Towards a Unified Framework for Contextual Variability in Requirements

Raian Ali[1], Yijun Yu[2], Ruzanna Chitchyan[3], Armstrong Nhlabatsi[2], Paolo Giorgini[1]
[1] *DISI, University of Trento, Italy*
{raian.ali;paolo.giorgini}@disi.unitn.it
[2] *Department of Computing, The Open University, UK*
{y.yu;a.nhlabatsi}@open.ac.uk
[3] *University of Lancaster, UK*
r.chitchyan@lancaster.ac.uk

Abstract

Context is a significant factor in deciding the set of requirements relevant to a system (i.e., software product construction), the alternatives the system can adopt to satisfy these requirements, and the quality assessment of each alternative. By context we mean the conditions in the operating environment of an system that influences how the system should behave in different situations. However, the relationship between context and requirements can be challenging to capture and analyze. Presently this area of requirements engineering is largely under-researched. In this position paper, we discuss several ways by which context can be related to requirements and subsequently used for product derivation. We outline an approach that facilitates better understanding and use of contextual information in requirements. Our approach integrates three requirements engineering approaches - goal modeling, feature modeling, and problem frames - and is aimed at facilitating treatment of contextual variability in requirements.

1. Introduction

Requirements can be tightly coupled with the context of a software system, which has been considered as a major factor in determining which requirements are to be satisfied, how, and how well each of the alternatives satisfies them [1]. On the other hand, the system might change context while tasks are performed in satisfying these requirements.

In software product line engineering (SPLE) research, variability modeling is concerned with eliciting and representing a static (design-time) space of product variants and facilitating the product derivation based mainly on stakeholders choices. The context-related information is normally implicitly taken into consideration while making such choices. However, in dynamic (or runtime) SPL, the context plays an even more important role in product derivation, as each product configuration should be validated against the changing context in which it must function. Configurations that are inconsistent with the current context should be prevented from realisation.

Therefore an explicit notion of the relation between context and the product family model would allow for more systematic product derivation. For example, in feature modeling [2] an early consideration of context can determine if a feature is mandatory, optional or even unneeded. For instance, in an email editing system, encryption could be an optional feature if the system is to operate within one organization where staff trusts each other. On the other hand, it should be mandatory if users need to compose emails using a public network.

Context is the reification of the environment, that is, whatever provides a surrounding in which the system operates [3]. Initial research has already started on the relation between context and software variability at the requirements level. For example, Ali et al [4] investigate the relation between context and requirements at the beginning of goal-oriented analysis. Salifu et al [5] extends the application of the problem frames approach with context monitoring and switching problems. Similarly, Hartmann et al [6] propose that the concept of context variability is a major factor for deriving products in product lines engineering. All these approaches recognize the role of context for a decision maker to derive a system variant to better satisfy the system objectives. Yet, there are still a number of open problems related to contextual variability understanding, modeling, and analysis.

In this position paper, we outline an initial attempt to integrate various perspectives on contextual variability

into a unified framework in order to use context information in analysis and derivation of holistic products. By holistic product we mean a product which is derived with consideration of stakeholder intentions, desired functionality, quality properties, as well as understanding of how the software and its world context affect each other.

To this end, we study how a set of selected requirements engineering approaches (goals, features, and problem frames) treats context for requirements adaption (section 2). We show the potential of integration of this set of approaches into a framework that allows for better expressing and reasoning on contextual variability in requirements (section 3), and demonstrate this framework with an example problem of conflicted sharing of resources (section 4). A brief conclusion completes the paper (section 5).

2. Modeling Contextual Variability in Requirements

In this section, we compare three requirements engineering (RE) approaches that have considered the relation between requirements and context: goal modeling, feature modeling, and Problem Frames. This subset of RE approaches was selected due to the complementary perspectives they provide on expressing contextual variability in requirements.

Context of goals: the goal-based analysis elicits different alternatives to satisfy a goal, but it does not explicitly specify which alternative should be used for a particular case or context. Supporting alternatives without specifying when to follow each of them raises the question "why does the system support several alternatives?" On the other hand, the consideration of different contexts that the software has to adapt to without knowing supporting alternatives leads to the question "what can the system do if the context changes?" The work of Ali et al. [2, 7] proposes to elicit the relation between each alternative to goal satisfaction and the corresponding context, and provides constructs to analyze context and discover the data the system needs to monitor.

Context of features: features are characteristics of the system, and feature model represents the variability of these characteristics for configuring a family of software products. As we mentioned in the introduction, context influences the set of features to be included in a software product variant. Considering context at the design time can model a feature as mandatory or optional, whilst at the runtime context needs to be considered when switching to an alternative feature. As Hartmann et al. [4] suggest: studying the relation between context and features can support the engineering of software supply chains, which allows for more accurate derivation of a product that fits to the environment in which it operates.

Context in Problem Frames: Salifu et al. [3] apply Problem Frames approach to analyze different specifications that can satisfy the core requirements, under different contexts. The relationship between contexts, requirements, and the specification (machine) are represented by a problem description. Alternative problem descriptions corresponding to different contexts are elicited to identify variant problems. Variant problems are variations of the original problem adapted for a particular context. The specifications to the variant problem are then composed into a context-aware system. A change in context that violates the requirement triggers a switching action to an alternative specification for restoring the satisfaction of requirements. Here there is a clear distinction between the system and the world perceived as its context. The way that the system modifies the context is clearly described.

3. Integrating RE Approaches for Contextual Variability

Goal models capture stakeholder needs and intentions [8] at a time when variability of features in a product line-to-be has not been conceptualized. Relating goals to solution-oriented features leads to a requirement traceability problem [9].

The Problem Frames (PF) approach makes explicit the distinction between the Requirements (R), the World (W), and the Specification (S). They are related by the entailment relation W, S ⊢ R. Problems frames capture such a structural relation of a problem more explicitly than both goal models and feature models [10]. However, the PF approach has the notion of a 'variant problem' it does not natively support a hierarchy of variability as goal and feature modeling approaches do. For a comparison, Figure 1 summarizes the contribution that each of these modeling approaches can provide to the others, and their relations with context.

Besides its role of giving a rationale to features in the solution space and constraining, at the intentional level, the variability in problem frames, goal modeling can also represent quality requirements as softgoals that cannot have a clear-cut satisfaction criterion. The different goal satisfaction alternatives might contribute differently to reaching these softgoals. User preferences over alternatives might be expressed by prioritizing the quality measures, i.e. softgoals [11].

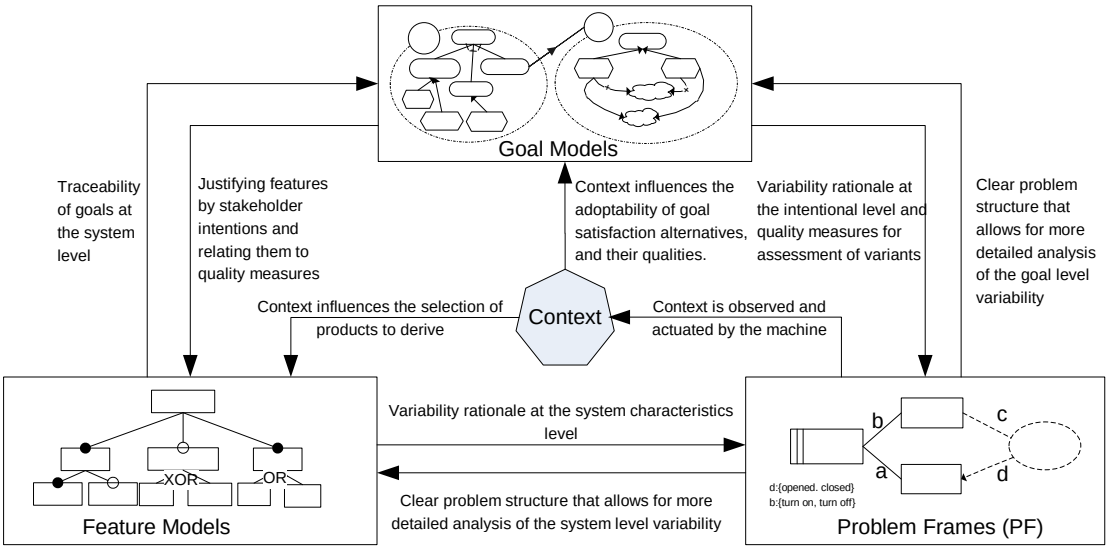

Figure 1. The integrated framework for contextual variability in requirements

Considering the context dimension, we believe that context influences human intentions and choices before the software is made. Consequently, context influences the variability at the goal level that in turn would help to manage the variability at the system level (features). Problem frames contain an explicit notion of the context, i.e. the world (W), and how requirements influences and are influenced by it. Therefore, PFs have the expressive power to represent the actions the system can do on the physical context, which is important for analyzing the bidirectional relation between requirements and context. In other words, context can determine the variability at the intentional and system level, while problems frames express the relationship between context, requirements, and specifications.

4. The Benefits of Integration: Example

The integration of the three models together with context has the potential for better expression of and reasoning about the requirements, which is potentially useful for product configuration choices. For instance such problems as conflicts between the system requirements on sharing the context objects can be detected and resolved early on. To illustrate this we sketch an example of a "smart home" - an automated adaptable living environment that supports patients with dementia. In this sketch (Fig. 2) the system might need to communicate with the caregiver and patients' relatives (see goal model in Fig. 2a). Since such communication can be required for different goals that are not alternatives, it may happen at the same time and for different intentions (e.g., to manage the patient's anxiety, and to arrange a social meeting). One way to establish the communication is by making a phone call (shown in Fig 2b). If phone is to be used for all communications, this may cause a conflict on this shared resource. Such a conflict can be easily detected when problem frames are used to depict the interaction between the system and its environment (Fig. 2 c).

In Fig. 2, we show how each of the three discussed approaches contributes to detection and resolution of such a conflict while configuring a product:

Problem frames have a clear distinction between the physical environment elements (e.g., phone) and the way the system interacts with them. This clear distinction helps the detection of potential conflicts on a shared element (i.e. exclusive use of the phone). Worth noting is that in order to ascertain that sharing of a resource does lead to a conflict, we need to model the behavior of the shared resource.

Feature models support representation of system alternative solutions that may help to avoid the detected conflict (simultaneous use of phone to contact the caregiver and patient's relatives). E.g., relative could normally be contacted via an SMS instead of establishing a voice call.

Goal model holds the upper level goals that the system alternatives of the feature model are meant to satisfy. Knowing the goals behind each feature is essential to get better conflict resolution. E.g., if the goal of calling a caregiver is to save the patient from extreme anxiety, and calling relative is for informing him/her about the next scheduled meeting, then the resolution policy could be postponing the call to the relatives.

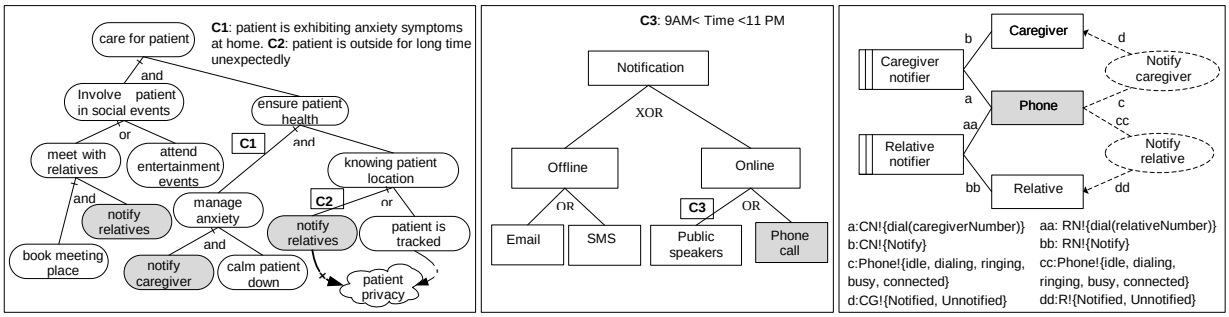

Figure 2. Modeling the requirements using the three integrated models.

Context can determine if a conflict might ever happen. For instance, if the call to the relatives is made to find out if the patient is visiting them in the context *"the patient is away from Smart Home for a long time"*, and the call to caregiver is to treat the patient in the context *"the patient is exhibiting anxious behavior inside the home"*, then there will be no conflict as the two contexts stimulating the two calls could never hold together (we assume that only one patient lives in each smart home). Moreover, context might decide the adoptability of alternatives. E.g. if issuing a public call for a caregiver through the healthcare institute speakers is adoptable only during the day, then this alternative might not be always possible as a way to resolve the conflict on using the phone.

Product Configuration: the integrated information provided by the three approaches is invaluable in configuring a product. For instance, knowing the details of goals for which the communication is needed, we can choose to always use email/SMS for meeting arrangement, always use public speakers for calling caregivers at day time, and always prioritize calls to caregiver in the night time over that calls to relatives.

5. Conclusions

In this paper, we have introduced our vision of the contextual variability in requirements and briefly discussed the treatment of the relation between requirements and context in a framework based on integration of three main-stream requirements engineering modeling languages. We discussed how one may benefit from this framework by better expressing requirements adaptation to context and being able to better reason and configure products through it. We remark here that extra modeling constructs and a methodological process are needed to map the three models and to enable the reasoning on the integrated model. Our future work is to look at provision of mechanisms to verify the proposed context variability models among specification, requirements and context.

6. Acknowledgements

This work has been partially funded by EU Commission, through the SecureChange, and DiVA projects, and by the PRIN program of MIUR under the MEnSA project. We would also like to thank Prof. Bashar Nuseibeh for the valuable discussions we had about this work.

7. References

[1] Fickas, S., Feather, M.: Requirements monitoring in dynamic environments. RE' 95, IEEE SC, 140.

[2] Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: Form: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng. 5 (1998) 143168

[3] Finkelstein, A., and Savigni, A.: A framework for requirements engineering for context-aware services. STRAW'01.

[4] Ali, R., Dalpiaz, F., Giorgini, P.: A Goal Modeling Framework for Self-Contextualizable Software. EMMSAD'09. LNBIP 29-0326, pp. 326–338. Springer .

[5] Salifu, M., Yu, Y., Nuseibeh, B. Specifying Monitoring and Switching Problems in Context Proc. RE'07, 211-220.

[6] Hartmann, H., Trew, T.: Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains. SPLC '08, pp. 12-21.

[7] Ali, R., Dalpiaz, F., Giorgini, P.: Location-based software modeling and analysis: Tropos-based approach. In: Li, Q., Spaccapietra, S., Yu, E., Olive, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 169-182. Springer, Heidelberg (2008)

[8] Yu, E. and Mylopoulos, J.: Why goal-oriented requirements engineering. REFSQ 1998, pp. 15–22

[9] Yu, Y., do Prado Leite, J.C.S., Lapouchnian, A., Mylopoulos, J.: Configuring features with stakeholder goals. SAC 08, 645649

[10] Classen, A., Heymans, P., Laney, R., Nuseibeh, B., Tun, T.: On the structure of problem variability: From feature diagrams to problem frames. VaMoS07, pages 109-117, Limerick, Ireland, January 2007.

[11] Hui, B., Liaskos, S., Mylopoulos, J.: Requirements analysis for customizable software goals-skills- preferences framework. RE 2003, IEEE SC, 117-126