, 16:504-509.

[16] Engels D., Saarinen M.J.O., Schweitzer P., Smith E.M. (2012); The hummingbird-2 lightweight authenticated encryption algorithm. *Lecture Notes in Computer Science*, 19-31.

[17] Le X.H, Khalid M., Sankar R., Lee S. (2011); An Efficient Mutual Authentication and Access Control Scheme for Wireless Sensor Networks in Healthcare. *Journal of Networks*, 355-364.

[18] Tekli J.M, Damiani E., Chbeir R., Gianini G. (2012); SOAP processing performance and enhancement. *IEEE Transactions on Services Computing*, 3: 387-403.