On the Design of Context-Aware Applications

Boris Shishkov and Marten van Sinderen

University of Twente, Department of Computer Science, Enschede, The Netherlands
{b.b.shishkov, m.j.vansinderen}@ewi.utwente.nl

Abstract. Ignoring the dynamic context of users may lead to suboptimal applications. Hence, context-aware applications have emerged, that are aware of the end-user context situation (for example, "user is at home", "user is travelling"), and provide the desirable services corresponding to the situation at hand. Developing context-aware applications is not a trivial task nevertheless and the following related challenges have been identified: (i) Properly deciding what physical context to 'sense' and what high-level context information to pass to an application, and bridging the gap between raw context data and high-level context information; (ii) Deciding which end-user context situations to consider and which to ignore; (iii) Modeling context-aware application behavior including 'switching' between alternative application behaviors. In this paper, we have furthered related work on context-aware application design, by explicitly discussing each of the mentioned interrelated challenges and proposing corresponding solution directions, supported by small-scale illustrative examples. It is expected that this contribution would usefully support the current efforts to improve context-aware application development.

Keywords. Application development; Context-Awareness; Behavior modeling.

1 Introduction

Traditional application development methods do not consider the context of individual users of the application under design, assuming instead that end-users would have common requirements independent of their context. This may be a valid assumption for applications running on and accessed at desktop computers, but would be less appropriate for applications whose services are delivered via mobile devices [1, 9]. Ignoring the dynamic context of users may lead to suboptimal applications, at least for a subset of the context situations the end-user may find him/herself in. Therefore, especially driven by the successful uptake of mobile telephony and wireless communication, a new strand of applications has emerged, referred to as *context-aware* applications [12]. Such applications are, to a greater or lesser extent, aware of the end-user context situation (for example, "user is at home", "user is travelling") and provide the desirable services corresponding to the situation at hand. This quality points also to another related characteristic, namely that context-aware applications must be able to capture or be informed about information on the context of end-users, preferably without effort and conscious acts from the user part.

Shishkov B. and van Sinderen M.
On the Design of Context-Aware Applications.
DOI: 10.5220/0004464600210034
In *Proceedings of the 2nd International Workshop on Enterprise Systems and Technology (I-WEST 2008)*, pages 21-34
ISBN: 978-989-8111-50-0

Developing context-aware applications is not a trivial task nevertheless and the following related challenges have been identified: (i) Properly deciding what physical context to 'sense' and what high-level context information to pass to an application, and bridging the gap between raw context data and high-level context information; (ii) Deciding which end-user context situations to consider and which to ignore; (iii) Modeling context-aware application behavior including 'switching' between alternative behaviors.

Inspired by the mentioned challenges, we have furthered a related work on context-aware application design [12], by explicitly discussing each of these interrelated challenges and proposing corresponding solution directions, supported by small-scale illustrative examples. It is expected that this contribution would usefully support the current efforts to improve context-aware application development.

The outline of the remaining of this paper is as follows: Section 2 further motivates the actuality of context-awareness as a desirable application quality. Section 3 provides relevant background information to be used as a basis for proposing improvements. Section 4, Section 5, and Section 6 address in more detail the challenges mentioned above, respectively. Finally, Section 7 presents our conclusions.

2 Motivation for Context-Awareness

The basic assumption underlying the development of context-aware applications is that end-user needs are not static, however partially dependent on the particular situation the end-user finds him/herself in. For example, depending on his/her current location, time, activity, social environment, environmental properties, or physiological properties, the end-user may have different interests, preferences, or needs with respect to the services that can be provided by applications.

Context-aware applications are therefore primarily motivated by their potential to increase user-perceived *effectiveness*, i.e. to provide services that better suit the needs of the end-user, by taking account of the user situation. We refer to the collection of parameters that determine the situation of an end-user, and which are relevant for the application in pursue of user-perceived effectiveness, as end-user context, or *context* for short, in accordance to definitions found in literature [4].

Context-awareness implies that information on the end-user context must be captured, and preferably so without conscious or active involvement of the end-user. Although in principle the end-user could also provide context information by directly interacting with the application, one can assume that in practice this would be too cumbersome if not impossible; it would require deep expertise to know the relevant context parameters and how these are correctly defined, and furthermore be very time consuming and error-prone to provide the parameter specifications as manual input.

Context-aware applications can be particularly effective if the end-user is *mobile* and uses a personal handheld device for the delivery of services. The mobile case is characterized by dynamic context situations often dominated by changing location (however not necessarily restricted to this). Different locations may imply different social environments and different network access options, which offer opportunities for the provision of adaptive or value-added services based on context sensitivity. Especially in the mobile case, context changes are continuous, and a context-aware

application may exploit this by providing near real-time context-based adaptation during a service delivery session with its end-user. The adaptation is 'near real-time' because context information is an approximation (not exact representation) of the real-life context and thus there may be a time delay.

Through context-awareness, applications can be pro-active with respect to service delivery, in addition to being just re-active, by detecting certain context situations that require or invite the delivery of useful services which are then initiated by the application instead of by a user request. Otherwise said, traditional applications provide service in reaction to user requests (re-active), whereas context-aware applications have also the possibility of initiating a service when a particular context situation is detected, without user input (pro-active).

Although context-aware applications have received much attention within the research community, they have not been fully successful so far from a business point of view. This situation may change rapidly however, due to the observed growing power and reduced prices of mobile devices, sensors, and wireless networks, and due to the introduction of new marketing strategies and service delivery models [6,5].

In summary, context-awareness concerns the possibility of delivering effective *personalized* services to the end-user, taking into account his/her particular situation or context. Technological advances enable better and richer context-awareness, beyond mere location-sensitivity. Hence, service delivery models, specifically those targeting the mobile market, would allow companies to offer the added value in more attractive ways to the end-user.

Concerning the development and introduction of context-aware applications, as it has been mentioned already, this is not a trivial task. Efficiency and productivity would greatly benefit from an architecturally well-founded context infrastructure and design framework [17, 16, 3].

3 Architectural Implications and Design Considerations

In this section, we consider essential architectural/design issues concerning context-aware applications, and we also identify and briefly outline (on this basis) three important related interconnected challenges (to be elaborated in the following sections).

3.1 Architectural Implications

Context-aware applications acquire knowledge on context and exploit this knowledge to provide the best possible service. As already mentioned, the particular focus in this work concerns the end-user context, i.e. the situation of a person who is the potential user of services offered by an application. Examples of end-user context are the location of the user, the user's activity, the availability of the user, and the user's access to certain devices or facilities. The assumption we make is that the end-user is in different contexts over time, and as a consequence (s)he has changing preferences or needs with respect to services.

A schematic set-up for a context-aware application is depicted in Fig. 1. Here, the application is informed by sensors of the context (or of context changes), where the sensing is done as unobtrusively (and invisibly) for the end-user as possible. 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 inference and aggregation (using input from multiple sensors) before it is presented to applications which in turn can decide on the current context of the end-user and the corresponding service(s) that must be offered.

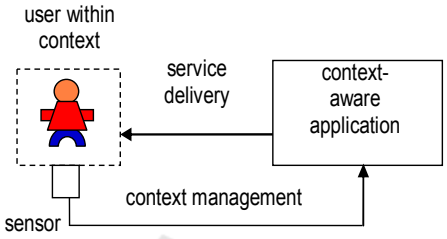

Fig. 1. Schematic representation of a context-aware application.

The design, implementation, deployment, and operation of context-aware applications have many interesting concerns, including:

- social/economical: how to determine useful context-aware services, where useful can be defined in terms of functional and monetary value?
- methodological: how to determine and model the context of the end-user that is relevant to the application; how to relate the context to the service of the application and how to model this service; how to design the application such that the service is correctly implemented?
- technical: how to represent context in the technical domain; how to manage context information such that it is useful to the application; how to use context information in the provisioning of context-aware services?

Addressing the last two concerns (especially the last one) starts with considering the possible architectures and in our view, two principle architectures could be proper:

- **Context-aware Selection**: end-user request(s) and end-user related context information are used to discover a matching service (or service composition). Discovery is supported by a repository of context-enhanced service descriptions. A context-enhanced service description not only specifies the functional properties (goals, interactions, input, output) and non-functional properties (performance, security, availability), but also the context properties of the service. Context properties indicate what context situations the service is targeting. For example, a service could provide information which is region-specific (such as a sightseeing tour), and therefore the context properties could indicate the relevance for a particular geographical area.
- **Context-aware Execution**: after the end-user request(s) has been processed and a matching service(s) has been found (possibly in the same way as described

above), the service delivery itself would adapt to changing context during the service session with the end-user. When the context of the end-user changes in a relevant (to the application) way, the service provided is adapted to the situation at hand. For example, the user may move from one location to another while using a service that offers information on objects of interest which are close-by (such as historic buildings within a radius of five kilometres, for example).

In both architectures, a new role is introduced, namely the role of *context provider*. A context provider is an information service provider where the information is context information. A context provider captures raw context data and/or processes context information with the purpose of producing richer context information which is of (commercial) interest. Interested parties could be other context providers or application providers. Further, a context-ware application obviously requires an *adaptive service provisioning component* and a *context information provisioning component*.

3.2 Design Considerations

Our design approach is a partial refinement of an existing approach [14] that concerns a general design life cycle, comprising, amongst others:

- **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.

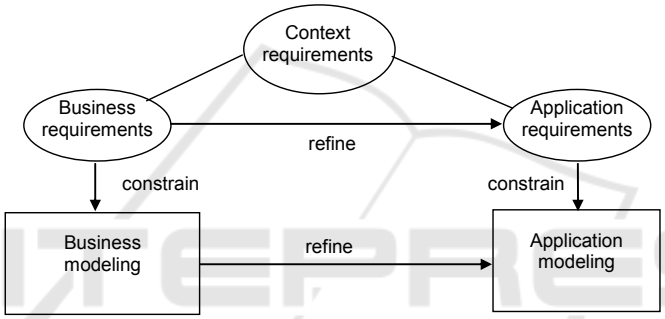

Fig. 2. Application design life cycle.

Following [12], we assume that an end-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.

3.3 Challenges

As mentioned already, developing context-aware applications is not a trivial task and the following related challenges have been identified:

- Properly deciding what to 'sense' and how to interpret it in adapting application behavior can be problematic since the interpreted sensed information must be a valid indication for a change in the situation of the end-user and it is not always trivial to know how context information is to correspond to a user situation.

- Deciding which end-user context situations to consider and which to ignore is challenging because there may be tens or even hundreds of possible end-user situations, with only several of them with high probability to occur, and therefore considering the others at design time is not sensible with respect to adequate resource expenditure.
- Modeling the application behavior including the 'switching' between alternative desirable application behaviors can be complicated because alternative behaviors are behaviors themselves which also are to be considered in an integrated way, allowing for modeling the 'switching' between them, driven possibly by rules.

In the following sections, we will address explicitly each of these challenges.

4 Deriving Context Information

An adequate decision about what should be 'sensed' and how it is to be interpreted, concerns the extraction of context information from raw data, which relates broadly to *context reasoning* [2].

Context reasoning is concerned with inferring context information from raw sensor data and deriving higher-level context information from lower-level context information. As for the extraction of context information from raw data, related algorithms are needed to support it, and two main concerns are to be taken into account:

- specific target applications, e.g. in domains such as healthcare or finance, requiring the *output* of the algorithms;
- the availability of sensors providing *input* to the algorithms.

Current standard mobile devices can already operate as sensors, e.g. they can gather GPS info, WiFi info, cellular network info, Bluetooth info, and voice call info. In addition, dedicated sensors (that for example measure vital signs) can be integrated with existing mobile networked devices. Next to that, future standard mobile devices may even include other types of sensors, e.g. measuring temperature.

Hence, it is considered crucial developing efficient context reasoning algorithms, by investigating whether it is possible to derive certain specific context information from certain specific sensor information. In order to adequately refine such algorithms, additional restrictions would need to be taken into account:

- restrictions concerning the (specific) processing environments of mobile devices;
- restrictions on memory usage, processing power, battery consumption, wireless network usage;
- restrictions that concern real-time versus delayed availability of extracted context.

In order to develop adequate algorithms that extract context from raw sensor data, it is thus important to appropriately consider gathering raw sensor data which is augmented with user input. Concerning the sensor data, it should be pre-processed and filtered, in order to be properly structured as input for the context reasoning

algorithms which in turn would be expected to automatically yield the desired output. The (delivered) context information must be of certain (minimal) quality in order to be useful; otherwise said, certain Quality-of-Context levels should be maintained.

Finally, some issues that have more indirect impact, need also to be taken into account: (i) The delivered context information would have to be often applied in real-time environments where failures, performance requirements, available interfaces, and operational environments are to be taken into careful consideration; (ii) In order new applications to be enabled, it is important to investigate how the algorithms could be integrated in the infrastructure for context awareness.

5 Occurrence of Context Situations

Reasoning concerning context should point to the different situations the end-user may appear to be (situations that are characterized by corresponding context information. Often it is worthwhile considering the *occurrence probabilities* of these situations since, as mentioned already, usually only several (out of more) end-user situations are of high probability to actually occur. We call such an investigation *situation analysis*.

As studied in [12], it is helpful to support such an analysis by means of 'pragmatic' decisions (for instance: to ignore end-user situations which usually do not occur, although they might occur with some (certainly small) probability). Such subjective decisions should however be rooted in more objective studies that justify the decision(s) taken. In our view, a possible way of approaching this is through *random variables*. Exploring their probabilities allows one to apply statistical analysis, including *hypotheses testing* and *parameters estimation* [7].

Considering just possible outcomes is sometimes not enough in approaching a phenomenon; one might need to refer to an outcome in general. This is possible through a random variable, if the occurrence probability of the outcomes is studied (a random variable is a function that associates a unique numerical value with every outcome of an experiment).

An experimental data bank could be built through observations. Then, by applying statistical analysis, the development team would get the right insight on: (i) which end-user situation to be defined as the 'default' situation (the situation that points to the 'default' application behavior); (ii) which of the other situations are to be put 'for consideration'; (iii) which (obviously the rest) should be ignored. This is illustrated in Fig. 3 (where **n** should be certainly equal to **m**+**p**+**1**):

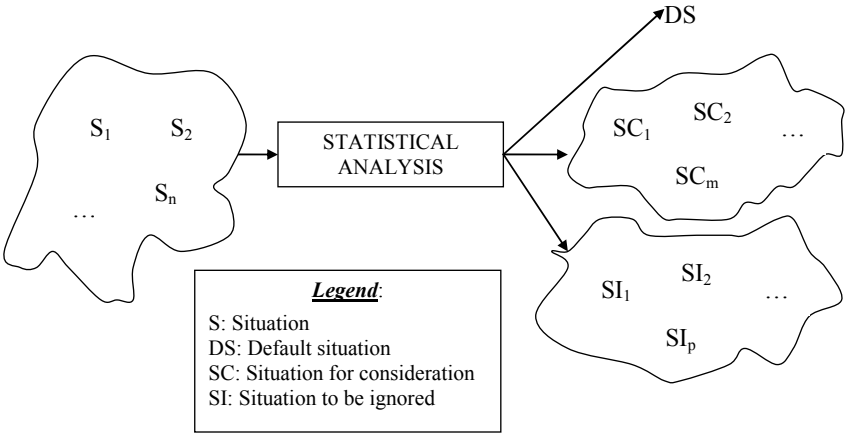

Fig. 3. Applying statistical analysis.

In a healthcare-related example, considered in [12], a hospital could be viewed as an end-user and there are exactly two possible end-user situations or states (considered as possible outcomes), namely: 'not too busy' (some medical doctors are immediately available to provide help) and 'very busy' (all medical doctors are occupied or have scheduled appointments within half an hour, for example). We consider the random variable **Y** with respect to these outcomes. **Y** would be a discrete random variable [7] since it may take on only a countable number of distinct values (in our case two). Provided the number of possible distinct values is exactly two, we have the case of *a priori probabilities* of each of the alternative outcomes (this means that one of these probabilities can be calculated by deducting the other one from **1**).

Only for the purpose of exemplifying how statistical analysis (applied to information that has been collected through observations) could be of use for the application designer, we take the probabilities from the mentioned example: the a priori probability of the first of the mentioned possible outcomes ("not too busy") is **0.9** and the a priori probability of the second alternative outcome ("very busy") is therefore **0.1**.

Knowing the occurrence probability of each outcome helps in deciding (in this particular example) which to be the 'default' desirable application behavior (the other one – that points to the other alternative outcome – would be the alternative behavior). It would be of course sensible considering the application behavior that corresponds to the first possible outcome as the 'default' behavior.

Once the designer has grouped the possible end-user situations, as suggested by Fig. 3 (only a 'default' and 'alternative' situations to be considered in the example), it is important making sure that the application is capable of 'sensing' the end-user situations. The proposed way of solving this is through observation of the values of appropriate *parameters*. If there are **n** parameters relevant to a scenario, then each of the parameters would have certain possible *values*. Then each value combination would point to a particular end-user situation.

In the example, we might distinguish two parameters ($\mathbf{p}_1$ and $\mathbf{p}_2$) and five corresponding values, as follows:

- $\mathbf{p}_1$ is about the ratio between the number of patients and the number of medical doctors at the particular moment, and is with just three possible values: $\mathbf{v}_{11}$ (the number is less than **1**), $\mathbf{v}_{12}$ (it is exactly **1**), and $\mathbf{v}_{13}$ (it is more than **1**)
- $\mathbf{p}_2$ concerns the particular moment – *normal* (the hospital is supposed to function as usual during working hours) or *extreme* (the hospital can rely on limited (human) resources, as during night-time, for example), and has just two possible values, respectively for normal and extreme, namely $\mathbf{v}_{21}$ and $\mathbf{v}_{22}$.

There are six possible value ($\mathbf{p}_1$,$\mathbf{p}_2$) combinations, namely $\mathbf{v}_{11}.\mathbf{v}_{21}$, $\mathbf{v}_{11}.\mathbf{v}_{22}$, $\mathbf{v}_{12}.\mathbf{v}_{21}$, $\mathbf{v}_{12}.\mathbf{v}_{22}$, $\mathbf{v}_{13}.\mathbf{v}_{21}$ and $\mathbf{v}_{13}.\mathbf{v}_{22}$. Driven by some additional domain analysis, omitted here for brevity, we determine the last combination only as validly corresponding to the ***0.1****-probability* alternative (the 'Second' alternative), and thus all the rest, corresponding to the ***0.9****-probability* alternative (the 'First' alternative), as depicted in Fig. 4.

	Parameters' values' combinations
First alternative	$v_{11}.v_{21}$, $v_{11}.v_{22}$, $v_{12}.v_{21}$, $v_{12}.v_{22}$, $v_{13}.v_{21}$
Second alternative	$v_{13}.v_{22}$

Fig. 4. Recognition of end-user situations.

Hence, knowing the values of the two parameters (the values can usually be captured using sensors), one could actually 'sense' the end-user situation at a particular moment [12].

6 Managing Alternative Application Behaviors

After a consideration of the different possible end-user situations that point to (corresponding) alternative application behaviors, the application designer has to adequately address the challenge of managing these behaviors; even though the 'switching' between behaviors would take place at real time, proper design time preparations are to be realized. These preparations should not only concern the modeling of each of the alternative behaviors to be considered but they should also address the 'switching' between behaviors (driven by a change in the end-user situation).

Taking into account that the 'switching' between alternative behaviors is insufficiently elaborated in current approaches [11,12] and inspired by previous experience, we propose the usage, in combination, of *Petri Net* [15] and *Norm Analysis* [8,13].

Petri Net could be considered 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*. They express some typical constructs, such as sequence, choice, parallelism, and iteration. Hence the strengths of Petri Net, concerning the modeling of decision points and parallel processes, are especially relevant to the challenge of modeling alternative behaviors. Using the same notations for conveniently modeling at different abstraction levels, gives the precious possibility to grasp the 'big picture' and go consistently in details, and also to map to other notations, and also to simulate. A further challenge nevertheless that concerns not only Petri Net but also other process modeling formalisms, is the insufficient elaboration facilities with regard to 'decision' and other complex points. We claim that combining Petri Net and Norm Analysis (to be introduced further in the current section) could be a good solution in this perspective [10].

Norm Analysis essentially concerns Semiotic Norms, or *norms* for short, 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. When the norms of an organization are learned, it would 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, this could serve as a basis for programming intelligent agents to perform many regular activities. 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 a behavioral norm:

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, who could 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. 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 would take an action or not is seldom deterministic. This elasticity characterises the business processes, and therefore is of particularly value to understand organisations.

The combination between Petri Net and Norm Analysis is of interest, especially with regard to the challenge of managing alternative application behaviors, for a number of reasons, among which are the following:

- Petri Net is a well-established process modeling formalism with sound theoretical roots and 'convenient' notations, that only misses facilities for exhaustive elaboration concerning complex points, while Norm Analysis is a well established rule modeling formalism possessing also sound theoretical roots and impressive (process-elaboration-related) expressiveness.
- There are examples of applying Petri Net and Norma Analysis in combination [10].
- The useful capability of modeling and elaborating (through Petri Net + Norm Analysis) complex process constructs makes the Petri Net – Norm Analysis combination attractive particularly with regard to the challenge of managing alternative application behaviors.

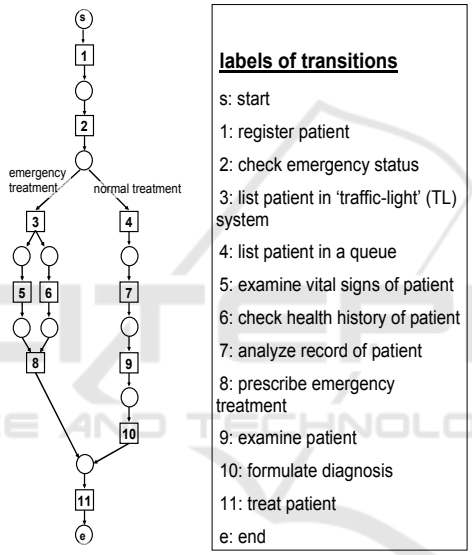

Fig. 5. A typical health-care process.

Fig. 5 (left) presents a typical health-care process, using Petri Net, and it is easily seen that there are two alternative behaviors, namely emergency and normal treatment. We could use Norm Analysis in such cases to usefully elaborate the process model. For instance, two norms corresponding to the choice construct in Fig. 5 (left) can be identified and specified in detail – consider Fig. 5 (right).

Therefore, by combining Petri Net and Norm Analysis, one could substantially facilitate the handling of (alternative) application behaviors.

7 Conclusions

This paper has presented further results that concern the development of context-aware applications. In particular, following a related motivation statement and based on architecture/design visions on the development of context-aware applications, we have identified and outlined three related interconnected challenges, proposing and motivating afterwards corresponding solution directions, summarized as follows:

- To decide what to 'sense' and how to interpret it in adapting application behavior, one would need to apply *context reasoning* for the purpose of properly extracting context information from raw data (the guidelines presented in Section 4 could be useful in this direction).
- To decide (at design time) which should be the 'default' application behavior, which alternative behaviors to remain under consideration, and which behaviors may be ignored, one could get useful support through analyzing (considering random variables) the occurrence probabilities of end-user situations; on the basis of observations, statistical analysis can be applied in support of such decisions. As for the 'sensing' the end-user situations corresponding to these application behaviors, one could consider observing the values of appropriate parameters (the guidelines presented in Section 5 could be useful in this direction).
- To appropriately model the complex behavior of a context-aware application including 'switching' between alternative behaviors, one would require not only a powerful process modeling formalism but also an appropriate elaboration facility to be applied to complex points (the proposed in Section 6 combined application of Petri Net and Norm Analysis could be useful in this direction).

It is expected that these results would usefully support the current efforts to improve context-aware application development

However, all addressed challenges and corresponding solution directions must be considered in an integrated manner, as part of a context-aware application development approach, since they are interrelated. Hence, we plan (as further work) to use the results reported in the current papers for extending usefully an existing business-application-alignment approach [12].

Acknowledgements

This work is part of the Freeband A-MUSE project (http://a-muse.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services, Concepts, Architectures and Applications. Springer-Verlag, Berlin-Heidelberg (2004)

2. AWARENESS, Freeband AWARENESS project, http://www.freeband.nl/ project.cfm?id= 494&language=en (2008)
3. Broens, T.H.F., van Halteren, A.T., van Sinderen, M.J., Wac, K.E.: Towards an Application Framework for Context-Aware m-Health Applications. International Journal of Internet Protocol Technology, 2 (2) (2007)
4. Dey, A. K.: Understanding and Using Context. Personal Ubiquitous Computing 5 (1): 4-7 (2001)
5. Hristova, N., O'Hare, G.M.P.: Ad-me: Wireless Advertising Adapted to the User Location, Device and Emotions. In: HICSS'04, 37th Hawaii International Conference on System Sciences (2004)
6. Kurkovsky, S., Harihar, K.: Using Ubiquitous Computing in Interactive Mobile Marketing. Pers Ubiquit Compt, vol. 10, no. 1 (2006)
7. Levin, R.I., Rubin, D.S.: Statistics for Management. Prentice Hall, USA (1997)
8. Liu, K.: Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge (2000)
9. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: WMCSA'94, Workshop on Mobile Computing Systems and Applications (1994)
10. Shishkov, B. and Dietz, J.: Deriving Use cases from Business processes, the Advantages of DEMO. *In: Enterprise Information Systems V*. Eds. O. Camp, J.B.L. Filipe, S. Hammoudi, and M. Piattini. Kluwer Academic Publishers, Dordrecht/Boston/London (2004)
11. Shishkov, B., Quartel, D.: Combining SDBC and ISDL in the Modeling and Refinement of Business Processes. *In: Enterprise Information Systems VIII, Eds.: Y. Manolopoulos, J. Filipe, P. Constantopoulos, J. Cordeiro, Lecture Notes in Business Information Processing.* Springer-Verlag, Berlin-Heidelberg (2008)
12. Shishkov, B., Van SInderen, M. J.: From User Context States to Context-Aware Applications. *In: Enterprise Information Systems IX, Eds.: J. Cordoso, J. Cordeiro, J. Filipe, V. Pedrosa, Lecture Notes in Business Information Processing.* Springer-Verlag, Berlin-Heidelberg (2008)
13. Shishkov, B., Dietz, J.L.G., Liu, K.: Bridging the Language-Action Perspective and Organizational Semiotics in SDBC. In: ICEIS'06, 8th International Conference on Enterprise Information Systems (2006)
14. Shishkov, B., Van Sinderen, M.J., Quartel, D.: SOA-Driven Business-Software Alignment. In: ICEBE'06, IEEE International Conference on e-Business Engineering (2006)
15. Van Hee, K.M., Reijers, H.A.: Using Formal Analysis Techniques in Business Process Re-Design. *In: Business Process Management; Models, Techniques, and Empirical Studies, Eds.: W. van der Aalst, J. Desel, A. Oberweis, Lecture Notes in Computer Science.* Springer-Verlag, Berlin-Heidelberg (2000)
16. Van Sinderen, M.J.: Architectural Styles in Service Oriented Design. In: ICSOFT'06, International Conference on Software and Data Technologies (2006)
17. Van Sinderen, M.J., Van Halteren, A., Wegdam, M., Meeuwissen, E., Eertink, H.: Supporting Context-Aware Mobile Applications: An Infrastructure Approach. IEEE Communications Magazine 44 (9): 96-104 (2006).