Towards a Norm-Driven Design of Context-Aware e-Health Applications

Boris Shishkov[1], Hailiang Mei[2], Marten van Sinderen[2], Thijs Tonis[3]

[1]Delft University of Technology, Department of Systems Engineering
Delft, The Netherlands
b.b.shishkov@tudelft.nl

[2]University of Twente, Department of Computer Science
Enschede, The Netherlands
{meih, m.j.vansinderen}@ewi.utwente.nl

[3]Roessingh Research and Development
Enschede, The Netherlands
t.tonis@rrd.nl

Abstract. In this paper, we explore the usefulness of elaborating process models with norms, especially focusing on the Norm Analysis Method (NAM) as an elaboration tool that can be combined with a process modeling tool, such as Petri Net (PN). The PN-NAM combination has been particularly considered in the paper in relation to a challenge that concerns the design of context-aware applications, namely the challenge of specifying and elaborating complex behaviors that may include alternative (context-driven) processes (we assume that a user context space can be defined and that each context state within this space corresponds to an alternative application service behavior). Hence, the main contribution of our paper comprises an adaptability-driven methodological and modeling support to the design of context-aware applications; modeling guidelines are proposed, considered together with corresponding modeling tools (in particular PN and NAM), and partially illustrated by means of an e-Health-related example. Given the multi-disciplinary nature of the e-Health domain, it is expected that the current research will be useful for it. In particular, e-Health system developers might benefit from the relevant methodological and modeling support, proposed in the paper.

Keywords: Business modeling; Application modeling; Context-Aware applications; Norm Analysis; Petri Net; e-Health.

1 Introduction

Context-Aware (CA) applications are characterized by adaptability which is the capability of adequate derivation of user context states (this involves sensing the user environment and transforming the sensed raw data into context information) and

appropriate reaction to user context state changes [11]. Such a reaction should include 'switching' from one desirable behavior to another. Even though the application would be supposed to realize such a 'switching' at real time, it must be foreseen at design time. CA features are of relevance to the e-Health domain since they can enhance the dependability and user-friendliness of healthcare services [6].

The designer of CA applications should not only specify each desirable behavior but also determine the rule patterns governing the 'switchings' between behaviors. Thus, the issues to be addressed in this work are: (i) how to define each alternative behavior not only at high level of abstraction but possibly also at lower levels; (ii) how to relate these alternative behaviors in an overall behavior (including the 'switching' between alternative behaviors); (iii) how to analyze these behaviors and apply appropriate (rule-driven) 'switching' patterns (where correctness is determined by the fact that exhibited behavior matches the desirable behavior for a given context situation). We nevertheless claim that these challenges are interrelated since we consider the 'switching' rule patterns as naturally complementing the corresponding behavior flow patterns.

Hence, our particular focus is the flow-rule combination and we approach this issue, taking as a basis previous results on designing CA applications, driven by corresponding business modeling [9,11]. We take as a starting point in this paper a behavior model (that is to be derived based on corresponding static model) of a CA application and we consider the model's normative (rule) elaboration.

Realizing this, it is not only studied how to conduct conceptually and methodologically such an elaboration but this is also complemented by suggestions that concern a possible realization in terms of modeling tools/techniques. These suggestions are not trivial however. It might be that one modeling formalism is suitable for defining each of the alternative behaviors and for relating these alternative behaviors in an overall behavior, while another modeling formalism is suitable for analyzing these behaviors, and still another formalism is suitable for defining the 'switching', and so on. Maybe a language which is close to the architectural domain would be suitable for high level behavior specification, whereas for the aim of analysis, a language that allows automated evaluation of the properties of concern should be chosen. Hence, the design process would comprise transformations between design models and analysis models, in addition to transformations between levels of abstraction. In such complex modeling, it is essential to know which exactly are the challenges and which techniques are suitable for approaching them. We take in this work only the perspective of norm (rule) elaboration to high-level behavior models including such ones that reflect context-driven behavior (characterized by alternative processes).

With respect to this, and inspired by previous research results [8], we turn particularly to Petri Net (PN) and the Norm Analysis Method (NAM), as a possible combination of techniques that is relevant and suitable to the above-mentioned challenge. PN is a well-established and widely popular modeling technique that depicts the structure of a distributed system as a directed bipartite graph [18]. As such, PN has place nodes, transition nodes, and directed arcs connecting places with transitions, with whose support, the modeling technique can be useful to the modeling and analysis of (business) processes. Using PN, one could: (i) represent a process in a straightforward way; (ii) conduct process analysis; (iii) make adequate pre-simulation

models - helpful with regard to discrete-event simulation [16]. Addressing such challenges with PN, one usually takes a PN as a triple (*P*,*T*,*F*) that consists of two node types called 'places' and 'transitions', and a flow relation between them. Places are used to model milestones reached within a business process and transitions - as the individual tasks within the business process to execute. Having the possibility to model (using these modeling primitives) all widely used process patterns, we expect that PN would be adequate, suitable and useful with regard to the challenge of modeling alternative behaviors and relating them in an overall behavior. As for analyzing these behaviors, PN could also be useful (as it has been mentioned already); however, this goes beyond the scope of the current paper. Further, concerning the related 'switching' specification, we would consider NAM as a possible complement to PN [15]. NAM is a semiotic method to identify norms in an explicit and articulate manner [5]. Norms, which include formal and informal rules and regulations, define the dynamic conditions of the pattern of behavior existing in a community and govern how its members (agents) behave, think, make judgements and perceive the world. Their presence enables agents to exhibit normative patterns to be more or less 'predictable' [3]. Therefore, NAM is useful for studying an organisation from the perspective of agents' behavior which is governed by norms [17]. There are four types of norms existing. In business process modeling, most rules and regulations fall into the category of behavioral norms. These norms prescribe what agents must, may, and must not do, which are equivalent to three deontic operators: 'is-obliged', 'is-permitted' and 'is-prohibited'. With the introduction of the deontic operators, norms are broader than the normally recognised business rules and therefore provide more expressiveness. For those actions that are 'permitted', whether the agent will take an action or not is seldom deterministic. This elasticity characterises the business processes, and therefore is of particularly value to understand the organizations [3]. Moreover, a complete NAM can be performed in precise steps and results of the NAM can be written in a natural language and further in a programming language for automatic execution [5].

Thus, main contribution of the current paper comprises an adaptability-driven methodological and modeling support to the design of CA applications; modeling guidelines are proposed, considered together with corresponding modeling tools (in particular PN and NAM), and partially illustrated by means of an e-Health-related example. Given the multi-disciplinary nature of the e-Health domain, it is expected that the reported research will be useful for it. In particular, e-Health system developers might benefit from the relevant methodological and modeling support, proposed in this paper.

The paper's outline is as follows. Section 2 motivates further our proposed design views, by introducing CA applications and presenting a vision on their design, relating this to PN and NAM. Section 3 introduces briefly PN and NAM, discussing their strengths (and the adequacy of combining them), especially related to the challenge of realizing the proposed design vision. Section 4 presents a small example which is used in Section 5 to partially illustrate some of the proposed design steps. Section 6 contains the conclusions.

2 Context-Aware Applications

Taking into consideration the social/economical, methodological, and technical concerns related to the design, implementation, deployment, and operation of CA applications (as already introduced in previous (related) work [9]), and focusing particularly on part of the methodological concerns (especially those concerns that are relevant to the design of CA applications), we find it fundamental that CA applications acquire knowledge/information on the situation of the user (referred to as 'user context information') and exploit this knowledge to provide the best possible service [11]. Hence, this concerns the user context (this is, for example, the location of the user, the user's activity, the user's access to particular devices, and so on) and the assumption that the user is in different context over time, and as a consequence – (s)he has changing preferences or needs with respect to services. A schematic set-up for a CA application is depicted in Figure 1.

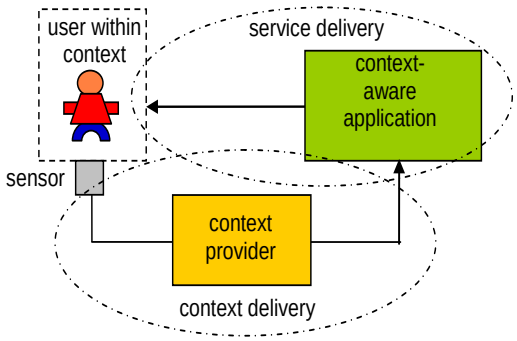

Fig. 1. Schematic representation of a context-aware application.

As seen from the figure, in considering the design of a CA application, one should not only address the service delivery, from the application to the user, but also the delivery of context information that is needed by the application for adaptation of its service delivery. Sensors sample the user's environment and produce (primitive) context information, which is an approximation of the actual context, suitable for computer interpretation and processing. Higher level context information may be derived through a process of inference and aggregation (using input from multiple sensors) before it is presented to the application, which in turn can decide on the current context of the user and the corresponding service that must be delivered.

Furthermore, we follow an application design lifecycle inspired by previous work [9] and extending the design process of an existing approach [13]. Hence, our design lifecycle comprises a number of phases, the most important among which are:

- **Business modeling**: during this phase, the end-user is considered in relation to processes that either support him/her directly or the goal(s) of related business(es). These processes have to be identified, modeled and analyzed with respect to their ability to (collectively) achieve the stated goals. A model of these processes and their relationships is called a *business model*.
- **Application modeling**: during this phase, the attention is shifted from the business to the IT domain. The purpose is to derive a model of the application, which can be used as a blueprint for the software implementation based on a target technological platform. A model of the application, whether as an integrated whole or as a composition of application components, is called an *application model*. Business models and application models should certainly be aligned, in order to achieve that the application properly contributes to the realization of the business/user goals. As a starting point for achieving proper alignment, one could delineate in the final business model which (parts of) processes are subject to automation (i.e., are considered for replacement by software applications). The most abstract representation of the delineated behavior would be a service specification of the application (as an integrated whole), which can be considered as the initial application model.
- **Requirements elicitation**: both the business model and the application model have to meet certain requirements, which are captured and made explicit during the phase called *requirements elicitation*. Application requirements can be seen as a refinement of part of the business requirements, as a consequence of the proposition that the initial application model can be derived considering (parts of) the business processes (within the final business model), especially those processes selected for automation.
- **Context elicitation**: an important part of the design of a context-aware application is the process of finding out the relevant end-user context from the application point of view; we will refer to this phase as *context elicitation*. End-user context is relevant to the application if a context change would also change the preferences or needs of the end-user, regarding the service of the application. Context elicitation can therefore be seen also as the process of determining an end-user context state space, where each context state corresponds to an alternative desirable service behavior. Since relevant end-user context potentially has many attributes (location, activity, availability, and so on), a context state can relate to a complex end-user situation, composed of (statements on) several context attributes. Moreover, context elicitation relates to requirements elicitation in the sense that each context state is associated with requirements (i.e., preferences and needs of the end-user) on desirable user behavior. Context elicitation can best be done in the final phase of business modeling and the initial phase of application modeling, when the role and responsibility of the end-user and the role and responsibility of the application in their respective environments are considered.

Fig. 2 depicts these different phases and activities.

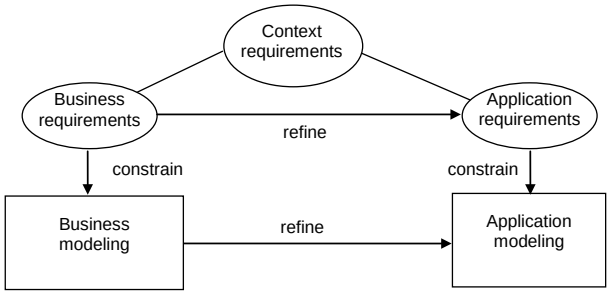

Fig. 2. Application design life cycle [9].

Following [11], we assume that a user context space can be defined and that each context state within this space corresponds to an alternative application service behavior. In other words, the application service consists of several sub-behaviors or variations of some basic behavior, each corresponding to a different context state. Any service behavior model would have to express the context state dependent transitions from one sub-behavior (or behavior variation) to another one.

Focusing especially on the context-driven application behavior adaptation, and inspired by the related background information considered above, we propose design guidelines according to which: (i) it is assumed that there are several main states in which the user might be (we take into account that for example, hundreds of states might be possible, only several of which are of high probability to occur, however) and that each of these states requires a particular application behavior; (ii) design preparations are necessary in order to specify each of these behaviors as well as the way in which their related context states are to be 'sensed' by the application; (iii) each change in the state of the user is to trigger a change in the application behavior. This is depicted on Figure 3:

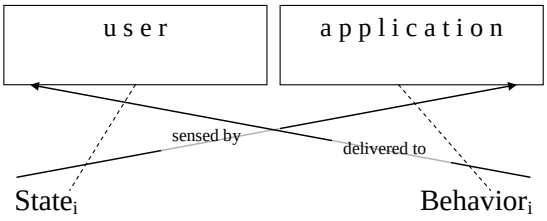

Fig. 3. Towards a context-driven application behaviour adaptation.

As it is seen from the figure, we assume that the context state relates to the user of the application (as indicated by a dashed line), taking nevertheless into account that other (non-user) entities and their situations could also (indirectly) affect the desirable application behavior; we as well assume that, in delivering a service to its user, the application performs a particular behavior (indicated by a dashed line). Moreover, it is

essential that the user context states are sensed by the application and that the (user-context-driven) application behavior is delivered to the user.

Inspired by previous relevant work concerning PN and NAM [13,15,16], we claim that by applying these tools in combination, it is possible to realize the design vision introduced above. We thus especially focus on the combined use of NAM and PN, where PN is used to specify and analyze the dynamic aspects of the services, and NAM to specify and analyze the constraints that govern the enabling of transitions. By introducing and briefly discussing these tools in the following section, and by offering exemplification of their combined use directed towards the design of CA applications, we will further justify the above-stated claim.

3 Design and Analysis Tools

Considering the CA application design vision presented in Section 2, we briefly introduce PN and NAM, especially discussing their related strengths and the usefulness of combining them. We show further how by combining these tools, one could acquire benefits especially relevant to the design of CA applications.

3.1 Petri Net

To allow adequate dynamic business/application modeling that corresponds to the above-mentioned design vision, we claim important that some modeling-related aspects are incorporated, especially aspects that corresponds to the following requirements, as formulated by Van Hee and Reijers [18]:

- possibility to adequately grasp business process structures (sets of tasks that have to be completed in some kind of order);
- possibility to apply choice, parallel, cycle, and other constructions typical for most current business processes;
- possibility to apply qualitative/quantitative simulation to the (dynamic) models, as an appropriate way of achieving validation.

Being a well-known and widely used formalism and a graphical language for the design, specification, and verification of systems, PN has been used in a number of cases demonstrating strengths related not only to process structures and typical constructions (such as choice and parallelism) but also to the derivation of simulation models [16]. We hence claim that PN is a suitable modeling tool for realizing some of the demands concerning the design vision presented in Section 2.

PN has been introduced already in Section 1, as a triple (*P*,*T*,*F*) that consists of two node types (*places* and *transitions*), and a flow relation between them. Places are to model milestones reached within a business process and transitions should correspond to the individual tasks to execute; places are represented by circles, transitions are represented by rectangles. The process constructions which are applied to build a business process, are called *blocks*. The blocks that express some typical constructs (sequence, choice, parallelism, iteration) are depicted as PN in Figure 4.

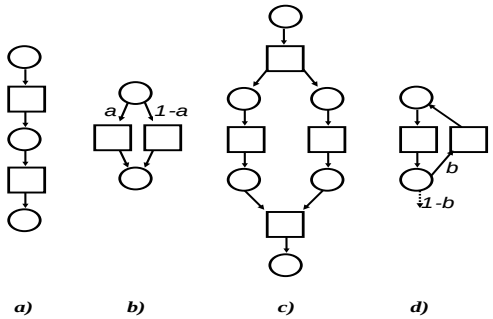

Fig. 4. Typical process constructs expressed with PN.

Sequence block (*a)*) considers tasks that are in sequential order (the first task has to be completed before the second task can be started). The Choice block (*b)*) represents a construction in which exactly one of two alternatives is carried out. The Parallelism block (*c)*) shows how two tasks can be modeled such that they can be executed simultaneously. Finally, the Iteration block (*d)*) represents a construction in which the execution of a task can be repeated.

Arc labels occur in the Choice and Iteration blocks, representing the values of a Bernoulli-distributed random variable that is associated with these blocks; an independent draw from such a random variable determines the route of the flow [16,18]. Each new application of such a block is accompanied by the introduction of a new, independent random variable.

PN is hence not only a proper formalism with regard to the demands formulated in the previous section but it is also capable of representing most typical process constructs, having at the same time sound theoretical background. For this reason, we consider realizing via PN the dynamic modeling activities concerning the design vision introduced in Section 2.

Moreover, the strengths of PN concerning the modeling of decision points and parallel processes is especially relevant to the challenge of modeling alternative behaviors. Using the same notations for modeling this all gives the precious possibility to grasp the big picture and go consistently in details, and also to map to other notations and simulate. These are all strengths that concern in particular the design of CA applications.

A further challenge nevertheless that concerns not only PN but also other process modeling formalisms is the insufficient elaboration facilities with regard to ‘decision’ points (whether it should be decided which arc to follow, for example). Inspired by our earlier experience, we claim that combining PN and NAM could be a good solution [15]. We hence introduce NAM in the following sub-section, putting stress not only on its relevance to the problems outlined in Section 2 but also on its suitability as a combination with PN is concerned.

3.2 Norm Analysis Method

Norms, which include formal and informal rules and regulations, define the dynamic conditions of the pattern of behavior existing in a community and govern how its members (agents) behave, think, make judgments and perceive the world [3]. As Von Wright explains: "'norm' has several partial synonyms which are good English, 'pattern', 'standard', 'type' are such words. So are 'regulation', 'rule', 'law'" [19].

Norms are developed through practical experiences of agents in a community, and in turn have functions of directing, coordinating and controlling their actions within the community. When modeling agents and their actions, which may reveal the repertoire of available behaviors of agents, norms will supply rationale for actions. Norms will also provide guidance for members to determine whether certain patterns of behavior are legal or acceptable within a given context. An individual member in the community, having learned the norms, will be able to use the knowledge to guide his or her actions, though he or she may decide to take either a norm-conforming or a norm-breaking action. When the norms of an organization are learned, it will be possible for one to expect and predict behavior and to collaborate with others in performing coordinated actions. Once the norms are understood, captured and represented in, for example, the form of deontic logic, it will serve as a basis for programming intelligent agents to perform many regular activities [5,13].

The long established classification of norms is probably that drawn from social psychology, partitioning them into perceptual, evaluative, cognitive and behavioral norms; each governing human behavior from different aspects. However, in business process modeling, most rules and regulations fall into the category of behavioral norms. These norms prescribe what people must, may, and must not do, which are equivalent to three deontic operators "of obligation", "of permission", and "of prohibition". Hence, the following format is considered suitable for specification of behavioral norms.

whenever <condition>
if <state>
then <agent>
is <deontic operator>
to <action>

The condition describes a matching situation where the norm is to be applied, and sometimes further specified with a state-clause (this clause is optional). The actor-clause specifies the responsible actor for the action. The actor can be a staff member, or a customer, or a computer system if the right of decision-making is delegated to it. As for the next clause, it quantifies a deontic state and usually expresses in one of the three operators - permitted, forbidden and obliged. For the next clause, it defines the consequence of the norm. The consequence possibly leads to an action or to the generation of information for others to act [5].

With the introduction of deontic operators, norms are broader than the normally recognised business rules; therefore provide more expressiveness. For those actions that are "permitted", whether the agent will take an action or not is seldom deterministic. This elasticity characterises the business processes, and therefore is of particularly value to understand the organisations [3].

NAM, as a semiotics method, can be carried out to identify norms in an explicit and articulate manner and a complete norm analysis can be performed in four steps, which are responsibility analysis, proto-norm analysis, trigger analysis and detailed norm specification [5].

The responsibility analysis enables one to identify and assign responsible agents to each action. The analysis focuses on the types of agents and types of actions. In other words, it would answer the question as to which agent is responsible for what type of actions.

The proto-norm analysis helps one to identify relevant types of information for making decisions concerning a certain type of behavior. After the relevant types of information are identified, they can be used as a checklist by the responsible agent to take necessary factors into account when a decision is to be made. The objective of this analysis is to facilitate the human decisions without overlooking any necessary factors or types of information.

The trigger analysis is to consider the actions to be taken in relation to the absolute and relative time. The absolute time means the calendar time, while the relative time makes use of references to other events. The results of trigger analysis are specifications of the schedule of the actions. All the actions can be organised in dynamic sequences. By setting up and managing triggers for the actions, the automated system can prompt human agents to respond to the situation in time.

In the detailed norm specification step, the contents of norms will be fully specified in both a natural language and a formal language. The purposes for this are (1) to capture the norms as references for human decision, and (2) to perform actions in the automated system by executing the norms in the formal language. For example, adopting the format given above for specification of behavioral norms, a credit card company may state norms governing interest charges as follows [4].

whenever an amount of outstanding credit
if more than 25 days after posting
then the card holder
is obliged
to pay the interest.

As long as the norms are specified, computing technologies such as active databases, object technology and artificial intelligence will have different approaches towards software realisation.

Therefore, the combination of PN and NAM is of essential value for the design of CA applications, in which design we do not only need to properly model the alternative behaviors but also to exhaustively elaborate on the corresponding ‘decision’ points.

4 Example

We provide an e-Health application example in this section in order to explain the modeling support in Section 5; the example has been considered in [1,2].

John is an epileptic patient who has suffered seizure for several years. He enrolled in a remote monitoring system which monitors his health state and can give him a few

seconds' advance warning of an upcoming seizure. He wears a Body Area Network (BAN) consisting of a set of body-worn sensors and a Smartphone. The BAN can collect bio-signals of the patient or other data like location or activity and sends the data to the healthcare centre via GPRS or UMTS network. An automatic seizure detection program running at the Smartphone can keep monitoring and analyzing John's bio-signals.

If an upcoming seizure is detected by the local detection program, an alarm on John's Smartphone will be triggered and a warning message will be displayed with a "cancel" button on the screen (Figure 5). Once receives the alarm, John should stop his current activities and sit down. After a short period, if nothing has happened to John, then it was a "false alarm". In this case, John should cancel this alarm by pressing the "cancel" button and he can resume his activities.

Fig. 5. The screenshots of a seizure attack warning message and the application for healthcare centre to locate a nearby care-giver [2].

However, in case John does not cancel the alarm within the given period, it could be because that John is already attacked by a seizure and lost his control. This timeout will trigger an alarm at the healthcare centre and doctor should examine the received bio-signals of John. If the doctor sees it is indeed a seizure attack based on the received signal, he should locate a care-giver who is nearby John and send a notification to this care-giver with John's location. The notified care-giver should then go to John's location and provide John the help.

5 Combining PN and NAM

In this section, we illustrate by means of the example (see Section 4) the combined application of PN and NAM, sticking to the design vision formulated in Section 2.

We firstly build the PN model (Figure 6), and for brevity we omit all early analysis steps that start from the analysis of the case briefing and end up with the PN model – information on that can be found in [11].

On Fig. 6, there are 13 places (including the 'start' and 'end') and 12 transitions. There is a choice construct (concerning transitions labelled with 2, 3, and 1) which indicates that essentially there are two user context states – the situation of the patient is either 'normal' or suggesting 'risk' (this cases triggering some system actions).

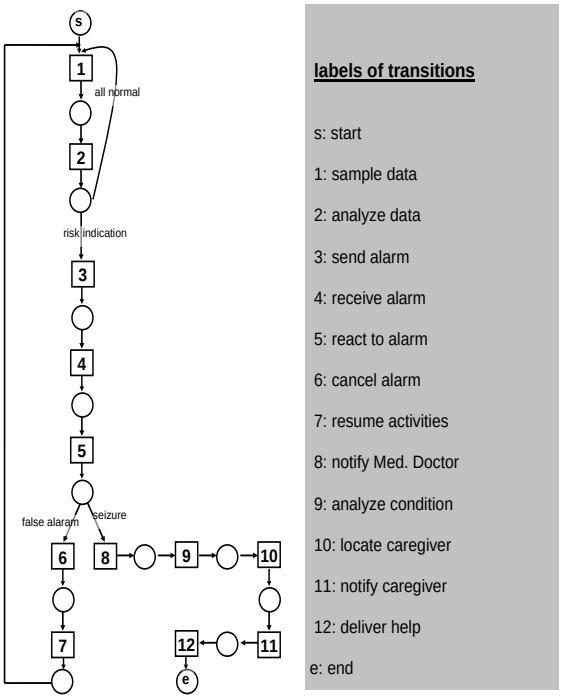

Fig. 6. – The Health-care process expressed with PN.

There is another choice construct (concerning transitions labelled with 5, 6, and 8) which is about the 'choice' between 'false alarm' (when no seizure appears) and the case of a seizure appearing.

As depicted on the figure, vital signs are 'captured' (by sensors) and sampled (by the device) - Transition 1. The data is analyzed by the device (Transition 2), which continues on and on if all seems normal. In case of risk indication nevertheless, alarm is triggered (Transition 3) and after it is perceived by the user (Transition 4), the user has to react, by stopping his or her current activities (Transition 5). If this actually appears to be a 'false alarm' (as it was discussed in the previous section), the user should firstly cancel the alarm (Transition 6) and secondly – resume the interrupted

activities (Transition 7), in which the user is back to the monitoring. If seizure appears (that is 'indirectly' indicated by not receiving of alarm cancellation), then the system notifies the Medical Doctor (Transition 8) who should in turn analyze the condition of the patient on the basis of the information received (Transition 9), locate a caregiver (in the area of the user) and notify the caregiver (Transition 10 and Transition 11, respectively). The caregiver then would go for delivering help (Transition 12), which is marking the end of the process considered.

Furthermore, NAM can provide useful elaboration to the PN model depicted in Figure 6. According to the four steps of NAM introduced in Section 3, two norms corresponding to the second choice construct in the figure has been identified and specified in detail, to only illustrate partially the usefulness of a norm elaboration to a PN model.

Concerning Transition 5, both norms below are associated.

whenever no seizure has occurred
then the user
is obliged
to send 'cancel alarm' signal.

whenever a seizure indication has occurred
then the system
is obliged
to notify the Medical Doctor (MD).

Going down the 'hierarchy' of norms, concerning Transition 8, the norm below is associated:

whenever seizure notification has been sent to MD
then the MD (Medical Doctor)
is obliged
to analyze the condition of the user.

This is only a partial illustration, with many norm-elaboration steps omitted for brevity. These steps mainly concern the overall norm structuring – the hierarchy in which all norms relate to each other. Such a hierarchy can be seen as a tree with a 'constitutional norm' on top [13], the general norm that 'governs' the whole behavior; all norms should be consistent with this norm, in the same way in which the overall behavior must be consistent with the general requirement towards the (desirable) functionality. Those norms which are immediately 'below' the constitutional norm not only obey it but also govern in turn other norms which are below them. All these levels of refinement should be approached both in the process model and in the hierarchy of norms.

The process model and the norms give thus two interrelated and complementing perspectives on the modelled behavior – for example:

- the PN model depicts the process patterns;
- the norms define the dynamic conditions concerning these patterns; these dynamic conditions are essentially driven by (changes in) the user context states.

This is how, combining a process modeling tool with norms, by possibly applying together PN and NAM, could usefully support the design of CA applications.

6 Conclusions

This paper proposes improvements with respect to the *business-process modeling* concerning the design of *context-aware applications*. A model-driven considered that has been enriched with relevant guidelines and demands. Moreover, their fulfilment has been approached through a combination of modeling tools, namely PN (Petri Net) and NAM (Norm Analysis Method), whose selection was motivated. Hence, combining PN and NAM, we have shown how a complex behavior (concerned with context-driven alternative processes) could be adequately specified, analyzed, and norm-elaborated. A NAM-driven elaboration especially applied to complex process constructs (such as choice, parallelism, and iteration) helps not only to define with precision these process chunks but also to relate the behavior choices to corresponding context information (a norm defines not only a rule but also the context in which this rule appears to operate). This all however does not concern the 'switching' between desirable behaviors at real time; the focus is put instead to useful design preparations in cases of desired adaptability of the application to possible context changes. A further step in the design would be a consideration of particular technology platforms, such as Web services, CORBA or J2EE, which is left nevertheless beyond the scope of this work.

However hence claim that this paper makes useful contributions concerning (i) the possibility to consider user context in support of the (application's) design; (ii) the proposed extension to a (modeling) approach, expressed as guidelines+demands; (iii) the achievement of an appropriate fulfilment of the (mentioned) guidelines+demands, by the combined application of Petri Net and Norm Analysis. To justify our claim, we have studied related work. On the basis of the study, we have identified several approaches/methods which usefully address the modeling of (business) processes, notably SDBC [12,13], UML Activity Diagram [7], ISDL [12].

SDBC supports the PN-NAM combination, making however no connection to user context states and corresponding alternative (desirable) behaviors. The UML Activity Diagram does not support rule-driven elaboration. ISDL's notations appear to be too complex to allow a straightforward normative elaboration.

To further this research, we plan to work on bridging the current behavior-modeling-related results to previous results on identifying entities and relationships, especially in the context of SOA [10,14], so that we capture the challenge of context-aware applications design from both static and dynamic perspectives. We also intend to project this to NAM-related aspects, tracing them back to semantic aspects, in tune with the theory of Organizational Semiotics [5], which would allow us to achieve a more sound and coherent overall application architecture.

Acknowledgements

This work has been supported by the Systems Engineering Department of Delft University of Technology and the Freeband A-MUSE project (http://a-muse.freeband.nl) sponsored by the Dutch government under contract BSIK 03025.

References

1. A-MUSE Project, 2008: http://a-muse.freeband.nl.
2. AWARENESS Project, 2007, http://awareness.freeband.nl.
3. Liu, K., 2005. Requirements Reengineering from Legacy Information Systems Using Semiotic Techniques, Systems, Signs and Actions. Int. Journal on Communication Information Technology & Work, 1(1): 36-61.
4. Liu, K., Sun, L., Dix, A., Narasipuram, M., 2001. Norm Based Agency for Designing Collaborative Information Systems, Information Systems Journal, 11: 229-247.
5. Liu, K., 2000. Semiotics in information systems engineering, Cambridge University Press. Cambridge.
6. Mei, H., Widya, I.A., Broens, T.H.F., Pawar, P., Van Halteren, A.T., Shishkov, B., Van Sinderen, M.J., 2007. A Framework for Smart Distribution of Bio-signal Processing Units in m-Health. In ICSOFT'07, 2nd International Conference on Software and Data Technologies. INSTICC Press.
7. Rational / OMG UML, 2007. Unified Modeling Language, Object Management Group, http://www.omg.org/uml.
8. Shishkov, B., Van Sinderen, M.J., Liu, K., Du, H., 2008. Norm Analysis Supporting the Design of Context-Aware Applications. In ICEIS'08, 10th International Conference on Enterprise Information Systems. INSTICC Press.
9. Shishkov, B. and Van Sinderen, M.J., 2008. On the Design of Context-Aware Applications. In I-WEST'08, 2nd International Workshop on Enterprise Systems and Technology. INSTICC Press.
10. Shishkov,.B., Van Sinderen, M.J., Tekinerdogan, B., 2007. Model-Driven Specification of Software Services. In: IEEE International Conference on e-Business Engineering, ICEBE 2007, 24-26 Oct 2007, Hong kong, China. pp. 13-21. IEEE Computer Society Press.
11. Shishkov, B. and Van Sinderen, M.J., 2007. Model-Driven Design of Context-Aware Applications. In ICEIS'07, 9th International Conference on Enterprise Information Systems. INSTICC Press.
12. Shishkov, B. and Quartel, D., 2006. Refinement of SDBC Business Process Models Using ISDL. In ICEIS'06, 8th International Conference on Enterprise Information Systems. INSTICC Press.
13. Shishkov, B., Dietz, J.L.G., Liu, K., 2006. Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In ICEIS'06, 8th International Conference on Enterprise Information Systems. INSTICC Press.
14. Shishkov, B., Van Sinderen, M.J., Quartel, D., 2006. SOA-Driven Business-Software Alignment. In ICEBE'06, IEEE International Conference on e-Business Engineering. IEEE Press.
15. Shishkov, B. and Dietz, J.L.G., 2004. Deriving Use Cases from Business Processes, The Advantages of DEMO In: Enterprise Information Systems V Publisher: Kluwer Academic Publishers Edited by: Camp, O.; Filipe, J.B.; Hammoudi, S.; Piattini, M.G.
16. Shishkov, B. and Barjis, J., 2002. Modeling of e-Business Brokerage Systems Using UML and Petri Net In: Information Systems: The e-Business Challenge Publisher: Kluwer Academic Publishers Edited by: Traunmuller, R.
17. Stamper, R. K., 1992. Language and Computer in Organised Behaviour. In R. P. v. d.Riet (Ed.), Linguistic instruments in knowledge engineering. North-Holland: Elsevier Science.
18. Van Hee, K. and Reijers, H.A., 2000. Using Formal Analysis Techniques in Business Process Re-Design. W. van der Aalst et al. (Eds.): Business Proc. Management, LNCS 1806.
19. Von Wright, G. H., 1963. Norms and Action - a Logical Enquiry. Routledge and Kegan Paul, New York.