Reasoning with Contextual Requirements: Detecting Inconsistency and Conflicts

Raian Ali[a,b], Fabiano Dalpiaz[b], Paolo Giorgini[b]

[a] *Lero - The Irish Software Engineering Research Centre.*
[b] *University of Trento - DISI, 38100, Povo, Trento, Italy.*

Abstract

CONTEXT. The environment in which the system operates, its context, is variable. The autonomous ability of a software to adapt to context has to be planned since the requirements analysis stage as a strong mutual influence between requirements and context does exist. On the one hand, context is a main factor to decide whether to activate a requirement, the applicable alternatives to meet an activated requirement as well as their qualities. On the other hand, the system actions to reach requirements could cause changes in the context. **OBJECTIVES**. Modelling the relationship between requirements and context is a complex task and developing error-free models is hard to achieve without an automated support. The main objective of this paper is to develop a set of automated analysis mechanisms to support the requirements engineers to detect and analyse modelling errors in contextual requirements models. **METHOD**. We study the analysis of the contextual goal model which is a requirements model that weaves together the variability of both context and requirements. Goal models are used during the early stages of software development and, thus, our analysis detects errors early in the development process. We develop two analysis mechanisms to detect two kinds of modelling errors. The first mechanism concerns the detection of inconsistent specification of contexts in a goal model. The second concerns the detection of conflicting context changes that arise as a consequence of the actions performed by the system to meet different requirements simultaneously. We support our analysis with a CASE tool and provide a systematic process that guides the construction and analysis of contextual goal models. We illustrate and evaluate our framework via a case study on a smart-home system for supporting the life of people having dementia problems. **RESULTS**. The evaluation showed a significant ability of our analysis mechanisms to detect errors which were not notable by requirements engineers. Moreover, the evaluation showed acceptable performance of these mechanisms when processing up to medium-sized contextual goal models. The modelling constructs which we proposed as an input to enable the analysis were found easy to understand and capture. **CONCLUSIONS**. Our developed analysis for the detection of inconsistency and conflicts in contextual goal models is an essential step for the entire system correctness. It avoids us developing unusable and unwanted functionalities and functionalities which lead to conflicts when they operate together. Further research to improve our analysis to scale with large-sized models and to consider other kinds of errors is still needed.

Keywords: Requirements Engineering; Contextual Requirements; Goal Modelling; Consistency and Conflicts Analysis; Adaptive Systems Engineering.

1. Introduction

The recent advances in computing and communication technology led to the emergence of novel computing paradigms which are sensitive to the changes in their environment [1]. Examples include mobile and ubiquitous systems, ambient intelligence, smart-home, autonomic computing, adaptive systems, etc. Such systems weave together computing with humans' living environment in order to facilitate and make more transparent the role of technology in supporting humans' needs. Notably, the environment where the system operates (its context) is highly variable, and this can threaten the purpose of the system if it is not designed to be adaptive to context changes. Adaptation to context is a key feature of this category of systems. Many dimensions of context—spatio-temporal, environmental, social, task and personal—influence the requirements of a system and how it should achieve these requirements [2].

Email addresses: raian.ali@disi.unitn.it (Raian Ali), dalpiaz@disi.unitn.it (Fabiano Dalpiaz), paolo.giorgini@disi.unitn.it (Paolo Giorgini)

Adaptation to context has to be planned since the very early stages of system development; the requirements stage. The reason is that a context affects the decision about the requirements to achieve, the choice between the alternative ways to meet these requirements, and the assessment of the quality of each alternative. Moreover, while acting to meet users' requirements, the system often causes changes in the context. In spite of this mutual influence, context is either ignored or presumed uniform in most requirements engineering literature. It is mostly considered at the later stages of software development (architecture [3], runtime adaptation [4], Human-Computer Interaction [5], services [6]). Requirements adaptation is essential for a comprehensive and correct system adaptation and devising requirements engineering approaches specialized for those systems which responds to the context changes is a challenging research question [7, 8, 9]

Goal models in Requirements Engineering (i^* [10], Tropos [11, 12], and KAOS [13]) represent the stakeholders' desired states of the world (goals) and the possible alternative sets of tasks to reach these states. Software comes as a means to reach users' goals and execute tasks on their behalf. Goal models are capable to represent the rationale of both humans and software systems [14], and they have been shown very useful for the engineering of adaptive systems in particular. The reason is that they incorporate a space of alternatives sets of operations which gives more flexibility to meet stakeholders' goals in a dynamic environment [15, 16, 17]. Goal models fit the early stages of the software development and explain the functionality a system to operate and why to operate it.

Software are motivated by the need to reach certain goals of users and organizations [18, 19]. Context has a strong impact on the goals to activate and the decision about the alternatives set of operations to execute and reach activated goals. For example, in a healthcare institute for people with dementia, a caregiver may have the goal to *"involve the patient in certain social activities"* (G_1) whenever the context *"the patient is feeling bored"* (C_1) holds. The caregiver can reach G_1 by reaching the goal *"take the patient for a trip in the city"* (G_2) or by reaching the goal *"ask a relative of the patient to come and chat to him"* (G_3). G_2 is adoptable if the context *"the city is not crowded"* (C_2) holds, since people with dementia might get anxious in a noisy place. G_3 is adoptable if the context *"the patient has relatives that can come to visit"* (C_3) holds. A requirements model for a smart-home system designed to support the patients with dementia should operate to meet the caregiver's goals G_1, G_2, and G_3. Also, it should correctly represent the relation between goals and context: (i) $G_2 \vee G_3 \rightarrow G_1$ and (ii) if $C_1 \wedge C_2$ then G_2 and (iii) if $C_1 \wedge C_3$ then G_3.

We have proposed contextual goal models in [20, 21] to capture the relation between the variable context and the space of alternatives in a goal model. The notion of context denotes the physical and tangible environment surrounding a system. We refer to an alternative way to achieve a top-level goal as a variant. A contextual goal model may incorporate a large number of interrelated goal model variants and context specifications. This may easily lead to modelling errors in the requirements model which should be detected and resolved early on. In this paper, we develop a set of automated analysis techniques to detect and analyse modelling errors in a contextual goal model. Our objective is to detect inconsistencies of the contexts specified as preconditions for a goal model variant and inconsistencies in the changes on the context caused by the actions (tasks) executed in each variant. We propose an automated tool to support our analysis mechanisms. We present a systematic process for capturing and analysing contextual requirements. Finally, we evaluate our framework via a case study of a smart-home for supporting the daily life of people with dementia.

The paper is structured as follows. In Section 2, we articulate our research problem and contribution and position that within our own state of the art. In Section 3 we describe the contextual goal models proposed in our previous work. In Section 4 we propose analysis mechanisms to detect inconsistencies and conflicts in a contextual goal model. In Section 5 we describe an automated support tool implementing our analysis mechanisms. In Section 6 we outline a systematic process for modelling and analysing contextual goal models. We evaluate our framework in Section 7, discuss related work in Section 8, and present our conclusions in Section 9.

2. Problem Statement

There is a mutual influence between a system and its surrounding environment (context). This is particularly true in certain emerging computing paradigms such as ambient intelligence, ubiquitous and pervasive computing, mobile computing, and self-adaptive systems. Theses systems should be able to continuously monitor their context, detect relevant changes and decide upon what suitable actions to perform. The performed actions could lead to changes in the context as well. Indeed, changing the context could be the purpose of the actions performed by a system or a side-effect of it.

The requirements model is the natural place where the mutual influence between a system and its context is captured and analysed [15, 8, 21]. The context changes can potentially activate a requirement, enable/disable certain system alternatives to reach the activated requirements, and also affect the quality of each of the adoptable alternatives. For example, a smart-home requirement could be r_1= "fresh air inside home". This requirement is satisfiable by applying one of two system alternatives a_1= "open the windows" and a_2= "turn on a ventilator". r_1 is activated in a context like "the humidity level inside home is high". a_1 is adoptable only if the context "it is a good weather outside" holds. Considering "minimum disturbance" as a quality attribute, a_1 could be qualified negatively when it is applied in the context "there is some noise outside" holds. Finally, a_1 leads to changes in the context as the home windows will be opened and the light level might increase as a side-effect of that.

A model for contextual requirements should capture the mutual influence between a system and its context. Such model should be analysed to detect modelling errors early on. The unfixed errors in a requirements models will spread across the next stages in the development process and, thus, lead to a malfunctioning final system. In this paper, we propose an automated analysis to detect modelling errors in a contextual requirements model which is the contextual goal model where requirements are perceived as goals to achieve. Namely, we propose the following two kinds of automated analysis:

- *Inconsistency analysis*. It concerns the detection of inconsistent context specifications in a contextual goal model. The purpose is to identify alternatives (variants) which may be inoperable due to inconsistent context preconditions. For example, suppose a system alternative preconditioned by the context "the patient has a moderate dementia, it is night time and there is a caregiver accompanying the patient at the moment". If the healthcare institute policy assigns the caregivers to patients with severe dementia all the time and to others only during the day hours, then in a smart-home for a patient with moderate dementia this context does not hold. Thus, the system alternative preconditioned by this context is inapplicable due to the fact that its context is inconsistent. Our analysis detects the cases where context inconsistency leads to make certain system alternatives unadoptable. In such cases, the system alternatives should not be implemented or the context specification should be fixed.
- *Conflict analysis*. It concerns the detection of conflicts between certain system actions executed simultaneously to meet requirements. For example, suppose we have two requirements r_1= "fresh air inside home" and r_2= "protect home from robbery". Suppose that one of the system alternatives to reach these two requirements is based on opening the windows to let fresh air enters the home and closing the windows to prevent any potential robber from entering. Thus, when both requirements are active, in a context where the humidity level is high and there is a suspicious movement outside, then we have a conflict. This conflict becomes severe if the system has no other conflict-free alternatives to reach the two requirements. Our analysis detects the set of system alternatives which has conflicting tasks. It further process this set to make sure that at certain context there is at least a conflict-free alternative.

Context is a broad concept and could accommodate all the facets of a system environment. Capturing all these facets is a very challenging and, probably, impractical task. Our modelling and analysis framework captures and analyses one facet of a context which is the physical tangible environment surrounding a system. Examples of this fact include information about temperature, user's age, location, breathing, heart rate, etc. We do not capture those context facets which relates to man-made concepts such as laws, culture and social conventions. The context we capture is prominent and essential in domains that exhibit a clear and direct interaction between the system and its physical environment such as ambient intelligence and smart-homes and buildings. Our framework is tailored to fit the requirements engineering for these domains. Other domains such as contextual search engines and contextual communication would need other contextual requirements models expressly tailored to the nature of context they deal with.

Our previous works creates the baseline of this work. In Table 1, we summarize the contribution of our previous work. The purpose of doing that is to clarify the contribution of this paper and give a holistic view of our approach. In our previous work [9, 20], we have observed that the dynamism of a system environment affects whether a requirement needs to be achieved, restricts the space of adoptable alternatives to achieve it, and affects the quality of each of these alternatives. We have advocated the need to weave together context and requirements. We have proposed to specify context (we called it "location" in that work) as a precondition at cer-

tain variation points of a goal model. We have also proposed basic automated analysis techniques to (i) derive the set of adoptable system's alternatives for a certain context and (ii) determine the context that supports a given alternative. In [22, 23], we have proposed the context analysis model, which includes constructs (statement, fact, support, decomposition) to hierarchically refine a context into a formula of monitorable facts. We also proposed a way to derive the contextual workflow of the tasks of a contextual goal model. In [21], we have proposed an automated analysis to reason about contextual goal models and derive, at runtime, alternatives which best fit both a monitored context and the preferences of a user expressed as ranking on softgoals. In the same work, we also developed a design-time reasoning for deriving a reduced set of systems' alternatives requiring minimal implementation costs.

3. Baseline: Contextual Goal Models

In this section we introduce our baseline: the contextual goal model [20, 21]. We will illustrate this model via a scenario of a smart-home designed to support the daily life of people with dementia. The scenario is a variant of the smart-home system described in [24] and used in the EU sponsored Serenity project[1]. The smart-home supports some of the daily tasks that a person with dementia would forget to do, such as eating on time, circulating the air inside the home, taking medicines, and so on. Another requirement is to facilitate the rescue activities. For example, in case of health emergencies the Medical Emergency Rescue Centre (MERC) should be notified and requested to send a rescue team to the patient place. Besides their memory impediments, patients with dementia could suffer from anxiety attacks. The smart-home should manage such situations by smoothening events to avoid surprises and by preventing the patient from getting out the home when this is risky. The smart-home then has to calm the patient down, and call the caregiver to come and give a treatment. The smart-home supports also some other general tasks, such as preventing potential house robberies. To this end, it can give the illusion that the home is lived in when the patient is out for long time.

Figure 1 shows a partial Tropos goal model for the smart-home of our case study. Tropos goal analysis views the system as a set of interdependent actors where each actor has its own strategic interests (*goals*). Goals are analysed iteratively and in a top-down way, to identify a set of more specific subgoals which can lead to the satisfaction of the upper-level goals. Goals can be ultimately satisfied by means of executable processes (*tasks*). The actor *"Patient Caregiving System"* has the top-level goal *"home is managed for safety of patient"*, which is iteratively decomposed into subgoals by AND-decomposition (all subgoals must be achieved to fulfil the top goal) and OR-decomposition (at least one subgoal must be achieved to fulfil the top goal). The subgoal *"home is protected against robbery"* is AND-decomposed into the subgoals *"give illusion of being lived in"* and *"act against potential robbery"*; the subgoal *"enforce routine exit procedure"* is OR-decomposed into the subgoals *"patient is alerted"* and *"patient is prevented of exiting"*. Goals are finally satisfied by means of executable tasks; the goal *"fresh air inside home"* can be reached by one of the tasks *"open windows"* and *"turn air ventilator on"*.

A dependency between actors indicates that an actor (*depender*) depends on another (*dependee*) to attain a goal or to execute a task. The actor *"Patient Caregiving System"* depends on actor *"Neighbour Assistance System"* for the goal *"a neighbour comes"*. This last goal is an alternative to the goal *"police comes"* and they both are alternatives for achieving a higher level goal which is the goal *"assistance comes to act against robbery"*. Softgoals (*"patient privacy"*) are qualitative objectives for whose satisfaction there is no clear cut criteria and they can be contributed either positively or negatively by goals and tasks. The task *"open windows"* contributes negatively to the softgoal *"patient privacy"*, while the task *"turn on air ventilator"* contributes positively to it.

Let us now explain briefly the goal model shown in Figure 1. The system is required to manage the home and guarantee the safety of the patient. To this aim, our smart-home is designed to achieve three main goals. The first is to ensure that patients do not leave homes unconsciously and is activated in the context where the patient seems to be anxious φ_1 (the list of context descriptions is shown in Table 2). When the level of a patient's dementia is not severe and the patient does not seem to be extremely anxious (φ_3) then the smart home could simply notify the patient about the need to stay at home, otherwise (φ_4) the system has to prevent the patient from getting out the home, to notify the caregiver to come, and to calm down the patient in the meantime. The second goal is to ensure a healthy living environment and we here take the goal "refresh air inside home" as an example of this category of goals. The system can refresh the air inside home by opening the windows if

[1] http://www.serenity-project.org/

Work	Contribution
Weaving context with requirements [9, 20]	- Theoretical foundations for weaving together the variabilities of both the context and requirements - Defining a set of variation points on the goal models where a context condition can be specified - A basic formalization of the model and a set of basic analyses implemented in Datalog
Context analysis [22, 23]	- A systematic way to refine context and elicit its specification - A modelling language for context refinement - A systematic way to derive contextual workflows from a contextual goal model
Automated analysis of contextual goal models [21]	- Derivation of the goal alternatives which fit a context monitored at runtime and a set of user preferences expressed over softgoals - Derivation of a set of systems alternatives which required minimal development costs and able to meet all goals in all analysed contexts - A first version of our CASE tool (RE-Context) to implements these 2 checks - Application of the proposed automated analysis on a case study of a museum-guide mobile information system
This work	- Checking the consistency of context specification at a contextual goal model - Checking the consistency of executable tasks to avoid conflicting actions - A second version of our CASE tool (RE-Context) to implement these 2 new checks - A systematic process that guide requirements engineers to build and use the automated analysis proposed in this paper and in [21] - Application of the automated analysis proposed in this paper on a case study of a smart-home for people with dementia

Table 1: Summary of our previous work and the contribution of this paper

the weather outside allows for that (φ_5), while the option of turning on the ventilator to refresh air is always adoptable. However, each of these two ways of refreshing the air are evaluated differently against certain quality measures (softgoals) which are "patient privacy" and "energy spent wisely". The third is about protecting the home against a potential robbery. The smart-home will turn the light on and off iteratively when the patient is outside and it is night time (φ_7) to give impression that the home is lived in. The home will act against a potential robber who is entering the home area in a suspicious way (φ_8) by locking all the entrances of the home and calling for assistance of the police or neighbours.

3.1. Context and Contextual Goal Models

In goal modelling, actors should be provided by the rationale needed to take the decision on how to reach their goals. This includes the ability to decide what goals to reach, how, and how well to reach them. For example, a caregiving system is an actor that may have the goal of acting against a potential robbery and keep the home protected. The caregiving system has the ability to decide when to activate this goal and what to do to reach it. The caregiving system may activate this goal when there is a person who is trying to enter the home area in a suspicious way. The caregiving system may reach this goal by calling the police or one of the neighbours.The decision between these last two options should be taken by the caregiving system itself. The decision taken by an actor may depend on the state of the world in which the actor is living. We call such a state *context*:

Definition 1 (Context). *A context is a partial state of the world that is relevant to an actor's goals.*

Contexts can be specified at a set of variation points to precondition certain alternatives in a goal model. A context is initially represented by a label (φ_i in the goal model of Figure 1) and described as a sentence (Table 2). We define different variation points on Tropos goal model (for a precise semantic of these variation points please see [21]):

1. **OR-decomposition**: the adoptability of each subgoal (subtask) in an OR-decomposition may require a specific context to hold.
2. **Means-end**: goals can be satisfied by means of specific executable processes (*tasks*). The adoptability of each task may require a specific context to hold as a precondition.
3. **Actors dependency**: a certain context is required for an actor to get a goal reached, or a task executed, by delegating it to another actor.

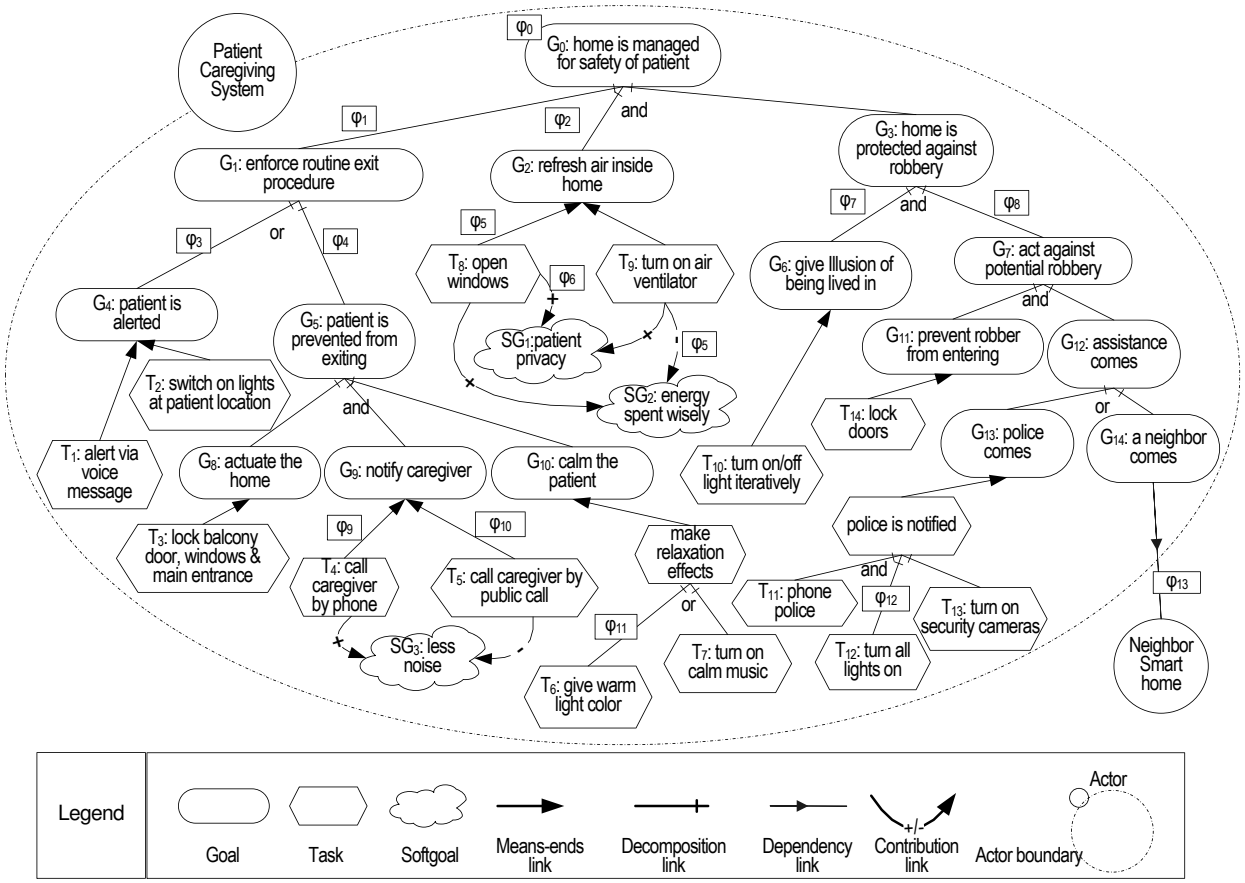


Figure 1: A partial Tropos goal model for the smart-home case study

4. **Root goals**: a root goal becomes activated only in a certain set of contexts.
5. **AND-decomposition**: the satisfaction (execution) of a subgoal (subtask) in an AND-decomposition is needed only in certain contexts. In other words, they are not always mandatory.
6. **Contribution to softgoals**: softgoals are qualitative objectives, i.e., there is no clear-cut criteria for their satisfaction. Softgoals can be contributed either positively or negatively by goals and tasks. The contributions to softgoals can also vary from one context to another.

Similarly to goals, context needs to be analysed and refined. On the one hand, goal analysis provides a systematic way to discover the alternative sets of tasks an actor is capable to adopt and execute to reach a goal. On the other hand, context analysis should provide a systematic way to discover the alternative sets of facts an actor is capable to observe to verify if a context applies. Context, however, is a broad concept and would potentially refer to all of the aspects of a system's environment. In our contextual goal model, we capture context that concerns the tangible and visible aspects of the environment surrounding a system. Examples include the location, the age, the temperature, the movement, etc. We do not treat other facets of context which cannot be transformed to a set of tangible and visible facts which can be verified by the system based on sensed or stored data. We specify context as a formula of world predicates wp; its EBNF syntax is as follows:

$Form \leftarrow wp \mid (Form) \mid Form \wedge Form \mid Form \vee Form$

We classify world predicates, based on their observability by an actor, into two kinds, *facts* and *statements*:

Definition 2 (Fact). *A world predicate F is a fact for an actor A iff F is observable by A.*

Definition 3 (Statement). *A world predicate S is a statement for an actor A iff S is not observable A.*

	Description	Technology
φ_0	The home is lived in, and the patient is expected to have some dementia problems, and there is no awaken caregiver or healthy relative at home.	Database (info about home and patient), RFID tags (caregiver and relative)
φ_1	The patient is at home and anxious.	Smart-shirt or oxymeter, camera with motion recognition
φ_2	The humidity level is high, or home windows and doors haven't been opened for a long time.	Humidity sensor, magnetic sensor (open-close), database
φ_3	The patient dementia disease is not in an advanced stage or he is moderately anxious.	Database (disease status), smart-shirt (anxiety)
φ_4	The patient suffers of advanced dementia, and he seems to be extremely anxious	Database, smart-shirt
φ_5	It is sunny and not very windy.	Barometer and wind sensor
φ_6	The patient is outside home.	GPS or RFID
φ_7	The patient is outside home since a long time and it is night time.	GPS/RFID, database, digital clock
φ_8	A person is trying to get into the yard in a suspicious way (e.g., enter from a place different from the main gate).	Surveillance camera
φ_9	The phone is free and the caregiver is not using his phone for a call.	Information from telephone provider company, phone status sensor
φ_{10}	It is not night time.	Digital clock
φ_{11}	The light level at the patient's location is too low or too high.	Light sensor
φ_{12}	It is too dark inside home.	Light sensor
φ_{13}	The neighbour is healthy, is at home, and can see or reach easily the patient's home.	Database (health status and house location), GPS/RFID (neighbour position)
φ_{14}	The patient health turns bad or he has fallen down.	Smart-shirt, oxymeter, camera with motion recognition
φ_{15}	The MERC is reachable and online.	Check connection
φ_{16}	The home has a screen that shows the patient's medical record.	Database

Table 2: The description of Figure 1 contexts and the technology needed to monitor them

An actor has a clear way to observe a fact. That is, it is capable to monitor the necessary data and compute upon the truth value of a fact. A statement can not be observed by an actor for different reasons, such as (i) the inability to capture needed data to observe it directly and (ii) the abstract nature which makes it hard to find its evaluation criteria. Some decisions that an actor takes may depend on contexts specifiable by means of only facts, while some other decisions may depend on contexts that include also statements. However, a statement can be refined into a formula of facts and other statements. We call the relation between such a formula of word predicates and a refined statement *Support*, and we define it as following:

Definition 4 (Support). *A statement S is supported by a formula of world predicates φ iff φ gives enough evidence to the truth of S.*

Some statements are iteratively refinable to a formula of facts that supports their truth values. In our approach, we only allow for **monitorable contexts**. A context is monitorable if it can be specified in terms of facts and/or statements that are supported by facts. A monitorable context, specified by a world predicate formula φ, applies if all the facts in φ and all the formulae of facts that support the statements in φ are true. In Figure 2, we analyse context φ_1. In the figure, *statements* are represented as shadowed rectangles and *facts* as parallelograms. The relation *support* is represented as curved filled-in arrow. The *and*, *or* logical operators are represented as black triangles, white triangles, respectively.

The analysis of context allows us to discover the data an actor has to collect from the world. The analysis allows us to identify the facts which an actor has to observe. These facts are observable on the basis of certain data of the world to be collected by the actor. For example, taking the facts of the context analysis shown of Figure 2, the analyst could develop a data conceptual model such as the one shown in Figure 3. This model has to be implemented and maintained by the smart-home as a preliminary step to observe facts, judge if the analysed contexts apply, and take decisions at the corresponding variation point in the goal model.

We classify context into three kinds based on the influence it has on goal model. Each context kind is associated to a set of variation points:

1. **Activation context** which makes it necessary to achieve (execute) a set of goals (tasks). In contextual goal models, activation contexts are those specified at the variation points (i) *root goal* and (ii) *AND-decomposition*. These two kinds of contexts determine whether a goal has to be reached or a task has to be executed. The activation context of a goal model variant is the conjunction of the contexts at the variation points of these two kinds. When a context on an AND-decomposition

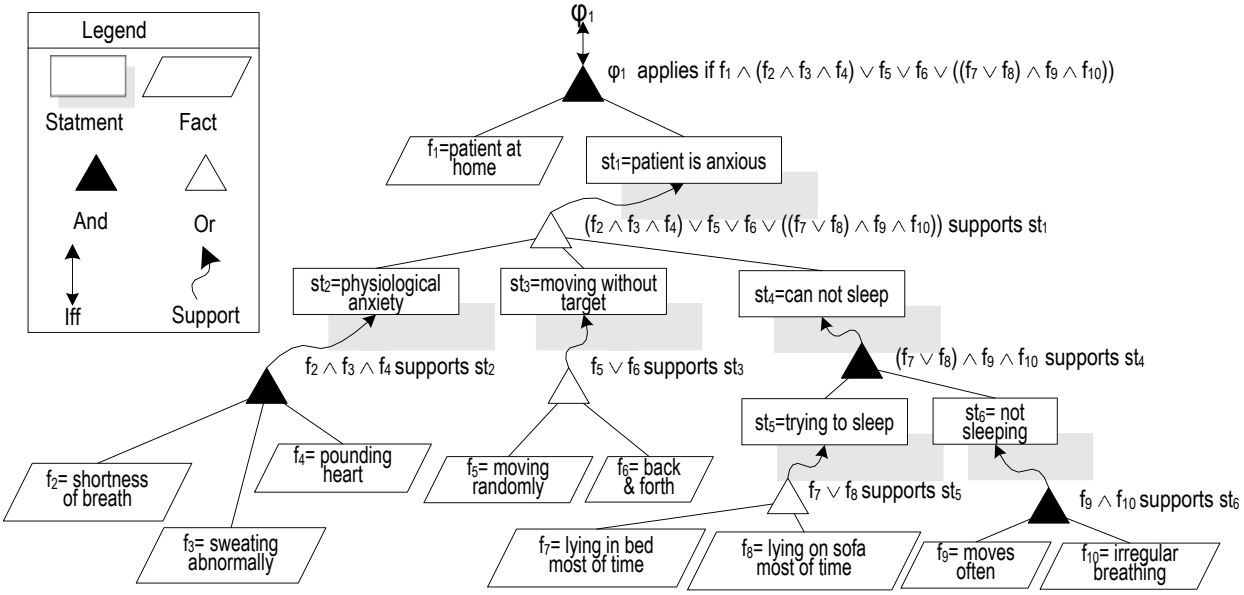


Figure 2: A context analysis for φ_1

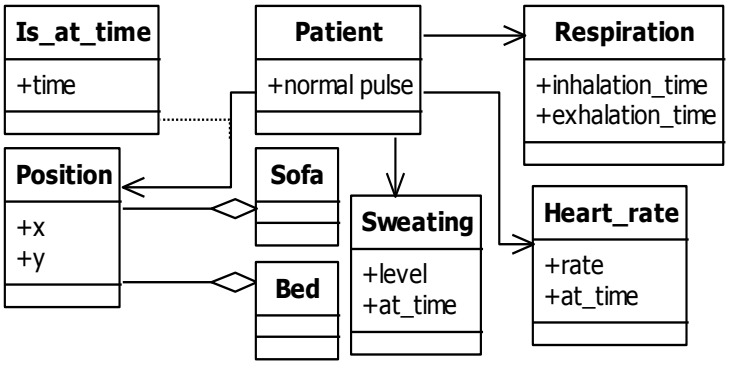


Figure 3: The data needed to observe the facts of context φ_1 shown in Figure 2

holds then the corresponding goal (task) has to be achieved (executed), otherwise that goal (task) is unneeded. While this is syntactically equivalent to a contextual OR-decomposition, the semantic is apparently different. A context on an AND-decomposition influence the need for the reaching/executing the corresponding subgoal/subtask, while a context on an OR-decomposition is itself needed to hold before adopting the corresponding subgoal or the subtask. This semantical difference is essential answer several queries about the active requirements and the adoptable alternatives when a context change occurs at runtime and also for the semantic of context inconsistency which we are going to explain in the next section.

2. **Required context** which is a context necessary to hold before adopting a certain alternative for achieving (executing) a set of activated goals (tasks). The contexts specified on (i) *OR-decomposition*, (ii) *Means-end*, and (iii) *Actors dependency* are required contexts. These contexts are required to hold as a precondition for the applicability of the corresponding goal model variant. The required context of a goal model variant is the conjunction of contexts at the variation points of these three kinds.
3. **Quality context** which influences the quality of a variant of a goal model. Only the contexts at the variation point *Contribution to softgoals* are quality contexts. Contributions to softgoals are, indeed, used in Tropos to capture the impact of a goal/task on a quality attributes (i.e., the softgoal). In contextual goal model, the quality differs according to context changes and it is not static.

Figure 4 shows two partial goal model variants (derived from Figure 1) and the contexts associated to them. Notice that only the subgoals G_1 and G_2 are activated to reach the root goal G_0. However, and since none of the contexts φ_7 and φ_8 is true, there is no need to execute any tasks to reach G_3 which is already satisfied in both of the variants V_1 and V_2. The classification of contexts into these three categories allows us, amongst other things, to answer questions like: "in a given context, which are the requirements the system has to meet?", "which are the possible variants to meet them?", and "what is the quality of each variant?". In the rest of the paper, the term **context of a goal model variant** refers to the conjunction of the activation and required contexts of that variant.

Context is just one of the criteria to consider when the system has to take decision on what alternative to adopt at runtime. The decision can be based on different other criteria such as the potential of success in meeting a root goal, minimizing costs, or satisfying users' preferences. In our previous work [21], we allow users to express their preferences by ranking the importance of softgoals so that the goal model variant with the best contribution to softgoals according to a certain ranking will be applied. The system will monitor context and derive the set of applicable variants and also the value of the contextual contributions to the set of softgoals as a preliminary step to do that.

Our modelling of contextual requirements via contextual goal model is in line with the principles of the four-variable model of Parnas and Madey [25], which is a mathematical model that specifies the function of a software system, i.e. what should be in a software design documentation. They define four variables (input, monitored, controlled, output) which express the relationship between the system and its environment. Then, they specify operations on these variables: IN (from monitored to input), REQ (from controlled to monitored), OUT (from output to controlled), and NAT (from monitored to controlled). REQ represents the requirements of the system, whereas NAT represents the environmental context of the system. In our approach, we cover the "logical world" part by expressing the REQ function via goal modelling, and the NAT function via context modelling.

It is important to emphasize that our contextual goal model and context analysis capture only a certain facet of a system context. This facet concerns the tangible and visible facts in a system environment. These facts can be verified based on data the system is able to capture from its environment. However, we emphasize that there are other facets of the system environment which are unnecessarily transformable or representable via tangible facts. Examples of such facets include the laws and the regulations of the system organization and the social relationships between the different system actors, culture and conventions, etc. Our modelling framework addresses just one facet of context, the tangible one, and thus the analysis framework which we propose in the next section detects conflicts and inconsistencies related to this facet only. This fits well to application areas where the system and its surrounding physical environment affect each other such as ambient intelligence, smart-home, ubiquitous computing, etc.

4. Analysing Contextual Goal Models

In this section, we develop two kinds of analysis mechanisms about contextual goal models. The purpose

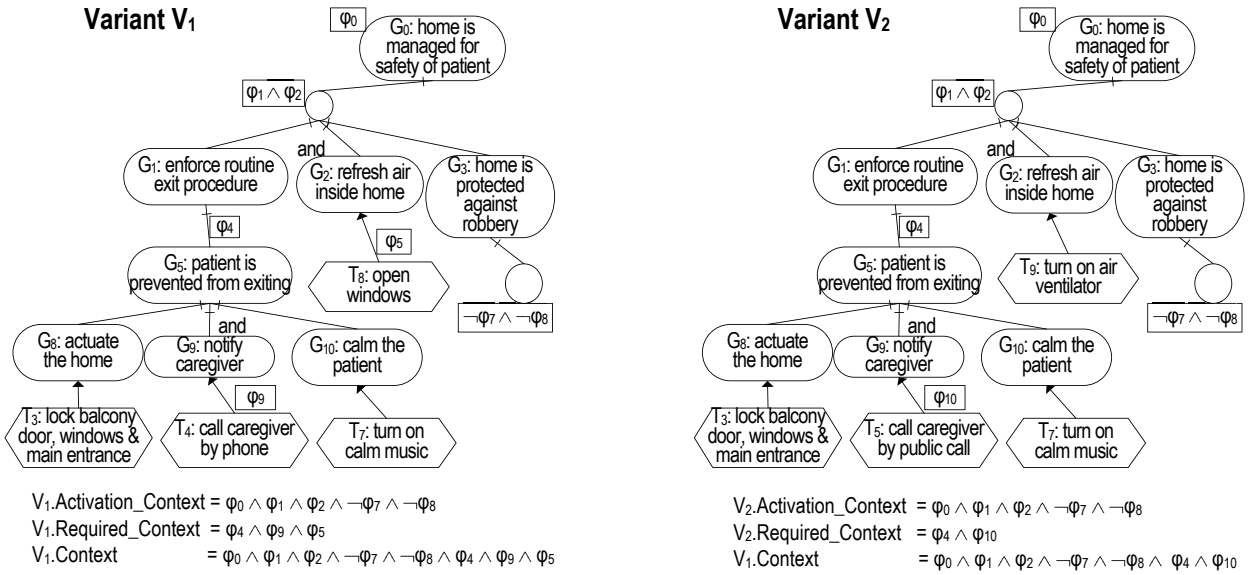


Figure 4: Two goal model variants and their contexts

is to detect modelling errors early in the development process and support the construction of a well specified requirements model and thus a correctly implemented system. The first kind is the *consistency analysis* which determines whether the formulae expressing the contexts of the goal model variants are consistent. In other words, it checks if the precondition for the adoptability of a certain variant is a state of the world that might ever occur. For example, an inconsistent context is one including "patient is at home" and "patient is visiting a neighbour". Our second proposed analysis is the *conflict analysis* which identifies conflicts between the tasks (executable processes) of each goal model variant, in terms of the way these tasks change the context. For instance, a conflict exists when two concurrent tasks need to exclusively use a resource or modify a resource in different ways (e.g. task T_i opens a window to refresh the air, while T_j closes all the windows to ensure silence when a person is sleeping).

4.1. Consistency analysis

Context analysis allows us to refine contexts at the variation points of a goal model and discover formulae of facts which specify these contexts (see Figure 2). In our framework, we only allow the specification of monitorable contexts, i.e., the contexts which can be refined to formulae of facts. However, when deciding if a goal model variant is applicable, a conjunction of these formulae, which express the accumulative context of that variant, has to be verified. A formula expressing a context (or a conjunction of contexts) could be inconsistent. Inconsistencies could be due to certain modelling errors which should be fixed to get a correct requirements model.

To check the consistency of a formula specifying a context, we need to take into consideration all possible contradictions among the variables (world predicates) of that formula. For example, in Figure 1 we have $\varphi_7 = wp_{7.1} \wedge wp_{7.2}$ where $wp_{7.1}$ =*"patient is outside home for long time"* and $wp_{7.2}$=*"it is night time"*, and $\varphi_{10} = wp_{10.1}$ where $wp_{10.1}$ = *"it is not night time"*. In this example, $\varphi_7 \rightarrow \neg\varphi_{10}$ because $wp_{7.1} \rightarrow \neg wp_{10.1}$, so any goal model variant whose context includes $\varphi_7 \wedge \varphi_{10}$ will be inapplicable. The logical relations between world predicates formulae (contexts) can be *absolute* or *dependent* on the characteristics of the system operational environment:

1. *Absolute* relations hold in any operational environment. For example, given the three world predicates wp_1= *"caregiver [c] has never worked in another institute"*, wp_2= *"patient [p] is in the institute for the first day"* and wp_3= *"caregiver [c] was assigned to patient [p] some date before today"*, then $wp_1 \rightarrow \neg(wp_2 \wedge wp_3)$ holds in any institute the system operates in.
2. *Operational environment dependent* relations hold in a specific environment where the system operates in. Consider the two world predicates wp_1 =

"the temperature is less than 15 degrees at the patient's location" and wp_2 ="*patient is at home*". If in one institute, the heating system keeps temperature above 20 degrees, then $wp_1 \rightarrow \neg wp_2$ holds always in that institute. Moreover, the operational environment itself may assure that some world predicates are always true or false. Thus, we have to consider a special kind of environment dependent relations: $Env \rightarrow wp\_formula$. For example, if the system operates in an institute hosting only patients with severe dementia, then the implication $Env \rightarrow \neg wp_3$ where wp_3 ="*patient has basic dementia*" always holds.

We apply SAT-based techniques [26] to check if a formula, expressing a context, is consistent under a set of assumptions. Given a formula and a set of assumed logical relations between its variables, a SAT-solver checks if there exists a truth assignment for all variables that makes the conjunction of the formula and the logical relations formula satisfiable. The context specified by a formula is consistent iff such assignment exists. In this paper, we presume that these relations are to be provided collaboratively by the designers and the domain experts. We still do not provide an automated support to facilitate this task. A possible automated support could infer new relations from the already defined ones and also avoid the designer defining useless relations which do not lead to discovering new inconsistencies even if defined. The design of such an automated tool is one of our future work directions. However, in Section 6, we explain how an iterative consistency check could minimize the amount of relations required from the designers and the domain experts. The pseudo-code of the algorithm (*CheckSAT*) is reported in Figure 5.

Input: context φ
Output: $\perp$ ($\top$) if φ is inconsistent/consistent
1: ξ := *get_logical_relations*(ξ)
2: **if** Is_Satisfiable($\varphi \wedge \xi$) **then**
3: **return** $\top$
4: **else**
5: **return** $\perp$
6: **end if**

Figure 5: Checking context consistency (CheckSAT)

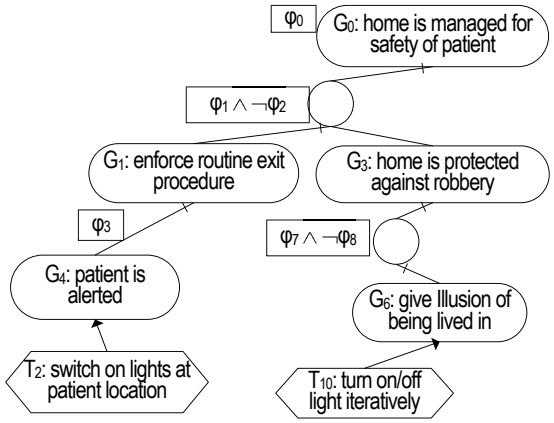
The context associated to each variation point has to be consistent, otherwise that context could never become true. Inconsistency of a single context is caused by a modelling error that should be fixed. The accumulative contexts (activation, required, ...) for goal model variants could also be inconsistent. However, the inconsistency of these accumulative contexts does not always indicate a modelling error and fixing or accepting (tolerating) such an inconsistency is an analyst's decision. Since goal models represent variability in a compact form (a same subtree belongs to multiple variants), inconsistencies related to a certain variant need not necessarily be corrected as the variant itself may not be indeed applicable. Moreover, the semantic of context inconsistency depends on the kind of accumulative context in which it occurs. In what follows, we illustrate the above ideas via examples taken from the contextual goal model of Figure 1.

Case 1 (Unneeded variant). The inconsistency of an activation context of a goal model variant means that the variant is not needed. In other words, it means that the requirements represented in that variant are never activated.

Example 1. *The variant in Figure 6 has an inconsistent activation context because of the contradiction between* φ_1 *and* φ_7*. These contexts are defined as follows:* $\varphi_1 = wp_{1.1} \wedge wp_{1.2}$*, where* $wp_{1.1}$="*patient is inside home*", $wp_{1.2}$="*patient feels anxious*", *and* $\varphi_7 = wp_{7.1} \wedge wp_{7.2}$*, where* $wp_{7.1}$= *"patient is outside the home area for long time" and* $wp_{7.2}$= *"it is night time". Though this variant is inapplicable, the context inconsistency is acceptable. Indeed, giving illusion of being lived in to protect home from robbery is needed when patient is outside, whereas treating his anxiety is needed when he is at home. Since these requirements are not active together, designers could accept the inconsistency. In other cases, the inconsistency of activation contexts has to be fixed. Suppose that* φ_0 *is replaced by* φ'_0 *that adds the fact "patient is at home". Therefore,* $\varphi'_0 \wedge \varphi_7$ *is inconsistent and* G_8="*give illusion of being lived in*" *will never be activated. In such case, the designers would decide to fix the inconsistency.*

Case 2 (Unadoptable variant). The inconsistency of the required context of a goal model variant having a consistent activation context implies that such variant can be activated but there is no context that makes it adoptable. In other words, a set of requirements could be activated but a certain variant to meet them is not adoptable.

Example 2. *Figure 7 shows an example of inconsistent required context. In this example, the administration of the health care institute decides that calling the caregiver through institute speakers requires to be in a context where the patient has extreme anxiety. In all other cases, the caregiver can be called by phone. Therefore,* φ_9 *is modified into* φ'_9 *that adds the fact "patient anxiety is moderate". This makes* $\varphi'_9 \wedge \varphi_4$ *inconsistent. In this*



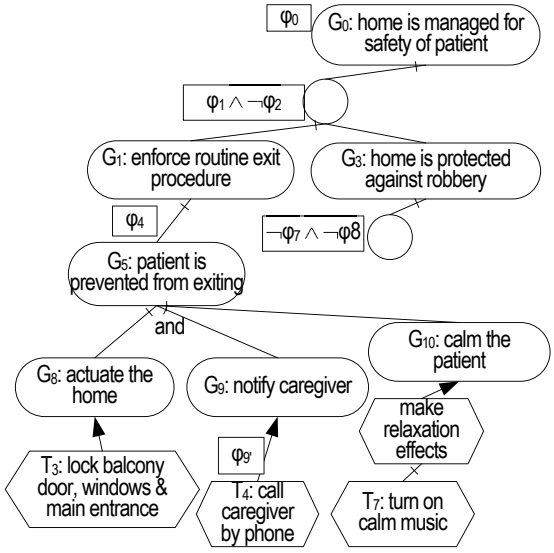
V.Activation_Context = $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \varphi_7 \wedge \neg\varphi_8$
V.Required_Context = φ_3
V.Context = $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \varphi_7 \wedge \neg\varphi_8 \wedge \varphi_3$

φ_1 = ($wp_{1.1}$:patient is inside home) $\wedge$ ($wp_{1.2}$: patient is anxious)
φ_7 = ($wp_{7.1}$:patient is outside home for long time) $\wedge$ ($wp_{7.2}$: it is night)

The contradictions between contexts: $wp_{1.1} \rightarrow \neg wp_{7.1}$

V.Activation_Context is *inconsistent*

Figure 6: A partial goal model variant with an inconsistent activation context



V.Activation_Context = $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_7 \wedge \neg\varphi_8$
V.Required_Context = $\varphi_4 \wedge \varphi_{9'}$
V.Context = $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_7 \wedge \neg\varphi_8 \wedge \varphi_4 \wedge \varphi_{9'}$

φ_4 = ($wp_{4.1}$: severe dementia) $\wedge$ ($wp_{9.3}$: extreme anxiety)
$\varphi_{9'}$ = ($wp_{9.1}$: the home phone is free) $\wedge$
($wp_{9.2}$: caregiver phone is not being used) $\wedge$ ($wp_{9.3}$: moderate anxiety)

The contradictions between contexts: $wp_{9.3} \rightarrow \neg wp_{4.2}$

V.Activation_Context is *consistent*
V.Required_Context is *inconsistent*

Figure 7: A partial goal model variant with an inconsistent required context

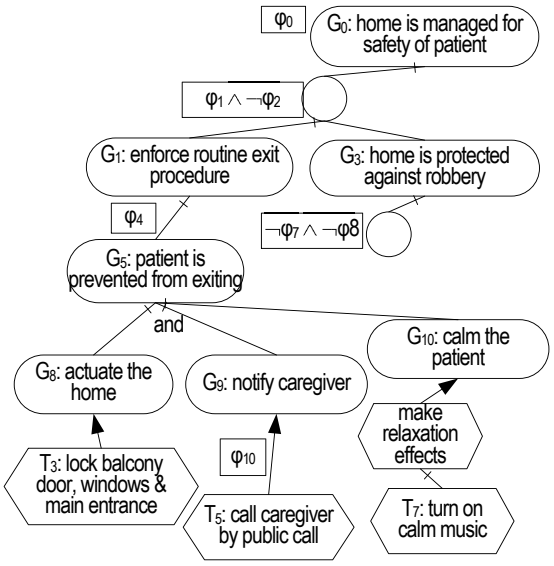
new specification, the context required for calling caregiver by phone never holds and designers would decide to fix the inconsistency.

Case 3 (Incompatible contexts). The inconsistency of the context of a goal model variant, when its activation and required contexts are consistent separately, means that the variant could be activated and adopted in different contexts, but cannot be adopted in any context where it can be activated.

Example 3. *Figure 8 shows a goal model variant with inconsistent context of this kind. In this example, the institute assigns a caregiver to each patient except for night time. This creates a contradiction between φ_0 and φ_{10} and makes the context of variant V_3 inconsistent. If T_5 does not appear in other variants with a consistent context, designers might decide to exclude it from the implemented system.*

Case 4 (Static contribution). The inconsistency in quality contexts occurs when the conjunction of a context of one contribution to a softgoal and a context of a goal model variant in which this contribution exists is inconsistent.

Example 4. *Suppose that administration of an institute decides that calling caregivers through the institute*



V.Activation_Context= $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_7 \wedge \neg\varphi_8$
V.Required_Context = $\varphi_4 \wedge \varphi_{10}$
V.Context = $\varphi_0 \wedge \varphi_1 \wedge \neg\varphi_2 \wedge \neg\varphi_7 \wedge \neg\varphi_8 \wedge \varphi_4 \wedge \varphi_{10}$

φ_0 = ($wp_{0.1}$: home is lived in) $\wedge$ ($wp_{0.2}$: patient has dementia) $\wedge$ ($wp_{0.3}$: no awaken relative) $\wedge$ ($wp_{0.4}$: no caregiver)
φ_{10}=($wp_{10.1}$: it is not night time)

In certain institute, caregivers are assigned to patients except for night time. This implies the following contradiction: $wp_{10.1} \rightarrow \neg wp_{0.4}$

V.Activation_Context is *consistent*
V.Required_Context is *consistent*
V.Context is *inconsistent*

Figure 8: A partial goal model variant with an inconsistent context

speakers has a negative impact on the softgoal "less noise" at the night hours (while the impact is ignorable at the day hours). The negative contribution from T_5 to SG_3 will be preconditioned by the context φ_x="it is night time". Since T_5 requires day hours time then $\varphi_{10} \rightarrow \neg\varphi_x$ and, therefore, there will be no contribution between T_5 to SG_3 and the designers could just remove this contribution from the model.

4.2. Conflict analysis

Adaptability to context indicates that a system has a high degree of autonomy and flexibility to chose how to achieve users' goals in a variety of contexts. However, the system itself might lead to different changes over the context as a consequence of the tasks it executes to meet users' goals. These changes could originate conflicts preventing the correct achievement of user's goals. Understanding conflicts is preliminary for their resolution and requires to answer questions like: (i) Why does a conflict occur?, i.e. what are the conflicting tasks and the goals behind them?; (ii) What is the context in which a conflict occurs?; (iii) Is there any alternative to avoid the conflict?; (iv) What are the core conflicts that the system, at certain context, can not avoid?, i.e. which conflicts are severe?

Most conflicts arise due to contradicting actions on *objects* in the environment wherein the system operates [27]. In this paper, we focus on two kinds of conflicts:

- **Conflicting changes**: two or more system executable processes (tasks in a goal model) try simultaneously to change the same object in the system environment into different states. For example, the task T_8: *"open windows to circulate air"* and the task T_3: *"lock balcony door, windows, and main entrance to prevent patient of getting out"* aim to change an object, that is the windows, into two contradictory states, *"closed"* and *"open"* respectively. If these two tasks execute in parallel, a conflicting change occurs.

- **Exclusive possession**: two or more executable processes need an exclusive possession of an object in the environment. For example, both tasks T_{11}: *"phone police"* and T_4: *"call caregiver by phone"* need an exclusive possession of the landline phone in the patient's home. If these two tasks execute in parallel, an exclusive possession conflict occurs.

4.2.1. Detecting conflicts

To identify the two kinds of conflicts that we have listed, we need to enrich contextual goal models with additional information:

- The effect of tasks execution on the system operational environment: similarly to the contradictions between contexts, tasks can be also contradicted in terms of inconsistent changes they cause on the

system environment. To infer such relations between tasks automatically, we require the designer to specify explicitly the effect of tasks execution on the objects in the system environment. This will avoid the designer the time-consuming activity of defining these relations directly over the space of tasks of a goal model. For each object with which the system interacts, the designer needs to define if the execution of a task changes the state of that object or requires exclusive possession on it. In Table 3, we show the influence of some of tasks in Figure 1 on the patient's home objects.

Object	**States**
External doors (balcony, main entrance)	open, closed, locked
Windows (living room, bedroom)	open, closed, locked
Lights (living room, bedroom, balcony)	on, off, soft
Siren, security camera, ventilator	on, off

Task	**Object**	**State**	**Exclusive**
T_1	Home speakers		true
T_5	Institute speakers		true
T_2	Lights	on	
T_3	External doors Windows	locked locked	
T_4, T_{11}	Landline		true
T_8	Windows	open	
T_5	Institute network		false

Table 3: Objects in the patient's home, and the tasks impact on them

- The sequence/parallelism operators between tasks: we need to specify if two tasks, in each goal model variant, execute in parallel or in sequence. Specifying this information for each pair of tasks is a time-consuming and error-prone activity. For this reason, we adopt the goal model extension proposed in [28], where business process operators are introduced to represent the process carried out to reach goals. Out of these operators, we use the parallelism (||) and sequencing (;) operators. This way, we can infer if two tasks may execute simultaneously. In Figure 9, we annotate the smart-home contextual goal model, shown in Figure 1, with these two operators.

The algorithm reported in Figure 10 processes a contextual goal model and enriches its variants with information concerning adoptability and conflicts. For each variant (line 1), the algorithm checks the consistency of its context (line 2), and excludes inconsistent variants from further processing as they are unadoptable (line 3). Each variant with consistent context is checked for conflicts between its tasks (lines 5-14). The set of tasks in each variant is extracted (line 6) and partitioned based on the parallel execution (line 10). Each partition of tasks is checked to know if it includes tasks changing an object in the system environment into different states (line 11) or to exclusively posses it (line 12). Each variant is enriched with information about conflicts happening between its tasks (line 13). This algorithm detects not only the conflicts between tasks, but also identifies the goals behind the tasks originating the conflicts and the context in which such conflicts happen.

We emphasize here that conflicts do not necessarily happen in every instance of the conflicting goals. That is, when two goals are activated simultaneously and the variants to achieve each of them contain conflicting tasks, this does not necessarily mean that a conflict must occur . For example, T_4= "call caregiver by phone" and T_{11}= "phone the police" could take place at different moments where both goals G_1= "enforce routine exit procedure" and G_7= "act against potential robbery" are active and the tasks make part of the variants to achieve them.

We also emphasize the role of activation contexts in deciding what goals to achieve and thus what conflict might occur. For example G_0= "home is managed for safety of patient" is a goal to maintain. To keep this goal maintained, certain goals might need to be achieved and other goals might need to be maintained as well. The decision is based on the values of the activation context of those goals. For example, to maintain G_0, we may need to achieve G_1 only when the context φ_0 and φ_1 are true and the contexts φ_2 and φ_7 and φ_8 are false. In this case, there will be no possibilities of having conflicts between T_4 and T_{11}. When none of the contexts φ_1 and φ_2 and φ_7 and φ_8 is true, then G_0 does not require the system to execute any task to maintain it satisfied. When the contexts φ_0 and φ_1 and φ_2 are all true then there could be a conflict between T_3 and T_8 in the case that these two tasks are selected as part of achieving G_1 and G_2. That is to say, the truth values of contexts decide whether to activate certain goals and tasks and thus some variants. A conflict potentially occurs if a variant containing two conflicting tasks is activated and adopted.

4.2.2. Detecting core conflicts

Conflicts in one goal model variant can be resolved by adopting another variant that is conflict-free and applicable in all the contexts where the conflicting one is applicable. In some cases, there could be no such conflict-free variant and a resolution has to be provided by designers. In this section, we develop a reasoning framework to discover when a conflict belongs to this kind, i.e. when it is *core*. We first give some basic definitions and then devise an algorithm that processes a

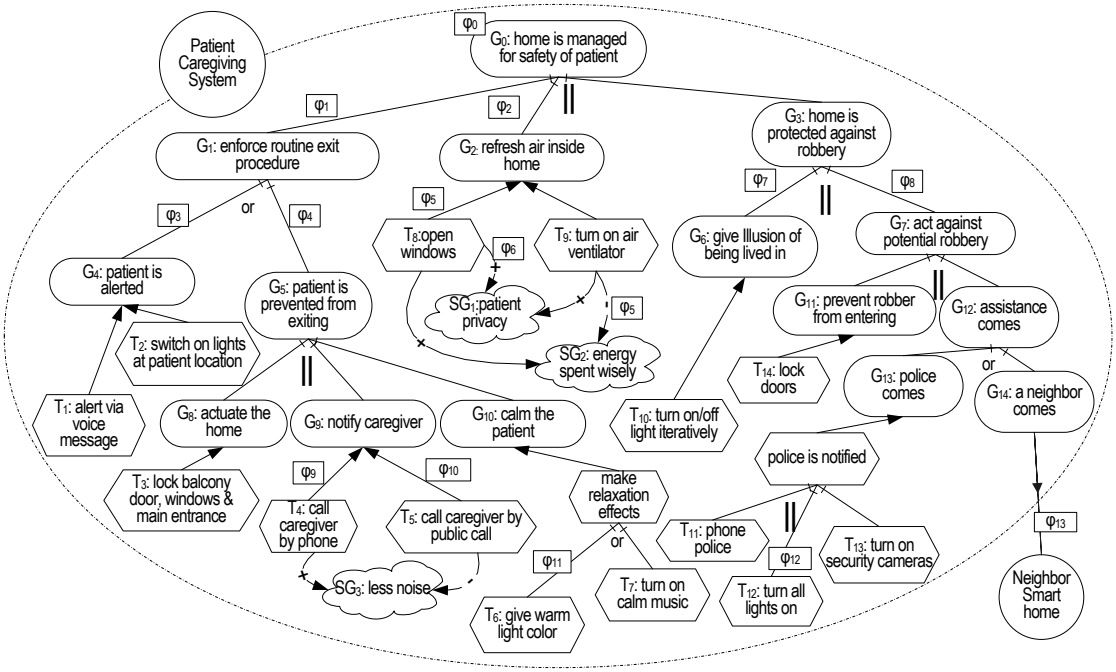


Figure 9: Goal model annotated with parallelism (||) and sequence (;) operators

Input: *S: the set of all goal model variants*; *E: the set of environment objects*
Output: *S enriched by adoptability and conflicts information*

1: **for all** $V \in S$ **do**
2: **if** $CheckSAT(V.context) = \bot$ **then**
3: $V.adoptability := \bot$
4: **else**
5: $V.adoptability := \top$
6: $T := V.tasks$
7: $V.conflicts := \emptyset$
8: **while** $|T| > 1$ **do**
9: $t_i := pop\_element(T)$
10: $T_{t_i\|} := \{t_j : t_j \in T \wedge parallel(t_i, t_j)\}$
11: $T_{t_i\|conflicting\_changes} := \{\langle t_i, t_j, o, t_i.o.state, t_j.o.state\rangle : t_j \in T_{t_i\|} \wedge\ o \in E \wedge t_i.o.state \neq t_j.o.state\}$
12: $T_{t_i\|exclusive\_possession} := \{\langle t_i, t_j, o, \text{“}exclusive\text{”}\rangle : t_j \in T_{t_i\|} \wedge\ o \in E \wedge t_i.o.exclusive \wedge t_j.o.exclusive\}$
13: $V.conflicts := V.conflicts \cup T_{t_i\|conflicting\_changes} \cup T_{t_i\|exclusive\_possession}$
14: **end while**
15: **end if**
16: **end for**
17: **return** S

Figure 10: Detecting conflicts in contextual goal models

contextual goal model to detect core conflicts.

Definition 5 (Core variant). *A variant V_i with a context specified by a formula φ_i is core iff φ_i is consistent and $\nexists$ variant V_j with a context specified by a consistent formula φ_j: $(\varphi_i \rightarrow \varphi_j) \wedge \neg(\varphi_j \rightarrow \varphi_i)$.*

This definition says that any variant that is non-core has a set of core variants applicable in all contexts where it is applicable, but not vice-versa. However, non-core variants are not to be discarded: in certain contexts they might assure better quality than core ones. The core variants are grouped, on the basis of the equivalence of

their contexts, in core groups of variants.

Definition 6 (Core groups set). *A core groups set is a set of core variants partitioned on the basis of context equivalence.*

Definition 7 (Core group of variants). *A core group of variants is an element of a core groups set.*

Example 5. *In Figure 11, we show two partial goal model variants $\{V_1, V_2\}$. These two variants are two alternatives to satisfy goal G_5, each in a specific context. The contexts of these two model variants are consistent and $V_1.context \rightarrow V_2.context \wedge \neg(V_2.context \rightarrow V_1.context)$. This means that V_1 is non-core since there is variant V_2 that can replace V_1 in all the contexts where V_1 is applicable.*

Having a conflict-free variant in a core group of variants means that any conflict in the other variants in the same group is non-core. If all the variants in a core group of variants have conflicts, then we face a core conflict and a resolution has to be crucially provided for one, at least, of the variants in that group.

Definition 8 (Conflicting core group of variants). *A conflicting core group of variants is a core group of variants that does not include any conflict-free variant.*

The algorithm reported in Figure 12 returns the core groups of variants in conflict from a contextual goal model. It calls the algorithm shown in Figure 10 to enrich each variant with information about adoptability and conflicts (line 1). The algorithm excludes unadoptable variants, i.e., variants with inconsistent contexts, as they are obviously non-core (line 2). Then, the core groups of variants are identified (lines 4-11). To this end, the algorithm partitions the set of variants based on context equivalence (line 6). The algorithm CheckSAT, shown in Figure 5, can be used to check the equivalence between boolean formulae expressing contexts. Given the logical relations (implications) (ξ) between the variables of two formulae φ_1 and φ_2 then $\varphi_1 \rightarrow \varphi_2$ iff $\neg(\varphi_1 \rightarrow \varphi_2)$ is inconsistent under the assumptions ξ. Then the algorithm checks if each group is core (line 8) and keeps it for further processing if it is like that (line 9). The algorithm then checks each core group of variants to decide if it contains at least one conflict-free variant. If this occurred, then the group is not conflicting and it is excluded from the output set (line 12–16).

Example 6. *As shown in Figure 13, the assumption is that the subgoals of the root goal "home is managed for patient safety" are not dependent on each other and may need to be reached in parallel when their corresponding contexts hold (notice the notation $\|$). The variant V_1 includes a conflict between tasks T_3 and T_8 for the environment object "windows". Each task changes the state of this object differently, as specified in Table 3. Variant V_2 can replace V_1 in all of its contexts since $V_1.context \rightarrow V_2.context$, which means that V_1 and its conflict are non-core. An example of a core conflict is that occurring in V_3 because of the exclusive use of the environment object "phone" between the two tasks T_4 and T_{11} and the absence of variants that are adoptable whenever V_3 is adoptable and that are conflict free.*

5. Automated Support Tool

To support the analysis mechanisms proposed in Section 4, we have developed a prototype automated tool called *RE-Context*. This tool takes as input a contextual goal model expressed as an input file for DLV[2], a disjunctive Datalog [29] implementation. The tool has been developed to demonstrate the usefulness of our reasoning techniques when applied in practice. Currently, we do not provide a graphical goal modelling editor and automated translation to the DLV input format. We also remind the reader that the first version of the tool was proposed in our previous work [21]. That versions implemented to two analyses proposed in that work as we already mentioned in Section 2.

Figure 14 (a) shows a sample contextual goal model; part (b) represents its translation to the DLV input format. The top-level goal G_1 is AND-decomposed to G_2, G_3 and G_4, and the decomposition to G_2 is subject to the context φ_1. The mapping is shown in the first four lines of code: whenever G_1 has to be achieved (the predicate *ach* is used for non-leaf-level goal achievement): (i) either G_2 should be achieved or context φ_1 should not hold; (ii) if G_2 should be achieved, then φ_1 should hold; (iii) if G_1 should be achieved, then the leaf-level goals G_3 and G_4 have to be done (the predicate *todo* is used for leaf-level goals achievement). The OR-decomposition from G_2 to G_5 and G_6 is shown in lines 5-7. If G_2 has to be achieved, then either G_5 or G_6 should be done. If G_5 is chosen, then context φ_2 should hold; if G_6 is chosen, then context φ_3 should hold. The last line states that goal G_1 is an active requirement.

The first task of our tool is to derive all variants, and what RE-context does is to run the DLV reasoner using

[2]http://www.dbai.tuwien.ac.at/research/project/dlv/

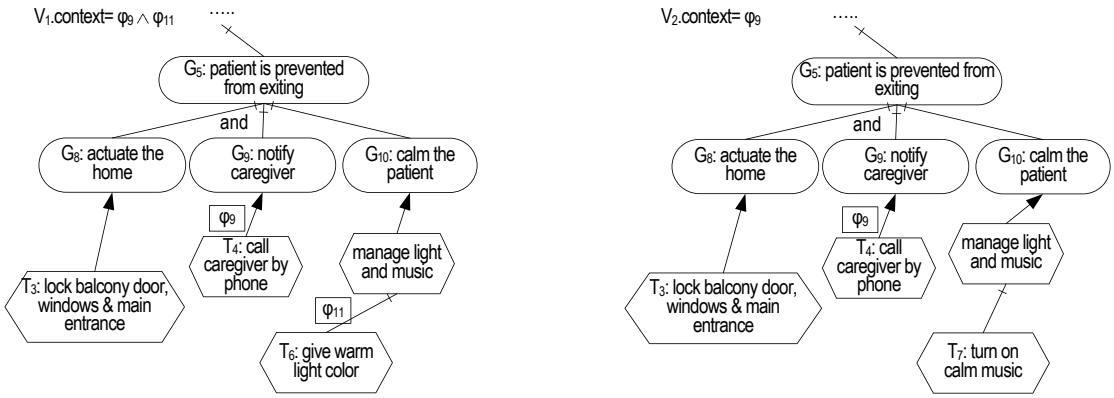


Figure 11: An example of a non-core variant: V_1 is non-core as $V_1.Context \rightarrow V_2.Context$

Input: S: *all goal model variants set*
Output: S'':*the set of all core groups of variants with conflict*

1: $S' := Detect\_Conflict(S)$
2: $S' := S' \setminus \{V \in S : V.adoptability = \perp\}$
3: $S'' := \emptyset$
4: **while** $|S'| > 0$ **do**
5: $V := pop\_element(S')$
6: $temp := \{V\} \cup \{V' \in S' : CheckSAT(\neg(V.context \leftrightarrow V'.context)) = \perp\}$
{i.e. Check if V.context ↔ V'.context}
7: $S' := S' \setminus temp$
8: **if** $\nexists V' \in S' : V.context \rightarrow V'.context$ **then**
9: $S'' := S'' \cup \{temp\}$
10: **end if**
11: **end while**
12: **for all** $U \in S''$ **do**
13: **if** $\exists V \in U : V.conflicts = \emptyset$ **then**
14: $S'' := S'' \setminus U$
15: **end if**
16: **end for**
17: **return** S''

Figure 12: Extracting the conflicting core groups of variants

it as a planner on the goal model: the output consists of all the valid models that satisfy the rules in the input file, i.e. the goal model variants. In particular, we are interested in the set of tasks to be carried out (the *todo* predicates) and the contexts that should (not) hold (the *phi* predicates).

The next step is to check context consistency for each variant, which corresponds to run the *CheckSAT* algorithm described in Figure 5. To verify the consistency of a context, RE-Context uses an external tool (MathSAT[3]) that is based on MiniSat SAT solver[4]. In order to carry out this step, RE-Context loads the definition of contexts from a separate file (contexts.msat), which contains the representation of all contexts as boolean formulas expressed over a set of variables. The relations between contexts/variables are defined in another MathSAT input file called relations.msat. In order to check an inconsistency, RE-Context takes the contexts that refer to the analysed variants from contexts.msat and merges these formulas with the relations between contexts, then runs the SAT solver to determine the formulae consistency.

The inconsistency of individual contexts at variation points is, obviously, a modelling error to be fixed and not subject to design decisions. RE-Context checks each individual context for inconsistency and alerts the

[3]http://mathsat4.disi.unitn.it
[4]http://minisat.se/

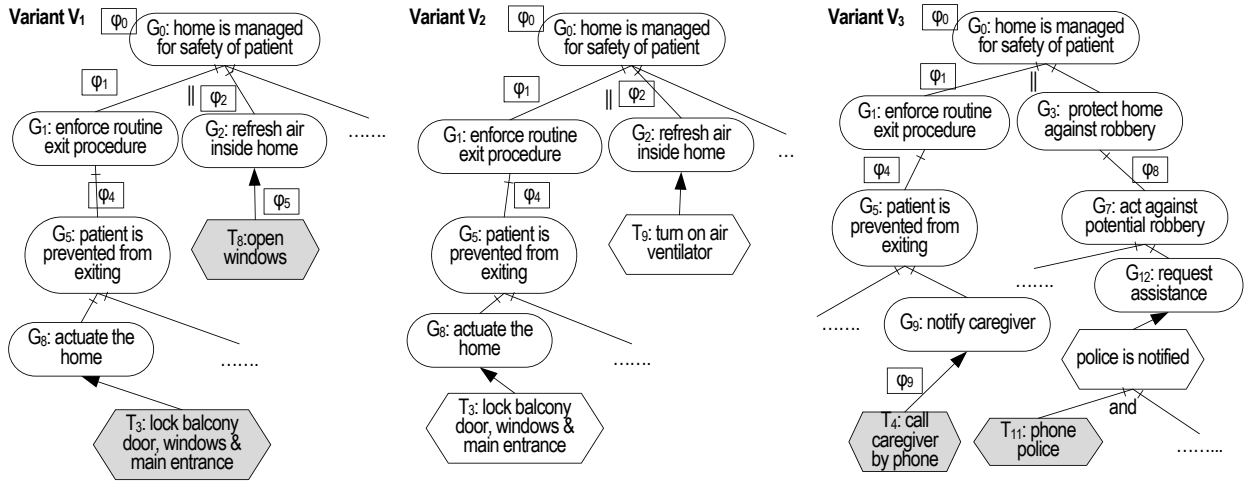
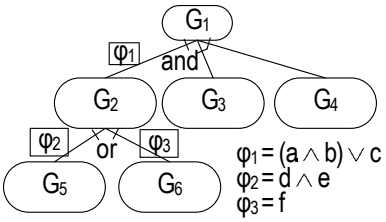


Figure 13: Non-core variant with conflict (V_1), its conflict-free alternative (V_2), and variant with core conflict (V_3)



(a)

```
1 ach(g2) v -phi(1) :- ach(g1).
2 phi(1) :- ach(g2).
3 todo(g3) :- ach(g1).
4 todo(g4) :- ach(g1).
5 todo(g5) v todo(g6) :- ach(g2).
6 phi(2) :- todo(g5).
7 phi(3) :- todo(g6).
8 ach(g1).
```

(b)

Figure 14: Sample goal model (a) and its representation as an DLV input file

analyst when it is inconsistent. As explained in Section 4.1, fixing or accepting inconsistencies of the accumulative contexts of goal model variants is a design decision. To minimize interaction with the analysts, this activity propagates a design decision for one variant to the others when possible. If an inconsistency in one accumulative context is discovered, RE-Context asks the analyst to fix or accept it showing the variant, the context, and the contradictions. If the inconsistency is accepted, RE-Context scans the rest of the variants and marks the inconsistency of those containing the processed one (i.e. containment of tasks) as accepted. In order to further minimize the interaction with analysts, variants with less tasks are examined first.

The third phase consists of identifying conflicts in variants. In order to achieve this result, we need to express information concerning resources. Code 1 shows how we express that a task changes a resource (lines 1-3) and that a task requires a resource (lines 4-6).

Line 1 declares $r1$ as a resource; line 2-3 say that task G_6 (G_4) changes G_1 to *value*1 (*value*2). Line 4

Code 1 Expressing the link between tasks and resources

```
1 resource(r1).
2 changes(g6,r1,value1).
3 changes(g4,r1,value2).
4 exclusiveUsage(r1).
5 requires(g4,r1).
6 requires(g3,r1).
```

states that $r1$ requires exclusive usage; lines 5-6 say that task G_4 (G_3) requires resource $r1$. In our example, a variant containing both G_6 and G_4 would imply a conflicting change, whereas a variant with both G_3 and G_4 would entail an exclusive usage conflict. Moreover, we also need to express information about the sequentiality of tasks and goals: conflicts concerning resources exist only in case of tasks executing in parallel. In our formalization, we make usage of the `parallel` and `sequence` predicates in the DLV input file. The sequentiality predicates are then propagated top-down in the goal trees, in order to identify which tasks should be executed in sequence and which have to be executed in parallel. If the code in Code 1 included the predicate

`sequence(g4,g3)`, there would not be the exclusive usage conflict for resource $r1$. The last step our tool currently supports is the discovery of core conflicts. In order to carry out this analysis, we implemented in RE-Context the algorithm of Figure 12.

6. A Systematic Modelling and Analysis Process

In this section, we propose a systematic process for constructing and analysing a contextual goal model. The overall picture is depicted in the activity diagram in Figure 15. Four macro-activities are identified: goal analysis, context analysis, reasoning about contextual goal models, and identifying monitoring requirements. We emphasize here that the activities "Context consistency" and "Conflict detection" concerns the automated analysis and the extension to RE-Context tool proposed in this paper. The activities of "Minimal-cost analysis" and "Contextual goal model validation" concern the automated analysis and the first version of RE-Context proposed in [21]. The other activities are modelling activities meant as guidelines for the analyst on how to model contextual goal models and data to capture and monitoring infrastructure to deploy.

1. **Goal analysis**: the high level goals need to be elicited and analysed. Goals can be iteratively identified through scenarios [30, 31]. Moreover, an intentional variability taxonomy [32] can guide variability acquisition when refining a goal/task to discover alternative ways of reaching/executing it. Each refinement step is followed by a context analysis.
2. **Context analysis**: it weaves goal modelling with context, so that to link the requirements, at the goal level, to the context in which they are activated and adoptable. Context analysis is composed of three sub-activities:
   (a) *Contextual variation points identification*: for each variation point in the goal model, a decision has to be taken on whether context plays a role in the selection of variants at that point. In other words, the analyst has to decide if a variation point is contextual or not. When a contextual variation point is identified, a high level description of the correspondent context has to be made. As a result of this activity, the contextual variation points at the goal model are annotated as shown in Figure 1 and the contexts associated with them are described as shown in Table 2
   (b) *Context refinement*: the contexts at each contextual variation point are analysed. The analysis aims at identifying how contexts can be verified. In other words, it is to define the facts of the environment the system has to capture and the way these facts are composed to judge if an analysed context holds (as shown in Figure 2). Moreover, the analyst has to deal with different views of different stakeholders about the analysed contexts. Stakeholders may define context differently, and even in contradictory ways. In case of inconsistency between stakeholders on context refinement, the analyst has to reconcile their viewpoints through a consensus session.
   (c) *Specifying logical relations between contexts*: after the refinement of each context, the logical relations (implications and contradictions) between it and the previously refined contexts need to be specified. These relations are essential for the forthcoming reasoning about contextual goal models. In some cases, defining these relations at the level of context analysis is possible, i.e. defining that the context C_1 at the variation point VP_1 is contradicted with C_2 at VP_2, as we could do in our case study. For larger contexts, we may need to specify these relations at a more fine-grained level (e.g. between facts).
3. **Reasoning about contextual goal models**: this activity is supported by our automated reasoning tool RE-Context. The tool enables different reasoning on contextual goal models. It analyzes a contextual goal model in order to detect inconsistencies in contexts specified on it and potential conflicts among its executable processes (tasks). Moreover, it is possible to check whether the model reflects stakeholders' requirements. To this end, this reasoning derives and shows to stakeholders the goal model variants that reflect a given context and user priorities. Three kinds of reasoning can be performed:
   (a) *Reasoning about context consistency*: this reasoning checks if a context can eventually hold. First, it has to be done for the contexts defined at each variation point. If a context of this kind is inconsistent, the analyst must either fix the inconsistency or remove the context and mark the variation point as context-independent. The inconsistency of accumulative contexts of goal model variants

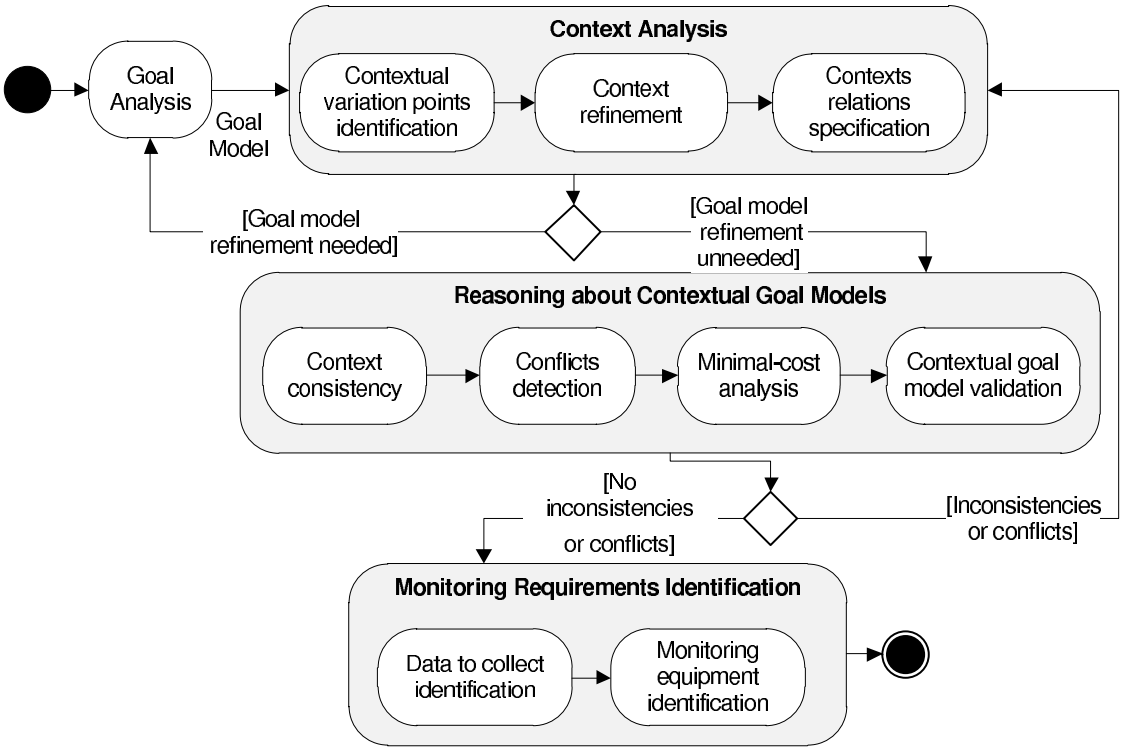


Figure 15: The modelling and analysis process for contextual goal modelling

are subject to design decisions as explained in Section 4.1. When an inconsistency in this kind of context is discovered, the tool asks the analyst to decide whether to fix the inconsistency or accept it and therefore exclude the correspondent goal model variant. When an inconsistency in one goal model variant is accepted, the tool (RE-Context) excludes the rest of variants that include the one being examined and marks their inconsistency as accepted as well.

(b) *Reasoning about conflicts*: to enable the automated discovery of conflicts in a contextual goal model, the analyst has to enrich the model with further information. This information includes: (i) the objects in the system environment and the impact of the execution of goal model tasks on them, and (ii) the sequence/parallel operators between goals/tasks in AND-decompositions. Adding this information, RE-Context is able to detect conflicts and classify them into two categories: core and non-core. The analyst needs to resolve core conflicts crucially as this kind of conflicts leads to situations where there is no way to meet some requirements.

(c) *Minimal-cost analysis*: a goal model incorporates a space of alternative behaviours (alternative sets of tasks) to reach a goal. The implementation of all these tasks may be impractical due to the time and costs required for that. In [21], we proposed an automated analysis that elects the set of tasks which requires the minimum-cost set of resources to implement and which enables at least one way to reach all goals in all the analysed contexts. This activity can use our RE-Context tool to derive this set of tasks. The reasoning is implemented in the version of RE-Context proposed in [21].

(d) *Validating contextual goal models*: this reasoning is to ensure that the contextual goal model reflects stakeholders' expectation of the system in different contexts and in compliance with their priorities. To this end, the analyst can ask stakeholders to specify a context and then show them the correspondent goal model variants. Alternatively, the analyst may ask stakeholder to specify a variant for a given context and compare it with

the ones obtained by our automated analysis. This test might be done for the whole goal model or parts of it. Moreover, user prioritization might be considered to select between goal model variants when more than one is adoptable in a given context. User prioritization can be specified over softgoals as proposed in [33, 34, 21]. The version of RE-Context which we proposed in [21] already supports the activity of deriving goal model variants for a given context and prioritization.

4. **Identifying monitoring requirements**: after context analysis and reasoning, the analyst can identify the monitoring requirements. Monitoring requirements are fundamental to develop systems adaptive to context. We identify these requirements in terms of the data to collect from the system environment and the equipments needed to collect them.
   (a) *Identifying the data to collect*: by analysing the facts obtained by context analysis, the analyst can identify the data needed to verify them as shown in Fig 3. We suggest to keep track of the relation between each facts and the fragment of the data conceptual model needed to verify it. This link is important to promote re-usability and modifiability of the contextual goal model. In case a part of the context refinement is reused/modified, we will be able to identify which fragments of the data model are to be reused/modified. Although we have used class diagrams to represent the data conceptual model, more specific models can be exploited. Several models have been proposed to represent the data to monitor in context-aware systems (e.g. [35, 36, 37]).
   (b) *Identifying the monitoring equipments*: for systems operating in and reflecting varying context it is important to specify the equipments to install and to use in order to enable data collection. This activity defines the equipments needed to capture the data identified in the previous activity. For example, the analyst needs to specify the kinds of sensors to use, the topology of sensor distribution, the interval of data sensing, and so on. To achieve such specification, expertise in new technology is needed. In Table 2, we have given a brief description of the equipments needed for each of the contexts at the goal model. This specification becomes more accurate after knowing the data to monitor in the environment, i.e, after the previous activity.

RE-Context supports the activity "reasoning about contextual goal model". This activity requires the analyst to provide manually some input, namely the relation between contexts and the influence of tasks on the environment objects. RE-Context tool can be used iteratively in order to minimize the involvement of the analysts in specifying the contextual goal model and allow the automated reasoning. We here list two guidelines of such a usage:

- *Upon refinement*. After each refinement of the goal model, i.e., a step down in the hierarchy, the analyst should define the new context relations and the influences on the environment objects if applicable. After doing so, the RE-Context can be run to detect if a conflict and/or inconsistency are in the current version of the model. This will help us fixing modelling error at an earlier step of the modelling and enables us to minimize the size of input the engineer has to provide which does not lead to different results. If the partial goal model is inconsistent or having conflicts then the complete one will still have that. The iterative checking after refinement enables us to fix the problems where and once they emerge and thus minimize the analyst's effort and time.

- *Upon relation specification*. After each specification of a relation between contexts or the influence of tasks on the environment objects, the designer can run the RE-Context tool. That is because, a newly specified relation or influence can lead to remove a variant of the goal model or marking it as conflicting. Thus, defining new relations and influences in that variant will not change the decision about it and the designer will be relieved of defining useless new relations.

7. Evaluation

We evaluate our analysis framework by applying it on a case study concerning a smart-home for supporting the life of patients with dementia. We want to assess whether our new analysis mechanisms support consistency and conflicts checking of contextual requirements expressed via contextual goal models. To this end, we address the following evaluation questions:

Q1. *Are our automated analysis techniques able to identify inconsistencies and conflicts which are not notable by the requirements engineers?*

This question is concerned with the usefulness and efficacy of our automated analysis, implemented in RE-Context, as a designer-support tool, i.e., if it is capable to identify non-trivial inconsistencies and conflicts in a contextual goal model.

Q2. *Is the modelling extension required by our analysis framework understandable by the requirements engineers and does it require them much time to capture?*

We want to get feedback from requirements engineers about the modelling extension to contextual goal models which we proposed in this paper. This extension consists of the relations between contexts, the influence of goal model tasks on the environment objects, the specification of sequence and parallel operators on goal model. We are also interested in the overhead—in terms of time—this extension puts on the engineers.

Q3. *Does our automated analysis scale well in terms of reasoning time?*

We need to investigate how our automated analysis scales when processing contextual goal models of different sizes. This affects how practical our analysis is in terms of time complexity.

7.1. Study settings

In order to answer our evaluation questions, we involved five requirements engineering researchers in the modelling of contextual goal models and in the interview we did afterwards. Figure 16 illustrates the workflow of our case study. The *subjects* are academics working at the Department of Information Engineering and Computer Science of University of Trento: three are junior researchers (Ph.D. students), two are senior researchers (one post-doc and one professor). We chose subjects having no prior expertise with our proposed analysis framework. All the subjects have research expertise in requirements engineering and are familiar with goal modelling.

Concerning the *object* of the study, we focussed on a scenario taken from a research project the subjects were already involved in. The project involves a direct interaction with industrial partners with whom the subjects interacted heavily. This means that all subjects have comparable level of expertise in the topic. The scenario is about a smart-home which is autonomously able to support the life of old people who suffer from dementia problems. It is taken from the EU-funded project Serenity[5], a consortium of both academic and industrial partners.

[5] http://www.serenity-project.org/

After choosing and inviting the subjects to participate in the case study, the process proceeded as follows. First, a domain expert with a practical experience in the health-care system has explained in details the scenario to all of the participants and answered their questions. Moreover, a video demo has been shown to the participants to describe an example of the desired behaviour of the smart home. This phase took us one hour. Second, we gave a one-hour presentation about our modelling and analysis framework. We briefly explained Tropos goal modelling as the subjects are already familiar with (10 minutes), then we explained the variation points on goal model where context can be specified (15 minutes), then we explained conflicts and inconsistencies in contextual goal model and gave examples about that (20 minutes) and, in the rest of the hour, we explained what modelling extension is needed to enable the automated detection of inconsistencies and conflicts. After that, we asked the subjects to practice our modelling framework and answered their questions which emerged while practising it. This lasted for an hour and half.

After a break, the participants were asked to capture the smart-home requirements using contextual goal models and define the input needed for the automated analysis. We asked the participants to ensure that the final model does not have harmful inconsistencies and conflicts. The reason is that we need to see if our analysis framework is helpful in detecting cases which were not recognized by the requirements engineers. This task took the participants three hours to accomplish. After that, we have formalized the model which was delivered to us by the participants and applied our analysis framework to detect and deal with inconsistencies and conflicts. The formalization took us 3 hours,

The day after, we showed the participants the results obtained by applying our automated analysis and got their confirmation that the detected cases are indeed harmful inconsistencies and conflicts. We have conducted a semi-structured interview in which we asked the subjects to answer our questions in a written way and then we had an individual meeting with each of them to clarify the answers. The goal was to gain their feedback about the extension to contextual goal models which is required to accomplish our automated analysis implemented in the RE-Context tool. We asked the subjects to write their answers after they tried to model the requirements of smart home using contextual goal model. The face-to-face interview with each participant took around 25 minutes. The interview form consisted of the following questions:

1. How would you evaluate the ease-of-

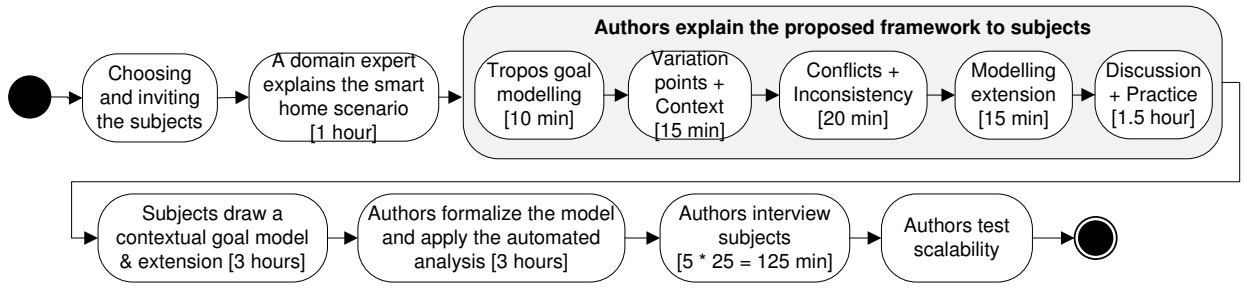


Figure 16: The evaluation workflow

understanding of our extension to the contextual goal model? [1: very difficult, . . ., 5: very easy]

2. How difficult is it to specify the effects of goal model tasks on the systems environment objects? [1: very difficult, . . ., 5: very easy]
3. How difficult is it to specify the sequence and parallel operators? [1: very difficult, . . ., 5: very easy]
4. How difficult is it to specify the relationships between contexts? [1: very difficult, . . ., 5: very easy]
5. Do you have any concerns and suggestions to improve our framework?

Finally, we conducted a scalability analysis. We took chunks of the goal model provided by the participants and artificially constructed goal models of different sizes by cloning these chunks (following an approach similar to that proposed in [38]).

7.2. Study results

In the following we try to answer the three questions we raised in the beginning of this section. To answer Q1, we discuss the results of applying our automated analysis, implemented in RE-Context tool, on the model developed by the subjects. To answer Q2, we discuss the results of interviewing the subjects. Finally, and to answer Q3, we present and discuss our scalability test where we applied our automated analysis on goal models of different sizes.

7.2.1. Answering Q1

Q1 stands for the usefulness of our proposed automated analysis in detecting conflicts and inconsistencies which are not notable by requirements engineers. Table 4 summarizes the results concerning the contextual goal model constructed by the participants and the results of applying our automated analysis on it. The goal model size is described in terms of the number of actors (A), goals (G), tasks (T), softgoals (SG), variation points (VP), and variants (V). The next columns relates to the results of applying the automated analysis. First, we report the results of inconsistency analysis: how many times we have been asked to fix or accept an inconsistency (Iterations), how many times we needed to fix inconsistency and considered it unacceptable (Fixed), and how many variants with acceptably inconsistent contexts were left without fixing and removed from the model (VwAccInc). After inconsistency checking, the tool processes the variants with consistent contexts to detect conflicts. We present the number of detected conflicts (C), the number of variants with conflicts (VwC), the number of non core variants which always have replacement variants (NonCV), the number of core groups of variants (CGV), and the number of conflicting core groups of variants(CCGV).

As the table indicates, our proposed automated analysis detected considerable amount of inconsistencies and conflicts which have not been recognized by the participants who developed the model. The number of inconsistencies which were detected was 27. This corresponds to the number of iterations where the automated analysis asked us to decide whether to accept or fix an inconsistency. Out of this, we needed to fix 3 cases which indicated harmful inconsistency. Moreover, the automated analysis detected 29 conflicts between tasks. A conflict emerges when two tasks executes in parallel and each requires an exclusive possession of an object in the system environment or changing it to different states. These conflicts caused 184 core groups of variant to be conflicting out of the total number core group of variants which is 192. In these conflicting core groups of variants, the system is unable to resolve the conflicts by switching to a conflict free variant. These results, considering the fact that we asked the participants to ensure that the model is free of harmful inconsistencies and conflicts and the fact that they confirmed that the

Goal Model Size						Context Inconsistency			Conflicts				
A	**G**	**T**	**SG**	**VP**	**V**	**Iterations**	**Fix**ed	**VwAccInc**	**C**	**VwC**	**NonCV**	**CGV**	**CCGV**
5	35	50	5	25	25560	27	3	11556	29	13789	1908	192	184

Legend:

A: nr. actors
G: nr. goals
T: nr. tasks
SG: nr. softgoals
VP: nr. variation points
V: nr. variants
Iterations: nr. iterations to fix/accept context inconsistency
Fixed: nr. times the inconsistency was fixed
VwAccInc: nr. variants with accepted inconsistency
NonCV: nr. non-core variants
C: nr. conflicts
VwC: nr. variants with conflicts
CGV: nr. core groups of variants
CCGV: nr. conflicting core groups of variants

Table 4: The results of applying our automated analysis on the contextual goal model of smart-home scenario

detected cases are true during the interview, answer our question Q1. That is, our automated analysis discovers harmful inconsistencies and conflicts which are not recognizable by the requirements engineers who develops a contextual goal model.

7.2.2. Answering Q2

Now we discuss the interviews which we made with the participants to answer our question Q2. Concerning the easiness to understand, all the participants agreed that the modelling extension is easy to understand and quite intuitive (three subjects gave 4 and two gave 5). Concerning the easiness to specify and the time required for that, the participants found that specifying the influence of tasks on the environment objects is easy (four subjects gave 4 and one subject gave 5) and that specifying the sequence and parallel parameters is also straightforward and does not require much time (three subjects gave 5 and two gave 4). However, they all agreed that specifying of the relations between contexts is not an easy task and is a time consuming one as it requires a high number of comparisons between contexts and their combinations (three subjects gave 2, two subjects gave 1). Therefore, two of the modelling extensions were found easy to understand and also easy to capture and one extension was found easy to understand but hard and time-consuming to capture. Indeed, these answers were not surprising and we anticipated that based on the observation we made while the subjects were doing the modelling activity.

Here we summarize the main concerns and concerns suggestions raised by the participants when answering the third question of the interview:

- The specification of the relations between contexts shall be supported by an automated tool. It is time-consuming and could be incomplete (5 subjects).
- For complex contexts, there could be a disagreement (viewpoints) regarding the way to refine context. This, consequently, affects the definition of the relations between contexts and, thus, limits the discovery of inconsistency and conflict cases. We still need to develop a systematic management of such disagreement via techniques like voting or domain expert consultation (3 subjects).
- The usage of temporal relations between contexts is important as this would help to discover more inconsistencies and conflicts (2 subjects).
- A richer ontology of the effect of tasks on the environment objects would help for a broader discovery of conflicts. For example, the effect of a task on an object may be indirect as it can result from its effect on another object (2 subjects).

7.2.3. Answering Q3

To answer Q3, which concerns the scalability of our proposed automated analysis, implemented in our RE-Context tool, we executed it on a varying sizes of goal model. We ran our tool RE-Context on a machine with two CPUs AMD Athlon(tm) 64 X2 Dual Core Processor 5000+ and 4 GB of RAM. Figure 17 reports the results of the performance analysis with respect to the time needed (in milliseconds) to perform reasoning. The first two columns represent the size of the goal model as number of nodes (goals and tasks) and number of variants; then, the table reports the time needed to derive all variants (T_Deriv), to identify inconsistency (T_Inc), to get the core groups of variants (T_CGV). The time to compute the core groups of variant with conflicts is negligible in comparison to T_CGV. RE-Context scales well for medium-size goal models, e.g. it took us 30 minutes to process the goal model developed by the participants

Size of goal model		T_Derive	T_Inc	T_CGV
NN	NV			
18	**3**	62	3	5
30	**12**	79	18	10
42	**108**	273	53	288
49	**540**	582	195	3826
64	**2565**	1224	1351	23076
79	**4275**	2484	2009	59221
90	**15300**	7553	3926	100339
90	**25560**	10424	12006	1819126
150	**104976**	21861	63868	2348941

Legend

NN: the number of nodes in the processed model.
NV: the number of variants in the processed model.
T_Derive: time to derive all variants (in ms).
T_Inc: time to get all variants with inconsistent context.
T_CGV: time to get the core groups of variants.

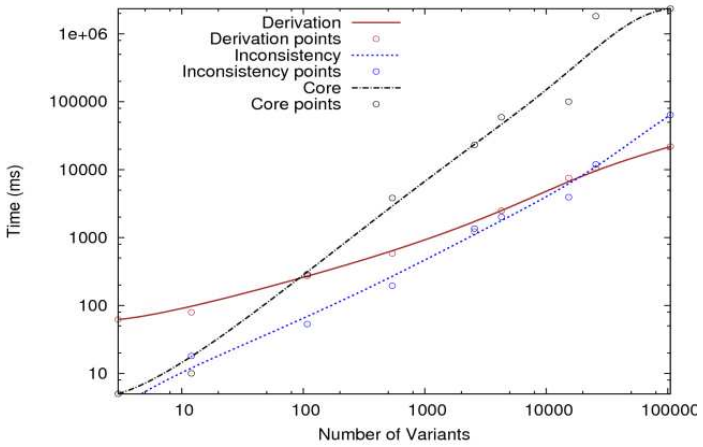


Figure 17: Scalability of our automated analysis implemented in RE-Context tool: tabular and graphical representations

in our study which contained 90 nodes and 25,560 variants.

To obtain goal models of different sizes, we adopted an approach similar to the one used in [38]: we cloned the original goal model that was delivered by the participants in our study. RE-Context needed 40 minutes for a goal model of 100,000 variants. As shown in Figure 17, the derivation of goal model variants and the inconsistency check scale quite well, whereas the identification of core groups of variants has scalability limitations for large-scale goal models. The number of nodes is not a critical factor for scalability, whereas the number of variants and the relations between contexts are critical. Since our framework applies at design-time, we can conclude that the tool is adequate till medium-sized goal models. However, we still need to optimize our algorithm to make reasoning with larger goal models faster. Thus, as an answer to Q3, our scalability testing indicates that our automated analysis scales well for small- and medium-sized goal models, such as the one developed or our smart-home scenario, while it needs further optimization to deal with goal models of large size.

To improve scalability, we might benefit from two techniques. The first is the iterative check of the model during construction. We can analyse consistency and conflicts while constructing the goal model instead of treating the entire final goal model at once. This way, problems are identified as soon as they arise and can fixed immediately. The second is by using *divide-and-conquer* techniques. Computing the core groups of variants could be complex due to the high number of invoking SAT solver. A way to reduce this complexity, is by dividing the model into parts, reasoning about each part separately, and then combining the results. For example, for an AND-decomposed goal, we can compute the core groups of variants of each subgoal and then combine the results by Cartesian product.

7.3. Threats to validity

We discuss the main threats to validity related to our evaluation. As suggested by relevant literature in empirical software engineering [39, 40, 41], we distinguish between internal, external, reliability and construct threats. We could identify threats related of the first three kinds and we summarize that in Table 5.

The first internal threat to validity (Th1) relates to the fact that it was difficult for us to isolate issues related to goal modelling from those related to the extensions introduced in this paper. For example, the specification of the relation between contexts relies on the understandability of the goal model and its relationship with context as well. We assumed that all participants were familiar with goal modelling and focused their effort and comments on the extension itself.

The second internal threat to validity (Th2) relates to the fact that the participants in the study are academic researchers who have already good expertise in requirements engineering and particularly the use of goal models. Moreover, they are familiar with the smart-home system as they all working on and EU project which uses it as a case study. We needed to train the participants on only an extension to the goal model (the context and the analysis input) which was easily understood and applied. However, communicating the principles

Id	Type	Related to	Short description
Th1	Internal	Q2	Difficulty in assessing if issues relate to goal modelling or to our extensions
Th2	Internal	Q2	Participants in the study are researchers with good expertise in goal modelling
Th3	Internal	Q1, Q2	Participants were aware of the purpose of the study
Th4	External	Q1, Q2	Generalizability to different application domains is limited
Th5	External	Q2, Q3	The medium-size scenario used in the case study does not reveal all limitations
Th6	External	Q1	Our analysis detects limited kinds of conflicts and inconsistencies
Th7	Reliability	Q3	Scalability of automated reasoning assessed on a single domain
Th8	Reliability	Q2	Participants carried out only one modelling activity, no re-testing

Table 5: Threats to validity related to our evaluation

and guidelines to novice practitioners might raise other concerns and limitations related to the understandability and the acceptability of our entire modelling framework.

The third threat affects internal validity (Th3), as it relates to the fact that the participants were aware of the overall purpose of the study. This would influence (bias) their specification of the models and the analysis input. We had to explain the purpose to the participants as a preliminary step to ask them to specify the relations between contexts and the objects affected by each task. Future work includes conducting studies in real-world settings where participants focus on the modelling and rely on the tool interactively to detect errors.

The fourth threat affects external validity (Th4), as it concerns the generalizability of our modelling and analysis to other applications domains. Indeed, we conducted our study on a single application domain which exhibits a clear and visible interaction (sensing and actuation) with the physical tangible environment surrounding it. Our modelling framework allows to specify such tangible and visible context and the effect of requirements on tangible objects in the environment. However, in other application areas, such as contextual communication and contextual search engines, this might not be the case. In such domains, context concerns a less visible world and thus the specification of contextual requirements might be problematic and the consistency and conflict check will be consequently error-prone. We still need to adjust, and probably extend, our modelling and reasoning framework to fit a range of applications domains where the nature of requirements and context is different.

A second threat to external validity (Th5) concerns the medium size of our used scenario. This has several implications on the feasibility of specifying contextual requirements. As we mention in the paper, the specification of the relations between context is still not supported by automated means which would predict certain relations and infer some other relations from the already defined ones. Defining these relations for the goal model from the size of this study (shown above) was reasonably hard task and did not require extremely long time. However, more complex models may result in incomplete and/or inconsistent specifications of contexts and their relations. This leads to incomplete and/or incomplete analysis of consistency and conflicts.

Another threat to external validity (Th6) concerns the expressiveness of our adopted model, the contextual goal model. Our context analysis allows capturing context which is tangible, i.e., monitorable based on concrete facts in the system environment. However, context in its broader sense could include things which does not have a tangible nature such as laws and regulations, users preferences, etc. These context could also be preconditions on the various system alternatives and could also change as a consequence of the actions performed by the system. As a result of the limitation of the model we are adopting, our automated analysis detects limited set of inconsistencies and conflicts and should not be treated as comprehensive analysis.

Reliability threats affect the stability, accuracy, and precision of the measurements. In general, these threats concern the repeatability of the study. A threat to validity of this type concerns the scalability of our automated reasoning (Th7). Our tests were conducted by cloning parts of the contextual goal model created by the participants in the case study on the smart-home scenario. A different scenario could create different types of relations between the elements and threaten the scalability of the automated reasoning. That is, different scenarios might require larger amount of input or input of different nature which lead to different performance results.

Another threat to reliability (Th8) arises because participants carried out the modelling activity only once, immediately after being trained. No further re-testing was performed, e.g., in the subsequent few days. It could be the case, for example, that in further modelling

activities the participants would have a different understanding of the concepts, or would have taken more (or less) time to complete the modelling.

8. Related Work

Several authors studied the automated checking of the consistency of requirements specification. Heitmeyer et al [42, 43] analyse requirements specifications expressed in the SCR (Software Cost Reduction) tabular notation for detection of errors, such as type errors, non-determinism, missing cases, and circular definitions. van Lamsweerde et al [44] study the management of conflicts in goal-driven requirements engineering. The work focuses on the case of conflicting formulations of goals and requirements and provide formal techniques and heuristics for detecting and resolving conflicts. Consistency checking for Tropos goal models [11, 12] is discussed in [45]. Following the specification of certain relations between the constructs of goal model, the proposed algorithm finds out if a configuration of goals is a consistent way to satisfy a higher-level goal. Our work enriches these works in two ways. First, we study the consistency of the specification of the context, i.e., the world in which requirements are situated. Second, we study the conflict of actions (tasks) performed to reach the requirements , i.e., we compliment the study of conflicted intentions by studying the conflicted actions. However, our work still does not provide mechanisms for resolving our kinds of conflicts and inconsistencies, and we aim to benefit from the mechanisms proposed in [44] to accomplish that.

Research about feature interaction (for a survey see [46]) concerns predicting scenarios in which an interaction between system features occurs, and judging if an interaction is harmful and providing resolution mechanisms if this is the case. Recently, Nhlabatsi et al. [27] observed, following a large survey of the literature, that the feature interaction problem is essentially a problem about shared context; i.e. there is no interaction without a subject that is an object in the system environment. In line with this observation, our model supports an explicit notion of context and captures how context influences, and how it is influenced by, the requirements at the goal level. We develop reasoning mechanisms that use the model to detect and provide essential information about the conflicts occurring between system executable processes (tasks). This information includes the goals for which and the context in which a conflict happens, the alternatives the system has to avoid conflicts, and so on.

Requirements engineering for adaptive systems raises several challenges [47, 48]. These challenges cover a wide spectrum of topics such as requirements monitoring, requirements models uncertainty and flexibility and requirements-driven adaptation at runtime. In the next three paragraph we discuss how our modelling and analysis framework aligns to each of these three areas.

Requirements monitoring is about the injection of 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 behaviour to them if a deviation occurs [15]. The objective is to have more robust, maintainable, and self-evolving systems. In [49], the GORE (Goal-Oriented Requirements Engineering) framework KAOS [13] was integrated with an event-monitoring system (FLEA [50]) in an architecture that enables runtime automated reconciliation. The developed reconciliation is between the system goals and the system behaviour with respect to a priori anticipated or evolving changes in the system environment that is mostly technical. In our work, we start earlier and capture the influence of context on users goals. Such context concerns the environment, not necessarily technical, of the user and the system. Moreover, we provide a conceptual modelling language that supports an explicit notion of context and a systematic way to analysed it in conjunction with goal model. Our work could be integrated with FLEA towards more holistic reconciliation between system and user goals from one side and system and user contexts from the other.

Qureshi and Perini [51] emphasize on uncertainty and flexibility of requirements refinement and provide a method that supports the runtime refinement of requirements artifacts as a repetitive activity performed collaboratively between the users and the application itself. Sawyer et al. [52] discuss that runtime representation of requirements model, synchronizing the model with the architecture, dealing with uncertainty, multiple objective decision making, and self-explanation are areas need to be considered in realizing a requirements-aware system. Our framework presumes certainty while specifying the relation between context and requirements. Thus, we still need to investigate how to deal with situations when designers are not fully certain about it. Bencomo et al. [53] advocate that adaptation is planned either in a pre-defined way at design time or via an evolvable and reflexive response to some monitored parameters at runtime. In line with our work, they advocate that the gap between goals and the system has to be bridged so that the system adaptation is guided by goals and the adaptation correctness is judged by the fulfilment

of goals (requirements reflection).

Several authors studied requirements-driven adaptation at runtime. Souza et al. [54] note that the (partial) un-fulfilment of requirements triggers adaptation. They introduce awareness requirements to refer to success, failure, performance and other properties of software requirements (i.e. meta-requirements) and propose to monitor changes in these properties and decide when adaptation should take place. Baresi et al. [55] propose FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) for requirements-driven adaptation at runtime. FLAGS extend KAOS [13] mainly with adaptive goals which incorporate countermeasures for adaptation. When goals are not achieved by the current course of execution, adaptation countermeasures are triggered. The ultimate target is to alter the goal model at runtime and enforce adaptation directives on the running system. Our framework has the potential to enrich these two works by the consideration of another trigger of adaptation at runtime which is the changes of context perceived as the environment surrounding the system.

In a recent work [56], we have argued that user's judgement of the quality of each software behaviour is a main driver for adaptation. This feedback has to be engineered and captured in a systematic and structured way so that software can make use of it. We have classified feedback into quality feedback reflecting users' judgement on the degree of excellence of a behaviour and validity feedback reflecting their judgement whether a behaviour is a valid means to reach certain requirements. We have also proposed that users can act as monitors for contextual attributes which are unmonitorable when we rely on automated means solely [57]. This means that users act as software collaborators rather than acting as pure consumers of the software functionalities. In our other recent work [58] we have proposed analysis mechanisms to optimize the monitoring requirements in adaptive systems by deriving the minimum amount of contextual data to capture with minimum costs.

Contextual requirements research studies an early influence of context on the adaptive systems. Research in context modelling (e.g., [37]) concerns finding modelling constructs to represent software and user's context. Contextual requirements research reduces the gap between the context model and the requirements model, i.e. between context and one of its usages. In our approach, we reduce such gap 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 requirements?*". Salifu et al. [8] apply the Problem Frames approach to analyse 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. Hartmann et al. [59] suggest studying the relation between context and features to support the engineering of software supply chains. Their approach allows for more systematic derivation of a product that fits with the environment wherein it operates. In our position paper [60], we suggested the integration of our work with the above works that considered adaptability to context and we showed, theoretically, how such integration could help for holistic software production. In another position paper [61], we advocated the role of user's collective judgement on the quality of each feature configurations as a main factor to drive the dynamic configuration process of a product in a product line.

In our previous work, we have proposed to weave together the variability of the system and the variability of its environment. Our argument is that the environment changes can activate certain requirements, restrict the space of applicable alternatives to reach activated requirements and influence the quality of each alternative as [9, 20]. We have proposed context analysis model which included constructs to analyse context to reach visible facts the system can monitor and judge upon if a context holds [22]. We have also provided several mechanisms to exploit and reason about this relation at design and runtime [21]. This includes a runtime derivation of alternatives compliant with contexts and user priorities expressed via ranking the importance of softgoals, and deriving alternatives with minimized development costs. Similarly to contextualizing requirements, we have investigate the contextualization of business process so that a business process is adaptive to context changes [62, 63]. The execution course will be context-dependent so that the correctness probability of business process is maximized. However, we still miss a multi-factor decision making that leads to a compromise between different adaptation drivers (priorities, development costs, effort, privacy, ..) either at design time or at runtime.

Software variability modelling, mainly feature models [64, 65], concerns capturing a variety of possible configurations of software functionalities. This allows for tailoring a product depending on stakeholders choices. However, there is still a gap between each functionality and the context where such functionality

can or has to be adopted. We have tried to solve this problem at the level of goals. Furthermore, our work is in line, and has the potential to be integrated, with the work in [66] and the FARE method proposed in [67] that show possible ways to integrate features with domain goals and knowledge to help for eliciting and justifying features.

Customizing goal models to fit to user skills and preferences was studied in [34, 68]. The selection between goal model variants 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 expressed over softgoals. Lapouchnian et al. [69] propose techniques to design autonomic software based on an extended goal modelling framework, but their approach does not focus on the relation with context. Liaskos et al. [32] study variability modelling under a requirements engineering perspective and propose a classification of intentional variability that originate goal satisfaction alternatives. We focused on context variability, i.e. the unintentional variability, that highly influences the applicability and quality of each goal satisfaction alternative.

9. Conclusions and Future Work

In this paper, we have extended our goal-based contextual requirements engineering framework (proposed in [21]) with novel mechanisms for (i) context consistency analysis; and (ii) conflict analysis. Both techniques are supported by our tool RE-Context, which supports analysts by checking consistency of and conflict in requirements models. In addition to these techniques, we proposed a methodology for contextual goal modelling and its associated reasoning techniques. The first reasoning technique is designed to detect inconsistencies between contexts specified on a goal model. We have discussed the semantics of different kinds of detected inconsistencies and illustrated each inconsistency type. The second reasoning mechanism is designed to detect and assess the severity of conflicts originating from changes in the context (produced by the system itself).

We have evaluated our framework from three perspectives. The first is about the usefulness of our designed techniques in detecting inconsistencies and conflicts which are not easily recognizable by requirements engineers. Our automated analysis detected a considerable amount of such cases and proved to be useful in that sense. The second is the ability to understand and capture our extension of goal model which we proposed in this paper. Most of our modelling extension constructs were found easy to understand and easy to capture. Only one construct, the relation between contexts, were found hard to capture and time-consuming and, thus, further automated support to the specification of these relations is still needed. The third perspective concerns the scalability of our automated reasoning. Our analysis was proved to scale well with small and medium sizes of goal model while it needs further optimization to cope with large-size ones.

In future work, we plan to address various problems in the area of contextual requirements engineering, such as:

- **Reasoning about monitoring requirements:** the system at runtime needs to collect environmental data to judge if a certain context holds and adapt to it by adopting a suitable behavior. This raises a new category of requirements called *Monitoring Requirements*, i.e., what data the system has to capture from its environment and the way these data are logically composed to judge if a certain context holds. Monitoring requirements need specific analysis. An example of such analysis is the optimization of monitoring requirements, i.e., finding the less expensive set of data to capture required to verify if a given context holds. For example, a context specified as $(\varphi_1 \wedge \varphi_2)$ where φ_1 = "patient is inside home" and φ_2 = "it is cold at the patient's location" can be reduced into (φ_1) if the health-care institute regulates temperature inside homes in way that prevents going lower than a certain level. A positioning system would suffice to verify if context φ_1 holds, for the system need not verify φ_2.
- **Lifelong contextualization**: context-aware systems need to monitor context at runtime and adapt to it. It is desirable that the system evolves over time and enhances the way it satisfies user's needs in different contexts. Indeed, not all decisions can be fully specified at design time. A running system might collect data and learn which requirements and which alternatives fit better to particular environments. For example, patients could prefer remote communication with caregiver in one context and in person in some other context. This knowledge is typically unavailable at design time, for considered patients might change their habits and unknown patients might use the system. The system can therefore evolve choosing the best way of communication benefiting of the history of the patient's behaviour in different contexts.
- **Automated support of modelling activities**: a

CASE tool is needed for supporting all the modelling activities proposed in this paper. For example, we still transform the graphical representation of contextual goal models into Datalog manually. Moreover, we presume that the logical relations between contexts, i.e., contradictions and implications, are also manually specified by the designers. Defining these relations for small-medium size systems could be doable manually. For larger systems, manual specification is error-prone and time consuming.

Acknowledgement

This work has been partially funded by the EU Commission, through the ANIKETOS and FastFix projects and by Science Foundation Ireland grant 10/CE/I1855. We also thank Jaelson Brelaz de Castro, Bashar Nuseibeh, John Mylopoulos, Sarah Beecham, Alberto Griggio, Anders Franzen, Yijun Yu, Armstrong Nhlabatsi, and Amit K. Chopra for the helpful discussions that enriched the ideas in this paper.

References

[1] M. Weiser, The Computer for the Twenty-First Century, Scientific American 265 (1991) 94–104.

[2] J. Krogstie, K. Lyytinen, A. L. Opdahl, B. Pernici, K. Siau, K. Smolander, Research Areas and Challenges for Mobile Information Systems, International Journal of Mobile Communications 2 (2004) 220–234.

[3] J. Kramer, J. Magee, Self-Managed Systems: an Architectural Challenge, in: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), IEEE Computer Society, 2007, pp. 259–268.

[4] P. Oreizy, N. Medvidovic, R. N. Taylor, Runtime Software Adaptation: Framework, Approaches, and Styles, in: Companion of the 30th International Conference on Software Engineering (ICSE Companion '08), pp. 899–910.

[5] A. Schmidt, Implicit Human Computer Interaction through Context, Personal and Ubiquitous Computing 4 (2000) 191–199.

[6] S. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, Y. Berbers, Easy: Efficient Semantic Service Discovery in Pervasive Computing Environments with QoS and Context Support, The Journal of Systems & Software 81 (2008) 785–808.

[7] J. Krogstie, Requirement Engineering for Mobile Information Systems, in: Proceedings of the 7th International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ'01).

[8] M. Salifu, Y. Yu, B. Nuseibeh, Specifying Monitoring and Switching Problems in Context, in: Proceedings of the 15th International Conference on Requirements Engineering (RE'07), pp. 211–220.

[9] R. Ali, F. Dalpiaz, P. Giorgini, Location-based Variability for Mobile Information Systems, in: Z. Bellahsene, M. Léonard (Eds.), Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), volume 5074 of *LNCS*, Springer, 2008, pp. 575–578.

[10] E. S.-K. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. thesis, University of Toronto, Toronto, Ont., Canada, Canada, 1996.

[11] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An Agent-Oriented Software Development Methodology, Autonomous Agents and Multi-Agent Systems 8 (2004) 203–236.

[12] J. Castro, M. Kolp, J. Mylopoulos, Towards Requirements-Driven Information Systems Engineering: The Tropos Project, Information Systems 27 (2002) 365–389.

[13] A. Dardenne, A. van Lamsweerde, S. Fickas, Goal-directed Requirements Acquisition, Science of Computer Programming 20 (1993) 3–50.

[14] J. Mylopoulos, L. Chung, E. Yu, From Object-Oriented to Goal-Oriented Requirements Analysis, Communications of the ACM 42 (1999) 31–37.

[15] S. Fickas, M. S. Feather, Requirements Monitoring in Dynamic Environments, in: Proceedings of the 2nd IEEE International Symposium on Requirements Engineering (RE'95), IEEE Computer Society Washington, DC, USA, 1995, pp. 140–147.

[16] D. Sykes, W. Heaven, J. Magee, J. Kramer, From Goals to Components: a Combined Approach to Self-Management, in: Proceedings of the 2008 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008), pp. 1–8.

[17] G. Brown, B. H. C. Cheng, H. Goldsby, J. Zhang, Goal-Oriented Specification of Adaptation Requirements Engineering in Adaptive Systems, in: Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems (SEAMS '06), ACM, New York, NY, USA, 2006, pp. 23–29.

[18] A. van Lamsweerde, Goal-oriented Requirements Engineering: A Guided Tour, in: Proceedings of the 5th IEEE International Symposium on Requirements Engineering (RE 2001), pp. 249–263.

[19] E. Yu, J. Mylopoulos, Why Goal-Oriented Requirements Engineering, in: Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'98), pp. 15–22.

[20] R. Ali, F. Dalpiaz, P. Giorgini, Location-based Software Modeling and Analysis: Tropos-based Approach, in: Q. Li, S. Spaccapietra, E. Yu, A. Olivé (Eds.), Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008), volume 5231 of *LNCS*, Springer, 2008, pp. 169–182.

[21] R. Ali, F. Dalpiaz, P. Giorgini, A Goal-based Framework for Contextual Requirements Modeling and Analysis, Requirements Engineering 15 (2010) 439–458.

[22] R. Ali, F. Dalpiaz, P. Giorgini, A Goal Modeling Framework for Self-Contextualizable Software, in: Proceedings of the 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2009), volume 29 of *LNBIP*, Springer, pp. 326–338.

[23] R. Ali, F. Dalpiaz, P. Giorgini., Goal-based Self-Contextualization, in: In the Forum of the 21st International Conference on Advanced Information Systems (CAiSE 09 - Forum), volume Vol-453, CEUR-WS, 2009, pp. 37–42.

[24] S. Campadello, L. Compagna, D. Gidoin, S. Holtmanns, V. Meduri, J.-C. R. Pazzaglia, M. Seguran, R. Thomas, Serenity Deliverable A7.D1.1: Scenario Selection and Definition, 2006.

[25] D. L. Parnas, J. Madey, Functional Documents for Computer Systems, Science of Computer Programming 25 (1995) 41–61.

[26] A. Biere, M. J. H. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2009.

[27] A. Nhlabatsi, R. Laney, B. Nuseibeh, Feature Interaction: the Security Threat from within Software Systems, Progress in In-

formatics (2008) 75–89.

[28] A. Lapouchnian, Y. Yu, J. Mylopoulos, Requirements-Driven Design and Configuration Management of Business Processes, in: G. Alonso, P. Dadam, M. Rosemann (Eds.), Proceedings of the 5th International Conference on Business Process Management (BPM 2007), volume 4714 of *LNCS*, Springer, 2007, pp. 246–261.

[29] T. Eiter, G. Gottlob, H. Mannila, Disjunctive Datalog, ACM Transactions on Database Systems 22 (1997) 364–418.

[30] C. Rolland, C. Souveyet, C. B. Achour, Guiding Goal Modeling using Scenarios, IEEE Transactions on Software Engineering 24 (1998) 1055–1071.

[31] J. Kim, M. Kim, S. Park, Goal and Scenario Based Domain Requirements Analysis Environment, Journal of Systems and Software 79 (2006) 926–938.

[32] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, J. Mylopoulos, On Goal-based Variability Acquisition and Analysis, in: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE 2006), IEEE Computer Society, 2006, pp. 76–85.

[33] F. Dalpiaz, P. Giorgini, J. Mylopoulos, An Architecture for Requirements-Driven Self-Reconfiguration, in: Proceedings of the 21st International Conference on Advanced Information Systems Engineering (CAiSE'09), volume 5565 of *LNCS*, Springer, 2009, pp. 246–260.

[34] B. Hui, S. Liaskos, J. Mylopoulos, Requirements Analysis for Customizable Software: A Goals-Skills-Preferences Framework, in: Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE 2003), IEEE Computer Society Washington, DC, USA, pp. 117–126.

[35] X. H. Wang, D. Q. Zhang, T. Gu, H. K. Pung, Ontology Based Context Modeling and Reasoning using OWL, in: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04), pp. 18–22.

[36] S. S. Yau, J. Liu, Hierarchical Situation Modeling and Reasoning for Pervasive Computing, in: Proceedings of the 4th IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS '06), pp. 5–10.

[37] K. Henricksen, J. Indulska, A Software Engineering Framework for Context-Aware Pervasive Computing, in: Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom'04), pp. 77–86.

[38] Y. Wang, S. McIlraith, Y. Yu, J. Mylopoulos, An Automated Approach to Monitoring and Diagnosing Requirements, in: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07), ACM New York, NY, USA, 2007, pp. 293–302.

[39] R. K. Yin, Case Study Research: Design and Methods, volume 5, Sage publications, 4 edition, 2008.

[40] P. Runeson, M. Höst, Guidelines for Conducting and Reporting Case Study Research in Software Engineering, Empirical Software Engineering 14 (2009) 131–164.

[41] C. Wohlin, Experimentation in Software Engineering: an Introduction, volume 6, Springer, 2000.

[42] C. L. Heitmeyer, R. D. Jeffords, B. G. Labaw, Automated Consistency Checking of Requirements Specifications, ACM Transactions on Software Engineering and Methodology 5 (1996) 231–261.

[43] C. Heitmeyer, B. Labaw, D. Kiskis, Consistency Checking of SCR-style Requirements Specifications, in: Proceedings of the Second IEEE International Symposium on Requirements Engineering, pp. 56–63.

[44] A. Van Lamsweerde, R. Darimont, E. Letier, Managing Conflicts in Goal-driven Requirements Engineering, IEEE Transactions on Software Engineering 24 (1998) 908–926.

[45] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, R. Sebastiani, Reasoning with Goal Models, in: Proceedings of the 21st International Conference on Conceptual Modeling (ER 2002), pp. 167–181.

[46] M. Calder, M. Kolberg, E. Magill, S. Reiff-Marganiec, Feature Interaction: a Critical Review and Considered Forecast, Computer Networks 41 (2003) 115–141.

[47] B. H. C. Cheng, H. Giese, P. Inverardi, J. Magee, R. de Lemos, Software Engineering for Self-Adaptive Systems: A Research Road Mapre, in: Software Engineering for Self-Adaptive Systems, pp. 1–26.

[48] B. H. C. Cheng, J. M. Atlee, Research Directions in Requirements Engineering, in: 2007 Future of Software Engineering, FOSE '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 285–303.

[49] M. S. Feather, S. Fickas, A. van Lamsweerde, C. Ponsard, Reconciling System Requirements and Runtime Behavior, in: Proceedings of the 9th International Workshop on Software Specification and Design (IWSSD'98), IEEE Computer Society Washington, DC, USA, 1998, pp. 50–59.

[50] D. Cohen, M. S. Feather, K. Narayanaswamy, S. S. Fickas, Automatic Monitoring of Software Requirements, in: Proceedings of the 19th International Conference on Software Engineering (ICSE 1997), ACM New York, NY, USA, 1997, pp. 602–603.

[51] N. A. Qureshi, A. Perini, Requirements engineering for adaptive service based applications, in: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 108–111.

[52] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, A. Finkelstein, Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems, in: Proceedings of the 2010 18th IEEE International Requirements Engineering Conference, RE '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 95–103.

[53] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, E. Letier, Requirements Reflection: Requirements as Runtime Entities, in: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE '10, ACM, New York, NY, USA, 2010, pp. 199–202.

[54] V. E. Silva Souza, A. Lapouchnian, W. N. Robinson, J. Mylopoulos, Awareness Requirements for Adaptive Systems, in: Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems, SEAMS '11, ACM, New York, NY, USA, 2011, pp. 60–69.

[55] L. Baresi, L. Pasquale, P. Spoletini, Fuzzy Goals for Requirements-Driven Adaptation, in: Proceedings of the 18th IEEE International Requirements Engineering Conference (RE'10), IEEE Computer Society, Washington, DC, USA, 2010, pp. 125–134.

[56] R. Ali, C. Solis, I. Omoronyia, M. Salehie, B. Nuseibeh, Social Adaptation: When Software Gives Users a Voice, in: the proceedings of 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'12).

[57] R. Ali, C. Solis, M. Salehie, I. Omoronyia, B. Nuseibeh, W. Maalej, Social Sensing: When Users Become Monitors, in: the proceedings of the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2011), pp. 476–479.

[58] R. Ali, A. Griggio, A. Franzen, F. Dalpiaz, P. Giorgini, Optimizing Monitoring Requirements in Self-Adaptive Systems, in: the Proceedings of 17th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'12).

[59] H. Hartmann, T. Trew, Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains, in: Proceedings of the 12th International Software Product Line Conference (SPLC'08), IEEE Computer Society, 2008, pp. 12–21.

[60] R. Ali, Y. Yu, R. Chitchyan, A. Nhlabatsi, P. Giorgini, Towards a Unified Framework for Contextual Variability in Requirements, in: Proceedings of the 3rd International Workshop on Software Product Management (IWSPM09).

[61] R. Ali, C. Solis, F. Dalpiaz, W. Maalej, P. Giorgini, B. Nuseibeh, Social Software Product Lines, in: the Proceedings of the 1st international Workshop on Requirements Engineering for Social Computing (RESC 2011), pp. 14–17.

[62] J. L. De La Vara, R. Ali, F. Dalpiaz, J. Sánchez, P. Giorgini, Business Processes Contextualisation via Context Analysis, in: Proceedings of the 29th international conference on Conceptual modeling (ER'10), Springer-Verlag, Berlin, Heidelberg, 2010, pp. 471–476.

[63] J. L. De La Vara, R. Ali, F. Dalpiaz, J. Sánchez, P. Giorgini, COMPRO: a Methodological Approach for Business Process Contextualisation, in: Proceedings of the 2010 International Conference on On the Move to Meaningful Internet Systems - Volume Part I (OTM'10), Springer-Verlag, Berlin, Heidelberg, 2010, pp. 132–149.

[64] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, 2005.

[65] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, M. Huh, FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architectures, Annals of Software Engineering 5 (1998) 143–168.

[66] Y. Yu, J. do Prado Leite, A. Lapouchnian, J. Mylopoulos, Configuring Features with Stakeholder Goals, in: Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008), ACM New York, NY, USA, pp. 645–649.

[67] M. Ramachandran, P. Allen, Commonality and Variability Analysis in Industrial Practice for Product Line Improvement, Software Process: Improvement and Practice 10 (2005) 31–40.

[68] S. Liaskos, S. McIlraith, J. Mylopoulos, Representing and Reasoning with Preference Requirements using Goals, Technical Report, Tech. rep. CSRG-542, Computer Science Department, University of Toronto, 2006.

[69] A. Lapouchnian, Y. Yu, S. Liaskos, J. Mylopoulos, Requirements-driven Design of Autonomic Application Software, in: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative research (CASCON'06), ACM Press New York, NY, USA, 2006.