NORM ANALYSIS SUPPORTING THE DESIGN OF CONTEXT-AWARE APPLICATIONS

Boris Shishkov, Marten van Sinderen
Department of Computer Science, University of Twente, Drienerlolaan 5, Enschede, The Netherlands

Kecheng Liu
Informatics Research Centre, The University of Reading, Whiteknights, Reading, U.K.

Hui Du
School of Economics and Management, Beijing Jiaotong University, Beijing, China

Keywords: Business modelling, Application design, Context-Aware applications, Norm Analysis.

Abstract: In this paper, we consider the challenge of designing context-aware applications, stressing especially on the usefulness of elaborating process models with semiotic norms. Such an elaboration can bring value in 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 this paper comprises an adaptability-driven methodological support to the design of context-aware applications.

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 (A-MUSE, 2007). Such a reaction is to 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. This means that the designer should not only specify each desirable behavior but also determine the rule patterns that govern the 'switching' between behaviors. Thus, the issues to be directly or indirectly addressed in this work are: (i) how to define each of the alternative behaviors; (ii) how to relate these alternative behaviors in an overall behavior (including the 'switching' between alternative behaviors); (iii) how to analyze these behaviors, applying appropriate (rule-driven) 'switching' patterns (where correctness is determined by the fact that an exhibited behavior matches the desirable behavior for a given context situation). We claim nevertheless that these challenges are interrelated since we consider the 'switching' rule pattern as naturally complementing the corresponding behavior flow patterns.

Hence, our particular focus is the design of CA applications, with a stress particularly on the rule-flow-driven elaboration of process models. We consider as a starting point the CA-application-design challenge in general, and we consider further on the need for such elaboration as well as a (proposed) possible way of realizing it.

Approaching this, we envision not only the conceptual problem of such modeling and elaboration but also the need to reflect a conceptual model in possible realizations in terms of modeling techniques. 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

Shishkov B., van Sinderen M., Liu K. and Du H. (2008).
NORM ANALYSIS SUPPORTING THE DESIGN OF CONTEXT-AWARE APPLICATIONS.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 334-337
DOI: 10.5220/0001713403340337

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 a Norm Analysis elaboration (Liu, 2000) to high-level behavior models including such ones that reflect context-driven behavior (characterized by alternative processes).

Hence, the main contribution of the current paper comprises an adaptability-driven methodological support to the design of CA applications, inspired by our addressing in combination several key issues that concern such a design. Such issues are the context-driven service delivery, the related (alternative) application behaviors, as well as the 'switching' among different alternative behaviors.

The paper's outline is as follows: Sect. 2 considers the design of CA applications. Sect. 3 addressed the added value of Norm Analysis in elaborating application behavior modeling. Sect. 4 presents the conclusion and outlines further research.

2 ON THE DESIGN OF CA APPLICATIONS

We will present our CA-application-design views on top of a more general modeling background concerning the specification of an automated (software) system. Such a specification is claimed to typically stem from a corresponding business model (Shishkov et al., 2006*b*). Thus, we will firstly consider the challenge of specifying an application, based on business modeling, and we are going to study secondly what else needs to be added for achieving context-awareness.

2.1 Business-modeling-driven Software Specification

Producing a sound relevant business model is claimed to be a *must* in specifying an automated (software) system (Shishkov et al., 2006*a*), and considering from this perspective the notions of *system* (the entities of main interest to us and their relations) and *environment* (the other entities and their relations) seems useful (Shishkov & Quartel, 2006).

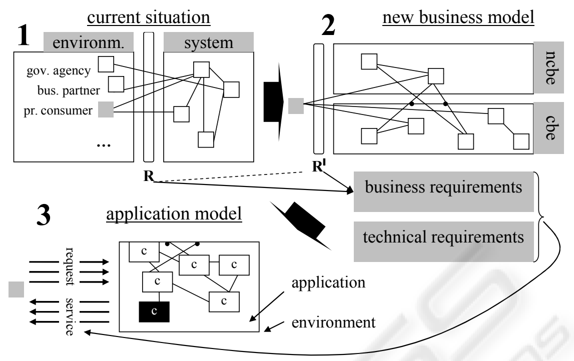


Figure 1: From business modeling to appl. modeling.

We have represented this in Figure 1 (1), as the 'current situation'. There, we have, 'within the system', some entities (entities could be humans, could be machines, could be anything), represented as squares. The entities have some relations among themselves – this is represented by solid lines. Assuming that the entities belonging to the system, deliver some service to the outside (just as a broker, an insurance company, or a hospital deliver services to the outside), we consider all the outside entities as belonging to the system environment; we might identify there the entity(s) 'consuming' the service, labelled as 'primary consumer' (represented as a grey square), entities that have partnership relations with entities belonging to the system, labelled as 'business partners', entities that have controlling functions, such as government agencies, for example, and so on. All these entities belong to the environment, have relations among themselves, and what is more important – they have relations with the system, in particular – relations to entities belonging to the system. These relations concern the service(s) that the system provides to its environment. These service(s) are restricted by corresponding system/environmental demands, such as quality standards, working hours, legal issues, which demands are labelled as *imposed requirements* – represented as 'R'. It should be noted that no automation is yet envisioned. Deciding to introduce such automation, the system architect typically abstracts from most entities belonging to the environment and mainly focuses on the primary consumer that turns out to become the main environmental entity, while the service delivered to the primary consumer is to essentially drive the further design activities. Hence, the primary consumer should 'face' all entities belonging to the system. However, the architect not always envisions automating all of them. Thus, the entities to be automated should be distinguished from the entities which are not going to be automated, as depicted in Figure 1 (2). There, '(n)cbe' stands for '(non)- to be – computerized business entities'. The primary consumer thus has relations not only to entities that will be automated but also to ones that will not be automated. These relations are to be restricted by the imposed requirements (R) which are to be updated nevertheless by the requirements presented to the system architect by the future user of the automated system under development – these requirements are labelled as *user requirements*. Both the imposed requirements and user requirements are reflected in the overall *business requirements*, represented as R^I. Hence, we arrive at a 'new business model' that might differ from what we associate with the 'current situation' not only because we have introduced more requirements and we have grouped the system entities but also because the entities from the 'current situation' are not necessarily mapped one-to-one to the entities in the 'new business model'. This is because often introducing software is not only about efficiency (to replace human(s) by software), it is also about innovation – the system architect may wish to consider introducing new services, re-arranging and/or updating the (observed) entities and their relations, and so on. Figure 1 (2) depicts therefore a new model that is only inspired by the 'current' model (Figure 1(1)). Further, the delimitation and representation of the entities that are to be automated should be driven not only by their relation to the primary consumer but also by their relation(s) to the rest of the entities – those entities that will not be automated – these 'interface' points are represented as black dots in Figure 1(2). As for the software specification model, it is to be derived from the new business model, and the system architect may wish to abstract from the entities that are not going to be automated – abstracting from them however means that the future software system will be adequately 'accessible' through the 'interface points' above mentioned, this is what the system architect should take care of (depicted in Figure 1(3) by the replication of the black dots). The application model (Figure 1(3)) depicts hence software components (c) and their relations – all mapped from the *cbe entities* (see Figure 1(2)). However, some software components may be introduced, which have no root in the new business model, they are represented as black square(s) – such components are introduced driven by technical requirements, those requirements that concern issues, such as platforms and operating systems to be used, for example. We have thus (in Figure 1(3)): (i) the application represented, as consisting of components; (ii) the relation to the primary consumer to whom the application should deliver service(s), restricted by corresponding business and technical requirements; (iii) the 'interface points' through which the outsider entities (different from the primary consumer) can collaborate with the application.

2.2 Towards Service Orientation and Context Awareness

An application modeled in the way considered in Sub-section 2.1, typically should not be expected to support context-awareness. This is because, as mentioned in the Introduction, supporting context-awareness means 'sensing' the context of the user and adapting (on this basis) the delivered behavior. Such sensing is usually driven by technology – for example, it is 'sensed' that Mary is at home or at work, by receiving some GPS-related support. Such kind of 'support' goes beyond the application and concerns a service infrastructure. Further, to manage context information through the infrastructure, one would often need sensors, the context data 'providers' whose role in supporting CA applications is very important. This all is illustrated in Figure 2 and further elaborated.

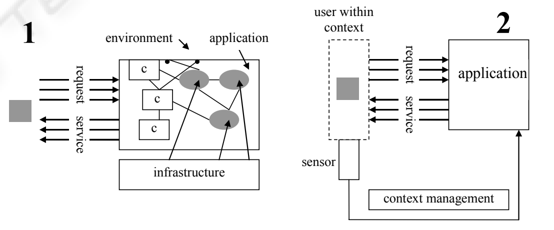


Figure 2: Towards service orientation and context awareness.

2.2.1 Towards Service Orientation

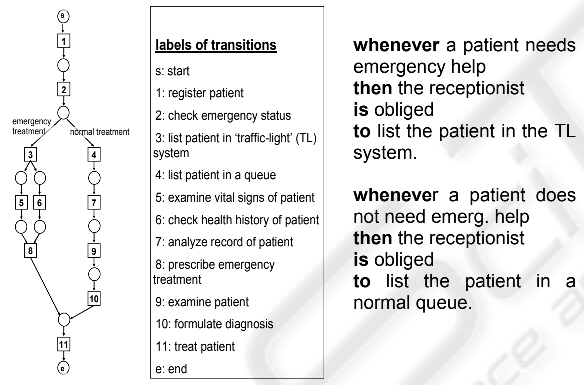
As Figure 2 (1) suggests, when an application is platform-dependent (this means that it runs on top of a service infrastructure), it realizes some of its functions not through its components but through addressing the infrastructure (and receiving some services from it) – this is depicted through the grey discs in the figure.

2.2.2 Towards Context Awareness

A platform-dependent application is not necessarily a context-aware application. To be context-aware, an application should adapt its behavior, based on information concerning the context of its primary consumer (user), which information the application receives usually through an infrastructure. This is shown in Figure 2 (2) where a sensor is also depicted – this is the entity that captures the context information and makes it available to the application.

3 MANAGING ALTERNATIVE BEHAVIORS

Taking into account that the proper 'switching' between alternative behaviors is to be adequately addressed by the designers of CA applications, an issue insufficiently elaborated in current approaches (Shishkov & Van Sinderem, 2007), we propose the usage of Norm Analysis (Liu, 2000) combined with Petri Net (Van Hee & Reijers, 2000), inspired by well-known relevant advantages of these techniques (introducing them is omitted for brevity), widely considered in literature.



labels of transitions

s: start
1: register patient
2: check emergency status
3: list patient in 'traffic-light' (TL) system
4: list patient in a queue
5: examine vital signs of patient
6: check health history of patient
7: analyze record of patient
8: prescribe emergency treatment
9: examine patient
10: formulate diagnosis
11: treat patient
e: end

whenever a patient needs emergency help
then the receptionist
is obliged
to list the patient in the TL system.

whenever a patient does not need emerg. help
then the receptionist
is obliged
to list the patient in a normal queue.

Figure 3: A typical health-care process.

Figure 3 (left) is presenting a typical health-care process, using Petri Net, and it is seen easily 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. 3 (left) can be identified and specified in detail – consider Fig. 3 (right).

4 CONCLUSIONS

By analyzing the design of software applications (and enriching this with innovative views that concern particularly CA applications) and proposing the combined use of two well-known modeling techniques, for the purpose of facilitating a modeling problem that is relevant especially to the design of CA applications, we have delivered in this paper, as a first step in an on-going research, some limited adaptability-driven methodological support to the design of such application. To further the reported research, we plan to work on bridging the current behavior-modeling-related results to previous results on identifying entities and relationships (Shishkov et al., 2007).

ACKNOWLEDGEMENTS

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

REFERENCES

A-MUSE project, 2007: http://a-muse.freeband.nl

Liu, K., 2000. Semiotics in information systems engineering, Cambridge University Press. Cambridge.

Shishkov, B. and van Sinderen, M.J. and Tekinerdogan, B., 2007. Model-driven specification of software services. In: IEEE Int. Conf. on e-Business Engineering, ICEBE 2007, 24-26 Oct 2007, Hong kong, China. pp. 13-21. IEEE Computer Society Press.

Shishkov, B. and van Sinderen, M.J., 2007. Model-Driven Design of Context-Aware Applications. In ICEIS'07, 9th Int Conf on Enterprise Inf Systems. INSTICC Press.

Shishkov, B., Quartel, D., 2006. Refinement of SDBC business process models using ISDL. In ICEIS'06, 8th Int. Conf. on Enterprise Inf. Systems. INSTICC Press.

Shishkov, B., Dietz, J.L.G., Liu, K., 2006. Bridging the Language-Action Perspective and Org. Semiotics in SDBC. In ICEIS'06, 8th Int. Conf. on Enterprise Information Systems. INSTICC Press.

Shishkov, B., Van Sinderen, M.J., Quartel, D., 2006. SOA-driven business-software alignment. In ICEBE'06, IEEE Int. Conf. on e-Business Engineering. IEEE Press.

Van Hee, K., H.A., Reijers, H., 2000. Using formal analysis techniques in business process re-design. W. van der Aalst et al. (Eds.): Business Proc. Management, LNCS 1806.