Contextual Goal Models

Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini

February 2010

Technical Report # DISI-10-020

Contextual Goal Models

Raian Ali, Fabiano Dalpiaz and Paolo Giorgini

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

Abstract. In emerging computing paradigms, such as mobile, ubiquitous, and pervasive computing, the relation between context and requirements is evident. Context might be a main factor to determine the set of requirements relevant to a system, the alternatives that can be adopted to meet these requirements, and the quality of each of such alternatives. In spite of that, most requirements engineering (RE) research ignores, or presumes a uniform nature of, the context where the system operates. A RE framework specialized for systems reflecting their context is still missing. Before influencing the behavior of software, context influences the behavior of users. It influences user's goals and their choices to reach these goals. Capturing this latest influence is an essential step towards software developed to meet user's requirements in different contexts.
In this report, we develop a goal-based RE modeling framework for systems operating in and reflecting varying contexts. We propose the *contextual goal model* that captures the relation between alternatives for goal salification and context. Moreover, it provides constructs to hierarchically analyze context and identify alternative ways to judge if a context holds. We illustrate our proposed model via a scenario of promotion mobile information system.

1 Introduction

The advances of computing, sensors, and communication technology helped the realization of new computing paradigms such as ambient, ubiquitous and pervasive computing. These paradigms weave computing systems with humans' living environments to transparently meet their needs [1]. Context, a core element of these settings, can be defined as the reification of the environment, that is whatever provides a surrounding in which a system operates [2]. Context can influence the requirements of a system and the variants a system can adopt to meet its requirements. Moreover, context is by nature variable in these paradigms and it calls for new approaches to create system that can adapt to context changes.

Software systems are means to reach user requirements and they are not requirements per se [3, 19, 4]. One important source of requirements is the user's goals and their variant choices to reach them. Context has influence at this level, the goal level, deciding what goals to reach and how to reach them. For example, in a health care institute for people with dementia, a caregiver may have the goal G_1: *"patient is involved in social life"* which is activated in a context like C_1: *"the patient is feeling bored and it has been long time since his last social activity"*.

The caregiver could reach G_1 by reaching one of the alternative subgoals $G_{1.1}$: *"take the patient for a trip in the city"* and $G_{1.2}$: *"a relative or an old friend of the patient comes to visit him"*. The alternative $G_{1.1}$ is adoptable if the context $C_{1.1}$: *"the city is not crowded"* holds, since people with dementia usually get anxious in crowded places. An automated system built to support the life of people with dementia should reflect the caregiver goals (G_1, $G_{1.1}$, and $G_{1.2}$), the rationale ($G_{1.1} \vee G_{1.2} \rightarrow G_1$) and adaptation to contexts (if $C_1 \wedge C_{1.1}$ then $G_{1.1}$).

Goal models (*i** [5], Tropos [6], and KAOS [7]) represent an intentional ontology used at the early requirements analysis phase to explain the *why* of a software system. They have been used to represent the rationale of both humans and software systems [8] and they provide useful constructs to analyze high level goals and ways to satisfy them. Such features are essential for the analysis and the design of a software system supposed to reflect stakeholders' rationale and adaptation to varying contexts [9, 10].

In this report, we propose a modelling language for contextual requirements. We propose the *contextual goal model* that associates context and requirements at the goal level [11, 12]. To this end, we propose a set of variation points on Tropos goal model where context may influence the selection between variants for goal satisfaction. Moreover, the context at each of these points needs to be specified. The specification here refers to the way through which the truth of context is judged. For this reason, we propose modeling constructs to analyze contexts and identify variant ways to judge if a context holds (refining [13, 14]).

This report is structured as follows. Section 2 outlines a system scenario of promotion information system to be used as a running example; Section 3 briefly discusses Tropos goal modeling concepts; Section 4 gives definition of context from the perspective of requirements engineering; Section 5 proposes the contextual goal model that incorporates context with goals and provide constructs to analyze context; and Section 6 concludes the report.

2 Running Example

In this section, we briefly explain an example of a system operating in and reflecting varying contexts and we use it to explain our proposed contextual goal model. We consider an information system for promoting products to customers inside shopping malls. The customers and sales staff are provided by PDAs as a communication and interaction device. The system can satisfy its main goal *"promoting a product to a customer"* through different execution courses. The adopted execution course depends on the context the includes the characteristic of customers, products, sales staff and other elements in the shopping mall.

To initiate the promotion process, a certain initial context has to hold: the customer is inside the mall building and he may accept an interaction for the promotion of a product. This context activates the need to reach the main goal of the system that is the useful promotion of the products to interested customers to increase their sales. The system at runtime has to monitor such a context to decide upon when to activate the promotion of a product.

Promotion can be done through several alternatives and each alternative may require a valid context. One of these alternatives is cross-selling. Promoting a product, by cross-selling it, requires that the product complements or it is usually sold together with an already chosen or bought product by the customer. If this context holds, the system has to show a demo to persuade the customer and then to display the place where the customer can go and pick up the product from.

Another alternative to promote a product is offering a discount on it. To promote by offering a discount, the system has to monitor if the product needs to be finished soon and if the customer is interested in the product through analyzing his sales history or current behavior inside the mall. If such context holds, the system generates and gives a discount code for the customer to provide at the cash desk and benefit from the discount.

Some products can be promoted by giving a free samples of them to customers. To promote by offering a free sample of a product, the system has to monitor if the product is new to the customer. *"product is new to a customer"* is a high level context which may mean that the customer never bought the product, the product is newly released, or the product is local to the mall region while the customer lives in a different region where the product is not local in.

Adopting the promotion by free sample, the system has to decide the way to deliver the sample to the customer. Assigning the delivery task to a sales staff is adoptable when the sales staff is close to and speaks a language in common with the customer, and knows well about the product. Delivering the sample by self-service machine requires that the customer knows how to use such kind of machines, and the machine has a short queue, and it is not so far from the customer.

If delivering the free sample by a sales staff is adopted, the system has to notify and guide the sales staff to meet the customer. If delivering the sample by machine is adopted, the software has to explain about the product, get customer confirmation, give an authentication code to the customer to provide to the self-service machine, and guide the customer to arrive to such machine. Guidance to the machine can be done by showing a map and the path on it if the machine is close or the path is simple. Otherwise, the system has to trace the customer and direct him step by step.

3 Tropos Goal Model: Overview

Goal analysis represents a paradigmatic shift with respect to object-oriented analysis. While object-oriented analysis fits well to the late stages of requirement analysis; goal-oriented analysis is more natural for the earlier stages where the organizational goals are analyzed to identify and justify software requirements and position them within the organizational system [8]. Goal analysis justifies the developed system by relating it to users strategic goals. In other words, goal analysis answers the question "why is a software needed".

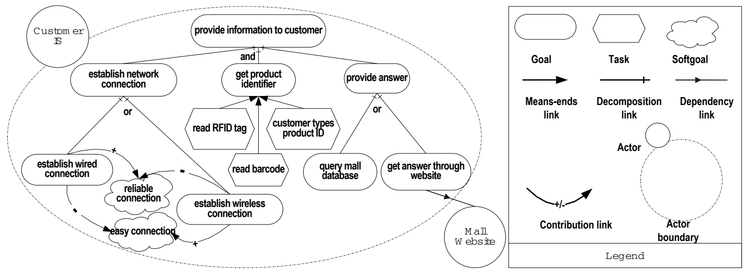


Fig. 1. Tropos goal model example

In Fig. 1, we show a partial Tropos goal model of the running example of promotion information system to clarify our goal analysis main concepts. Tropos goal analysis projects the system as a set of interdependent actors, each having its own strategic interests (*goals*). Goals are analyzed iteratively and in a top-down way, to identify the more specific sub-goals needed for satisfying the upper-level goals. Goals can be ultimately satisfied by means of executable processes (*tasks*).

Actors (*Customer IS* and *Mall Website*) have a set of top-level goals (*provide information to customer*), which are iteratively decomposed into subgoals by and-decomposition (all subgoals should be achieved to fulfil the top goal) and or-decomposition (at least one subgoal should be achieved to fulfil the top goal). The goal *provide information to customer* is and-decomposed into *establish network connection*, *get product identifier*, and *provide answer*; the goal *provide answer* is or-decomposed into *query mall database* and *get answer through website*. Goals are finally satisfied by means of executable tasks; the goal *"get product identifier"* can be reached by one of the tasks *"read RFID tag"*, *"read barcode"*, *"let customer type product ID"*.

A dependency indicates that an actor (*depender*) depends on another actor (*dependee*) to attain a goal or to execute a task: the actor *Customer IS* depends on the actor *Mall Website* for achieving the goal *get answer through website*. Soft-goals are qualitative objectives for whose satisfaction there is no clear cut criteria (*easy connection* is a rather vague objective), and they can be contributed either positively or negatively by goals and tasks: *establish wireless connection* contributes positively to *easy connection*, while *establish wired connection* contributes negatively to *easy connection*.

4 Context in Requirements

Context has been defined in multiple computer science disciplines especially in artificial intelligence (for a survey see [15]). It has been also defined in the literature of emerging computing paradigms, such as ubiquitous, adaptive, and mobile systems [16, 2, 17], that our requirements engineering framework is developed for. A specific definition of context strongly depends on the domain it is used in.

For example, in a context sensitive search engines, a user may search the term "java" that could mean a programming language or an island. To disambiguate the searched term, the engine may look to the context that can be the query history. If the user asked recently for the term "cgi programming", then most probably he is looking for the Java programming language [18]. In the rest of this section, we adapt a definition of context from the perspective of requirements engineering, namely goal-oriented requirements engineering.

As broadly accepted, software is a means to meet user requirements [3, 19, 8, 20]. Software is developed to solve a problem in the users world and to help them to reach their goals. In line with this view of requirements, Tropos requirements analysis projects a system, either organizational or software, as a set of interdependent actors. Each actor has goals which are partial states of the world an actor attempts to reach. Tropos goal analysis represents alternative sets of tasks that an actor may execute trying to reach its goals. In other words, tasks are not required per se, but are means to reach goals. Actors are autonomous in deciding what goals to reach, how, and how well to reach them. We here give a definition of actor, adapted from [6], that is going to be the observer of a context:

Definition 1 (Actor) *an actor is an entity that has goals and can decide autonomously how to achieve them.*

An actor can be of different types such as human actors, software actors, or organizational actors. The main characteristic of an actor is the autonomy in deciding the way to reach its goals. This includes the ability to decide what goals to reach, how, and how well to reach them. For example, a sales staff is a human actor that may have the goal of conveying appropriately information about products to customers. The sales staff has the ability to decide when to activate this goal and what to do to reach it. The staff may reach such goal by making a phone call with the customer or by delivering information to him in person and the decision between these two options is left to the sales staff himself. The decision taken by an actor depends on the state of a portion of the world such actor lives in. We call such a state as context:

Definition 2 (Context) *a context is a partial state of the world that is relevant to an actor's goals.*

The decision about the parts of the world that are relevant to an actor decisions is of subjective nature. An actor does not observe the world for the purpose of observation per se. An actor does that for better decision about what goals to reach and what actions to do to reach them. Therefore, such decision is influenced by properties over the world that an actor needs to observe. For example, "customer is interested in a product" is relevant for a sales staff when deciding whether to promote a product to a customer. The same context is irrelevant when a sales staff needs to decide if to announce some new offers via speakers. Moreover, there could be always viewpoints about that parts of the world that are relevant to a decision. For example, to decide the adoptability of conveying information to a customer via an information terminal in the shopping mall, one sales staff attempts to verify the context "customer is very close to one free terminal" and another one may attempts to verify "visitor is close to a terminal or to a map showing the locations of terminals in the mall".

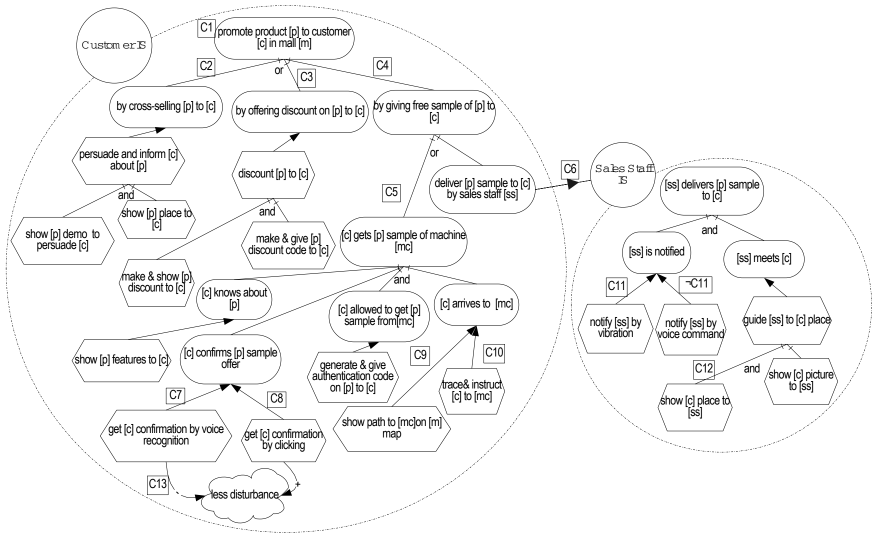
Context is inherently partial and of a volatile nature. Actors may have partial view of the state of the world. They may not be interested or able to capture all the information that fully capture such a state. A state of the world may be partitioned into dimensions such as spatio-temporal, personal, tasks, social as proposed in [17]. This partitioning is a way of facilitating the way a state of the world can be described and captured. The world is volatile and could be in different states. A partial state of the world that is uniform does not influence the decisions of an actor. For example, if the promotion information system operates in a geographic area where shopping malls do not provide information offices, then the system does not need to observe if there are such offices when deciding the way to convey information to a customer. The decision is made once while developing the system and applied in all museums the system will operate in.

5 Contextual Goal Model

Current goal modeling approaches do not support the relationship between goals and contexts. Although, they offer a way to identify variant solutions for goal satisfaction, there is no explicit representation of the relation between a solution and the context where it can be adopted. Supporting variants and contexts separately raises two important questions: *"why does the system support several variants and not just one?"* and *"what can the system do if context changes?"*. This means that an integrated analysis of goal model variants and contexts may allow for a more complete specification of requirements for systems reflecting their varying context. However, enumerating the goal model variants and specifying their corresponding contexts separately may be very hard: *(i)* we may have a huge number of variants and then specifying a context for each of them can be extremely time consuming; *(ii)* each variant can be very complex and then it may become very difficult to specify its corresponding context.

We propose the *Contextual Goal Model* where variation points are introduced to link contexts and goal model variants. For example, Fig. 2 shows a partial Tropos goal model for the promotion mobile information system where contexts are annotated (C_i) at a set of variation points. We define the following variation points where context can be specified influencing the goal model variants:

1. *Or-decomposition*: the adoptability of each sub-goal (sub-task) in an Or-decomposition may require a specific context. For example, *"promoting the product by cross-selling"* can be adopted when the product can be used with another product the customer already has (C_2), while *"promoting by offering discount"* is adopted when product is discountable and interesting to the customer (C_3), and *"promoting by free sample"* can be adopted when product is free sampled and new to the customer ($C4$). The alternative *"get*



CustomerIS
C1
promote product [p] to customer [c] in mall [m]
C2
or
C3
C4
by cross-selling [p] to [c]
by offering discount on [p] to [c]
by giving free sample of [p] to [c]
persuade and inform [c] about [p]
discount [p] to [c]
or
deliver [p] sample to [c] by sales staff [ss]
C6
C5
and
show [p] demo to persuade [c]
show [p] place to [c]
and
make & give [p] discount code to [c]
[c] gets [p] sample of machine [mc]
make & show [p] discount to [c]
and
[c] knows about [p]
[c] allowed to get [p] sample from[mc]
[c] arrives to [mc]
show [p] features to [c]
[c] confirms [p] sample offer
generate & give authentication code on [p] to [c]
C9
C10
trace& instruct [c] to [mc]
C7
C8
get [c] confirmation by voice recognition
get [c] confirmation by clicking
show path to [mc]on [m] map
C13
-
less disturbance
+
Sales Staff IS
[ss] delivers [p] sample to [c]
and
[ss] is notified
[ss] meets [c]
C11
¬C11
notify [ss] by vibration
notify [ss] by voice command
guide [ss] to [c] place
C12
and
show [c] place to [ss]
show [c] picture to [ss]

Fig. 2. Tropos goal model with contexts annotation at the variation points

free sample from a machine" can be adopted when customer has experience with such machines and can reach the machine and start to use it in a little time (C_5).

2. *Means-end*: goals can be ultimately satisfied by means of specific executable processes (*tasks*). The adoptability of each task may require a specific context. For example, *"get customer confirmation by voice recognition"* can be adopted when the customer place is not noisy, and the system is trained enough on the customer voice (C_7), while the alternative *"get customer confirmation by clicking"* can be adopted when the customer has a good level of expertise with regards to using technology and a good control on his fingers, and the used device has a touch screen (C_8). The task *"show path to sample machine on the mall e-map"* is adopted when customer can arrive easily to that machine (C_9), while *"trace and instruct customer to sample machine"* task is adopted when the path is complex (C_{10}). The task *"notify by vibration"* can be adopted when sales staff is using his PDA for calling (C_{11}), while *"notify by headphone voice command"* is adopted in the other case ($\neg C_{11}$).
3. *Actors dependency*: a certain context may be required for an actor to attain a goal/get a task executed by delegating it to another actor. For example, the customer information system can satisfy the goal *"deliver a sample of the product to customer by sales staff"* by delegating it to the sales staff information system, when the corresponding sales staff has the ability and time to explain sufficiently about the product to customer (C_6).

4. *Root goals*: root goals may be activated only in certain contexts. For example, to activate the goal *"promote product to customer in mall"*, the customer has to be inside the mall building and may accept getting promotion of the product (C_1).
5. *And-decomposed goal/task*: the satisfaction (execution) of a sub-goal (sub-task) in an And-decomposition might be needed only in certain contexts; i.e., some sub-goals (sub-tasks) are not always mandatory to fulfil the top-level goal (task). For example, the sub-task *"show customer current place to sales staff"* is not needed if the customer stays around and can be seen directly by the sales staff (C_{12}).
6. *Contribution to soft-goals*: 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. For example, a goal like *"establish wireless connection"* contributes differently to the softgoal *"reliable connection"* according to the distance between the customer's device and the wireless access point. A task like *"get customer confirmation by voice recognition"* contribute negatively to the softgoal "less disturbance" when there are other people around the customer and it is so quite (C_{12})

5.1 Context Analysis

Similar to goals, context may need to be analyzed. On the one hand, gaol analysis allows for a systematic way in discovering alternative set of tasks an actor may execute trying to reach a goal. On the other hand, context analysis should allow for a systematic way in discovering alternative sets of facts an actor may verify trying to judge if a context applies.

We specify context as a formula of world predicates. The EBNF of this formula is as shown in Code 1:

Code 1 The EBNF of world predicates formula

```
Formula :- World_Predicate | (Formula) | Formula AND Formula | Formula OR Formula
```

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

Definition 3 (Fact) *a world predicate F is a fact for an actor A iff F can be verified by A.*

Definition 4 (Statement) *a world predicate S is a statement for an actor A iff S can not be verified by A.*

An actor has a clear way to verify a fact. It has the ability to capture the necessary data and compute the truth value of a fact. A fact is not a subject of viewpoints. In other words, when a fact is true for an actor it will be also true for others. For example, a world predicate such as *"customer recently bought the product from the mall"* is a fact. To verify this fact, the Customer IS system actor can check the purchase history of the customer since a number x of days ago. A world predicate such as *"two products, p_1 and p_2, are usually sold together"* is also a fact. The system can check the sales record of all customers and check if the two products p_1 and p_2 are often sold together. *"product is not in the shopping cart of the customer"* is a world predicate that is a fact the system can verify using an RFID reader in the cart and check if the product (identified by its RFID tag) is in the cart of the customer.

Some world predicates are not verifiable by an actor. We call such predicates *statements*. A world predicate can not be verified by an actor for reasons such as:

- lack of information: an actor may be unable to verify a world predicate because of the inability to capture the information necessary to verify it. For example, "customer does not know about a new product" is a statement from the perspective of an actor such as the sales staff in a shopping mall. The staff can not obtain all the information needed to verify this statement. The staff can not monitor if a customer has read about the product somewhere on the web or has been told about it by a friend.
- abstract nature: some world predicates are abstract by nature and do not have clear criteria to be evaluated against. For example "customer is interested in a product" is a world predicate that an actor, such as a sales staff, has no precise way to judge if it holds and be certain of the judgement. It is a concept that refers to a customer's mood that there is no way to verify it by an actor rather than the visitor himself.

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. For example, to decide if to promote a product via offering a discount, the system (Customer IS system) has to judge if the context C_3 applies. This includes deciding the truth of the world predicate *wp*="customer is interested in the product". Such world predicate is a statement that the system can not verify. However, this statement can be refined into a formula of facts and other statements. For example, the refinement could consider the behavior of the customer in the mall and his purchase history. If customer is in the product area for long time examining it or if he is coming to the product area often and touch the product, then the system may judge that *wp* holds, i.e., judge that the customer tends to be interested in the product. We call the relation between such a formula of word predicates and a refined statement *Support*, and we define it as following:

Definition 5 (Support) *a statement S is supported by a formula of world predicates φ iff φ provides evidence in support of S.*

In an iterative way, a statement could be ultimately refined to a formula of facts that supports it. That is to say, the relation *support* is transitive. If a formula φ_1 supports a statement S_1 and $S_1 \wedge \varphi_2$ supports S_2, then $\varphi_1 \wedge \varphi_2$ supports S_2. However, refining a statement to a formula of facts is not always possible. We may have statements that could be unrefinable to facts. For example, "visitor did not visit any other shopping malls during the last month" is a world predicate that can not be verified by a sales staff for the lack of information. Moreover, the staff would not be able to find a formula of facts that he can verify to support such a statement. In our contextual goal model, we allow only for contexts that are specified by means of facts and/or statements that are supported by facts. We call the kind of statements and contexts that we deal with as *monitorable statements* and *monitorable contexts* and we define them as follows:

Definition 6 (Monitorable Statements) *a statement S is monitorable iff there exists a formula of facts φ that supports S.*

Definition 7 (Monitorable context) *a context C is monitorable iff C can be specified by a formula of facts and monitorable statements*

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.

Context analysis aims to discover if a context is monitorable and to find the formula of facts that specifies it. Context analysis starts with specifying a world predicate formula that represents a context. This formula may contain both facts and statements. For example, taking the context C_1 of the contextual goal model shown in Fig 2, this context can be specified as a formula of world predicates $C_1 = wp_1 \wedge wp_2$ where wp_1="customer is inside the mall building" and wp_2= "customer may accept getting promotion of the product". Obviously, the world predicate wp_1 is a fact that the system can verify on the base of obtainable data (position of the customer can be obtained through a positioning system) while wp_2 is a statement and we need to find if it is refinable into a formula of facts.

To see if a context is monitorable, the statements in the formula that specifies that context need to be refined into formulae of facts that support them. A statement can be analyzed iteratively to ultimately discover a formula of facts that an actor can visualize in the world and that gives evidence in support of the analyzed statement. In Fig. 3, we analyze the context C_1. In this figure, *statements* are represented as shadowed rectangles and *facts* as parallelograms. The relation *support* is represented as curved filled-in arrow, and the *and*, *or*, *implication* logical operators are represented as black triangles, white triangles, filled-in arrows, respectively.

As we mentioned earlier, we consider the relation *support* as a transitive relation. For example, as shown in Fig. 3, the formula $w_1 \wedge w_2 \wedge w_3$ supports the statement wp_2, the formula $f_5 \wedge f_6$ supports the statement w_3, then the formula $w_1 \wedge w_2 \wedge f_5 \wedge f_6$ supports the statement wp_2. Consequently, a statement may be refined iteratively to reach the level of facts. In the same figure, we show the

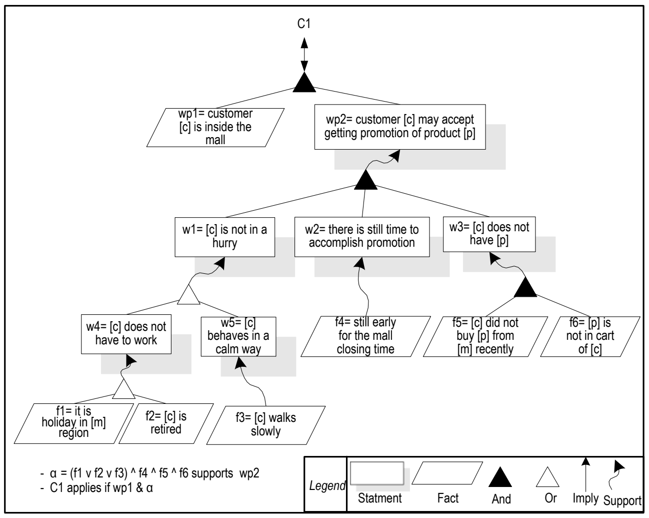

C1
wp1= customer [c] is inside the mall
wp2= customer [c] may accept getting promotion of product [p]
w1= [c] is not in a hurry
w2= there is still time to accomplish promotion
w3= [c] does not have [p]
w4= [c] does not have to work
w5= [c] behaves in a calm way
f4= still early for the mall closing time
f5= [c] did not buy [p] from [m] recently
f6= [p] is not in cart of [c]
f1= it is holiday in [m] region
f2= [c] is retired
f3= [c] walks slowly
- α = (f1 v f2 v f3) ^ f4 ^ f5 ^ f6 supports wp2
- C1 applies if wp1 & α
Legend
Statment
Fact
And
Or
Imply
Support

Fig. 3. The context analysis for C_1

formula of facts that supports the statement wp_2. The Customer IS system actor can verify this formula to judge if wp_2 applies.

Analyzing context allows us to discover what data an actor has to collect of the world. The analysis allows us to identify the facts that an actor has to verify. These facts are verifiable on the base of data an actor can collect of the world. For example, taking the facts of the context analysis shown of Fig. 3, we could develop a data conceptual model, shown in Fig. 4, that the promotion system has to implement and maintain in order to verify facts, judge if the analyzed contexts apply, and take decisions at the corresponding variation point of the goal model.

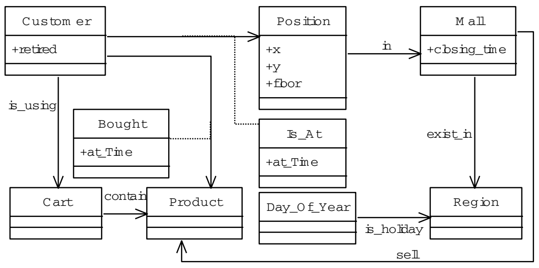


Fig. 4. The conceptual model of data needed to verify C_1 facts

To illustrate our proposed constructs to analyze context, we show more examples of context analysis and their corresponding data models In Fig. 5.

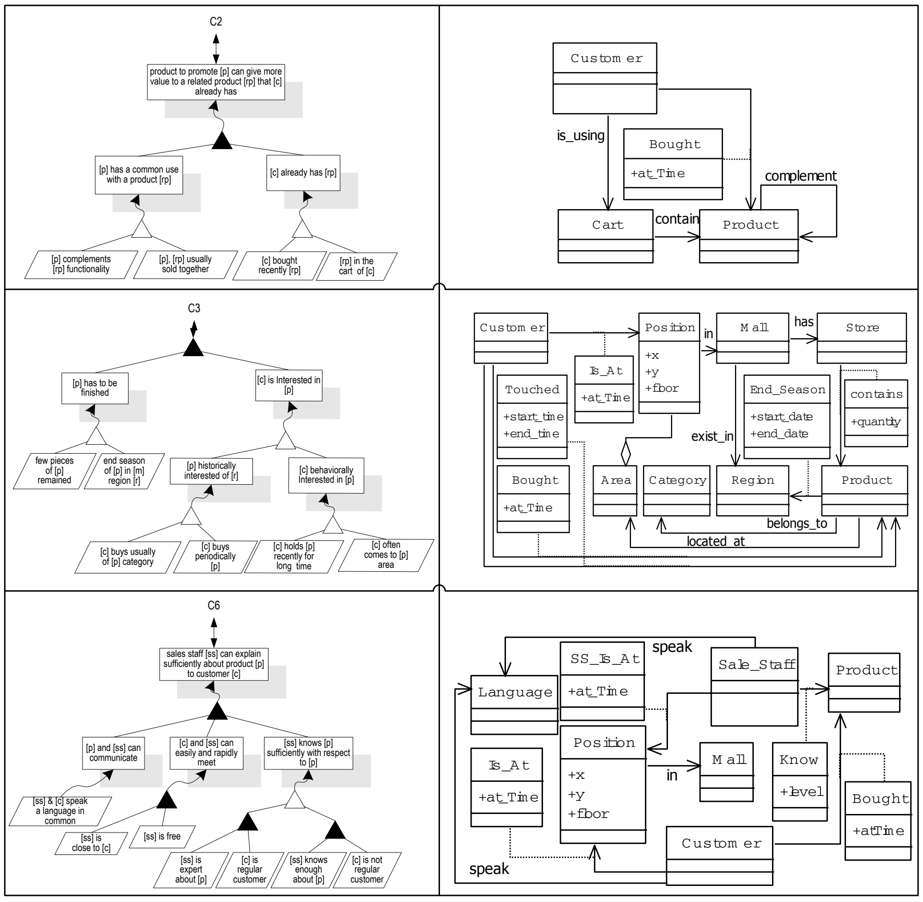

C2
product to promote [p] can give more value to a related product [rp] that [c] already has
[p] has a common use with a product [rp]
[c] already has [rp]
[p] complements [rp] functionality
[p], [rp] usually sold together
[c] bought recently [rp]
[rp] in the cart of [c]
Customer
Bought
+at_Time
is_using
Cart
contain
Product
complement
C3
[p] has to be finished
[c] is Interested in [p]
few pieces of [p] remained
end season of [p] in [m] region [r]
[p] historically interested of [r]
[c] behaviorally Interested in [p]
[c] buys usually of [p] category
[c] buys periodically [p]
[c] holds [p] recently for long time
[c] often comes to [p] area
Customer
Position
+x
+y
+floor
in
Mall
has
Store
Is_At
+at_Time
Touched
+start_time
+end_time
End_Season
+start_date
+end_date
contains
+quantity
exist_in
Bought
+at_Time
Area
Category
Region
Product
belongs_to
located_at
C6
sales staff [ss] can explain sufficiently about product [p] to customer [c]
[p] and [ss] can communicate
[c] and [ss] can easily and rapidly meet
[ss] knows [p] sufficiently with respect to [p]
[ss] & [c] speak a language in common
[ss] is close to [c]
[ss] is free
[ss] is expert about [p]
[c] is regular customer
[ss] knows enough about [p]
[c] is not regular customer
speak
SS_Is_At
+at_Time
Sale_Staff
Product
Language
Position
+x
+y
+floor
Is_At
+at_Time
Mall
in
Know
+level
Bought
+atTime
Customer
speak

Fig. 5. Examples of context analysis and elicited data models.

The analogy between goal and context analysis is shown in Fig. 6. Goal analysis provides constructs to hierarchically analyze goals and discover alternative sets of tasks that can be used to achieve such goals. Context analysis provides constructs to hierarchically analyze contexts and discover alternative sets of facts the system has to verify to judge if a certain context holds.

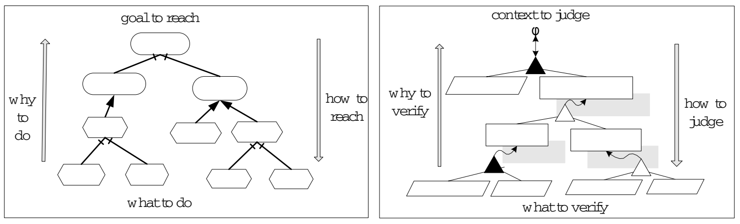


Fig. 6. The analogy between Goal and Context Analysis

6 Discussion

The decomposition of the system into the functional part captured by goal model and the monitoring part that is captured by context analysis, and the association between variation points of goal model and the analyzed context allow for a systematic contextualization of the system at the goal level of abstraction. Contextualization can be done at two different times:

- *contextualization at deployment time*: when deploying the system to one specific environment, and when we know a priori some contexts that never change in that environment, we can consequently exclude the support of the goal model variants that their contexts never apply at that environment. For example, if the software is going to be deployed in a mall where the noise level is always high due to the nature of that mall (for instance, the mall is located in an open area, or the mall sells products of a specific nature), the context $C7$ will never, or rarely, be satisfied, and therefore the deployed software for that mall can exclude the functionality of voice recognition as a way of interaction with customers.
- *contextualization at runtime*: some other contexts are highly variable and should be monitored at runtime to know what variant to adopt. Consequently, the software has to monitor context, by collecting data of its environment and validating the formulae of facts that specify contexts assigned to the variation points, and then adopt a suitable goal model variant. For example, the distance between customer and the self-service machine is a context which has always different values, and whether the software has to guide the customer to the machine using the alternative *"trace and instruct customer to machine"*, or *"show path to machine on the mall map"* depends on the actual value of this variable distance.

The hierarchical context analysis has the potential to make a context (i) more understandable for the stakeholders, (ii) easily modifiable as it is not given as one monolithic block, and (iii) more reusable as parts of the statement analysis hierarchy can be also used for other variation points or other stakeholders context specifications. Specifying for each fact the related fragments of the data conceptual model is useful for purpose of tracking. For example, if for some reason, a group of stakeholders decided to drop, to alter, or to reuse one alternative, statement, or fact, we still can track which fragments in the conceptual data model could be influenced.

Example 1. a certain mall administration could decide that to promote by offering discount, it is not required that *"few pieces of the product left"*, and it is, instead, required that the fact *"[p] sales < 80 percent of the average sales of [p] in this period last years"* is true. In this new context specification ($C3'$), one part of $C3$ is deleted, one is reused, and another is added as shown in Fig. 7a. Removing the fact *"few pieces of product[p] remained"*, leads to remove the corresponding data conceptual model fragments (the class *store*, and the association class *contain*). To verify the new fact, the system needs the sales records that are already represented in the data model fragment $MC3$. Therefore, the new data conceptual model for $C3'$ will be like shown in Fig. 7b.

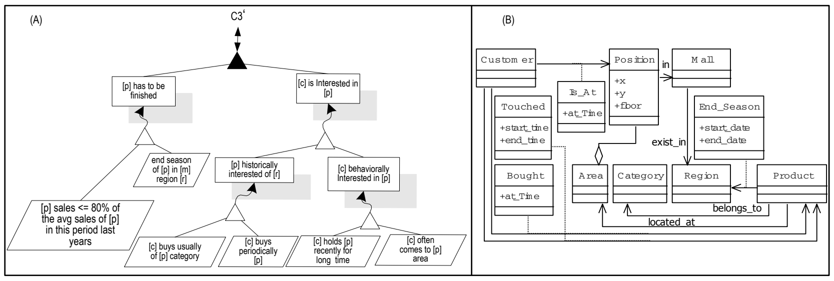

(A)
C3'
[p] has to be finished
[c] is Interested in [p]
end season of [p] in [m] region [r]
[p] historically interested of [r]
[c] behaviorally Interested in [p]
[p] sales <= 80% of the avg sales of [p] in this period last years
[c] buys usually of [p] category
[c] buys periodically [p]
[c] holds [p] recently for long time
[c] often comes to [p] area
(B)
Customer
Position
+x
+y
+floor
in
Mall
Is_At
+at_Time
Touched
+start_time
+end_time
End_Season
+start_date
+end_date
exist_in
Bought
+at_Time
Area
Category
Region
Product
belongs_to
located_at

Fig. 7. A modified context C_3' and its data model.

References

1. Mark Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.
2. A. Finkelstein and A. Savigni. A framework for requirements engineering for context-aware services. In *Proceedings of 1st International Workshop From Software Requirements to Architectures (STRAW 01)*, 2001.
3. M. Jackson. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.
4. E. Yu and J. Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, pages 15–22, 1998.
5. E.S.K. Yu. Modelling strategic relationships for process reengineering. *Ph.D. Thesis, University of Toronto*, 1995.
6. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

7. Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.
8. John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Commun. ACM*, 42(1):31–37, 1999.
9. S. Fickas and M.S. Feather. Requirements monitoring in dynamic environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, page 140. IEEE Computer Society Washington, DC, USA, 1995.
10. D. Sykes, W. Heaven, J. Magee, and J. Kramer. From goals to components: a combined approach to self-management. In *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*, pages 1–8. ACM New York, NY, USA, 2008.
11. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. Location-based software modeling and analysis: Tropos-based approach. In Qing Li, Stefano Spaccapietra, Eric S. K. Yu, and Antoni Olivé, editors, *ER*, volume 5231 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2008.
12. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. Location-based variability for mobile information systems. In Zohra Bellahsene and Michel Léonard, editors, *CAiSE*, volume 5074 of *Lecture Notes in Computer Science*, pages 575–578. Springer, 2008.
13. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. Goal-based self-contextualization. In *In the Forum of the 21st International Conference on Advanced Information Systems (CAiSE 09 - Forum)*, volume Vol-453, pages 37–42. CEUR-WS, 2009.
14. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A goal modeling framework for self-contextualizable software. In *In the Proceedings of the 14th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD09)*, LNBIP 29-0326, pages 326–338. Springer, 2009.
15. P. Brezillon. Context in Artificial Intelligence: I. A survey of the literature. *Computers and artificial intelligence*, 18:321–340, 1999.
16. A.K. Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.
17. J. Krogstie, K. Lyytinen, A.L. Opdahl, B. Pernici, K. Siau, and K. Smolander. Research areas and challenges for mobile information systems. *International Journal of Mobile Communications*, 2(3):220–234, 2004.
18. X. Shen, B. Tan, and C.X. Zhai. Context-sensitive information retrieval using implicit feedback. page 50, 2005.
19. A. Van Lamsweerde et al. Goal-oriented requirements engineering: A guided tour. *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, 249:263, 2001.
20. I. Jureta, J. Mylopoulos, and S. Faulkner. Revisiting the core ontology and problem in requirements engineering. pages 71–80, 2008.