Context for Goal-level Product Line Derivation

Raian Ali[1], Ruzanna Chitchyan[2], Paolo Giorgini[1]
[1]*DISI, University of Trento, Italy;* [2]*University of Lancaster, UK*
[1]*{raian.ali; paolo.giorgini}@disi.unitn.it;* [2]*rouza@comp.lancs.ac.uk*

Abstract

Product line engineering aims at developing a family of products and facilitating the derivation of product variants from it. Context can be a main factor in determining what products to derive. Yet, there is gap in incorporating context with variability models. We advocate that, in the first place, variability originates from human intentions and choices even before software systems are constructed, and context influences variability at this intentional level before the functional one. Thus, we propose to analyze variability at an early phase of analysis adopting the intentional ontology of goal models, and studying how context can influence such variability. Below we present a classification of variation points on goal models, analyze their relation with context, and show the process of constructing and maintaining the models. Our approach is illustrated with an example of a smart-home for people with dementia problems.

1. Introduction

Each software system is situated in a context. Context is the reification of the environment that is whatever provides a surrounding in which the system operates [23].

In product line engineering, context can be a main factor in deciding what product variants to derive. In other words, context influences the need for, the applicability and the appropriateness of, each variant. We use the term self-contextualizability to denote a system ability to adapt to context in order to keep its objectives satisfied. Self-contextualizable product line is a product line that incorporates the reasoning needed to derive product variants fitting to their contexts. Consequently, the relation between products variants and context has to be explicitly captured and reasoned on to derive contextualized products.

Software variability modeling, e.g., feature modeling [10, 16], addresses the problem of modeling the space of possible product variants and facilitating derivation of a product upon stakeholders choices. However, there is still a gap between each variant and the context where it can, or has to, be adopted. Speaking in terms of feature modeling, context can determine if a feature is mandatory or optional or even redundant. E.g. for an email editing system, encryption could be an optional feature if the system is to operate within one organization where staff trust each other. On the other hand, it could be mandatory if the editor is for users who will write emails from a public network.

Goal models (e.g., i* [21], Tropos [3], and KAOS [5]) are used as an intentional ontology that fits well with the early requirements analysis phases. They support analysis of different alternatives for satisfying user needs [20]. As proposed in [22], a goal model is a good starting point for feature model construction as it justifies feature configurations in terms of stakeholder goals. Thus, in this paper we advocate a perspective on a goal model as a core domain model: it acts as the source of all stakeholder-related variability. Goal models justify existence of all functional requirements (hard goals) and quality measures (soft goals) of a software system in terms of stakeholder intentions. Thus, variability in intentions is a primer source of system variability. Of course, goals are not the sole source of system variability – they pertain to variability of the problem domain. Technical solutions devised to satisfy these goals will add their own variability dimensions. The later however, are meaningless without the former, as a system will be useful only if it is addressing some set of stakeholder needs.

Thus, in the rest of this paper the goal models are perceived as the initial sources of variability models. Consequently, capturing the relation between context

and the goals is directly relevant to the system models (e.g., feature models) based on such goal models.

The integration between Tropos [3] goal model and context as a way to capture the relation between context and variability at the intentional level was proposed in [1, 2]. In this paper, we discuss this integration between goal model and context from the perspective of software product lines, present a classification of variation points on goal models, analyze their relation with context and show how these points can be used to help for a systematic derivation of product variants that fit to their contexts. Thus, this work is directly relevant to the topic of dynamic software product lines as each of the derived goal model variants is suited to a particular context, and can be dynamically substituted into the running system (i.e., a transition of system from the current to the newly identified goal model could be initiated) when the appropriate context occurs[1]. Furthermore, we show the process of constructing, using, and maintaining the self-contextualizable goal model, and illustrate our concepts and approach on a running example of smart-home.

The paper is structured as follows: in Section 2 we overview Tropos goal modeling concepts and introduce the smart-home scenario. In Section 3 we discuss and provide modeling construct for capturing the relation between context and variability at the goal level. In Section 4, we show the methodological process of constructing, using, and maintaining the goal-based self-contextualizable product lines. We discuss related work in Section 5 and conclude in Section 6.

2. Tropos Goal Modeling: Overview

Goal analysis represents a paradigmatic shift with respect to object-oriented analysis. While object-oriented analysis fits well to the late stages of requirement analysis, the goal-oriented analysis is more natural for the earlier stages where the organizational goals are analyzed to identify and justify software requirements and position them within the organizational system [15]. Tropos goal analysis projects the system as a set of interdependent actors, each having its own strategic interests (goals). Goals are analyzed iteratively and in a top-down way, to identify the more specific sub-goals needed for satisfying the upper-level goals. Goals can be ultimately satisfied by means of specific executable processes (tasks).

To clarify Tropos goal modeling concepts and exemplify our proposed approach in the rest of the paper, we take a scenario of a smart home [11]. A smart home is a ubiquitous computing area where computing is integrated with the environment and context is considered as an implicit input that influences what the system needs to do and how it does it. Amongst their different scenarios, smart homes were proposed for elderly, health care, entertainment, and so on.

We consider here a smart home designed for patients with dementia problems. The smart home supports some daily tasks that the patient might forget to do, like refreshing the air inside the home. Besides their memory impediments, the patients suffer from unexpected anxiety attacks. The smart home has to manage such situations by making the patient aware of the anxiety attack, or by preventing him/her from getting out of the house in an unusual way. The home then needs to calm the patient, and call the caregiver to come and administer a treatment. The smart home supports also some general tasks, like preventing a potential robbery by giving illusion that the home is lived in when the patient is away for a long time, and asking the police or a neighbor smart home for help when a suspected robber enters the home area.

To illustrate the main concepts of goal analysis used in this paper, in Fig. 1 we demonstrate a partial Tropos goal model for the smart home. In this model the actors ("Patient Smart Home" and "Neighbor Smart Home") have a set of top-level goals ("Manage home"), which are iteratively decomposed into subgoals by and-decomposition and or-decomposition. In and-decomposition all subgoals should be achieved to fulfil the top goal, while in or-decomposition at least one subgoal should be achieved to fulfil the top goal. For instance, the goal "Protect home against robbery" is and-decomposed into the subgoals "Give illusion of being lived in" and "Act against potential robbery"; the goal "Enforce routine exit procedure" is or-decomposed into the subgoals "Patient is alerted" and "Patient is prevented from exiting". Goals are finally satisfied by means of executable tasks; the goal "Refresh air inside home" can be reached via one of two tasks: "Open windows" or "Turn on air ventilator".

A dependency indicates that an actor (depender) depends on another actor (dependee) to attain a goal or to execute a task: the actor "Patient Smart Home"

[1] The topics of runtime system transition and context monitoring are not discussed further in this paper.

depends on the actor "Neighbor Smart Home" for achieving the goal "Request assistance".

Soft-goals are qualitative objectives for whose satisfaction there is no clear cut criteria (e.g., "Patient privacy" is a rather vague objective), and they can be contributed either positively or negatively by goals and tasks: "Open windows" usually contributes negatively to "Patient privacy", while "Turn on air ventilator" usually contributes positively to it.

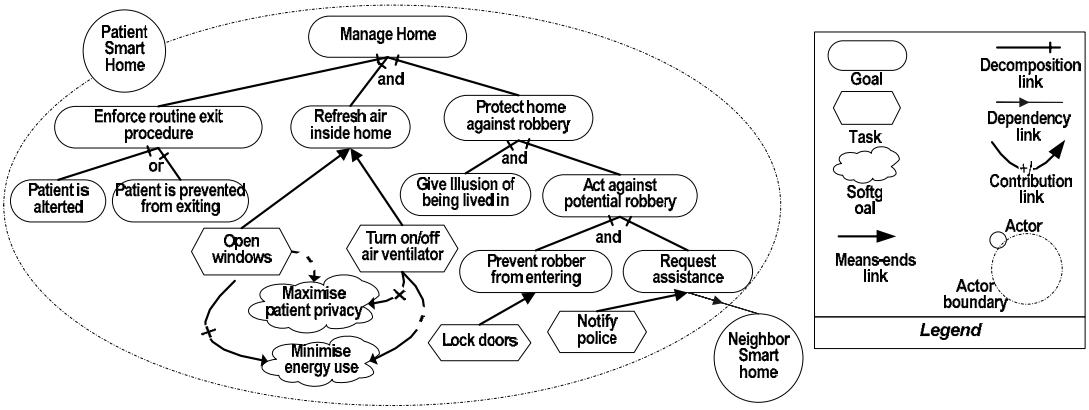

Fig. 1: Tropos goal model example

3. Goal-based Self-Contextualization

Goal analysis allows for different alternatives (model variants) to satisfy a goal, but does not specify when each alternative can be adopted. Since context can influence human goals and choices even before the software system is designed, the context has to be considered early in the analysis where stakeholders goals are identified and analyzed. We suggest that while analyzing goal-satisfaction alternatives it is necessary to specify the relation between each alternative and the corresponding context. This not only improves understanding of goal variants and their related contexts, but also facilitates construction of self-contextualizable systems, i.e., systems that are able to ensure satisfaction of their goals in changing context.

3.1. Context variability points in goal models

Our work leads to integration between goal analysis as an early analysis model, and context in order to enable a systematic derivation of goal model variants for each context. In this paper we use the term *"model variant"* to refer to an And-tree of the goal model hierarchy, i.e., deterministic tree. Fig. 2 depicts a context-annotated Tropos goal model for the smart home system described in the previous section. Here we explicitly represent the relations between goals and the context, where context is denoted by φ_i (see Table 1 for full set of relevant context for the discussed example). We classify contexts into three categories, each one associated with a different set of variation points in the goal model:

1. **Stimulating context**: this is the context that stimulates a set of requirements. For instance, a particular root goal could be "stimulated" only if its relevant context is in place. The points at which such context arises are:

a) *Root Goals/Tasks*: depending on the context, an actor might decide to satisfy a root goal/execute a root task or not. E.g., to stimulate the root goal "Manage home", the home has to be lived in with no awake caregiver or healthy relative inside, and the inhabitant is expected to have some dementia problem ($\varphi 0$).

b) *And-decomposition*: a sub-goal/sub-task in an And-decomposition might (or might not) be needed only in a certain context, that is some sub-goals/sub-tasks are not always mandatory for the fulfillment of the top-level goal/task in an And-decomposition. E.g., the goal G1 needs to be satisfied only if the patient is anxious and is behaving in unusual way ($\varphi 1$), while the goal G2 has to be satisfied only when the level of humidity inside home is more than an acceptable level, or the home windows and doors have not been opened for a long time ($\varphi 2$). The satisfaction of goal G6 is needed when the patient is away from home for a long time and it is night ($\varphi 7$), while G7

is needed when some person is trying to get into the yard in a suspicious way (e.g., entering from elsewhere than secured external gate) (φ8). The task T6 is executed if the light level is too low or too high (φ 11), and the task T12 is executed if it is too dark in the house (φ 12).

2. **Required context**: while stimulating contexts stipulates the conditions for the need for a set of requirements, a required context is itself needed to make it possible to satisfying stimulated requirements. For one model variant, the required context is the conjunction of contexts associated to the following variation points:

   a) *Or-decomposition*: each subgoal/subtask in Or-decomposition might require a valid context to be adoptable. E.g., the alternative subgoal G4 is adoptable when the patient dementia disease stage is not advanced and s/he is not extremely anxious (φ3); while the alternative G5 is adoptable if the patient suffers from advanced dementia, or s/he seems to be extremely anxious (φ4). Thus, though the goal model may have an or-decomposition, the availability and utility of the or-decomposition choices will have additional limitations stemming from context.

   b) *Means-end*: goals can be ultimately satisfied by means of specific executable processes (tasks). The adoptability of each task in means-end analysis might depend on the context. E.g., the task T8 is adoptable if the day is sunny and not too windy (φ5), the task T4 is adoptable if the phone is free and the caregiver is not using his/her phone for a call (φ9), and the task T5 is adoptable if it is not late in night.

   c) *Actors dependency*: in some contexts, an actor might attain a goal/get a task executed by delegating it to another actor. E.g., asking the help of a neighbor to act against the potential robbery is adoptable if the neighbor is healthy, stays at home, and can see or easily reach the patients' home (φ 13).

3. **Quality context**: context can influence the quality of each possible way of satisfying the requirements, i.e. the quality of each model variant. The variation point corresponding to this effect of context is:

   a) *Contribution to softgoals*: softgoals are qualitative objectives for their satisfaction there is no clear cut criteria. The positive and negative contributions to softgoals from other goals/tasks can vary from one context to another. We need to specify the relation between the context and the value of the contribution. E.g. the task T8 contributes positively to the softgoal SG1, if the patient is outside home (φ6) while it contributes negatively in the other case. The task T9 contributes negatively to the softgoal SG2 only if the day is sunny and not too windy (φ5).

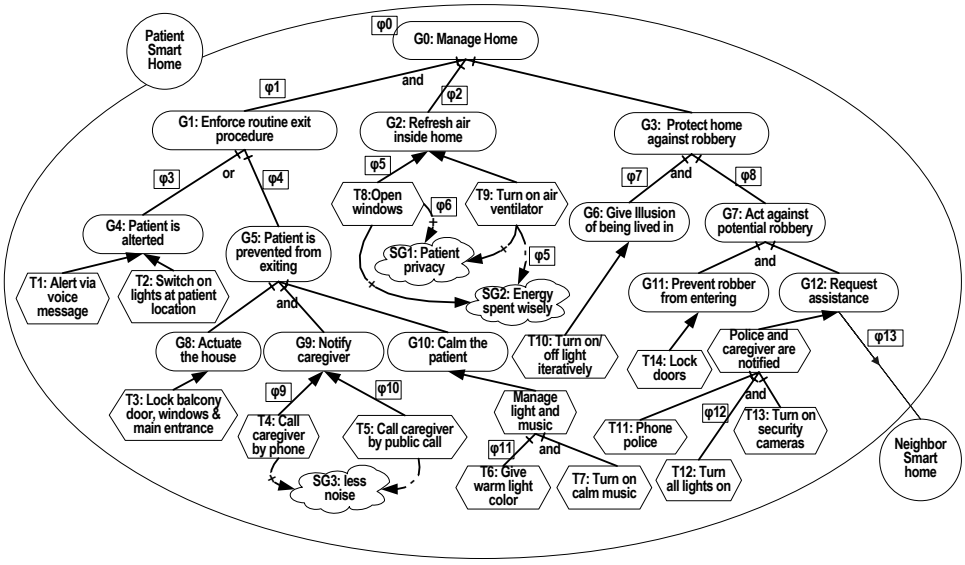

Fig. 2: Tropos goal model with context annotation on variation points

Φ 0	Home is lived in, and the patient is expected to have some dementia problem, and there is no awake caregiver or healthy relative at home.
Φ 1	Patient is anxious and behaving in unusual way.
Φ2	The level of humidity at home is more than the acceptable level, or the home windows and doors have not been opened for a long time.
Φ3	The patient's dementia disease stage is not advanced and s/he is not extremely anxious.
Φ4	The patient suffers of advanced dementia, or s/he seems to be extremely anxious.
Φ5	It is a sunny and not too windy day.
Φ6	The patient is out of the home.
Φ7	The patient is out for a long time and it is night time.
Φ8	A person is trying to get into the yard in a suspicious way, e.g., entering from a different place than the secured external gate.
Φ9	The phone is free and the caregiver is not using his/her phone for a call.
Φ10	It is not late at the night.
Φ11	The light level at patient's location is too low or too high.
Φ12	It is too dark inside of the home.
Φ13	The neighbor is healthy, stays inside home, and can see or reach the patient's home easily.

Table 1: Summary of contexts for Smart Home goal model

Fig. 3 shows two possible model variants for one stimulating context (see contexts detailed in Table 1). We spread the contextual And-decomposition and used artificial nodes (small circles) only for the purpose of more understandable presentation of the model. The classification of context in these 3 categories allows us to answer questions like: in a given context, is any variant stimulated? what are the possible alternatives/variants if one stimulus holds? and what is the quality of each one.

3.2. Context analysis

Having annotated goals with context, we then need to consider how to model and analyze the context itself in order to discover, represent, and agree on how it can be verified. This work has been discussed in [1, 2] and is briefly summarized in this subsection. In our work on context analysis, we provide constructs to hierarchically analyze context and discover alternative sets of facts the system has to monitor to verify a context. This is different from the other approaches in context modeling (for a survey see [18]) where either ontology or a modeling language for representing context is developed without the elicitation hierarchy like the one we propose. A simplified analysis of the context "patient is anxious" is shown in Fig. 4 (a).

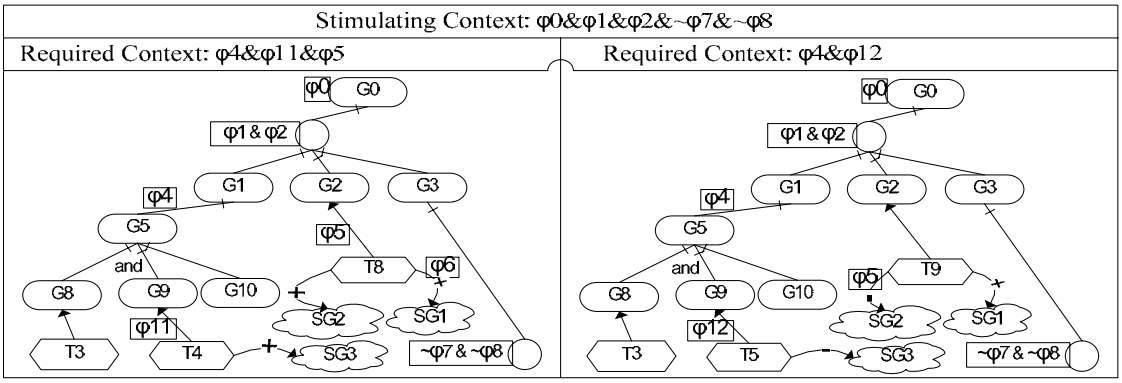

Fig. 3: Two model variants and their contexts

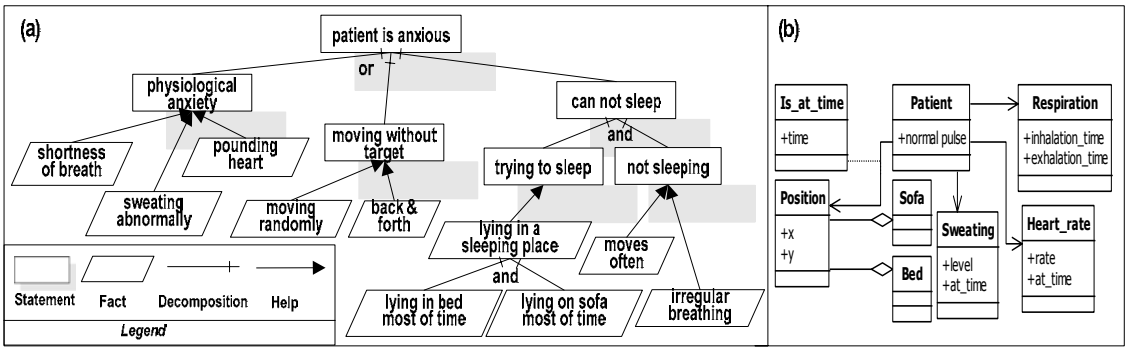

Fig. 4: A context analysis hierarchy for "patient is anxious"

The following set of modeling constructs is used to analyze a high level context and identify the atomic verifiable facts that give its truth value:

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/or history of a set of relevant environment elements. Facts are graphically represented as parallelograms.

Definition 2 (Statement) *a boolean predicate specifying a current or a previous context, whose truth value cannot 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. 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) <==> $f \rightarrow s$*

The relation *help* is strongly subjective, since different stakeholders could define different *help* relations for the same statement, e.g., one stakeholder could say $help(f_1,s) \wedge help(f_2,s)$, whereas another one could say $help(f_2,s) \wedge help(f_3,s)$. Statements are graphically represented as shadowed rectangles, and the relation *help* is graphically represented as a filled-end arrow between a fact and a related statement.

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

Definition 5 (Or-decomposition): *Let* $\{s, s_1,s\ldots,s_n\}$, $n \geq 2$ *be statements (facts).* $Or\_decomposed(<s, s_1,s\ldots,s_n\}) <==>$ *for all* i *member of* $\{1, \ldots n\}$, $s_i \rightarrow s_n$

Decomposition is graphically represented as a set of directed arrows from the sub-statements (sub-facts) to the decomposed statement (fact) and labeled by AND or OR. Let us illustrate the above context analysis constructs by examples:

- "patient is anxious" is a statement since the system can not objectively compute its truth value. This statement can be Or-decomposed into "physiological anxiety" and "moving without target" and "cannot sleep" substatements. The system can get some evidence of the first substatement through the help of the facts "shortness of breath", "sweating abnormally", and "pounding heart". All three of these last facts can be directly measured via sensors.
- "the moving without target" and "cannot sleep" are also statements. The first of these can be verified via observing the movement pattern or the patient, while the second via observing the behavior (i.e., sleeping routine) that the patient would normally display and its change.

Thus, each single situation (e.g., "cannot sleep") may affect the decision on a given variation point of the goal model. The relevant context is described via bollean formula of statements ("trying to sleep" and "not sleeping") which are then expanded to supporting facts (e.g., "laying in bed", "laying on sofa"). These facts can then be directly measured or observed.

Our context analysis is motivated by the need for constructs to analyze context to discover the relevant atomic data that represent that environment, i.e. the data the system has to monitor. The leaf facts of the context analysis help an analyst to elicit the data conceptual model relevant for the analyzed context as shown in Fig. 4 (b). Thus, the facts and statements are reifications over the environment and can be seen as views over the data conceptual model the system has to capture.

This hierarchical context analysis helps to make the context (i) more understandable for the stakeholders, (ii) more modifiable as it is not given as one monolithic block, and (iii) more reusable as parts of the 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 (like the data of Fig. 4 b), as such data is needed to verify leaf facts that in turn confirm or disprove the context needed to make a decision in the goal model.

4. Contextualizing Goals for Product Lines

As noted before, we propose to view goal models and their variability as the origin for product line and its problem-related variability. In this sub-section we outline some ideas for a process for construction, contextualization, and maintenance of such goal-level product lines.

4.1. Construction

When developing our goal-level product line models, we follow the general principles of domain analysis, by considering systems already developed for the same/similar domain and the models build for those. The most commonly used goals and their satisfaction patterns then can be modeled as the core part of system to be, while less frequently used goals and goal satisfaction alternatives can be perceived as optional extras.

Moreover, since our perspective on goal analysis has a strong context modeling counterpart, we can use the derived contextual information for analyzing how appropriate each particular context is for some potential user environment. We can do this simply by answering a set of questions, such as: is this goal and its context going to hold for all environments? When will it (not) be relevant and why? What alternatives contexts will this imply?

For instance, in Fig. 2 we discuss the G3 “Protect against robbery” goal which has two subgoals: G6 “Give illusion of being lived in” and G7 “Act against potential robbery”. Yet, while considering the question if G6 and its context that the patient is not in during the night, will always be desirable, we noted that this is only true for individual/family homes. Other likely venues of use of the Smart Home for patients with dementia are, for instance, retirement homes and hospitals. Neither of these locations needs to have the “Give illusion of being lived in” goal as these venues are always densely populated and largely communal. Thus, though in our initial analysis G6 was considered a core part of the Smart Home, more careful analysis of context and goal stability points to the optionally of this sub-goal. The respective contexts where the goal is desirable (personal/family home) or redundant (publicly owned, or densely populated house/institution) are also identified.

Moreover, more specific analysis of types of variability needs to be carried out. For instance, in our example the goal G1: “Enforce routine exit procedure” has two satisfaction or-decomposed sub-goals: G2 “Patient is alerted” and G3 “Patient is prevented from exiting”. Here we need to analyze if these are mutually exclusive alternative sub-goals, or could they both be present at the same time in a particular system?

As a result of domain analysis and goal/context stability analysis, we can derive a generic goal-level product line model for a particular domain.

4.2. Target Environment Contextualization

When the goal-level product line model (discussed above) is considered for use in a specific environment, the analyst will be able to adopt it for the specificities of the target environment by carrying out a two-step contextualization:

1. *Offline Contextualization*: at target environment analysis stage the analyst will be able to decide that some variant goals/contexts will never be adopted in the intended system operation environment. For instance, if the system is to be deployed in a care home for patients with severe dementia, the alternative that alerts a patient of an anxiety attack is unlikely to be useful, moreover, the light and voice notice may even further aggravate anxiety. Thus, this option may be excluded from the system and further analysis.
2. *Online Contextualization*: when some properties of the environment are changeable, the system

has to monitor and derive a suitable model variant for the current context at run time. For instance, the humidity level in the house is a changing context and it can not be decided at the design/deployment time. The system has to always monitor the humidity and to derive the suitable variant of goal satisfaction at run time. Thus, here the analyst has to identify viable ways of choosing from amongst the options.

In some cases it could also happen that more than one model variants are possible. For instance, regulating humidity for the goal G2 "Refresh air inside home" can be achieved either by T8 of opening windows or by T9 or turning on ventilator, as shown in Fig. 5. However both of these options have some side effects in that T8 affects patients' privacy, while T9 consumes too much of electrical energy. In this case the model selection could be based on user preferences expressed over quality criteria as proposed in [9]. For instance, if the patient asks that he values his privacy very much, and never wants to have the windows open, T8 could always be chosen for him. On the other hand, if environmental concerns are more important to her/him, T9 could be always chosen.

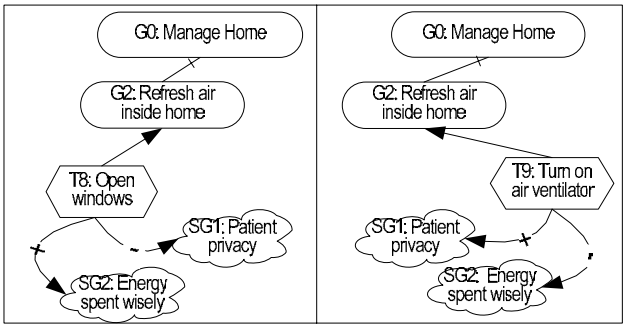

Fig. 5: Two instantiated model variants and their contributions

4.3. Maintenance

When a system is implemented based on the above discussed goal-level product line model, it will be highly advisable to retain a log of system use, contextual property values and user responses. In this way a review of system use and evaluation of goal/context models can be periodically carried out. Some core variants which are never or rarely used can then be moved to the optional set in the domain model, and frequently used optional variants may be re-considered for inclusion into the core. At the same time, the system may be optimized by reviewing contextual property sets, values, goal satisfaction alternatives, etc.

5. Related Work

The research in context modeling, (e.g., [8]), concerns finding modeling constructs to represent software and user context, but there is still a gap between the context model and software variability model, i.e. between context and its use. We tried to reduce such gaps at the goal level and 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. [17] investigate the use of problem descriptions to represent and analyze variability in context-aware software; the work recognizes the link between software requirements and context information as a basic step in designing context aware systems.

Requirements monitoring is about insertion of a code into a running system to gather information, mainly about the computational performance, and reason if the running system is always meeting its design objectives, and reconcile the system behavior to them if a deviation occurs [7]. The objective is to have more robust, maintainable, and self-evolving systems. In [6], a GORE (goal-oriented requirements engineer) framework KAOS [5] was integrated with an event-monitoring system (FLEA [4]) to provide an architecture that enables the runtime automated reconciliation between system goals and system behavior with respect to a priori anticipated or evolving changes of the system environment. Differently, we propose model-driven framework that concerns an earlier stage, i.e. requirements, with the focus on identifying requirements together with context, and eliciting the monitoring data.

Goal analysis (i* [21], Tropos [3], and KAOS [5]), provides a way to analyze high level goals and then to discover and represent alternative sets of the tasks that can be adopted to achieve such goals. Goal models - a mainstream in requirements engineering - are used to represent the rationale of both humans and software systems, and help representing software design alternatives [15]. These characteristics are also important for self-contextualizable software that must allow for alternatives and have a rationale to reflect users and software adaptation to the context in order to adopt one useful execution course [7, 19].

Customizing goal models to fit to user skills and preferences was studied in [9, 13]. The selection between goal satisfaction alternatives is based on one dimension of context, i.e. user skills, related to the atomic goals (executable tasks) of the goal hierarchy, and on user preferences which are expressed over softgoals. Lapouchnian et al. [12] 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 [14], study the variability modeling under the requirements engineering perspective and propose a classification of the intentional variability when Or-decomposing a goal. We focused on context variability, i.e. the unintentional variability, which influences the applicability and appropriateness of each goal satisfaction alternative.

6. Conclusions and Future Work

In this paper, we have shown the influence of context on the variability at the intentional level, adopting Tropos goal modeling as an intentional requirement engineering ontology and integrating it with contexts. By doing so we are capturing the product line variability at the intentional level providing a goal-level product line model. This model contains details of intentions and the context in which such intentions can be satisfied. This facilitates a more deterministic derivation or products with respect to context. Such context-based models can be used to specify dynamic transition between configurations of instantiated goal-based products when a specific context occurs. Such transitions, however, are not covered in the present paper. Instead, we have outlined a process for construction of goal-based product line models, and for their use and maintenance over time.

The main drawback of this approach is the assumption of a “closed world”, i.e., it cannot accommodate any *arbitrary* changes in requirements. Instead, it is well suited for the systems where the dynamic change scenarios are *relatively stable* and can be *expressed in goal models*. In addition, the critical systems require more careful and complete context specification, possibly complemented with some formalized checks.

Yu et al. [22] proposed a method to derive feature models from goal models. Feature models still do not have an explicit notion of the relation between features and contexts. As a future work, we intend to study the relation between feature models and context, as well as instantiation of the goal-level product lines with the supporting context as feature-level product lines. This will allow us to proceed from contextual goal models into contextual feature models. In other words, we will proceed from the intentional variability into functional one.

6. Acknowledgement

This work was partially supported by the European Commission grant IST-215412 - Dynamic Variability in complex, Adaptive systems (DiVA) and by the PRIN program of MIUR under the MEnSA project.

7. References

[1] R. Ali, F. Dalpiaz, and P. Giorgini, "Goal-Based Self-Contextualization", In Proc. of the Forum of the 21st International Conference on Advanced Information Systems (CAiSE'09 - Forum), 2009, pp. 37-42.

[2] R. Ali, F. Dalpiaz, and P. Giorgini, "A Goal Modeling Framework for Self-Contextualizable Software", In the 14th Intl. Conf. on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'09), 2009, Springer, LNBIP 29-0326, pp. 326-338.

[3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology", *Autonomous Agents and Multi-Agent Systems*, Vol. 8, No. 3, pp. 203-236, 2004.

[4] Cohen, M. S. Feather, K. Narayanaswamy, and S. S. Fickas, "Automatic monitoring of software requirements ", Proceedings of the 19th international conference on Software engineering (ICSE '97), 1997, ACM, pp. 602-603.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition", *Selected Papers of the Sixth International Workshop on Software Specification and Design table of contents*, pp. 3-50, 1993.

[6] M. Feather, S. Fickas, A. Van Lamsweerde, and C. Ponsard, "Reconciling System Requirements and Runtime Behavior", *Proceedings of the 9th International Workshop on Software Specification and Design*, pp. 50-59, 1998.

[7] S. Fickas and M. Feather, "Requirements monitoring in dynamic environments", *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pp. 140, 1995.

[8] K. Henricksen and J. Indulska, "A software engineering framework for context-aware pervasive computing", *Proc. Second IEEE Intl. Conference on Pervasive Computing and Communications (PerCom04)*, pp. 77, 2004.

[9] B. Hui, S. Liaskos, and J. Mylopoulos, "Requirements analysis for customizable software goals-skills-preferences framework", *Requirements Engineering*, No., pp. 117-126, 2003.

[10] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "Form: A feature-oriented reuse method with domain-specific reference architectures", *Ann. Softw. Eng.*, Vol. 5, pp. 143-168, 1998.

[11] C. D. Kidd and e. al., "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, Vol. 1670, No., pp. 190-197, 1999.

[12] A. Lapouchnian, Y. Yu, S. Liaskos, and J. Mylopoulos, "Requirements-driven design of autonomic application software", *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, No., 2006.

[13] S. Liaskos, S. McIlraith, and J. Mylopoulos, 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.

[14] S. Liaskos, Y. Yu, E. Yu, and J. Mylopoulos, "On Goal-based Variability Acquisition and Analysis", *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 76-85, 2006.

[15] J. Mylopoulos, L. Chung, and E. Yu, "From object-oriented to goal-oriented requirements analysis", *Commun. ACM*, Vol. 42, No., pp. 31-37, 1999.

[16] K. Pohl, G. Bockle, and F. v. d. Linden, *Software Product Line Engineering: Foundations,Principles, and Techniques.*: Springer, 2005.

[17] Salifu, M., Yu, Y., Nuseibeh, B. Specifying Monitoring and Switching Problems in Context. Proc. 15th Intl. Conference on Requirements Engineering (RE'07), 2007, 211–220.

[18] T. Strang and C. Linnhoff-Popien, "A context modeling survey", *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp*, No., 2004.

[19] D. Sykes, W. Heaven, J. Magee, and J. Kramer, "From Goals to Components: A Combined Approach to Self-Management", International IEEE/ACM Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2008, pp.

[20] E. Yu and J. Mylopoulos, "Why goal-oriented requirements engineering", *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, pp. 15-22, 1998.

[21] E. S. K. Yu, "Modelling strategic relationships for process reengineering", *Ph.D. Thesis, University of Toronto*, No., 1995.

[22] Y. Yu, J. C. S. d. P. Leite, A. Lapouchnian, and J. Mylopoulos, "Configuring features with stakeholder goals", SAC 08: Proceedings of the 2008 ACM symposium on Applied computing, 2008, ACM, pp. 645-649.

[23] A. Finkelstein, and A. Savigni: A framework for requirements engineering for context-aware services. In: Proceedings of the 1st Int. Workshop on From Software Requirements to Architectures (STRAW), 2001