Evaluating Data Modeling Aspects for Home Telemonitoring

Vilmos Szűcs
Department of Software Engineering
University of Szeged
Hungary
vilo@inf.u-szeged.hu

Ádám Zoltán Végh
Department of Software Engineering
University of Szeged
Hungary
azvegh@inf.u-szeged.hu

Miklós Kasza
Department of Software Engineering
University of Szeged
Hungary
kaszam@inf.u-szeged.hu

Vilmos Bilicki
Department of Software Engineering
University of Szeged
Hungary
bilickiv@inf.u-szeged.hu

***Abstract*—This paper addresses the evaluation of data models designed for Home Telemonitoring to store various data coming from a diverse set of devices on the grounds of two previously developed Telemonitoring Systems. The evaluation is based on quality metrics, measurements, and empirical validation, and it identifies the key differences between a generic and a problem specific data model regarding their main advantages and drawbacks.**

Keywords-telemonitoring; data modeling; data model metrics

I. INTRODUCTION

In recent years, the technical conditions have been changed and the evolution of information technology has made a wide range of affordable devices, almost infinite processing and storage capacities available. Wireless penetration and wireless based communication have also gained significant ground. This trend has led to the development of several embedded and sensor based systems. In the field of the e-Health domain, Telemonitoring Systems started to emerge as an adaptation of these technologies with the objective of providing an efficient basis for home care services [1][2][3][4]. The main goal of these systems is to overcome the typical problems of health care services. On the one hand, they offer a solution for collecting several kinds of physiological data at the patients' home without the need for medical supervision. On the other hand, with the help of data mining and signal processing algorithms they process, sort, and aggregate measured data to transfer and visualize the right information, at the right place, at the right time for medical experts or even for relatives.

To improve the quality and cost effectiveness of health care services, the necessary devices and sensors must be selected carefully to keep the system available for a low price. However, this can lead to dealing with a diverse set of hardware manufacturers and communication protocols. In addition, the various structures of medical data coming from the involved devices must be supported by the system. These issues all affect the design and development of the data model. Furthermore, the data model has an impact on the flexibility, the reusability, and the performance of the developed system. In general, the data modeling process is one of the most critical parts of the design phase in a development process [5].

In the last few years, we have developed two Telemonitoring Systems. The key differences between these Research & Development (R&D) projects were the budget and time constraints, which also affected the requirements and functionalities expected from each system. Regarding these constraints different data models were developed; a generic data model for the long-term project and a problem specific one for the mid-term project in order to keep up with the productivity requirements. The long-term project also provided a generic data visualization, while the mid-term project included accurately defined user interfaces for the doctors. Both systems were evaluated in a Living Lab (LL) experiment with the involvement of real patients and doctors. In this paper, we will examine each data model regarding metrics and measurements, and we will summarize the observations referring to user acceptance and satisfaction relying on the LL tests.

The following section gives a short overview of a typical Telemonitoring System and the requirements for its dataflow and data model. The next part of the document describes data modeling principles and metrics to define what makes a data model good and how we can measure its propriety and quality. The forthcoming section introduces the examined data models. The next part aims to present the comparison and evaluation of the models, which is followed by the analysis of user acceptance. Finally, the paper ends with a summary of the results and a conclusion on the different data modeling aspects presented in this paper.

II. THE ROLE OF DATA MODELS IN TELEMONITORING

Telemonitoring Systems are designed to help with the collection of different physiological parameters of people suffering from various diseases. They offer a remote care solution for medical experts and support the communication between the patients and their care providers while the patients can stay at home. To reach its goals, the system includes a diverse set of active and passive sensors, and a so-called Hub placed in the home of the patient. The communication between the sensors and the Hub is prevalently based on wireless technology [6], mainly over bluetooth or zigbee connections. The Hub is responsible for managing sensors, collecting measured data and transmitting these data to a central data

 ISBN: 978-1-61208-158-8

server, which affords a set of user interfaces for doctors to trace the patient's health conditions on a frequent basis. The system can provide data mining and signal processing solutions running on the central server side. However, if the process capacities meet to requirements for these algorithms, the Hub can also be used for such purposes. The central data server can provide various set of interfaces for other integrated 3rd party systems. A typical Telemonitoring System and its dataflow are illustrated in Figure 1. As it can be seen, several kinds of data model are presented in a Telemonitoring System. In general, different data models are used on the Hub and on the server side although they work with the same set of data. Developers also have to deal with the mappings to data models of integrated systems. In this paper, we will focus on the data model of the central server, which provides the basis for the communication and the dataflow between the server and the Hubs, the user interfaces and the integrated systems.

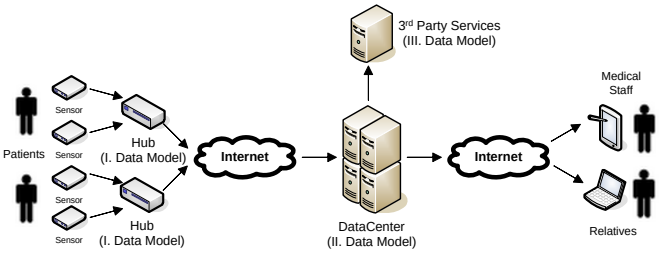

Figure 1. Dataflow in Telemonitoring Systems

Although all Telemonitoring Systems follow this common structure, a variance can be observed between them. This is mainly caused by the fact that these systems have different target diseases to monitor. In addition, for each disease a well-defined and peculiar set of sensors are needed to be integrated to the system. The discrepancy of the devices can lead to disease or sensor specific data models [7]. Accordingly, the integration and expansion of these systems is difficult or even impossible. On the other hand, a generic data model that does not contain such constraints is hard to design and develop because a fully detailed specification is required. This specification is time-consuming and in several cases the assumptions of the projects do not allow such pursuits.

These considerations reveal how difficult it is to design a sophisticated data model and as such, data modeling is one of the most critical proportions of the software design phase, although it signifies only a minor part of the total development effort. The data model also has an impact on system flexibility, reusability, implementability, performance, and on integration with other systems. A not warily designed data model results in various functional deficiencies of the final system. Modifying the requirements or adding previously unidentified requirements to the system specification in the later stages of the software development process costs multiple times more and it can dramatically increase the overall development costs. The completeness and the quality of data models are particularly important in Telemonitoring Systems, where in production usage human health or even human lives are at stake.

III. DATA MODELING PRINCIPLES AND QUALITY METRICS

In practice, it is relevant to differentiate the data model, which is the final product and data modeling, which is the process used to build the final product. In spite of both are substantial, improving the process quality results in a higher and more sustainable data model quality level, as it concentrates on defect prevention rather than detection. Moreover, this also means that a good data modeling process can reduce costs because data quality issues can be resolved at the earlier stages of the development [8]. A few process proposals were laboured out, but they were mainly advised as a toolkit for data modeling experts, rather than as a rigid guideline to be followed [5][9]. Although a set of data modeling patterns are represented in [10] based on real systems, only the semantic and conceptual correctness of these models are defined, the quality of them regarding numerical metrics is ignored. As it says, “patterns are a starting point, not a destination”. However, if the main principles are observed and enough time is allocated for analyzing not only the adaptable patterns and the data model but also the modeling process, a better data model quality can be achieved.

A continuous assessment by quality metrics is required to ensure the best data model quality during software development. The metrics help improve the quality of the data model, choose between alternative models, and improve the modeling process. Several metrics were laboured to measure data model quality [11]. One of the most comprehensive collections is assembled in [12], where 29 metrics were identified and organized around 8 main Quality Factors (Figure 2). Each Quality Factor also refers to a responsible group of stakeholders who are involved in evaluating the metrics related to the given factor.

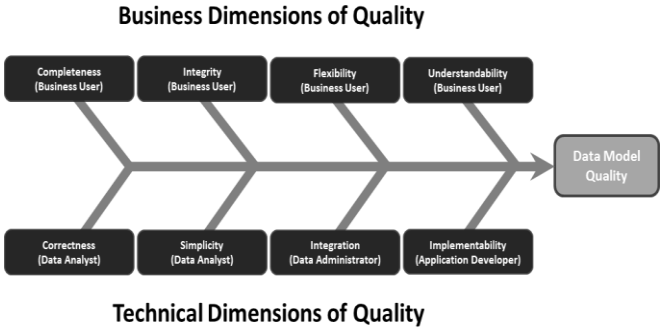

Figure 2. Data Model Quality Factors [12]

The definitions of the quality factors defined in [12], which were the ground for evaluating our models, are:

- Correctness was defined as whether the model conforms to the rules of the data modeling technique (i.e., whether it is a valid data model).
- Completeness refers to whether the data model contains all information required to support the required functionality of the system.
- Integrity is defined as whether the data model defines all business rules, which apply to the data.

 ISBN: 978-1-61208-158-8

- Simplicity means that the data model contains the minimum possible entities and relationships.
- Flexibility is defined as the ease with which the data model can cope with a business and/or regulatory change.
- Integration is defined as the consistency of the data model with the remaining data of the organization.
- Understandability is defined as the ease with which the concepts and structures in the data model can be understood.
- Implementability is defined as the ease with which the data model can be implemented within the time, budget, and technology constraints of the project.

Although these quality factors define a broad set of metrics for evaluating data models, in [13] these quality factors were used extensively in real scenarios and their usefulness was measured, where the empirical validation proved that only three of them were subservient in practice. In addition, two new metrics the reuse level and the number of issues by quality factors were found. Furthermore, sometimes the subjective ratings of the data modeling experts were more useful to comprehend overall data model quality, and the form of textual descriptions about quality issues disclosed a better view of current defects as these were listed the issues themselves rather than a quantitative measurement. In theory, metrics can measure quality, but in practice they are not definitely useful and that is why the opinions of data modeling experts about the importance of each quality metric are divided. The only principle that is accepted by a wide group of experts is the following: the usefulness of a quality metric can be justified if its added value is more than the cost of the effort to measure it.

According to our Telemonitoring Systems and their Living Lab based experiments, we could measure some other, but relevant metrics. These were performance, productivity and user satisfaction.

IV. EXAMINED DATA MODELS

In this section, the two examined data models will be presented. We will focus on a small but critical point of the overall models that can be easily compared as it is responsible for the same role in the same domain: storing and managing medical data in a Telemonitoring System. We illustrate each model using UML class diagram notation, as the structure of them is more relevant in our case than their semantic meanings.

The long-term project is called ProSeniis [14][15] and the mid-term project is called Medistance [16]. In both systems a broad range of sensors were integrated and several kinds of data were stored. In the ProSeniis project the final system had more than 150 user interfaces and ~170000 lines of code provided the overall functionality. 155 entities were responsible for storing all kinds of data. Several developers were involved in the development of the system for a two-year time period and the average of allocated person-months (PMs) per functionality was 1,5. Regarding the Medistance project, ~65000 lines of code and ~50 user interfaces were laboured out in about half a year (0,7 PMs/functionality). In the persistence layer of the system 41 entities were defined to store the data. Both persistence layers were designed and developed on the top of Hibernate [17].

Each project was managed with a project management tool, called Trac [18]. In the Trac system the full development processes were tracked in the form of structured tickets commented on a daily basis. Moreover, this kind of reporting was also helpful to evaluate the differences in productivity that contains design, development, and maintenance as well. Since we have finished the two mentioned projects, we have laboured out a more sophisticated methodology and a plug-in tool to measure productivity during the whole development process [19]. Based on the results, in the future hopefully we will be able to provide a more detailed overview on efficiency of different development aspects.

The overall model and functionality of the systems are out of the scope of this paper. We selected the simplest physiological data: blood pressure, blood-glucose, and bodyweight to represent the key differences between the models in the following sections.

A. Problem Specific Data Model

In our mid-term project, we were focusing only on the e-Health domain and its characteristics as much as the budget and time constraints allowed. The data model for storing and managing the physiological data described above was simply defined by a subtype/supertype hierarchy using inheritance. The parent entity is the *Data*, which contains the common attributes assigned for all types of data. Each derived entity represents a data type and its peculiar attributes. In Figure 3, the UML class diagram for this data structure can be seen.

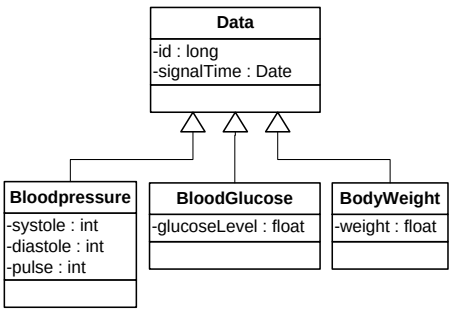

Figure 3. Problem Specific Data Model

The simple and clear data model served as a perfect basis to build up a fully broad system where all the functionalities were exactly fitted to the user requirements, and the doctors received a set of the accustomed views of the collected data, which were familiar to them from other medical systems. Furthermore, the predefined views were feasible to give a comprehensive insight into the patients' current health status along with ease-of-use user interfaces.

B. Generic Data Model

In the long-term project, the goal was to build a generic Telemonitoring System where the type of devices and the measured data the system deals with were not preliminarily

 ISBN: 978-1-61208-158-8

defined and limited to a number of sensors. The ability to simply or even in runtime add sensors to the system had the main impact on the design of the data model. Four entities were identified to store data. The *Datatype* entity defines the structured hierarchy of data types and for each entity a set of *DatatypeDescriptor* entities are appointed the attributes that belongs to the type. The *GenericData* entity can be imagined as an instance of the *Datatype* where the *GenericDataAttribute* instances hold the current values for attributes (Figure 4). The *Datatype* and *DatatypeDescriptor* describe the schema for the data type, e.g., the blood pressure data type has three attributes and all of them are integer values. The *GenericData* instances can be interpreted with the help of its *Datatype,* and the attributes can be processed as the *DatatypeDescriptors* circumscribe them.

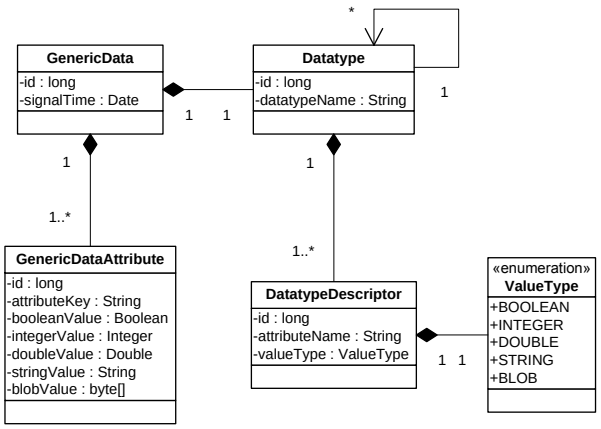

Figure 4. Generic Data Model

The model fulfilled all the predefined requirements. In addition, the generic data management was complemented with the ability to configure the measurements, signal processors and their dependencies in runtime by the end-users. A general data charting and representation could be built upon the data model where the doctors were free to define diagrams combining several measured and derived data with selecting a patient, a data type, and an attribute of the data type. The *Datatype* and *DatatypeDescriptor* entities were enough to define and render the diagrams containing the appropriate *GenericData* instances and the values of their attributes in a generic way. A similar data model is described in [7].

V. EVALUATION

The examination of each data model was based on the Quality Factors described above. The evaluation of *correctness* and *completeness* of the data models are out of the scope of this paper, but the Living Lab based experimentation pointed out that both of the models were actually correct and fulfilled the functional requirements.

Regarding *integrity*, the problem specific data model performs better as in the generic model some of the business rules are missing. Precisely, in the generic model the constraints on the values of the attributes are not enforced, which can cause inconsistency in the database and incorrect data could be stored. Such constraints are not allowed to specify on the model since different attributes of different data types have been stored in the same database column if the types of the attributes have been the same. These rules must be enforced in the higher layers of the system, e.g., in the data access layer or by triggers or interceptors. However, this is not a critical issue as the constraint validation can be supported to be able to automatically evaluate the rules by extending the *DatatypeDescriptor* entity with some attributes according to the constraints for the given data attribute. To stay with the models presented above, if we want to store blood pressure, blood-glucose, and bodyweight data 5 constraints are missing currently. This number is growing in line with the number of the data types defined to store. In the problem specific data model all these constraints can be exactly defined because each attribute is stored in a separate column. Regarding this integrity issue, the key difference can be observed between the architecture of each persistence layer based on these data models. Data validation can be found at a different level as it can be seen in Figure 5 and in Figure 6.

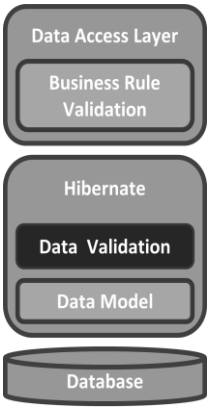

Figure 5. Specific Model based Persistence Layer

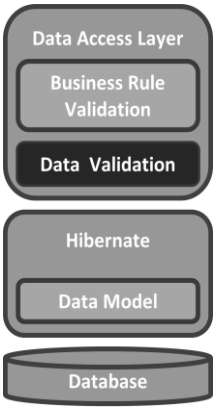

Figure 6. Generic Model based Persistence Layer

The *simplicity* of the data model can be defined by the number of entities and relationships presented in the model. The inheritance does not improve this number as in a physical database it is often mapped to a single table where all instances of the derived classes are stored and the derived entities represent only subcategories within a single construct. The complexity of the problem specific data model is 1, while the complexity of the generic data model is 8 (4 entities and 4 relationships). As it could be expected, the problem specific data model is simpler than the generic one, but this rate is invariant and does not depend on the number of data types defined to be stored.

The number of elements in the model, which are subject to change in the future and the cost estimation to implement the changes are defined as the metrics to evaluating *flexibility*. Although the probability of change is small in the structure of medical data, alteration in the auxiliary attributes could occurred, e.g., the current models do not deal with measure units. Also, it is possible that new data types appear later on to extend the system. The generic data model can handle these changes without any modification on the model, but in the problem specific data model all the 4 entities could be expected to be modified in the future.

 ISBN: 978-1-61208-158-8

The *reusability* of the data models in a Telemonitoring System or even partially in any kind of system in the e-Health domain is unambiguous and perfectly fits into this scope. However, the generic data model is not specialized only for e-Health development purposes, the overall data model is reusable in a different domain, e.g., in the field of agriculture to store data about plant status in a greenhouse. It means 4 entities are reusable from the generic data model in contrast with the problem specific data model where this number is 0 when we leave the scope of e-Health systems.

The *integration* of these models with other systems were out of scope in our R&D projects, but it is apparent that the generic model is more flexible and reusable as the previous metrics pointed out above, so it could be easier to provide interfaces and adapters towards other systems than implementing them over the problem specific data model. Furthermore, if the integration process requires some modifications in the data model, it could be seamlessly done in the generic data model.

For the *understandability* of a data model, not a voluble, quantitative metric exists. It is mostly the subjective opinion and rating of the users and the developers that provide information about this quality factor. Regarding our experience, the effort required to understand the generic data model was twice as much than the effort needed for the problem specific model. The ratio of entities and attributes also shows this deviation, where the metric is $^4/_{14}$ for the generic data model and $^4/_7$ for the problem specific model. The key difference is that the generic model is unable to interpret its context without concrete, domain specific examples.

According to the *implementability* the problem specific model is easier to implement, the only technical risk is the ability to provide an adequate solution to handle inheritance. The generic data model has more entities and relationships and the developers also have to deal with the integrity issues along with the data model implementation. The risk and effort are higher in the case of generic models, but not considerably.

Additionally, the *productivity* can be mentioned as a partial subcategory for implementability where the maintenance of the model is also engaged. In the design and implementation phases the problem specific data model was laboured and developed in $^1/_3$ less time than the generic model (the overall data model). However, the costs of maintenance are much higher in the case of the specific model. If changes affect the data model, all system layers have to be modified. Our experience was that it took $^1/_{10}$ less time to implement modifications in the system that was based on the generic model.

During the Living Lab experimentations both systems were applied in production usage. The clinical trial allows us to monitor the *performance* of the system and the model within real conditions. Although both systems were successfully used in the field of e-Health, the performance of the generic data model based system was continuously but not significantly falling back as the number of stored data increased. In Figure 7 the performance of each model can be seen. The diagram shows the required time in millisecond to provide a patient's blood-pressure data for one month along with the size of the database (number of stored data). The performance was monitored during the regular usage of the charting modules in each system.

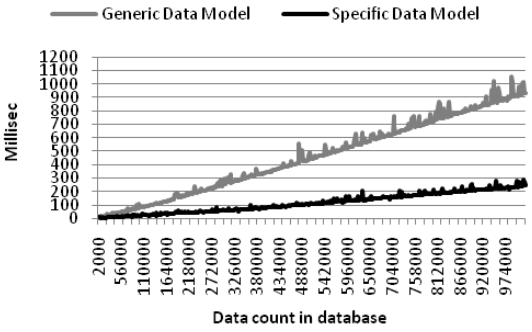

Figure 7. Performance Measurement on Data Models

To summarize the result of the evaluation, an overview of the Quality Factors is presented in the form of a Polar chart in Figure 8. The values are based on the subjective ratings of data modeling experts and the quantitative metrics described above. The chart helps with the conceptual representation of the overall quality and gives a comprehensive view to appraise the models.

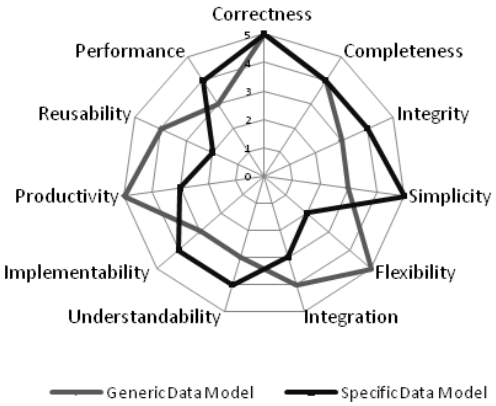

Figure 8. Overview of Data Model Quality Factors

Each model has its advantage in some of the factors however the overall rating is nearly the same. Moreover, there are unambiguous correlations between the Quality Factors [8], e.g., the simpler model is more understandable or the more flexible model is more reusable, which contributes to a fairly distinct set of factors that characterize the benefits of each model.

VI. USER ACCEPTANCE AND SATISFACTION

The benefits of the generic model definitely support the software developers. Although it takes more efforts in the design phase, in the overall development process and after the release the modifications could be easier to adapt. In addition, the flexibility and reusability factors are more relevant for software development. However, in practice the generic software is not the best solution to provide a product, which is

expected to fully cover all user requirements. In our experience, the generic user interfaces where the doctors could build up the charts and views on their own were not as successful as it could be expected. Ironically, it turned out that regarding the predefined views the doctors were more satisfied and they preferred to use that kind of user interface. The generic views require more competence from the users, which they cannot accept easily.

VII. CONCLUSION

In this paper, we overviewed the importance of the data model in a Telemonitoring System and we summarized the results of the evaluation of two different data models designed and developed in the field of e-Health. We revealed and compared the benefits and drawbacks of each model. While the generic data model is more feasible for software developers and can be adopted in long-term development procedures, the problem specific data model allows a rapid development with valuable results in short- or mid-term projects. Currently, it depends on the context and the conditions of the project which data modeling aspect would be better as a balance can be identified between the two sides (Figure 9).

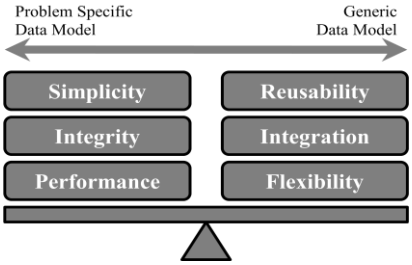

Figure 9. Balance of Data Modeling Aspects

Setting up adequate weightings along the quality factors while analyzing and identifying the key requirements of the system and the available resources could be helpful to decide which data modeling aspect will be the most convenient. However, based on our experience a specialized user interface based on a generic data model could be an acceptable intermediate solution in any case.

ACKNOWLEDGMENT

The work presented was partly funded by the National Innovation Office, Hungary (project No. OM-00191/2008-AALAMSRK 'ProSeniis'), Telenor 19/55 1I066, Nokia Komarom 19/55 1C117 and GOP-1.1.2-07/1-2008-0007.

REFERENCES

[1] M. Suh, L. S. Evangelista, C. Chen, K. Han, J. Kang, M. K. Tu, V. Chen, A. Nahapetian, and M. Sarrafzadeh, "An automated vital sign monitoring system for congestive heart failure patients," Proceedings of the ACM international conference on Health informatics - IHI '10, Arlington, Virginia, USA: 2010, pp. 108.

[2] M. Sarriegui, G. Sáez, H. Pérez, M. Elena, R. Cros, E. Brugués, A. Leiva, G. Aguilera, and J. Enrique, "Mobile Telemedicine for Diabetes Care," Mobile Telemedicine A Computing and Networking Perspective, CRC PRES, Taylor & Francis Group; ISBN: 9781420060461, 2008.

[3] J. G. Cleland, A. A. Louis, A. S. Rigby, U. Janssens, and A. H. Balk, "Noninvasive Home Telemonitoring for Patients With Heart Failure at High Risk of Recurrent Admission and Death: The Trans-European Network-Home-Care Management System (TEN-HMS) study," Journal of the American College of Cardiology, vol. 45, 2005, pp. 1654–1664.

[4] S. Scalvini, M. Vitacca, L. Paletta, A. Giordano, and B. Balbi, "Telemedicine: a new frontier for effective healthcare services," Monaldi Arch Chest Dis, vol. 61, 2004, pp. 226–233.

[5] Graeme C. Simsion and Graham C. Witt, Data modeling essentials, Morgan Kaufmann; ISBN: 978-0126445510, 2004.

[6] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: A survey," Computer Networks, 2010.

[7] J. Cai, S. Johnson, and G. Hripcsak, "Generic Data Modeling for Home Telemonitoring of Chronically Ill Patients," AMIA, Inc., 2000.

[8] D. L. Moody, G. G. Shanks, and P. Darke, "Improving the Quality of Entity Relationship Models," Lecture Notes in Computer Science, vol. 1507/1998, Springer-Verlag Berlin Heidelberg, 1998, pp. 255−276.

[9] M. West and J. Fowler, "Developing High Quality Data Models," EPISTLE, 1996.

[10] M. Fowler, "Analysis Patterns: Reusable Object Models," Addison-Wesley Professional, ISBN: 978-0201895421, 1996.

[11] M. Piattini, M. Genero, and C. Calero, "Data model metrics," Handbook of Software Engineering and Knowledge Engineering, vol. 2, 2002, pp. 981–02.

[12] D. L. Moody, "Metrics for Evaluating the Quality of Entity Relationship Models," Lecture Notes in Computer Science, vol. 1507/1998, Springer-Verlag Berlin Heidelberg, 1998, pp. 211−225.

[13] D. L. Moody, "Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice," Proceedings of the Eleventh European Conference on Information Systems, ECIS, 2003.

[14] http://www.proseniis.com; 26.04.2011, "ProSeniis Home Page."

[15] I. Vassányi, G. Kozmann, B. Végső, I. Kósa, T. Dulai, D. Muhi, and Z. Tarjányi, "Alpha: multi-parameter remote monitoring system for the elderly," MIE'2010, Cape Town, South Africa, 2010.

[16] http://www.medistance.hu; 26.04.2011, "Medistance Home Page."

[17] http://www.hibernate.org; 26.04.2011, "Hibernate."

[18] http://trac.edgewall.org; 26.04.2011, "The Trac Project."

[19] G. Tóth, A. Z. Végh, A. Beszédes, and T. Gyimóthy, "Adding Process Metrics to Enhance Modification Complexity Prediction," International Conference on Program Comprehension, 2011

 ISBN: 978-1-61208-158-8