Goal-Based Self-Contextualization

Raian Ali, Fabiano Dalpiaz and Paolo Giorgini

University of Trento - DISI, 38100, Povo, Trento, Italy
{raian.ali, fabiano.dalpiaz, paolo.giorgini}@disi.unitn.it

Abstract. System self-contextualizability is the system ability to autonomously adapt its behavior to the uncontrollable relevant context to keep its objectives satisfied. Self-contextualizable system must have alternative behaviors each fitting to a set of contexts. We propose to start considering context at the level of requirements engineering, adopting Tropos goal model to express requirements and complementing it with our proposed context analysis. We define variation points on goal model where a context-based decision might need to be taken, and propose constructs to analyze context. While goal analysis provides constructs to hierarchically analyze goals and discover alternative sets of *tasks* to be *executed* to *satisfy* a goal, our proposed context analysis provides constructs to hierarchically analyze context and discover alternative sets of *facts* to be *monitored* to *verify* a context.

1 Introduction

Self-contextualizable software is able to monitor context and to autonomously adapt to it in order to keep its design objectives satisfied. This mainly means that the developed software has a space of alternative behaviors each fitting to a specific context. Consequently, and to allow for a systematic behavior adaptation to context, we need to specify the relation between each behavior and the context where such behavior is adoptable.

Context modeling, (e.g. [1]), concerns finding modeling constructs to represent context, but there is still a gap between the context models and the software behavior models, i.e. the relation between context and its use is missed. We need to reduce such gap and to allow for answering questions like: "*how do we decide the relevant context?*", "*why do we need context?*" and "*how does context influence software and user behavior adaptation?*". Salifu et al. [2] investigate the use of problem descriptions to represent and analyze variability in context-aware software; the work recognizes the link between requirements and context as a basic step to design context aware systems.

Goal analysis (*i** [3], Tropos [4], and KAOS [5]), provides a way of analyzing high level goals to discover and represent alternative sets of tasks that can be executed to achieve such goals. Goal model is used to represent the rationale of both humans and software systems, and is a mainstream technique for requirements engineering and helps for representing software design alternatives. These features are important for self-contextualizable software that must have alternatives and a rationale that reflects users and software adaptation to context in order to adopt a useful execution course [6]. From a goal perspective, a self-contextualiazable software is assigned a set of goals and has to keep them satisfied in a way tailored to the context. However, there is still a gap between goal analysis that discovers the space of alternatives to reach a goal, and the context where each alternative is adoptable.

Customizing goal models to fit to user skills and preferences was studied in [7, 8]. The selection between goal satisfaction alternatives is based on one dimension of context, i.e. user skills, related to the executable tasks of the goal hierarchy, and on user preferences which are expressed over softgoals. Lapouchnian et al. [9] propose techniques to design autonomic software based on an extended goal modeling framework, but the relation with the context is not focused on. Liaskos et al [10], study the variability modeling under the requirements engineering perspective and propose a classification of the intentional variability when Or-decomposing a goal. Differently, we focus on context variability, i.e. the unintentional variability, which influences the applicability and appropriateness of each goal satisfaction alternative.

In this paper, we extend our previous goal modeling framework [11–13] to support the self-contextualizability at the goal level. We use goal analysis in conjunction with our proposed context analysis to build self-contextualizable goal models. We provide constructs to hierarchically analyze context at the variation points of Tropos goal model. The proposed hierarchial context analysis will help us to identify the data the system has to monitor to verify each context, i.e. it helps to identify the monitoring requirements. In what follows, we describe our vision and proposed framework.

2 Self-Contextualizable Goal Model

Self-contextualization, from a goal oriented perspective, is the autonomous selection between goal satisfaction alternatives to fit to the variable context. The basic idea is to keep the goals satisfied in different contexts through an adaptable course of execution. Here we give four principles for self-contextualizability from a goal perspective:

Context effects: context influences the set of satisfiable goals, the need to satisfy each goal, the possible goal satisfaction alternatives, and the quality of each alternative.

Context is not adaptable: context does not adapt to users and software; rather, users and software have to adapt to context to satisfy, and/or to better satisfy, their goals.

Users contextualizability comes first: defining software self-contextualizability has to be based on how its users behave in different contexts. We need to analyze users goals to discover a space of satisfaction alternatives, and to analyze how context influences users in deciding and choosing among different alternatives for satisfying their goals. Self-contextulaizable software has to reflect users adaptation to the context first, and then adapt itself to context.

Contextualization times: software contextualization can be done while deploying the software, that supports different alternatives, in a certain environment to adapt it to some contextual characteristics that never change, i.e. some alternatives are not usable and redundant in that environment. Other variable characteristics necessitate keeping different alternatives and adopting one of them at runtime based on which context is valid. In both cases, the systematic contextualization needs specifying the relation between goal alternatives and context.

Fig. 1 represents a partial goal model for a promotion information system that will interact with customers and sales staff, through their PDAs, for promoting products with a number of alternatives. To make it self-contextualizable, we need to explicitly represent the relation between these alternatives and context. Contexts can be related to the following variation points of Tropos goal model:

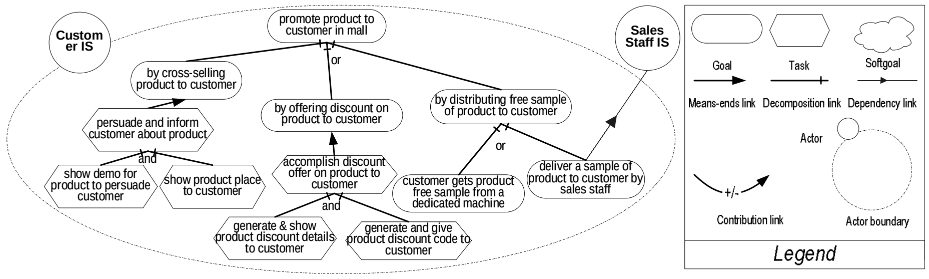
Custom er IS
promote product to customer in mall
or
Sales Staff IS
by cross-selling product to customer
by offering discount on product to customer
by distributing free sample of product to customer
persuade and inform customer about product
and
show demo for product to persuade customer
show product place to customer
accomplish discount offer on product to customer
and
generate & show product discount details to customer
generate and give product discount code to customer
or
customer gets product free sample from a dedicated machine
deliver a sample of product to customer by sales staff
Goal
Task
Softgoal
Means-ends link
Decomposition link
Dependency link
Actor
+/-
Contribution link
Actor boundary
Legend

Fig. 1. A partial goal model of the promotion scenario

1. *Or-decomposition*: Or-decomposition is the basic variability construct, we still need to specify in which context each alterative in an Or-decomposition can be adopted.
2. *Actors dependency*: in some contexts, an actor might attain a goal / get a task executed by delegating it to another actor.
3. *Goal activation*: an actor, and depending on the context, might find necessary or possible triggering (or stopping) the desire of satisfying a goal.
4. *And-decomposition*: a sub-goal / sub-task might (or might not) be needed in a certain context, that is, some sub-goals / sub-tasks are not always mandatory to fulfil the top-level goal / task in an And-decomposition.
5. *Means-end*: goals can be ultimately satisfied by means of specific executable processes (*tasks*). The adoptability of each task might depend on the context.
6. *Contribution to soft-goals*: the contributions to the softgoals can vary from one context to another. We need to specify the relation between the context and the value of the contribution.

We need to analyze context to discover, represent, and to agree on how it can be verified. Differently from the research in context modeling (see the survey [14]), we do not provide an otology or a modeling language for representing context, but modeling constructs to hierarchically analyze context. We specify context by a high level *statement* and start to refine it to get by the end alternative sets of *facts* that are definable on monitorable data. The truth value of facts is bottom-up propagated to define the truth value of the high level statement (see Fig. 2). We also need to keep track of the elements of environment the different goals, tasks, statements and facts are concerned with. To do this, we propose to *parameterize* goals, tasks, statements, and facts. Parametrization is necessary to decide and make it explicit the domain of discourse of goals, tasks, statements, and facts, i.e. the elements of the environment under discussion.

To better explain the approach, let us consider the Or-alternative *" Promote by cross-selling the product to customer"* which concerns the elements: *product*, and *customer*. The parameterized goal is *"Promote by cross-selling the product [p] to customer [c]"*. A simplified analysis of the context where this alternative is adoptable is shown in Fig. 2.a. Moreover, by analyzing the elicited facts, the analyst can extract the conceptual data model the monitoring system has to instantiate to make facts validation possible. The analysis of the last context facts leads to the conceptual data model of Fig. 2.b.

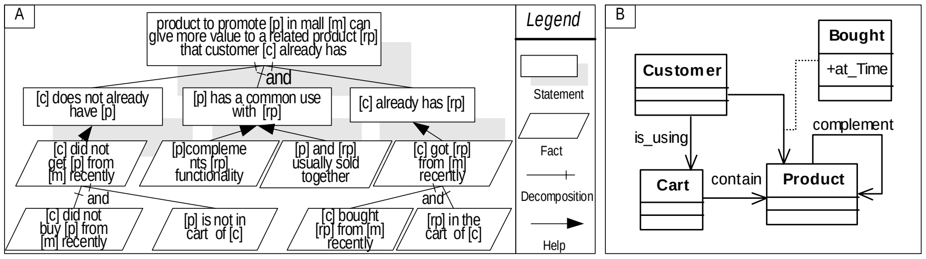
A
product to promote [p] in mall [m] can give more value to a related product [rp] that customer [c] already has
and
[c] does not already have [p]
[p] has a common use with [rp]
[c] already has [rp]
[c] did not get [p] from [m] recently
[p]compleme nts [rp] functionality
[p] and [rp] usually sold together
[c] got [rp] from [m] recently
and
[c] did not buy [p] from [m] recently
[p] is not in cart of [c]
and
[c] bought [rp] from [m] recently
[rp] in the cart of [c]
Legend
Statement
Fact
Decomposition
Help
B
Bought
+at_Time
Customer
complement
is_using
Cart
contain
Product

Fig. 2. Statement analysis (a). Correspondent data conceptual model (b).

We provide a set of modeling constructs to analyze high-level context statements and to elicit the monitorable facts that are used to assign truth values to statements. Context, the reification of the environment that surrounds the system and influences it, can be monitored but not controlled by the system itself [15]. Under this assumption, systems cannot change the context but should adapt to it for satisfying users objectives.

Definition 1 (Fact) *a boolean predicate specifying a current or a previous context, whose truth value can be computed objectively.*

The objective method to compute a fact truth value requires monitoring some characteristics and history of a set of relevant environment elements.

Definition 2 (Statement) *a boolean predicate specifying a current or a previous context, whose truth value can not be computed objectively.*

Statement verification could not be objectively done because the system is not able to monitor and get all the data needed to compute the truth value of a statement, or because there could be no consensus about the way of knowing the truth value of a statement. Anyhow, to handle such problem we adopt a relaxed confirmation relation between facts, which are objectively computable by definition, and statements, in order to assign truth values to statements. We call this relation *"help"* and define it as following:

Definition 3 (Help) *Let f be a fact, s be a statement.* $help(f, s) \iff f \rightarrow s$

The relation $help$ is strongly subjective, since different stakeholders could define different $help$ relations for the same statement.

Definition 4 (And-decomposition) *Let* $\{s, s_1, \ldots, s_n\},\ n \geq 2$ *be statements (facts).*
$and\_decomposed(s, \{s_1, \ldots, s_n\}) \iff s_1 \wedge \ldots \wedge s_n \rightarrow s$

Definition 5 (Or-decomposition) *Let* $\{s, s_1, \ldots, s_n\},\ n \geq 2$ *be statements (facts).*
$or\_decomposed(s, \{s_1, \ldots, s_n\}) \iff \forall i \in \{1, \ldots, n\}, s_i \rightarrow s$

Here we give some examples to illustrate the previous concepts:

- *"two products, [p1] and [p2], are usually sold together"* is a fact the system can verify through checking the sales record of all customers and check if the two products are often sold together.
- *"customer [c] is interested in the product [p]"* is a statement since the system can not objectively compute its truth value. This statement can be Or-decomposed into *"customer [c] is behaviorally interested in product [p]"* and *"customer [c] is historically interested in product [p]"* substatements. The system can get some evidence of the first substatement through the help of the fact *"[c] holds [p] for long time"*, or the fact *"[c] comes to [p] area so often"*. The second substatement can be verified through the fact *"[c] buys periodically [p]"* or the fact *"[c] buys usually of [p] category"*.
- *"customer [c] does not have the product [p]"* is a statement that can be verified through the fact *"[c] did not get [p] recently from the mall [m]"*, that is And-decomposed into *"[c] did not buy [p] recently from [m]"* and *"[c] does not have [p] in his/her shopping cart"*. However both facts do not ensure that the customer does not have the product (e.g., the system cannot verify if the customer was given the product as a gift, or bough it from another mall).

Our proposed hierarchical context analysis has the potential to make a context (i) more understandable for the stakeholders, (ii) easily modifiable as it is not given as one monolithic block, and (iii) more reusable as parts of the statement analysis hierarchy can be also used for other variation points or other stakeholders context specifications. Moreover, the analysis justifies *why* the monitoring system has to capture environmental data, as such data are needed to verify facts that in turn confirm or disapprove statements that are needed to make a decision at the variation points of the goal model. Specifying for each fact the related fragments of the data conceptual model is useful for purpose of tracking. For example, if for some reason, a group of stakeholders decided to drop, to alter, or to reuse one alternative, statement, or fact, we still can track which fragments in the conceptual data model could be influenced.

3 Conclusions and Future Work

In this paper, we have introduced a goal-oriented framework for self-contextualizable systems. We have discussed the self-contextualizability from goal-based perspective. We have proposed the association between goal satisfaction alternatives and context. In turn, context is defined through statement analysis that elicits the alternative sets of facts the system has to validate on monitorable data so to confirm the high level statements. As future work, we will define models covering all development phases of self-contextualizable software and a process to facilitate the development of high quality self-contextualizable software. We also want to extend the formalization and reasoning mechanisms we proposed in [11, 12] to fit to the modeling framework introduced in this paper. We will work on complex case studies in order to better validate our approach. Finally, statements and facts suffer of interaction problems; therefore, supporting tools and reasoning techniques should be proposed to support the design and verification of models.

Acknowledgement

This work has been partially funded by EU Commission, through the SERENITY, and COMPAS projects, and by the PRIN program of MIUR under the MEnSA project. We would also like to thank Prof. Jaelson Brelaz de Castro for the valuable discussion we had about this work.

References

1. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: Proc. Second IEEE Intl. Conference on Pervasive Computing and Communications (PerCom'04). (2004) 77
2. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In: Proc. 15th Intl. Conference on Requirements Engineering (RE'07). (2007) 211–220
3. Yu, E.: Modelling strategic relationships for process reengineering. Ph.D. Thesis, University of Toronto (1995)
4. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems **8**(3) (2004) 203–236
5. Dardenne, A., Van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of computer programming **20**(1-2) (1993) 3–50
6. Fickas, S., Feather, M.: Requirements monitoring in dynamic environments. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, IEEE Computer Society Washington, DC, USA (1995) 140
7. Hui, B., Liaskos, S., Mylopoulos, J.: Requirements analysis for customizable software goals-skills-preferences framework. In: RE, IEEE Computer Society (2003) 117–126
8. Liaskos, S., McIlraith, S., Mylopoulos, J.: Representing and reasoning with preference requirements using goals. Technical report, Dept. of Computer Science, University of Toronto (2006) ftp://ftp.cs.toronto.edu/pub/reports/csrg/542.
9. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. In: Proc. 2006 conference of the Center for Advanced Studies on Collaborative research (CASCON '06), ACM (2006) 7
10. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On goal-based variability acquisition and analysis. In: Proc. 14th IEEE Intl. Requirements Engineering Conference (RE'06). (2006) 76–85
11. Ali, R., Dalpiaz, F., Giorgini, P.: Location-based variability for mobile information systems. In Bellahsene, Z., Léonard, M., eds.: CAiSE. Volume 5074 of Lecture Notes in Computer Science., Springer (2008) 575–578
12. Ali, R., Dalpiaz, F., Giorgini, P.: Modeling and analyzing variability for mobile information systems. In Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L., eds.: ICCSA (2). Volume 5073 of Lecture Notes in Computer Science., Springer (2008) 291–306
13. Ali, R., Dalpiaz, F., Giorgini, P.: Location-based software modeling and analysis: Tropos-based approach. In Li, Q., Spaccapietra, S., Yu, E., Olivé, A., eds.: ER. Volume 5231 of Lecture Notes in Computer Science., Springer (2008) 169–182
14. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp. (2004)
15. Finkelstein, A., Savigni, A., Street, G.: A framework for requirements engineering for context-aware services. Proc. 1st Int. Workshop on From Software Requirements to Architectures (STRAW) (2001)