A Service Architecture for Sensor Data Provisioning for Context-Aware Mobile Applications

Bernardo Gonçalves[1], José G. Pereira Filho[1]
[1]Computer Science Department
Federal University of Espírito Santo
(UFES), Vitória (ES), Brazil
{bgoncalves, zegonc}@inf.ufes.br

Giancarlo Guizzardi[1,2]
[2]Laboratory for Applied Ontology
ISTC-CNR
Trento, Italy
guizzardi@loa-cnr.it

ABSTRACT
One of the main issues that inhibit the development of context-aware mobile applications is the lack of systematic methods for sensor data acquisition. This lack, however, is a result of the diversity of sensor data and its acquisition devices. In face of this, there is a need for general engineering solutions in order to address the common sensor data acquisition concerns. This paper presents a service-oriented architecture that allows the rapid prototyping of sensor data provisioning systems. This architecture is then applied to the Healthcare domain for providing cardiac signals in the scope of a context-aware telemonitoring system. The architecture is defined by entity and behavior models through a service-oriented design (SOD) language that has tool support.

Categories and Subject Descriptors
D.4.7 **[Operating Systems]**: Organization and Design- *distributed systems, hierarchical design, real-time systems and embedded systems*; D.2.11 **[Software Engineering]**: Software Architectures – *domain-specific architectures*; D.2.13 **[Software Engineering]**: Reusable Software – *domain engineering, reuse models*.

Keywords
Context-aware mobile computing, Pervasive computing, Domain engineering, sensor data acquisition, service-oriented architecture.

1. INTRODUCTION
The effort to develop context-aware mobile applications has been increasing with the introduction of novel application domains and usage scenarios. Similarly to other development practices, the development of such applications can be facilitated by infrastructural support for handling recurrent challenges of design and technology in a generic manner. With this in mind, several middleware platforms for supporting context-aware services have been proposed [3], [4], [9]. As a rule, the approach taken in such initiatives focus on sensor data (or context data) usage and abstract sensor data acquisition by assuming that context-aware platforms should be supplied by context sources (or sensor data providers). Thus, one may wonder what these context sources are.

Considering this particular issue, most efforts in the literature focus on context interpretation, context services management, subscription and privacy control, among others. In contrast, a general perspective of sensor data acquisition from heterogeneous devices remains quite unaddressed as a research topic. We argue that the diversity of data types and acquisition devices constitutes a special challenge in the development of context-aware applications and platforms, which motivates a more systematic approach to handle sensor data provisioning. As an example, consider sensor data acquisition in a hospital environment. In such place, several sort of data is acquired, either data presented as discrete such as temperature and blood pressure or continuous data such as cardiac signals. These data may require an integrated, simultaneous and/or homogeneous treatment.

We advocate that, in fact, there is a need for capturing common problems and solutions regarding sensor data acquisition. These solutions can be adapted and customized whenever a particular requirement has to be addressed. This paper elaborates on a systematic method for sensor data acquisition and provisioning in mobile and pervasive scenarios. Our proposal constitutes a service-oriented architecture, named **Context Wrapper**, to support sensor data provisioning for context-aware mobile applications. This architecture is a result of an approach concerned with domain engineering analysis and design phases. We took into account the sensor data acquisition requirements obtained from the analysis phase and used the service-oriented computing paradigm to produce it.

We have used a service-oriented design (SOD) methodological support that is proposed by Quartel et al. in [14]. The architecture proposed is defined by an entity model as well as a behavior model. Both them were conceived using the Interaction System Design Language (ISDL) [10]. The combination of ISDL with a tool support, has allowed us to check the consistency of the architecture behavior through simulation. In addition, we have adopted and followed design-quality principles in order to drive our work and then constitute objective criteria to later evaluate it.

As a proof of concept, the proposed architecture is applied to develop **ECG Wrapper** [8], a cardiac signals provisioning system for patients' heart telemonitoring in the TeleCardio project [2]. The developed system serves then as a context data wrapper to the Infraware platform [13] - a middleware for supporting services to context-aware mobile applications. In fact, as previously mentioned, Healthcare constitutes a rich field of application due to its diversity of data and devices. The applicability of the architecture is also discussed in other pervasive scenarios.

SAC'08, March 16-20, 2008, Fortaleza, Ceará, Brazil.

The paper is organized as follows. Section 2 brings in sensor data acquisition issues; Section 3 introduces the SOD methodological support we have used; Section 4 presents the proposed architecture; Section 5 states a proof of concept in the Healthcare domain; Section 6 shows the applicability of the architecture in three other scenarios; Section 7 discusses our proposal based on the methodological support we have used and Section 8 discusses related work; lastly, Section 9 concludes the paper.

2. ASPECTS OF SENSOR DATA ACQUISITION

A domain is characterized by a set of problems or functions that applications related to it should address. In this work, we deal with the sensor data acquisition domain, which comprises aspects of sensors communication, data processing, persistence, conversion into a desirable format and delivery for data consumers. Following we introduce a discussion on these core technical challenges as a result of a domain analysis activity.

2.1 Sensor Communication

Obtaining data from sensor devices by computer communication is a natural requirement in pervasive computing scenarios. However, in these scenarios, it has a different purpose than in traditional computer communications. In the latter case, communication involves typical interconnection between computers, printers, routers, etc, by which a substantial quantity of data is bi-directionally transmitted. In this case the emphasis is on higher transmission rates for supporting faster downloads (e.g. of multimedia data) by final users; these aspects are addressed by IEEE 802.11 and 802.15.1 standards (e.g. under Wi-fi and Bluetooth technologies respectively). In contrast, sensor data transmission mostly involves control and monitoring as it is required by context-aware systems and wearable computers; these aspects rather are addressed by the IEEE 802.15.4 standard (e.g. under ZigBee [22]). It is quite common in this case that sensor devices integrate each other in a local wireless sensor network, whereby desirable requirements are reliability, adaptability, latency and scalability [18], as opposed to transmission rates up to 11 or 54 Mbps. That is because sensor devices are committed to low transmission rates to keep as far as it is possible their batteries. In case the sensor data provisioning system is embedded in the sensor device (e.g. a wearable system), this kind of sensor communication does not make sense anymore.

2.2 Data Processing

The context data obtained from sensor devices may be processed to carry out some brief data handling and/or to perform data analysis in order to enhance its semantics. In both cases the data processing depends of whether we have discrete or continuous data. The latter (e.g., waveform) calls for signal processing on account of noise filtering and/or patterns recognition. As an example, consider the electrocardiogram (ECG) signal processing such as in the work of Andreão [1]. Discrete data as location and temperature, otherwise, calls for the usage of statistical functions on data samples, e.g., an average or standard deviation in given time window, to obtain a cue, i.e., a reference value, reducing this way, the amount of data obtained from sensor devices as well as to increase the reliability of the data acquired [17]. It is important to consider, however, that the data processing function introduced here deal with a single piece of data acquired by a single sensor box. Therefore, it is an orthogonal function to context reasoning typically performed by context-aware middleware that combines multiple pieces of information to infer a new one. In fact, data processing transforms sensor data to contextual information at a higher abstraction level enabling it for direct context interpretation by combining multiple pieces of information. An example of such data processing feature can be found in [11].

2.3 Data Wrapping

The wrapping issue constitutes to encapsulate the context data in an appropriate format. Considering pervasive scenarios and thus, the need for data delivery over a communication channel, the design of such model has to take into account the following non-functional sub-requirements: *(i)* interoperability between heterogeneous systems; *(ii)* flexibility, for allowing minimal effort in modifications; *(iii)* lightness of data as quite as possible, for reaching efficient transmission; *(iv)* truthfulness, for being consistent with the real world domain which is captured by it; and *(v)* readability, for permitting evaluation of its domain experts. Indeed, this model is related to an abstract and to a transfer syntax by means of (*i*, *ii*, *iv* and *v*) and *(iii)* requirements respectively. These two perspectives focus on data representation and data transmission and are addressed in the OSI reference model, for example, by the application and presentation layers. The XML technology is quite suitable for addressing this requirement. Despite it does not meet the transfer syntax sub-requirement due to the rather large size of XML files, this drawback may be avoided through a compression procedure for reaching size reduction of the XML document as it is showed by Erfianto in [6].

2.4 Data Persistence

Sensor data have often spatio-temporal aspects to be considered, as far as user profile, and different abstraction levels as a result of data processing and/or context interpretation. In addition, context data is acquired from different devices, and have diverse types and heterogeneous formats. Therefore, their persistence has to be done in such a way that client applications could later have access to them in a standardized manner, remarkably by remote queries through the Internet. The XML format may also be used for sensor data persistence. It can be mapped into a tabular model for reaching applications' querying support. In fact, it is suitable not only for interoperation over the Internet but also between heterogeneous platforms in general. However, the spatio-temporal aspect of sensor data asks for a special management as discussed by Sashima et al. [16]. A suitable management of this sort of data is, indeed, an open topic of research.

2.5 Graphical User Interface

For usability, it is rather common in context-aware systems the acquisition not only of implicit data obtained from sensor devices, but also of explicit data obtained from graphical user interfaces (GUI). By means of a GUI the user can insert either personal data or some context related data to combine with data acquired from sensor devices. The explicit data should be encapsulated jointly with implicit data in a model that copes with the issues just mentioned (in the two previous subsections).

2.6 Data Delivery

As a rule, a sensor data acquisition system is part of a larger telematics system including middleware platforms and end-user applications. These components are usually physically distributed. That is why the data acquired by the acquisition system has to be delivered over a communication channel to its data consumers, whether they are directly final applications or a context-aware middleware. On one side, because we are speaking of context-awareness, these data should be delivered not only on consumer requests, but also on event in case there is relevance for one or more consumers. The RMI Java standard can be used to this end (i.e., remote procedure calling). It allows event-driven messages coming from the data provider to listener applications. On the other side, due to the regular configuration changes, this delivery function should maintain weak coupling between data providers and consumers. Moreover, the delivery service, in general, should be public whether in the Internet or at least in the environment monitored by the sensor devices to allow consumer applications to discovery and to use it. These are the reasons why, jointly with RMI, web services (WS) technologies [20] sound suitable for addressing such a delivery service, rather than, e.g., TCP/IP. In fact, a WS deploys the data available on the web through a standardized interface for answering then data requests coming from applications. In contrast, TCP/IP tightly connects the data provider and consumer inhibiting thus an open access to sensor data such as it is required in many ubiquitous scenarios.

3. THE SOD METHODOLOGICAL SUPPORT

In face of the issues just mentioned, in the domain design phase we have used a SOD methodological support to propose a service architecture for sensor data provisioning. The methodology consists of *(i)* using ISDL, *(ii)* validating the architecture by using a tool, and also *(iii)* the adoption of a design-quality metrics.

The ISDL modeling technique [14] is aimed at modeling systems at higher abstraction levels. In ISDL, a model comprises two viewpoints: an entity model and a behavior model, to describe the system respectively in its structural and behavioral aspects. In this way, a service architecture is presented in terms of an ISDL entity model that expresses an external perspective of the whole system as well as by a behavior model (see Figure 1) of each layer for expressing each (inter) action that occurs either in itself or at its boundaries. Moreover, despite the causality relations connection (inter) actions, each of these (inter) actions may be characterized by three attributes: (i) information – what is produced by the (inter) action, (ii) time – the time the result of the (inter) action becomes available (if successful), and (iii) location – where the (inter) action takes place. In this paper these attributes are expressed by the variables μ, π, and λ respectively. A tutorial on ISDL can be found at [10].

Since we are talking about a formal language, we could reach preciseness, unambiguity and clarity. For the same reason, we are able to validate such architecture with respect to consistency of the system behavior. We can then use AMBER, a business process design language based on ISDL that has tool support in Testbed Studio [5]. Finally, let us consider the following objective criteria advocated by Vissers et al. to assess the quality of a service design [19]:

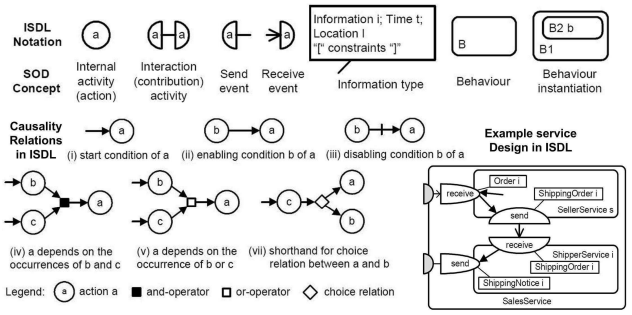

Figure 1. Part of the ISDL behavior metamodel [14].

- *Generality*: a function should be designed in its most general form. Thus, specific cases should be covered via instantiation of a general function by either adding restrictions and/or parameters settings.
- *Propriety*: do not introduce what is immaterial, i.e., functions which have no direct call for.
- *Orthogonality*: to strive for separation of concerns, i.e., to define separate functions for addressing independent needs.
- *Parsimony*: to design a single general function for addressing each requirement, rather than collapsing multiple functions for the same requirement.
- *Abstraction*: to focus on an abstraction level suitable for the goals of the phase or step of the design process, leaving out details that were deemed irrelevant in such a phase.
- *Open-endedness*: the service design can be easily extended in a later stage with no damage to its original design.

These criteria, in fact, sound suitable as an evaluation metrics for SOD as much as for architecture design in general. On the next section we present the *Context Wrapper* architecture. This architecture may be instantiated for several sensor data types for carrying out data acquisition from a sensor device, data processing, wrapping, persistence and delivery.

4. CONTEXT WRAPPER

The Context Wrapper architecture has a layered design that hierarchically organizes the system's functions. As it is norm in this architectural style, each layer should provide service to the adjacent layer above by building upon the services offered by the layer adjacent below. Following, we introduce both the entity model and the behavior model of Context Wrapper.

4.1 Context Wrapper Entity Model

By using an entity model we can represent the system parts and their relationships by means of the structure of entities interconnected by interaction points. Figure 2 shows the Context Wrapper entity model. The Context Wrapper is composed by the following entities:

- *Sensors Communication* layer (SC): lays up communication protocol with a sensor device;
- *Data Processing* layer (DP): carries out data processing including data filtering and/or cueing;
- *Graphical User Interface* (GUI): allows user to insert explicit data (this is an optional entity);

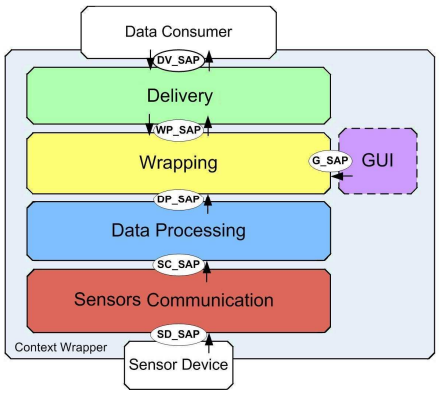

Figure 2. Context Wrapper entity model.

- *Wrapping* layer (WP): encapsulates context data (and possibly explicit user data) into a suitable format, and subsequently, performs data persistence;
- *Delivery* layer (DV): committed to context data delivery to consumers (either middleware or directly final applications).

Each Context Wrapper layer abstracts functionalities from its lower layer to its upper layer by means of the service it provides. The Sensor Device (SD) and Data Consumer (DC) entities interact with Context Wrapper entities at their boundaries to supply and consume sensor data, respectively. These interactions as much as the internal interactions take place at service access points (SAP) through service primitives (SP).

4.2 Context Wrapper Behavior Model

Complementing the entity model, by means of a behavior model we can state the functionally of the identified system parts and how they interact [19]. In this way, we present a constraint-oriented model of the Context Wrapper behavior, which is the combination of its sub-behaviors. This approach focus on the interaction contribution of each entity involved in an interaction.

Following, the behavior of each system layer is introduced as well as its possible configurations that meet specific needs. It is worth to remark, however, that the system layers may provide either a connection-less or connection-oriented service, which may be confirmed or not. Such a choice depends on the type of sensor data as much as the specific purpose of each application scenario. For brevity, we do not consider here service primitives neither concerned with connection establishment and release nor data confirmation, leaving the choice for such configurations open for the designer of specific systems. We then can focus on the major functions of the Context Wrapper entities. Analogously to the OSI model, the Context Wrapper architecture as a whole may be seen as three sets of layers: lower level layers, i.e., SC and DP; convergence layer, i.e., WP; and upper level layer, i.e., DV.

4.2.1 Lower Level Layers Behavior

The lower level layers, i.e., SC and DP, provide services committed to lower abstraction level functions such as hardware interconnection and data processing. The SC function is just passing *ContextData* data units from SD entity to DP entity (Figure 3): one *Dreq1* data request triggered at *SD_SAP* enables one *Dind1* data indication at *SC_SAP*. When *Dind1* arrives at

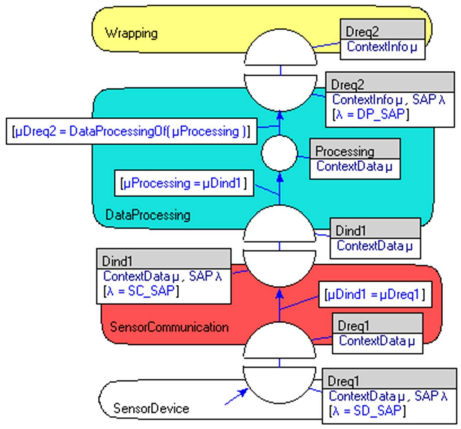

Figure 3. Behavior of the lower level layers.

SC_SAP, the *Processing* action calls a *DataProcessingOf* function. Thereby, sensor data assumes an information character on account of its enhanced semantics and also of the checking if an event has taken place. Then, one *Dreq2* SP takes place at *DP_SAP* to move the *ContextInfo* data unit to WP entity.

Notice at the bottom of Figure 3 that the interaction between SD and SC has a µ information attribute of the *ContextData* type and a λ location attribute of the *SAP* type. The latter is shown only in one entity of the interaction. Furthermore, a constraint on the arrow in SC entity express that the *µDind1* information result of the interaction between SC and DP must receives the *µDreq1* information result. A similar constraint must be satisfied for the arrow that enables a *Processing* action as far as for the arrow that enables an interaction between DP and WP entities.

4.2.2 Convergence & Upper Level Layers Behavior

The convergence layer handles sensor data flow from lower level layers to the upper level layer, as well as the upper level layer requesting for data. As a client, WP layer is connected to DP layer through *DP_SAP*, whereby the former receives data units (*ContextInfo*) from the latter through the *Dreq2* SP, see Figure 4. Thereafter, data units are embedded into a suitable format by the *WrapData* action and then stored by the *StoreData* action. In the meantime, WP may be receiving user data from GUI entity at *G_SAP* by means of *Dreq3* SP. This information is also wrapped into a suitable format and stored in the same way as context information. In case context information is flagged true for an event (it may be just the end up of a time cycle) WP plays the role of provider and the *WrapData* action also moves these data (*ContextInfo* and *UserData*) to *WP_SAP* by means of *Dreq4* SP. Hence, *Dreq4* triggers a *Dind3* indication at *DV_SAP*, which connects DV layer to data consumers. Notice in Figure 4 that either an interaction between DP and WP through *Dreq2* SP or an interaction between GUI and WP through *Dreq3* SP enables a *WrapData* action as long as constraints are satisfied.

From a top-down standpoint, WP as a provider may receive *Dind2* indications of *Dreq5* data requests came from data consumers through DV layer. Then, WP performs a *RetrieveData* action to move the data requested to *WP_SAP* by means of a *Dreq4* SP to be in turn delivered to data consumers by a *Dind3* SP at *DV_SAP*, as illustrated by Figure 4. On the next section, the architecture just presented is instantiated on the design of the ECG Wrapper, an

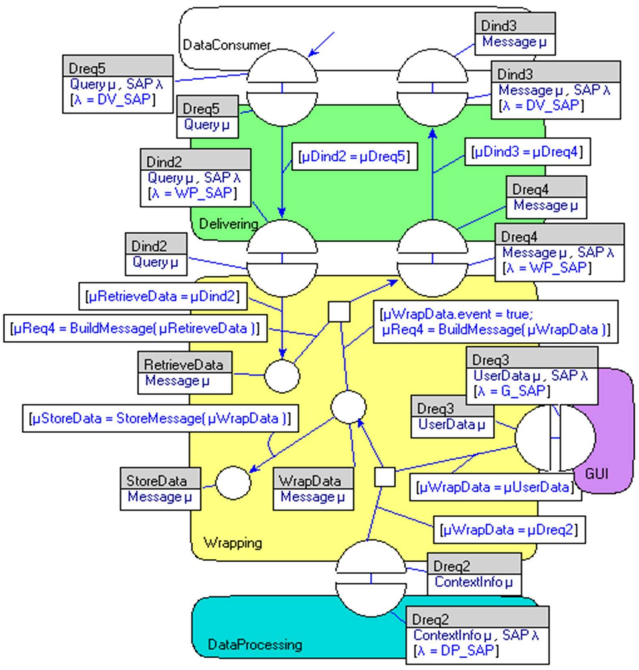

Figure 4. Behavior of the WP and DV layers.

ECG data provisioning system. This system is applied for patients' heart telemonitoring in a real scenario in the scope of TeleCardio project [2]. We elaborate on the derivation of ECG Wrapper from the Context Wrapper architecture in order to validate our proposal.

5. ECG WRAPPER: A CASE STUDY

The ECG Wrapper design took into account specific requirements related to ECG signal provisioning with telemonitoring purpose as follows. A mobile sensor device (e.g. Holter monitor) can acquire ECG signal obtained from electrodes placed at the body surface of a patient. In TeleCardio, this device must transmit this ECG data to a computer nearby where the ECG Wrapper system shall be built on. This data transmission should be carried on over a wireless link for giving mobility for the patient. Then a signal processing must take place in order to reach noise filtering and to perform ECG analysis. This signal processing can detect abnormal events on the patient's heart activity and also support physicians' decision making. Thereafter, the ECG data should be embedded into an appropriate platform-independent format possibly in combination with user data obtained from a GUI. Finally, this populated model should be delivered to data consumers, i.e., healthcare applications. There are multiple technology solutions to meet those requirements. The ECG Wrapper architecture may be directly derived from Context Wrapper by addressing its layers with the chosen technologies.

5.1 ECG Wrapper Entity Model

The ECG Wrapper entity model resembles the Context Wrapper one (Figure 2). Nevertheless, although it has the same entities, they are addressed by specifically technology solutions as follows:

- The Sensors Communication (SC) layer is addressed by ZigBee [22], a wireless communication protocol based on the IEEE 802.15.4-2003 Low Rate WPAN standard. As we pointed out in Subsection 2.1, this technology is suitable for monitoring and remote control purpose, i.e., low data transmission rate and low power consumption.
- The Data Processing (DP) layer comprises an ECG analysis system [1] that carries out the reception and processing of the raw signal through an ECG segmentation and classification approach based on hidden Markov models (HMM). In doing so, this system reaches noise filtering and event detection.
- A GUI entity allows user data acquiring such as patient anamnesis and recording session parameters configuration.
- Wrapping (WP) layer encapsulates ECG processed data as well as patient and recording session data into the *ecgAware* model [7], an ECG XML-based markup language. Once these data are embedded in XML files, they are locally stored.
- Delivery (DV) layer is addressed by the RMI Java standard for remote procedure calling (e.g., "log on Infraware platform", "an event has been detected") and by the WS technology [20] for making ECG data units (i.e., XML files) available on the web.

The sensor device (SD) constitutes a mobile Holter monitor equipped with a ZigBee radiofrequency (RF) transmitter. The ECG data is consumed by a health context-aware application through intermediation of the Infraware platform [13].

5.2 ECG Wrapper Behavior Model

Analogously to the entity model, the ECG Wrapper behavior model is straightforwardly derived from the Context Wrapper behavior model. However, on the ECG Wrapper design we can bring in an enhanced specification of the service provided, by presenting a more tangible behavior description. This is presented in the sequel by following a temporal ordering of one of the TeleCardio scenarios, whereby an ECG recording session is monitored remotely by a health context-aware application.

Before the beginning of an ECG recording session, a user such as a healthcare professional may insert patient personal data and electronic patient record data through a GUI. In addition, the user may provide some data about the recording session. Examples include the expected duration of the session, blood pressure acquired up to that point and so on. Those data are then embedded into the *ecgAware* format and stored. The system procedure for those operations has the same behavior as the one depicted in Figure 4, i.e., the Context Wrapper behavior.

At this point, the recording session can be started; during the first 30 seconds of ECG data acquisition, the SC service connects the SD and DP entities. Such a connection establishment is required because, in this scenario, ECG data units should be transmitted from the sensor device to SC layer at each 30 seconds, which is a short time interval that justifies keeping resources reservation. Once the connection is established, the data transfer phase is initiated, in which data request, indication and confirm SPs are performed for moving data from SD to DP through a confirmed service. A data confirm following each data indication is required taking into consideration the ECG data type, i.e., continuous data such that it only has a meaning as a whole, and therefore, all data units must arrive at its destination. With regard to data transfer,

the ECG Wrapper behavior is also more specific than the Context Wrapper one. In this case, we have to consider a timeout constraint stating that the *Dind1* at *SC_SAP* has to happen at most *Δmax* time units after *Dreq1* has happened on *SD_SAP*. This is because SC entity provides a confirmed service. The DP layer service is also connection-oriented and links SC and WP entities. DP works on demand by processing data units received at *SC_SAP* and moving them to WP entity through *DP_SAP*. Except by the timeout constraint, SC and DP behaviors in ECG Wrapper are the same as in Context Wrapper (Figure 3).

As a result of the *Processing* action, an ECG data unit is segmented and classified. Abnormal events can then be detected in order to identify an emergency situation. Thereafter, a *WrapData* action encapsulates such an ECG information unit into an *ecgAware* instance, stores it by means of a *StoreData* action, and forwards it to DV layer. While this whole procedure is carried out for each ECG data unit, one or more data requests may come from the DC entity asking for pieces of information kept in a data repository at the WP layer. The whole behavior related to the convergence and upper level layers in ECG Wrapper follows directly the Context Wrapper behavior shown by Figure 4. At the end of the recording session all ECG information units stored in the WP repository are aggregated accordingly to *ecgAware* model into a single ECG record. Also at this point, a termination phase eliminates the connection between SD and DP entities.

We have implemented ECG Wrapper in the context of TeleCardio project. It supplies a client web-based application with the ECG data obtained from a patient during recording sessions of 48h duration. This application is aware to the patient's context, changing its behavior as a reaction to event indications triggered by the ECG Wrapper.

6. APPLICABILITY OF THE CONTEXT WRAPPER IN OTHER SCENARIOS

The Context Wrapper architecture may be instantiated in several mobile and pervasive computing usage scenarios, whereby the same recurrent sensor data acquisition requirements take place. A representative set of such scenarios are:

i. For building an intelligence environment such as eye-tracking system [21], several video cameras may be distributed to set up a network that captures local images. From those cameras, low level data such pixels information may be acquired through *sensors communication*; thereafter, data analysis by means of *image processing* techniques is required. They may be either simple algorithms for light and color measuring to infer that one room is lighted or complex methods to detect movement, conversation and nearby people or objects in order to infer whether a meeting is taking place. In both cases there is a need for *wrapping* and *delivering* such contextual information for end-applications or even for an infrastructure of additional computational resources for reasoning, such as a computer cluster.

ii. Consider the application of pervasive computing either for tourism guide, or user assistance in fieldwork environments such as biodiversity surveys or archeological excavation [12]. In such scenarios, the following requirements are common: *(i)* the need for acquiring user location through a satellite navigation system, i.e., *sensors communication*; *(ii)* to carry out an average of recent acquired values, as well as to process the cue value obtained for mapping it into a coordination system, e.g., a chart (*data processing*); *(iii)* to *wrap* this contextual information possibly in combination with user inserted data according to a truthful model of the domain; and *(iv)* to *delivery* this populated model through a communication channel, e.g., the Internet, for its data consumer.

iii. For user assistance in entertainment scenarios such as cinemas and theaters, it is quite useful, for example, monitoring of light and sound in the environment for several inferences [17]. Therefore, contextual data has to be acquired from sensors by an interface, and then subsequently processed in order to map it into a discrete scale, embedded into one suitable model and delivered for its listeners.

These scenarios exhibit the quite generality of the proposed architecture constituting a subset of the domains in which the Context Wrapper can be employed.

7. DISCUSSION

Let us look at how the Context Wrapper architecture design fits in the design-quality principles we have adopted:

- *Generality*: the architecture functions are, in fact, quite general for handling sensor data provisioning issues. We have demonstrated this concern on the ECG Wrapper case study in Section 5 by adding constraints and configuring parameters as well as in Section 6 by discussing some scenarios.
- *Propriety*: with sensor data acquisition requirements discussed in Section 2 in mind, we did not introduce in the Context Wrapper functions which have no direct call for.
- *Orthogonality*: the proposed architecture conveys separation of concerns; we designed a single layer for addressing each independent issue. One could state that data persistence and wrapping were collapsed into the same layer. However, this design choice reflects the close relation of these two issues that, indeed, were addressed by the *StoreData* and *WrapData* different functions, respectively.
- *Parsimony*: we have designed a single general function for addressing each requirement, rather than designing multiple alternative functions for the same requirement.
- *Abstraction*: we focus on a high abstraction level of the system (see Figure 2), leaving out details that were deemed irrelevant in such a phase of the design process. In fact, this abstraction level we have adopted is in consonance with such a domain engineering approach, which has a more general scope than a single application.
- *Open-endedness*: the Context Wrapper design may be extended either at a later stage or for the design of specific cases with no jeopardizing its original design, as we have demonstrated in the ECG Wrapper case study.

We also have used the tool support of AMBER to check consistency in the Context Wrapper behavior. The simulation we carried out has also validated the architecture, see Figure 5. Such simulation has allowed us to follow the architecture workflow in

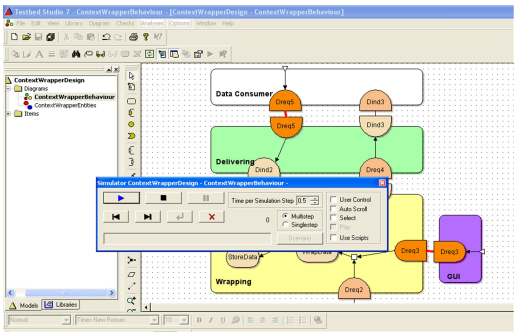

Figure 5. Snapshot of Context Wrapper behavior simulation.

order to see whether deadlocks take place or even to evaluate cost paths looking for optimization.

8. RELATED WORK

There quite a few research initiatives in mobile, pervasive and context-aware computing literature indeed correlated with our proposal. That is to say, there are few works that tackle in an integrated way basic tenets of sensor data acquisition such as the issues we outlined in Section 2. The work of Dey et al. [4], for example, has been very influential to our research. However, it focuses on context usage, leaving out context acquisition itself still quite vague. Pascoe's work [12], likewise, provides a worth contribution to the field, but it does not elaborate on details the sensor data acquisition; his proposal neither is application-independent. Raento et al. in turn tackles the non-trivial problem of providing a quite general architecture (including a sensor data acquisition module) for off-the-shelf hardware [15]. However, only few sensor data types (the ones which cellular phones are able to sense) are considered. Overall, the platform proposed in [15] constitutes a very human-centered software artifact specific for smartphones. In our work, otherwise, we have strived for proposing an architecture focused on the challenge of sensor data acquisition and provisioning itself whatever are the final-applications purposes, scenarios or devices used.

9. CONCLUSIONS

Our work has used a domain engineering approach to address the sensor data acquisition problem. As a result of the analysis and design phases, we have compiled a requirements specification and proposed a sensor data provisioning architecture by using a SOD methodological support. Both contributions cover a gap in literature w.r.t. systematic methods for sensor data acquisition that supports context-aware mobile applications. Context Wrapper is a service architecture that meets sensor data acquisition requirements. By using such an architecture either through service configuration or extension, we can reach a rapid prototyping of specific sensor data provisioning systems. This feature was shown in a case study involving ECG data acquisition in a real scenario. Furthermore, we have discussed the applicability of Context Wrapper in a representative set of pervasive scenarios. Future work includes investigating how to add autonomic computing tenets to Context Wrapper architecture.

10. ACKNOWLEDGMENTS

This research has received financial support from FAPES (grant no. 31024866/2005) and CNPq (grant no. 50.6284/04-2). We would like to thank João Paulo Almeida for his valuable comments and careful reading of this manuscript.

11. REFERENCES

[1] Andreão, R. et al. ECG Signal Analysis through Hidden Markov Models. *IEEE Transactions on Biomedical Engineering*, vol. 53, n. 8, 2006.

[2] Andreão, R., et al. TeleCardio - Telecardiologia a Serviço de Pacientes Hospitalizados em Domicílio. In *X Brazilian Conference in Health Informatics (CBIS'06)*, Florianópolis, Brazil, 2006.

[3] Chen, H. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. Ph.D. Thesis, University of Maryland, USA, 2004.

[4] Dey, A., et al. *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications*. Human-Computer Interaction Journal, 16(2-4):97-166, 2001.

[5] Eertink, H, et al. A Business Process Design Language. In *World Congress on Formal Methods (1)*, pp. 76-95, 1999.

[6] Erfianto, B. *Design of a Vital Sign Protocol Format Using XML and ASN.1*. M.Sc. Thesis. University of Twente, The Netherlands, 2004.

[7] Gonçalves, B., et al. EcgAware: An ECG Markup Language for Ambulatory Telemonitoring and Decision Making Support. In *Proc. of the International Conf. on Health Informatics*, Funchal, Portugal, 2008.

[8] Gonçalves, B., et al. ECG Data Provisioning for Telehomecare Monitoring. In *Proc. of the 23rd ACM Symposium on Applied Computing (SAC'08)*, Fortaleza, Brazil, 2008.

[9] Gu, T., et al. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 2005.

[10] IDSL home. http://isdl.ctit.utwente.nl/, n.d.

[11] Laerhoven, v. K. *On-line Adaptive Context Awareness Starting From Lowlevel Sensors*. Ph.D. Thesis, University of Brussels, 1999.

[12] Pascoe, J. Adding Generic Contextual Capabilities to Wearable Computers. In *Proc. of the 2nd IEEE International Symposium on Wearable Computers (ISWC'98)*, pp. 92-99, Pittsburgh, PA, 1998.

[13] Pereira Filho, J. G., et al. Infraware: um *Middleware* de Suporte a Aplicações Móveis Sensíveis ao Contexto. In *Proc. of the 24th Brazilian Symposium of Computer Networks*, Curitiba, Brazil, 2006.

[14] Quartel, D., et al. Methodological Support for Service-oriented Design with ISDL. In *Proceedings of the 2nd International Conference on Service-Oriented Computing (ICSOC)*, 2004.

[15] Raento, M., et al. ContextPhone: a Prototyping Platform for Context-aware Mobile Applications. *IEEE Pervasive Computing*, 51-59, 2005.

[16] Sashima, A., et al. Spatio-Temporal Sensor Data Management for Context-aware Services. In *Proc. of the 1st International Workshop on Advanced Data Processing in Ubiquitous Computing*, 2006.

[17] Schmidt, A.; Laerhoven, K. *How to build smart appliances?*. IEEE Personal Communications, 2001.

[18] Tilak, S., et al. A taxonomy of wireless micro-sensor network models. In *Proceedings of the ACM Workshop on Wireless Security*, ACM Press, pp 28-36, 2002.

[19] Vissers, C., et al. *The architectural design of distributed systems*. Lecture Notes, University of Twente, The Netherlands, 2000.

[20] WS architecture. W3C website: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#id226089.

[21] Zhai, S. What's in the eyes for attentive input. *Communications of the ACM*, 46(3), 2003.

[22] ZigBee Specification. Available at http://www.zigbee.org/.