Improving Context-aware Applications for the Well-being Domain
Model-driven Design Guided by Medical Knowledge

Steven Bosems and Marten van Sinderen
Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands

Keywords: Well-being, Context-aware Applications, Sensors, Model Driven Design, Medical Knowledge, Causal Reasoning.

Abstract: Computing applications for among others well-being and health become increasingly advanced as a result of their sensor-based awareness of the context in which they are used. Context-aware applications have the potential of providing enriched services to their users, i.e. services that are appropriate for the context at hand. A challenge for the design of context-aware applications is to identify and develop service enrichments which are effective and useful while not being overly complex and costly. It is hard to imagine, both for the designer and end-user, all possible relevant contexts and best possible corresponding enriched services. An enriched service which is not appropriate for the context at hand can irritate or even harm the user, and (eventually) leads to avoiding the use of the service. This paper discusses a model-driven approach that incorporates domain knowledge concerning the causal relationship between context factors and human conditions. We believe that such an approach facilitates the identification and development of appropriate sensor-based context-aware services. We focus on context-aware applications for the well-being domain.

1 INTRODUCTION

Sensors have become ubiquitous in our modern society, being literally almost always around us to play a role in many day-to-day computing applications. Sensors collect data from the physical world. They provide computing applications with the ability to observe and learn about the environment and situation of interest to the user that interacts with these applications.

Important application domains include, smart cities, smart environments, security, logistics, industrial control, home automation, and ehealth (Avci et al., 2010; Nakajima and Shiga, 2011). In this paper, we focus on well-being.

Well-being is the state of being comfortable, healthy, or happy. Computing applications that support well-being obviously can benefit from sensor data that provide information on the user's environment or situation. For example, the user may be informed or advised, such that he can take action that may improve his well-being, or the user's environment (such as light and temperature) may be controlled to this effect.

The development of context-aware well-being applications, being new and innovative products, requires a different approach than the design of traditional applications: as they are new to the market, users have no prior experience with comparable products, making it hard to formulate the desired functionality. Furthermore, applications in this domain are typically mobile; this causes the context of the application and user to change continuously, increasing the difficulty to envision fitting services for every possible environment and situation. Due to these challenges, the use of an iterative development process is more suited. In this paper, we shall elaborate on a process geared at the development of context-aware well-being applications.

Context-aware applications use an internal model of the context. This context model is based on the data that is captured by sensors. Applications apply causal reasoning to determine which enriched service (e.g., status report, advice, intervention, or control action) best suits the current context as represented in the context model. For the well-being domain, we are therefore interested in consolidated medical knowledge on the causal relationship between context factors and human health/well-being conditions. Such knowledge captured in a domain model would provide a good starting point for the design process. Firstly, the model would allow discussing the appli-

Bosems S. and van Sinderen M..
Improving Context-aware Applications for the Well-being Domain - Model-driven Design Guided by Medical Knowledge.
DOI: 10.5220/0004877503970403
In *Proceedings of the 3rd International Conference on Sensor Networks* (SENSORNETS-2014), pages 397-403
ISBN: 978-989-758-001-7
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

cation functionality (which conditions should be influenced, by which enrichments, based on which context factors/sensors) before (a prototype of) the application is built. Secondly, the model provides boundary conditions for the design, particularly regarding the internal context model and the causal reasoning model of the application.

The objective of this paper is to explore one such model, especially with respect to its role in a model-driven design process of context-aware applications for well-being. The design process should exploit the model, and so facilitate the identification and development of more effective and useful sensor-based context-aware services. Furthermore, the design process has potential of supporting semi-automated derivation of the application model and code.

The structure of this paper is as follows: section 2 discusses the need for domain modeling, section 3 describes how this domain model is to be used throughout the development process of well-being applications, section 4 discusses our method, section 5 gives an overview of related work, and section 6 provides concluding remarks.

2 DOMAIN MODELING

The development of context-aware applications which are effective and useful while not being overly complex and costly is very challenging. Approaches that focus on collecting correct and complete user requirements as a way to achieve effectiveness and usefulness, only address one side of the problem. Moreover, they have to deal with the drawback of innovative products: users are unable to formulate requirements beforehand, i.e. before they have seen and used the product. Approaches in this area that have so far been proposed therefore primarily focus on complementary requirements capturing methods, employing "playtesting" through executable use cases and mobile discovery (Jorgensen and Bossen, 2003; Seyff et al., 2008).

Our approach uses dynamic domain models that incorporate medical knowledge on causal relations between context factors and well-being conditions. Context- aware applications use related internal models to allow context reasoning. We believe that such models to discuss effectiveness and usefulness, and can be exploited in model-driven development to address complexity and costs.

2.1 Model Contents

In order to model the domain, we first have to identify the relevant knowledge on well-being. The concept of well-being is abstract, and tells us in a subjective way if a person is at ease and comfortable.

Not all factors that affect well-being are at the same level of abstraction. We categorize them as being either *physical* or *conceptual*. The former entails those elements in the world that are tangible or visible, the latter includes those that are abstract ideas or concepts. For example: a person's blood pressure or heart rate is considered a physical factor, whereas stress is a conceptual factor.

In our domain model, we want to capture conditions and factors from both these categories, and express how they influence each other. By doing so, we can analyze causal relationships between elements of the domain. Furthermore, if we want to influence a specific element, we can reason how this should be done.

2.2 Modeling Language

By interviewing experts on the domain of well-being and well-working, we can construct a domain model that covers the entire field of well-being. For representing domain models, we use an extended version of the causal loop diagram (CLD) language (Sterman, 2000). Variables in CLD models represent context factors and well-being conditions, arrows between variables express positive or negative causality between them: a positive causal link between the variables "Stress" and "Blood pressure" indicates that an increase in stress will cause the blood pressure to increase as well. The inverse holds for negative causal links. Our extension consists of annotating the model variables with an [o] (Observable) if we can directly observe the variable using sensors, with a [d] (Derivable) if the variable can be derived, and with a [c] (Controllable) if the variable can be directly controlled using actuators.

This dynamic domain model allows us to perform causal reasoning about context factors and well-being conditions in the domain: if we would like to improve a condition, we can argue how we are to reach this increase and how we should measure it. The overall domain model can be used to derive reasoning algorithms for context-aware well-being applications. Furthermore, by including norm values for vital signs, we can improve our reasoning capabilities: these norms are taken from medical literature such as (Hall, 2010), defining upper and lower bounds for vital signs. For example, the average resting heart rate

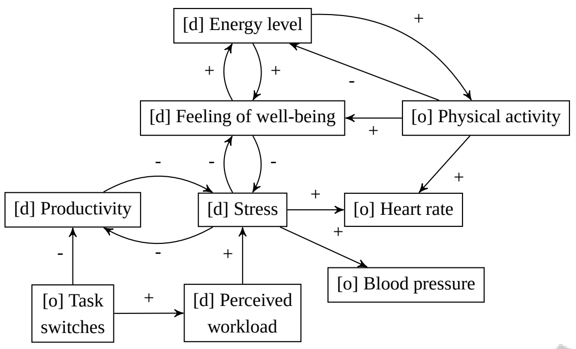

Figure 1: Domain model excerpt.

of an adult is between 60 and 100 beats per minute. By including this information, we can reason what should be done if for a user this value is out of these bounds. An expert of our domain model, excluding the norm values, is given in Figure 1.

3 DESIGN PROCESS

The domain model is the starting point for our development method for context-aware well-being applications. The process consists of four steps, which are discussed in the following sections. In these steps, we move from a generic well-being domain model to a prototype application which can be tested by users and stakeholders. This prototype is then to evolve into a marketable application.

3.1 Domain Model Reduction

The initial input for our development process is a model that describes the entire domain of well-being, covering all aspects of both mental and physical well-being. The scope of this model is too broad to start the development of a specific application. We must therefore narrow down the scope, capturing the right amount of information in an application specific domain model.

In order to create a model that is specific to the application we are developing, we must first decide what part of the well-being domain is to be supported by the application. For example, an application can focus on reducing stress, or increasing physical condition. We shall call this our application goal.

Once we have decided on a direction for the application, we have to identify which factors and are related to this focus. To do so, we first analyze which variables in the model are influenced by our goal. Looking at an example that focuses on stress management as a goal, we see (see Figure 1) that stress causes, among others, a high blood pressure, reduced productivity, and an increased heart rate. As such, these are of importance to our application. Once these conditions have been identified, we can look at other related variables: a reduction in productivity, for example, both causes and is caused by stress. However, it is also caused by the number of times a user switches from one task to another during work. As such, we can reason that the number of task switches made indirectly causes stress.

During this process step, we need information from a number of stakeholders. For example, prospective clients and commissioning companies may help to determine the application goal. Additionally, we need feedback from domain experts regarding the factors relevant to our application.

After this step, we have obtained a model that contains an application specific subset of the knowledge of the complete domain model.

3.2 Application Structure Specification

The next step of our development process entails the construction of a model that captures the static structure for our application. (Bosems et al., 2013) describe a high-level architecture for context-aware systems to support well-being. The authors found that context-aware systems and applications follow a similar pattern as traditional control systems. Sensors are used in order to obtain information about the context of the application. Using the sensor data, the application populates an internal context model. This model also contains derived context information, i.e. information about the context that cannot be directly measured. Based on the context model, applications decide on actions that are to be taken. These actions are then performed by actuators; these can either be physical (valves, lights, electro motors), or lexical (user interfaces). Both the derivation of context information and the decision process regarding actions involve reasoning governed by a a reasoning model. By continuously measuring the context, applications can potentially determine whether an action had the desired effect, and if this is not the case, change the reasoning model to compensate for the mismatch.

This step also has to consider whether the application is stand-alone or uses an existing and shared infrastructure (e.g., that captures raw sensor data and provides access to meaningful context information).

The application specific domain model is used as the input for this process step. We need the distinction between physical and conceptual factors in order to decide whether factors can be directly measured by sensors or should be derived, or how we should in-

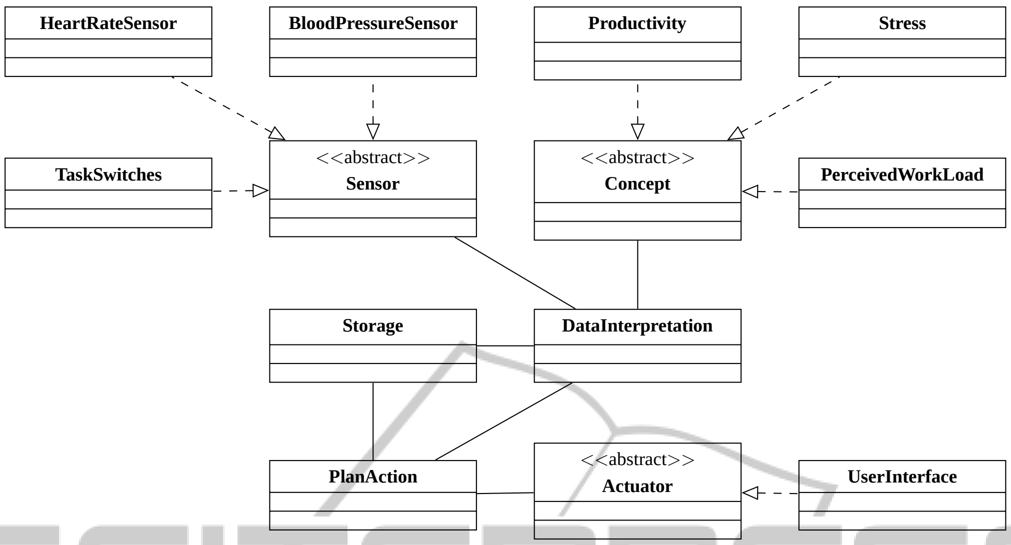

Figure 2: Example static application structure.

fluence them. If we are dealing with physical factors such as heart rate and blood pressure, they usually can be measured using sensors. Corresponding sensors are the first components to be added to the application structure.

Secondly, we look conceptual conditions. As these are not directly measurable, their values are decided through interpretation and deduction of the values that can be obtained through sensors. Components that model the conceptual well-being conditions are to be added to the model. Additionally, we introduce a component that interprets the data collected by the sensor components and derives the meanings for the conceptual components.

An interpretation component performs the reasoning in the application regarding the abstract and the physical domain elements. The result of this interpretation process is then fed to a component that performs the planning of actions: if the reasoning process concludes that a value for a domain element is not within the normal range, the component that plans the required actions is to deduce how to influence this value. In this decision process, the application specific model can be used to perform the needed causal reasoning.

Also part of the static application structure is the data model that will be used by the application to store run-time values and historic data. The latter can be used for situation analysis: we might have norm values from literature, but these norms might not fully align with the normal values for the specific user. In order to make decisions, we also need this information to reason. Historic data can also be used to determine whether the feedback approach for the user has been effective.

The development of the static application structure is performed by a system architect who can model the application based on the high level architecture, incorporating the information from the application model. An example application structure based on the dynamic domain model depicted in Figure 1 can be found in Figure 2.

The output of this process step are two models: (i) a model that captures the static application structure, and (ii) as well as a data model. As a modeling language, we use UML class diagrams, as these have been proven suitable to model such information in practice.

3.3 Application Behavior Specification

With the knowledge of norm values, the way these values are measured (this is captured in the static application structure), and means of reasoning about how the physical and conceptual conditions can be affected in order to increase or decrease measured or deduced values of others, we can start describing how the application is to interact with the world outside the application, how different components interact with each other, and what processes are to be performed by the components.

The sensors and actuators are the application's way of observing and interacting with context. When sensors obtain data from the world around the application, they only provide raw data. This data has to be interpreted before the application can use it. This is the task for the components modeling the sensors. Using an interpretation strategy that is specific for the

Table 1: Process summary.

Step	Input	Output	Stakeholders
Domain Model Reduction	Domain model	Application specific model	End-users, domain experts, prospective buyers
Application Structure Specification	Application model, High level architecture	Static application structure model	System architect
Application Behavior Specification	Application model, static structure	Behavior model	Domain experts, application designer
Prototype Development	Application model, Static structure, behavior model	Prototype application	Software engineer, end-user

type of sensor that is modeled, we can translate from raw data to units that are common in our domain (for example beats per minute for heart rate, and mm Hg for blood pressure). This way, the obtained data can be compared to the norm values. The behavior of the actuator components is described in a similar way, translating from the units used in the description of required actions to the activation of the actuators.

The behavior of the component that reason about the context can be either push or pull oriented: the information from the sensor components can be fed to the reasoning component as it arrives, or the component can request this information from the sensors when it is needed. Pushing information might result in an abundance of data being transferred, whereas only pulling information could result the usage of data that is out of date. The best strategy for this process is dependent on the application and is considered outside the scope of this paper. The communication between the reasoning component and the component that plans the appropriate action is to be described as push communication: if action is deemed needed, the planning of this action has to start as soon as possible. If no action is needed, no information is transferred.

When reasoning about conceptual factors, we have to combine sensor information to be able to say something about their state. We can analyze which physical conditions are to be measured in order to deduce conceptual factors using the application specific model. How these measurements should be combined will depend on which concept we are dealing with, e.g. the current level of productivity of a person can be deduced from current sensor measurements, but the level of stress will also require the evaluation of historic data.

To deduce what feedback should be given, we have to analyze our domain model. Since this model captures how factors influence each other, we can find possible causes for a variable value to be out of range. As a result, we can also analyze how we should influence the user in order for the value to return to normal. This analysis is to be performed per condition.

In this step, domain experts and application designers have to work together to model the dynamic behavior of the different application components.

The result of this process step is a behavior model per application component that illustrates what the feedback strategy should be depending on measured values.

3.4 Prototype Development

With the dynamic and the static parts of the application described, we can start developing a prototype. This process is similar to that of developing an ordinary application, with the added difficulty of having to deal with and reason about context information and well-being related factors. However, as we have methodologically been dealing with these specifics, the implementation of a prototype will consist primarily of providing the user with an interface to the application, and by deciding which method of feedback is suitable for the type of application and the intended user: even if the way of collecting and reasoning about context information is perfect, and the right conclusions are drawn, the application will be pointless if the appropriate service can not be provided in the right way. Which type of feedback strategy to choose, however, is outside the scope of this paper.

A summary of this process is shown in Table 1.

4 DISCUSSION

Because the primary outputs per development step are models, we see the possibility of incorporating model-driven techniques, as described by (Kent, 2002). Using model-driven techniques, we may be able to partially automate stops in our design process. These techniques employ model transformation rules that specify how input models of a design step can be translated in output models which serve as input for the next step. Transformation rules are based on the identification of regularities in design activities, models and languages (where automation can save time and prevent human error). However, since design is a creative process, manual input from the designer/developer almost always remains necessary in any design step.

Looking at step 1 of our process, we see the first need for human intervention: deciding and selecting which parts of the well-being domain are to be covered by the application under development can not be done automatically. We can support the designers by aiding the selection process, making sure the set of selected elements is as complete as possible, but the final decision in this step has to be made by humans.

In order to complete step 2, we have to translate between two modeling languages, requiring a mapping between the two. Because of the structuring of context-aware applications, we an predict which variables from the application specific model are to be mapped to which application structure model element: observable variables are modeled as sensors, deducible variables are part of the data interpretation of the application, and the controllable variables are mapped onto actuators. As we can see, most of this step can be performed automatically.

Because we are dealing with the behavior of the application in step 3, we will require human intervention again. It will be possible to deduce part of the reasoning to be done by the application, this can be deduced from the causal links in our application specific model. However, how these actions are to be planned and in what way they are to be shown to the user requires human creativity.

The final step in our process, the creation of the prototype, can only be supported by the automatic generation of code based on the models created in steps 2 and 3. User interface design and implementation can not be performed automatically. However, the development can be aided by the domain- and application specific models: they allow engineers to reason about behavior, making it easier to implement the needed actions the application is to present.

In this paper we have not discussed aspects such as the conceptualization of sensor data, the way conditions can be quantified or how the capturing and storage of large data volumes should be handled by the application. We consider these aspects to be specific for the application under development and as such not generalizable for the development process of all context-aware well-being applications.

5 RELATED WORK

With smartphones containing an increasing number of sensors, the development of "apps" that advise the user on medical subjects has rapidly increased. However, as any developer, regardless of their medical background and expertise, can create such a smartphone application. The users of this app expect it to work properly and provide the right information, but this is not always the case. As a response to this problem, the European Union and the United States of America have developed certification and programs for medical apps (European Commission, 2013; U.S. Department of Health and Human Services Food and Drug Administration et al., 2013). Applications that do not use measurements or provide medical treatment, e.g. the issuing of medication, would not require certification. However, those that do utilize sensors to obtain the user's vitals are to be subjected for testing.

(Jorgensen and Bossen, 2003) discuss a method of performing requirements engineering for pervasive health care systems. Even more so than with well-being applications, the correct working of these systems is crucial, the treatment of patients depending on them. The authors propose a three-tier approach to creating and using executable use cases, going from prose, to executable models, and finally to an animated example the user can interact with. The main difference with our type of application, is that the pervasive systems looked at by the authors have a fixed context, a hospital, in which the users move around. This is unlike our application, in which the user is the only constant of the system context, with the environment continuously changing.

In (Maiden et al., 2004), the authors describe a model driven approach called RESCUE that integrates 4 modeling techniques in order to obtain a set of requirements that is as complete as possible. The four techniques that are integrated are activity modeling, system goal modeling (which is performed in the i^* language), use case modeling and requirements management. Using model transformations, the authors synchronize the data captured in each of these models in order to keep them consistent. When comparing this work to our own, we primarily identify the difference between the domains: our systems operate in highly dynamic, changing environments, while the system the authors of RESCUE looked at are predominantly static, allowing for more complete requirements elicitation. The use of different angels to look at the same problem in order to obtain a more complete picture is, however, an interesting approach which may be a direction for future work.

6 CONCLUSIONS

Sensor-based context-awareness has the potential to improve the usefulness of applications, allowing developers to create applications that are able to observe the world around them, adapting their services accordingly. However, design and development of context-aware applications is a task that is troublesome at best. Users have a hard time imagining contexts in which they are not currently situated. Design approaches that rely on traditional requirements capturing are therefore not feasible. Requirements capturing methods that involve iterative prototyping and playtesting may be too time consuming and may overlook potential. We propose a development method that uses a model of the domain as a starting point.

This domain model captures the medical knowledge concerning the causal relationships between context factors and human well-being conditions. It also captures the medical boundary values of well-being conditions. The scope of this overall domain model is then to be reduced to fit the application's goal. Using this application specific model, we can first determine the static structure of the application, after which the dynamic behavior can be described. The final step in the process entails the development of a prototype of the application and providing it to the user to test it in practice.

Our method is better suited for context-aware well-being applications than traditional development methods, as it focuses primarily on the structure of the domain and the causal relations between context factors and well-being conditions. Furthermore, our domain model makes it easier to reason about the desired behavior of the application, preventing the development of applications that do not satisfy the user's ideas and demands.

ACKNOWLEDGEMENTS

This publication was supported by the Dutch national program COMMIT (project P7 SWELL).

REFERENCES

Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., and Havinga, P. (2010). Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: a survey. In *23rd Int. Conf. on Architecture of Computing Systems (ARCS 2010)*, pages 1–10.

Bosems, S., van Sinderen, M., Achterkamp, R., van der Meulen, F., van Dantzig, S., Goorden, M., and Mulder, S. (2013). COMMIT SWELL D1.2 Overall architecture. Technical report.

European Commission (2013). Guidelines on a medical devices vigilance system. Technical report.

Hall, J. E. (2010). *Guyton and Hall Textbook of Medical Physiology*. Saunders.

Jorgensen, J. and Bossen, C. (2003). Requirements engineering for a pervasive health care system. In *11th IEEE Int. Requirements Engineering Conf.*, pages 55–64. IEEE Comput. Soc.

Kent, S. (2002). Model-Driven Engineering. In *Integrated Formal Methods*, pages 286–298. Springer.

Maiden, N. A. M., Jones, S. V., Manning, S., Greenwood, J., and Renou, L. (2004). Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study. In *16th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2004)*, pages 368–383. Springer.

Nakajima, H. and Shiga, T. (2011). Smart devices and services in healthcare and wellness. In *Symposium on VLSI Circuits*.

Seyff, N., Graf, F., Grünbacher, P., and Maiden, N. (2008). Mobile Discovery of Requirements for Context-Aware Systems. In *14th Int. Working Conf., REFSQ 2008*, pages 183–197.

Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw-Hill/Irwin.

U.S. Department of Health and Human Services Food and Drug Administration, Center for Devices and Radiological Health, and Center for Biologics Evaluation and Research (2013). Mobile Medical Applications: Guidance for Industry and Food and Drug Administration Staff. Technical report.